DEEP LEARNING FOR COMPUTER VISION AND ITS APPLICATION TO MACHINE PERCEPTION OF HAND AND OBJECT

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Sangpil Kim

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF DISSERTATION APPROVAL

Dedicated to my family.

ACKNOWLEDGMENTS

Most of all, I would like to thank my advisor, Professor Karthik Ramani for his insightful advices and dedicated support during my Ph.D. work. He supported my research works all the time and introduced many great people for getting advices. I am truly grateful for sharing his knowledge in depth. I would also like to acknowledge my Ph.D. committee members, Professor Mireille Boutin, Professor Guang Lin, and Professor Alex Quinn.

I would like to thank my friends and colleagues in the C-Design lab and e-Lab from the past to the present: in particular, Hyung-gun Chi, Jonghoon Jin, Xiao Hu, Nick Winovich, Anirudh Vegesana, Weizhi Li, Brian Lee, Andre Chang, Ananya Ipsita, and Hyojin Do for their valuable feedback and help in my research. Additionally, I would like to thank Professor Eugenio Culurciello and Professor Qixing Huang for guidance of my research. Especially, I appreciate Professor Eugenio Culurciello for mentoring me in the early stage of my Ph.D. work.

Last but not least, I would like to thank my father, Hwa Yeong Kim and mother, Jong Suk Lee for their advices and support. They gave me sincere help when it was needed and always encouraged me whenever I needed. I also would like to acknowledge my brother, Sang Yun Kim to whom I always asked general advices.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

SYMBOLS

$\mathcal{D}$	discriminator
D	decoder
R	refiner
H	mean curvature
isa	intersected surface area in voxel grids
p_d	data distribution
p_z	prior distribution of the decoder
$\mathbf{v}$	the Boolean voxels of the true data distribution
μ	mean of the latent vector from the encoder
ϵ	covariance of the latent vector from the encoder
N	normal distribution
$\odot$	an element-wise multiplication
$\mathbf{z}$	a noise vector from $\mu + r \odot exp(\epsilon)$
$\mathcal{T}$	transformation function from a LWIR plane to a RGB plane
$\boldsymbol{K_D}$	intrinsic parameter matrix of depth camera
$\boldsymbol{K_T}$	intrinsic parameter matrix of thermal camera
$\boldsymbol{T}$	translation matrix of a camera
$\boldsymbol{R}$	3D rotation matrix of a camera

ABBREVIATIONS

SSIM	structure similarity image metric
GAN	generative adversarial network
VAE	variational Auto-Encoder
AR	augmented reality
VR	virtual reality
CNN	convolutional neural network
HCI	human computer interaction
IK	inverse kinematics
LTNN	latent transformation neural network
MC	mean curvature
ISA	intersected surface area
IV	interior volumetric
MCB	mechanical components benchmark
LWIR	long-wave infrared
IMU	inertial measurement unit
TM	thermal mask
AABB	axis-aligned bounding box
OMBB	oriented minimum bounding box

ABSTRACT

Sangpil, Kim Ph.D., Purdue University, December 2020. Deep Learning for Computer Vision and its Application to Machine Perception of Hand and Object. Major Professor: Karthik Ramani.

The advances in computing power and artificial intelligence have made applications such as augmented reality/virtual reality (AR/VR) and smart factories possible. In smart factories, robots interact with workers and, AR/VR devices are used for skill transfer. In order to enable these types of applications, a computer needs to recognize the users hand and body movement with objects and their interactions. In this regard, machine perception of hands and objects is the first step for human and computer integration. This is because personal activity is represented by the interaction of objects and hands. For machine perception of objects and hands, vision sensors are widely used in a wide range of industrial applications since visual information provides non-contact input signals. For these reasons, computer vision-oriented machine perception has been researched extensively. However, due to the complexity of object space and hand movement, machine perception of hands and objects remains a challenging problem.

Recently, deep learning has been introduced with groundbreaking results in the computer vision domain, which address many challenging problems and significantly improves the performance of AI in many tasks. The success of deep learning algorithms depends on the learning strategy and the quality and quantity of the training data. Therefore, in this thesis, we tackle machine perception of hands and objects with four aspects: learning underlying structure of 2D data, fusing surface and volume content of a 3D object, developing an annotation tool for mechanical components, and using thermal information of bare hands. More broadly, we improve the machine perception of interacting hand and object by developing a learning strategy and framework for large-scale dataset creation.

For the learning strategy, we use a conditional generative model, which learns conditional distribution of the dataset by minimizing the gap between data distribution and the model distribution for hands and objects. First, we propose an efficient conditional generative model for 2D images that can traverse the latent space given a conditional vector. Subsequently, we develop a conditional generative model for 3D space that fuses volume and surface representations and learns the association of functional parts. These methods improve machine perception of objects and hands for not only 2D images but also in 3D space. However, the performance of deep learning algorithms has positive correlation with the quality and quantity of datasets, which motivates us to develop the a large-scale dataset creation framework.

In order to leverage the learning strategies of deep learning algorithms, we develop annotation tools that can establish a large-scale dataset for objects and hands and evaluate existing deep learning methods with extensive performance analysis. For the object dataset creation, we establish a taxonomy of mechanical components and a web-based annotation tool. With this framework, we create a large-scale mechanical components dataset. With the dataset, we benchmark seven different machine perception algorithms for 3D objects. For hand annotation, we propose a novel data curation method for pixel-wise hand segmentation dataset creation, which uses thermal information and hand geometry to identify and segment the hands from objects and backgrounds. Also, we introduce a data fusion method that fuses thermal information and RGB-D data for the machine perception of hands while interacting with objects.

1. INTRODUCTION

Advancement of deep learning in computer vision greatly increased the performance of machine perception of objects and hands and the interest in computer vision applications in many diverse areas such as a smart factory, AR/VR devices for skill transfer, and robotics. In particular, human computer integration, machine perception of objects and hands with visual signal which is contactless is crucial because it is low cost and easy to move around. The development of vision-based machine perception of objects and hands is a crucial research topic in the computer vision literature and industry. Therefore, the goal of this thesis is improving visual machine perception of objects and hands based on deep learning.

1.1 Inspiration

Everyday people interact with outside of the world with their hands which is one of the most complex body part, and people use eyes to recognize environments. From this natural behavior of human, machine perception of objects and hands with computer vision is widely studied in the literature and industry. Advancement of computation power enables the usage of deep neural networks in real time applications in three-dimensional space for AR/VR applications. Additionally, the improvement of vision sensors, a low-cost depth camera, and a thermal sensor has led us to develop data fusion method of multi-source visual signals for deep neural networks.

Even though the success of the deep learning significantly improved the performance of objects and hands recognition, it is excessively challenging problems because of their complicated structure and curse of dimensionality as well as the infinite kinds of object space. Therefore, modeling loss functions for training the deep neural networks and designing efficient structure of the neural networks have become extremely-important tasks to overcome these problems. Although the extensive research works have been conducted in the last decade including utilizing multi-modal information of visual sensors and developing robust perception algorithms for the objects and hands, there are no ultimate solutions for the problem.

1.2 Research Aims and Objectives

The research goal is developing a robust deep learning based vision algorithms for hand and object recognition. To achieve the research goal, we split the research area in two parts: developing learning strategies and dataset curation. For improving learning strategies, we develop conditional generative models which learn the underline data structure of objects for distinguishing an object from others. To improve the efficiency of the model, we make the model to be fully-convolutional and fused surface and volumetric representations of objects. The other important thing is the dataset for deep learning which is a data driven method, and the performance is upper bounded by the size and quality of the dataset. Therefore, we develop robust data curation methods for both rigid and non-rigid objects. For rigid object, we establish a large-scale annotated mechanical components dataset by establishing mechanical components taxonomy and web-based interactive annotator. With this dataset we benchmark seven different deep learning oriented shape descriptors by measuring performances of classification and retrieval tasks. From the benchmark experiment, orientation invariant property is critical for mechanical components in both retrieval and classification task. For non-rigid object dataset curation, We develop labeling pipeline for pixel-wise hand segmentation in first-person view videos. We use body temperature to narrow down the search space in the scene and refine the labels with hand geometry. For using hand geometry, we use deep learning based tracking algorithms which use hand geometry. Furthermore, we collect sequences of thermal, depth, and color images in single shot. With this multi-modal videos, we introduce a data fusion method that fuses thermal, depth, and color information spaces for machine perception of hands, resulting significant improvement of machine perception of hand while interacting with complex objects.

1.2.1 Learning strategies

The generative model is adequate for representing complex, structured datasets and generating realistic samples from underlying data distributions [1, 2]. Therefore, the generative model can sample syntactic data within parametric distribution, which should be ideally identical with the data distribution. Extension of the generative model is a conditional generative model that can sample objects from conditional distribution in order to allow selectivity of generated objects [3,4]. We use conditional generative models for synthesizing novel views of objects from a reference view, which has a variety of practical applications in computer vision, graphics, and robotics. We propose a fully-convolutional conditional generative model, which is much faster than the conventional conditional generative model.

Many works [5, 6] have shown that jointly solving multiple tasks, named multi-task learning [7], helps to improve the generalizability of estimation models. From these observations, multi-task learning is adopted to increase the performance of the generative model. To be more specific, two distinct representations were fused as a multi-task learning approach, which has not been well-examined for learning compact object representations and synthesizing objects. Surface representation imposes boundary and connectivity information of each part of an object [8–10], and volumetric representation determines interior geometry which is used for heat flow calculation [11, 12]. Knowing both surface and volumetric representations are crucial for learning not only a perceptual set of attributes but also the connectivity of each part of the objects and the details of local interior regions, hence reducing defects in synthesized objects. The model learns surface properties to learn the interior volumetric representations of objects and vice-versa [7]. Additionally, the model learns not only the data distribution but also the geometric properties of the objects. By learning the geometry properties, the generative models do not tend to collapse in a specific mode, increasing the sampled objects' diversity.

1.2.2 Dataset curation

The performance of deep learning algorithms depends on the quantity and quality of the training dataset. This is because the deep neural network learns features of objects with stochastic gradient descent method by observing objects iteratively. While iteratively observing the object, the model captures the functionality of objects and distinct shapes. For this reason, a large-scale annotated dataset is required to find an optimal model that can extract features of objects for recognizing multiple objects. For example, there have been extensive works for developing feature extractors for common objects that can be used for object classification and retrieval tasks. However, researches on mechanical components shape descriptors are elusive. This is because there are no large-scale annotated datasets. From this motivation, we establish a large-scale mechanical components dataset by establishing the taxonomy of mechanical components and developing a web-based annotation tool for mechanical components. This dataset is impactable in many areas, in particular smart factories. Nowadays, in smart factories, robots co-work with people, and the AR device helps people in assembling task by overlapping the guidance of instruction in 3D space. Therefore, vision-based perception algorithms for mechanical components are highly demanding.

Data fusion of multiple sensors from multiple sources reduces the uncertainty of the machine perception of hands and objects since diverse information provides more informative features than unimodal information. For example, an RGB camera cannot capture thermal information but can effectively capture the geometry and textures of hands. In contrast, a thermal sensor can capture thermal information but cannot capture color-oriented patterns and textures. We also noticed that human body temperature is bounded in a specific range, which can be used as distinct characteristics to distinguish hands from other objects. With this observation, the thermal information was fused along with RGB images to reduce the model's uncertainty that segments hands in pixel-level from holding objects and backgrounds. Therefore, we propose a framework that is jointly using a thermal camera and an RGB camera. The thermal camera captures the body temperature to restrict the search space in the scene and an RGB image is used to capture the hands' geometry to remove the objects within a similar temperature range of hands. The framework efficiently identifies hands in pixel-level with the first-person view. We further propose a data fusion method with deep neural networks for a pixel-wise hand segmentation task by inputting thermal, RGB, and depth modality into deep neural networks. The proposed data fusion method significantly increases the performance of the model for the pixel-wise hand segmentation task.

1.3 Contributions

The main contributions are summarized as the following:

1. A fully-convolutional generator was introduced by utilizing conditional transformation unit, with a family of modular filter weights, to learn high-level mappings within a low-dimensional latent space.

2. A novel framework was proposed for 3D object view synthesis which separates the generative process into distinct network components dedicated to learning i) coarse pixel value estimates, ii) pixel refinement map, and iii) the global RGB color balance of dataset.

3. The generalizability of 3D object synthesizer was improved by jointly estimating surface and volumetric representations as multi-task learning.

4. A learning method of part geometry was proposed, which improves the fidelity of each part of the synthesized objects. Learning part geometry was done explicitly by optimizing the model which estimates the geometry of each part of objects.

5. A hierarchical taxonomy of mechanical components was established based on the international classification for standards.

6. A large-scale mechanical components benchmark was annotated and collected, seven deep learning-based classifiers were benchmarked, and their performances were analyzed with the dataset.

7. A framework was developed that can significantly reduce segmentation efforts by leveraging hand temperature for creating pixel-wise hand segmentation ground truth when a person is holding tools. The method does not require hand pose labels nor a hand mesh model.

8. Large-scale action videos in a first-person view and pixel-wise hand segmentation labels, which contains LWIR, RGB, depth, and IMU information, was collected. The dataset can be used for hand segmentation research using multiple modalities.

9. The effectiveness of multiple modalities for hand segmentation task with deep neural networks was analyzed, and the optimal combination which is fusing thermal (LWIR), RGB, and depth modalities was found.

1.4 Thesis overview

To cover the details of all above contributions, the rest of the thesis is organized as the following. Chapter two reviews the background information and related works in deep learning area on generative models, 3D object representations, data fusion, and pixel-wise segmentation methods. Then, a conditional generative model as an application for a novel view synthesis is described in chapter three, proposing a novel layer that traverses the latent space and task-dived decoder. In chapter four, the deep learning framework that fuses surface and volumetric representations of the conditional generative model is presented. The 3D object space is further explored in chapter five by establishing a large-scale mechanical components dataset and benchmarking seven deep learning shape descriptors with the dataset. Chapter six shows data fusion method for deep learning to be applied for a pixel-wise hand segmentation while interacting with objects. Finally, in chapter seven, the contributions are summarized, and the future research directions are discussed in detail.

2. BACKGROUND

2.1 Generative Model

Generative models have been widely studied in the field of computer vision. There are two types of well-known generative models: Variational Auto-Encoder (VAE) [13] and Generative Adversarial Network (GAN) [14]. VAE consists of an encoder and decoder. The encoder encodes input information into a low dimensional vector which is perturbed by the Gaussian noise. In GAN, the model is optimized to mimic a data distribution by playing a minimax game to find a Nash equilibrium [15] between the generator and discriminator; the generator is optimized to fool the discriminator by generating realistic data, and the discriminator is optimized to identify faked data from the generator.

Generative models with object priors

Generative models which synthesize 3D objects with object priors use an encoder-decoder structure for generating objects. Achlioptas et al. [16] proposed the encoder-decoder structure model, which learns point cloud representation by minimizing Earth Movers Distance and Chamfer Distance with the supervision of objects' class information. However, this method cannot synthesize 3D objects from a noise distribution, and must have reference models. Umetani et al. [17] synthesized deformed quad meshes from reference objects by exploring the manifold of the parameterized mesh surfaces with an encoder-decoder framework. Liu et al. [18] proposed a method that reconstructs user inputted 3D objects by mapping them into the hidden latent space and decoding it. Xie et al. [19] introduced an energy-based model which approximates the 3D shape probability distribution with Markov Chain Monte Carlo methods. Groueix et al. [20] generated the surface of 3D shapes with a generative model from point cloud objects or images. Kalogerakis et al. [21] and Carlson et al. [22] synthesized new 3D objects by reassembling the parts which are retrieved from an existing database with non-parametric approaches. These works require reference objects to synthesize or deform objects. However, our proposed model synthesizes objects without observing objects or images as prior information.

Generative models without object priors

Generative models without object priors use a decoder structure for generating synthetic objects. Wu et al. [23] proposed a generative adversarial loss with 3D volumetric convolution and synthesized novel 3D objects from normal distribution. Smith et al. [24] improved 3D-GAN with Wasserstein GAN [25], which enhances the stability of the learning process. The conditional generative adversarial network (CGAN) [3] generates targeted images given a one-hot encoded class vector and noise vectors. Chen et al. [26] generated colored 3D objects in voxel grids given shape descriptions by jointly learning representations of the text description and 3D colored shapes with metric learning. Conditional Variational Auto-Encoder (CVAE) [27] has been used to learn specific patterns which are structured in the underlying data distribution. Bao et al. [28] combined the CVAE framework and adversarial loss, which performed fine-grained image generation. The conditional generative model for 3D objects has not been well-explored, and thus, existing methods in the 3D domain have no explicit controllability to sample a specific class of 3D objects with a single pipeline. In our method, a single pipeline is used to generate targeted objects given a targeted class one-hot vector.

Conditional Generative Model

Conditional generative models have been widely used in computer vision areas such as geometric prediction [29–32] and non-rigid object modification such as human face deformation [33–36]. Dosovitskiy et al. [37] has proposed a supervised, conditional generative model trained to generate images of chairs, tables, and cars with specified attributes which are controlled by transformation and view parameters passed to the network. MV3D [30] is pioneering deep learning work for object view synthesis which uses an encoder-decoder network to directly generate pixels of a target view with depth information in the loss function along with view point information passed as a conditional term. The appearance flow network (AFN) [31] proposed a method for view synthesis of objects by predicting appearance flow fields, which are used to move pixels from an input to a target view. However, this method requires detailed camera pose information and is not capable of predicting pixels which are missing in the source views. CVAE-GAN [28] further adds adversarial training to the CVAE framework in order to improve the quality of generated predictions. The work from Zhang et al. [33] have introduced the conditional adversarial autoencoder (CAAE) designed to model age progression/regression in human faces. This is achieved by concatenating conditioning information (i.e. age) with the input's latent representation before proceeding to the decoding process. The framework also includes an adaptive discriminator with conditional information passed using a resize/concatenate procedure. The conditional variational autoencoder (CVAE) incorporates conditioning information into the standard variational autoencoder (VAE) framework [13] and is capable of synthesizing specified attribute changes in an identity preserving manner [38, 39]. Other works have introduced a clamping strategy to enforce a specific organizational structure in the latent space [34, 40]; these networks require extremely detailed labels for supervision, such as the graphics code parameters used to create each example, and are therefore very difficult to implement for more general tasks (e.g. training with real images).

2.2 Discriminative Learning in 3D Space

3D shape descriptors

Shape descriptors extract features for classifying or retrieving objects and trained by discriminative learning approach. There are three different 3D representations: (1) point clouds, (2) projected views, and (3) voxel grids, for representing 3D space with computer, which are used in deep learning. In this section, we illustrate the deep learning based 3D shape descriptors on each representation for classifying the 3D objects.

Point Clouds A point cloud is a collection of points in Euclidean space. PointCNN [41] relaxes irregularity of point clouds by approximating the transformation matrix with multi-layer perception, which simultaneously weights and permutes the input features for point cloud data feature learning. PointNet [42] learns a set of optimization functions for selecting feature points that contain meaningful content, which canonicalizes the input point clouds and aggregates all feature points to capture global point cloud features. PointNet++ [42] is an advanced version of PointNet. This work focused on recognizing fine-grained patterns with a hierarchical neural network which iteratively applied on a nested partitioned point set. SpiderCNN [43] proposes a convolutional layer, which is a product of a step function that captures local geodesic information and a Taylor polynomial to convolve in point cloud.

Projected Views A 3D object can be represented as multiple-views that covers all details of the object. View-based methods [44–46] extract features of 3D shape representations by observing multi-view images of an object and jointly estimating their poses. Their method successfully works for object classification and shape retrieval tasks, but performed poorly on unknown orientation models. Su et al. [44] uses a collection of multiple views of 3D objects, which is effective for learning their representations. MVCNN [46] further improves Su et al. with cross-modal distillation and adversarial inputs with a differentiable renderer.

Voxel Grids Three-dimensional objects can be discredited and represented in voxel grids, and voxel-based classifiers use voxel grids as their inputs. DLAN [47] proposes Rotation Normalized Grids (RNGs), which are samples of oriented point sets rotated by PCA for shape retrieval. Multiple blocks of RNGs are converted into local features with 3D convolution, and these features are aggregated with average pooling as object representations. VSL [48] learns the probabilistic manifold of the underlying structure of voxelized 3D shapes with an auto-encoder in an unsupervised manner. VoxNet [49] converts point clouds into voxels in voxel grids and extracts features with a 3D convolution layer for the classification tasks. VRN [50] uses a series of 3D convolutional layers to extract features for classifying objects compactly.

Large-scale 3D object datasets

General object dataset has been widely developed for 3D computer vision applications. The Princeton Shape Benchmark (PSB) [51] is an early work that collected and annotated 3D objects for shape matching and classification benchmarking. It collected 3D polygonal models from the World Wide Web and classified them based on the method of construction, such as man-made and natural objects. ShapeNet [52] is a large-scale dataset of high-quality 3D models of objects, which are widely used in various tasks such as instance segmentation [53], shape retrieval [54], and shape reconstruction [55]. ModelNet [56] consists of two datasets (a 10-class dataset and a 40-class dataset) and demonstrates a comprehensive clean collection of 3D CAD models of objects. PartNet [53] is a fine-grained, instance-level, hierarchical parts dataset. It used 3D objects from ShapeNet and was annotated by 66 annotators.

There are not many engineering shape dataset compared with general objects due to the lack of 3D models and requiring domain knowledge to annotate the mechanical components. Engineering shape datasets has been developed to improve the shape-based retrieval of 3D data of mechanical parts [57]. The Engineering Shape Benchmark (ESB) [58] is an annotated engineering shape dataset. It proposed an approach that defines the class by mechanical part's name—not by functionality—and benchmarked analytical shape descriptor. However, the number of models in the ESB dataset is not sufficient for training a robust feature extractor, and classes are only classified by their shape, which limits the usage of the dataset. The Actual Artifacts Dataset (AAD) [59] consists of four datasets with a total around 700 models and provides several classifications for engineering artifacts selected from the National Design Repository (NDR) [60]. Recently, A Big Cad (ABC) Model Dataset [61] proposed one million Computer-Aided Design (CAD) models dataset without annotations.

Part-aware / structure-aware 3D object learning

Shape structure is a relation of individual parts and a high-level representation of object, which is crucial for object perception [62]. Generative models for 3D objects [63–66] have been developed based on learning the structure of objects by encoding parts to latent vectors with encoder networks and decoding the latent vectors with decoder networks, which leverages the learning capability for object structure. Dubrovina et al. [67] labeled parts given unlabeled shapes by decomposing each part embedding with decoders and composing them with a spatial transformation network. Mo et al. [68] used a hierarchical graph network which encodes part structures of object to modify a source object into a target object. Schor et al. [69] generated each part of an object and composed these parts into a novel object. The difference between our work and above works is that networks explicitly learn part surface area and volume of each part of object with their location in Cartesian coordinates for learning local geometry and object structure. We also expand the learning space of each part by deforming each part individually given hand crafted parameters.

3D Object reconstruction

Various methods have been proposed for object reconstruction in three-dimensional domain. Dai et al. [70] reconstructed a corrupted distance field and state voxel grid with an encoder-decoder model and a shape database. The final output of this method was the reconstructed distance fields of 3D objects in the 3D space. Sung et al. [71] used the structure of 3D objects as prior knowledge for shape completion given noisy depth scans of objects. Surface and volumetric information is related and imposes complementary information of objects. These works [70, 71] have not utilized the correlation of surface and volumetric information of objects for object reconstruction.

Our method jointly learns surface and volumetric information for object reconstruction.

2.3 Pixel-wise Segmentation

Deep learning based pixel-wise segmentation algorithms have been widely used for segmenting objects in videos [72–75]. Deep-learning based methods use CNNs to predict each pixel in an image by extracting feature representations. One of the popular structures is an encoder-decoder structure which projects a high-dimensional image into the latent vector and decodes the latent vector into class-wise pixel space [76, 77]. Indeed, making large-scale pixel-wise datasets is crucial for training the CNNs. Although complex boundaries can be traced manually, labeling image/video datasets with object masks in pixel-level is extremely time consuming [78]. A successful method is to have a neural network produce polygon annotations of objects interactively using humans-in-the-loop to improve the accuracy [79–81]. The machine can provide the human with information to manipulate [82] and generate segments [83], matting layers [84], or boundary fragments [85]. Sequences in video datasets consist of similar frames which contain redundant information. For moving objects especially, several notable and established annotation tools with auto-tracking demonstrate great performance on efficiency improvement [86, 87].

Researchers create hand segmentation datasets by manually drawing polygon or coloring the hand on RGB frames [88, 88–91]. Other works use hardware sensors to predict joint position and render with mesh models; however, the sensors are visible in the RGB images [92, 93]. Alternatively, studies showed good performance when utilizing hand pose estimation and mesh models to segment hands in frames [94, 95]. Our method doesn't require training a hand pose network [96] to generate high accuracy on hand pose estimation under heavily occluded situations.

2.4 Data fusion

Data fusion of high-quality multimodal information from multi-source for machine perception is one of the key concepts in machine perception. The mammal, in general, can sense at least three essential sources: texture, vision, and sounds. These three basic senses are processed in the brain to understand and perceive the world around them. Similarly, the autonomous driving industrys advance uses a single-modal sensor and multi-source and multiple sensors in 3D space. This is because combining multimodal information reduces the uncertainty of machine perception. Multimodal visual information is widely used to capture robust features because of color and texture invariance and its expression of an object in an explicit 3D information [97–100]. Therefore, Long Wavelength InfraRed (LWIR) is widely used in detecting objects [100] and controlling unmanned aerial vehicles due to its outstanding ability of identifying objects with a distinct temperature from the surroundings [101–103]. MFNet [104] showed that fusing LWIR and RGB frames significantly enhances the segmentation performance. Luo et al. [105] uses thermal information to segment coarse point clouds scenes. Thus, many RGB pixels can not be labeled from the coarse labeled point clouds. Many works with LWIR sensors are for scene segmentation and autonomous driving system.

3. LATENT TRANSFORMATIONS NEURAL NETWORK

View synthesis can be used to generate 3D point cloud representations of objects from a single input image [30]; in the context of hand pose estimation algorithms, generating additional synthetic views can also help reduce occlusion and improve the accuracy of the estimated poses [106, 107]. However, synthesizing novel views from a single input image is a formidable task with serious complications arising from the complexity of the target object and the presence of heavily self-occluded parts.

To address this problem, we propose the latent transformation neural network (LTNN) and provide a general framework for effectively performing inference with conditional generative models by strategically controlling the interaction between conditioning information and latent representations within a generative inference model.

In this framework, a Conditional Transformation Unit (CTU), Φ, is introduced to provide a means for navigating the underlying manifold structure of the latent space. The CTU is realized in the form of a collection of convolutional layers which are designed to approximate the latent space operators defined by mapping encoded inputs to the encoded representations of specified targets (see Figure 3.1). This is enforced by introducing a *consistency loss* term to guide the CTU mappings during training. In addition, a Conditional Discriminator Unit (CDU), Ψ, also realized as a collection of convolutional layers, is included in the network's discriminator. This CDU is designed to improve the network's ability to identify and eliminate transformation specific artifacts in the network's predictions.

