Single Sketch Image based 3D Car Shape Reconstruction with Deep Learning and Lazy Learning

Naoki Nozawa[1], Hubert P. H. Shum[2,*,a], Edmond S. L. Ho[2,b] and Shigeo Morishima[3]

[1]*Department of Pure and Applied Physics, Waseda University, Tokyo, Japan*

[2]*Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K.*

[3]*Waseda Research Institute for Science and Engineering, Waseda University, Tokyo, Japan*

Keywords: Deep Learning, Lazy Learning, 3D Reconstruction, Sketch-based Interface, Car.

Abstract: Efficient car shape design is a challenging problem in both the automotive industry and the computer animation/games industry. In this paper, we present a system to reconstruct the 3D car shape from a single 2D sketch image. To learn the correlation between 2D sketches and 3D cars, we propose a Variational Autoencoder deep neural network that takes a 2D sketch and generates a set of multi-view depth and mask images, which form a more effective representation comparing to 3D meshes, and can be effectively fused to generate a 3D car shape. Since global models like deep learning have limited capacity to reconstruct fine-detail features, we propose a local lazy learning approach that constructs a small subspace based on a few relevant car samples in the database. Due to the small size of such a subspace, fine details can be represented effectively with a small number of parameters. With a low-cost optimization process, a high-quality car shape with detailed features is created. Experimental results show that the system performs consistently to create highly realistic cars of substantially different shape and topology.

1 INTRODUCTION

Car shape design is a common area in automotive manufacturing, computer animation and games. The design process is time-consuming and labour-intensive, as it is a combination of arts and engineering. In the automotive manufacturing industry, concept arts of the car surface, which are typically represented as sketches on predefined viewpoints, are designed first. Such concept arts are then converted into 3D for engineering design. However, there is no guarantee that such a design can be finalized until the engineers confirmed the interior fitting of mechanical parts. Sometimes, it may take several iterations to fulfil the requirements of both design and engineering aspects. In the animation and games industry, while there is no real-world engineering constraint, often the artists need to build a large number of cars that resemble similar features from the real-world ones. We believe that with an automatic system that can generate realistic 3D cars based on simple 2D sketches, the design of cars, especially in the initial stages of concept designs, can be a lot more efficient.

There are two major challenges of reconstructing 3D cars from 2D sketches. First, to improve the efficiency of car design, artists want to minimize the number of sketches required to design each car. As a result, there is not enough information to fully define the 3D features of a car. Second, cars modelling is a distinct problem as cars have common features such as where to place the wheels, but also distinctive parts such as the shape of rear wings and roofs. Past research (Umetani, 2017) shows that it is challenging to learn a diverse car subspace that represents both common and distinctive car features well.

In this paper, we propose a new 3D car design interface that is based on a single 2D sketch, which contains only the outline information on the car's shape. Since a single outline sketch cannot provide enough information on 3D car reconstruction, our framework estimates such missing information from a 3D car shape database. Collecting pairwise samples of sketches and 3D car shapes is costly, which hinders the use of powerful data-driven methods such as deep learning for learning the reconstruction. We propose to synthesize sketches from 3D car models

[a] https://orcid.org/0000-0001-5651-6039
[b] https://orcid.org/0000-0001-5862-106X
[*] Corresponding author

Nozawa, N., Shum, H., Ho, E. and Morishima, S.
Single Sketch Image based 3D Car Shape Reconstruction with Deep Learning and Lazy Learning.
DOI: 10.5220/0009157001790190
In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020) - Volume 1: GRAPP*, pages 179-190
ISBN: 978-989-758-402-2; ISSN: 2184-4321
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

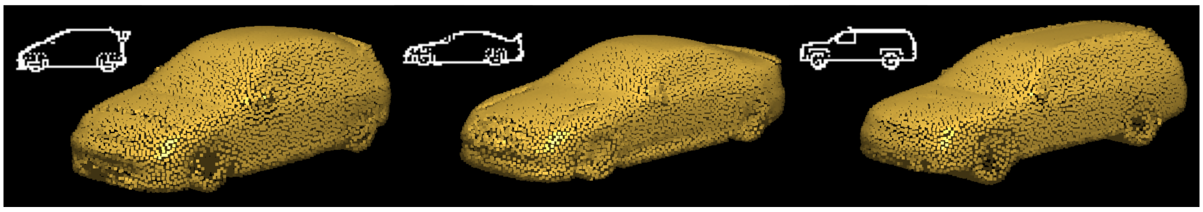


Figure 1: Examples of 3D car shapes generated by out system with side-view sketches.

obtained from ShapeNet (Chang et al., 2015). We further propose a feature-preserving car mesh augmentation pipeline to increase the size of the database.

