공학석사 학위논문

Object Classification and Shape Retrieval Based on 3D Object Surface

3차원 물체 표면 기반의

물체 인식 및 형상 복원 기법

2017년 2월

서울대학교 대학원

전기 · 정보공학부

김 진 원

Abstract

This thesis proposes a machine learning based object classification method to recognize the object category when a single-view point cloud of an object surface is given. In addition, given multiple single-view surface point clouds capturing the shape of the object, a registration method that models the complete object shape is presented.

Existing methods for object classification that use point clouds mostly depend on hand-crafted features. However, it is disadvantageous to use hand-crafted features in two aspects. First, they cannot fully utilize the raw data. Second, they lack extensibility in that they often require other types of feature such as surface normal. In this thesis, in order to circumvent these weaknesses, automatically learned features from voxelized grids of surface point clouds are used for classification. To this end, a deep learning architecture with Convolutional Neural Network (CNN) as a building block is proposed. However, most deep learning architectures suffer from the overfitting problem where the model fits even the stochastic noise in the training data. To reduce overfitting of the deep architecture, two types of non-random weight initialization methods are adopted. One is the initialization with the encoding stacks of a pre-trained de-noising Convolutional Auto-Encoder (DCAE) and the other is the initialization with the cluster centers learned by k-means clustering on a set of input data patches. Quantitative analysis shows that the suggested initialization methods reduce overfitting to a certain degree. The neural network is further designed to make predictions on the

orientations of the surfaces as well as the class labels. It is shown analytically and quantitatively that this additional task improves the classification result.

For the modeling of multiple single-view point clouds, a registration method that utilizes the Signature of Histogram of Orientation (SHOT) descriptors for correspondence association and Random Sample Consensus (RANSAC) algorithm for rigid transformation estimation is suggested. It is shown that by using the proposed RANSAC method to estimate the rigid transformations between successive point clouds and fine-tuning them by the Iterative Closest Point (ICP) algorithm, reliable object shape retrieval can be performed.

Keywords : Object classification, Convolutional neural network, Unsupervised pre-training, Shape retrieval, RANSAC

Student Number : 2015-20909

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the progressive development of sensors and micro-processors, a robot is now deviating from an automated machine that repeatedly does a series of fixed routines and becoming an intelligent one that can achieve human-level performance in fields where only humans have been involved so far. One of the fundamental requirements for this kind of intelligent robot is to perceive its own environment. For the past decade, the most noticeable development in robotics was Simultaneous localization and mapping (SLAM) where a robot builds a map of its environment and at the same time, estimates its location in this map. The importance of SLAM has been emphasized in that if a robot implements SLAM in a perfect manner, it implies that the robot can navigate autonomously. For a robot to be truly autonomous, however, it has to recognize specific objects in the built map and be able to interact with them. In this aspect, the ability to detect objects and classify them into proper categories is essential for an autonomous robot.

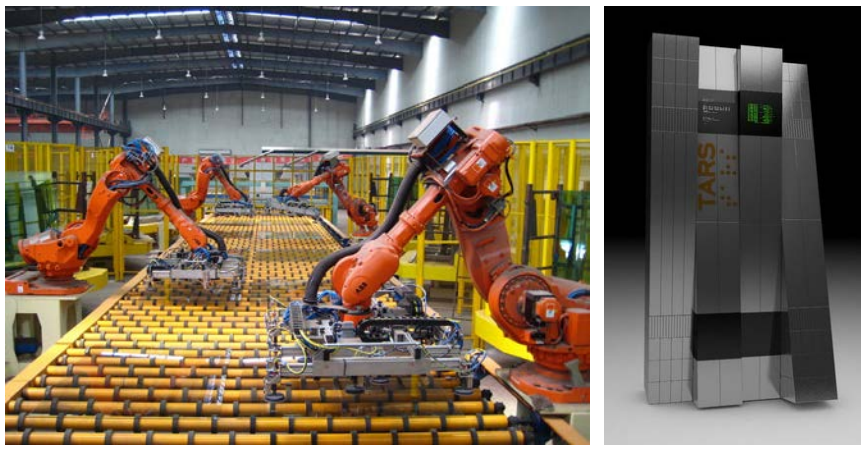


Figure 1.1: Examples of a fixed automation (left) and an intelligent robot (TARS from movie Interstellar) (right).

To detect and classify objects in a robot's neighborhood, it must first be able to see its surroundings. Two most widely used data types for a robot's 3-D vision are the RGB-D image and the point cloud. RGB-D images have regular sizes and pixels, where each pixel has four scalar values – red, green, blue and depth. The depth channel allows the image to hold 3D spatial information. However, a depth value rests on the infrared patterns projected by the camera which is unreliable when the surrounding environment is lit by natural light rays. This makes RGB-D cameras fall out of use in outdoor scenarios. On the other hand, point clouds can be collected from outdoors in good quality as well as indoors because they are generated from Light Detecting And Ranging (LiDAR) sensors. Thus, detection and classification methods that require only point cloud data should be developed in order for a robot to carry out various missions both indoors and outdoors.

The first part of this thesis focuses on the classification method that can identify an unknown object's category given only a single-view surface point cloud of the object. For this task, it is assumed that the robot can segment the object's

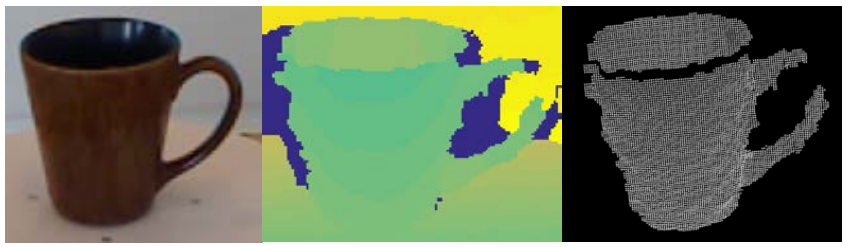


Figure 1.2: Examples of an RGB image (left), a depth image (middle) and a point cloud (right) of an object.

surface point clouds from the point cloud of the environment. This can be done by removing the ground part first and then segmenting individual object point clouds.

In the past, weak classifiers trained on hand-crafted features of point clouds were mainly used for classification. However, engineered features, in many cases, do not fully utilize the intrinsic information contained in the raw data. Many of them also depend on other engineered features such as surface normal so that they lack robustness and extensibility. Therefore, learned features which in general represent the raw data more faithfully are desired. A simple but powerful convolutional neural network (CNN) [1] based architecture is presented to this end. The proposed neural network, once trained on large training data, becomes an efficient feature extractor and classifier.

However, neural networks do have problems. They are typically very complex models with many parameters, so they tend to fit training data very faithfully while fail in making good predictions on new data. This is the so-called overfitting of neural networks. The performance gap revealed by the trained data and the unseen data must be narrowed because a robot carrying out a mission mostly encounters unseen objects and classification performance on these newly detected objects is all that matters. Dropout [2] is one common choice for reducing overfitting in neural

networks. However, in this thesis, another kind of measures so called unsupervised pre-training [3] is chosen in addition to dropout to further reduce overfitting.

The second part of this thesis concerns retrieving the full 3-D volumetric shape of an object. A 3-D volumetric representation of an object contains richer visual information than a single object surface and can be used as a reliable map feature in the SLAM process. A 3-D volumetric shape can also be provided to users in virtual reality (VR) and telepresence applications. In this respect, a shape retrieval method that can register multiple single-view point clouds into a shape-recovered volumetric point cloud is suggested. The most utilized point cloud registration methods are the Iterative Closest Point (ICP) algorithm [4] and the variants thereof. However, ICP is prone to false convergence when the initial rigid transformation, especially the rotation, between two point clouds is large. In this thesis, a RANSAC [5] based method is given that estimates a reliable rigid transformation between two point cloud surfaces. The proposed method can compute rigid transformation between two unaligned point clouds irrespective of the initial transformation if only a portion of overlap exists.

1.2 Contributions

The main contributions of this paper are outlined here.

- A deep neural network architecture adopting several convolutional layers is proposed in order to classify objects based on automatically learned features. The network is trainable on voxel grid representations obtained from single-view surface point clouds of

objects. Equipped with the trained weights, object classification can be performed in real-time with the help of the general-purpose graphical processing units (GP-GPU).

- To reduce overfitting and lessen the generalization error, filter weights of the convolutional neural network are initialized non-randomly in unsupervised way. This thesis reports classification performances of three different CNNs with the same structure, but with different initializations of weights. The first is a CNN whose filter weights are initialized with random numbers distributed according to a scaled uniform distribution. The second CNN is initialized with the encoder stacks of a pre-trained de-noising convolutional auto-encoder (DCAE). The third CNN is initialized with the k-means cluster centers of small input patches of the training data.
- It is shown that the classification performance can be improved by making the CNN optimize two separate loss functions at once, one for the object class label and the other for the surface orientation label. This improvement is expectable since surfaces of an object look quite different depending on the orientations of them.
- A fast and effective RANSAC algorithm that aligns two point clouds before ICP refinement is presented. The whole modeling procedure is shown to significantly outperform ICP modeling with no information on the initial transformation.

1.3 Organization

This thesis is organized as follows. Related works in the literature are reviewed in Chapter 2. In Chapter 3, preliminary knowledge on deep learning and convolutional neural network (CNN) is described. Underfitting and overfitting problems are also explained. Then, the proposed CNN structure and pre-training methods are given. Chapter 4 describes the background knowledge regarding point cloud registration and Random Sample Consensus (RANSAC) algorithm. Detailed explanation of the proposed methods then follows. Chapter 5 reports evaluation results. Classification results on a public dataset containing single-view surface point clouds of objects in multiple classes are shown. Shape retrieval results including modeling time as well as transformation accuracy are given. Finally, this thesis is concluded in Chapter 6.

Chapter 2

Related Work

2.1 Point cloud segmentation

In typical robotic scenarios where a robot handles point cloud data, point cloud segmentation is an important preprocessing step for higher-level applications such as semantic environment understanding and object detection. In [6], real time segmentation of streaming 3D point cloud data is presented. It computes normal vectors of incoming points from their local neighborhood and clusters the new points by assessing their Euclidean and angular distances to previously clustered points. This method is useful in scenarios where sweeping range sensors are used. In another research [7], point clouds are segmented by growing ellipsoidal regions via minimum spanning-tree. Merge criterion for region growing is suggested and parameters are tuned to minimize Akaike's Information Criterion (AIC). In the work of [8], object segmentation is done via local convexity. Range scans are transformed into graphs with the nodes being the points and edges connecting the points. Local convexity property is defined

between locally pairing points and used in conjunction with vertical structure criterion to determine the final segmentation criterion. They start segmentation with random graph nodes and remove segmented nodes until termination.

2.2 Hand-crafted features for object classification

Object classification, sometimes called object category recognition, can be categorized by two types. One is to classify objects based on engineered features and the other based on learned features. Engineered features (hand-crafted features) are commonly referred to as descriptors in the sense that they describe either the local neighborhood of points or the whole point cloud globally. Most descriptor-based classification method is subdivided into two stages. First, they extract keypoints from surface point cloud and define descriptors on the extracted keypoints by summarizing local neighborhood information. If a global descriptor is used, keypoint extraction procedure is omitted and a single descriptor describes the entire point cloud. Second, they train a weak classifier – usually linear Support Vector Machine (SVM) - with the extracted features.

Spin Images (SI) [9] has been a popular local shape descriptors used for object recognition and classification. A spin image can be computed with point clouds whose points are oriented, i.e. attached with normal vectors. To compute SI, point clouds are transformed into meshes. An arbitrary oriented point is chosen and a tangent plane whose normal direction is aligned with the normal vector of that point is attached. This plane defines the local reference frame (α, β) along which all the other mesh vertices can have radial and elevation distances. Projected

vertices are then discretized to form the spin image of that oriented point.

Following the success of Histogram of Oriented Gradient (HoG) [10] and Scale Invariant Feature Transform (SIFT) [11] in 2D image processing, Signature of Histogram of OrienTation (SHOT) descriptor is proposed for 3D point cloud in [12]. In [12], a robust local reference frame computed by eigenvalue decomposition of a distance-weighted covariance matrix is attached to a keypoint. The neighborhood region of the keypoint is divided into eight divisions along the azimuth angle, two divisions along the elevation angle and two divisions along the radial distance to form 32 sub-regions. In every division, a histogram representing the distribution of inner products between point normals belonging to the region and the normal of the keypoint is assigned. This 32 histograms distributed around the local reference frame finally form a SHOT descriptor on the keypoint.

Fast Point Feature Histogram (FPFH) [13] is another local descriptor. Given two oriented points p and q with normals n_p, n_q, respectively, a fixed reference frame (u, v, w) on this point pair is defined as below.

$$u = n_p \tag{2.1}$$

$$v = u \times \frac{(q-p)}{\|q-p\|_2} \tag{2.2}$$

$$w = u \times v \tag{2.3}$$

The $\times$ operator is the outer product of two vectors. With this reference frame, a set of angular features, (α, ϕ, θ) can be computed as

$$\alpha = cos^{-1}(v \cdot n_q) \tag{2.4}$$

$$\phi = cos^{-1}(u \cdot \frac{(q-p)}{\|q-p\|_2}) \tag{2.5}$$

$$\theta = tan^{-1}(w \cdot n_q, u \cdot n_q) \tag{2.6}$$

The $\cdot$ operator is the inner product of two vectors. A keypoint can be paired with k-nearest neighbors resulting in k sets of angular features. Three kinds of angles in (2.4)-(2.6) are distributed in 11-bin histogram respectively and the concatenation of three histograms yields one 33-bin histogram. This is called the Single Point Feature Histogram (SPFH). A weighted sum of SPFHs of k-nearest neighbor points of a keypoint finally yields the Fast Point Feature Histogram (FPFH) of that keypoint.

The local descriptors mentioned so far have shortcomings that are mentioned before. All of them need other engineered feature, i.e. the normal vector for each point and thus robust normal estimation plays a critical role for them. Furthermore, to avoid excessive computations, these descriptors should be defined on a few repeatable keypoints that also have to be engineered.

While local descriptors describe local neighborhoods of keypoints, global descriptors describe the entire point cloud. Point Feature Histogram (PFH) [14] is a global version of FPFH. Unlike FPFH, PFH computes (α, ϕ, θ) angles between all possible pairs of points in the cloud. Thus, the number of angle sets generated by PFH grows quadratically with the total number of points and is usually very large especially for high-resolution point cloud. The way these angle sets are binned into histogram is also different from FPFH. Each angle is discretized independently into five bins and the total number of bins in PFH amounts to 125

($=5^3$). PFH also requires robust normal estimation and the computation is usually excessive.

In [15], Ensemble of Shape Function (ESF) descriptor is described. It is an ensemble of ten 64-bin histograms of shape functions describing characteristic properties of the point cloud cluster. Ten histograms comprise sub-histograms of three angle shapes, three area shapes, three distance shapes and one ratio of line distance shape. Distances, areas and angles are computed from sets of randomly selected points and their natures are classified into three states – on, out and mixed – depending on whether the end points rest on the surface or not. This categorization thus results in three different histograms for angle, area and shape respectively. No preprocessing and tuning of parameters are required. However, due to the random sampling of points, ESF is not uniquely determined for a given point cloud and has a relatively low dimension of 640.

Except SI, descriptors explained so far are SHOT, FPFH, PFH and ESF descriptors. They showed the highest classification performances [16] on the dataset [17]. Therefore, these descriptors are chosen for comparison to the proposed classification method.

2.3 Learned features for object classification

Many researches on learned features for object recognition and classification with RGB-D images have been under way since the late 2000s. This is because reliable and cheap RGB-D cameras like Microsoft Kinect and Asus XtionPro have been made available to the public since then. In the work of Blum et.al. [18], a

learned feature called the convolutional k-means descriptor is described. They compute SURF keypoints in the training images and extract small patches around these keypoints. After that, they cluster the unlabeled image patches by simply running k-means. With the obtained k centroids, a feature mapping is learned that maps an input image into a histogram of indicator vectors indicating the closest centroids of image patches. They soft-bin the histogram, rather than hard binning by considering not just the closest centroid but half of the centroids in the order of distances from the image patch. That is, instead of using an impulse indicator vector, they use several impulses distributed in half the histogram region as an indicator vector. This smooths the feature vector so that the feature vector is almost continuous-valued. Even though they propose a feature learning framework, they also depend on an engineered feature for the extraction of keypoints thereby making their feature not a fully learned one. In [19], the authors propose to learn a dictionary via which an input vector is transformed into a sparse code. They alternately compute the sparse code matrix using Orthogonal Matching Pursuit (OMP) and update the codebook by Singular Value Decomposition (SVD). They use Hierarchical Matching Pursuit (HMP) that sequentially builds a feature hierarchy. They extract patch-level features from sparse codes of image pixels and generate features of the whole image by applying sparse coding and max-pooling to the patch-level features. Their final feature vector has a very high dimension amounting to 188,300 which is very descriptive but unsuitable for real-time application. In another study [20], a recursive neural network on top of single CNN layer for RGB-D images is proposed. The authors in [20] in some way follow the work of Coates [21] and Blum [18] by initializing CNN filter weights with the centroids of k-means

clusters of training image patches. The feature maps generated from CNN layer are divided spatially into sub-regions called blocks. Each block is passed to fully-connected layers and the outputs from all fully-connected layers are concatenated and passed to the final softmax classifier.

All the methods mentioned so far aim at 3D object classification but they use RGB-D them and treat the images as 2D arrays with four channels.

There are feature learning methods on general 3D shape data [22] [23] [24]. These methods basically adopt CNN layer at the bottom, introducing the need to voxelize the input data. Computer Aided Design (CAD) models and point cloud models generated by LiDAR can be used. In [22], a simple neural network with two CNN layers and two fully-connected layers is suggested. The authors voxelize the input data with three different occupancy models and augment the training data with the rotated versions of themselves. They achieved relatively good performance on CAD models [23] and LiDAR [25] data. In [24], the authors design multi-view CNN on top of image based CNN to achieve the state-of-the-art classification performance. They render 2D images from a volumetric CAD model [23] for classification with only these rendered views. Then, they pass the rendered 2D images into the image based CNN pre-trained on ImageNet and aggregate them into feature maps by pooling. The pooled features are passed into the second CNN layer to finally yield output feature maps for class detection. They proved that their network also work with coarse 2D sketches of CAD models.

The mentioned CNN based methods targeting for 3D CAD model, especially the ModelNet data, use either volumetric objects or rendered views of originally volumetric object as inputs. However, in this thesis, surface point clouds collected

directly from the objects are voxelized and given as the input to CNN, which is a more realistic approach in robotic scenarios.

2.4 3D object shape retrieval

Object shape retrieval using point clouds of an object's surfaces requires robust registration of these surface clouds. One of the most typical approaches to point cloud registration is the Iterative Closest Point (ICP) algorithm proposed in [4]. ICP is a naïve way of registering two point clouds based on point distances. The idea is to find pairs of corresponding points between two clouds first and then minimize the sum of squared distances between corresponding points with respect to transformation parameters. This process is iterated until convergence. Correspondences are obtained with Euclidean distance criterion, i.e. closest points are considered as corresponding points. ICP is relatively fast when nearest neighbor search of closest point is boosted by approximate kd-tree [26]. However, it is very vulnerable to a large relative rotation between two point clouds in which case the correspondences are unlikely to be correct. Other methods including variants of ICP [27] [28] and Normal Distributions Transform (NDT) [29] can also be used to register two point clouds. These scan matching methods do not usually preprocess the point cloud data. They are commonly chosen for the registration of successive LiDR scan data that differ slightly, e.g. dense map scan data. For the retrieval of object shape, it is more general to preprocess the point clouds and extract shape descriptors to establish correspondences. Global descriptors that describe a point cloud entirely have limitations in that if they do

not sufficiently contain overlapping regions, it is hard to get a good matching. On the other hand, if only three correct pairs of local descriptors extracted from overlapping regions exist, they can determine registration parameters. Often, sample consensus based algorithms are used for the determination of registration parameters.