The network has also been equipped with RGB balance parameters consisting of three values $\{\theta_R, \theta_G, \theta_B\}$ designed to give the network the ability to quickly adjust the global color balance of the images it produces to better align with that of the true data distribution. In this way, the network is easily able to remove unnatural hues and focus on estimating local pixel values by adjusting the three RGB parameters rather than correcting each pixel individually. In addition, we introduce a novel estimation strategy for efficiently learning shape and color properties simultaneously; a *Task-divided* Decoder (TD) is designed to produce a coarse pixel-value map along with a refinement map in order to split the network's overall task into distinct, dedicated network components.

3.1 Latent Transformation Layer

In this section, we introduce the methods used to define the proposed LTNN model. We first give a brief overview of the LTNN network structure. We then detail how conditional transformation unit mappings are defined and trained to operate on the latent space, followed by a description of the conditional discriminator unit

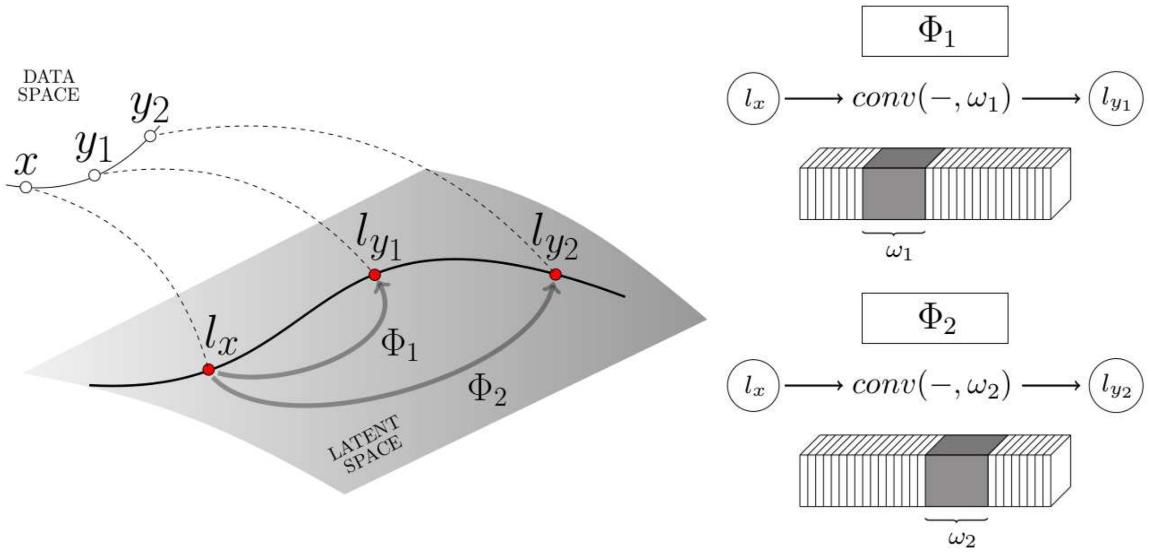

DATA SPACE
x
y_1
y_2
l_{y_1}
l_{y_2}
l_x
Φ_1
Φ_2
LATENT SPACE
Φ_1
$l_x \longrightarrow conv(-, \omega_1) \longrightarrow l_{y_1}$
ω_1
Φ_2
$l_x \longrightarrow conv(-, \omega_2) \longrightarrow l_{y_2}$
ω_2

Fig. 3.1. The conditional transformation unit Φ constructs a collection of mappings $\{\Phi_k\}$ in the latent space which produces object view changes to the decoded outputs. The encoding l_x of the original input image x is transformed to $\widehat{l}_{y_k} = \Phi_k(l_x) = \text{conv}(l_x, \omega_k)$ and provides an approximation to the encoding l_{y_k} of the attribute-modified target image y_k.

implementation and the network loss function used to guide the training process. Lastly, we describe the task-division framework used for the decoding process.

The basic workflow of the proposed model is as follows:

1. Encode the input image x to a latent representation $l_x = \text{Encode}(x)$.
2. Use conditioning information k to select conditional, convolutional filter weights ω_k.
3. Map the latent representation l_x to $\widehat{l}_{y_k} = \Phi_k(l_x) = \text{conv}(l_x, \omega_k)$, an approximation of the encoded latent representation l_{y_k} of the specified target image y_k.
4. Decode $\widehat{l}_{y_k}$ to obtain a coarse pixel value map and a refinement map.
5. Scale the channels of the pixel value map by the RGB balance parameters and take the Hadamard product with the refinement map to obtain the final prediction $\widehat{y}_k$.
6. Pass real images y_k as well as generated images $\widehat{y}_k$ to the discriminator, and use the conditioning information to select the discriminator's conditional filter weights $\overline{\omega}_k$.
7. Compute loss and update weights using ADAM optimization and backpropagation.

3.1.1 Conditional transformation unit

Generative models have frequently been designed to explicitly disentangle the latent space in order to enable high-level attribute modification through linear, latent space interpolation. This linear latent structure is imposed by design decisions, however, and may not be the most natural way for a network to internalize features of

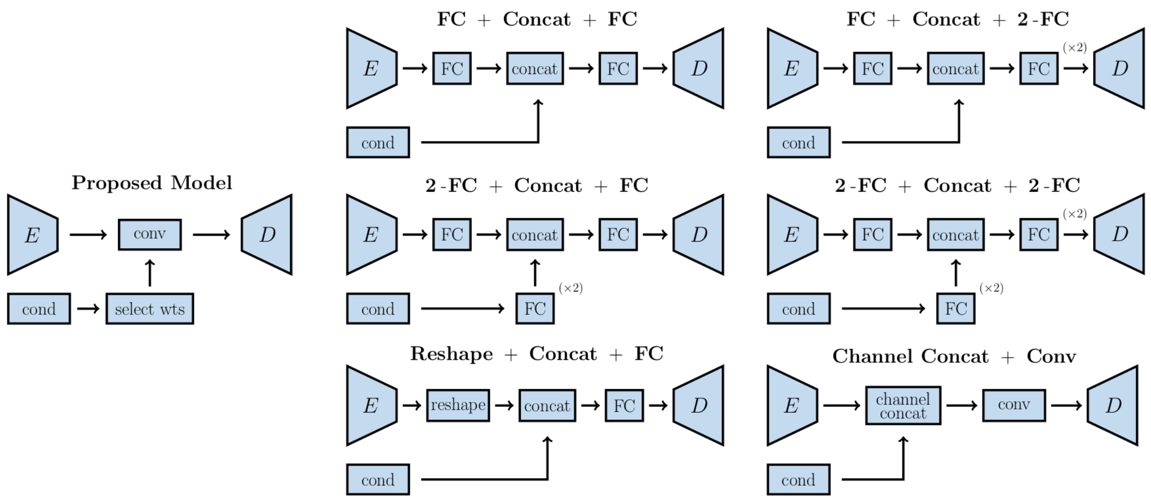


Fig. 3.2. Selected methods for incorporating conditioning information; the proposed LTNN method is illustrated on the left, and six conventional alternatives are shown to the right.

the data distribution. Several approaches have been proposed which include nonlinear layers for processing conditioning information at the latent space level. In these conventional conditional generative frameworks, conditioning information is introduced by combining features extracted from the input with features extracted from the conditioning information (often using dense connection layers); these features are typically combined using standard vector concatenation, although some have opted to use channel concatenation.

In particular, conventional approaches for incorporating conditional information generally fall into three classes: (1) apply a fully connected layer before and after concatenating a vector storing conditional information [30, 31, 33, 40], (2) flatten the network features and concatenate with a vector storing conditional information [29], (3) tile a conditional vector to create a two-dimensional array with the same shape as the network features and concatenate channel-wise [28, 108]. Since the first class is more prevalent than the others in practice, we have subdivided this class into four cases: FC-Concat-FC [33], FC-Concat-2FC [40], 2FC-Concat-FC [31], and 2FC-

Concat-2FC [30]. Six of these conventional conditional network designs are illustrated in Figure 3.2 along with the proposed LTNN network design for incorporating conditioning information.

Rather than directly concatenating conditioning information with network features, we propose using a conditional transformation unit (CTU), consisting of a collection of distinct convolutional mappings in the network's latent space. More specifically, the CTU maintains independent convolution kernel weights for each target view in consideration. Conditioning information is used to select which collection of kernel weights, i.e. which CTU mapping, should be used in the CTU convolutional layer to perform a specified transformation. In addition to the convolutional kernel weights, each CTU mapping incorporates a Swish activation [109] with independent parameters for each specified target view. The kernel weights and Swish parameters of each CTU mapping are selectively updated by controlling the gradient flow based on the conditioning information provided.

The CTU mappings are trained to transform the encoded, latent space representation of the network's input in a manner which produces high-level view or attribute changes upon decoding. In this way, different angles of view, light directions, and deformations, for example, can be generated from a single input image. In one embodiment, the training process for the conditional transformation units can be designed to form a semi-group $\{\Phi_t\}_{t\geq 0}$ of operators:

$$i.e. \quad \begin{cases} \Phi_0 = id \\ \Phi_{t+s} = \Phi_t \circ \Phi_s \quad \forall\, t,s \geq 0 \end{cases} \tag{3.1}$$

defined on the latent space and trained to follow the geometric flow corresponding to a specified attribute. In the context of rotating three-dimensional objects, for example, the transformation units are trained on the input images paired with several target outputs corresponding to different angles of rotation; the network then uses conditioning information, which specifies the angle by which the object should be rotated, to select the appropriate transformation unit. In this context, the semi-group criteria corresponds to the fact that rotating an object 10 degree twice should align with the result of rotating the object by 20 degree once.

Since the encoder and decoder are not influenced by the specified angle of rotation, the network's encoding/decoding structure learns to model objects at different angles simultaneously; the single, low-dimensional latent representation of the input contains all information required to produce rotated views of the original object. Other embodiments can depart with this semi-group formulation, however, training conditional transformation units to instead produce a more diverse collection of non-sequential viewpoints, for example, as is the case for multi-view hand synthesis.

To enforce this behavior on the latent space CTU mappings in practice, a *consistency* term is introduced into the loss function, as specified in Equation 3.2. This loss term is minimized precisely when the CTU mappings behave as depicted in Figure 3.1; in particular, the output of the CTU mapping associated with a particular transformation is designed to match the encoding of the associated ground truth target view. More precisely, given an input image x, the consistency loss associated with the k^{th} transformation is defined in terms of the ground truth, transformed target view y_k by:

$$\mathcal{L}_{consist} \;=\; \Big\| \, \Phi_k(\text{Encode}[x]) \;-\; \text{Encode}[y_k] \, \Big\|_1 \tag{3.2}$$

3.1.2 Conditional discriminator unit and loss function

The discriminator used in the adversarial training process is also passed conditioning information which specifies the transformation which the model has attempted to make. The Conditional Discriminator Unit (CDU), which is implemented as a convolutional layer with modular weights similar to the CTU, is trained to specifically identify unrealistic artifacts which are being produced by the corresponding conditional transformation unit mappings. This is accomplished by maintaining independent convolutional kernel weights for each specified target view and using the conditioning information passed to the discriminator to select the kernel weights for the CDU layer. The incorporation of this context-aware discriminator structure has significantly boosted the performance of the network (see Table 3.3). The discriminator, $\mathcal{D}$, is trained using the adversarial loss term $\mathcal{L}_{adv}^{\mathcal{D}}$ defined below in Equation 3.3. The proposed model uses the adversarial loss in Equation 3.4 to effectively capture multimodal distributions [110], which helps to sharpen the generated views.

$$\mathcal{L}_{adv}^{\mathcal{D}} = -\log \mathcal{D}(y_k, \overline{\omega}_k) - \log\big(1 - \mathcal{D}(\widehat{y}_k, \overline{\omega}_k)\big) \tag{3.3}$$

$$\mathcal{L}_{adv} = -\log \mathcal{D}(\widehat{y}_k, \overline{\omega}_k) \tag{3.4}$$

Reducing the total variation is widely used in view synthesis methods [29, 33]. In particular, the L_{smooth} term is used to reduce noise in the generated images by reducing the variation of pixels, which is inspired by total variation image denoising.
Experimental evidence shows that the inclusion of the L_{smooth} loss term leads to an improvement in the overall quality of the synthesized images (see Table 3.3). We have experimented with various shift sizes and found that the shift size $\tau = 1$ yields the best performance. Additional loss terms corresponding to accurate structural reconstruction and smoothness [111] in the generated views are defined in Equations 3.5 and 3.6:

$$\mathcal{L}_{recon} = \|\, \widehat{y}_k - y_k \,\|_2^2 \tag{3.5}$$

$$\mathcal{L}_{smooth} = \sum_{i \in \{0, \pm 1\}} \sum_{j \in \{0, \pm 1\}} \big\|\, \widehat{y}_k - \tau_{i,j} \widehat{y}_k \,\big\|_1 \tag{3.6}$$

where y_k is the modified target image corresponding to an input x, $\overline{\omega}_k$ are the weights of the CDU mapping corresponding to the k^{th} transformation, Φ_k is the CTU mapping for the k^{th} transformation, $\widehat{y}_k = \text{Decode}\big(\Phi_k\big(\text{Encode}[x]\big)\big)$ is the network prediction, and $\tau_{i,j}$ is the two-dimensional, discrete shift operator. The final loss function for the encoder and decoder components is given by:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{adv} + \rho \cdot \mathcal{L}_{recon} + \gamma \cdot \mathcal{L}_{smooth} + \kappa \cdot \mathcal{L}_{consist} \tag{3.7}$$

with hyperparameters typically selected so that $\lambda, \rho \gg \gamma, \kappa$. The consistency loss is designed to guide the CTU mappings toward approximations of the latent space mappings which connect the latent representations of input images and target images as depicted in Figure 3.1. In particular, the consistency term enforces the condition that the transformed encoding, $\widehat{l_{y_k}} = \Phi_k(\text{Encode}[x])$, approximates the encoding of the k^{th} target image, $l_{y_k} = \text{Encode}[y_k]$, during the training process.

3.2 Task-Divided Decoder

The decoding process has been divided into three tasks: estimating the refinement map, pixel-values, and RGB color balance of the dataset. We have found this decoupled framework for estimation helps the network converge to better minima to produce sharp, realistic outputs without additional loss terms. The decoding process begins with a series of convolutional layers followed by bilinear interpolation to upsample the low resolution latent information. The last component of the de-

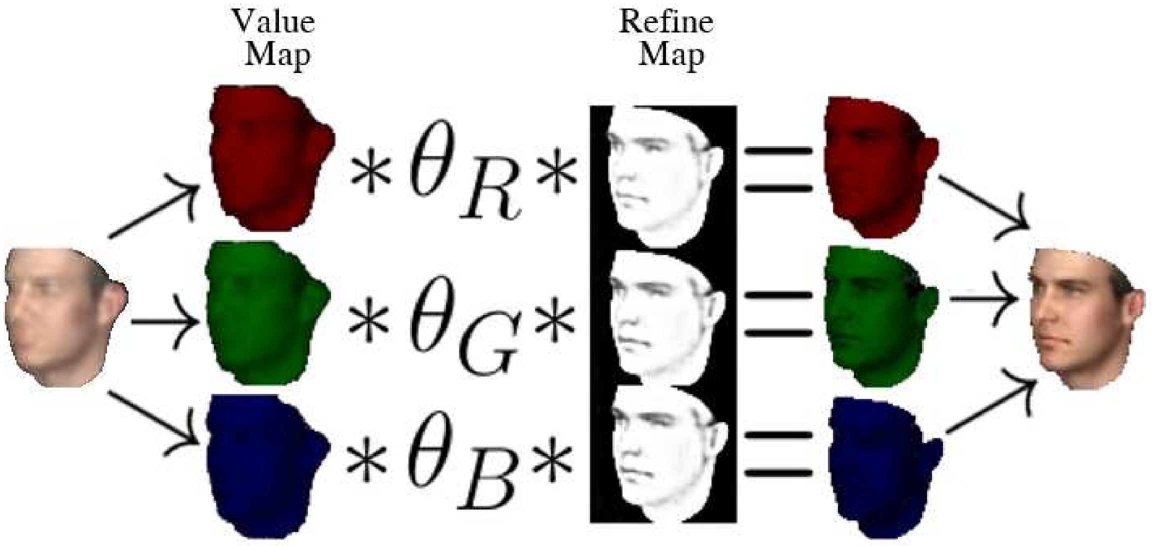


Fig. 3.3. Proposed task-divided design for the LTNN decoder. The coarse pixel value estimation map is split into RGB channels, rescaled by the RGB balance parameters, and multiplied element-wise by the refinement map values to produce the final network prediction.

coder's up-sampling process consists of two distinct convolutional layers used for task divided; one layer is allocated for predicting the refinement map while the other is trained to predict pixel-values. The refinement map layer incorporates a sigmoidal activation function which outputs scaling factors intended to refine the coarse pixel value estimations; the pixel-value estimation layer does not use an activation so that the output values are not restricted to the range of a specific activation function. RGB balance parameters, consisting of three trainable variables, are used as weights for balancing the color channels of the pixel value map. The Hadamard product, $\odot$, of the refinement map and the RGB-rescaled value map serves as the network's final output:

$$\begin{aligned} \widehat{y} \;&= [\,\widehat{y}_R,\, \widehat{y}_G,\, \widehat{y}_B\,] \text{ where} \\ \widehat{y}_C \;&= \theta_C \cdot \widehat{y}_C^{value} \odot \widehat{y}_C^{refine} \text{ for } C \in \{R, G, B\} \end{aligned} \tag{3.8}$$

In this way, the network has the capacity to mask values which lie outside of the target object (i.e. by setting refinement map values to zero) which allows the value map to focus on the object itself during the training process. Experimental results show that the refinement maps learn to produce masks which closely resemble the target objects' shapes and have sharp drop-offs along the boundaries. No additional information has been provided to the network for training the refinement map; the masking behavior illustrated in Figures 3.3 and 3.6 is learned implicitly by the network during training, and is made possible by the design of the network's architecture. As seen in Figure 3.3, the refinement map produces a shape mask and mask out errors in each pixels by masking values which lie outside of the target object (i.e. by setting refinement map values to zero).

3.3 Architecture Details

The overview of the pipeline is shown in Figure 6.1. Input images are passed through a Block v1 collaborative filter layer (see Figure 3.4) along with a max pooling layer to produce the features at the far left end of the figure. At the bottle-neck between the encoder and decoder, a conditional transformation unit (CTU) is applied to map the 2×2 latent features directly to the transformed 2×2 latent features on the right. This CTU is implemented as a convolutional layer with 3×3 filter weights selected based on the conditioning information provided to the network. The features near the end of the decoder component are processed by two independent convolution transpose layers for non-rigid object and bilinear interpolation for the rigid object: one corresponding to the value estimation map and the other corresponding to the refinement map. The channels of the value estimation map are rescaled by the RGB balance parameters, and the Hadamard product is taken with the refinement map to produce the final network output. For rigid object experiment, we added tangent hyperbolic activation function after the Hadamard product to bound the output values range in [-1,1]. The CDU is also designed to have the same 3×3 kernel size as the CTU and is applied between the third and fourth layers of the discriminator.

For the stereo face dataset [112] experiment, we have added an additional Block v1 layer in the encoder and additional convolutional layer followed by bilinear interpolation in decoder to utilize the full 128×128×3 resolution images and two Block v1 lay-

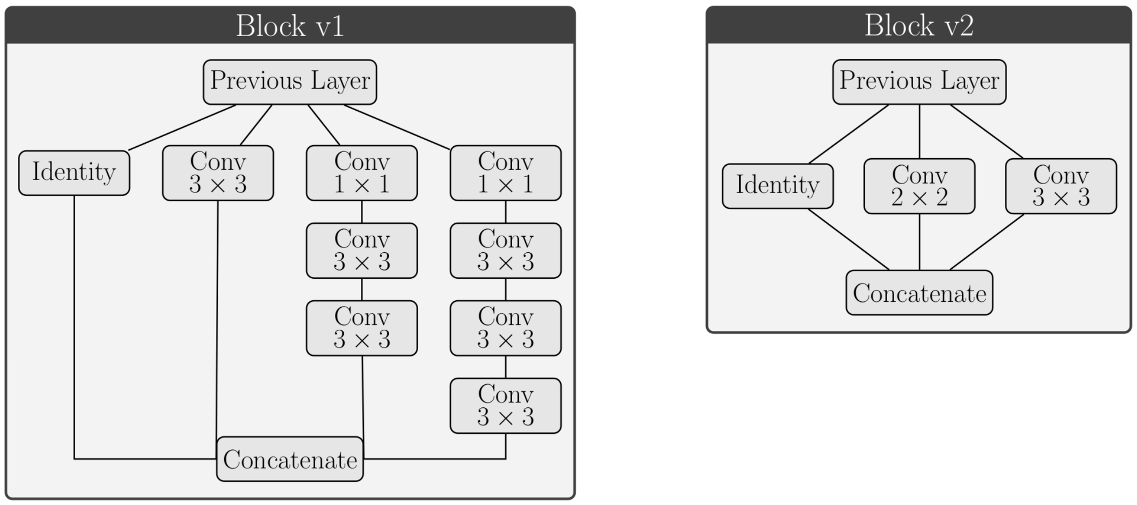


Fig. 3.4. Layer definitions for Block v1 and Block v2 collaborative filters.

ers and two convolutional layer followed by bilinear interpolation for the 256×256×3 resolution image of rigid object views.

The encoder incorporates two main block layers, as defined in Figure 3.4, which are designed to provide efficient feature extraction; these blocks follow a similar design to that proposed by [113], but include dense connections between blocks, as introduced by [114]. We normalize the output of each network layer using the batch normalization method as described in [115]. For the decoder, we have opted for a minimalist design, inspired by the work of [72]. Standard convolutional layers with 3×3 filters and same padding are used through the penultimate decoding layer, and transpose convolutional layers with 1×1 filters for non-rigid objects and 5×5 for other experiments. We have used same padding to produce the value-estimation and refinement maps. All parameters have been initialized using the variance scaling initialization method described in [116].

Our method has been implemented and developed using the TensorFlow framework. The models have been trained using stochastic gradient descent (SGD) and

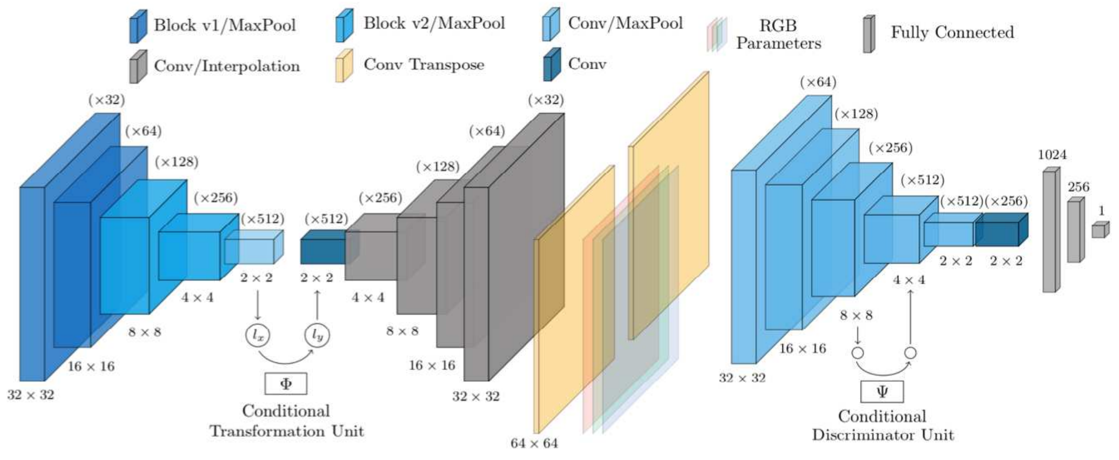


Fig. 3.5. The proposed network structure for the encoder/decoder (left) and discriminator (right). Features have been color-coded according to the type of layer which has produced them.

the ADAM optimizer [117] with initial parameters: learning_rate = 0.005, β_1 = 0.9, and β_2 = 0.999 (as defined in the TensorFlow API r1.6 documentation for tf.train.AdamOptimizer), along with loss function hyper parameters: λ = 0.8, ρ = 0.2, γ = 0.000025, and κ = 0.00005 (as introduced in Equation 3.7). The discriminator is updated once every two encoder/decoder updates, and one-sided label smoothing [110] has been used to improve stability of the discriminator training procedure.

3.4 Experiments

We conduct experiments on a diverse collection of datasets including both rigid and non-rigid objects. To show the generalizability of our method, we have conducted a series of experiments: (i) hand pose estimation using a synthetic training set and real NYU hand depth image data [118] for testing, (ii) synthesis of rotated views of rigid objects using the 3D object dataset [52], (iii) synthesis of rotated views using a real face dataset [112], and (iv) the modification of a diverse range of attributes on a synthetic face dataset [119]. For each experiment, we have trained the models using 80% of the datasets. Since ground truth target depth images were not available for the real hand dataset, an indirect metric has been used to quantitatively evaluate the model as described in Section 3.4.2. Ground truth data was available for all other experiments, and models were evaluated directly using the L_1 mean pixel-wise error and the Structural Similarity Index Measure (SSIM) [120] used in [29, 108]. To evaluate the proposed framework with existing works, two comparison groups have been formed: conditional inference methods, CVAE-GAN [28] and CAAE [33], with comparable hourglass structures for comparison on experiments with non-rigid objects, and view synthesis methods, MV3D [30], M2N [108], AFN [31], and TVSN [29], for comparison on experiments with rigid objects. Additional ablation experiments have been performed to compare the proposed CTU conditioning method with other conventional concatenation methods (see Figure 3.2); results are shown in Figure 3.8 and Table 3.3.

3.4.1 Experiment on rigid objects

Rigid object experiment: We have experimented with novel 3D view synthesis tasks given a single view of an object with an arbitrary pose. The goal of this experiment is to synthesize an image of the object after a specified transformation or change in viewpoint has been applied to the original view. To evaluate our method in the context of rigid objects, we have performed a collection of tests on the chair and car datasets. Given a single input view of an object, we leverage the LTNN model to produce ˚360 views of the object. We have tested our model's ability to perform ˚360 view estimation on 3D objects and compared the results with the other state-of-the-art methods. The models are trained on the same dataset used in M2N [108]. The car and chair categories from the ShapeNet [121] 3D model repository have been rotated horizontally 18 times by ˚20 along with elevation changes of ˚0, ˚10, and ˚20.

Table 3.1.
FLOPs and parameter counts corresponding to inference for a single image with resolution 256×256×3. These calculations are based on code provided by the authors and the definitions prescribed in the associated papers. Smaller numbers are better for parameters and GFLOPs/Image.