To tackle the challenge of modelling 3D car shapes, we propose a novel 2-stage framework. The first stage adapts the Variational Autoencoder (VAE) (Kingma et al., 2014) deep learning network for correlating a 2D sketch with the respective 3D mesh. Since it is inefficient to learn the mesh in 3D, and 2D to 3D correlation is not trivial, we propose to learn an intermediate representation of multiple depth and mask images instead, and reconstruct the 3D car mesh as a post-processing step. Global systems like deep learning have limited capability in representing fine details (Umetani, 2017; Güler et al., 2018). As a solution, we introduce a second processing stage that adapts a lazy learning framework to learn a local subspace from the relevant samples in the database. Such a subspace enables a low-cost optimization process to generate a 3D car shape with fine details.

Experimental results show that high-quality car shapes of substantially different shapes and topologies can be generated with rough 2D sketch images (Figure 1). Apart from resembling the overall shape in the 2D sketches, the generated car shapes contain fine-detail features such as rear wings. The process takes around 15 seconds to generate a shape using a low-end computer.

The major contributions of this paper are summarized as follows:

- We propose a Variational Autoencoder (VAE) (Kingma et al., 2014) deep learning network to learn the correlation between a 2D sketch and the corresponding 3D shape. Instead of learning the 3D shape directly, we propose to learn an intermediate multi-view depth and mask images representation, which are then combined to form the 3D shape, for better training performance.
- We propose a lazy learning algorithm to learn a local subspace to reconstruct the fine detail features of the car. Such a subspace bases only on the relevant car shapes in the database, and therefore facilitates effective training and robust usage.
- To facilitate the training processes, we propose a feature-preserving mesh augmentation framework to construct a large car database with pairwise 3D mesh and 2D sketch, based on the small number of car meshes in ShapeNet (Chang et al., 2015).

The rest of the paper is organized as follows. We review previous work in Section 2. We explain the construction of our car database in Section 3. We present our Variational Autoencoder for generating 3D car shapes from 2D sketches in Section 4. We present our lazy learning for constructing fine details for the car in Section 5. We show the results of our system in Section 6. We conclude the paper and discuss possible future directions in Section 7.

2 RELATED WORK

In this section, we review related work in the areas of sketch-based interfaces for 3D design and machine learning for 3D shape reconstruction.

2.1 Sketch-based Interfaces for 3D Design

Contrary to professional computer-aided design software that requires professional training and has an engineering focus, sketch-based interfaces (Olsen et al., 2009) are more designer-friendly. Existing research has used sketches for a variety of applications such as image retrieval (Eitz et al., 2011), motion synthesis (Thorne et al., 2004), clothes design (Turquin et al., 2007) and crowd control (Henry et al., 2012).

In particular, we are interested in utilizing sketches to create 3D meshes. Igarashi et al. (Igarashi et al., 2007) propose a 3D modelling system via a sketch drawing interface that can reconstruct 3D polygon surfaces by specifying contours. They further extend the method such that it construct smooth mesh surfaces (Igarashi et al., 2006) and the internal structure of meshes (Owada et al., 2006). Nealen et al. (Nealen et al., 2007) also extend this method to support ridge and valley area, while Gingold et al. (Gingold et al., 2009) focus the extension on specifying extra annotations such as angles and symmetrical information for generating the more detailed surface.

Joshi et al. (Joshi and Carr, 2008) generate a 3D inflating surface that interpolates input curves from a sketch image while using mean curvature at boundary vertices. Shtof et al. (Shtof et al., 2013) propose a 3d modelling method that fits predefined 3D primitives to the specified sketched curves. Schmidt et al. (Schmidt et al., 2009) present a modelling interface that constrains 3D lines and curves from single-view sketches to generate surfaces in the right spatial location. Shao et al. (Shao et al., 2012) infer the normal map in an outline sketch with cross-sections to generate a 3D shape.