Many local descriptors were proposed in the literature [9] [12] [13] [30] [31], including those explained in the preceding chapter 2.2. As a matter of fact, local shape descriptors are applicable to both classification and shape retrieval. Most shape retrieval methods using local descriptors follow the coarse-to-fine strategy where two point clouds are coarsely aligned with the transformation determined by correspondences of local descriptors and then fine-tuned by ICP. This is because ICP shows very good performance and can be efficiently computed if two point clouds are almost aligned [32].

Chapter 3

3D CNN based object classification

3.1 Preliminaries

3.1.1 Deep learning and convolutional neural network

Deep learning is a class of machine learning technique that attempts to learn an intrinsic structure of the input data at multiple levels of abstraction [33]. In general, learning techniques that adopt deep neural networks are collectively called as deep learning. A deep neural network is generally composed of multiple layers and each layer consists of multiple units of parameters. Layers other than the first and the last ones are called the hidden layers of the neural network. Each hidden layer learns a hierarchical representation of the data as the learning proceeds. The first and the last layers are called the input layer and the output layer respectively. The learning protocol for which not just deep learning but other machine learning techniques have been studied the most is the supervised learning. In a general supervised learning, training data come as input and label pairs $(\mathbf{x_n}, y_n)$, $n = 1, \cdots, N$. N is the number of training data. It is called supervised in the sense that the network knows what the correct output (label) is for a given input $\mathbf{x_n}$. The goal

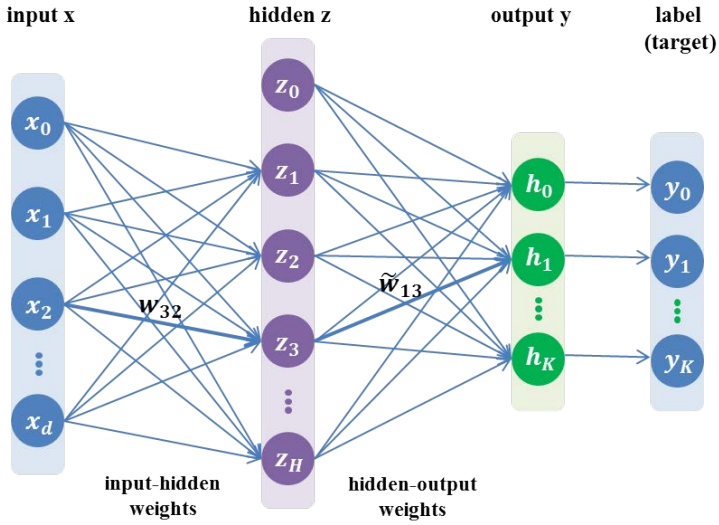


Figure 3.1: A multilayer perceptron (MLP) with one hidden layer. Input, hidden and output vectors are denoted as x, z and y respectively and their components are expressed by using the subscripts.

of the supervised learning is to learn the highly nonlinear function that maps an input to its correct label by learning the training data. (A more accurate statement would be that the supervised learning aims to learn the highly nonlinear posterior probability distribution as a decision boundary.) If successfully trained, the neural network can be used to determine output labels for unseen (new) input data.

The simplest form of deep neural network is the multilayer perceptron (MLP) shown in Figure 3.1. For simplicity, an MLP with only one hidden layer is depicted. Input vector **x**, hidden variable vector **z** and the output vector **h** are denoted as in (3.1) – (3.3).

$$\mathbf{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d \tag{3.1}$$

$$\mathbf{z} = (z_1, \cdots, z_H) \in \mathbb{R}^H \tag{3.2}$$

$$\mathbf{h} = (h_1, \cdots, h_K) \in \mathbb{R}^K \tag{3.3}$$

Each component of the vector is called the attribute or the feature of that vector. Bias terms for each vector (**x**, **z** and **h**) are written as zero-indexed components x_0 (or z_0, h_0) even though they are not actual components of the vector. Synaptic weights (network parameters) of the network are written as in (3.4) and (3.5).

$$\text{input} - \text{hidden weights } \mathbf{w} = w_{ji}, \qquad 1 \le i \le d, 1 \le j \le H \tag{3.4}$$

$$\text{hidden} - \text{output weights } \widetilde{\mathbf{w}} = \widetilde{w}_{kj}, \qquad 1 \le j \le H, 1 \le k \le K \tag{3.5}$$

Training procedure for this neural network is as follows. The first input layer accepts an input vector **x**. In the second hidden layer, each hidden unit accepts the weighted sum of the input features (forward signal) and computes the activation of this signal to output the function signal z_j as in (3.6).

$$\text{hidden unit } z_j(\mathbf{x}) = \sigma(\Sigma_{i=0}^{d} w_{ji} x_i) \tag{3.6}$$

An activation function $\sigma(\cdot)$ is typically nonlinear. Sigmoid, hyperbolic tangent and rectified linear functions are commonly used as an activation function. In the last output layer, each output unit accepts the weighted sum of the function signals of hidden units and computes the activation of this weighted sum to yield the final output feature h_k.

$$\text{output units } h_k(\mathbf{x}) = \tilde{\sigma}(\Sigma_{i=0}^{d} w_{ji} x_i), \qquad 1 \le j \le H, 1 \le k \le K \tag{3.7}$$

Since the correct output (label) is known, the error measure between the output of

the network and the label can be defined for each $(\mathbf{x}_n, y_n)$. For example, sum of squared error in (3.8) can be used as the error measure.

$$\mathcal{E}_n = \frac{1}{2}\Sigma_{k=1}^{K} e_{k,n}^2 \tag{3.8}$$

$$e_k = h_k - y_k, \mathrm{k} = 1, \cdots, \mathrm{K} \tag{3.9}$$

For all N training data, the average error measure is

$$\mathcal{E} = \frac{1}{N}\Sigma_{n=1}^{N} \mathcal{E}_n \tag{3.10}$$

Sum of squared error is just an example. In effect, categorical cross-entropy error and many other error measures can be used. It is shown in [34] that a neural network which minimizes the sum of squared error or the categorical cross-entropy error learns posterior probabilities $\mathrm{p}(\mathrm{y}_j|\mathbf{x}), j = 1, \cdots, \mathrm{M}$ where M is the number of output values. This statement will be discussed again in Chapter 3. The learning objective is to find the set of weight vectors $(\mathbf{w}^*, \tilde{\mathbf{w}}^*)$ that minimizes the average error $\mathcal{E}$.

$$(\mathbf{w}^*, \tilde{\mathbf{w}}^*) = \underset{\mathbf{w}, \tilde{\mathbf{w}}}{\operatorname{argmin}}\, \mathcal{E} \tag{3.11}$$

The error in (3.11) is non-linear and non-convex for $\mathbf{w}, \tilde{\mathbf{w}}$ if non-linear activation functions are used in the hidden layer. That is, there exists no closed form solution to the optimization of (3.11). However, local minimum of (3.11) can be achieved

by back-propagation algorithm that iteratively minimizes the error as in (3.12).

$$\mathcal{E}(\mathbf{w}) \leftarrow \mathcal{E}(\mathbf{w}) - \eta \cdot \nabla_{\mathbf{w}} \mathcal{E}(\mathbf{w}) \tag{3.12}$$

The dependence of the error $\mathcal{E}$ on the weights is denoted only by $\mathbf{w}$ for brevity. Back-propagation algorithm updates every weight between units in the network by computing the gradients. For the MLP of Figure 3.1, weight update rules for input-hidden weights and hidden-output weights are

$$w_{ji} \leftarrow w_{ji} - \eta \cdot \delta_j \cdot x_i, \qquad 1 \le i \le d, 1 \le j \le H \tag{3.13}$$

$$\widetilde{w}_{kj} \leftarrow \widetilde{w}_{kj} - \eta \cdot \delta_k \cdot z_j, \qquad 1 \le j \le H, 1 \le k \le K \tag{3.14}$$

where δ_j and δ_k are called delta errors computed by the chain rule of derivatives.

The performance of a learning model is tested on another dataset called the test set. Data in the test set are input and label pairs that the learning model (neural network) has not been trained on. Since the learning model has never seen the test data before, its general performance can be fairly evaluated with the test data. The difference in overall mean error between the test data and the training data is defined as the generalization error in that if this error is small, then it means that the learning model generalizes well to unseen data. Also, if the generalization error is small, it can be expected that the learning model will make fair predictions on new data.

Deep learning has been successfully applied to visual data recognition and classification as well as text and speech recognition. What empower the deep

learning are large amount of training data, with which the learning algorithm tries to find out inherent structures – may it be more compressive representatives of the training data or other patterns therein. Object classification is a typical problem where deep learning is the most utilized. Since the advent of deep learning, object classification with gray-scale images [35], RGB-(D) images [18] [19] [21] and 3D Computer-Aided Design (CAD) model [22] [23] [24] have experienced drastic progress. Most of them adopt CNN as their basic network layer for its efficiency in processing visual data. Besides classification, CNN has also been successfully applied to other visual data recognition methods such as face detection [36] and human action recognition [37].

Convolution between a filter w and an input x is a weighted average operation given in (3.15) – (3.17). Filters are also termed as kernels in CNN terminology.

1D: $y(n) = x(n) * w(n) = \Sigma_k x(k) w(n-k)$ (3.15)

2D: $y(n,m) = x(n,m) * w(n,m) = \Sigma_k \Sigma_j x(k,j) w(n-k, m-j)$ (3.16)

3D:

$$y(n,m,l) = x(n,m,l) * w(n,m,l) \quad (3.17)$$
$$= \Sigma_k \Sigma_j \Sigma_i x(k,j,i) w(n-k, m-j, l-i)$$

CNN is a special form of neural network where convolution operations are implemented in each of its convolutional layer. The success of CNN in handling visual data is ascribed to three factors. The first is the local connectivity. One convolution filter has a small size compared to the size of the input and thus convolution operation for that filter is applied only to a small region in the input. This also means that CNN can control the number of its parameters (of its

convolution filters) somewhat independently of the input data size. By connecting its filters to only local region of the input data, CNN can also preserve the spatial relationship of the input data whether the input is a one-dimensional vector or a multi-dimensional tensor. Since the spatial relationship of the preceding input is preserved, the output of a convolutional layer is usually called the feature map, instead of a feature vector. The second property of CNN is the parameter sharing. Parameter sharing is to use the same convolution filter at every position of the input (or the feature map). This property also accounts for the fact that CNN can control the number of parameters independently of the input size. For example, if one convolution filter acts like a Sobel filter when applied to 2D images, then this filter can be used to detect edge features at every position of the input image – there is no point of using many other redundant Sobel filters at other positions in the input. Parameter sharing also gives CNN the equivariance property to translation so that a translation in the input is effectively reflected as a translation in the feature map. That is, if an edge is translated in the input image, its representation will also be translated in the feature map with no additional changes. The third property of CNN is the pooling (subsampling). CNN more or less has pooling layers preceded by convolutional layers that down-sample non-overlapping sub-regions of the previous feature map with some summary statistics. Pooling gives CNN an invariance property to local translation. Thus, CNN is robust to small translational variations in the input.

Likewise for an ordinary multilayer perceptron, each component in the feature map is an activation of the signal computed by convolution operation. That is, the representation of an input region x_i convolved with a filter k_j in the feature map is $\sigma(x_i * k_j + b_j)$ where b_j is a bias term for the filter k_j.

3.1.2 Underfitting and overfitting

Every machine learning algorithm suffers from two types of problem, namely, underfitting and overfitting. Underfitting means that the learning model is too simple to capture the underlying input-output relationship of training data. This can be described in terms of deterministic noise. Deterministic noise is the difference between the target function and the learning model and it signifies the part of the target function that the model cannot follow up. As a result of underfitting, the learning model poorly fits the data yielding a highly biased solution. On the other hand, overfitting is attributed to the high complexity of the learning model itself. Overfitting can be explained in terms of stochastic noise contained in the input data. If a model has too many parameters, it can even fit the high-frequency stochastic noise existing in the training data. Because the model is capable of fitting the noise well enough, the model gets tied to the training data, and predicts on new, unseen data poorly. Overfitting results in a low-bias and high-variance solution. Any type of real-world data can be viewed as noisy samples. It cannot be asserted that any data is an exactly noiseless sample (realization) of a deterministic target function because in the real world, deterministic target function is unknown most of the time. From this aspect, learning the representation of any type of dataset can be swayed by overfitting problems.

Figure 3.2 provides a pictorial explanation of overfitting and underfitting.

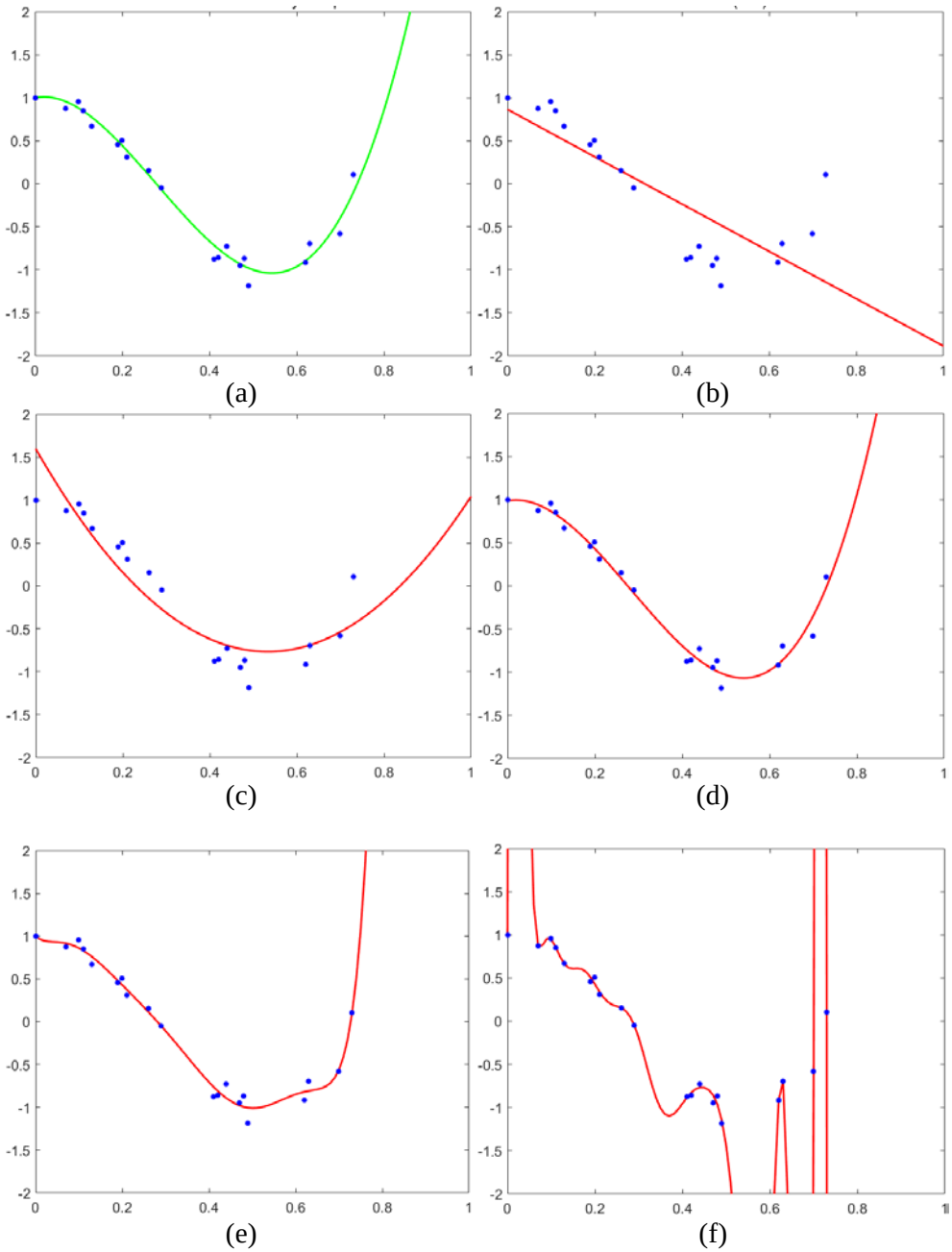


Figure 3.2: Underfitting and overfitting in the regression problem. The 1st order polynomial and the 2nd order polynomial underfit the true model whereas the 8th order polynomial and the 16th order polynomial overfit the true model. 20 sample points are drawn in each plot. (a) True model. (b) 1st order fit. (c) 2nd order fit. (d) 4th order fit. (e) 8th order fit. (f) 16th order fit.

A deterministic target function which is a fourth order polynomial is plotted in

Figure 3.2(a). Some sample points from this function with additive Gaussian noise are depicted. This true target is $f(x) = -2x^4 + 31x^3 - 25x^2 + x + 1$. It has five parameters. 20 data points are sampled from it. Since sample points are noisy, fitting these samples is a regression problem, one of the most popular learning problems. Five different models are used to learn from these samples, all of which are polynomial. Figures 3.2(b) and 3.2(c) show underfitting situations. The first order polynomial model is just a line that has only two parameters and the second order polynomial model is a quadratic function that has only three parameters. They are in a sense too simple to capture the target polynomial which is fourth order, and underfit the samples. By contrast, Figures 3.2(e) and 3.2(f) show overfitting situations. These models have nine and seventeen parameters respectively, which are excessive compared to the number of parameters of the target function. They fit the given (noisy) samples even more correctly than the target function. However, it is apparent that they will poorly fit other samples from the target that they are not trained on.

Deep learning models generally have high complexity. This implies that deep learning models are more likely to suffer from overfitting than underfitting. Overfitting is also prominent in the case that the size of the training data is small. This is consequent because small sample (training data) itself cannot represent the population sufficiently.

Many techniques to control the overfitting of deep learning models have been proposed in the literature. There are cross-validation, unsupervised pre-training and adopting dropout layer for neural networks to name a few.

3.2 Proposed method

3.2.1 Dataset and assumptions

Point cloud dataset used in this thesis is the Washington dataset [17]. This dataset contains segmented surface point clouds and RGB-D images of 300 object instances organized in 51 categories. The term category refers to the abstract class that an object belongs to while the term instance refers to a specific object. For example, ‘cap’, ‘apple’ and ‘notebook’ are categories. On the other hand, the red apple, the green apple, the apple on the table, etc. are all physically different object instances all belonging to the same category ‘apple’.

Every object instance in the Washington dataset is captured by a stationary camera placed 1 meter away while the object is made to spin on its own axis. This is equivalent to the setup that the object is placed still and the camera circles around it in a fixed distance. The camera makes one revolution around an object and captures surface point clouds at three different viewpoint heights (30°, 45°, 60°). About 200~250 single-view images per one height that amount to 600~750 single-view images per instance are generated. The exact number of view data differs from instance to instance and the number of instances is different from category to category. This situation is illustrated in Figure 3.3. Next, basic setup for classification is described. Ten categories from 51 categories are selected for classification. For each category, every instance except one is selected as training data. The remaining instance of each category is not trained on. Instead, they are used as test data to evaluate the general classification accuracy and error. Table 3.1 lists the selected categories and instances thereof. Instances are numbered by their designated numbers in the dataset.

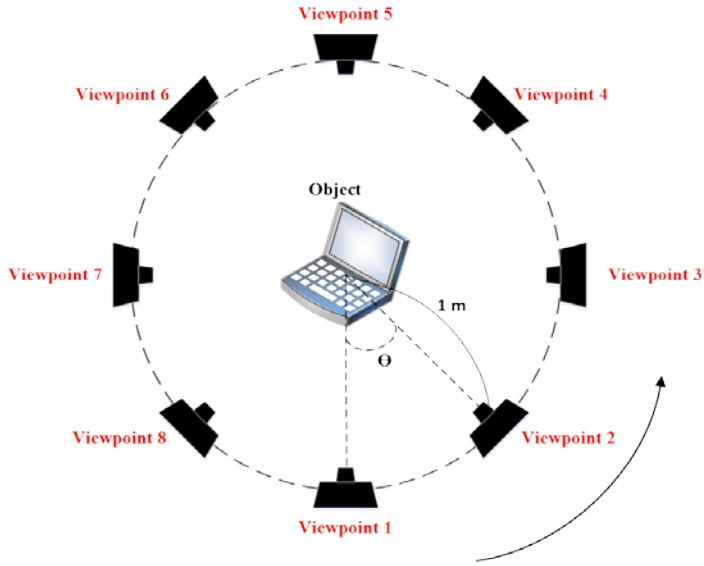


Figure 3.3: Data collection method of Washington dataset (Top view). The revolving direction of the camera is drawn arbitrarily. Viewpoints are spaced equally on the circle so that the angles between neighboring viewpoints are inversely proportional to the number of total viewpoints. For brevity, only 8 viewpoints are indicated.