Model	Parameters (Million)	GFLOPs / Image
Ours	**17.0**	**2.183**
M2N	127.1	341.404
TVSN	57.3	2.860
AFN	70.3	2.671
MV3D	69.7	3.056

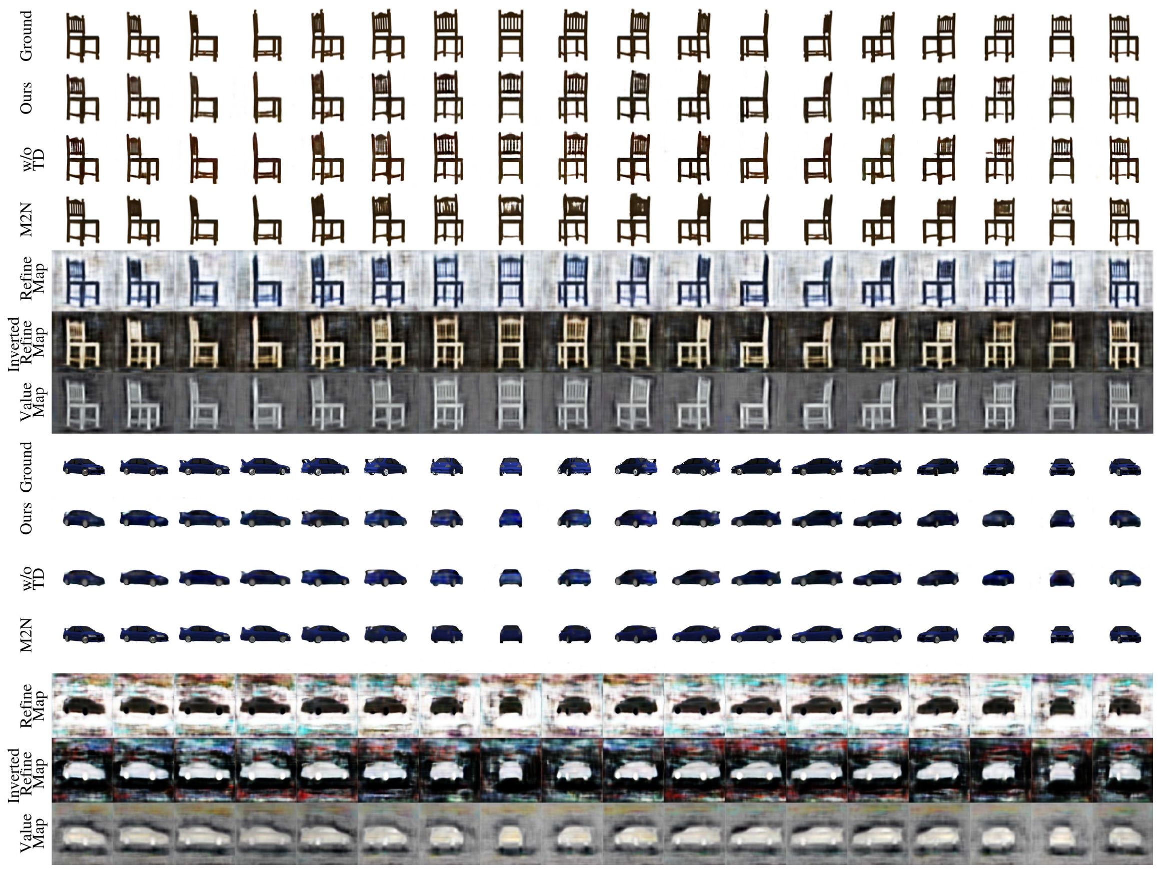


Fig. 3.6. Qualitative comparison of °360 view prediction of rigid-objects. A single image, shown in the first column of the "Ground" row, is used as the input for the network. Results are shown for the proposed network with and without task-division "w/o TD".

The M2N and TVSN results are slightly better for the car category, however these works have incorporated skip connections between the encoder layers and decoder layers, proposed in U-net [122], which substantially increases the computational demand for these networks (see Table 3.1). As can be seen in Tables 3.1 and 3.2, the proposed model is comparable with existing models specifically designed for the task of multi-view prediction while requiring the least FLOPs for inference compared with all other methods. The low computational cost of the LTNN model highlights the efficiency of the CTU/CDU framework for incorporating conditional information into the network

Table 3.2.
Quantitative comparison for °360 view synthesis of rigid objects. Smaller numbers are better for L_1 and higher numbers are better for SSIM. We performed ablation experiment with and without Task-divided Decoder (TD) and compared with other methods.

	Car		Chair	
Model	SSIM	L_1	SSIM	L_1
Ours	.902	.121	**.897**	**.178**
Ours (w/o TD)	.861	.187	.871	.261
M2N	**.923**	**.098**	.895	.181
TVSN	.913	.119	.894	.230
AFN	.877	.148	.891	.240
MV3D	.875	.139	.895	.248

for view synthesis. Moreover, as seen in the qualitative results provided in Figure 3.6, using a task-divided decoder helps to eliminate artifacts in the generated views; in particular, the spokes on the back of the chair and the spoiler on the back of the car are seen to be synthesized much more clearly when using a task-divided decoder.

3.4.2 Experiment on non-rigid objects

Hand pose experiment: To assess the performance of the proposed network on non-rigid objects, we consider the problem of hand pose estimation. As the number of available view points of a given hand are increased, the task of estimating the associated hand pose becomes significantly easier [106]. Motivated by this fact, we synthesize multiple views of a hand given a single view and evaluate the accuracy of the estimated hand pose using the synthesized views. The underlying assumption of the assessment is that the accuracy of the hand pose estimation will be improved precisely when the synthesized views provide faithful representations of the true hand

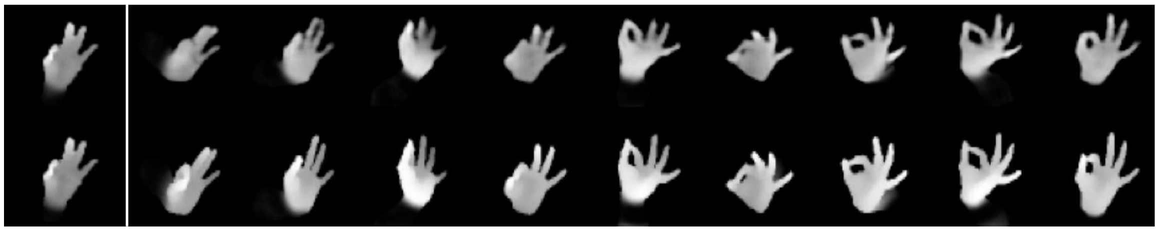


Fig. 3.7. Comparison of CVAE-GAN (top) with proposed LTNN model (bottom) using the noisy NYU hand dataset [118]. The input depth-map hand pose image is shown to the far left, followed by the network predictions for 9 synthesized view points. The views synthesized using LTNN are seen to be sharper and also yield higher accuracy for pose estimation (see Figure3.10).

pose. Since ground truth predictions for the real NYU hand dataset were not available, the LTNN model has been trained using a synthetic dataset generated using 3D mesh hand models. The NYU dataset does, however, provide ground truth coordinates for the input hand pose; using this we were able to indirectly evaluate the performance of the model by assessing the accuracy of a hand pose estimation method using the network's multi-view predictions as input.

More specifically, the LTNN model was trained to generate 9 different views which were then fed into the pose estimation network from Choi et al. [123] (also trained using the synthetic dataset). For an evaluation metric, the maximum error in the predicted joint locations has been computed for each frame (i.e. each hand pose in the dataset). The cumulative number of frames with maximum error below a threshold distance ϵ_D have then been computed, as is commonly used in hand pose estimation tasks [123, 124]. A comparison of the pose estimation results using synthetic views generated by the proposed model, the CVAE-GAN model, and the CAAE model are presented in Figure 3.8, along with the results obtained by performing pose estimation using the single-view input frame alone. In particular, for a threshold distance $\epsilon_D =$ 40mm, the proposed model yields the highest accuracy with 61.98% of the frames having all predicted joint locations within a distance of 40mm from the ground truth values. The second highest accuracy is achieved with the CVAE-GAN model with 45.70% of frames predicted within the 40mm threshold.

A comparison of the quantitative hand pose estimation results is provided in Figure 3.8 where the proposed LTNN framework is seen to provide a substantial improvement over existing methods; qualitative results are also available in Figure 3.7. Ablation study results for assessing the impact of individual components of the LTNN model are also provided in Figure 3.8; in particular, we note that the inclusion of the CTU, CDU, and task-divided decoder each provide significant improvements to the performance of the network. With regard to real-time applications, the proposed

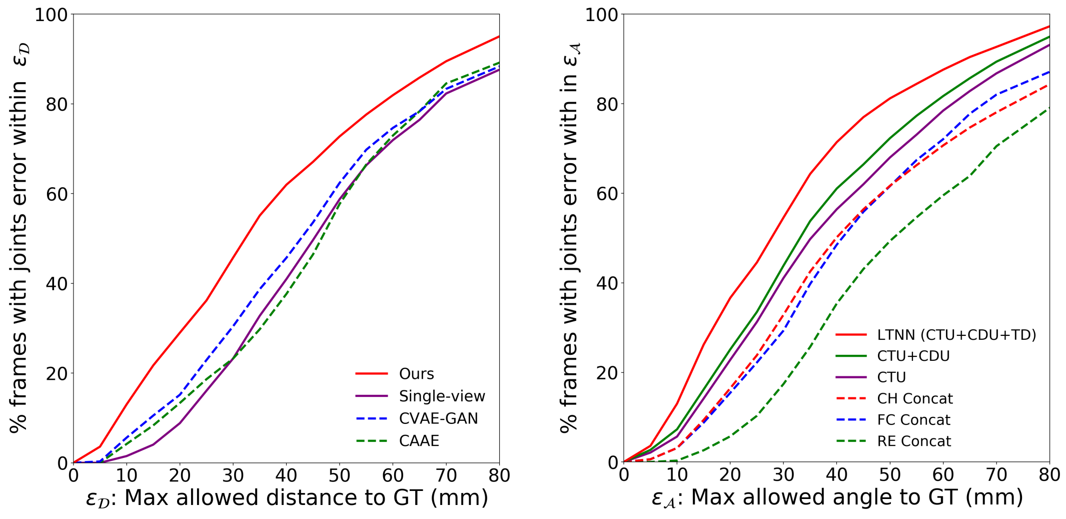


Fig. 3.8. Left figure shows Quantitative evaluation for multi-view hand synthesis using the real NYU dataset. Right figure shows LTNN ablation experiment results and comparison with alternative conditioning frameworks using synthetic hand dataset. Our models: Conditional Transformation Unit (CTU), Conditional Discriminator Unit (CDU), Task-divide Decoder (TD), and LTNN consisting of all previous components. Alternative concatenation methods: CHannel-wise Concatenation (CH Concat), Fully Connected Concatenation (FC Concat), and Reshape fully connected feature vector Concatenation (RE Concat).

model runs at 114 fps without batching and at 1975 fps when applied to a mini-batch of size 128 (using a single TITAN Xp GPU and an Intel i7-6850K CPU).

Real face experiment: We have also conducted an experiment using a real face dataset to show the applicability of LTNN for real images. The stereo face database [112], consisting of images of 100 individuals from 10 different viewpoints, was used for experiments with real faces. These faces were first segmented using the method of [125] and then we manually cleaned up the failure cases. The cleaned faces have been cropped and centered to form the final dataset. The LTNN model

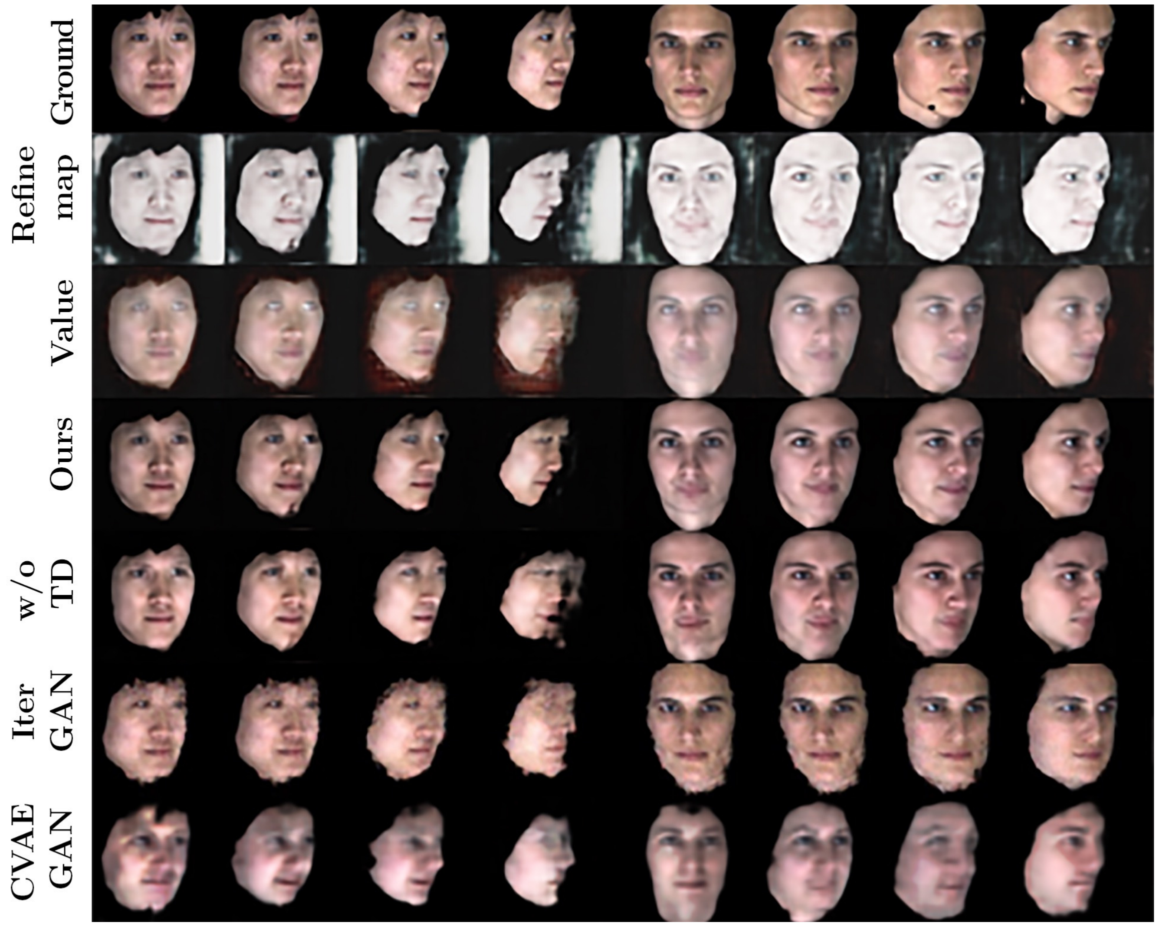


Fig. 3.9. Qualitative evaluation for view synthesis of real faces using the image dataset [112].

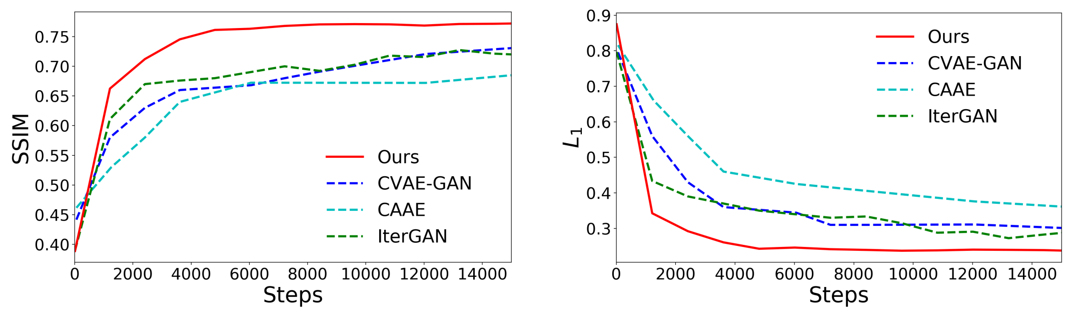


Fig. 3.10. Left figure shows quantitative evaluation with SSIM of model performances for experiment on the real face dataset [112]. Higher values are better. Right figure shows Quantitative evaluation with L_1 of model performances for experiment on the real face dataset [112]. Lower values are better.

was trained to synthesize images of input faces corresponding to three consecutive horizontal rotations. Qualitative results for the real face experiment are provided in Figure 3.9; in particular, we note that the quality of the views generated by the proposed LTNN model is consistent for each of the four views, while the quality of the views generated using other methods decreases substantially as the change in angle is increased. This illustrates the advantage of using CTU mappings to navigate the latent space and avoid the accumulation of errors inherent to iterative methods. Moreover, as shown in Figures 3.10, the LTNN model provides substantial improvements to alternative methods with respect to the SSIM and L_1 metrics and converges much faster as well.

3.4.3 Diverse attribute exploration

To evaluate the proposed framework's performance on a more diverse range of attribute modification tasks, a synthetic face dataset and other conditional generative models, CVAE-GAN and CAAE, with comparable hourglass structures to the LTNN model have been selected for comparison. The generated images from the LTNN model are available in Figure 3.11. These models have been trained to synthesize discrete changes in elevation, azimuth, light direction, and age from a single image. As shown in Table 3.4 and 3.5, the LTNN model outperforms the CVAE-GAN and CAAE models by a significant margin in both SSIM and L_1 metrics; additional quantitative results are provided in Table 3.3, along with a collection of ablation results for the LTNN model.

Multiple attributes can also be modified simultaneously using LTNN by composing CTU mappings. For example, one can train 4 CTU mappings $\{\Phi_k^{light}\}_{k=0}^3$ corresponding to incremental changes in lighting and 4 CTU mappings $\{\Phi_k^{azim}\}_{k=0}^3$ corresponding to incremental changes in azimuth. In this setting, the network predictions for lighting and azimuth changes correspond to the values of Decode$[\Phi_k^{light}(l_x)]$ and Decode$[\Phi_k^{azim}(l_x)]$, respectively (where l_x denotes the encoding of the original input image). To predict the effect of simultaneously changing both lighting and azimuth, we can compose the associated CTU mappings in the latent space; that is,

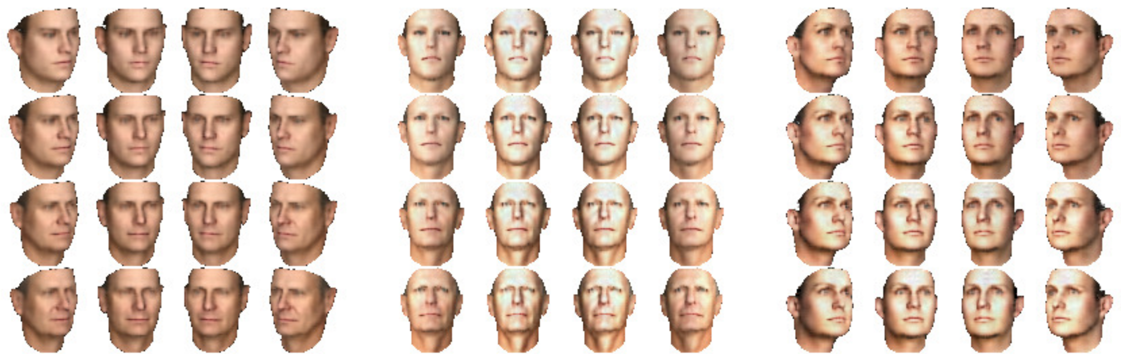


Fig. 3.11. Simultaneous learning of multiple attribute modifications. Azimuth and age (left), light and age (center), and light and azimuth (right) combined modifications are shown. The network has been trained using 4 CTU mappings per attribute (e.g. 4 azimuth mappings and 4 age mappings); results shown have been generated by composing CTU mappings in the latent space and decoding.

Table 3.3.
Ablation/comparison results of six different conventional alternatives for fusing condition information into the latent space and ablation study of conditional transformation unit (CTU), conditional discriminator unit (CDU), and task-divided decoder (TD). For valid comparison we used identical encoder, decoder, and training procedure with synthetic face dataset. CC, RC, and C stands for channel concatenation, reshape concatenation, and concatenation, respectively.

	Elevation		Azimuth		Light Direction		Age	
Model	SSIM	L_1	SSIM	L_1	SSIM	L_1	SSIM	L_1
LTNN	**.923**	**.107**	**.923**	**.108**	**.941**	**.093**	**.925**	**.102**
LTNN w/o L_{smooth}	.918	.118	.921	.114	.935	.112	.911	.110
CTU + CDU	.901	.135	.908	.125	.921	.121	.868	.118
CTU	.889	.142	.878	.135	.901	.131	.831	.148
CC + Conv	.803	.179	.821	.173	.816	.182	.780	.188
2-FC + C + 2-FC	.674	.258	.499	.355	.779	.322	.686	.243
2-FC + C + FC	.691	.233	.506	.358	.787	.316	.687	.240
FC + C + 2-FC	.673	.261	.500	.360	.774	.346.	.683	.249
FC + C + FC	.681	.271	.497	.355	.785	.315.	.692	.246
RC + FC	.671	.276	.489	.357	.780	.318	.685	.251

we may take our network prediction for the lighting change associated with Φ_i^{light} combined with the azimuth change associated with Φ_j^{azim} to be:

$$\begin{aligned} \widehat{y} \;&=\; \text{Decode}[\,\widehat{l_y}\,] \;\; \text{where} \\ \widehat{l_y} \;&=\; \Phi_i^{light} \circ \Phi_j^{azim}(l_x) \;=\; \Phi_i^{light}\big[\Phi_j^{azim}(l_x)\big] \end{aligned} \tag{3.9}$$

3.4.4 Near-Continuous Attribute Modification

Near continuous attribute modification is also possible within the proposed framework; this can be performed by a simple, piecewise-linear interpolation procedure in

Table 3.4.
Quantitative results for light direction and age modification on the synthetic face dataset.

	Light Direction		Age	
Model	SSIM	L_1	SSIM	L_1
Ours	**.941**	**.093**	**.925**	**.102**
CVAE-GAN	.824	.209	.848	.166
CAAE	.856	.270	.751	.207

Table 3.5.
Quantitative results for azimuth and elevation modification on the synthetic face dataset.

	Elevation		Azimuth	
Model	SSIM	L_1	SSIM	L_1
Ours	**.923**	**.107**	**.923**	**.108**
CVAE-GAN	.864	.158	.863	.180
CAAE	.777	.175	.521	.338

the latent space. For example, we can train 9 CTU mappings $\{\Phi_k\}_{k=0}^{8}$ corresponding to incremental °7 changes in elevation $\{\theta_k\}_{k=0}^{8}$. The network predictions for an elevation change of $\theta_0 =$ °0 and $\theta_1 =$ °7 are then given by the values Decode$[\Phi_0(l_x)]$ and Decode$[\Phi_1(l_x)]$, respectively (where l_x denotes the encoding of the input image). To predict an elevation change of °3.5, we can perform linear interpolation in the latent space between the representations $\Phi_0(l_x)$ and $\Phi_1(l_x)$; that is, we may take our network prediction for the intermediate change of °3.5 to be:

$$\widehat{y} \;=\; \text{Decode}[\,\widehat{l_y}\,] \;\;\text{where}\;\; \widehat{l_y} \;=\; 0.5 \cdot \Phi_0(l_x) + 0.5 \cdot \Phi_1(l_x) \tag{3.10}$$

More generally, we can interpolate between the latent CTU map representations to predict a change θ via:

$$\widehat{y} \,=\, \text{Decode}[\,\widehat{l_y}\,] \;\; \text{where} \;\; \widehat{l_y} \,=\, \lambda \cdot \Phi_k(l_x) + (1-\lambda) \cdot \Phi_{k+1}(l_x) \tag{3.11}$$

where $k \in \{0, \ldots, 7\}$ and $\lambda \in [0, 1]$ are chosen so that $\theta \,=\, \lambda \cdot \theta_k + (1-\lambda) \cdot \theta_{k+1}$. In this way, the proposed framework naturally allows for continuous attribute changes to be approximated while only requiring training for a finite collection of discrete changes. Qualitative results for near continuous attribute modification on the synthetic face dataset are provided in Figure 3.12; in particular, we note that views generated by the network effectively model gradual changes in the attributes without any noticeable degradation in quality. This highlights the fact that the model has learned a smooth latent space structure which can be navigated effectively by the CTU mappings while maintaining the identities of the original input faces.

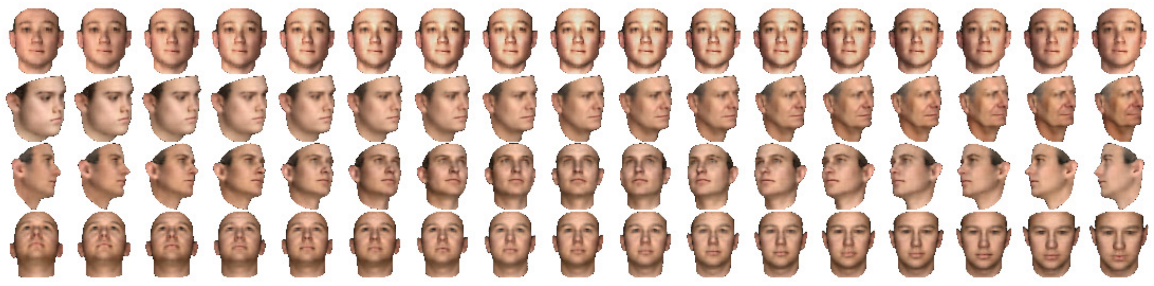


Fig. 3.12. Near continuous attribute modification is attainable using piecewise-linear interpolation in the latent space. Provided a gray-scale image (corresponding to the faces on the far left), modified images corresponding to changes in light direction (first), age (second), azimuth (third), and elevation (fourth) are produced with 17 degrees of variation. These attribute modified images have been produced using 9 CTU mappings, corresponding to varying degrees of modification, and linearly interpolating between the discrete transformation encodings in the latent space.

3.5 Conclusion and Discussion

In this work, we have introduced an effective, general framework for incorporating conditioning information into inference-based generative models. We have proposed a modular approach to incorporating conditioning information using CTUs and a consistency loss term, defined an efficient task-divided decoder setup for deconstructions the data generation process into manageable subtasks, and shown that a context-aware discriminator can be used to improve the performance of the adversarial training process. The performance of this framework has been assessed on a diverse range of tasks and shown to perform comparable with state-of-the-art methods while reducing computational operations and memory consumption.