The focus of these works is to enable a user to specify the important information of a mesh, while utilizing artificial intelligence to estimate the unspecified information. Since the big data of 3D shapes has become more available, we are more interested in machine learning-based approaches for shape reconstruction.

In recent years, machine learning approaches based on big data has shown to be effective in modelling the relationship between sketches and 3D meshes. Han et al. (Han et al., 2017) propose a convolutional neural network (CNN) based system to generate 3D faces from input sketches. Nishida et al. (Nishida et al., 2016) adapt a CNN to generate building models by adding surface curve information as a style of sketching. We also utilize deep learning for constructing the sketch-based interface, but we adapt the Variational Autoencoder (VAE) (Kingma et al., 2014) for correlating the 2D sketch and the output represented as depth and mask images. This is because such a generative model has shown promising results in the translation of image style.

2.2 Machine Learning for 3D Shape Reconstruction

The core of the problem of 3D shape reconstruction is an effective representation of shapes. With the introduction of comprehensive 3D shape databases such as (Chang et al., 2015) and PASCAL3D (Xiang et al., 2014), it has become possible to learn a more representative latent space to represent and reconstruct 3D shapes. In general, shapes can be represented as point clouds, voxel and depth maps.

The point cloud representation has been used by Fan et al. (Fan et al., 2017), in which they reconstruct a 3D mesh with a single real-world image. Charles et al. (Charles et al., 2017) presented PointNet, which directly consume unstructured point clouds for a range of applications such as 3D classification and segmentation. They further extend the method into PointNet++ (Qi et al., 2017) to capture local structures such that the network can recognize fine-grained patterns. Groueix et al. (Groueix et al., 2018) utilize point cloud or an image as an input to reconstruct 3D surfaces, but they infer a surface representation directly for the output shape utilizing the UV coordinate. There has been little work on reconstructing point clouds with sketches, as it is more difficult to find a good correlation between the 2D sketches and the unstructured 3D point clouds.

The voxel representation has been used by Delanoy et al. (Delanoy et al., 2018). They take a sketch as an input, and construct a CNN to predicts occupancy of a 3D voxel grid that represents the output mesh. Choy et al. (Choy et al., 2016) follow the idea of using CNN to generate a volumetric occupancy map for 3D reconstruction, and their method does not require 2D image annotations or 3D object classes. To enhance memory efficiency, Tatarchenko et al. (Tatarchenko et al., 2017) propose to learn both the structure of an octree and its corresponding occupancy values of individual cells. Wang et al. (Wang et al., 2017) also propose an octree-based CNN for shape analysis. Despite this optimization, voxel methods are still a memory and computational cost consuming due to the need to model a 3D space.

The depth map representation has been used by Lun et al. (Lun et al., 2017) to reconstruct 3D surface form sketches. They use both depth and normal maps to represent the shape, causing the use of extra memory. Due to the use of the U-net structure (Ronneberger et al., 2015) like pix2pix (Isola et al., 2017), the input sketch structure is over-preserved and sketches that do not look like the objects in the database cannot be effectively reconstructed. Li et al. (Li et al., 2018a) propose a method to generate a detailed surface with considering curvature flow. They construct multiple decoders for depth, mask and curvature flow, and therefore the system requires extra memory. We also utilize depth maps as our network output to model the 3D shape; however, our system only needs to generate depth images and binary masks, without the need for normal maps or curvature flow. We propose a Laplacian-based loss function to compensate for the lack of normal maps, allowing a more memory efficient system.

As a global model, deep learning has shown to have difficulties in representing fine details of the shapes that are specific to a small cluster of samples (Umetani, 2017). In the area of car reconstruction, different categories of cars have different specific details such as side mirrors and rear wings. Past research has shown that local models utilizing lazy learning can help to preserve fine details in different problems. Chai et al. (Chai and Hodgins, 2005) generate a human surface from a sparse input with a large motion database. Shen et al. (Shen et al., 2018) map complex gestures to crowd movement for gesture-based crowd control. Shum et al. (Shum et al., 2013) reconstruct noisy human motion captured by Kinect. The main idea is to extract relevant data based on a run-time query and construct a local model during run-time. In this work, we adapt lazy learning to generate the fine details of a car based on the output generated by a deep learning network.