Table 3.1: Selected categories for classification and training/test split

Selected category	Training instance	Test instance
apple	1,2,3,4	5
bell pepper	1,2,3,4,5	6
bowl	1,2,3,4,5	6
cap	1,2,3	4
cell phone	1,2,3,4	5
coffee mug	1,2,3,4,5,6,7	8
dry battery	1,2,3,4,5	6
notebook	1,2,3,4	5
pliers	1,2,3,4,5	6
water bottle	1,2,3,4,5,6,7,8,9	10

Point clouds of randomly picked instances from the ten categories are visualized in Figure 3.4.

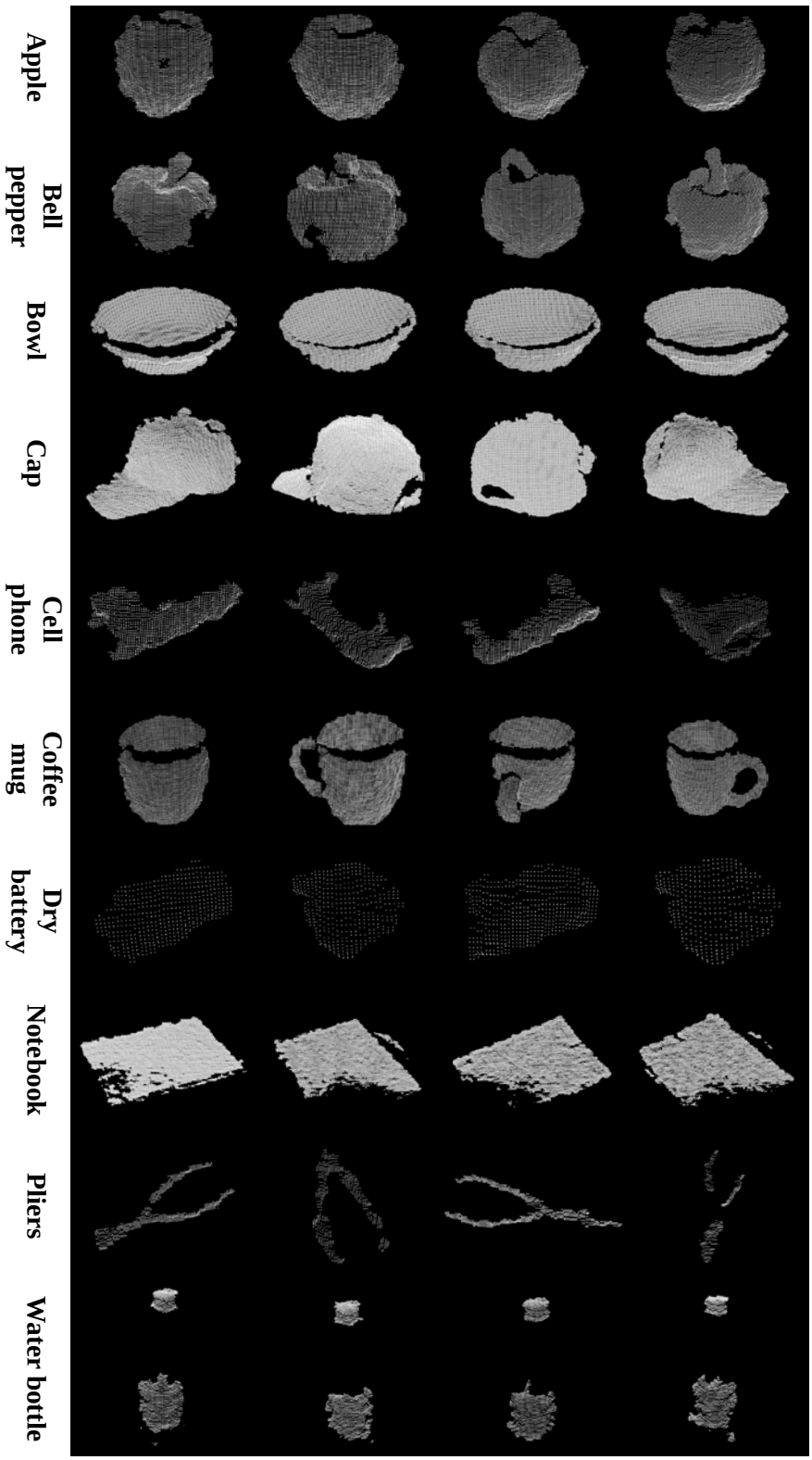


Figure 3.4: Ten object categories and four surfaces of an arbitrary instance from each category

3.2.2 Preprocessing

To train a CNN based neural network, the input data must have a grid-like topology. Hence, point cloud data should be preprocessed and transformed into voxel grids. Voxelizing point clouds is explained in this section. Coordinates of a point cloud are represented by the notation (x, y, z). Coordinates of a voxel are represented by the notation (i, j, k).

For each surface point cloud, the radius of the minimum bounding sphere is obtained by computing the maximum distance from the centroid to all the other points of the cloud.

$$r = \max_i \text{dist}(centroid, p_i) \tag{3.18}$$

Then, a square-shaped minimum bounding box is defined with its width, length and height being $2r$. A virtual reference point p_r with respect to which all the other points can have displacement is defined as (3.19)

$$p_r = (x_{centroid} - r, \quad y_{centroid} - r, \quad z_{centroid} - r) \tag{3.19}$$

This reference point represents the grid coordinate whose coordinate is $(\text{i}, \text{j}, \text{k}) = (0,0,0)$. Then, a point p_n of the point cloud has a grid coordinate as in (3.20)-(3.22).

$$i(p_n) = [(x(p_n) - x(p_r)) \,/\, (2r/div)] \tag{3.20}$$

$$j(p_n) = [(y(p_n) - y(p_r)) \,/\, (2r/div)] \tag{3.21}$$

$$k(p_n) = [(z(p_n) - z(p_r)) \,/\, (2r/div)] \tag{3.22}$$

In the equations of (3.20) – (3.22), $x(p_n)$, $y(p_n)$ and $z(p_n)$ are the *x*, *y*, and *z* coordinates of point p_n and $x(p_r)$, $y(p_r)$ and $z(p_r)$ are the *x*, *y*, and *z* coordinates of the reference point p_r. A factor div is introduced that denotes the number of grid divisions along an axis. Almost all CNN based approaches that voxelize 3D object use regular voxel grids, i.e. voxel grids of the same number of grid divisions along all three axes. Accordingly, in our approach, we use the same div for all three axes. The [·] expression is the Gauss' notation, an operator that returns the largest possible integer that does not exceed the operand. To rephrase it, the width, length and height of the minimum bounding box is divided into div divisions and the virtual reference point p_r is placed at the voxel whose grid coordinate is $(i, j, k) = (0,0,0)$. The (i, j, k) axes are aligned along the $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ directions respectively. In this way, another virtual reference point $p_r' = (x_{centroid} + r,\ y_{centroid} + r,\ z_{centroid} + r)$ will be placed at the voxel whose grid coordinate is $(i, j, k) = (div - 1, div - 1, div - 1)$. With this alignment of voxel grid, a point of the point cloud is given a grid coordinate of a voxel at which it is placed. Then, the voxelized point cloud data is obtained by counting the number of points in each voxel grid. That is, each voxel holds the number of points residing inside it as its numerical value. This is depicted in Figure 3.5 (b). The reason for this is to retain local shape information as much as possible which otherwise will be lost as in the case that the voxel only holds the binary state of being occupied as 0 (unoccupied) or 1 (occupied) (Figure. 3.5 (c)).

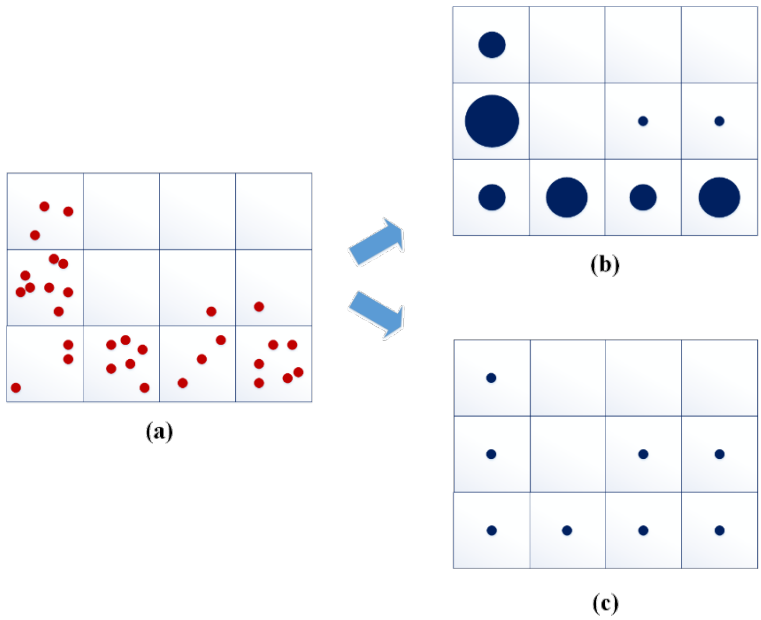
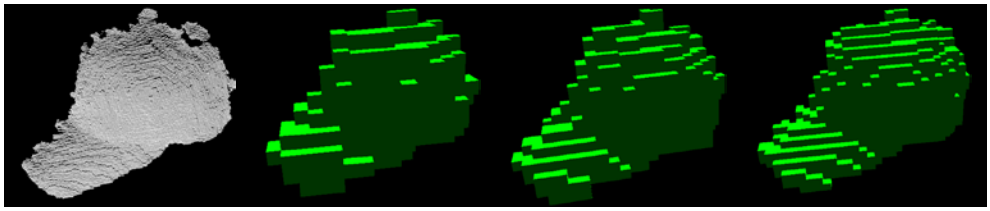


Figure 3.5: 2D illustration of two different ways of voxelizing a point cloud. (a) Point cloud. (b) Voxel grid whose voxels holds the number of points in it. (c) Voxel grid whose voxels hold binary values denoting whether they are occupied or not.

Voxelizing a point cloud data causes loss of information due to discretization. In other words, the number of occupied voxels is usually much smaller than the total number of points in the cloud. The parameter that controls the amount of this information loss is div. If div is large enough so that the leaf size of a voxel is comparable to the cloud resolution, then no information loss occurs. However, in consideration of computation and memory, div is usually set to a value in the range 20~30. In addition, since CNN takes a voxel grid data as a 3-dimensional array, the leaf size of a voxel is not reflected to the final feature vector of the neural network. In our approach, we use the value of $div = 32$. As one can easily expect, different values of div will yield different resolutions of the voxel grid so that the voxelized versions look quite differently from coarse to fine.



(a) **(b)** **(c)** **(d)**

Figure 3.6: A surface point cloud and its voxelized versions at three resolutions. (a) Surface point cloud. (b) Voxel grid of resolution 16. (c) Voxel grid of resolution 24. (d) Voxel grid of resolution 32.

Voxelized versions of instances in Figure 3.4 are visualized in Figure 3.7

.

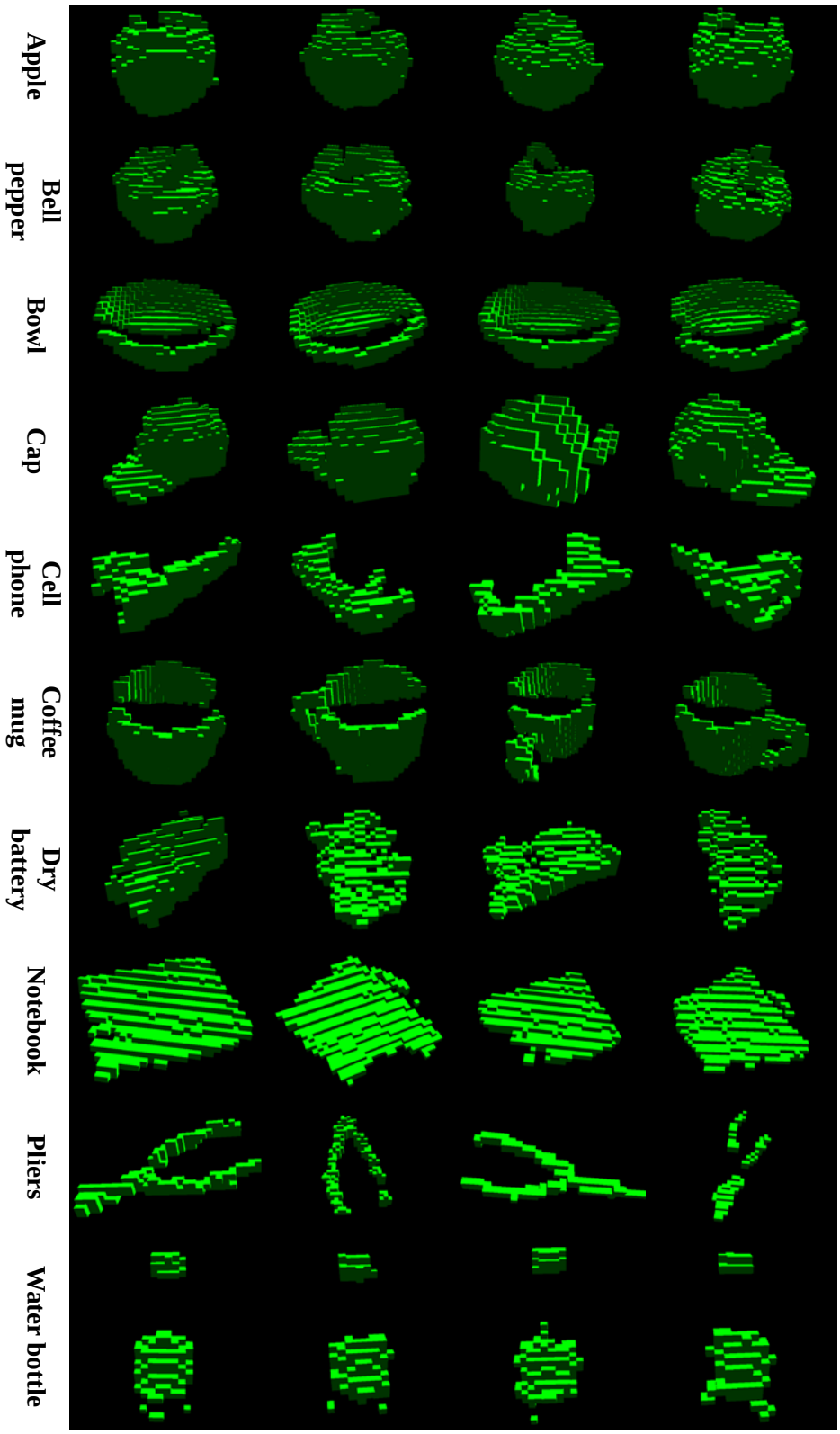


Figure 3.7: Voxelized point clouds.

3.2.3 Network structure and initialization of filter weights

Various network structures have been proposed in the literature of visual data classification. The structures and depths of these network vary from task to task, from dataset to dataset. In fact, it is difficult to determine the most suitable neural network structure among the innumerable structures. Therefore, in this thesis, a simple structure which has proved efficient for classification task is adopted. The network structure used in this thesis consists of three convolutional layers, two pooling layers and one fully-connected layer of softmax activation.

$$C(32,5,2) - C(32,3) - P(2) - C(16,3) - P(2) - FC(10)$$

The convolutional layers are denoted by a shorthand expression of $C(n,k,s)$ where the parameter n is the number of convolution kernels (filters), k is the kernel dimension and s is the subsample factor, also referred to as stride elsewhere. Since input data shape is cubic, the convolution kernel shape is chosen to be cubic. Thus, the parameter k indicates that the convolutional layer uses k×k×k convolution kernels. As mentioned before, ‘kernel’ is a term that can be used interchangeably with the term ‘filter’ in CNN terminology. The pooling layers are represented by $P(s)$ where s is the subsample factor. Fully-connected layer is represented by $FC(n)$ where n is the number of hidden units. For a fully-connected layer to play the role of a classifier, softmax activation is a common choice because softmax function produces the categorical probability for each class. It is typical to attach dropout layers after the pooling layers or hidden fully-connected layers to prevent overfitting. Hence, dropout layers are adopted in our approach.

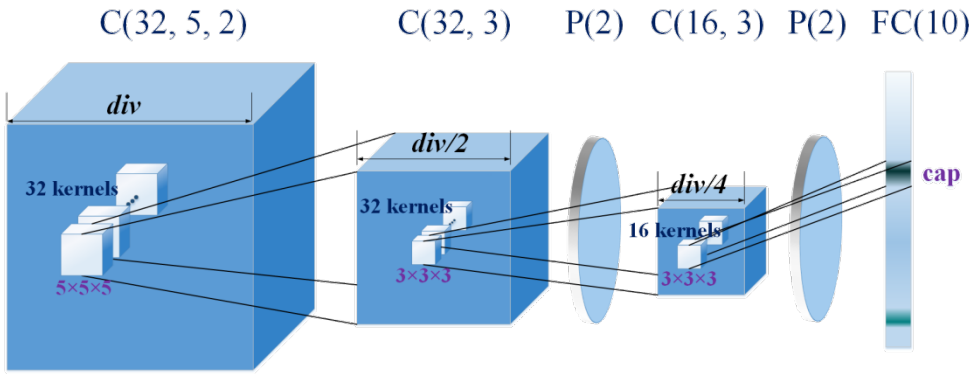


Figure 3.8: The proposed neural network structure. Dropout layer is not drawn in the figure. Softmax classifier produces categorical probabilities for the 10 categories (highest with the cap in this example).

This structure is very similar to the one proposed in [22]. However, it is observed that attaching additional fully-connected layers other than the final softmax classifier easily leads to overfitting and causes large generalization error. Thus, only one fully-connected layer is attached in our approach.

3.2.3.1 Cluster centers learned by k-means clustering

k-means clustering [38] is an unsupervised learning method that clusters the given input samples based on their Euclidean distance similarity. It carries out expectation-maximization (EM) [39] iteratively to maximize the clustering score. Using k-means clustering to learn a dictionary (feature mapping) that maps an input to a feature representation has been widely adopted in many vision frameworks [40] [41]. It has been empirically proved in many image-based classification methods that using the k-cluster centers as convolution filters helps the CNN perform better [21] [42] [43]. In this thesis, we extend this idea of 2D

CNN to 3D CNN. The whole procedure to get CNN filters are outlined in Figure 3.9. Intrisic Shape Signatures (ISS) [44] keypoints which are points showing large variations along the principal directions of the point clouds are extracted for this procedure. Detailed explanation of ISS keypoints is given in Chapter 4. The idea behind extracting keypoints is to make use of low-level intrinsic features of the input data. Because ISS keypoints are repeatable and discriminative points of a point cloud, the neighborhood of these keypoints are also distinctive sub-regions of the point cloud. Thus, the cluster centers of voxel grid patches around keypoints can be considered as representatives for the low-level primitives (patches) of the input data.