4. FUSING SURFACE AND VOLUMETRY OF OBJECTS

Technical advancement of 3D printing, virtual reality, and augmented reality has greatly increased the interest of handling three-dimensional shapes such as three-dimensional object synthesis [16,19,20], reconstruction [55,70], and classification [126], which has been deeply studied in computer design communities [58,127–130]. Emergence of neural networks and creation of large-scale three-dimensional object datasets [52, 56] inspired researchers to rediscover three-dimensional object representation learning and synthesis with view-based projections [45], polygon meshes [131,132], point clouds [16,133], and voxelized three-dimensional objects in voxel grids [26,134]. In this work, we utilize complementary properties of surface and volumetric representa-

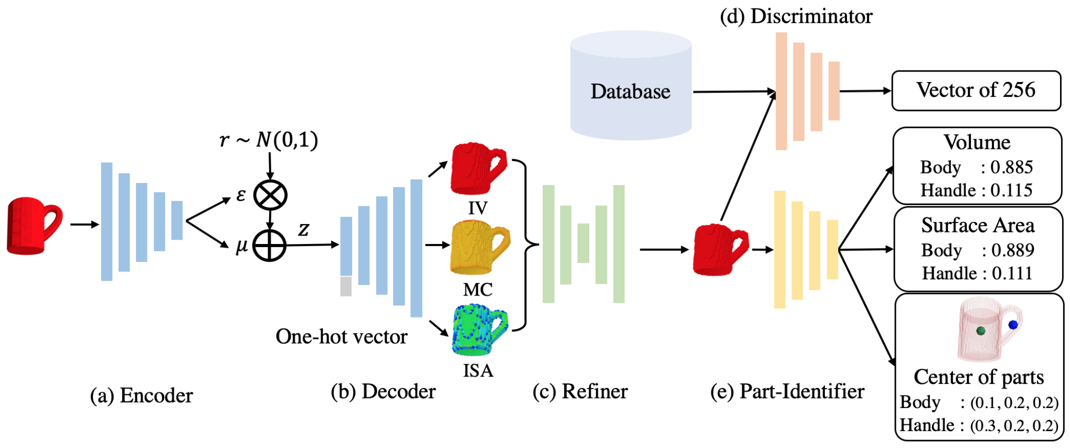


Fig. 4.1. Overview of PG-Net. The encoder encodes IV of the objects, and the decoder synthesizes IV, MC, and ISA. A diverse 3D object from a specified category is synthesized given a one-hot encoded class vector and a noise vector from the normal distribution. During the test stage, the encoder is removed from the pipeline. Noise vector **z** is concatenated with a one-hot vector.

tions to learn the features of shapes with a proposed framework named Part Geometry Network (PG-Net) that synthesizes realistic objects as a conditional generative model.

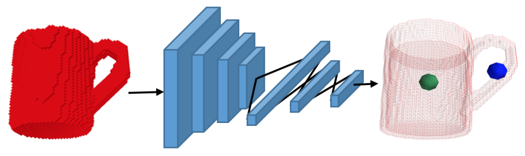
Many works [5, 6, 135, 136] have showed that jointly solving multiple tasks, named multi-task learning [7], helps improving generalizability of estimation models. From this motivation, we adopt multi-task learning to optimize PG-Net.

Surface representation imposes boundary and connectivity information of each part of an object [8–10], and volumetric representation determines interior geometry which is used for heat flow calculation [11, 12]. However, using these two representations for multi-task learning has not been well-examined for learning compact object representations and synthesizing objects. Surface and volumetric representations of objects contain unique features that can be complementary.

We used these modalities for multi-task learning to enhance generalizability of the model by designing the model to estimate surface and volumetric representations with the encoder-decoder structure. Knowing both surface and volumetric representations is crucial for learning not only a perceptual set of attributes but also the connectivity of each part of objects and the details of local interior regions, hence reducing defects in synthesized objects. In this way, the model learns surface properties to learn the interior volumetric representations of objects and vice-versa [7].

Part geometry of objects is critical to learn object distribution with parametric models [53,137,138] since objects are combinations of specific parts. However, learning part geometry is not trivial because defining the boundary of each part is a complex problem since parts can be connected in many different ways.

Each part has their own unique shape and location for their purpose [139], which creates unique part geometry. Therefore, we propose Part-Identifier which learns part geometry for learning each part as depicted in Figure 4.2 (a). Part-Identifier shares the part geometry information with other networks through back-propagating the meaningful gradients during training. With the understanding of part geometry, PG-Net can generate realistic objects.



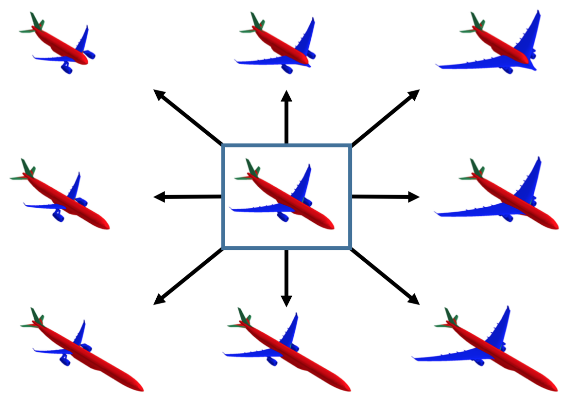


Fig. 4.2. Part-Identifier and part interpolation. (a) Part-Identifier is optimized to predict the location, volume, and surface area of parts. (b) Mesh models are expanded by relocating vertices of parts that expand searching space of optimization process.

However, current datasets [52, 56] have been assembled by collecting individual objects, and therefore, each part of the objects are unique. For this reason, the datasets can not effectively share the information of parts among similar models. This makes it hard to learn connectivity between each part in the objects. To alleviate the shortage of current datasets, we expanded the dataset [140] which has part-labeled triangle mesh models by reshaping specific parts of objects as shown in Figure 4.2 (b). We also demonstrated ablation studies and comparison experiments with other methods. We performed an object synthesis with a one-hot encoded class vector and a vector from normal distribution. For evaluating shape representations obtained from PG-Net, we performed two applications: object reconstruction and classification. From our experiments our proposed method outperformed the other state-of-the-art methods in three-dimensional object synthesis [24, 39], reconstruction [70], and classification [16, 23, 141–144].

Intersected Surface Area (ISA), which calculates intersected area within a cubic voxel for all voxels in defined voxel grids, was introduced by Yarotsky [131]. ISA

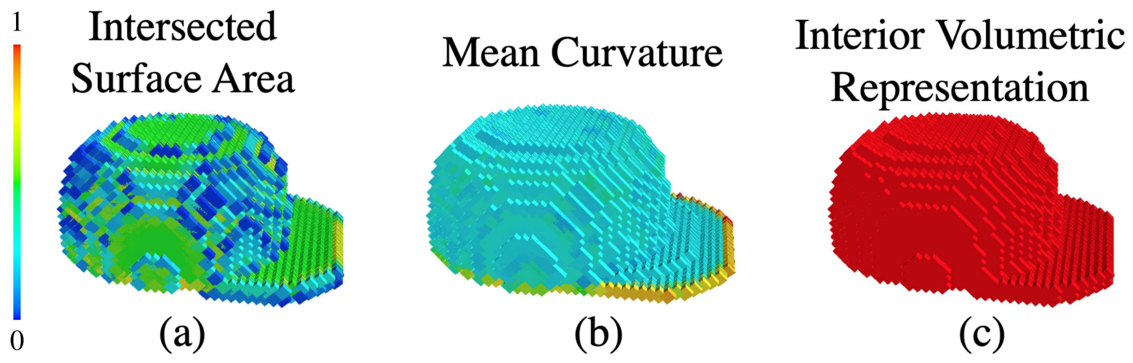


Fig. 4.3. Object representation modalities. (a) is the Intersected Surface Area (ISA) in a cubic voxel, (b) is the final representation in voxel grids of Mean Curvature (MC), and (c) is the Interior Volumetric (IV) representations of an object.

Table 4.1.
Complementary properties of three different modalities.

Properties	IV	MC	ISA
Surface area information			✓
Local curvature		✓	
Volume information	✓		

contains the surface area value of each voxel grid as depicted in Figure 4.3 (a). The Mean Curvature (MC), noted as H, is the mean value of maximum and minimum curvatures, which are extrinsic measures of curvature. We also tested 2-ring and 3-ring neighborhoods of vertices for mean curvature calculation in triangle mesh models, but the result was the best when using a 1-ring neighborhood. Therefore, we used 1-ring neighborhood for mean curvature and assigned the values into voxel grids, which are illustrated in Figure 4.3 (b).

Interior Volume (IV) is a collection of Boolean values in voxel grids as shown in Figure 4.3 (c). In IV, if the cubic voxel is enclosed by the surface of an object;

then, the voxel value is assigned as true. These three modalities have complimentary properties, and it is listed in Table 4.1.

Generator is the combination of the decoder and refiner. H and isa are mean curvature and intersected surface area in a voxel, respectively. pl_i, pv_i and ps_i are a $\{x, y, z\}$ Cartesian coordinate of central location, volume, and surface area, respectively, where the lower index $i \in \{\text{Body, Wheel, ... Legs}\}$ is the index of each part of objects. The noise vector $\mathbf{z} \in \mathbb{R}^K$ is a vector of $\mu + r \odot exp(\epsilon)$, where $r \sim N(0, 1)$, N is normal distribution, and $\odot$ is an element-wise multiplication. We experimentally found that the model performs best when K is 200. The performance of the model is slightly lower than when K is 100 or 300 as listed in Table 4.4. The normal distribution minimizes the mode collapse problem of GAN by incorporating VAE [28]. μ and ϵ are the mean and covariance of the latent vector from the encoder. p_d is data distribution, and p_z is a prior distribution of the decoder. $\mathbf{v} \in \mathbb{B}^{n^3}$,where n determines the resolution of the voxel grids, represents an object of Boolean voxels from true data distribution. All values of $\mathbf{v}$ are initialized as zeros and iterates over each voxel to assign one if a voxel is located at the inside of the closed surface or intersected with the surface of the object.

4.1 Learning Strategies

4.1.1 Multi-task Learning

In PG-Net, the decoder estimates three modalities as multiple tasks. PG-Net estimates MC and ISA which are surface modalities because recent works [145, 146] showed that surface properties are informative for data-driven representation learning. As shown in section 4.4.2, unlike a volumetric-representation-alone framework, consolidating surface knowledge penalizes inaccurate surface estimates. Therefore, we designed the decoder to estimate MC and ISA of objects along with IV, which is illustrated in Figure 4.1 b. We further experimented with surface normal vectors, but the result was not better than using curvature and surface area information together.

This is because surface normal vectors are sensitive to orientation. As one of the tasks, the decoder was optimized to learn the surface representation by minimizing the cost function $\mathcal{L}_{surf}$:

$$\sum_k ||\widehat{isa} - isa||_1 + \lambda \cdot \sum_k ||\widehat{H} - H||_1 \tag{4.1}$$

, where λ is a hyper-parameter and $k \in \{1, 2, 3 \ldots n^3\}$. n^3 is the total number of voxels in the voxel grid. Therefore, n represent the resolution of the voxel grids. isa and H are intersection area and mean curvature per each voxel in the voxel grid, respectively. $\widehat{isa}$ and $\widehat{H}$ indicate that the values are sampled from the decoder.

Another task is estimating volumetric representation of objects. PG-Net learns interior volume information of objects by minimizing sum of Sigmoid Cross-Entropy loss $\mathcal{L}_{vol}$:

$$-\mathbf{v}\log(D(\mathbf{z})) - (1 - \mathbf{v})\log(1 - D(\mathbf{z})) \tag{4.2}$$

In terms of the learning strategy, synthesizing IV, MC, and ISA with a single decoder solves multiple tasks. Expressed mathematically, encoder-decoder is optimized to jointly minimize the cost functions $\mathcal{L}_{multi-task}$:

$$\mathcal{L}_{vol} + \kappa \cdot \mathcal{L}_{surf} \tag{4.3}$$

,where κ is a hyperparameter. Jointly, learning both surface and volumetric representations is an informative way to discriminate objects because they provide complementary information of objects [147]. Therefore, PG-Net uses the knowledge of surface and volumetric representations for learning object distribution in a three-dimensional space.

4.1.2 Objective Function

Adversarial training has been remarkably successful for synthesizing images and objects [29]. The goal of adversarial training is finding equilibrium between a generator and discriminator by playing a minimax game between the generator and discriminator. Motivated by adversarial training, we experimented with an adversarial loss term [14] and found that high fidelity objects are generated. The discriminator discriminates fake objects from model distribution and true objects from the database. To enhance the stability of adversarial training, we used the least square adversarial loss $\mathcal{L}_{gan}$ [148]:

$$\mathcal{L}_{discriminator} = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[(\mathcal{D}(R(D(\mathbf{z}))))^2] + \mathbb{E}_{\mathbf{v}\sim p_{data}(\mathbf{v})}[(\mathcal{D}(\mathbf{v})-1)^2] \tag{4.4}$$

$$\mathcal{L}_{generator} = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[(\mathcal{D}(R(D(\mathbf{z})))-1)^2] \tag{4.5}$$

$$\mathcal{L}_{gan} = \mathcal{L}_{discriminator} + \mathcal{L}_{generator} \tag{4.6}$$

Additionally, to synthesize objects given $\mathbf{z}$, we minimize the gap between $p_{\mathbf{z}}(\mathbf{z})$ which is from normal distribution and three-dimensional object distribution in the latent space [28] by minimizing $\mathcal{L}_{KL}$:

$$\mu^T\mu + sum(exp(\epsilon) - \epsilon - 1) \tag{4.7}$$

These losses are jointly minimized in $\mathcal{L}_{total}$ which is shown in Equation 4.10 and minimizing process is described in Section 4.3.

The refiner is composed of an hourglass architecture [149] which is shown in Figure 4.1 c and refines coarse objects from the decoder by consolidating estimated three modalities. We found that sometimes the estimation of three modalities are not accurate or consistent. To avoid these problems, we designed the refiner to penalize poorly estimated modalities from the decoder with Sigmoid Cross-Entropy loss $\mathcal{L}_{ref}$:

$$-\mathbf{v}\log(R(D(\mathbf{z}))) - (1-\mathbf{v})\log(1-R(D(\mathbf{z}))) \tag{4.8}$$

For training the refiner, we jointly trained it with Part-Identifier by optimizing the cost function $\mathcal{L}_{total}$:

$$\alpha \cdot \mathcal{L}_{gan} + \beta \cdot \mathcal{L}_P + \gamma \cdot \mathcal{L}_{ref} + \mathcal{L}_{KL} \tag{4.9}$$

$\mathcal{L}_P$ is defined as Equation 4.10. The refiner and discriminator are optimized with $\mathcal{L}_{total}$.

4.2 Object Geometry Constraints

Part geometry is an important factor to understand three-dimensional object space [71]. From this motivation, we propose Part-Identifier for learning part geometry, which is shown in Figure 4.1 e. To expand the learning space of each part of an object, we further augmented objects by interpolating the parts of the objects such as wings of an airplane.

4.2.1 Part-Identifier

Part-Identifier in PG-Net estimates part geometry information, and the information is back-propagated through the pipeline of PG-Net. For learning the part geometry, we considered using point-wise segmentation of parts. However, it was computationally expensive since it needs to add an additional decoder module with 3D de-convolutional layers, and therefore, we regressed the center location, surface area, and the volume of each part. The position of the central coordinate, surface area, and the volume of each part are normalized by dividing the resolution of voxel grids, the surface area, and the volume of the object, respectively. Volume and surface area of parts impose meaningful information of objects such as relations of each part. Part-Identifier implicitly learns the relationship between different parts and part geometry by estimating the locations, volumes and surface areas of parts. This module guides the generator towards lower local/global minima by minimizing the $\mathcal{L}_p$:

$$\sum_i ||\widehat{pl_i} - pl_i||_1 + ||\widehat{ps_i} - ps_i||_1 + ||\widehat{pv_i} - pv_i||_1 \tag{4.10}$$

,where $i \in$ {Body, Wheel, ... Legs}, the index of each part. Part-Identifier is optimized by minimizing $\mathcal{L}_p$.

4.2.2 Part expansion

The goal of part expansion is to capture part geometry effectively and broaden the learning space for optimizing PG-Net. In order to create our dataset, named *Expanded*, we augmented Projective dataset [140] which consists of mesh models that were converted from the subset point cloud objects of the Scalable dataset [150]. Scalable dataset consists of 93,000 shape parts with semantic per-point region labels.

Projective dataset has part annotations that every vertices of object are labeled by part indices. Therefore, by relocating the vertices belonging to specific part group into the predefined direction, we augmented data while preserving the overall shape of the original object as shown in Figure 4.

We carefully choose parts an group for part expansion in each class as illustrated in Table 4.2. To preserve the overall shape of an object, we predefined the rescaling direction and constraint of each part. For example, we rescale the top parts of tables to (x, y) plane, and leg part to a direction z axis on Cartesian coordinate. Then we filtered out expanded data based on the constraint that the horizontal surface area of the top part cannot be smaller than the leg part. The detailed directions and scale ratios for part expansion is explained in the supplementary document. Additionally, we defined the sub-groups for specific parts vertices because the expansion direction of each group should be different to preserve the part structure of the object. For instance, the expansion direction of left-wing and right-wing in the Airplane class

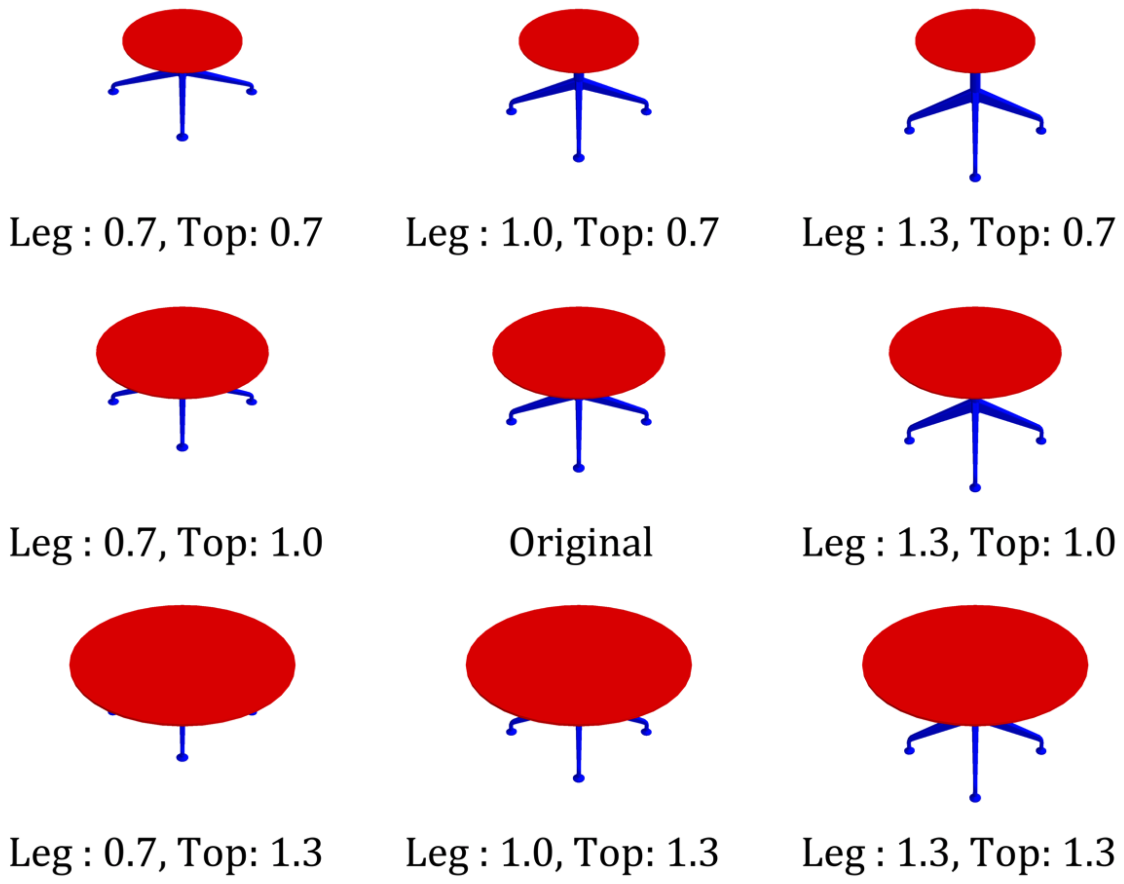


Fig. 4.4. Example of part expansion. Each part of tables is expanded by relocating vertices of parts to share each part information within the object. Parts are numbered based on their scaling proportion.

must be the opposite and the Wheel part of the Car class is a similar case. The vertices in these parts share part labels but have more than one sub-groups. To define different expansion directions for each group in the same parts, we further sub-grouped the vertices within the same group. We applied Density-Based Spatial Clustering of Applications with Noise (DBSCAN) for these parts to grouping the vertices. We filtered out clusters which do not fit into the major group. For example, when we perform DBSCAN on the wing part of airplane class, it must have two clusters. Therefore, we left the largest two groups and filtered out the other groups. After this process, we further manually validated clusters to ensure the structure of object. This process help expansion because one mislabeled vertex ruins the shape of the overall parts while expanding the parts. In the final stage, we manually filtered out odd-looking expanded objects. After the dataset augmentation, we added the Scalable dataset [140] to the *Expanded* to increase the diversity of objects. The statistics of our dataset are detailed in Table 4.3.

Table 4.2.
Parts for *Expanded* dataset creation.

Classes	Interpolated Parts		
Airplane	Body	Wings	
Bag	Handle	Case	
Cap	Crown	Brim	
Car	Roof	Wheel	Hood
Chair	Back	Seat	Leg
Lamp	Legs	Base	Shade
Mug	Cup	Handle	
Table	Top	Leg	

4.3 Architecture Details

Implementation details

We used two NVIDIA TITAN Xp GPUs and an Intel i7-6850K CPU with 64GB of RAM for all of our experiments. The artificial neural network was developed with TensorFlow deep learning framework [151], which was accelerated by the CUDA instruction for the GPU computation. The networks were optimized by using the

Table 4.3.
Statistics of *Expanded*, Scalable [150], and Projective [140].

Classes	Expanded (Ours)	Scalable [150]	Projective [140]
Airplane	36,018	2,690	500
Bag	666	76	76
Cap	477	55	55
Car	12,739	878	500
Chair	13,041	3,746	500
Lamp	13,014	1,546	500
Mug	1,629	184	184
Table	30,015	5,263	500
Total	107,599	14,438	2,815

ADAM optimizer [117] with the initial parameters: learning rate $= 0.0025$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. For the hyper parameters, we used $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 10^{-4}$, $\kappa = 1$, and $\gamma = 0.1$.

We trained PG-Net in two stages with a batch size of 16. In the first stage, we trained the encoder and decoder separately from other networks with the cost function $\mathcal{L}_{multi-task}$, which were converged approximately after 250 epochs. Then we stacked the refiner, discriminator, and Part-Identifier for the second-stage training and optimized the networks by minimizing $\mathcal{L}_{total}$, which required approximately 200 epochs to converge. During the second-stage training, we did not update the encoder and decoder. In order to improve the stability of training the discriminator, we updated the refiner and Part-Identifier twice per each discriminator update to enhance the stability of the learning process [15]. After training the second-stage, we jointly updated all networks with the learning rate of 10^{-7}, which required approximately 100 epochs to converge. We used the initial learning rates of 0.0005 and 0.0025 for the first-stage and second-stage training, respectively. We dropped the running rate by half in every 25 epochs. For object synthesis, we synthesized the objects from $\mathbf{z} \in \mathbb{R}^{200}$ which was sampled from a normal distribution $N(0, 1)$ and a one-hot encoded class vector.

The networks for synthesizing task convergence time of stage one and stage two with Expanded dataset are 31 hours and 98 hours, respectively. Fitting time of SVM classifier is 293 seconds on ModelNet40 and 46 seconds on ModelNet10. The convergence time of the reconstruction application with EPN [70] dataset is 6 hours. The synthesis, classification, and reconstruction inference time per object are 4.41 ms, 2.11 ms, and 6.72 ms, respectively. For inference time calculation, we averaged the time of one thousand runs.

Network architecture specifications

Encoder consists of six Down Squeeze-and-Excitation Blocks (DSEB) shown in the supplementary document with the numbers of output channels as {16, 32, 64, 128, 256, 512}. Zero padding is applied when the size of the output from a previous layer is not divisible by two.

Decoder consists of three 2D convolution layers with bilinear interpolation, kernel sizes {3, 3, 3}, stride sizes {1, 1, 1}, and output sizes {2×2×300, 5×5×150, 10×10×40}. The output from the last 2D convolution layer is passed into three Up Squeeze-and-Excitation blocks shown in the supplementary document with output sizes {64, 32, 3}. Between 2D convolution and 3D convolution layers, we reshape 10×10×40 tensor into 5×5×5×32 tensor.

Refiner is an hourglass structure. An encoder consists of three 3D convolution layers with numbers of channels {8, 16, 32}, kernel sizes {3, 3, 3}, and stride sizes {2, 2, 2}. We add batch normalization layer and Swish [109] activation function in between the convolution layers. A decoder consists of three 3D convolution layers with numbers of channels {32, 8, 1}, kernel sizes {3, 3, 3}, and stride sizes {1, 1, 1}. We add batch normalization layer and Swish in between the convolution layers. 3D interpolation is used to enlarge the output of the convolution layers by a factor of two. Swish is applied in between the interpolation layers and the convolution layers. The outputs from the first and second layers of the encoder are concatenated with the inputs of the third and second convolution layers of the decoder, respectively.

Part-Identifier consists of four DSEB layers with numbers of channels {64, 128, 256, 512}, kernel sizes {3, 3, 3, 3}, and stride sizes {4, 2, 2, 2}. Additionally, two fully connected layers with output numbers {512, 100} are stacked onto the DSEB layers.

Discriminator consists of four DSEB layers with numbers of channels {64, 128, 256, 512}, kernel sizes {3, 3, 3, 3}, and stride sizes {4, 2, 2, 2}. Additionally, two fully connected layers with output numbers {512, 256} are stacked onto the DSEB layers. The Swish activation function is replaced with the LeakyReLu activation function on each DSEB layer.

4.4 Experiments

In this section, we present experimental validations and analysis of three-dimensional object synthesis, classification, and reconstruction to validate the efficacy of PG-Net. We used two standard large-scale three-dimensional object datasets: ModelNet [56] and *Expanded* which is detailed in Table 4.3. For dataset splitting, we divided our dataset into three sets: 70% as a training set, 10% as a validation set, and 20% as a test set. We performed isosurface and Laplacian smoothing for object visualization and used L_1 metric for quantitative evaluation.

4.4.1 Dataset preprocessing

We preprocessed mesh models to ensure the existence of mean curvature per each voxel grid. First, we voxelized the mesh models with the resolution of 40^3 voxel grids, and then we used the marching-cubes algorithm [152] to regenerate triangle meshes since voxelized objects are deformed from the original shape as depicted in Figure 4.5 Remeshed. We further smoothed the meshes with Laplacian smoothing [153] to reduce the variation of mean curvature as shown in Figure 4.5 Smoothed. From the preprocessed meshes, we calculated MC and ISA and assigned in 40^3 voxel grids. For the ISA extraction, we used SurfaceArea operation in a library [131] that computes face areas within each cubic voxel. We obtained MC by calculating the average of the minimum and maximum principal curvatures with 1-ring neighborhood distance. After obtaining all data, we normalized and scaled them to be within the interval $[-1, 1]$.