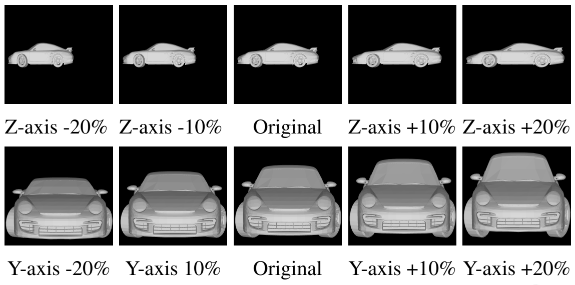


Figure 2: Examples of 3D car meshes synthesized with our feature-preserving data augmentation method.

3 DATABASE CREATION

As a data-driven approach, the diversity and quality of the database is key to our system. We present a robust and efficient process to construct a 3D car mesh database. With a novel feature-preserving data augmentation techniques, we create a large variety of logically correct car meshes. They are converted into two sets of representations: (1) 2D sketch, depth and mask images for shape reconstruction, and (2) registered 3D point clouds for details synthesis.

3.1 Feature-preserving Car Mesh Augmentation

We create our 3D car mesh by adapting data argumentation process, which enhances the diversity and size of our car model database by synthesizing high-quality data.

While there are plenty of car images available online, training a reliable 2D to 3D neural networks would require pairwise 2D and 3D training samples. Such pairwise data is challenging to obtain in the real-world. Therefore, we propose to synthesize training data from 3D car models, which are of high-quality and can be used to produce different types of 2D images via projections.

We first gather car models from the ShapeNet (Chang et al., 2015), which is one of the most comprehensive 3D shapes databases. However, the number of car models is not enough for training a deep neural network. To solve the problem, we employ data argumentation to synthesize more data samples.

Traditional linear scaling such as (Sela et al., 2017) does not work well for car meshes because different parts of the car have to be handled in different ways. For example, simply scaling the height of the car would make a taller car body, but the wheels will become ovals, which is logically incorrect.

As a solution, we adapt Kraevoy et al.'s method (Kraevoy et al., 2008) for data argumentation, which can resize the car shape while preserving important features. The method utilizes a voxel grid that is resized non-homogeneously according to vulnerability of each voxel, and interpolate mesh vertices based on the edited voxel grid. It then calculates the curvature of contained vertices in each cell with the consideration of neighbourhood vulnerability, and perform mesh scaling while maintaining local features.

In our implementation, we set the resolution of the voxel as $5 \times 10 \times 15$ in the x, y, and z axes respectively, and use Akenine-Möllser's method (Akenine-Möllser, 2001) to detect the intersection between vertices and voxel. Finally, we resize the voxel grid along with the height (i.e., y-axis) and length (i.e., z-axis) directions only. This scaling in height and length only is because the width direction is restricted by that of the road, which is a constraint also applied in existing car design processes. We set the resizing parameter as $\pm 20\%$, $\pm 15\%$, $\pm 10\%$ and $\pm 5\%$ of each scalable direction, and interpolate mesh vertices based on the resized voxel grid using radial basis function (RBF) interpolation.

With the original data of 7,028 cars, we synthesized 56,224 cars using the aforementioned data augmentation method. Figure 2 shows some examples of the resized 3D models, in which features like the wheels and the headlight are largely preserved.

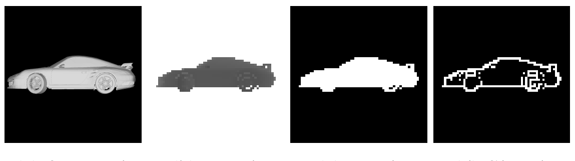


Figure 3: An example of converting a 3D car model into different types of images.

3.2 The 2D Sketch, Depth and Mask Images Representation

With the 3D meshes created, we produce the 2D sketch, depth and mask images representation for

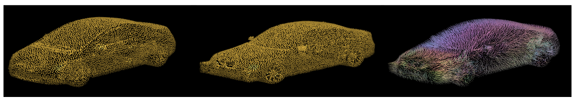


Figure 4: From left to right: the template, a car shape, and the flow for mapping them.

training our deep learning system on car shape reconstruction. Figure 3 shows an example of converting a 3D car mesh into different types of images.