Before clustering, as custom preprocessing steps, local intensity normalization and whitening are applied to the extracted voxel grid patches. Two whitening methods, namely, Principal component analysis (PCA) whitening and Zero-phase

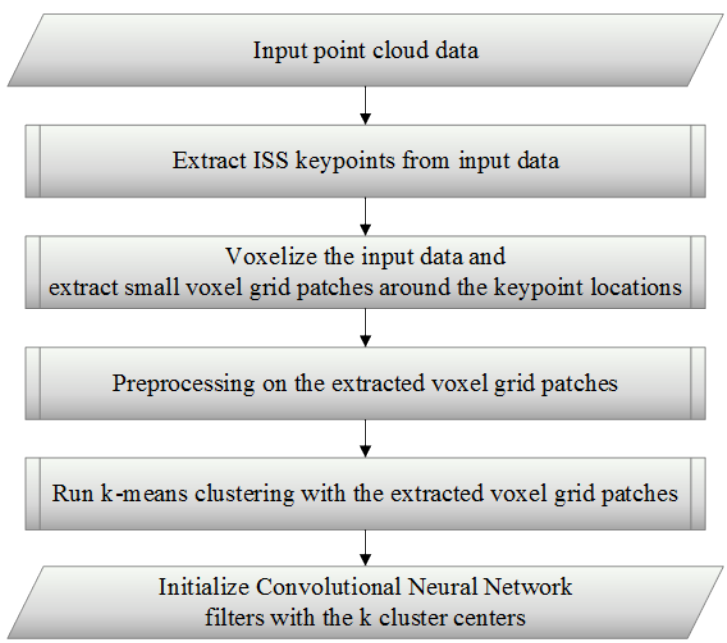


Figure 3.9: Flowchart of obtaining CNN filters by using k-means clustering.

component analysis (ZCA) whitening are applied respectively. Whitening is a prerequisite in the sense that it de-correlates the input data. Since k-means clustering cannot handle correlations of the input data, the obtained cluster centers of correlated input data usually contain redundancy. Also by whitening, the inputs have unit variances along each axis of their components. PCA whitening de-correlates the input data and makes them have unit variances. ZCA whitening only makes the input have unit variances. Detailed descriptions of the local intensity normalization and whitening are as follows. A voxel grid patch of size k×k×k can be viewed as a k^3-dimensional vector in the $\mathbb{R}^{k^3}$ space. First, local intensity normalization is carried out where each vector is subtracted by the mean of its k^3 components and then divided by the standard deviation of its components. The normalization scales every voxel patch into a comparable range irrespective of where they are extracted from. This makes them have only shape information. After normalization, a matrix V is defined whose columns are the vectors of normalized voxel grid patches. V matrix is then refined as in (3.23)-(3.24).

$$V = [v_1, v_2, \cdots, v_n] \in \mathbb{R}^{m \times n} \text{ where each } v_i \in \mathbb{R}^m \text{ and } m = k^3 \quad (3.23)$$

$$\bar{V} = [\bar{v}_1, \bar{v}_2, \cdots, \bar{v}_n] \quad \text{where each } \bar{v}_i = v_i - \mu \text{ and } \mu = \frac{1}{n}\Sigma_{j=1}^{n} v_j \quad (3.24)$$

Then, the covariance matrix C is computed as $\mathrm{C} = \frac{1}{n}\bar{V}^t\bar{V} \in \mathbb{R}^{m \times m}$. It is decomposed into $C = P\Lambda P^t$ by Eigenvalue decomposition (EVD) where P is an m ×m orthogonal matrix whose columns are eigenvectors of C and Λ is an m×m diagonal matrix whose diagonal elements are the eigenvalues of C. Eigenvalue decomposition is a special case of Singular value decomposition (SVD) when the

matrix to decompose is symmetric. SVD will be discussed next Chapter when applied to find the optimal rotation for rigid transformation. The PCA whitening and ZCA whitening are implemented as in (3.25)-(3.26).

$$\text{PCA whitening}: \ v_i \mapsto \Lambda^{-\frac{1}{2}} P^t (v_i - \mu) \tag{3.25}$$

$$\text{ZCA whitening}: \ v_i \mapsto P\Lambda^{-\frac{1}{2}} P^t (v_i - \mu) \tag{3.26}$$

As mentioned above, PCA whitening rotates the vectors so that they are aligned with the principal directions and de-correlated. The PCA whitened vectors have unit variance as well. The ZCA whitening only makes the vectors have unit variance and does not rotate them in the direction of principal axes, thereby transforming the vectors as close as the original ones in the least square sense.

Convolution filters in deeper convolutional layers usually have more parameters (weights) than those in the first layer because the feature map computed from the previous convolutional layer has channels as many as the number of convolution filters of the previous layer. For example, if the first convolutional layer has *m* convolution filters, then the feature map computed from the first layer has *m* channels. Thus, convolution filters in the second convolutional layers should have as many weight values as to handle all *m* channels of the feature map. This means that initializing the filter weights in deeper convolutional layers requires the k-means clustering algorithm to perform on a very high-dimensional input space, in which case the k-means clustering yields poor cluster centers and takes excessive computation time. In our approach, therefore, only the first convolutional layer of the network is initialized with the k-means cluster centers.

3.2.3.2 Encoding stacks of a pre-trained de-noising convolutional auto-encoder

An auto-encoder (AE) is a generative model that outputs a reconstructed version of the given input. An auto-encoder comprises two parts, namely, the encoder and the decoder. The encoder maps the given input **x** into another representation **y** and the decoder reversely maps **y** into **x**. The representation **y** is usually under-complete meaning that its dimension is smaller than the dimension of the input. In general, auto-encoders are used as de-noising auto-encoders (DAE) that reconstruct a noiseless version given the noised input. That is, auto-encoders are trained to de-noise the noised input. This is not only meaningful for de-noising but also for restricting the auto-encoder to learn the identity mapping.

In [45], it is shown in information-theoretic perspective that minimizing the reconstruction error of an auto-encoder amounts to maximizing a lower bound on the mutual information $I(\mathbf{x};\mathbf{y})$ between the input **x** and the learned representation **y**. If true target values that the learned representation **y** should represent are given, then error measure can be defined and the encoding stack of the auto-encoder can be trained by back-propagation from the middle of the auto-encoder. That is, supervised training criterion can be applied to fine tune the encoding stack of the pre-trained auto-encoder. It is proved that using this unsupervised pre-training before fine-tuning actually helps the supervised learning system perform better.

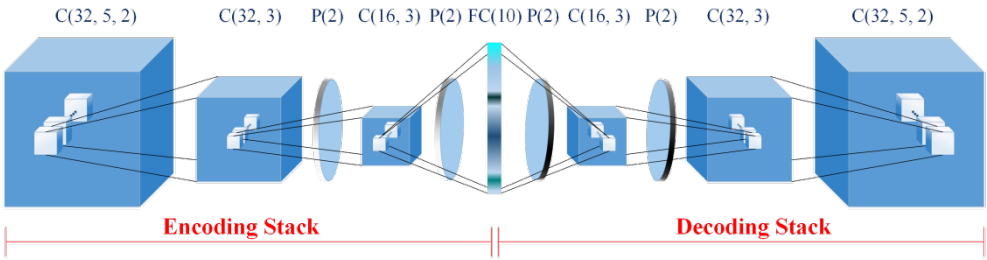


Figure 3.10: The structure of the de-noising convolutional auto-encoder (DCAE) for pre-training. The encoder stack decompresses the noised input data into an under-complete representation while the decoder stack reconstructs the noiseless input from the representation. The trained encoder stack is used as a pre-trained model and fine-tuned with supervised criterion.

In this thesis, since the network structure basis is the convolutional neural network, the Convolutional Auto-Encoder (CAE) [46] is adopted for pre-training. The encoder of the CAE has the same structure as the CNN used as the classifier and the CAE is restricted to de-noise the inputs. The 3D de-noising Convolutional Auto-Encoder (DCAE) proposed in our approach is structured like

$$C(32,5,2) - C(32,3) - P(2) - C(16,3) - P(2) - FC(10) - UP(2) - C(16,3) - UP(2) - C(32,3) - UP(2) - C(32,5)$$

It is shown in Figure 3.10. It is trained to minimize the reconstruction error. After this pre-training, the encoder is fine-tuned with supervised criterion by back-propagation.

3.2.4 Improvement of performance by orientation estimation

Classification task is meaningful only if inter-class variation causes significant change in the way the object looks, i.e. if the category of an object makes the object shape distinctive from the other object shapes in different categories. This is a basic ground on which a classification system (category recognition system) is built. If the training dataset comprises (shape) data that are not distinctive of the class, classification task with this dataset becomes pointless.

There is another factor that changes the shape of a surface point cloud. It is the orientation of the surface. The shape of an object looks different as the viewpoint changes. Equivalently, the orientation of a surface with respect to the object center influences the way the object is visually perceived. Thus, it can be stated that the orientation of a surface point cloud is another type of class to which single training datum belongs. Furthermore, in this thesis, it is asserted that exploiting this additional information brings about improvement in the classification performance.

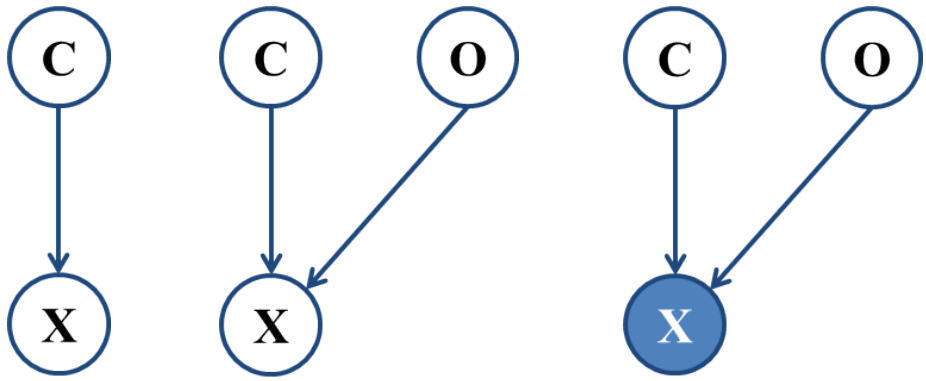
Two things must be emphasized in advance. First, the neural network that has been discussed so far is a discriminative model that aims to learn the target distribution $p(\mathbf{c}|\mathbf{x})$ rather than a target function $f(\mathbf{x}) = \mathbf{c}$. It is more natural to regard the variation of input $\mathbf{x}$ (whether intra-class or inter-class) as stochastically noising an ideal (unknown) input sample than to think of it as generating myriads of ideal input samples. From this viewpoint, no deterministic target function f that maps $\mathbf{x}$ into $\mathbf{c}$ exists. Instead, $\mathbf{x}$ is generated from an unknown distribution $p(\mathbf{x})$ and the classification system is an estimator for the posterior probability $p(\mathbf{c}|\mathbf{x})$ (not for $f(\mathbf{x}) = \mathbf{c}$). Probabilistic graphical models are useful in this respect. Second,

discussions that will be developed do not provide a rigorous theory for how the parameters of the neural network are tuned to improve the classification performance. However, they provide a sketchy but reasonable explanation that the proposed neural network gets improved by exploiting the additional orientation information.

The infomax principle of Linsker [47] asserts that the neural network should be trained to maximize the mutual information between the given input **x** and its representation **y**. The representation **y** translates into the estimated class **c** in our method. The goal of the classification system is then to maximize the mutual information $\mathrm{I}(\mathbf{x};\mathbf{c})$. The graphical model corresponding to the classification system that considers only object class **c** and its shape **x** is shown as Figure 3.11(a). If the orientation of the object surface is additionally considered, the graphical model will be like the one in Figure 3.11(b). In this model, both the class **c** and the orientation **o** are factors that determine the input shape **x**. In addition, **c** and **o** are d-separated meaning that they are independent from each other unless the common effect **x** is observed. Based on this observation, Equations (3.27) tells that the mutual information between the object class **c** and the input shape **x** given the orientation **o** is lower-bounded by the mutual information of **c** and **x** with no other variables. That is, $\mathrm{I}(\mathbf{c};\mathbf{x}|\mathbf{o}) \geq \mathrm{I}(\mathbf{c};\mathbf{x})$.

$$\mathrm{I}(\mathbf{c};\mathbf{x}) + \mathrm{I}(\mathbf{c};\mathbf{o}|\mathbf{x}) = \mathrm{I}(\mathbf{c};\mathbf{x}|\mathbf{o}) \tag{3.27}$$

It is derived from trivial relations $\mathrm{I}(\mathbf{c};\mathbf{x}) + \mathrm{I}(\mathbf{c};\mathbf{o}|\boldsymbol{x}) = \mathrm{I}(\mathbf{c};\mathbf{o}) + \mathrm{I}(\mathbf{c};\mathbf{x}|\mathbf{o})$ and $\mathrm{I}(\mathbf{c};\mathbf{o}) = 0$.



(a) (b) (c)

Figure 3.11: Graphical models describing the proposed classification system. (a) Class is the only factor that determines the input shape. (b) Class and orientation are factors that determine the input shape. Class and orientation are d-separated so that they are independent when x is unobserved. (c) Class and orientation are dependent if the input shape is observed.

To make use of orientation information in our approach, every object instance is first assigned with orientation labels. If m orientation labels are to be assigned to a single object instance, all the viewpoints at which the object instance is captured are grouped into m equally-spaced viewpoint sets. The m grouped viewpoint sets are then labeled as 0,1,⋯,m-1 in regular sequence. For example, if 6 orientations are to be assigned to 'cap 1' instance at 30° height that has 190 surfaces, the first 32 surfaces are labeled 0, the second 32 surfaces are labeled 1, and so forth. The number of orientation labels differ from class to class. For example, since class 'apple' has instances that are almost rotationally invariant, all the surfaces of apple instances are labeled as 0. Instances in the class 'cap', on the other hand, exhibit significantly different appearances as the viewpoint changes. Thus, more than one orientation labels are assigned to this class. Referring to prior knowledge $p(\mathbf{c})$ on the class, different numbers of orientation labels are assigned to instances in different classes as listed in Table 3.2.

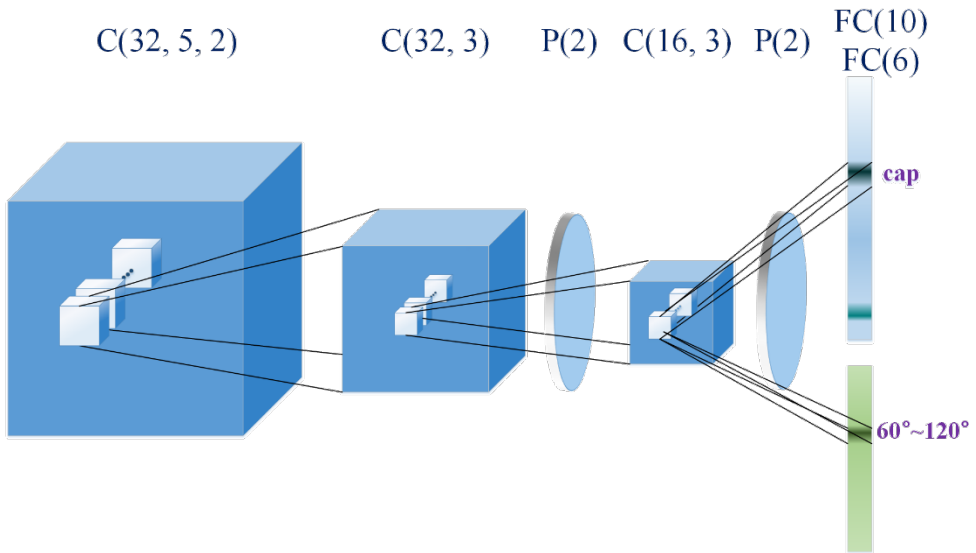


Figure 3.12: Proposed CNN structure that estimates both the class label and the orientation label.

Table 3.2: Number of orientation labels for each class

Category	apple bell pepper bowl water bottle	cap cell phone pliers	coffee mug dry battery notebook
Number of orientation labels	1	6	4

After labeling, the proposed CNN is made to optimize to separate loss functions simultaneously, one for the class label and the other for the orientation label. The term ‘loss’ refers to the cost incurred by the output of the neural network. The error measure of a neural network is defined by the loss function. Here and after, the term ‘loss function’ is used interchangeably with the term ‘error function’. Two errors made from two loss functions are propagated to tune the parameters of the network. The CNN now estimates the surface orientation of the given input as well as the class of it as shown in Figure 3.12.

Chapter 4

RANSAC based shape retrieval

4.1 Preliminaries

4.1.1 Rigid transformation

Rigid transformation, also called rigid motion, is a transformation on a vector space that preserves distances between every pair of vectors. Rotation around an axis, translation by an offset and their composite are rigid transformations. A rigid motion transforms one coordinate system into another. General rigid transformation in 3D space has six free parameters – roll, pitch, yaw angles for rotations around fixed X, Y, Z axes and three displacement components along three axes. Even if not expressed explicitly by these six components, other parameters that can be equivalently expressed by them add up six. Thus, to determine a unique rigid motion in 3D space, minimum of three pairs of points between two coordinate systems are required. Symbols and explanations for 3D rigid transformation are now given.

Table 4.1: Symbols for point and transformation

Point p referenced to frame A	${}^{A}p$
Point p referenced to frame B	${}^{B}p$
Translation vector describing frame B relative to frame A	T
Rotation matrix describing frame B relative to frame A	${}^{A}_{B}R$
Homogeneous transformation matrix describing frame B relative to frame A	${}^{A}_{B}T$

Table 4.1 shows symbols related to 3D rigid motion. Two coordinate systems A and B are referred to as frame A and frame B respectively. From now on, the terms 'coordinate system' and 'frame' have the same meaning. A point is said to be referenced to a coordinate system if its components have numerical values along the axes of that coordinate system. A transformation matrix describes frame B relative to frame A if it changes reference frame of points from B to A.

Rotation matrix and translation vector that describe frame B relative to frame A can be written as

$$ {}^{A}_{B}R = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix} \tag{4.1} $$

$$ \mathrm{t} = \begin{bmatrix} x_{BORG} - x_{AORG} \\ y_{BORG} - y_{AORG} \\ z_{BORG} - z_{AORG} \end{bmatrix} \tag{4.2} $$

A general rigid transformation that changes the reference frame of points from B to A has the following relationship in (4.3).

$$^{A}p = {}^{A}_{B}R \cdot {}^{B}p + t \tag{4.3}$$

Here, the translation vector t is the same as origin of frame B referenced to frame A, i.e. $^{A}p_{BORG}$. This relationship can be re-written into a more compact form, namely, homogeneous transformation matrix relation as in (4.4).

$$^{A}p = {}^{A}_{B}T \, {}^{B}p \tag{4.4}$$

In this expression, vectors representing the points are four-dimensional ones with the fourth component being a dummy constant of 1. Homogeneous transformation matrix $^{A}_{B}T$ is a 4×4 matrix in the form of (4.5)

$$^{A}_{B}T = \begin{bmatrix} {}^{A}_{B}R & t \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{4.5}$$

The discussions so far deal with the situation in which two different coordinate systems describe the same point. Thus, the rigid transformation gives a description of one coordinate system with respect to the other. However, these arguments is also valid when only one reference frame exists and a point is transformed into another by rigid transformation.