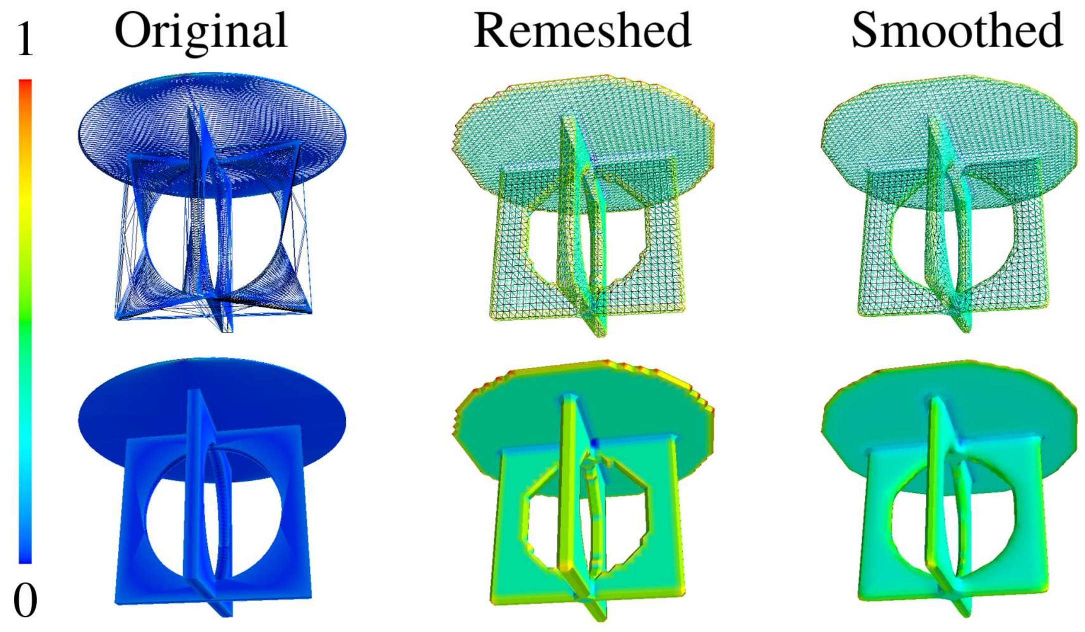


Fig. 4.5. Comparing an original mesh with an isosurface from a voxelized object. The first column shows mean curvatures from an original mesh. The second column shows mean curvatures from the mesh generated by marching-cubes algorithm given a voxelized object in 40^3 grids. The third column shows mean curvatures from the smoothed mesh by applying Laplacian smoothing with the second column mesh.

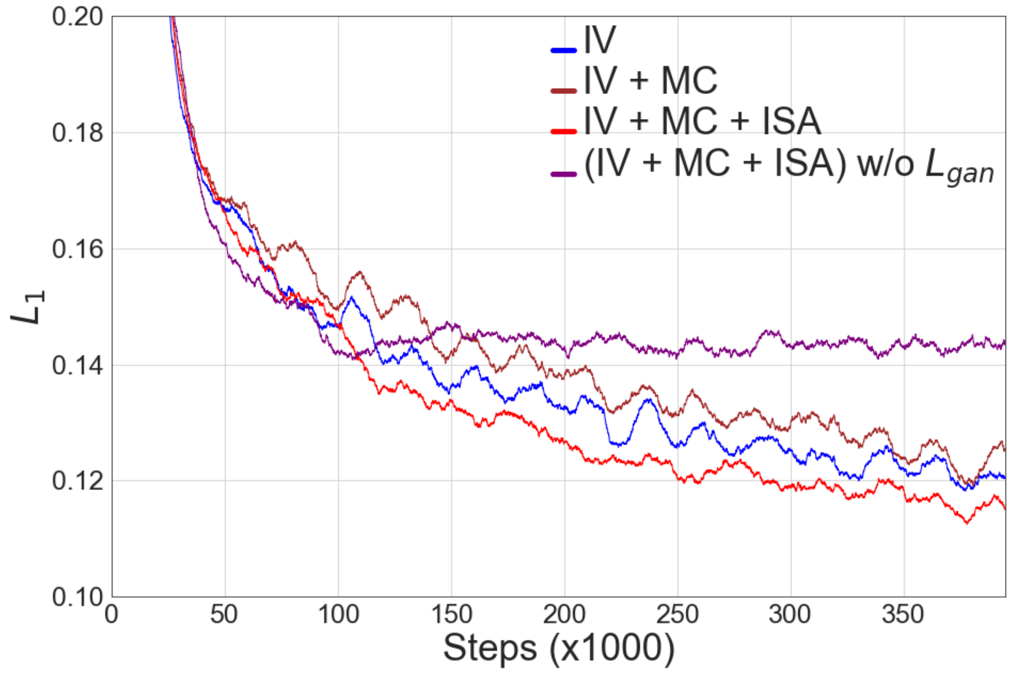


Fig. 4.6. L_1 loss plots of four experimental models on the test dataset.

4.4.2 Ablation study

Does multi-task and adversarial learning lead to the synthesis of better quality objects? For the ablation study, we ran four experiments: (i) IV, which only estimates volumetric representations of objects for training; (ii) IV+MC, which estimates IV and MC; (iii) IV+MC+ISA, which estimates IV, MC and ISA; (iv) (IV+MC+ISA) w/o L_{gan}, which is the same as (iii) but without L_{gan}. Figure 4.6 shows the performances of all experiments, quantitatively, and the qualitative results in Figure 4.7 indicate that the network learns a structural correlation across the local surface and volume descriptors to improve the fidelity of final outputs. This ablation study validates the rationale of multi-task and adversarial learning with IV, MC, and ISA modalities.

Why learn the geometry of parts? Furthermore, we explored the efficacy of learning part geometry as illustrated in Figure 4.8. Table 4.4 shows the quantitative evaluation of the proposed method for the following baselines: (i) *w/o Part-Identifier*, which does not have the part identifier; (ii) *w/o Part interpolation*, which is the same pipeline as PG-Net but uses dataset without data augmentation with part interpolation; (iii) *w/o Refiner*, which is trained without the refiner to evaluate the efficacy of the refiner. From the result shown in Table 4.4, we verified that *part identifier*

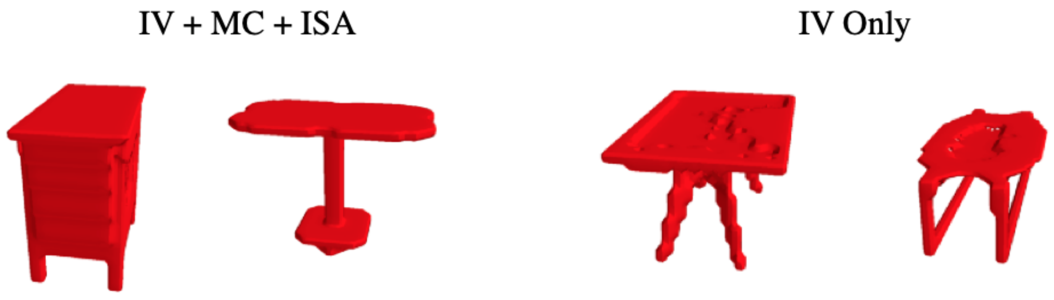


Fig. 4.7. Effect of the multi-task learning with surface and volumetric representations on object synthesis. The sampled objects on the left are from the the model which was trained with IV + MC + ISA modalities, and the sampled objects on the right are from the model which was trained with IV only.

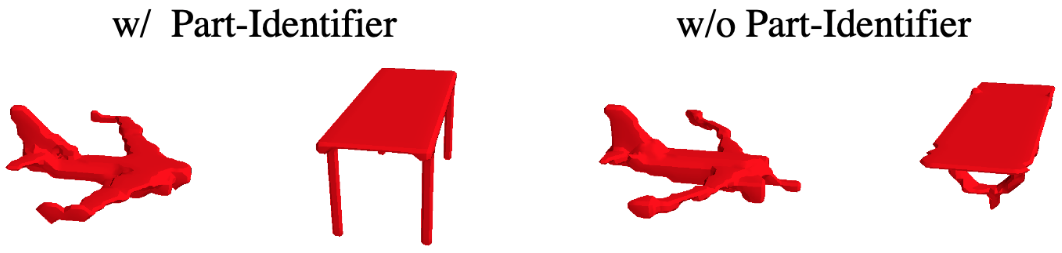


Fig. 4.8. Effect of the part geometric learning on object synthesis. Sampled objects from the models w/ Part-Identifier and w/o Part-Identifier given normal distribution and a one-hot encoded class vector.

Table 4.4.
Quantitative results as L_1 metric of baselines and other methods for 3D object generation without 3D object references. Lower value is better. k is the length of the latent vector $\mathbf{z}$. k is set as 200 in PG-Net where @k is not placed.

	Seen	Unseen
3D-CIWGAN [24]	0.225	0.232
3D-CVAE [39]	0.173	0.199
PG-Net w/o Part-Identifier	0.164	0.194
PG-Net w/o Part Interpolation	0.183	0.204
PG-Net w/o Refiner	0.112	0.135
PG-Net @k=300	0.088	0.123
PG-Net @k=200 (Ours)	**0.083**	**0.117**
PG-Net @k=100	0.090	0.131

improves the quality of synthesized objects since the L_1's of our *PG-Net* are lower than those of *w/o Part-Identifier*. Also, our PG-Net gave lower metric scores than *w/o* Part interpolation and *w/o Refiner*, which directly shows that the refiner and part interpolation results in learning robust shape representations. As a conclusion, PG-Net performs better than the other baselines, and each of our proposed methods reduces artifacts and holes of synthesized objects.

4.4.3 Synthesizing 3D objects

We conditionally generated 3D objects from latent vectors and one-hot encoded class vectors. We compared our method with 3D-CVAE [39] and 3D-CIWGAN [24]. For 3D-CVAE, we followed the CVAE base model in [39] and used the same encoder/decoder definitions as PG-Net. We combined 3D-IWGAN [24] and CGAN [3] for 3D-CIWGAN. All methods used *Expanded* with a one-hot encoded class vector

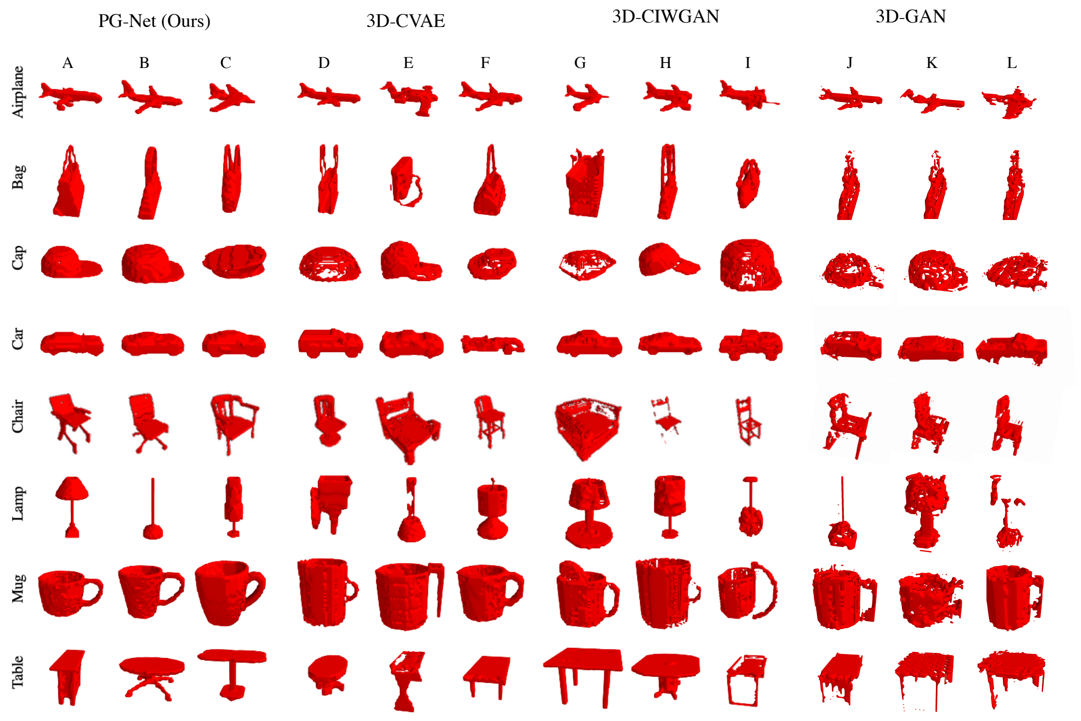


Fig. 4.9. Objects generated from the model given a one-hot encoded class vector and a vector from the normal distribution without a reference object. The resolution of the voxel grids was 40^3, and the objects were binarized by the threshold of 0.5. We generated fifty objects per class and randomly sampled three objects.

as a condition. For qualitative evaluation, we first generated 100 objects per class and randomly sampled three objects. Figure 4.9 shows the synthesized objects from our method and the others. Since we were sampling objects from noise distribution without ground truth, displaying the exact same objects for each method was impossible. Each voxel value of the synthesized objects was binarized with a threshold of 0.5. We visualized volumetric data after applying the marching-cubes algorithm and Laplacian smoothing. To assess the performance of PG-Net, we have randomly sampled objects from each class and compared the results with objects generated using alternative methods. The objects generated using PG-Net are seen to possess far

Table 4.5.
Evaluating diversity of generated objects from the networks with inception scores on each class. Higher number is better.

Method	Inception Score
3D-CVAE [39]	6.12
3D-CIWGAN [24]	6.23
3D-GAN [56]	5.91
PG-Net(Ours)	**6.51**
Ground	8.21

fewer holes and artifacts than the objects generated using other existing methods. As shown in Figure 4.9, for example, both the 3D-CVAE and 3D-CIWGAN models produce objects from the "cap" class which contain unrealistic holes. The 3D-CIWGAN results for the "chair" class are also seen to contain a substantial amount of artifacts, particularly in columns G and H. However, our proposed PG-Net method does not suffer from these issues and produces objects from all classes which do not contain unrealistic holes or any substantial artifacts. 3D-CIWGAN is not able to synthesize the nose of the airplane, and 3D-CVAE was effective on airplanes but had many holes in the table class. Unlike 3D-CVAE and 3D-CIWGAN, PG-Net preserved part geometry and well-defined surfaces. We also evaluated the diversity of the generated objects with inception score [110] as listed in Table 4.5. Our method achieved the highest scores than the others, which represents that our method generates more diverse objects than the other methods.

4.4.4 3D Object classification

We evaluated object representations from PG-Net by performing 3D object classification on the ModelNet [56], which offers two kinds of datasets: ModelNet10 and

Table 4.6.
The comparison on classification accuracy between PG-Net and the other unsupervised methods. Higher number is better.

Method	ModelNet10	ModelNet40
SPH [141]	79.8%	68.2%
Vconv-DAE [142]	80.5%	75.5%
ECC [143]	90.0%	83.2%
3D-GAN [23]	91.0%	83.3%
3D-DescripNet [19]	92.4%	83.8%
Primitive GAN [154]	92.2%	86.4%
FoldingNet [144]	94.4%	88.4%
GMPC [16]	95.4%	84.5%
PG-Net (Ours)	**95.6**%	**89.1**%

ModelNet40. ModelNet10 and ModelNet40 comprises 4,899 objects and 10 classes and 12,331 objects and 40 classes, respectively. For unsupervised training, we used the same method and a dataset for fine-tuning PG-Net from Achlioptas et al. [16]. The dataset consists of 57,000 objects from 55 categories in ShapeNet [52]. We fine-tuned the optimized PG-Net without Part-Identifier since the parts' labels are not available in the dataset. After fine-tuning PG-Net, we extracted features from the last two layers of encoder and concatenated them to create high-dimensional representations. Then we classified the representations with a linear SVM trained on the 3D classification benchmark [56]. Our method outperforms other existing works which are shown in Table 4.6. From the results, PG-Net extracts robust features which can effectively distinguish objects, and therefore, PG-Net can be used as a feature descriptor for the object analysis and other applications.

4.4.5 Object reconstruction

We performed three-dimensional object reconstruction to evaluate the efficacy of PG-Net and compared the results with 3D-EPN [70]. For our experiment, we used a dataset with 5 classes with around 25,000 objects from the 3D-EPN project. We interpolated the input objects from 32^3 voxel grids into 40^3 voxel grids to match the input resolution of PG-Net. For the quantitative evaluation, we converted the reconstructed objects into Boolean voxels with the threshold of 0.5. Then we counted wrongly-estimated voxels and divided with the total number of occupied voxels in ground truth. We used the pre-trained weights of EPN-unet w/ class version from the 3D-EPN project page as a comparison. The quantitative results of PG-Net and 3D-EPN are compared in Table 4.7. The error values of PG-Net are lower than those of 3D-EPN. As shown in Figure 4.10, PG-Net also shows better qualitative results as compared to 3D-EPN. The quantitative and qualitative experiment results suggest that PG-Net outperforms 3D-EPN in a large margin along the all classes. Therefore, PG-Net trained with multi-task and part geometry learning methods effectively learns object distribution and nicely reconstructs three-dimensional objects.

Table 4.7.
Comparison of the quantitative results of reconstruction task. Lower value is better.

Class (# of train / # of test)	3D-EPN [70]	PG-Net (Ours)
Air. (3.3K / 0.8K)	0.226	**0.202**
Car (5K / 1K)	0.197	**0.191**
Chair (5K / 1K)	0.309	**0.273**
Lamp (1.8K / 0.5K)	0.407	**0.392**
Table (5K / 1K)	0.338	**0.251**
Total (20.1K / 4.3K)	0.286	**0.249**

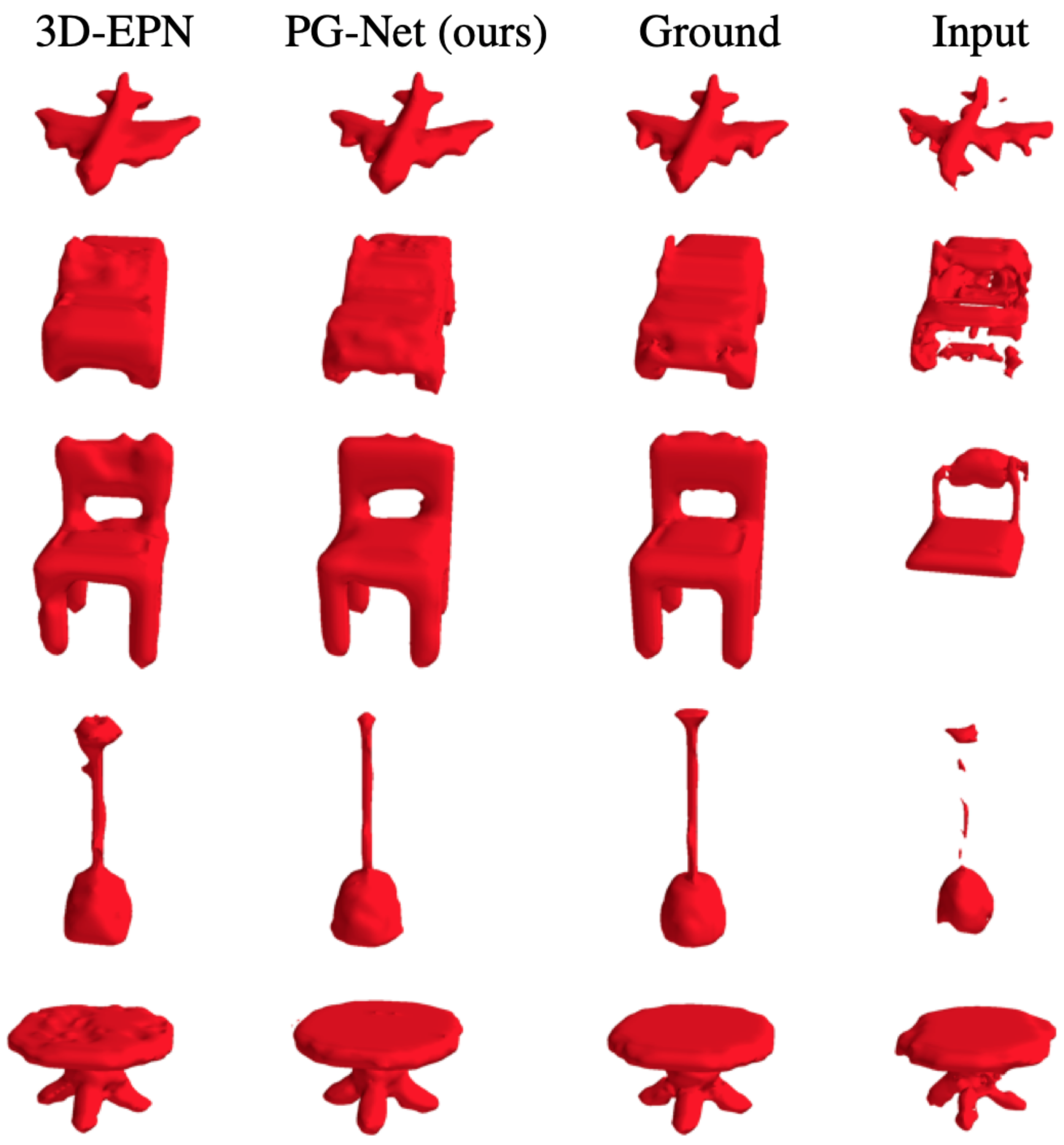


Fig. 4.10. Comparison of the qualitative results of three-dimensional object reconstruction experiment.

4.5 Conclusion and Discussion

In this section we propose a conditional generative model PG-Net which is optimized with multi-task learning and part geometry learning for object synthesis.

Results from our study suggest that multi-task learning increases the fidelity of generated objects and that learning part geometry enhances the realism of each part of the synthesized objects. PG-Net exceeded the other state-of-the-art methods in object synthesis, classification, and reconstruction. As limitations, 3D convolution layers with voxels are computationally expensive and thus require ample memory. However, these limitations can be solved by using a recurrent neural network with polygonal mesh vertices or point clouds representations. For a future work, fusing surface and volumetric representations as an input with point clouds representation could be further explored with a recurrent neural network.

5. MECHANICAL COMPONENTS BENCHMARK

The application of machine learning is highlighted recently due to the improved effectiveness of the deep neural networks [53, 133, 136, 155–157]. Along with deep neural networks, data driven algorithms and the creation of a large-scale datasets [52, 56, 158, 159] have led to a series of breakthroughs in computer vision [14, 123, 160] and graphics [40, 161, 162]. The development of ImageNet [158], which used the class hierarchical structure from WordNet [163] to maximize the dataset coverage, showed that a well-structured and annotated dataset is crucial for developing geometric feature descriptors. The pre-trained deep neural network descriptors using ImageNet have been widely used to extract low-dimensional representations that are used in tasks of object detection [155, 164, 165], semantic segmentation [72, 76, 122, 166], image caption generator [167, 168], and image retrieval [44, 45]. The creation of a large-scale mechanical components dataset with well-organized hierarchical classes and annotations is needed for developing and benchmarking geometric feature descriptors in the manufacturing industry [58, 169]. Geometric features extracted from the descriptors are fundamental cues to retrieve objects given the query object and classifying objects given image, volumetric representation, or point clouds.

However, in the manufacturing, design, and supply chain areas, the classification of mechanical components with deep neural networks has not been addressed due to the lack of large-scale annotated datasets. Without a standardized dataset, it is difficult to develop and compare learning algorithms on mechanical components [51].

Creating a large-scale mechanical component dataset is challenging due to the significant difficulty of collecting 3D CAD models of mechanical components. Different from common-object datasets [51–53, 56], the accessibility of most mechanical components is limited because of proprietary and ownership issues with specially designed models. Products and manufacturing models are held by companies for commercial usages, resulting in a deficiency of open-source components datasets. The inconsistency and incompatibility of mechanical components from available sources require massive effort on filtering and annotating the data. Also, annotating mechanical components is harder than common objects since it demands more knowledge and expertise from annotators to properly annotate engineering components.

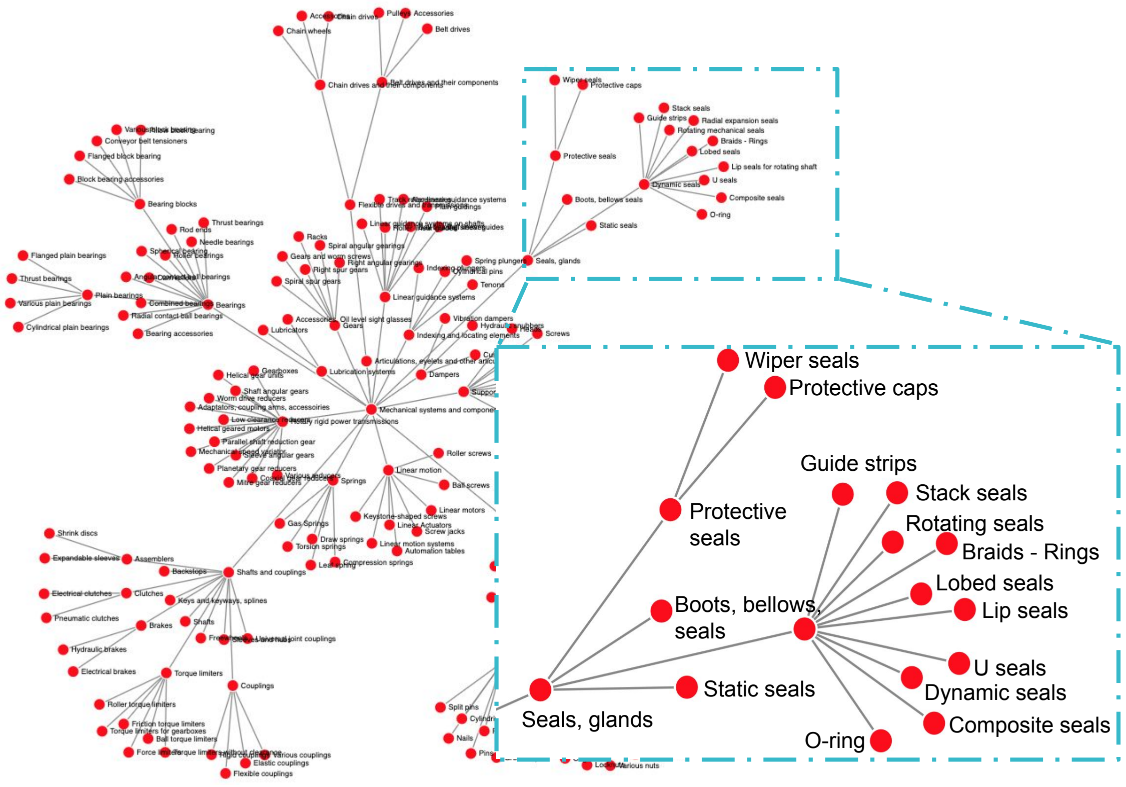


Fig. 5.1. The hierarchy taxonomy of mechanical components based on the International Classification for Standards.

To resolve this difficulty, we established a hierarchical semantic taxonomy as a guideline based on the International Classification for Standards (ICS) published by the International Organization for Standardization (ISO). The tree structure of our proposed hierarchical taxonomy. Details are provided in the supplementary document. To collect annotations, we developed a web application which reduce the difficulty of filtering and annotating (Figure 5.3). This application supports controllable viewing, displaying meta-information, annotation, and filtering parts by viewing multiple parts as a tabular to visually see the consistent shape features within the same class rather than viewing each individual part. These functionalities make the benchmark creation faster and more accurate for fine-grained categories.

Furthermore, we benchmark seven state-of-the-art shape descriptors to analyze the properties of mechanical components. Seven methods are carefully selected from three different 3D object representations: (1) point clouds, (2) voxel girds, and (3) view-based. From the benchmark results, the input representation is not the core factor that determines the performance. DLAN [47], which uses voxel grids, and PointCNN [41], which uses a point cloud input representation that focuses on local shape features, perform relatively well on both retrieval and classification tasks. The view-based methods are not robust on unseen orientation in shape retrieval tasks, which is also observed in common object retrieval tasks [54]. However, the descriptors [46,47,133] show significantly different results from common object classification tasks, which indirectly indicates that topological and geometrical characteristics of mechanical components are different from common objects. We report micro- and macro-precision, recall, F-score, mAP, and NDCG to evaluate retrieval tasks. For classification tasks, we report accuracy per class, accuracy per instance, F1 score, and average precision.