To create the depth and mask images, we set up a bounding box that contains the whole car. Then, we emit rays from each face of the bounding box and obtain the nearest intersection points with the car mesh. We ignore the bottom face of the car and do not produce the corresponding depth and mask images. This is because the bottom of the car typically consists of highly complicated geometry involving holes and complex shapes for mechanical gears.

We synthesize sketch images by employing a Laplacian filter on the projected images of the car meshes from the sides (i.e., x-axis), which can generate reliable sketch-like images that convey the outline of the car meshes. This method works robustly and does not require human intervention.

3.3 The Registered 3D Point Cloud Representation

We also create a point cloud representation from the 3D meshes for the system to learn how to add details to the car shapes. Since the focus of the research is the outer shape of the cars, we remove internal vertices inside the car body and generate a flat bottom plane for each car. These allow us to obtain a set of clean car shape point clouds that can be robustly registered to discover the correspondence among car shapes.

We convert the mesh into a point cloud by sampling the mesh surface uniformly using Corsini et al.'s method (Corsini et al., 2012). The major challenge is that the method is controlled based on sampling density, which is fed into a Poisson disk sampling process, instead of the total number of points. Because of this, cars of different size have a dramatically different number of sample points, which posts unnecessary complexity and unpredictable computational cost in the later on processes. Therefore, we implement an iterative function to find out the required sampling density of each car that can produce the point number within a predefined range. To further enforce the same number of points among cars, we randomly remove points sampled until the number reach the target. We found that point number of 10,000 has a good balance of visual quality and computational cost.

Finally, we propose a registration process to align the point clouds of different cars. Motivated by (Henry et al., 2014; Shen et al., 2019), we formulate the problem as an optimal transportation problem. We first pick a random point cloud of a car as a template, and then evaluate the Earth Mover's Distance (EMD) between the template and the rest of the cars. We utilize squared Euclidean cost as suggested by (Li et al., 2018b). Since we have the same number of points for all point clouds, the optimal flow from the template to the target point cloud of a car has a 1-to-1 correspondence, which therefore is considered as the registration result. Comparing to deformation-based methods (Amberg et al., 2007; Li et al., 2008) that fit a template to the target shape, our method is more robust to cars that have different mesh topologies. Comparing to feature point-based methods (Rusu and Cousins, 2011) that calculate the mapping between the template and the target shape directly using feature points, our method generates a much denser correspondence.

4 VARIATIONAL AUTOENCODER FOR CAR SHAPE RECONSTRUCTION

In this section, we present a deep neural network to reconstruct 3D car shapes from 2D sketches.

Following the advice and design culture of the car manufacturing industry, we take a single side view sketch as the input of our system. However, our system is expandable to supporting sketches from multiple views by duplicating the network architecture.

Instead of directly outputting the 3D point cloud (Fan et al., 2017; Charles et al., 2017) or the voxels (Delanoy et al., 2018; Choy et al., 2016) of a car, we propose to output a set of depth and mask images from the side, top, front and rear views, and reconstruct the 3D vertices by combining them. This is mainly due to the high complexity and memory cost for a network to maintain a smooth surface while preserving the volume for complicated 3D shapes.

We adapt Variational Autoencoder (VAE) (Kingma et al., 2014) for getting the depth and mask images, as such a generative model has shown promising results in image translation by altering the output with a different style. While VAE is inferior to Generative Adversarial Network (GAN) (Isola et al., 2017; Chen and Koltun, 2017; Wang et al., 2018) in terms of the appearance of the output, it is difficult to control the image synthesis process in GAN to create multiple outputs. Also, it takes much longer to train GAN and to guarantee network convergence. In our situation, we prefer VAE as it produces results with high enough quality to generate a car shape, and the details of the car are introduced as a second stage process.

4.1 Network Design

We adapt an encoder-decoder network structure for generating the depth and mask images, as shown in Figure 5. The decoder needs to generate both images in multiple predefined views. Existing research typically prepares multiple decoders, with one decoder generating one output view (Li et al., 2018a; Lun et al., 2017). However, such an approach increases computational cost and memory requirement significantly, considering that we need to generate four different views (i.e., front, rear, side, and top).