Two point sets $\{p_i\}$ and $\{q_i\}$ where $i = 1, \cdots, n$ are given. Let q_i be the transformed version of p_i, i.e. corresponding point of p_i. The points are 3×1 column vectors.

$$q_i = R \cdot p_i + t \tag{4.6}$$

In (4.6), R is a 3×3 rotation matrix and t is a 3×1 translation vector. Since true correspondences are known, it is natural to use the least square method to give the solution robustness to noise (even though the noise is not explicitly expressed). Thus, the objective function to minimize is the sum of squared error between corresponding points.

$$\text{SSE} = \Sigma_{i=1}^{n} \| q_i - (R \cdot p_i + t) \|^2 \tag{4.7}$$

The optimization is done with respect to R and t simultaneously. However, this task can be decoupled into finding the optimal R and t separately. To do so, the points $\{p_i\}$ and $\{q_i\}$ are subtracted by their means and denoted as $\{p_i'\}$ and $\{q_i'\}$ respectively.

$$p_i' = p_i - \bar{p} \quad , \quad q_i' = q_i - \bar{q} \quad \text{for} \quad i = 1, \cdots, n \tag{4.8}$$

$$\bar{p} = \frac{1}{n} \Sigma_{i=1}^{n} p_i \quad , \quad \bar{q} = \frac{1}{n} \Sigma_{i=1}^{n} q_i \tag{4.9}$$

Then the sum of squared error can be expressed as in (4.10)

$$\text{SSE} = \Sigma_{i=1}^{n} \| q_i' - (R \cdot p_i' + t') \|^2 \tag{4.10}$$

where $t' = t + R \cdot \bar{p} - \bar{q}$. Expanding this equality gives

$$\begin{aligned} \text{SSE} = \ & \Sigma_{i=1}^{n} \| q_i' - R \cdot p_i' \|^2 - 2\Sigma_{i=1}^{n} (q_i' - Rp_i') \cdot (t + R\bar{p} - \bar{q}) \\ & + \Sigma_{i=1}^{n} \| t + R\bar{p} - \bar{q} \|^2 \end{aligned} \tag{4.11}$$

The second term on the right goes to zero because by definition, $\Sigma_{i=1}^{n} q_i' = 0$ and $\Sigma_{i=1}^{n} p_i' = 0$. The last term is always non-negative, so it has to equal zero if SSE is to be minimized. Thus, the optimal translation t is obtained as in (4.12).

$$t = \bar{q} - R\bar{p} \tag{4.12}$$

Now, minimizing SSE is achieved by minimizing the first term, $\Sigma_{i=1}^{n} \| q_i' - R \cdot p_i' \|^2$. This term can be expanded as in (4.13)

$$\begin{aligned} \Sigma_{i=1}^{n} \| q_i' - R \cdot p_i' \|^2 &= \Sigma_{i=1}^{n} (q_i' - Rp_i')^t (q_i' - Rp_i') \\ &= \Sigma_{i=1}^{n} ({q_i'}^t q_i' + {p_i'}^t R^t R p_i' - 2{q_i'}^t R p_i') \end{aligned} \tag{4.13}$$

Minimizing (4.13) is equivalent to only maximizing the last negative signed term ${q_i'}^t R p_i'$ because the orthogonal rotation matrix satisfies the relation $R^t R = I$ and it makes the second term in (4.13) have no effect on the minimization.

Two approaches can be used to the maximization of ${q_i'}^t R p_i'$. One is to use quaternion [48] and the other is to use Singular Value Decomposition (SVD) of a symmetric matrix [49]. Both methods are non-iterative and assure closed-form solutions. In this thesis, we make use of the SVD based method.

Keeping the notations, a symmetric matrix H can be defined as in (4.14).

$$H = \Sigma_{i=1}^{n} p_i' q_i'^t \tag{4.14}$$

By SVD, H is decomposed into $H = U\Lambda V^t$ where U and V are 3×3 orthogonal matrices and Λ is a 3×3 diagonal matrix whose diagonal elements are the eigenvalues of H. Then the optimal rotation matrix is obtained as in (4.15).

$$R = VU^t \tag{4.15}$$

There is a case where the determinant of R equals -1 which is numerically possible but physically insensible. In this case, R is a reflection matrix. To make it a rotation matrix, the third column of V needs to be multiplied by -1 and then R needs re-computing afterwards.

4.1.2 Random sample consensus

Random sample consensus (RANSAC) [5] is a universal algorithm that can be applied to parametric modeling problems that determine model parameters with the given observed data. RANSAC is especially useful when the observed data contain too many outlier points for the Least Square (LS) method to provide a reliable solution.

Outliers are distinct from noisy data points. Noisy data points are considered as being generated from the model with stochastic noise added. On the other hand, outliers are false observations or data points affected by a completely different type of noise source. Data points that are not outliers are called inliers. The error

measure of LS method penalizes outliers heavily because their deviations from ideal model do not follow the noise distribution.

The basic idea of RANSAC is the following. First, we randomly sample the minimum number of data points required to determine the model parameters. Second, we solve for the model parameters with the sampled data points. Then, assuming that the determined model is true, an error is computed for every data point. Error measure is different from case to case but Euclidean distance is general. Points that have errors smaller than a pre-set tolerance are counted as inliers. The set of inliers is called the consensus set of the determined model. At this moment, if the determined model is close to the true target model, the number of inliers will be large. If this model is far from the target model, the number of inliers will be small. The process of randomly sampling data points and determining the model with these samples is repeated sufficiently and the best model that has had the largest number of inliers are chosen to be the final estimated model.

A pseudo code of the RANSAC algorithm is given below.

Algorithm 1 RANSAC

Objective Robust fit of a model to data set S that contain outliers

i. Randomly sample n data points from S
ii. Determine the model parameters using the n samples
iii. Determine inlier set S_i; a set of data points whose errors are within a pre-defined tolerance ε
iv. If the $|S_i|$ is greater than a threshold T, re-estimate the model using all the inlier samples and terminate
v. If $|S_i|$ is smaller than T, repeat i - iv
vi. After N trials, the largest consensus set S_i is given.

The predefined error tolerance ε, the consensus set size threshold T and the maximum number of trials N are the RANSAC parameters that control the quality of solution RANSAC returns. If the noises added to data points follow Gaussian distribution, then the squared distance of each point follows the Chi-distribution. The error tolerance (distance threshold) can be determined by the inverse of Cumulative Distribution Function (cdf) of Chi-distribution with the given confidence level. However, the noise distribution is unknown in most cases. Thus, the error tolerance is set empirically. Other two parameters T and N can be set reasonably with given confidence level. However, determining them requires us to know the probability that each data point is an inlier. This is not possible in most modeling problems. Therefore, termination criteria must be determined adaptively. A useful way of adaptively determining the number of samplings is described below.

Algorithm 2 Adaptive determination of termination criterion for RANSAC

1. Let $N = \infty,\ N_{sampling} = 0$
2. $While\ N > N_{sampling}\ Repeat:$
   i. Randomly sample n data points and determine model parameters
   ii. Count the number of inliers
   iii. Set $\varepsilon = \frac{number\ of\ inliers}{total\ number\ of\ data\ points}$
   iv. Set $N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^n)}$ with confidence $p = 0.99$
   v. Increment $N_{samping}$ by 1
3. Terminate

The meaning of the equality $N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^n)}$ is explained next. The probability

of a data point being inlier is given by ε. Then the argument holds with probability p that if the RANSAC algorithm runs for N times, at least one of the N estimated models is obtained solely from inliers.

4.2 Proposed method

The single-view surface point clouds used in our experiments are captured by a camera rotating around the object for a round. Accordingly, the coordinate system to which each surface point cloud is referenced changes as the pose of the camera changes.

To retrieve the complete 3-D shape of a volumetric object, all surface point clouds $\{ {}^{0}pc,\ {}^{1}pc, \cdots,\ {}^{n}pc \}$ must be registered and referenced to a single unified coordinate system. ${}^{i}pc$ means the ith captured surface point cloud. This superscript on the left represents the coordinate system. The coordinate system is denoted as $\{i\},\ i = 1, \cdots, n$ as of now. The coordinate system of the point cloud first captured $\{0\}$ is defined as the target frame for convenience. The description of all the other frames with respect to the target frame makes all the other surface point clouds be described in the target frame. This gives us a volumetric point cloud referenced to the target coordinate system.

To estimate the rigid transformation that describe a surface point cloud's frame $\{i\}$ relative to the target frame $\{0\}$, the rigid transformations between all contiguous frames preceding this frame are evaluated first.

$$ {}^{0}_{1}T, {}^{1}_{2}T, \cdots, {}^{i-1}_{i}T \tag{4.16} $$

After that, estimated rigid transformations are composed sequentially to yield the rigid transformation that transforms the point clouds into the target coordinate system.

$$ {}^{0}_{i}T = \Pi_{k=0}^{i}({}_{k+1}^{k}T) \tag{4.17} $$

This is much easier and more reliable than directly estimating ${}^{0}_{i}T$ from frames {0} and {i} since the estimation of a rigid transformation requires at least three physically (not numerically) corresponding points between coordinate systems but the frames {0} and {i} may have no overlapping region at all.

RANSAC algorithm is utilized to determine the rigid transformation between two neighboring frames. Two point clouds ${}^{i-1}pc$ and ${}^{i}pc$ are given which are referenced to the coordinate frames {i-1} and {i} respectively. At first, keypoints $\{{}^{i-1}p_0, \cdots, {}^{i-1}p_m\}$ and $\{ {}^{i}q_0, \cdots, {}^{i}q_l\}$ are extracted from two given clouds. Then, local shape descriptors $\{{}^{i-1}d_0, \cdots, {}^{i-1}d_m\}$ and $\{ {}^{i}f_0, \cdots, {}^{i}f_l\}$ are defined on the extracted keypoints. The descriptors are used to find putative correspondences $\{c_{jk}^{n} : n = 1, \cdots, N\}$ between keypoints of the two point clouds. Here, c_{jk}^{n} denotes the correspondence associated between ${}^{i-1}p_j$ and ${}^{i}q_k$. N is the total number of correspondences established. The obtained putative correspondence set usually contains a portion of false correspondences unless very exquisitely engineered descriptors are used - which is uncommon. Because two point clouds always have non-overlapping regions, it is likely that false correspondences are associated. This is the reason that RANSAC is adopted. True correspondences are considered as inliers and false correspondences are considered as outliers. RANSAC is known to

deal with outliers as much as 40~50 % of the data points. This implies that in practice, two point clouds with only halves of their region overlapped can be registered by the rigid transformation returned by RANSAC.

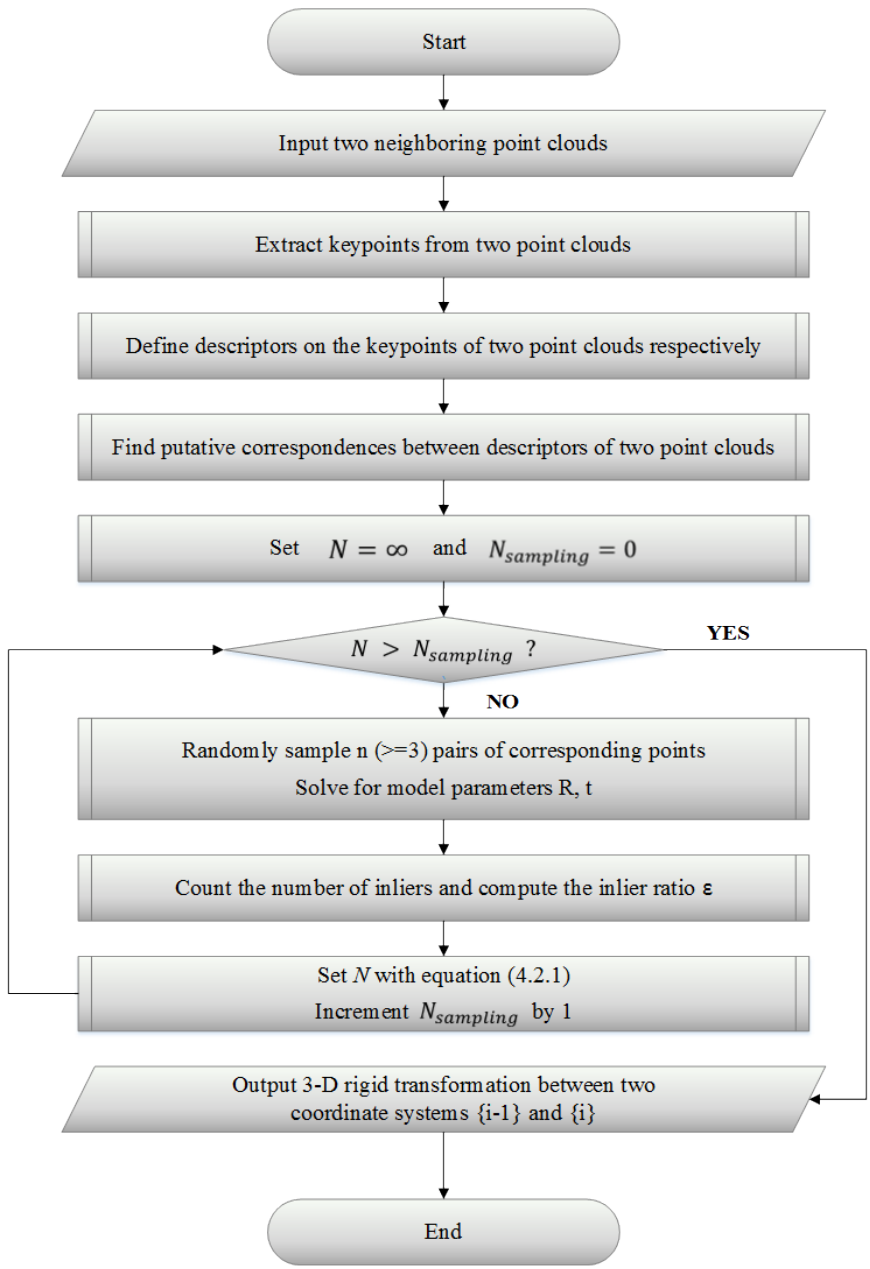


Figure 4.1: Flowchart of rigid transformation estimation by adaptive RANSAC.

Intrinsic Shape Signature (ISS) keypoints and Signature of Histogram of OrienTation (SHOT) descriptors are used in the proposed method. ISS keypoints are computed by eigenvalue decomposition of covariance matrix of the point cloud. Points whose spreads are similar along the principal directions on which repeatable local reference frames cannot be defined are pruned first. Non-maxima suppression over the magnitude of the smallest eigenvalue is performed and eventually keypoints with large variations along each principal direction are obtained. SHOT descriptor is explained in Chapter 2. Detailed procedures of estimating the rigid transformation by RANSAC algorithm is illustrated in Figure 4.1.

In our approach, Euclidean distance is used for the error measure and correspondences are associated between two descriptors that are closer than a given threshold. It is unknown how likely a false correspondences will be generated, i.e. the probability of a correspondence being an outlier is unknown. Therefore, the RANSAC implementation is made to terminate adaptively. As mentioned in Section 4.1, optimal rotation and optimal translation is computed separately. The H matrix to be decomposed is made by all corresponding points. To avoid abuse of small upper/lower suffixes, the subscripts j, k denoting the order of keypoints in each coordinate frame and the superscripts i-1, i denoting the coordinate frames are dropped. Instead, n is used as a superscript to denote the order of keypoints in the correspondence set. With these notations, H can be written as in (4.18)

$$\mathrm{H} = \Sigma_{n=1}^{N}(p^n - \bar{p})(q^n - \bar{q})^t \tag{4.18}$$

$$\bar{p} = \frac{1}{N}\Sigma_{n=1}^{N}p^n \quad , \quad \bar{q} = \frac{1}{N}\Sigma_{n=1}^{N}q^n \tag{4.19}$$

Bringing results from the previous Section 4.1, the optimal rotation R and t are

$$R = VU^t \tag{4.21}$$

where

$$[U, \Lambda, V] = SVD(H),\ t = \bar{q} - R\bar{p} \tag{4.22}$$

After the registration is finished, statistical outlier removal is applied that removes points whose mean distances from their neighbors are larger than the global mean distance multiplied by global standard deviation of distances. This trimming procedure smooths the retrieved object surface and removes arbitrarily grouped point sets.

Chapter 5

Evaluation

5.1 Classification performance

5.1.1 Network structure test and loss function

As mentioned in Chapter 3, various network structures were tested. Table 5.1 lists seven neural networks initialized by glorot-uniform distribution. They are notated by the shorthand expressions introduced in Chapter 3. Training was performed on 30546 training data from the 10 designated classes. Test dataset contains 6674 data. Mean test accuracy and loss are reported over 10 iterations of training and each training was performed over 2 epochs of the training data. Mean loss is a more important criterion evaluating the performance of the network. From the table, the structure C(32,5,2)-C(32,3)-P(2)-C(16,3)-P(2)-D(10) showed the least mean loss and the highest accuracy. This structure is thus selected as mentioned in Chapter 3. However, unless any network structure can be adopted as a basic structure without losing too much performance unless it incurs too much overfitting or underfitting.

Table 5.1: Seven network structures and their performances.

Network structure	Mean Loss	Mean Accuracy (%)
C(16,3,2)-C(16,3)-P(2)-C(8,3)-P(2)-D(10)	0.2866	89.36
C(32,3,2)-C(16,3)-P(2)-C(16,3)-P(2)-D(10)	0.2926	89.40
C(32,5,2)-C(32,3)-P(2)-C(16,3)-P(2)-D(128)-D(10)	0.3083	90.99
C(32,3,2)-C(32,3)-P(2)-C(16,3)-P(2)-D(10)	0.3365	91.99
C(32,5,2)-C(32,3)-P(2)-C(16,3)-P(2)-D(10)	0.2759	92.21
C(64,3,2)-C(32,3)-P(2)-C(32,3)-P(2)-D(10)	0.3880	90.95
C(64,3,2)-C(32,3)-P(2)-C(32,3)-P(2)-D(256)-D(10)	0.3392	91.12

The overall performances of a neural network including classification accuracy and loss value, how much they overfit the data and how well they generalize to unseen data depend on various factors such as the number of hidden layers (depth of neural network), number of hidden units, weight initialization and so on. However, the functional behavior that the neural network exhibits hinges on the objective function. Two neural networks with the same structure and the same parameter settings can behave completely different if their objective functions differ. Objective function is also referred to as loss function. In this thesis, categorical cross-entropy loss that proves the most suitable loss function for multi-class classification is used. Furthermore, the categorical cross-entropy error is a more accurate performance indicator than the classification accuracy. Following discussions explain the reason. Consider a three-class classification problem where each class is just represented by A, B and C. A classifier gets an input and outputs three categorical probability values indicating the probabilities that the given input belongs to each class.

Table 5.2: Classification performances of two different classifiers.

Classifier 1	**Classifier output**			**True label**			**Classified**
	A	B	C	A	B	C	
Input 1	0.3	0.3	0.4	0	0	1	Yes
Input 2	0.4	0.3	0.3	1	0	0	Yes
Input 3	0.1	0.1	0.8	0	1	0	No
Classifier 2	**Classifier output**			**True label**			**Classified**
	A	B	C	A	B	C	
Input 1	0.1	0.1	0.8	0	0	1	Yes
Input 2	0.8	0.1	0.1	1	0	0	Yes
Input 3	0.3	0.3	0.4	0	1	0	No

Assume there are two classifiers as shown in Table 5.2. For classifier 1, since two of the three inputs are assigned with the correct label, the classification error is 1/3 = 0.33. Then, consider the classifier 2 with the same input-output pairs. Classifier 2 has the same classification error of 0.33 but there is a difference between the classifier 1 and classifier 2. Classifier 2 makes correct expectations on input 1 and input 2 with high confidence and slightly misses the third input's label. On the other hand, classifier 1 narrowly makes correct expectations on input 1 and input 2 while making a ridiculous expectation on the third input. This tells that classification accuracy is a somewhat crude measure of performance for classification systems. Cross-entropy error measure is more elaborate in this case. Cross-entropy is an error measure approximating a true probability distribution p with an estimated probability distribution q. Cross-entropy is defined as $\mathrm{H}(\mathrm{p},\mathrm{q}) = -\Sigma_x p(x)\log q(x)$ for two probability mass functions (pmf) p and q. Based on this measure, the computed cross-entropy for classifier 1 is 3.763991 and the cross-entropy for classifier 2 is 1.502131. This tells that the classifier 1 has an error much larger than that of classifier 2.

From now on, the cross-entropy error measure will be considered a more important measure than the classification accuracy when it comes to evaluating the performances of classifiers. In general, a classifier with low cross-entropy error shows high classification accuracy. However, in some cases, it is possible that classification accuracies between classifiers show slight difference when there are observable gaps in the cross-entropy errors. The opposite situation is unlikely to occur.