5.1 Properties of Mechanical Components

Mechanical components, shown in Figure 5.4, have sharp edges, well-defined surfaces, and high genus, which distinguishes them from common objects. Since the shape of mechanical parts represents their physical functions, the functionality of machine elements is sensitive to small details, resulting in the difficulty in annotation. Therefore, mechanical components are often categorized by their detailed shape, whereas common objects are mainly identified by their general shape. The shape and location of detail features often determine the function of engineering parts.

The shape of the detail features and the function of engineering parts are usually interdependent. For example, the only difference in shape between a thrust washer and a lock washer is the split detail feature, as seen in Figure 5.2 (a), but they possess distinct functionality. A thrust washer spreads fastener loads, while a split lock washer uses the split feature to lock a nut and bolt in place. In another case, a hex nut and a lock nut share a hexagonal shape. However, the lock nut has an additional circular feature that houses a nylon insert, as seen in Figure 5.2 (b). A hex nut mates with a bolt to fasten materials together, and while the lock nut performs a similar function, the nylon insert keeps the nut from coming loose from the bolt. In another application, a spur gear transfers power to other toothed surfaces, while a timing pulley transfers power to a timing belt. Both parts' shapes and functions are similar, but small details in tooth shape and design differentiate the two parts, as seen in Figure 5.2 (c). In contrast, changing the shape of common objects, like using longer legs on chairs, may not change the function of the object. Because these characteristics do not appear in common object datasets [52, 56], the existing shape descriptors [41, 54, 133] need to be benchmarked on MCB to explore the shape descriptors on mechanical components. This is because recently-proposed deep neural networks descriptors are developed to capture the features from the common objects but not validated on the mechanical components. In this sense, Koch et al. [61] created a large CAD model dataset and benchmarked surface normal estimation, but they could not benchmark object classification or shape retrieval because they are not labeled.

An annotated benchmark dataset such as MCB can link the shape to the particular representation inside product data management systems of CAD kernels. Our work opens up ways for implementation of fine-grained searches with features of mechanical components, semantic text, and mechanical meta-data.

Table 5.1.
Comparison table of the MCB dataset with other datasets. CO and MC stands for common objects and mechanical components, respectively. ShapeNetCore, ShapeNetSem, and PartNet use models from the ShapeNet.

Dataset	# Class	# Models	Type
ModelNet [56]	40	12,311	CO
ShapeNet [52]	3,135	+3,000,000	CO
ShapeNetCore	55	51,300	CO
ShapeNetSem	270	12,000	CO
PrincetonSB [51]	92	6,670	CO
PartNet [53]	24	26,671	CO
ABC [47]	N/A	+1,000,000	MC
AAD [59]	9	180	MC
ESB [58]	45	867	MC
MCB (Ours)	68	58,696	MC

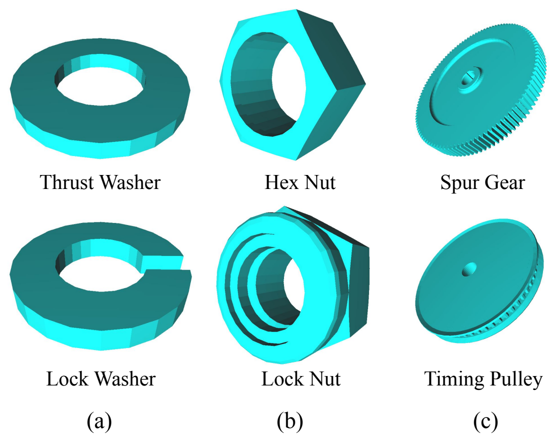


Fig. 5.2. Examples of detail features making categorical changes.

5.2 Dataset Creation

For the dataset creation, we first elaborate on the acquisition of mechanical components and explain how we annotated them. We acquire models from online 3D CAD repositories. To effectively annotate, CAD models are filtered and annotated using web-based tools. We define classes by following the field "Mechanical Systems and Components" of the International Classification Standard (ICS).

5.2.1 Data acquisition

We collect mechanical components from online large 3D CAD repositories: TraceParts[1], 3D Warehouse[2], and GrabCAD[3]. 3D Warehouse and GrabCAD are large online open repositories for professional designers, engineers, manufacturers, and students to share CAD models. They provide numerous CAD models with various classes, including mechanical components. The models from TraceParts are industry standard components and shape variation within class is small. By merging the

[1]https://www.traceparts.com/
[2]https://3dwarehouse.sketchup.com/
[3]https://grabcad.com/

models from different sources, we obtained 163K mechanical components before annotation and purification as shown in Table 5.2.

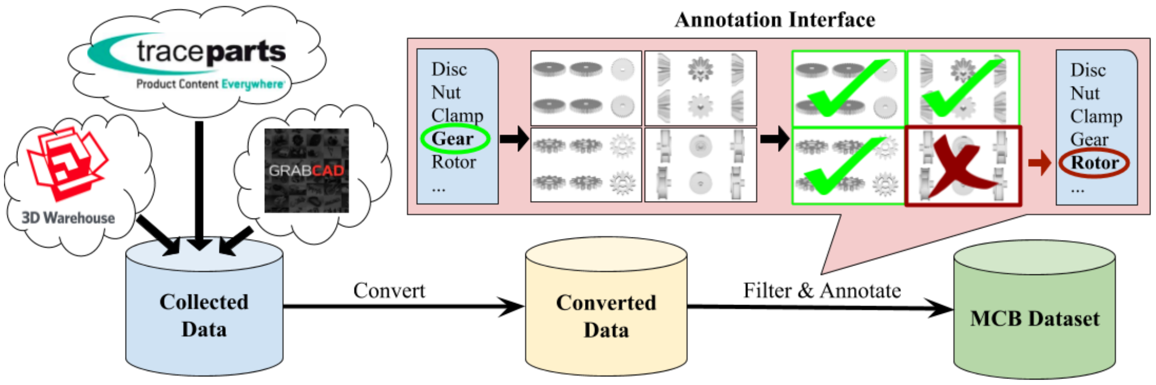


Fig. 5.3. Data acquisition and annotation overview for the creation of a large-scale mechanical components benchmark.

5.2.2 Acquired dataset purification

We developed a web-based platform to manage large-scale dataset functioning, collecting, viewing, filtering, and annotating data. The overview of the platform is available in Figure 5.3. Web-based applications have the advantage that users are free of installation and can easily access to the platform from any computer with internet connection. This accessibility accelerated the annotation process. We utilized the tool to trigger the scrapper collecting CAD models, also filtering and annotating the data with intuitive user interfaces, which is available in Figure 5.3. A dataset managing platform visualizes multi-view images of each engineering part, which gives users a more comprehensive understanding of the mechanical part during filtering and annotating. The data creation pipeline consists of consecutive three steps.

Step 1: *Conversion / Multi-view image generation.* The file format conversion process is necessary to create a unified file format dataset, since collected CAD models consist of various formats such as STL, STEP, and OFF. The converter module in the

Table 5.2.
The number of data before and after filtering.

Data source	#Data	
	Before	After
GrabCAD	22,703	5,301
3D Warehouse	20,478	12,737
TraceParts	120,665	40,658
Total	163,216	58,696

platform converts file format into OBJ format and captures projected images from multiple viewpoints. For 3D data conversion, we used Open Asset Import Library (Assimp) and Gmsh [170]. We used projected images for annotating engineering parts.

Step 2: *Filtering.* We filter scrapped CAD models by deleting broken and duplicated models and capturing wrongly categorized models for re-annotation. Meta-information (i.e. file size, file name, search keyword, and data source) tagged in the process of scrapping step helps users to filter the data. Eight ME experts manually filtered out the duplicates with our annotation tool. We group objects with similar meta-information and these experts manually removed duplicates. An annotation interface presents the models in a table format rather than one model at a time. Several models can be viewed at one time, which increases the speed of filtering and makes identifying duplicate models easier. A quantitative comparison of the dataset between before and after filtering is shown in Table 5.2.

Step 3: *Annotation.* After filtering, we re-annotate the missed categorized models to the correct category based on the tagged information and multi-view image. We use a 3D viewer in the annotation interface to present a close-up look when the multi-view image does not provide enough information for annotation. Some of the models do not belong to any of our mechanical components categories but are still relevant to engineering parts. We defined these models as miscellaneous and labeled them into new part categories as needed.

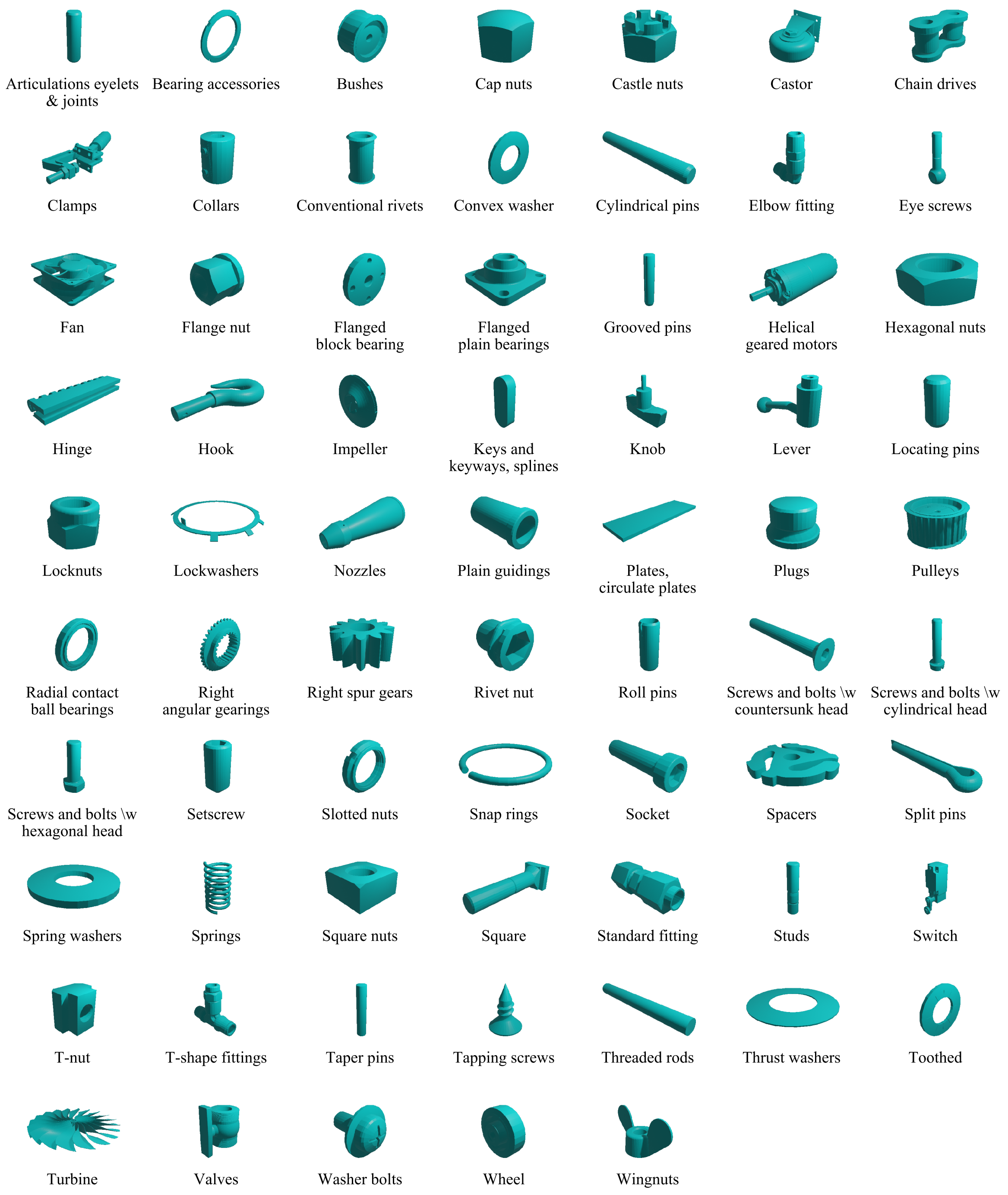


Fig. 5.4. Randomly sampled mechanical components from the MCB.

5.2.3 Statistics of the dataset

MCB has a total number of 58,696 mechanical components with 68 classes. The exact name of the types and amount of data in each category are shown in Table 5.3. Objects from TraceParts are aligned, but the objects from the other two sources (30 % of the objects) are not consistently oriented. We did not perform additional alignments as many object classes do not possess consistent orientations due to a variety of continuous/discrete symmetries. On the other hand, having unaligned models in shape classification and retrieval tasks helps to evaluate the generalization of the shape descriptors [171]. Unlike 3D Warehouse and GrabCAD that provide data from general usages, TraceParts stores data from the manufacturing companies. The CAD models from manufacturing companies show a tiny variation because they follow the parameterized catalogs for standardization. Therefore, to see the effect of data that has dense distribution and orientation invariance, we built two datasets for the experiment:

- Dataset A (MCB): Aggregated data from TraceParts[1], 3D Warehouse[2], and GrabCAD[3]
- Dataset B: Aggregated data from 3D Warehouse and GrabCAD.

The dataset A has the same statistics with the original MCB dataset, and the dataset B has 18,038 data with 25 classes. The detailed statistics of the dataset B is explained in supplementary material.

5.3 Experiments

To analyze the behavior of learning algorithms developed for common objects works on mechanical components, we benchmarked classification and retrieval tasks with three different representations: point could, projected views, and voxel grids. We use two NVIDIA GeForce RTX 2080Ti GPUs, i9-9900k CPU, and 64GB RAM for the experiments. We carefully choose seven state-of-the-art shape classifica-

Table 5.3.
The statistics of Mechanical Components Benchmark dataset.

Class	#Models	Class	#Models	Class	#Models
Articulations eyelets&joints	1,632	Impeller	145	Socket	858
Bearing accessories	107	Keys and keyways splines	4,936	Spacers	113
Bushes	764	Knob	644	Split pins	472
Cap nuts	225	Lever	1,032	Spring washers	55
Castle nuts	226	Locating pins	55	Springs	328
Castor	99	Locknuts	254	Square	72
Chain drives	100	Lockwashers	434	Square nuts	53
Clamps	155	Nozzle	154	Standard fitting	764
Collars	52	Plain guidings	49	Studs	4,089
Conventional rivets	3,806	Plates circulate plates	365	Switch	173
Convex washer	91	Plugs	169	T-nut	101
Cylindrical pins	1,895	Pulleys	121	T-shape fitting	338
Elbow fitting	383	Radial contact ball bearings	1,199	Taper pins	1,795
Eye screws	1,131	Right angular gearings	60	Tapping screws	2,182
Fan	213	Right spur gears	430	Threaded rods	1,022
Flange nut	53	Rivet nut	51	Thrust washers	2,333
Flanged block bearing	404	Roll pins	1,597	Toothed	47
Flanged plain bearings	110	Screws&bolts \w countersunk head	2,452	Turbine	85
Grooved pins	2,245	Screws&bolts \w cylindrical head	3,656	Valve	94
Helical geared motors	732	Screws&bolts \w hexagonal head	7,058	Washer bolt	912
Hexagonal nuts	1,039	Setscrew	1,334	Wheel	243
Hinge	54	Slotted nuts	78	Wingnuts	50
Hook	119	Snap rings	609	**Total**	**58,696**

tion algorithms from three different 3D shape representations: point cloud, multi-view, and voxel grids as the benchmark methods. In point cloud method, we use PointCNN [41], PointNet++ [133], and SpiderCNN [43]. For the multi-view based, we use MVCNN [46] and RotationNet [45]. DLAN [47] and VRN [50] are used to evaluate voxel grids representation. For training each method, we use the code and the hyper-parameters from seven deep-learning algorithm papers. We use 2,048 points density for point cloud, 32×32×32 grid for voxel grids, and 3×224×224 resolution for image-based representations. We follow the original papers for the input data processing and training procedures. For all the benchmark datasets, we randomly split the datasets into train and test set as 80% and 20%, respectively. Training is conducted for each method to prevent initialization variation and report the best results.

5.3.1 Object retrieval benchmark

At each entry, we calculate scores of the precision-recall curve in the retrieval results: precision, recall, F1-score, mAP, and Normalized Discounted Cumulative Gain (NDCG). In shape retrieval, NDCG has a heavier tail at high ranks, which means that it does not discount lower ranks as much as mAP does [54]. Therefore,

Table 5.4.
Summary table of evaluation metrics of shape retrieval benchmark for seven deep learning methods. They are grouped by their representation types. Each *, †, and $^{\boxtimes}$ symbol indicates the method point cloud, volumetric, and image, respectively.

Dataset	Method	micro					macro				
		P@N	R@N	F1@N	mAP	NDCG@N	P@N	R@N	F1@N	mAP	DCG@N
A	PointCNN* [41]	**0.892**	**0.892**	**0.690**	**0.889**	**0.898**	0.869	**0.797**	**0.833**	**0.886**	**0.854**
	PointNet++* [133]	0.778	0.778	0.613	0.794	0.754	0.772	0.678	0.712	0.803	0.746
	SpiderCNN* [43]	0.839	0.839	0.669	0.867	0.793	0.844	0.741	0.776	0.877	0.812
	MVCNN$^{\boxtimes}$ [46]	0.579	0.579	0.488	0.657	0.487	0.667	0.552	0.585	0.735	0.641
	RotationNet$^{\boxtimes}$ [45]	0.688	0.699	0.508	0.805	0.683	0.784	0.652	0.683	0.815	0.735
	DLAN† [47]	0.840	0.840	0.568	0.879	0.828	**0.878**	0.786	0.820	0.880	0.845
	VRN† [50]	0.537	0537	0.402	0.653	0.519	0.646	0.480	0.507	0.664	0.576
B	PointCNN*	0.905	0.905	**0.676**	**0.913**	0.899	0.895	0.829	0.853	**0.909**	**0.871**
	PointNet++*	0.847	0.847	0.657	0.892	0.798	0.873	0.799	0.823	0.903	0.846
	SpiderCNN*	0.779	0.779	0.609	0.829	0.728	0.782	0.698	0.719	0.841	0.757
	MVCNN$^{\boxtimes}$	0.786	0.786	0.609	0.831	0.742	0793	0.719	0.741	0.852	0.776
	RotationNet$^{\boxtimes}$	0.529	0.529	0.434	0.607	0.454	0.560	0.466	0.483	0.647	0.540
	DLAN†	**0.912**	**0.912**	0.674	0.908	**0.925**	**0.903**	**0.830**	**0.854**	0.902	0.870
	VRN†	0.607	0.607	0.460	0.628	0.613	0.565	0.468	0.484	0.619	0.534

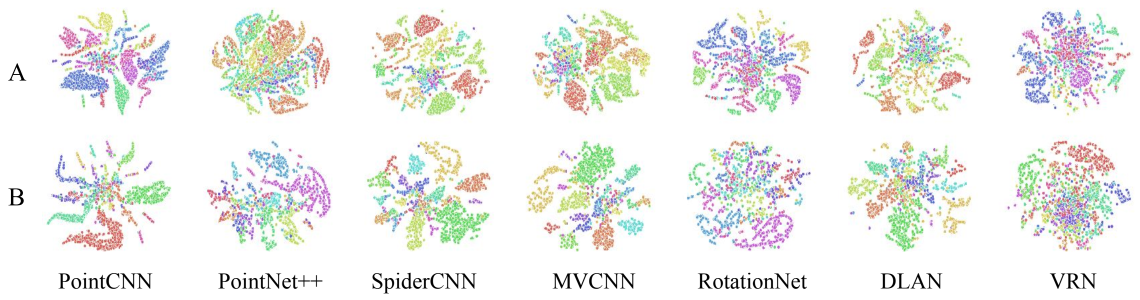


Fig. 5.5. t-SNE [172] plots of seven different deep neural networks trained with the dataset A and B. We set perplexity as 40 and iterate 300 times.

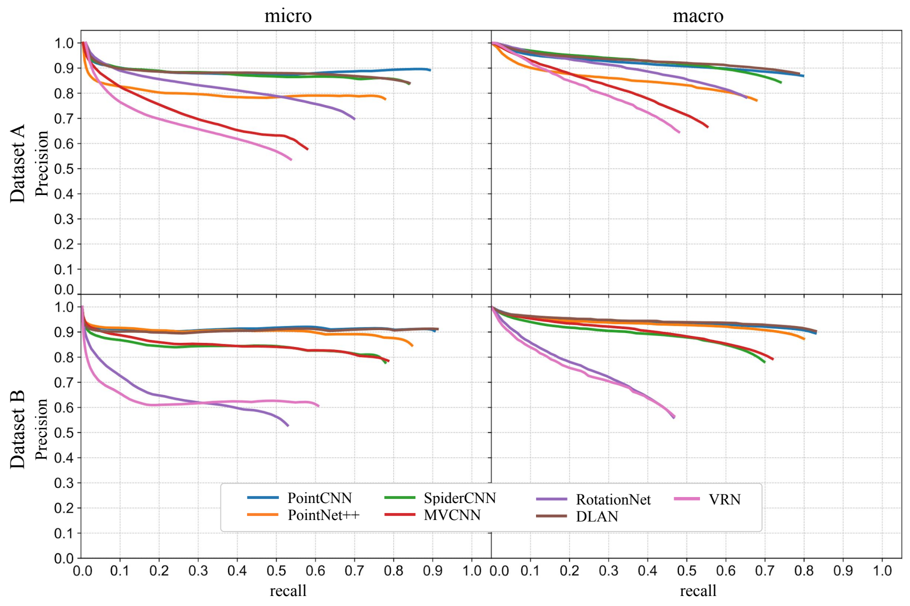


Fig. 5.6. Precision-recall curve plots for retrieval with seven different methods. The PointCNN shows best retrieval results for the dataset A, and DLAN shows best retrieval results for the dataset B.

NDCG has a better ability to show the ration between the real performance and ideal performance to evaluate the metrics.

Since each object has a different number of positive retrievals, the score table metrics are referred to as P@N, R@N, F1@N, and NDCG@N, where the N refers to the total retrieval list length of each object, which varies across queries. The macro-averaged version presents the performance of the dataset combining the result of each category. The micro-averaged version treats each query, and the retrieval result equally treats cross groups. Therefore, it eventually has the same P@N and R@N.

The summary results of all tested methods are given in Table 5.4. Corresponding precision-recall curves are given in Figure 5.6. To see the similarity of the geometric features from the descriptors, we perform t-distributed stochastic neighbor embedding (see Figure 5.5). We observe that the more the clusters are grouped, the more the retrieval results enhanced. Orientation invariance is crucial for the retrieval task. For example, DLAN and PointCNN, which have rotation invariance, perform best for the both datasets. However, VRN and RotationNet show poor results for the dataset B where the orientation is not aligned, even though it uses the same representation as DLAN. RotationNet also poorly performed on the common shape retrieval task [171] when the orientations of the objects are perturbed. The overall retrieval performance of the dataset A is relatively higher than the dataset B. Micro has slightly better results on P@N, while much better results on R@N show that the metrics have better performance in cross-category testing.

We observe that the performance of RotationNet and VRN dramatically decreases for the dataset B compared to the dataset A. This is because the object orientations are aligned in the dataset A but not in B. Similar behavior is observed for the common objects [54]. Specifically, RotationNet predicts view orders of given multi-views to learn rotation-invariant features. However, the camera viewpoints of solid of revolution shapes given multi-views are hard to determine and impossible to predict when the cameras are rotating along with the center axis of the object. DLAN and PointCNN perform well for the both datasets, with respect to both macro and micro metrics. We conclude that these methods extract rotation-invariant features across classes. As a point of view in data representation, point cloud methods show stable performance for the both datasets.

5.3.2 Object classification benchmark

For the classification task, we measure four metrics, mean accuracy over objects, average accuracy per class, F1-score and average precision (AP) and plotted precision-recall curves. We use the macro method for F1 and AP calculation. AP metrics are used to compare the network performance across the dataset A and B. F1-score is the harmonic mean of the precision and recall. The benchmark results for the datasets A and B are available in Table 5.5 and Figure 5.7. Additionally, to compare the performance between common objects and mechanical objects, we provide classification performance on MondelNet40 in Table 5.6.

Table 5.5.
Benchmark results of the seven classification models which were trained and evaluated on our mechanical engineering part benchmark. We trained five times per model and reported the highest result. Each *, †, and ⊠ symbol indicates the method: point cloud, volumetric, and image representation, respectively.

Method	Acc. over object (%)		Acc. over class (%)		F1-score		Average Precision	
	A	B	A	B	A	B	A	B
PointCNN* [41]	93.89	93.67	81.85	86.80	83.86	88.63	**90.13**	**93.86**
PointNet++* [133]	87.45	93.91	73.68	87.97	74.59	88.32	73.45	91.33
SpiderCNN* [43]	93.59	89.31	79.70	79.29	81.30	80.72	86.64	82.47
MVCNN⊠ [46]	64.67	79.17	80.47	84.09	69.69	77.69	79.82	86.66
RotationNet⊠ [45]	**97.35**	**94.73**	**90.79**	**89.70**	**92.29**	**91.05**	87.58	84.87
DLAN† [47]	93.53	91.38	82.97	84.21	83.81	83.88	89.80	90.14
VRN† [50]	93.17	85.44	80.34	70.15	81.48	73.01	85.72	77.36

Unlike the retrieval task, RotationNet outperforms the other methods for the both datasets (see Table 5.5 and 5.6). The performance of MVCNN drops significantly on the mechanical components compared to the common objects which is ModelNet40. On the other hand, the accuracy of RotationNet drops slightly. The major differences between MVCNN and RotationNet are estimating correspondence between each image and view order during training. This correspondence estimation relaxes rotation variant property by implicitly learning mapping function between each view of the object and camera view point. In point cloud methods, PointCNN shows the best performance on both datasets, and SpiderCNN perform better for the dataset A than B. PointCNN performs best for the AP (see Table 5.5). This is because mechanical components are sensitive to the local changes and PointCNN leverages spatially-local correlation. In the same sense, DLAN performs better on mechanical components due to oriented point sets. However, VRN performance drops on the mechanical components benchmark since voxel grids are orientation variant.

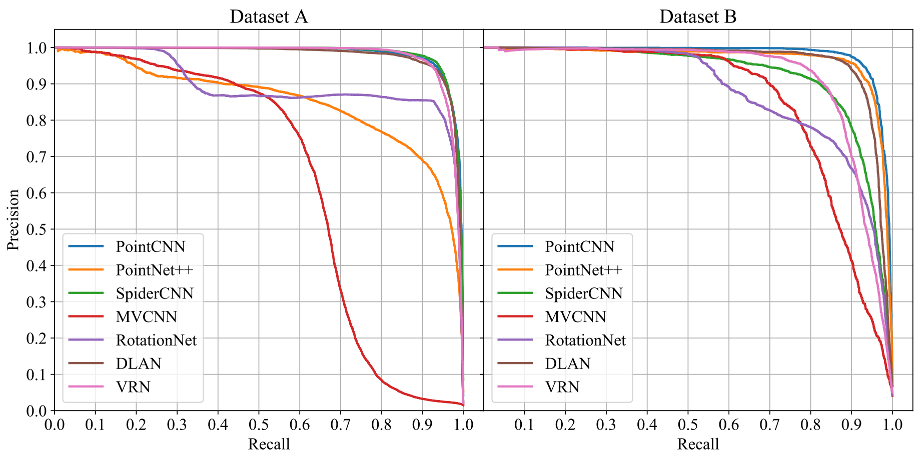


Fig. 5.7. Precision and Recall curve plots of classification task. Left plot shows the PR curve of the dataset A, and right plot shows the PR curve of the dataset B. The RotationNet shows the best performance in terms of accuracy.