As a solution, our network shares the decoder among multiple views at the first layer, in which we have an independent final layer for each view. Each output from the last layer has two channels, which are the depth and mask images respectively. This design is driven by the observation that there is shared information across different views. By sharing the first layer in the decoder, such information can be discovered. Apart from the massive reduction in memory usage and training time, such a setup allows the different output views to be more coherence and produces higher quality results. We justify our choice in the decoder network design by conducting an ablation test in Section 6.3.

4.2 The Loss Function

VAE typically involves two types of the loss function, which are the cross-entropy loss (i.e., reconstruction loss) and the Kullback-Leibler (KL) divergence loss. In the following, we explain our design of loss functions.

For the mask image, we use binary cross-entropy loss between the generated images and the ground truths. Such a loss is effective for data that follow the Bernoulli distribution like the mask images. Each mask image is normalized from [-1, 1] to [0, 1] to facilitate the binary cross-entropy. The loss has the effect of detecting the area where the car shape should exist in the world coordinate.

For the depth images, in order to ignore pixels that do not belong to the car, we mask the estimate depth images with the estimated mask images when calculating the loss. We implement two loss functions - the mean absolute error (MAE, L1 loss) on the masked depth images, and another MAE on their Laplacian representations.

We implement MAE between the generated depth images and the ground truths. MAE is used because depth images do not follow the Bernoulli distribution and therefore is not suitable for binary cross-entropy loss. While it is possible to apply cross-entropy loss or mean square error (L2 loss) by assuming Gaussian data distribution, such losses are known to be too sensitive to outlier and are less robust in deep learning. On the contrary, L1 loss considers neighbourhood information in image generation (Isola et al., 2017) and is suitable for depth estimation whose output tends to have a lower frequency than natural images.

We apply a second MAE loss on the Laplacian representation of the generated depth images and the ground truths. We apply a second MAE loss because while the first loss can represent the shape and position of the car, this first loss is too weak to represent surface appearance. The Laplacian representation is computed by applying the following Laplacian filter to the depth image:

$$\begin{bmatrix} \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & -1 & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \end{bmatrix} \tag{1}$$

The Laplacian representation is comparable to a normal representation, but it can be directly deduced from the depth images. Some may consider training a multi-task network with ground-truth normal images (Li et al., 2018a; Lun et al., 2017), but such an approach is memory consuming and difficult for hyper-parameters tuning (e.g. the relative weight between tasks). Our Laplacian MAE loss can represent the surface appearance with no impact on the network size.

Finally, a standard KL divergence loss added to the VAE network structure that encodes the latent vector. We assume that the latent vector follows a Gaussian distribution and can be expressed using mean and standard deviation.

The final loss function is expressed as:

$$\begin{aligned} E = & \left(\boldsymbol{D}^{ref} - \boldsymbol{D}^{rec}\right) \circ \boldsymbol{M}^{ref}\big)_{L1} + \left(\boldsymbol{M}^{ref} - \boldsymbol{M}^{rec}\right)_{BCE} \\ & + \left(\left(\Delta\boldsymbol{D}^{ref} - \Delta\boldsymbol{D}^{rec}\right) \circ \boldsymbol{M}^{ref}\right)\big)_{L1} + KLLoss \end{aligned} \tag{2}$$

where $\boldsymbol{D}^{ref}$ and $\boldsymbol{M}^{ref}$ are Depth and Mask images of the ground truths, $\boldsymbol{D}^{rec}$ and $\boldsymbol{M}^{rec}$ are those of reconstructed images, the subscripts *L1* and *BCE(binary cross-entropy)* represent the calculation metrics, Δ means Laplacian filtering and $\circ$ is the Hadamard product, *KLLoss* is the standard KL loss function.

4.3 Surface Reconstruction

Having generated the depth and mask images in different views as the output of our proposed deep learn-

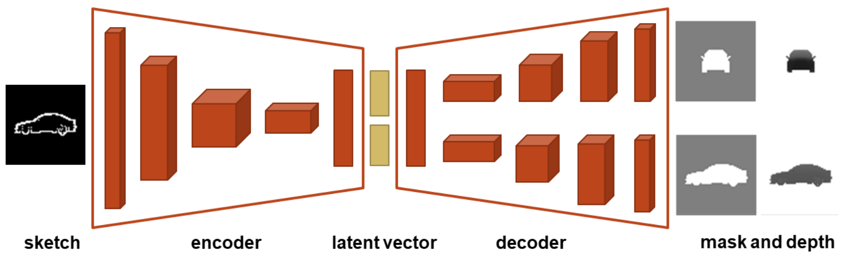


Figure 5: The encoder-decoder network structure.