5.1.2 Performance evaluation of the proposed CNN and different initialization methods

The experiment was conducted as follows. Three differently initialized neural networks mentioned in Chapter 3 were trained on 30546 training data of surface point clouds. Their classification accuracies and cross-entropy errors were evaluated by 6674 test data. Three neural networks are again as follows. The first one is initialized by scaled random numbers from a uniform distribution. This random initialization method proposed in [50] performs better than other types of random initializations. It is a basic choice for typical neural networks. The second one is initialized with the encoder stacks of a de-noising convolutional auto encoder (DCAE). The last one is initialized with the cluster centers obtained from k-means clustering where extracted voxel grid patches around ISS keypoints form the clusters. For the last neural network, two different types of whitening are performed respectively as a preprocessing step - the PCA and the ZCA whitening. As a result, four differently initialized CNNs were investigated. Because tuning of the CNN parameters is done with stochastically sampled mini-batches of training

data, the behavior of the proposed CNN architecture can differ when it is trained again. To avoid a selection bias, training was iterated 20 times and averaged results are reported. Each training was performed over 2 epochs of the entire training data. Classification by using the descriptors were conducted on a small subset of training/test data rather than on the entire dataset for time efficiency. The descriptors used for comparison are Signature of Histogram of OrienTation (SHOT), Fast Point Feature Histogram (FPFH), Point Feature Histogram (PFH) and the Ensemble of Shape Functions (ESF) as mentioned in Chapter 2. The descriptors were extracted on sub-sampled keypoints of the given surface point cloud where sub-sampling was done with 1cm leaf-sized voxel filtering. Classification was implemented by an exhaustive search on a database of descriptors extracted from surface point clouds of training data and finding the best matches with the set of descriptors extracted from the given test data. Table 5.3 shows classification accuracies and lead times of both the proposed methods and the descriptor-based methods. The lead times are times spent on classifying a batch of 475 test data. From the table, it is apparent that the proposed methods that use automatically learned features show superior classification performances than other classification methods using hand-crafted features. Furthermore, if provided with the trained network weights (parameters), the proposed neural network classifiers can classify large amount of input data in real-time with the help of the General Purpose GPU (GP-GPU) while other methods using hand-crafted features are unfit for the GP-GPU implementation and require considerable processing time.

In addition, it can be seen that CNNs with pre-trained weight initializations outperform the randomly initialized CNN in all performance indicators as shown in Table 5.4. Experimental results can be interpreted by the following statement:

Table 5.3: Classification accuracies and classification times of the proposed networks (glorot uniform initialized) and other classifiers trained on hand-crafted features (descriptors).

	Proposed				SHOT	FPFH	PFH	ESF
	Random	DCAE	PCA	ZCA				
Accuracy (%)	92.67	93.33	94.14	92.95	91.77	87.55	89.87	83.54
Time (s)	< 0.1				175	240	6049	15

Table 5.4: Test accuracies, cross-entropy errors and overfitting measures for four initialization methods.

	Glorot Uniform	DCAE Pre-trained	PCA Centers	ZCA Centers
Accuracy (%)	92.67	93.33	94.14	92.95
Loss	0.2825	0.2221	0.1984	0.2713
Overfitting	0.2105	0.1272	0.0835	0.0905

“Pre-trained non-random network parameters are initial points in the parameter space that can eventually converge to local minima that are closer to the global minimum than the local minimum that the initial point of the randomly initialized parameters converges to.”

The overfitting measure is defined as the difference between cross-entropy errors in test phase and in training phase. In Figure 5.1, overall performances of 20 iterations are plotted as box-plots. Above all, initialization with the cluster centers of PCA whitened patches leads to the best performance. As for overfitting, the CNN initialized with the cluster centers of ZCA whitened patches show the second best performance among the four CNNs but classification accuracy and cross-entropy error of it are just a little better than those of the randomly initialized CNN.

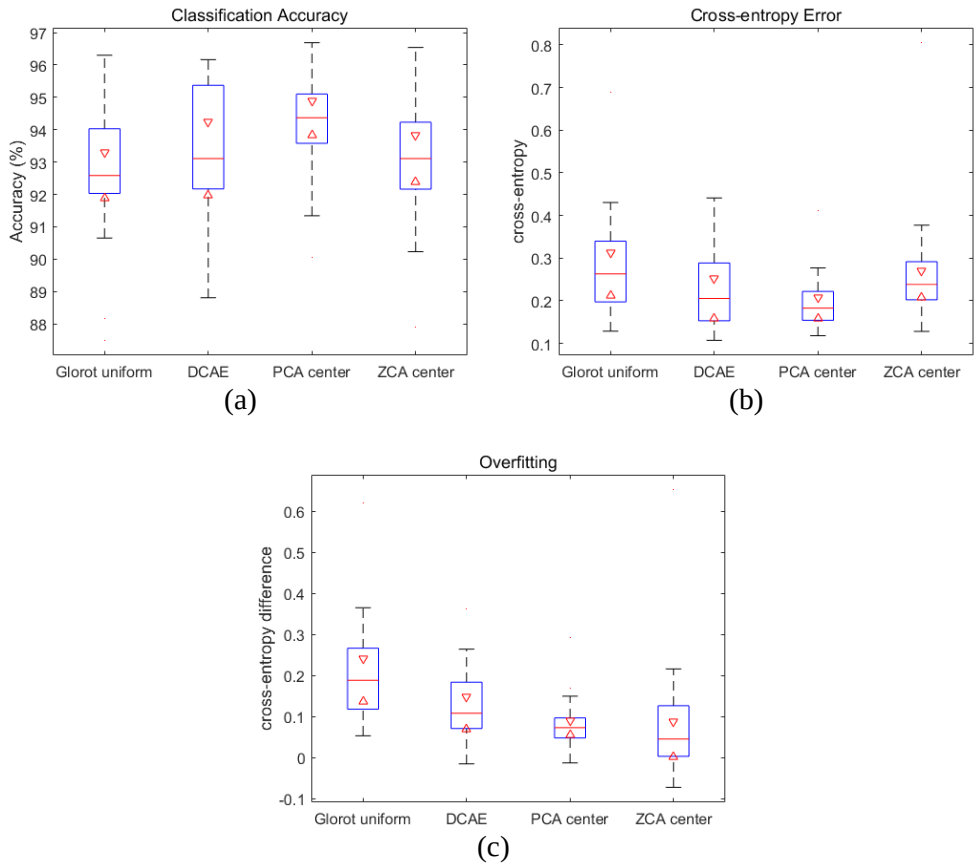
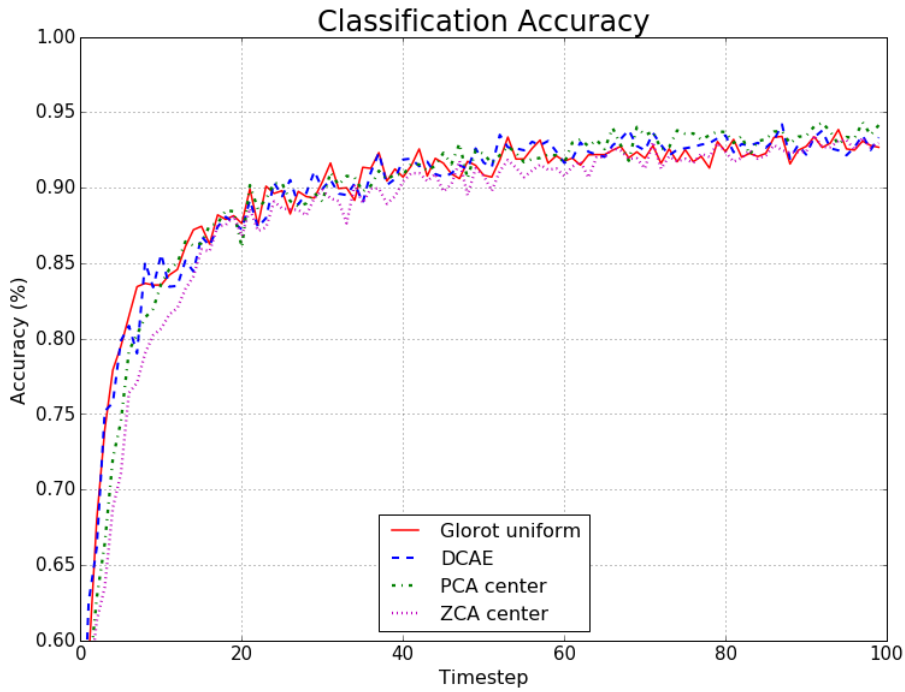
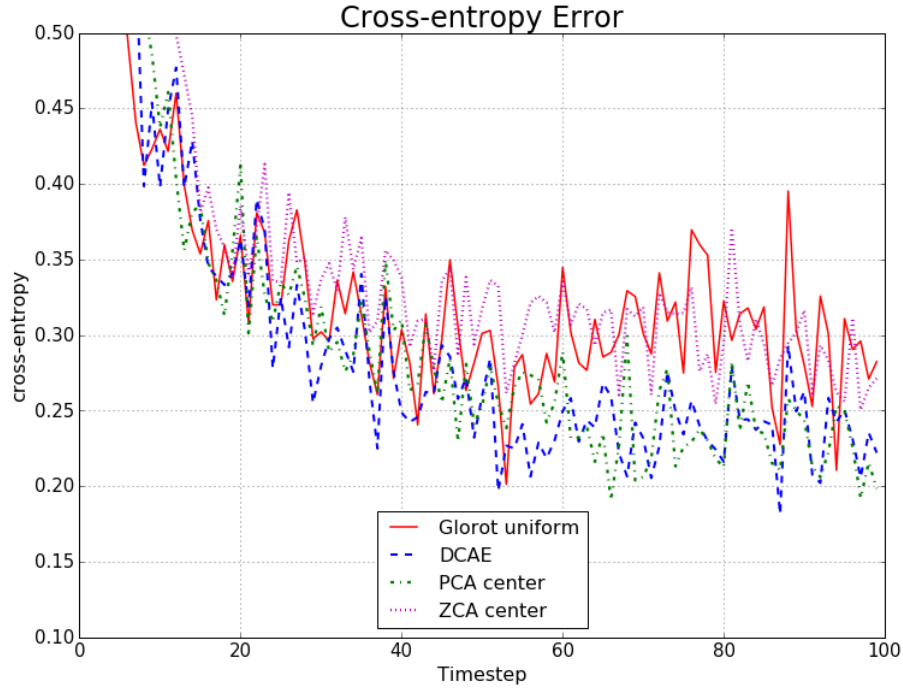


Figure 5.1: Performance evaluation for four different initialization methods. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme inlier data points. Notch markers specify comparison intervals. (a) Classification accuracy. (b) Cross-entropy error. (c) Overfitting measure.

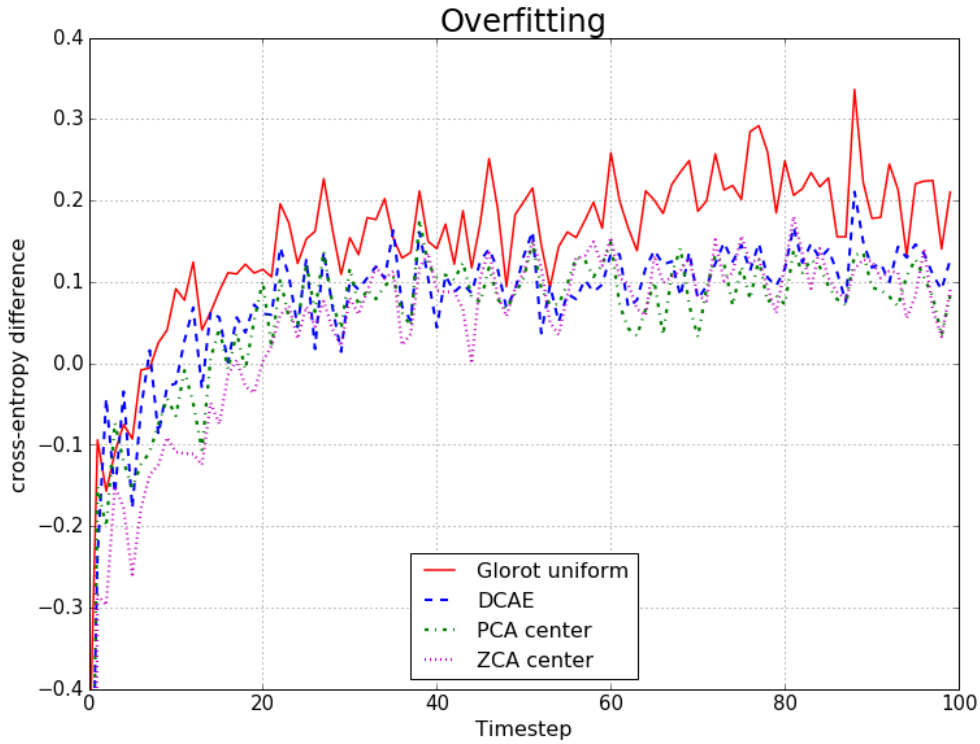
This is only because the performance gap between in training phase and in test phase is insignificant for the case of CNN initialized with cluster centers of ZCA whitened patches. From the results, effect of decorrelation of patch clusters can also be investigated. Obtained cluster centers of correlated patches generally contain redundancy, which effectively reduces the number of parameters of the CNN. This accounts for the performance gap between CNNs initialized with cluster centers of PCA whitened patches and ZCA whitened patches.


Classification Accuracy
Accuracy (%)
Timestep
Glorot uniform
DCAE
PCA center
ZCA center

(a)


Cross-entropy Error
cross-entropy
Timestep
Glorot uniform
DCAE
PCA center
ZCA center

(b)



(c)

Figure 5.2: Performance evaluation for four different initialization methods. Red solid line stands for glorot uniform (random) initialization, blue dotted line for initialization with the pre-trained DCAE, green dotted line for initialization with k-means cluster centers of PCA-whitened patches and magenta line for initialization with k-means cluster centers of ZCA-whitened patches. (a) Classification accuracy. (b) Cross-entropy error. (c) Overfitting measure. (Best Viewed in Color)

Figure 5.2 plots the classification accuracy, cross-entropy error and the overfitting of the four CNNs in time steps. Each plot shows the averages of the 20 iterations. The 2-epoch training/test phases are drawn in 100 time steps.

5.1.3 Performance evaluation of orientation estimation CNN

The assertion that making the neural network to estimate both the class label and the orientation label improves its inference performance is empirically proved.

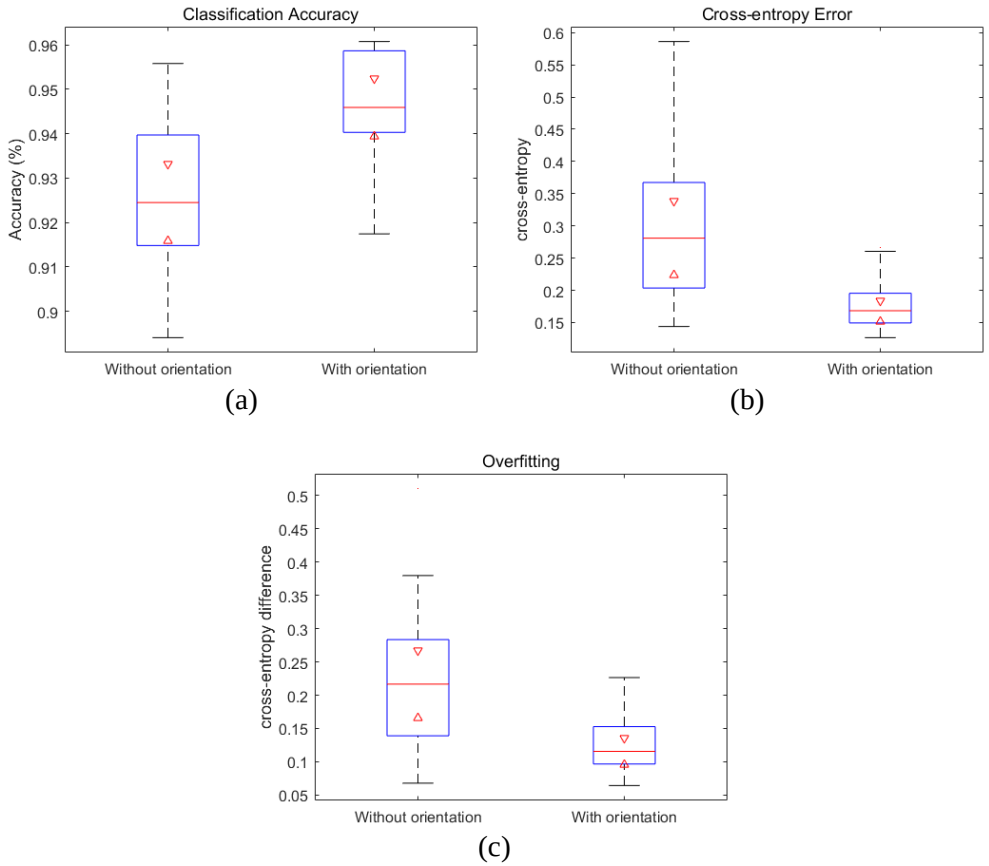


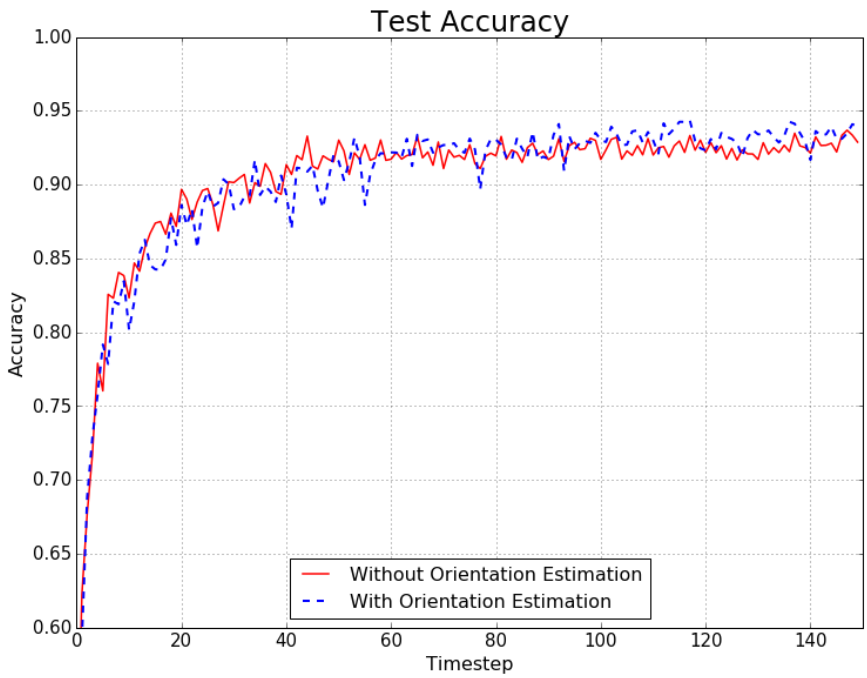
Figure 5.3: Performance evaluation of two networks with/without orientation estimation. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme inlier data points. Notch markers specify comparison intervals. (a) Classification accuracy. (b) Cross-entropy error. (c) Overfitting measure.