From our benchmark result, capturing local features and having orientation invariance are crucial for developing mechanical components classifier. Although RotationNet shows 97 % accuracy over an object, the accuracy over the class, which is 90.79%, is not good enough to utilize in the industry. For the deep learning application for mechanical components in the industry, a deeper understanding of mechanical parts is required. The classification result of the 'Flanged plain bearings' class shows a

Table 5.6.
Classification accuracy on ModelNet40. Each *, $\dagger$, and $^\boxtimes$ symbol indicates the method is based on point cloud representation, volumetric representation, and image representation, respectively.

Method	Acc. over object (%)
PointCNN* [41]	92.2
PointNet++* [133]	91.9
SpiderCNN* [43]	92.4
MVCNN$^\boxtimes$ [46]	95.0
RotationNet$^\boxtimes$ [45]	**97.37**
DLAN$\dagger$ [47]	84.0
VRN$\dagger$ [50]	95.5

low accuracy, which is under 87% for every network. This value is relatively lower than the accuracy of other classes (see appendix). This result shows the limitation of existing 3D object classification algorithms in terms of extracting local features. The general shape of the bearing is almost similar to thick washers or rings. Therefore, if the network cannot capture the local difference of ring-shaped object, it is hard to distinguish these objects.

We experiment how point cloud density affects the results in point cloud base algorithms. We perform five different densities: 128, 256, 512, 1,024, and 2,048 points on three point cloud classification methods [41, 43, 133] for the classification task for the dataset B. From our experiment results, the performance increases as the density of the point cloud increases, as shown in line plots in Figure 5.8. However, the enhancement of results saturates as the point cloud density grows and the performance of SpiderCNN [43] decreases even the density increases from 1,025 to 2,048. PointNet++ [133] is the most sensitive in the density of the point cloud, and PointCNN [41] is the least vulnerable in the variation of the point cloud density.

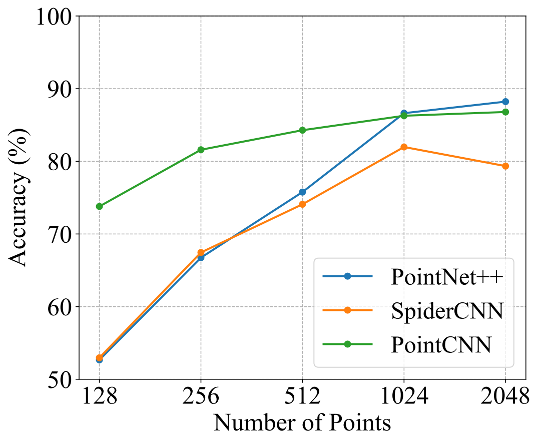


Fig. 5.8. Classification results of five different point cloud densities.

5.4 Conclusion and Discussion

We establish a large-scale mechanical component benchmark with annotations. For the creation of the dataset, we develop an annotation framework that enhances the efficacy of the annotation and filtering processes. We perform shape classification and retrieval experiments with seven deep-learning shape classification methods which are designed to classify common objects. We find that view-based and voxel grid presentation-based methods perform poorly on random orientation of mechanical components. However, DLAN, a voxel-based method, performs well on random orientation since it has orientation invariance. The creation of MCB and experimental results can be used for the development of data-driven algorithms of mechanical components.

6. FIRST-PERSON VIEW HAND SEGMENTATION OF MULTI-MODAL

Hands are crucial in many industrial computer vision applications, such as augmented reality, virtual reality, or human-computer interaction. Recognizing hands with vision systems is necessary to interact between people and digital devices. Therefore, understanding hands with computer vision systems has been deeply explored through hand tracking [173, 174], hand pose estimation [136, 175–178], grasp detection [179, 180], hand gesture recognition [181], multi-view prediction [136], and hand-action classification [89]. These works require segmenting hands from the background to increase the accuracy of performance.

However, segmenting hands when interacting with tools from the background is a challenging problem because (a) fingertips are heavily occluded by the hand dorsum and tools in the first-person view, (b) tools are held with various grasps, and (c) shapes of tools or objects are infinite. The traditional approach to create an RGB hand segmentation video dataset is through manual pixel-wise labeling [89, 93, 182, 183]. However, time and cost of person-in-the-loop segmentation grows linearly as

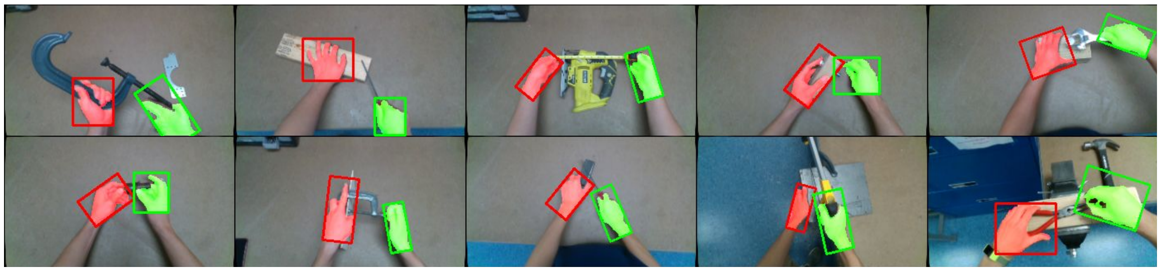


Fig. 6.1. Sample frames from our hand segmentation video dataset. Red and green masks represent left and right hand, respectively.

the number of frames increases, which reduces the scalability. Therefore, developing an efficient segmentation method is important for creating a segmented hand video dataset.

We utilize hand temperature as prior information for the labeling process. However, pixels from LWIR thermal images of hands may falsely include pixels from the surroundings that share the same temperature. In our method, we utilize crowd workers to provide an Axis-Aligned Bounding Box (AABB) to localize the detailed boundary of each hand. We further relax the AABB creation task by training a tracker with a small amount of AABB results. The tracker learns the shape features of hands and then uses them to estimate an Oriented Minimum Bounding Box (OMBB) for each hand. Therefore, we use both spatial and thermal features of hands to segment them from the background and tools. Our approach is effective regardless of finger tips or finger joints occlusion and do not require hand pose ground truth. We prove our method is much more efficient than the traditional pixel-wise labeling tasks while maintaining a high performance.

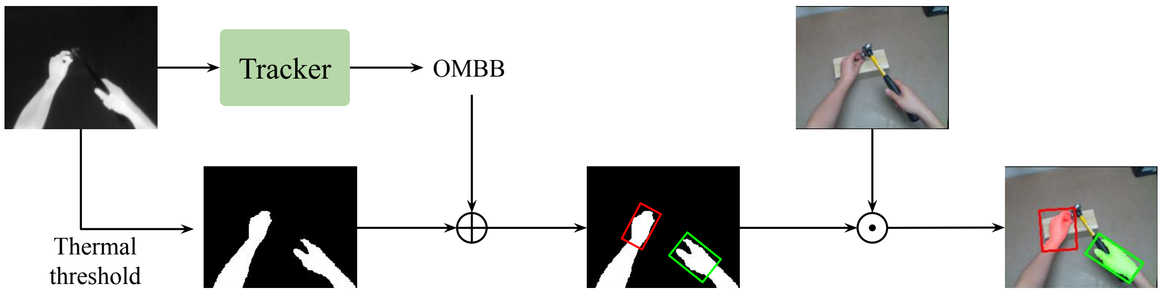


Fig. 6.2. Overview of our proposed segmentation method.

Optimizing deep neural networks with a single modality may lead to failures when the network fails to extract distinctive features from the input source. Multiple modalities are used to provide distinctive features to the networks [123, 184, 185] and has been an emerging area [186–188] due to the enhancement of computation power and sensors. Human body temperature is relatively constant [189] and has been widely used in pedestrian detection [190], biological image processing [191], and gesture recognition [192]. An additional advantage of Long-Wave InfraRed (LWIR) is that it is invariant to colors, textures, and lighting conditions. Color information may mislead vision systems distinguishing shape features. Therefore, we create a multi-modal (LWIR, RGB, and depth) hand segmentation video dataset which consists of 790 sequences and 401,765 frames of "hands using tools" videos. Compared to the other existing hand segmentation datasets, our dataset contains three different modalities and head-mounted camera Inertial Measurement Unit (IMU) information.

With the video dataset, we analyze fusing three modalities with DeepLabV3+ [76] and benchmark five different state-of-the-art segmentation methods [73, 76, 89, 103, 193]. We observe that the neural networks can automatically learn important cues from three different modalities: LWIR, RGB, and depth. The jointly learned features of these three modalities prevents confusion between hands and backgrounds.

We record our videos with a low-cost non-radiometric thermal camera, Flir Boson 320, to capture relative LWIR data. RGB resolution then is rescaled to match resolution of the LWIR sensor with two camera frustums. We use an Intel D435i depth camera for RGB, depth and IMU information acquisition. These sensors are placed within a 3D printed case and mounted in front of a helmet to make the camera location consistent over sequences.

6.1 Data Fusion for Pixel-Wise Hand Segmentation

To segment hands with LWIR frames, we narrow down the search space by finding a Thermal Mask (TM), denoted I_{tm}, with LWIR frames, denoted I_{lwir}^{raw}, and $I_{lwir} = \mathcal{T}(I_{lwir}^{raw})$ where $\mathcal{T}$ is transformation function that transforms an LWIR plane to an RGB plane.

$$I_{tm} = \omega(I_{lwir}) \tag{6.1}$$

The bounded value for a pixel in the target frame corresponding to a spatial location (i, j) in I_{tm} is defined as follows:

$$\omega^{(i,j)}(I_{lwir}^{(i,j)}) = \begin{cases} 1 \ \text{if } a \leq I_{lwir}^{(i,j)} \leq b & (6.2) \\ 0 \quad \text{otherwise} & (6.3) \end{cases}$$

,where a and b are upper bound and lower bound of hand temperature. $I_{tm} \in [0, 1]^{H \times W}$ and $I_{lwir} \in \mathbb{R}^{H \times W}$. We map the I_{lwir}^{raw} onto the RGB frame, denoted I_{rgb}, with depth maps. To align depth maps and I_{lwir}^{raw}, we find the spatial relationship between the I_{lwir}^{raw} and depth camera. A projection of an object in pixel space is derived by multiplying the camera matrix ($\boldsymbol{K_T}$ for LWIR camera and $\boldsymbol{K_D}$ for depth camera) with an object point.

$$\boldsymbol{p_D} = \boldsymbol{K_D} \cdot \boldsymbol{P_D} \tag{6.4}$$

$$\lambda \cdot \boldsymbol{p_T} = \boldsymbol{K_T} \cdot \boldsymbol{P_T} \tag{6.5}$$

,where $\boldsymbol{p_D} = [u_D, v_D, w_D]^T$, $\boldsymbol{p_T} = [u_T, v_T, 1]^T$, a projected point in depth and LWIR camera pixel plane, respectively. $\boldsymbol{P_D}$ and $\boldsymbol{P_T}$ are an object point in depth and LWIR camera coordinate, respectively. λ is a scale factor. The spatial relation between two cameras is defined by equation 6.6 where $\boldsymbol{R}$ is a 3D rotation matrix and $\boldsymbol{T}$ is a translation matrix.

$$\boldsymbol{P_T} = \boldsymbol{R} \cdot \boldsymbol{P_D} + \boldsymbol{T} \tag{6.6}$$

By combining equation 6.4, 6.5, and 6.6, we can get an equation:

$$\lambda \cdot \boldsymbol{p_T} = \boldsymbol{K_T} \cdot (\boldsymbol{R} \cdot \boldsymbol{K_D^{-1}} \cdot \boldsymbol{p_D} + \boldsymbol{T}) \tag{6.7}$$

By solving equation 6.7, we transform I_{lwir}^{raw} to the depth plane and depth plane to the RGB plane. The detail of solving equation 6.7 is explained in the supplementary document. The RGB and depth frames are aligned using the Intel RealSense API, and the different resolution and field of view between the two cameras is adjusted using intrinsic camera parameters. After the alignment, we threshold the hand temperature by setting the lower and upper bounds in human temperature as depicted in Figure 6.3 b. These bounds are manually captured for every sequence and used as priors that segment hands from surrounding backgrounds and the hand-held objects. To create accurate bounds, we overlapped the I_{lwir}^{raw} on the depth maps as seen in Figure 6.3 column a. Finally, we get the segmented hands by filtering thermal mask (see Figure 6.3 column d).

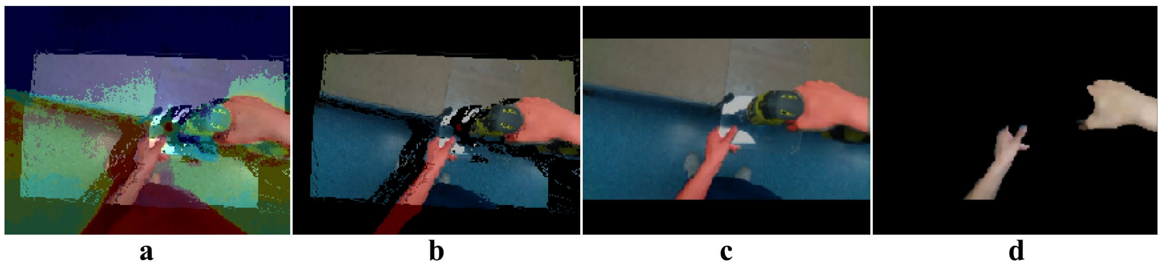


Fig. 6.3. Aligning I_{lwir}^{raw} into the I_{rgb} with depth maps. First, (a) I_{lwir}^{raw} and I_{rgb} are overlapped onto the depth maps. (b) Next, the projected I_{lwir}^{raw} is bounded by hand temperature to capture possible hand regions. (c) Then, the projected I_{rgb} and I_{lwir}^{raw} are transformed on the RGB plane. (d) Finally, hands are cropped from backgrounds. For visualization, I_{lwir} is color mapped by a high value as red and a low value as blue.

6.2 Removing Artifacts with Hand Geometry

We occasionally observe mislabeled pixels from backgrounds that have similar temperature as hands. To remove these mislabeled pixels, we use a tracking algorithm, named SiamMask [194], to localize hands with OMBBs shown in Figure 6.2. We train the tracker with I_{tm} and AABBs of the hand. We use Amazon Mechanical Turk (AMT) to crowdsource the creation of hand AABBs. For training the tracker, I_{tm} and corresponding AABBs are used as targets and I_{lwir} is used as inputs. Therefore, the tracker is color and texture in-variant, which improves tracking performance of the tracker (sec. 89). After the training process, given initial AABBs, the tracker predicts OMBBs and classifies them sequentially as a left hand or a right hand through frames. This implies that the tracker learns hand shape features. These OMBBs are used to remove mislabeled pixels by intersecting I_{tm} and OMBBs.

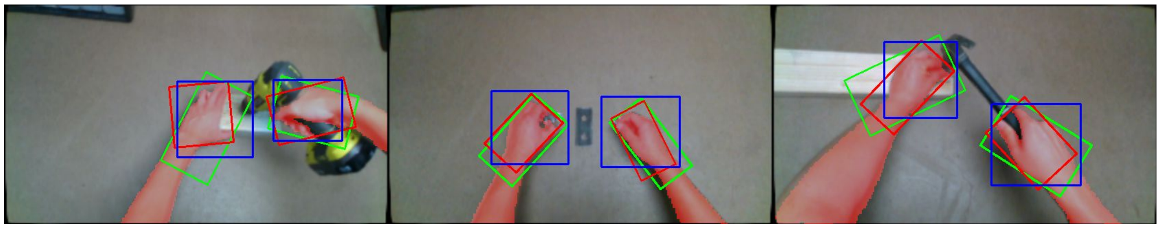


Fig. 6.4. Visualization of three different bounding boxes. I_{tm} is overlapped onto the I_{rgb} as a red mask for visualization. The bounding boxes with red, blue, and green represent AABBs, OMBBs (I_{rgb}), and OMBBs (I_{lwir}), respectively.

6.3 Dataset Overview and Efficiency Evaluation

In this section, we show the efficiency of our proposed segmentation method, the performance of the tracker which learns hand geometric. We consider the hand segmentation problem as a two-class segmentation task, and we plot the maximum probability between the two classes, background and hands, per pixel for the prediction mask creation. For evaluation metrics, we use the Intersection over Union (IoU) of the hand (hIoU) and the background (bIoU). We define the mean IoU (mIoU) as the mean of these two class IoUs.

6.3.1 Dataset statistic and overview

To segment hands from objects, we create a pixel-wise hand segmentation dataset with subjects holding objects and tools. The dataset consists of 401,765 frames and 790 sequences. Our dataset has a large number of sequences and frames compared with the other datasets. We manually annotated 13,792 frames from 136 sequences to create a test dataset. The video dataset contains five subjects, 15 actions, and 23 tools. The distribution of the dataset across the actions and tools is plotted in the supplementary document. For annotating per pixel label of hands, we use I_{lwir} as prior knowledge and used a tracker to identify orientation of hands as well as whether a hand is a left hand or a right hand.

6.3.2 Evaluation of data fusion efficiency

We evaluate the accuracy of I_{tm} with manually labeled frames. I_{tm} is defined by the temperature of the hands. It shows fairly reasonable accuracy of 0.849 in hIoU. We find that I_{tm} reduces false-positive in the sequences where non-hand area has same temperature as hands. These mislabeled pixels are the main reason of the hIoU degradation. We remove these mislabeled pixels with the tracker given initial AABBs [194]. Utilizing I_{lwir} improves the efficiency of the manually labeling frames. We profile the amount of time that it takes to annotate frames using four different methods for pixel-wise segmentation labeling. First, annotators use PolyRNN++ [79] with I_{rgb}. Second, annotators label on a tablet with a tablet pen given I_{rgb}. Third, annotators label on a tablet with a tablet pen given masked $I'_{rgb} = I_{rgb} \odot I_{tm}$, where $\odot$ is the Hadamard product. Lastly, annotators draw AABBs on top of I_{rgb}. Ten people annotated a random sample of twenty frames with the four different methods. We then averaged the annotation time, yielding the results in Table 6.1. We find that drawing AABBs is 24 times faster than the other methods. This implies that our method is 24 times faster than PolyRNN++ since our method intersects AABBs on I_{tm}. Additionally, the third method is two times faster than the second approach and six times faster than using PolyRNN++. Masking hands with I_{tm} significantly narrows down the region for annotators, which reduces the labeling time. To validate the quality of the annotation methods, we randomly sample 136 sequences and manually annotate 13,792 frames. With these manually annotated frames, we evaluate the IoUs of I_{tm} with and without AABBs labels (see Table 6.2).

Table 6.1.
Comparison table of annotation average time cost per frame. $I'_{rgb} = I_{rgb} \odot I_{tm}$. Lower average time is better.

Annotation Methods	Avg. Time (second)
Drawing polygon with PolyRNN++ [79]	122
Painting hands on I_{rgb} with a tablet pen	76
Painting hands on I'_{rgb} with a tablet pen	24
Drawing AABBs on I_{rgb}	**5**

6.3.3 Evaluation of learning hand geometry

We use SiamMask as our tracker [194] and train the tracker with a seeding dataset consisting of 518 sequences that have 11,718 frames labeled by crowd workers. The dataset is divided into 441 sequences that have 7,882 frames for training and 77 sequences that have 3,836 frames for validation. The tracker is trained in two ways: with I_{rgb} frames and with I_{lwir} frames. For evaluation, we use the manually labeled frames and metrics from Section 6.3.2. From the evaluation, we find that the tracker with I_{lwir} frames outperforms the others as shown in Table 6.2. The tracker with I_{rgb} tends to detect more forearm than the tracker with I_{lwir} as shown in Figure 6.4. This implies that the tracker with I_{lwir} is more sensitive in finding convex shape of wrist than the tracker with I_{rgb}, yielding better orientation of the hand and tighter OMBBs. The tracker performs well in most of cases; however, we need to re-initialize it with AABBs when the tracker fails to estimate the next frame. We also find that the tracker fails to track the hands when the two hands are heavily overlapping. In this case, we need to manually draw the OMBBs.

Table 6.2.
Comparison of the quality of the annotated AABBs and the tracker-generated OMBBs.

Annotation Source	mIoU	hIoU	bIoU
I_{tm}	0.913	0.849	0.977
I_{tm} with AABBs	0.917	0.842	**0.992**
I_{tm} with OMBBs (I_{rgb})	0.921	0.851	0.990
I_{tm} with OMBBs (I_{lwir})	**0.923**	**0.855**	0.990

6.4 Experiments

In this section, we present analysis of multi-modal sources for hand segmentation, and we benchmark our dataset with five different segmentation methods. For the test dataset, we use manually annotated labels which consist of sequences that are used in neither the training set nor the validation set. The models have been trained using Stochastic Gradient Descent (SGD) [195] and the ADAM optimizer [117] with initial parameters: learning rate as 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We decay the learning rate by 0.1 every 250K steps. Additional configurations of the experiments are listed in the supplementary document. We use a single TITAN RTX GPU and an Intel i7-6850K CPU for the experiments.

6.4.1 Data fusion analysis

We analyze the effect of multi-modal sequences, $\{I_{rgb}, I_{lwir}, I_{depth}\}$, for hand segmentation by conducting seven ablation studies using all possible combinations of $\{I_{rgb}, I_{lwir}, I_{depth}\}$ as input modalities. We perform the seven ablation studies to find out how I_{lwir} contributes in training neural networks. For all experiments in this section, we use randomly sampled 50K frames and split into two sets as following: 40K frames as train dataset and 10K frames as test dataset. The frames in the test dataset are labeled manually. We use DeepLabV3+ [76] as base model and add additional encoders to fuse additional modalities. Our fusing method is detailed on the supplementary document. The rationale of using DeepLabV3+ is that it outperforms other methods [73,89,193] in hand segmentation benchmark experiments, only using RGB, as shown in Table 6.4. It also has the fewest parameters. We use ResNet 101 [196] which is pre-trained on the ImageNet [158] as a backbone network. All experiments use an equal number of encoders as the number of input modalities. From experiments, we found that I_{lwir} guides the network in finding better minima by observing both the loss drops and performance improvement as shown in Figure 6.5 when the I_{lwir} is used also shown in Table 6.3. Including I_{lwir} enhances the performance of hand segmentation by 5% in hIOU score compared to $\{I_{rgb}, I_{depth}\}$. Increments are observed for bIOU and mIOU scores as well. Therefore, I_{lwir} is a robust feature for hand segmentation. The three modalities contain complementary properties, which generates robust features, compensates weak points of each other, and leverages their advantages.

Table 6.3.
The higher values are better in IoU, and the lower values are better in #Parameters. C, D, and L stands for RGB, depth, and LWIR frames, respectively. DeepLabV3+ [76] is used for the experiments.

Source	mIoU	hIoU	bIoU	#Parameters
D	0.857	0.753	0.960	**59.3 M**
L	0.907	0.840	0.974	**59.3 M**
C	0.884	0.800	0.968	**59.3 M**
D,L	0.897	0.822	0.972	118.0 M
D,C	0.906	0.838	0.975	118.0 M
L,C	0.921	0.862	0.979	118.0 M
L,C,D	**0.931**	**0.880**	**0.982**	176.6 M

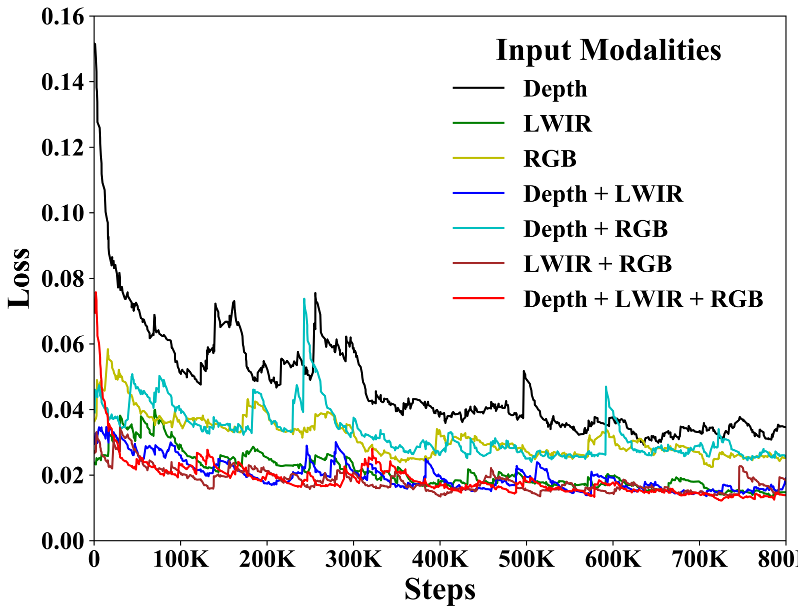


Fig. 6.5. Comparison of segmentation IoUs of seven experiments using different input modalities. DeepLabV3+ [76] is used for the experiments.

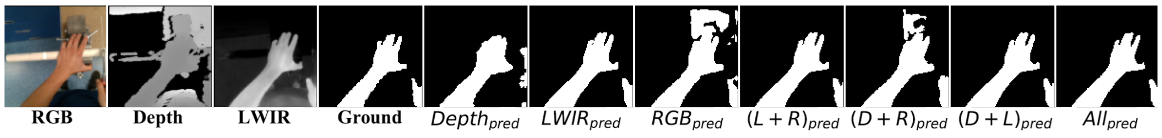


Fig. 6.6. Qualitative results of the seven different ablation experiments. L, R, D, and All stand for I_{lwir}, I_{rgb}, I_{depth}, and $\{I_{rgb}, I_{lwir}, I_{depth}\}$, respectively.

6.4.2 Hand segmentation benchmark

To validate the performance of using multiple modalities, we compare our method with five state-of-the-art segmentation methods [73, 76, 76, 89, 193] which use I_{rgb} and segmentation networks [103] and jointly use I_{rgb} and I_{lwir} as input modalities. We use the same dataset as Section 6.4.1 and hyper-parameters listed on the original papers. We notice that RTFNet [103] performs second-best among all methods, indicating

I_{lwir} provides the most meaningful prior knowledge for segmenting hands in frames. DeepLabV3+* with three modalities outperform the second-best method, RTFnet, by 4% in hIoU and 30% fewer parameters, as shown in Table 6.4.