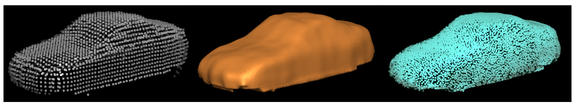


Figure 6: From left to right: the reconstructed shape by the network, the reconstructed surface, and the registered point cloud.

ing framework, a rough 3D point cloud is reconstructed as a post-processing steping step. Specifically, a point cloud that represents part of the car can be computed from the mask and depth image pair in each view. The shape of the whole vehicle can be reconstructed by combining the point clouds extracted from all views (Figure 6 (left)). Next, Poisson surface reconstruction is applied to the point cloud to reconstruct the entire surface of the car (Figure 6 (middle)). The surface of the car enables us to sample points uniformly, which facilitates the point cloud registration process. This step is equivalent to the point cloud standardization process as in the database creation (Section 3.3). Finally, the registered point cloud (Figure 6 (right)) computed from the output of our proposed deep learning framework can be compared with our car database in the lazy learning stage in Section 5.

4.4 Implementation

The system is implemented in Tensorflow. For optimization, we use Adam solver with a learning rate 1e-5. The decoder has a dropout ratio of 0.5 except for the last layer. We use ReLU as the activation function for the hidden layers in the encoder, and Leaky ReLU for that in the decoder, as in pix2pix (Isola et al., 2017). We use *tanh* as the activation function for output layers. To speed up the system, the resolution of the input images in VAE is only 64×64. We train the system using the data generated by data augmentation, and test the system with the original data from ShapeNet (Chang et al., 2015). This ensures that accurate testings are done with unaltered data.

5 LAZY LEARNING FOR FINE DETAILS

While the main bodies of cars share a lot of common geometric similarities, the fine details such as side mirrors and rear wings can be different. Learning a global model from all cars for fine details is therefore highly ineffective. Motivated by the success of lazy learning in mesh processing (Chai and Hodgins, 2005; Shen et al., 2018; Ho et al., 2013), we propose to adapt lazy learning to reconstructed the details of the car shape.

Unlike traditional machine learning approaches that generalize data in the whole database as a preprocess, lazy learning postpones the generalization to run-time (Chai and Hodgins, 2005). As a result, lazy learning can utilize run-time information to limit the scale of learning. In particular, given a run-time query, relevant data in the database can be extracted and a small scale learning process can be performed. Due to the consideration of the most relevant data, the common features can be represented, even though those features may be insignificant on a global scale. Also, the similarity of relevant data allows lazy learning to use a much lower dimensional latent space comparing to traditional methods.

5.1 Relevant Data Search

Here, we explain how we search for relevant data from the database to perform lazy learning. Given a car shape generated in Section 4, we search for the k nearest samples from the database. As the point cloud is registered (i.e., it aligns with a pre-defined template car shape), we can effectively calculate the distance using the sum of Euclidean distances from all points between two point clouds. However, such a search may focus too much on fine details, which the generated car shape does not have, and is slow due to the high dimensionality of the point clouds.

As a solution, we propose to apply Principal Component Analysis (PCA) onto the position of the point clouds to generate a search space, instead of using the Cartesian space. Searching with the more important components of PCA allows a faster search with less focus of fine details. From the experiments, we found that the first 40 components can represent the point cloud with $10,000$ 3D points reasonably well, with around 90% of accumulated variance ratio. We, therefore, perform a database search by considering the root mean square distances of the 40 PCA components to find k nearest neighbours. Empirically, we found that setting $k = 5$ produces good results.

5.2 Learning and Optimization in Local Space

With the k nearest neighbours selected from the database, we can then learn a small subspace with PCA. Since these neighbours are similar to each other, the details of the shape can be well preserved with a smaller number of components. In such a subspace, we optimize a set of eigenvalues to construct a car shape that is as similar as possible to the one generated by deep learning. We then back project the eigenvalues to formulate a car shape with shape details such as the headlight, which is served as our final output.