The experiment was conducted as follows. Two randomly initialized CNNs having the same structure as those used in preceding experiments were trained on training data of 30546 surface point clouds and tested by 6674 test data. Training was iterated 20 times for both CNNs and the averaged performance is reported. Each training was performed over 3 epochs of the training data. One CNN estimates only the class label and it is given training/test dataset that comprise only input shape and the class pairs $(\mathbf{x}, \mathbf{c})$. The other CNN estimates both the class label and the orientation label. Thus, it is given training/test dataset that are made up of

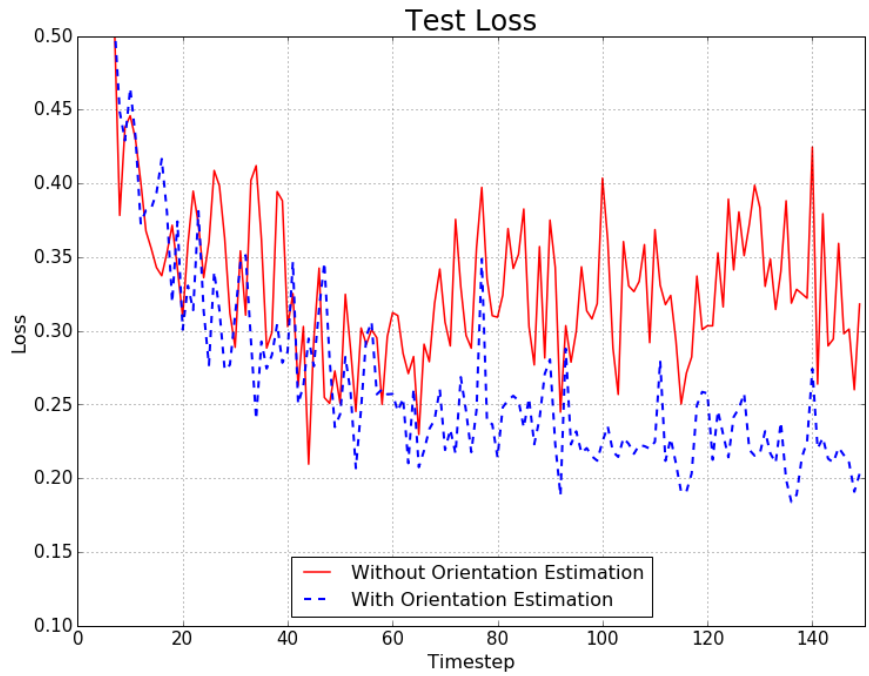
the input shape and corresponding class and orientation pairs ($\mathbf{x}, \mathbf{c}, \mathbf{o}$). The second CNN then can define two separate loss functions, one for the class and the other for the orientation label.

Figure 5.3 shows box plots of classification accuracy, cross-entropy error and overfitting of the two CNNs with and without orientation estimation. Figure 5.4 shows each performance indicator for the two CNNs in time steps. Each graph is the average of the 20 iterations of training and the 3-epoch training/test phases are drawn in 150 time steps. Confusion matrices for the two CNNs are visualized in Figure 5.5.

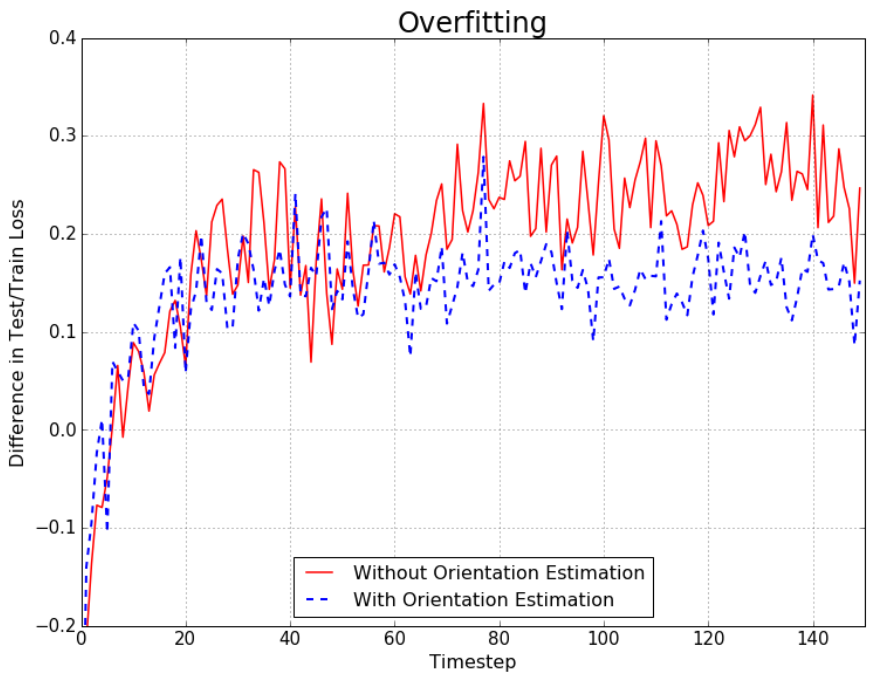
All simulations for classifications were implemented with Python 2.7 and Keras deep learning library [51].



(a)

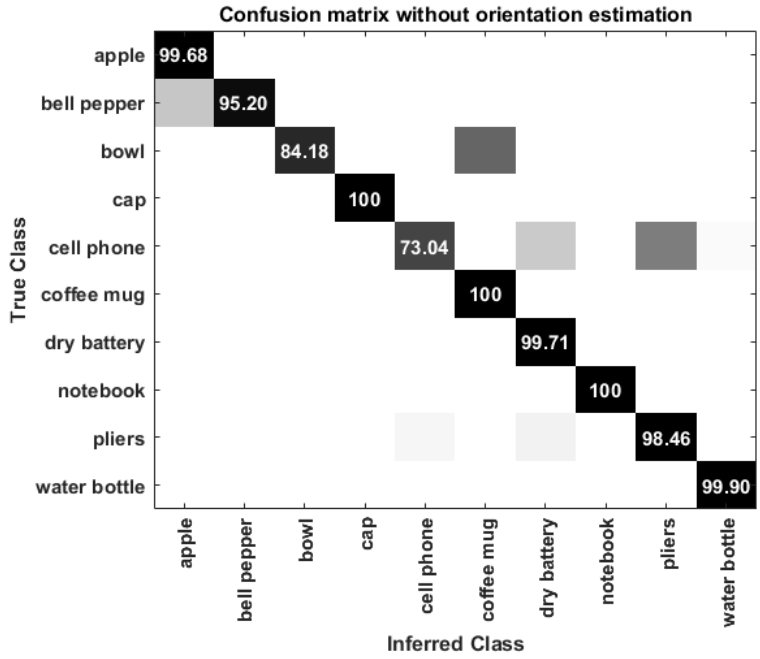


(b)

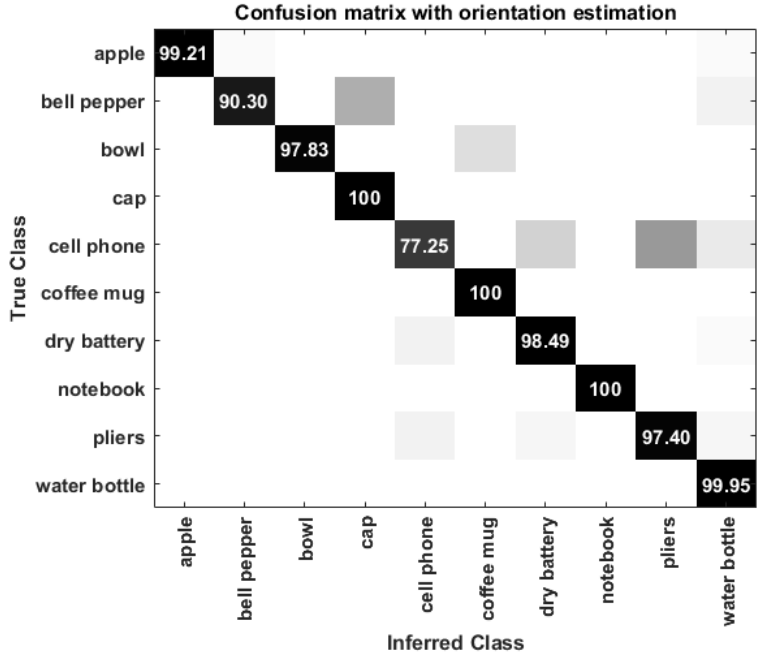


(c)

Figure 5.4: Performance evaluation of two networks with/without orientation estimation. The red solid line represents the performance of the network without orientation estimation and the blue dotted line represents the performance of the network with orientation estimation. (a) Classification accuracy. (b) Cross-entropy error. (c) Overfitting measure. (Best Viewed in Color)



(a)



(b)

Figure 5.5: Confusion matrices for classification results. (a) Classification accuracy. (b) Cross-entropy error. (c) Overfitting measure.

5.2 Shape Retrieval

In this Section, shape retrieval performance of the generic ICP modeling and the proposed method are assessed. The ICP modeling method estimates rigid transformations between successive frames by only ICP registration. ICP registration in this case works with no prior information of initial transformations in accordance with the assumption that odometry information is unreliable outdoors. Two criteria are presented: modeling accuracy and lead time. The simulation was implemented with C++ and the Point Cloud Library (PCL) [52].

Even though public dataset of Washington is used to test the classification performance, shape retrieval is tested on other point cloud data collected by the author. The main reason for this is that there is no evaluating the modeling a locations and orientations of the sensor.

Our dataset comprises five object instances from our laboratory and it contains ground truth locations and orientations of every camera viewpoint. Surface point clouds of these instances are collected in a similar way that those of Washington dataset were collected, but only one viewpoint height is considered. (See Figure 3.3.) Five instances and some of their surfaces are displayed in Figure 5.6 and Figure 5.7.

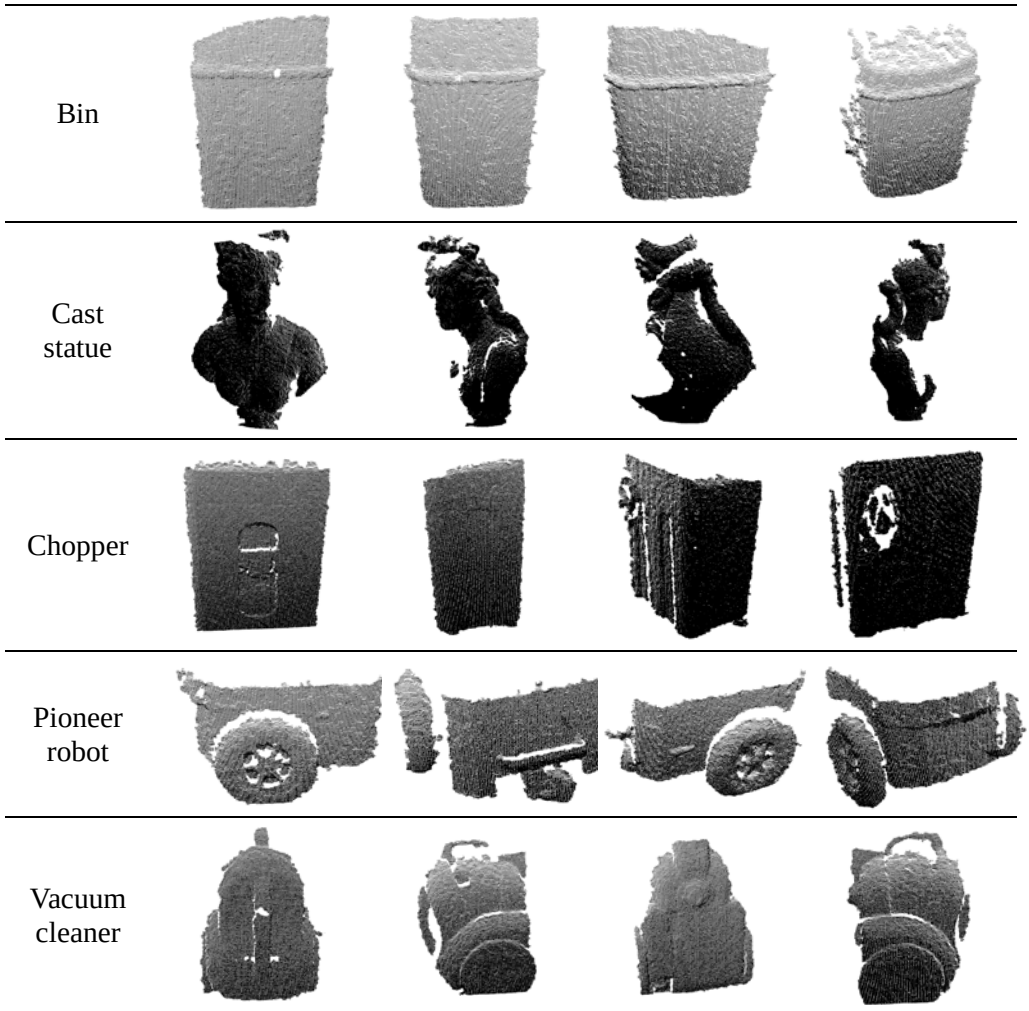


Figure 5.6: Five object instances and four of their surface point clouds. The instances are bin, cast, chopper, pioneer robot and vacuum cleaner.

Figure 5.7: Five object instances and four of their surface point clouds visualized with RGB-valued points. (Best Viewed in Color)

Every instance is uniformly captured by about 250~270 surface point clouds. This translates into a single instance being captured at 250~270 viewpoints on the viewing circle with the angle between contiguous viewpoints differing by about 1.3~1.4 degrees.

Full object shape can be more accurately modeled by using all surface point clouds but this requires significant amount of computation time and large memory to save the data. Thus, it is desirable to select reasonable amount of data so that both modeling accuracy and execution time do not compromise. In this consideration, about 17~18 equally spaced surface point clouds among all surfaces are used for registration. Each surface point cloud is described with respect to its own coordinate system. Viewpoints that capture these frames differ in angle by about 20 degrees on the viewing circle.

5.2.1 Modeling accuracy

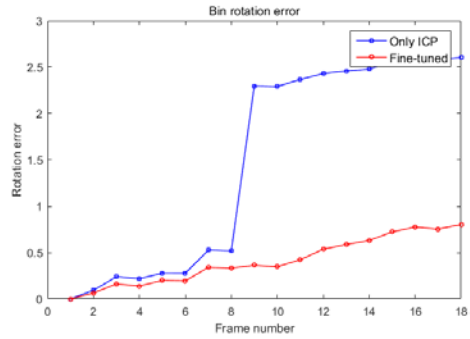
Modeling accuracy is assessed by two separate indices. One is the translation error and the other is the rotation error. Both errors are measured for every frame of surface point clouds. Since the ground truth location and orientation of each viewpoint are known, correct rigid transformation that changes the description of each frame to the target frame can be obtained. Accordingly, errors can be defined with respect to the correct ground truth transformations. Ground truth transformation that transforms frame {i} into target frame {0} is denoted as ${}^{0}_{i}T_{grd}$. The estimated transformation is denoted as ${}^{0}_{i}T_{est}$ where ${}^{0}_{i}T_{est} = \Pi_{k=0}^{i}({}^{k}_{k+1}T)$ and ${}^{k}_{k+1}T$ is obtained by the proposed RANSAC algorithm.

For frame {i}, translation error and rotation error are defined respectively as

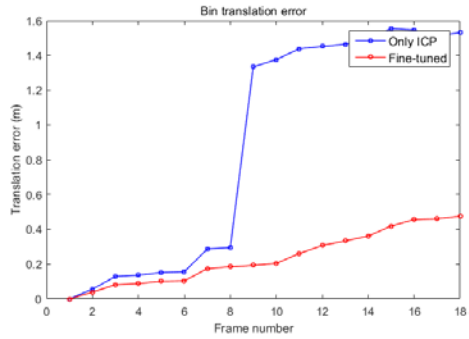
$$\text{Translation Error} = \left\| {}^{0}_{i}p_{grd} - {}^{0}_{i}p_{est} \right\| \tag{5.1}$$

$$\text{Rotation Error} = \left\| I - {}^{0}_{i}R_{grd}({}^{0}_{i}R_{est})^{t} \right\|_{F} \tag{5.2}$$

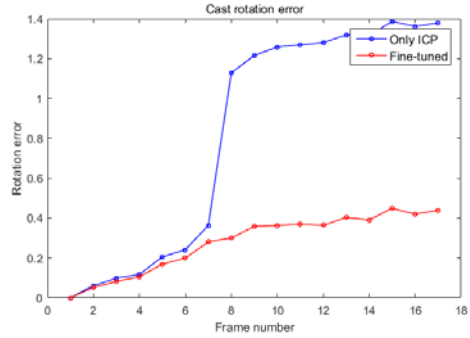
where ${}^{0}_{i}p_{grd}$, ${}^{0}_{i}R_{grd}$ are the translation and the rotation part of ${}^{0}_{i}T_{grd}$ and ${}^{0}_{i}p_{est}$, ${}^{0}_{i}R_{est}$ are the translation and the rotation part of ${}^{0}_{i}T_{est}$. Also, ${}^{0}_{i}p_{grd}$ denotes ${}^{0}p_{i\,ORG}$ of the ground truth translation and ${}^{0}_{i}p_{est}$ denotes ${}^{0}p_{i\,ORG}$ of estimated the translation. The I in (5.2) is the 3×3 identity matrix and the subscript F denotes the Frobenius norm. The definition of the rotation error is a metric on the group $SO(3)$ and is called as chordal distance. Since there are at most 18 multiplicative operations involving rotation matrices, round-off errors that can break the orthogonality of a rotation matrix are not influential.


Bin rotation error
Only ICP
Fine-tuned
Rotation error
Frame number

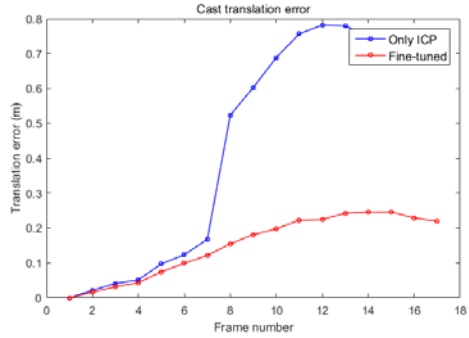
(a)


Bin translation error
Only ICP
Fine-tuned
Translation error (m)
Frame number

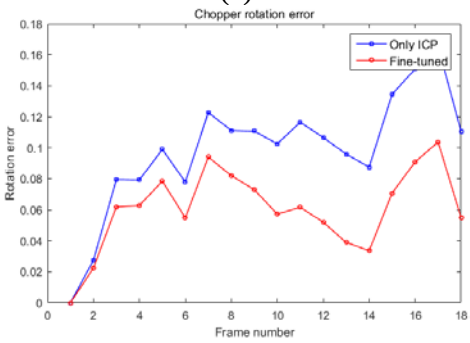
(b)


Cast rotation error
Only ICP
Fine-tuned
Rotation error
Frame number

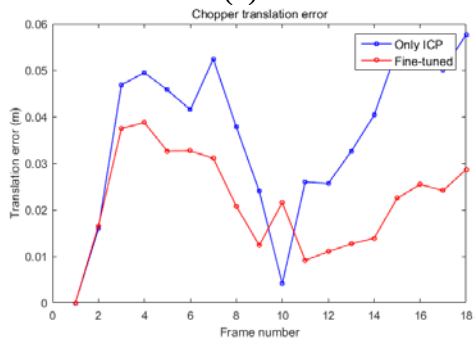
(c)


Cast translation error
Only ICP
Fine-tuned
Translation error (m)
Frame number

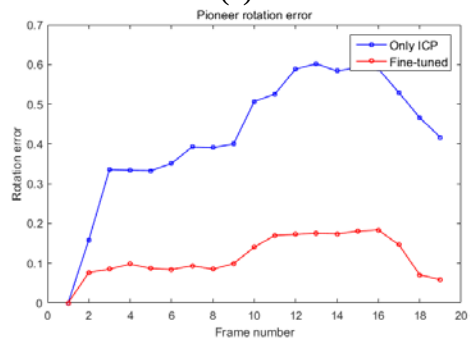
(d)


Chopper rotation error
Only ICP
Fine-tuned
Rotation error
Frame number

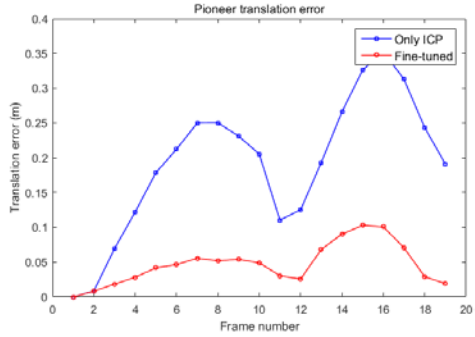
(e)


Chopper translation error
Only ICP
Fine-tuned
Translation error (m)
Frame number

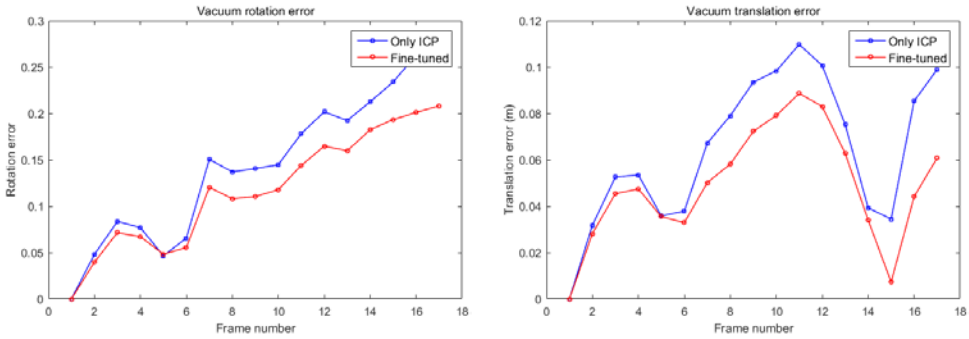
(f)


Pioneer rotation error
Only ICP
Fine-tuned
Rotation error
Frame number

(g)


Pioneer translation error
Only ICP
Fine-tuned
Translation error (m)
Frame number

(h)



(i) (j)

Figure 5.8: Rotation errors (left column) and translation errors (right column) for shape retrieval of five object instances. Line plots in blue are errors of generic ICP algorithms and lines in red correspond to errors of the proposed method (a), (b) Bin modeling errors. (c), (d) Cast statue modeling errors. (e), (f) Chopper modeling errors. (g), (h) Pioneer robot modeling errors. (i), (j) Vacuum cleaner modeling errors.