Table 6.4.
Comparison of quantitative results with other segmentation methods. DeepLabV3+* is trained with fused $\{I_{rgb}, I_{lwir}, I_{depth}\}$. The higher values are better in IoU and the lower values are better in size of model parameters.

	mIoU	hIoU	bIoU	Model Size
HIW [89]	0.865	0.770	0.865	118.0 M
PSPet [73]	0.897	0.823	0.972	70.4 M
DUC-HDC [193]	0.893	0.815	0.961	69.2 M
RTFNet [103]	0.911	0.846	0.976	254.5 M
DeepLabV3+ [76]	0.907	0.840	0.974	**59.3 M**
DeepLabV3+* [76]	**0.931**	**0.880**	**0.982**	176.6 M

6.5 Conclusion and Discussion

In this work, we propose a robust and efficient pixel-wise hand segmentation method and a multi-modal dataset. Our method and dataset can ease dataset creation for more research on vision problems. We record rich sequences with three different image modalities and IMU information of first-person-view images with pixel-wise hands and action labels. We found that using multiple modalities achieves 4% better hIoU when compared to the existing state-of-the-art methods for hand segmentation. We also show that our multi-modal dataset with fusing LWIR and RGB-D frames achieves 5% better hand IoU performance than using just RGB-D frames. Also, we notice that only using I_{lwir} gives poorer results than using other modalities such as RGB and depth. This could be because thermal signature of hand is shared by other body parts. The proposed method is 24 times faster than PolyRNN++ with similar quality of manually labeled frames. One limitation we find is that the tracker does not work properly when two hands are heavily overlapped. The future development will be focusing on improving the dataset for more diverse hand-related tasks such as hand-object pose estimation, object reconstruction when a person is holding the object, and hand action recognition.

7. CONCLUSION

7.1 Summary

We address developing computer vision-based machine perception of objects and hands with deep learning. Recent advances in deep learning in computer vision have made significant improvements in almost every data-powered machine perception applications in literature and industry. However, there are still many barriers to the vast and successful adoption of advanced computer vision systems for real applications. One of the barriers is improving the generalizability of the machine perception model to apply in diverse environments. We first tackle the problem by improving learning strategies. We propose a fully-convolutional conditional generative model and jointly learning the surface and volume of 3D objects. In addition, we further improve performance by developing a data curation methods. Deep neural networks needs to be optimized with a vast amount of data from diverse environments with advanced training methods. However, collecting and labeling large amounts of data is time-consuming and expensive. To relieve these issues, developing self-supervised or unsupervised learning with multimodal information as a practical approach for achieving human-level vision perception in computer vision. In addition, in order to minimize the human effort in dataset creation we leverage multimodal visual sensors and learned knowledge from diverse domains and tasks. In particular, we explore developing conditional generative models and automatizing a large-scale dataset creation by (1) traversing latent space to make conditional generative models fully-convolutional, (2) jointly learning surface and volumetric representations of 3D objects for conditional generative models, (3) establishing mechanical components benchmark for intelligent manufacturing, and (4) automating a pixel-wise hand segmentation labeling process with multimodal sensors. To this end, we focus on answering machine perception of hands and objects in computer vision and computer graphics.

7.2 Future Directions

Current AI works are mostly limited in weak AI, also called narrow AI; the model solves single tasks in a restricted domain. However, the demand for AI in real life is not restricted to a single task but expands to multiple tasks within multiple domains. For instance, AI applications in autonomous driving, smart factories, and smart homes require a visual perception module, a speech recognition part, language translation models, searching algorithms from the cloud, etc. Therefore, consolidating cross-domain information into single artificial consciousness is preferable, which can hallucinate multi-domain representations. For example, people can imagine a sound from a moving car by observing a car through windows when a person is in a soundproof room. In this regard, learning underline data structure with the generative model is important, but it also has to be designed such that the latent space is sharable between task-specific networks in multiple domains. The AI system can further expand with self-supervised learning methods to be applicable in diverse environments without human assistants. Additionally, deep neural networks have been evolved in computer vision perception in the 2D domain, but now deep neural networks is actively moving to 3D spaces with inverse simulation and computer graphics. This is not only because of the popularity of AR/VR, but also because of the nature of our world in which people live in 3D space. 2D information is limited and not sufficient for learning the full geometry of objects and real environments. Moreover, biomechanics domain knowledge can be used to develop deep learning algorithms for human understanding since biomechanics understands the biological system with deterministic functions which have a long history.

REFERENCES

REFERENCES

[1] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017. [Online]. Available: http://arxiv.org/abs/1701.00160

[2] J. Dinerstein, P. K. Egbert, and D. Cline, "Enhancing computer graphics through machine learning: a survey," *The Visual Computer*, vol. 23, no. 1, pp. 25–43, 2007.

[3] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[4] Q. Wang, T. Artières, M. Chen, and L. Denoyer, "Adversarial learning for modeling human motion," *The Visual Computer*, pp. 1–20, 2018.

[5] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 332–347.

[6] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 4460–4464.

[7] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[8] M. Chen, Q. Zou, C. Wang, and L. Liu, "Edgenet: Deep metric learning for 3d shapes," *Computer Aided Geometric Design*, vol. 72, pp. 19–33, 2019.

[9] L. Gao, J. Yang, T. Wu, Y.-J. Yuan, H. Fu, Y.-K. Lai, and H. Zhang, "SDM-NET: Deep generative network for structured deformable mesh," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, vol. 38, no. 6, p. To appear, 2019.

[10] H. Pan, Y. Liu, A. Sheffer, N. Vining, C.-J. Li, and W. Wang, "Flow aligned surfacing of curve networks," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 127, 2015.

[11] A. Kotte, N. Van Wieringen, and J. Lagendijk, "Modelling tissue heating with ferromagnetic seeds," *Physics in Medicine & Biology*, vol. 43, no. 1, p. 105, 1998.

[12] D. Nelson, S. Charbonnel, A. Curran, E. Marttila, D. Fiala, P. Mason, and J. Ziriax, "A high-resolution voxel model for predicting local tissue temperatures in humans subjected to warm and hot environments," *Journal of Biomechanical Engineering*, vol. 131, no. 4, p. 041003, 2009.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[15] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[16] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International Conference on Machine Learning*, 2018, pp. 40–49.

[17] N. Umetani, "Exploring generative 3d shapes using autoencoder networks," in *SIGGRAPH Asia 2017 Technical Briefs.* ACM, 2017, p. 24.

[18] J. Liu, F. Yu, and T. Funkhouser, "Interactive 3d modeling with a generative adversarial network," in *2017 International Conference on 3D Vision (3DV).* IEEE, 2017, pp. 126–134.

[19] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu, "Learning descriptor networks for 3d shape synthesis and analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8629–8638.

[20] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "Atlasnet: A papier-m\^ ach\'e approach to learning 3d surface generation," *arXiv preprint arXiv:1802.05384*, 2018.

[21] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun, "A probabilistic model for component-based shape synthesis," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 55, 2012.

[22] W. E. Carlson, "An algorithm and data structure for 3d object synthesis using surface patch intersections," *ACM SIGGRAPH Computer Graphics*, vol. 16, no. 3, pp. 255–263, 1982.

[23] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.

[24] E. J. Smith and D. Meger, "Improved adversarial systems for 3d object generation and reconstruction," in *Conference on Robot Learning*, 2017, pp. 87–96.

[25] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[26] K. Chen, C. B. Choy, M. Savva, A. X. Chang, T. Funkhouser, and S. Savarese, "Text2shape: Generating shapes from natural language by learning joint embeddings," *arXiv preprint arXiv:1803.08495*, 2018.

[27] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[28] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "Cvae-gan: Fine-grained image generation through asymmetric training," *arXiv preprint arXiv:1703.10155*, 2017.

[29] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg, "Transformation-grounded image generation network for novel 3d view synthesis," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 702–711.

[30] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Multi-view 3d models from single images with a convolutional network," in *European Conference on Computer Vision*. Springer, 2016, pp. 322–337.

[31] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *European Conference on Computer Vision*. Springer, 2016, pp. 286–301.

[32] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess, "Unsupervised learning of 3d structure from images," in *Advances in Neural Information Processing Systems*, 2016, pp. 4996–5004.

[33] Z. Zhang, Y. Song, and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5810–5818.

[34] S. Reed, K. Sohn, Y. Zhang, and H. Lee, "Learning to disentangle factors of variation with manifold interaction," in *International Conference on Machine Learning*, 2014, pp. 1431–1439.

[35] J. Gauthier, "Conditional generative adversarial nets for convolutional face generation," *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, vol. 2014, no. 5, p. 2, 2014.

[36] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," *arXiv preprint arXiv:1702.01983*, 2017.

[37] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1538–1546.

[38] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, 2015, pp. 3483–3491.

[39] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*. Springer, 2016, pp. 776–791.

[40] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in Neural Information Processing Systems*, 2015, pp. 2539–2547.

[41] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems*, 2018, pp. 820–830.

[42] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[43] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.

[44] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[45] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5010–5019.

[46] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3d shape classifiers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[47] T. Furuya and R. Ohbuchi, "Diffusion-on-manifold aggregation of local features for shape-based 3d model retrieval," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval.* ACM, 2015, pp. 171–178.

[48] S. Liu, L. Giles, and A. Ororbia, "Learning a hierarchical latent-variable model of 3d shapes," in *2018 International Conference on 3D Vision (3DV).* IEEE, 2018, pp. 542–551.

[49] 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 (IROS).* IEEE, 2015, pp. 922–928.

[50] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," *arXiv preprint arXiv:1608.04236*, 2016.

[51] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Proceedings Shape Modeling Applications, 2004.* IEEE, 2004, pp. 167–178.

[52] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[53] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 909–918.

[54] M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai *et al.*, "Large-scale 3d shape retrieval from shapenet core55: Shrec'17 track," in *Proceedings of the Workshop on 3D Object Retrieval.* Eurographics Association, 2017, pp. 39–50.

[55] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *European conference on computer vision.* Springer, 2016, pp. 628–644.

[56] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[57] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani, "Three-dimensional shape searching: state-of-the-art review and future trends," *Computer-Aided Design*, vol. 37, no. 5, pp. 509–530, 2005.

[58] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani, "Developing an engineering shape benchmark for cad models," *Computer-Aided Design*, vol. 38, no. 9, pp. 939–953, 2006.

[59] D. Bespalov, C. Y. Ip, W. C. Regli, and J. Shaffer, "Benchmarking cad search techniques," in *Proceedings of the 2005 ACM symposium on Solid and physical modeling.* ACM, 2005, pp. 275–286.

[60] W. C. Regli, C. Foster, E. Hayes, C. Y. Ip, D. McWherter, M. Peabody, Y. Shapirsteyn, and V. Zaychik, "National design repository project: a status report," in *International Joint Conferences on Artificial Intelligence (IJCAI), Seattle, WA, Aug*, 2001, pp. 4–10.

[61] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9601–9611.

[62] N. J. Mitra, M. Wand, H. Zhang, D. Cohen-Or, V. Kim, and Q.-X. Huang, "Structure-aware shape processing," in *ACM SIGGRAPH 2014 Courses*, 2014, pp. 1–21.

[63] J. Li, C. Niu, and K. Xu, "Learning part generation and assembly for structure-aware shape synthesis," *arXiv preprint arXiv:1906.06693*, 2019.

[64] H. Wang, N. Schor, R. Hu, H. Huang, D. Cohen-Or, and H. Huang, "Global-to-local generative model for 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–10, 2018.

[65] Z. Wu, X. Wang, D. Lin, D. Lischinski, D. Cohen-Or, and H. Huang, "Sagnet: Structure-aware generative network for 3d-shape modeling," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[66] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas, "Grass: Generative recursive autoencoders for shape structures," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.

[67] A. Dubrovina, F. Xia, P. Achlioptas, M. Shalah, R. Groscot, and L. J. Guibas, "Composite shape modeling via latent space factorization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8140–8149.

[68] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. J. Mitra, and L. J. Guibas, "Structurenet: hierarchical graph networks for 3d shape generation," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–19, 2019.

[69] N. Schor, O. Katzir, H. Zhang, and D. Cohen-Or, "Componet: Learning to generate the unseen by part synthesis and composition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8759–8768.

[70] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5868–5877.

[71] M. Sung, V. G. Kim, R. Angst, and L. Guibas, "Data-driven structural priors for shape completion," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 175, 2015.

[72] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[73] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[74] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[75] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[76] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[77] D. Liu, Y. Li, J. Lin, and H. Li, "Deep learning-based video coding: A review and a case study," 04 2019.

[78] H. Lin, P. Upchurch, and K. Bala, "Block annotation: Better image annotation for semantic segmentation with sub-image decomposition," 02 2020.

[79] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-rnn++," 06 2018, pp. 859–868.

[80] H. Le, L. Mai, B. Price, S. Cohen, H. Jin, and F. Liu, "Interactive boundary prediction for object selection," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[81] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Interactive image segmentation by maximal similarity based region merging," *Pattern Recognition*, vol. 43, no. 2, pp. 445–456, 2010.

[82] M. Andriluka, J. Uijlings, and V. Ferrari, "Fluid annotation: A human-machine collaboration interface for full image annotation," 10 2018, pp. 1957–1966.

[83] Y. Aksoy, T. O. Aydın, A. Smolić, and M. Pollefeys, "Unmixing-based soft color segmentation for image manipulation," *ACM Trans. Graph.*, vol. 36, no. 2, pp. 19:1–19:19, 2017.

[84] L. Zhang, C. Fu, and J. Li, "Collaborative annotation of semantic objects in images with multi-granularity supervisions," 10 2018, pp. 474–482.

[85] X. Qin, S. He, Z. Zhang, M. Dehghan, and M. Jagersand, "Bylabel: A boundary based semi-automatic image annotation tool," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1804–1813.

[86] T. A. Biresaw, T. Nawaz, J. Ferryman, and A. I. Dell, "Vitbat: Video tracking and behavior annotation tool," in *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2016, pp. 295–301.

[87] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, 05 2008.

[88] R. Shilkrot, S. Narasimhaswamy, S. Vazir, and M. Hoai, "Workinghands: A hand-tool assembly dataset for image segmentation and activity mining," in *BMVC*, 2019.

[89] A. K. Urooj and A. Borji, "Analysis of hand segmentation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4710–4719.

[90] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1925–1934.

[91] S. Narasimhaswamy, Z. Wei, Y. Wang, J. Zhang, and M. Hoai, "Contextual attention for hand detection in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9567–9576.

[92] A. Wetzler, R. Slossberg, and R. Kimmel, "Rule of thumb: Deep derotation for improved fingertip detection," in *Proceedings of the British Machine Vision Conference (BMVC)*, M. W. J. Xianghua Xie and G. K. L. Tam, Eds. BMVA Press, September 2015, pp. 33.1–33.12. [Online]. Available: https://dx.doi.org/10.5244/C.29.33

[93] A. K. Bojja, F. Mueller, S. R. Malireddi, M. Oberweger, V. Lepetit, C. Theobalt, K. M. Yi, and A. Tagliasacchi, "Handseg: An automatically labeled dataset for hand segmentation from depth images," in *2019 16th Conference on Computer and Robot Vision (CRV)*, May 2019, pp. 151–158.

[94] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, vol. 36, no. 6, Nov. 2017.

[95] C. Zimmermann, D. Ceylan, J. Yang, B. Russell, M. Argus, and T. Brox, “Freihand: A dataset for markerless capture of hand pose and shape from single rgb images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 813–822.

[96] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, “First-person hand action benchmark with rgb-d videos and 3d hand pose annotations,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 409–419.

[97] K. Makantasis, A. Nikitakis, A. D. Doulamis, N. D. Doulamis, and I. Papaefstathiou, “Data-driven background subtraction algorithm for in-camera acceleration in thermal imagery,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2090–2104, 2017.

[98] C. Devaguptapu, N. Akolekar, M. M Sharma, and V. N Balasubramanian, “Borrow from anywhere: Pseudo multi-modal object detection in thermal imagery,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[99] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon, “Kaist multi-spectral day/night data set for autonomous and assisted driving,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 934–948, 2018.

[100] P. Wang and X. Bai, “Thermal infrared pedestrian segmentation based on conditional gan,” *IEEE transactions on image processing*, vol. 28, no. 12, pp. 6007–6021, 2019.

[101] S. Ward, J. Hensler, B. Alsalam, and L. F. Gonzalez, “Autonomous uavs wildlife detection using thermal imaging, predictive navigation and computer vision,” in *2016 IEEE Aerospace Conference*. IEEE, 2016, pp. 1–8.

[102] W. Hartmann, S. Tilch, H. Eisenbeiss, and K. Schindler, “Determination of the uav position by automatic processing of thermal images,” *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 39, p. B6, 2012.

[103] Y. Sun, W. Zuo, and M. Liu, “Rtfnet: Rgb-thermal fusion network for semantic segmentation of urban scenes,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2576–2583, 2019.

[104] Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, and T. Harada, “Mfnet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5108–5115.

[105] R. Luo, O. Sener, and S. Savarese, “Scene semantic reconstruction from egocentric rgb-d-thermal videos,” in *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 593–602.

[106] H. Guan, J. S. Chang, L. Chen, R. S. Feris, and M. Turk, “Multi-view appearance-based 3d hand pose estimation,” in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW’06)*. IEEE, 2006, pp. 154–154.

[107] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3593–3601.

[108] S.-H. Sun, M. Huh, Y.-H. Liao, N. Zhang, and J. J. Lim, "Multi-view to novel view: Synthesizing novel views with self-learned confidence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 155–171.

[109] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, 2017.

[110] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[111] J. Y. Jason, A. W. Harley, and K. G. Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," in *Computer Vision–ECCV 2016 Workshops*. Springer, 2016, pp. 3–10.

[112] R. Fransens, C. Strecha, and L. Van Gool, "Parametric stereo for multi-pose face recognition and 3d-face modeling," in *International Workshop on Analysis and Modeling of Faces and Gestures*. Springer, 2005, pp. 109–124.

[113] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[114] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, 2016.

[115] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[116] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[117] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[118] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 5, p. 169, 2014.

[119] *A 3D Face Model for Pose and Illumination Invariant Face Recognition*. Genova, Italy: IEEE, 2009.

[120] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[121] A. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "An information-rich 3d model repository. arxiv preprint," *arXiv preprint arXiv:1512.03012*, vol. 1, no. 7, p. 8, 2015.

[122] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention.* Springer, 2015, pp. 234–241.

[123] C. Choi, S. Kim, and K. Ramani, "Learning hand articulations by hallucinating heat distribution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3104–3113.

[124] M. Oberweger and V. Lepetit, "Deepprior++: Improving fast and accurate 3d hand pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 585–594.

[125] Y. Nirkin, I. Masi, A. T. Tuan, T. Hassner, and G. Medioni, "On face segmentation, face swapping, and face perception," in *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on.* IEEE, 2018, pp. 98–105.

[126] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[127] S. Hou, K. Lou, and K. Ramani, "Svm-based semantic clustering and retrieval of a 3d model database," *Computer-Aided Design and Applications*, vol. 2, no. 1-4, pp. 155–164, 2005.

[128] O. Remil, Q. Xie, H. Chen, and J. Wang, "3d shape synthesis via content–style revealing priors," *Computer-Aided Design*, vol. 115, pp. 87–97, 2019.

[129] Y. Liu, H. Pottmann, and W. Wang, "Constrained 3d shape reconstruction using a combination of surface fitting and registration," *Computer-Aided Design*, vol. 38, no. 6, pp. 572–583, 2006.

[130] Z. Shu, S. Xin, H. Xu, L. Kavan, P. Wang, and L. Liu, "3d model classification via principal thickness images," *Computer-Aided Design*, vol. 78, pp. 199–208, 2016.

[131] D. Yarotsky, "Geometric features for voxel-based surface recognition," *arXiv preprint arXiv:1701.04249*, 2017.

[132] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 37–45.

[133] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[134] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *European Conference on Computer Vision.* Springer, 2016, pp. 484–499.

[135] C. Zhang and Z. Zhang, "Improving multiview face detection with multi-task deep convolutional neural networks," in *IEEE Winter Conference on Applications of Computer Vision.* IEEE, 2014, pp. 1036–1041.

[136] S. Kim, N. Winovich, H.-G. Chi, G. Lin, and K. Ramani, "Latent transformations neural network for object view synthesis," *The Visual Computer*, pp. 1–15, 2019.

[137] F. Yu, K. Liu, Y. Zhang, C. Zhu, and K. Xu, "Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9491–9500.

[138] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2motion: Joint analysis of motion parts and attributes from 3d shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8876–8884.

[139] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas, "Deep part induction from articulated object pairs," in *SIGGRAPH Asia 2018 Technical Papers.* ACM, 2018, p. 209.

[140] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3d shape segmentation with projective convolutional networks," in *Proc. CVPR*, vol. 1, 2017, p. 8.

[141] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3 d shape descriptors," in *Symposium on geometry processing*, vol. 6, 2003, pp. 156–164.

[142] A. Sharma, O. Grau, and M. Fritz, "Vconv-dae: Deep volumetric shape learning without object labels," in *European Conference on Computer Vision.* Springer, 2016, pp. 236–250.

[143] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[144] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215.

[145] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 90, 2019.

[146] I. Kostrikov, Z. Jiang, D. Panozzo, D. Zorin, and J. Bruna, "Surface networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2540–2548.

[147] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3712–3722.

[148] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "On the effectiveness of least squares generative adversarial networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 12, pp. 2947–2960, 2018.

[149] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*. Springer, 2016, pp. 483–499.

[150] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas *et al.*, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 210, 2016.

[151] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[152] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares, "Efficient implementation of marching cubes' cases with topological guarantees," *Journal of graphics tools*, vol. 8, no. 2, pp. 1–15, 2003.

[153] J. Vollmer, R. Mencl, and H. Mueller, "Improved laplacian smoothing of noisy surface meshes," in *Computer graphics forum*, vol. 18. Wiley Online Library, 1999, pp. 131–138.

[154] S. H. Khan, Y. Guo, M. Hayat, and N. Barnes, "Unsupervised primitive discovery for improved 3d generative modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9739–9748.

[155] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[156] T. Furuya and R. Ohbuchi, "Deep aggregation of local 3d geometric features for 3d model retrieval." in *BMVC*, 2016, pp. 121–1.

[157] J. Huang, T.-H. Kwok, and C. Zhou, "Parametric design for human body modeling by wireframe-assisted deep learning," *Computer-Aided Design*, vol. 108, pp. 19–29, 2019.

[158] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[159] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[160] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[161] K. Ellis, D. Ritchie, A. Solar-Lezama, and J. Tenenbaum, “Learning to infer graphics programs from hand-drawn images,” in *Advances in neural information processing systems*, 2018, pp. 6059–6068.

[162] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, “Geometric deep learning on graphs and manifolds using mixture model cnns,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[163] G. A. Miller, *WordNet: An electronic lexical database.* MIT press, 1998.

[164] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision.* Springer, 2016, pp. 21–37.

[165] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[166] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[167] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[168] X. Jia, E. Gavves, B. Fernando, and T. Tuytelaars, “Guiding the long-short term memory model for image caption generation,” in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[169] A. Farag, A. Ali, J. Graham, A. Farag, S. Elshazly, and R. Falk, “Evaluation of geometric feature descriptors for detection and classification of lung nodules in low dose ct scans of the chest,” in *2011 IEEE International Symposium on Biomedical Imaging: from nano to macro.* IEEE, 2011, pp. 169–172.

[170] C. Geuzaine and J.-F. Remacle, “Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities,” *International journal for numerical methods in engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[171] M. Savva, F. Yu, H. Su, M. Aono, B. Chen, D. Cohen-Or, W. Deng, H. Su, S. Bai, X. Bai *et al.*, “Shrec16 track: largescale 3d shape retrieval from shapenet core55,” in *Proceedings of the eurographics workshop on 3D object retrieval*, 2016, pp. 89–98.

[172] L. van der Maaten and G. Hinton, “Visualizing high-dimensional data using t-sne,” 2008.

[173] R. Y. Wang and J. Popović, “Real-time hand-tracking with a color glove,” *ACM transactions on graphics (TOG)*, vol. 28, no. 3, p. 63, 2009.

[174] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, “Efficient model-based 3d tracking of hand articulations using kinect.” in *BmVC*, vol. 1, no. 2, 2011, p. 3.

[175] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, “Vision-based hand pose estimation: A review,” *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 52–73, 2007.

[176] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in *Consumer depth cameras for computer vision.* Springer, 2013, pp. 119–137.

[177] O. Glauser, S. Wu, D. Panozzo, O. Hilliges, and O. Sorkine-Hornung, "Interactive hand pose estimation using a stretch-sensing soft glove," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 41, 2019.

[178] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "3d convolutional neural networks for efficient and robust hand pose estimation from single depth images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1991–2000.

[179] M. Cai, K. M. Kitani, and Y. Sato, "Understanding hand-object manipulation with grasp types and object attributes." in *Robotics: Science and Systems*, vol. 3. Ann Arbor, Michigan;, 2016.

[180] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a wii controller," in *Proceedings of the 2nd international conference on Tangible and embedded interaction.* ACM, 2008, pp. 11–14.

[181] F. Zhan, "Hand gesture recognition with convolution neural networks," in *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, 2019, pp. 295–298.

[182] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1949–1957.

[183] Y. Li, Z. Ye, and J. M. Rehg, "Delving into egocentric actions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 287–295.

[184] Q. Zhang, Y. Liu, R. S. Blum, J. Han, and D. Tao, "Sparse representation based multi-sensor image fusion for multi-focus and multi-modality images: A review," *Information Fusion*, vol. 40, pp. 57–75, 2018.

[185] X.-Y. Wei, Y.-G. Jiang, and C.-W. Ngo, "Concept-driven multi-modality fusion for video search," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 1, pp. 62–73, 2011.

[186] S. Rastegar, M. Soleymani, H. R. Rabiee, and S. Mohsen Shojaee, "Mdl-cw: A multimodal deep learning framework with cross weights," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[187] D. Hu, F. Nie, and X. Li, "Deep multimodal clustering for unsupervised audiovisual learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[188] K. Chen, T. Bui, C. Fang, Z. Wang, and R. Nevatia, "Amc: Attention guided multi-modal correlation learning for image search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2644–2652.

[189] A. Burton, "The range and variability of the blood flow in the human fingers and the vasomotor regulation of body temperature," *American Journal of Physiology-Legacy Content*, vol. 127, no. 3, pp. 437–453, 1939.

[190] M. Kieu, A. D. Bagdanov, M. Bertini, and A. Del Bimbo, "Domain adaptation for privacy-preserving pedestrian detection in thermal imagery," in *International Conference on Image Analysis and Processing.* Springer, 2019, pp. 203–213.

[191] X. Font-Aragones, M. Faundez-Zanuy, and J. Mekyska, "Thermal hand image segmentation for biometric recognition," *IEEE Aerospace and Electronic Systems Magazine*, vol. 28, no. 6, pp. 4–14, 2013.

[192] J. Appenrodt, A. Al-Hamadi, and B. Michaelis, "Data gathering for gesture recognition systems based on single color-, stereo color-and thermal cameras," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 3, no. 1, pp. 37–50, 2010.

[193] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding convolution for semantic segmentation," in *2018 IEEE winter conference on applications of computer vision (WACV).* IEEE, 2018, pp. 1451–1460.

[194] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 1328–1338.

[195] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010.* Springer, 2010, pp. 177–186.

[196] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.