In Figure 5.8, rotation/translation errors for all modeling results tell that the proposed method surpasses the ICP modeling by initially aligning two point clouds before fine-tuning. In modeling 'Bin' and 'Cast statue' the ICP without initial alignment failed to adequately register point clouds at some frame and it ended in large final errors. In general, both the translation and the rotation errors tend to increase as the number of frame goes up because small errors caused by individual rigid transformation estimation accumulate. The weakness of the proposed error measure can also be observed. The dependence of errors on the rigid transformation matrices caused sporadic troughs in the error plots. That is, some errors of individual transformation (rigid transformation between neighboring point clouds) estimation sometimes lead the accumulated error in the direction of correct ground truth transformation. However, it is practically very difficult to measure the physical error metric with the real volumetric object and modeled point cloud data.

Volumetric models retrieved by both methods are visualized in Figure 5.9.

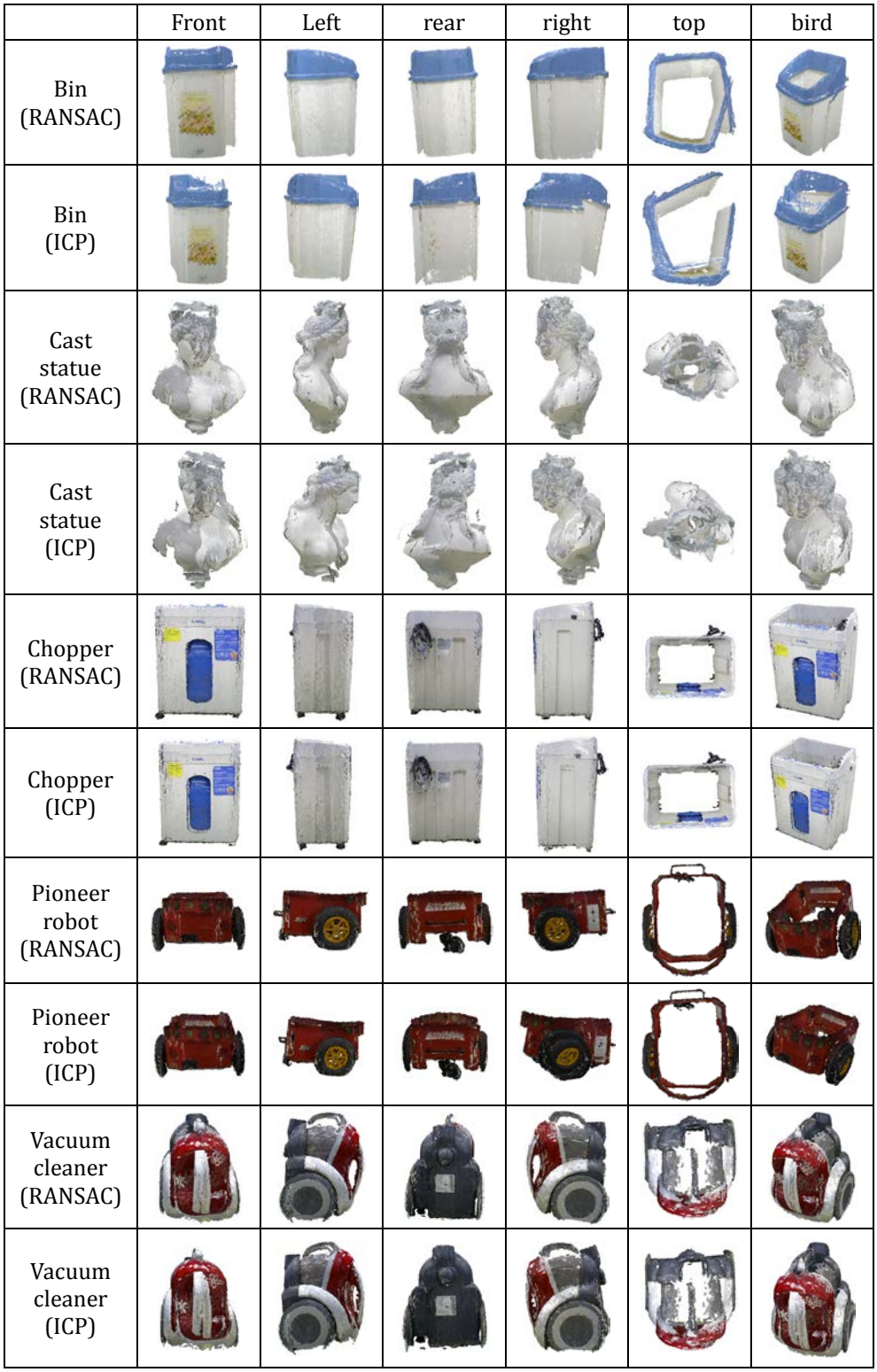


Figure 5.9: Volumetric object instances at top view (leftmost), two lateral views, front/rear views and bird's view (rightmost). The points are visualized with additional RGB values for better visualization. (Best Viewed in Color)

5.2.2 Speed

Processing of input data is especially important for robots to operate in real-time. Total times spent in retrieving shapes of each object instance are reported in Figure 5.10. Name of objects are expressed as ‘Bin’, ‘Cast’, ‘Chopper’, ‘Pioneer’ and ‘Vacuum’ for short. Each bar comprises two stacks, registration time on the bottom and statistical outlier removal time on the top. Our proposed method proves very effective in the reducing the registration time. In contrast to ICP modeling that works with most of the points in the point clouds, the proposed RANSAC based algorithm operates with only extracted keypoints and the descriptors defined on them. Although the number of keypoints of a point cloud gets larger as the size of the point cloud increases, they are still manageable compared to the number of total points in the point cloud. Furthermore, these keypoints are sorted out by similarity measures between attached descriptors before the RANSAC process. The correspondence search is boosted by kd-tree search algorithm as in the case of ICP correspondence search. In the result, retrieving the shape of ‘Chopper’ took the longest time. This is because the number of points in each surface point cloud of this instance amounts to 90,000 ~ 100,000. On the other hand, the number of points in each surface point cloud of other instances averages around 30,000. From this observation, it can be asserted that the ICP algorithm with no prior information on the initial transformation is vulnerable to large point clouds in which case where its computation time get excessive.

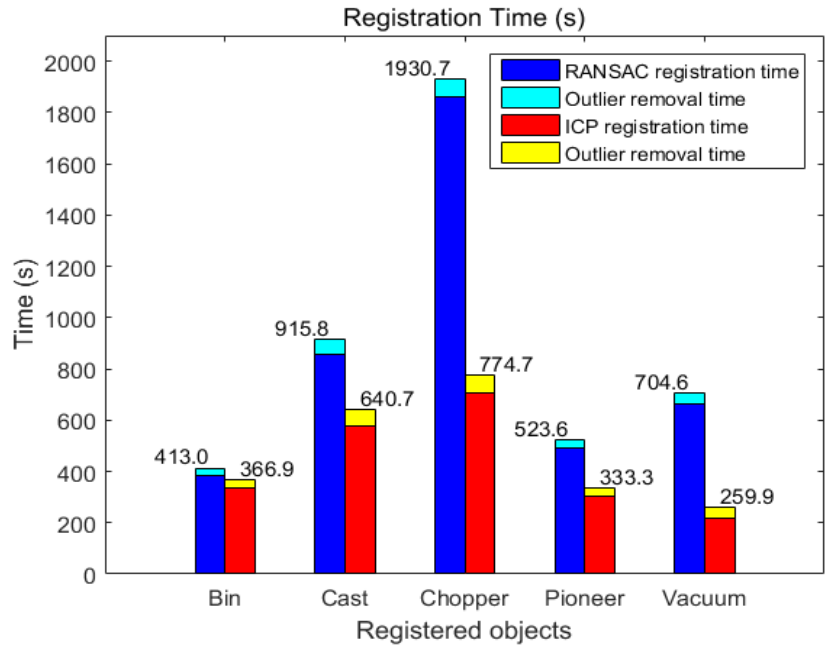


Figure 5.10: Shape retrieval times for five object instances. Modeling time with ICP modeling is colored in blue and cyan where blue portion represents the time spent in registration and cyan portion is the time spent in statistical outlier removal. Modeling time with the proposed method is colored in red and yellow where red portion stands for the time spent in registration and yellow portion represents the time spent in statistical outlier removal. (Best Viewed in Color)

Chapter 6

Conclusion

In this thesis, an object classification method using object's single-view surface point cloud is proposed. In addition, an effective RANSAC based method that registers multiple single-view point clouds into a volumetric point cloud is suggested. The proposed learning architecture automatically learns hierarchical features from training data and is shown to significantly outperform existing classification methods in terms of classification accuracy and processing time. Furthermore, three different weight initialization methods – initialization with the pre-trained de-noising convolutional auto-encoder (DCAE), initialization with k-means cluster centers of PCA/ZCA whitened patches - are adopted to reduce the overfitting problem. Evaluation results tell that initializations with the cluster centers of PCA whitened patches and with the encoder stacks of a pre-trained DCAE reduce overfitting to a certain degree, while initialization with the cluster centers of ZCA whitened patches produces slight reduction of overfitting. It is also shown that the classification performance can be improved by making the neural network estimate both the class and the orientation of the object simultaneously.

The validity of this relevant parameter estimation is proved by evaluation results.

A RANSAC based rigid transformation estimation that uses SHOT descriptor on top of ISS keypoints is suggested. It is shown that the proposed rigid transformation method can provide a reliable initial transformation to the generic ICP algorithm thereby bringing robustness and speed to shape modeling.

For future works, making the CNN architecture accept inputs of variable sizes can be considered. This will further open the possibility of voxelizing the surface point cloud into an irregular, variable-shaped voxel grid whose divisions are different for each side.

The proposed methods can be utilized in an outdoor scenario where a robot navigates in an unknown environment while collecting landmark information. If the robot is equipped with the trained neural network weights, then it can recognize object category in real-time manner. Furthermore, it can also decide whether to circle around the object to obtain a full volumetric object model as a reliable map feature.

Reference

[1] Y. LeCun, et al., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, pp. 541-551, 1989.

[2] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.

[3] D. Erhan, et al., "Why Does Unsupervised Pre-training Help Deep Learning?," *Journal of Machine Learning Research*, pp. 625-660, Feb. 2010.

[4] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, 1992.

[5] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.

[6] K. Klasing, D. Wollherr, and M. Buss, "Realtime Segmentation of Range Data Using Continuous Nearest Neighbors," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 2431-2436.

[7] F. Pauling, M. Bosse, and R. Zlot, "Automatic Segmentation of 3D Laser Point Clouds by Ellipsoidal Region Growing," in *Australasian Conference on Robotics and Automation*, vol. Vol. 10, 2009.

[8] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3D Lidar Data in Non-flat Urban Environments Using a Local Convexity Criterion," in *2009 IEEE Intelligent*

Vehicles Symposium, Xi'an, China, 2009.

[9] A. E. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433-449, 1999.

[10] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. Vol. 1, 2005, pp. 886-893.

[11] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[12] F. Tombari, S. Salti, and L. Di Stefano, "Unique Signatures of Histograms For Local Surface Description," in *European Conference on Computer Vision*, Berlin Heidelberg, 2010, pp. 356-369.

[13] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration," in *2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[14] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent Point Feature Histograms for 3D Point Clouds," in *Proceedings of 10th International Conference on Intelligent Autonomous Systems*, Baden-Baden, Germany, 2008, pp. 119-128.

[15] W. Wohlkinger and M. Vincze, "Ensemble of Shape Functions For 3D Object Classification," in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 2987-2992.

[16] L. A. Alexandre, "3D Descriptors for Object and Category Recognition: a Comparative Evaluation," in *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, 2012, pp. 7-12.

[17] K. Lai, L. Bo, X. Ren, and D. Fox, "A Large-Scale Hierarchical Multi-View RGB-D Object Dataset," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1817-1824.

[18] M. Blum, J. T. Springenberg, J. Wulfing, and M. Riedmiller, "A Learned Feature Descriptor for Object Recognition in RGB-D Data," in *2012 IEEE International Conference on Robotics and Automation*, 2012, May, pp. 1298-1303.

[19] L. Bo, X. Ren, and D. Fox, "Learning Hierarchical Sparse Features for RGB-(D) Object Recognition," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 581-599, 2014.

[20] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-Recursive Deep Learning for 3D Object Classification," *Advances in Neural Information Processing Systems*, pp. 665-673, 2012.

[21] A. Coates, H. Lee, and A. Y. Ng, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning," in *14th International Conference on Artificial Intelligence and Statistics*, 2011.

[22] D. Maturana and S. Scherer, "VoxNet : A 3D Convolutional Neural Network for Real-Time Object Recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[23] Z. Wu, et al., "3D Shapenets: A Deep Representation for Volumetric Shapes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912-1920.

[24] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-View Convolutional Neural Networks for 3D Shape Recognition," in *IEEE International Conference on Computer Vision*, 2015, pp. 945-953.

[25] M. D. Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised Feature Learning for Classification of Outdoor 3D Scans," in *Australasian Conference on Robitics and*

Automation, 2013.

[26] M. Greenspan and M. Yurick, "Approximate K-D Tree Search for Efficient ICP," in *Fourth International Conference on 3-D Digital Imaging and Modeling*, Banff, Canada, 2003.

[27] F. Lu and E. Milios, "Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 3, pp. 249-275, 1997.

[28] Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145-155, 1992.

[29] P. Biber and W. Straber, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 2743-2748.

[30] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational Projection Statistics for 3D Local Surface Description and Object Recognition," *International Journal of Computer Vision*, vol. 105, no. 1, pp. 63-86, 2013.

[31] J. Novatnack and K. Nishino, "Scale-Dependent/Invariant Local 3D Shape Descriptors for Fully Automatic Registration of Multiple Sets of Range Images," in *European Conference on Computer Vision*, Berlin Heidelberg, 2008, pp. 440-453.

[32] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient Simplification of Point-Sampled Surfaces," in *Proceedings of the Conference on Visualization*, 2002, pp. 163-170.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[34] M. D. Richard and R. P. Lippmann, "Neural Network Classifiers Estimate Bayesian A Posteriori Probabilities," *Neural Computatiton*, vol. 3, no. 4, pp. 461-483, 1991.

[35] Y. N. Chen, C. C. Han, and C. T. Wang, "The Application of a Convolution Neural

Network on Face and License Plate Detection," in *18th International Conference on Pattern Recognition*, 2006, pp. 552-555.

[36] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face Recognition: A Convolutional Neural-Network Approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98-113, 1997.

[37] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221-231, 2013.

[38] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281-297.

[39] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B*, pp. 1-38, 1977.

[40] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual Categorization with Bags of Keypoints," in *Workshop on Statistical Learning in Computer Vision*, vol. 1, 2004.

[41] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169-2178.

[42] A. Coates and A. Y. Ng, "Selecting Receptive Fields in Deep Networks," *Advances in Neural Information Processing Systems*, pp. 2528-2536, 2011.

[43] A. Coates and A. Y. Ng, "Learning Feature Representations with K-Means," *Neural Networks: Tricks of the Trade*, pp. 561-580, 2012.

[44] Y. Zhong, "Intrinsic Shape Signatures: A Shape Descriptor for 3D Object

Recognition," in *2009 IEEE 12th International Conference on Computer Vision Workshops*, 2009, pp. 689-696.

[45] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371-3408, 2010.

[46] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," in *International Conference on Artificial Neural Networks*, Berlin Heidelberg, 2011, June, pp. 52-59.

[47] R. Linsker, "Self-Organization in A Perceptual Network," *Computer*, vol. 21, no. 3, pp. 105-117, 1988.

[48] B. K. P. Horn, "Closed-form Solution of Absolute Orientation Using Unit Quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 1127-1135, 1987.

[49] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, pp. 698-700, 1987.

[50] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249-256.

[51] F. Chollet. (2015) https://github.com/fchollet/keras.

[52] R. B. Rusu and S. Cousins, "3D Is Here : Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1-4.

초 록

본 논문은 물체의 한 면에 대한 3차원 점 군(point cloud) 데이터가 주어졌을 때 이를 기계 학습 기반으로 어떤 물체인지 분류하는 물체 분류 (object classification) 방법을 제안한다. 또한 물체의 여러 면에 대한 점 군이 주어졌을 때 이를 정합하여 물체의 형상을 복원하는 기법을 제시한다.

기존의 3차원 점 군 기반의 물체 분류 방법은 주로 사용자가 디자인한 특징량(hand-crafted feature)에 의존하였다. 하지만, 디자인된 특징량은 원 자료(raw data)를 효과적으로 사용할 수 없고 대개 다른 특징량에 다시 의존하기 때문에 확장성이 떨어진다는 단점을 가지고 있다. 본 논문에서는 이러한 문제를 피하기 위하여 박셀 격자화된 점 군으로부터 학습된 특징량을 추출하여 물체 분류에 사용하고 이를 위하여 Convolutional Neural Network(CNN)를 기본 단위로 가지는 깊은 학습 구조를 제시한다. 깊은 학습 구조는 대개 주어진 데이터에 포함된 노이즈(noise)까지 학습하게 되는 과적합 문제를 지니는데 본 논문에서는 과적합을 완화하기 위해 두 가지 작위적인 가중치(weight) 초기화 방식이 제시된다. 첫 번째 방법은 선행 학습된 de-noising Convolutional Auto-Encoder의 인코딩 스택으로 초기화하는 방법이고 두 번째 방법은 입력 데이터의 작은 패치들의 집합을 k-평균 군집화하여 얻은 군집 중심으로 초기화하는 방법이다. 정량적인 분석을 통하여 제시된 초기화 방식이 과적합 문제를 완화함을 보인다. 추가적으로, 제안된 인공 신경망은 물체의 범주뿐만 아니라 물체 표면의 방향(orientation)을 동시에 추정하도록 고안된다. 인공 신경망에게 이러한 추가적인 임무를 부여함으로써 물체 분류 성능이 더욱 높아짐을 확인할 수 있었다.

여러 개의 물체 단면 점 군을 정합하여 모델링을 수행하기 위해서 본 논문에서는 Signature of histogram of orientation(SHOT) 기술자(descriptor)와 Random Sample Consensus(RANSAC) 알고리즘을 이용한 정합 기법이 제시된다. 제시된 RANSAC 기반의 방법을 통하여 연속된 두 개의 점 군 사이 강체 변환을 추정하고 이를 Iterative Closest Point(ICP) 알고리즘으로 미세 조정하였을 때, 신뢰성 있는 물체 형상 복원이 가능함을 보인다.

주요어 : 물체 분류(Object classification), Convolutional neural network, 비지도 선행학습(Unsupervised pre-training), 형상 복원(Shape retrieval), RANSAC
학 번 : 2015-20909