We utilize the 3D Morphable Model (Blanz and Vetter, 1999) to optimize the eigenvalues of the considered components with a non-linear optimization process. To evaluate the distance between the optimizing shape and target shape, we utilize the Cartesian space. Since the point clouds are registered, a simple point-to-point Euclidean distance works well. Obtaining the Cartesian representation of the optimizing shape is simple - we back-project the optimized eigenvalues to the Cartesian space.

To further improve the optimization process, we find that a simple pre-process can help to construct an even more representative local PCA space with the k nearest neighbours. The idea is based on the observation that there are still small variations in car shapes within the k nearest neighbours, which distracts the system from the main objective of obtaining the detailed shape features. As a solution, we pre-optimize each of these shape using the same 3D Morphable Model-based optimization process described above, such that they all have a similar car shape with the target one, before we construct the local PCA space. This way, the significant components of the local PCA space can represent more on the detailed shape features.

6 EXPERIMENTAL RESULTS

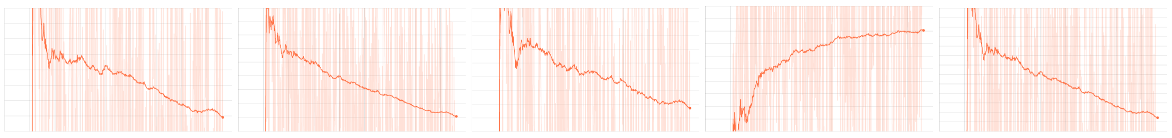
In this section, we will first present the experimental results on reconstructing 3D car shape from input sketches. Next, we quantitatively analyze the training loss during the training process to show the convergence of the proposed framework. Finally, an ablation test will be presented to demonstrate the results obtained from different decoder network architectures and justify our choice.

The training of the deep learning system is performed with an NVIDIA TITAN X Pascal GPU that has 12GB VRAM. With the batch size of 32, the training finishes within a day. The run-time system is performed on a lower-end computer with an NVIDIA GeForce 1060 GPU that has 3GB VRAM, an Intel Core i7-6700K CPU and 16GB of RAM. The reconstruction of a car takes around 15 seconds to finish, with 5 seconds on car shape reconstruction (i.e., deep learning) and 10 seconds on reconstruction detail features (i.e., lazy learning).

6.1 Reconstructing 3D Shape from Input Sketches

Since different users may have different drawing styles (e.g. more cartoon-like), real-world sketches are not objective to evaluate the performance of the proposed system. As a result, we utilize the synthetic sketch images for testing.

Figure 8 shows the system output during each of the processing steps, including the input sketches, the meshes constructed from depth images (i.e., the output of deep learning), registered point clouds from those meshes and detailed added point clouds (i.e., the output of lazy learning). It can be observed that the car shapes constructed by deep learning already have similar car body with the input sketch. The final results with the detailed added by the lazy learning are highly realistic, with side mirrors and rear wings. More results can be found in Figure 1. However, details like grilles or wheels are not encoded well for practical use of games or movies. The EMD registration process can cause such low-quality appearances because EMD is based on optimal transportation with global distribution, which can ignore small features. Also, the converting process into point clouds can reduce mesh resolution that is closely related to details. We will consider landmarks on 3D mesh in the sampling and registration process. Furthermore, the input sketches can affect appearance because of sparse information comparing with photorealistic images. Feature extraction from sketches is still an open problem in the field of deep learning, so we will up-



(a) Depth Loss (b) Mask Loss (c) Laplacian Loss (d) KL Loss (e) Total Loss

Figure 7: Losses across epoch during the training stage.

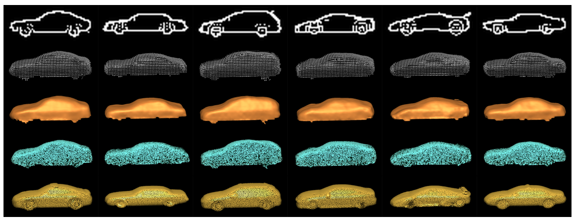


Figure 8: Intermediate outputs. From top to bottom: sketches, meshes from generated depth images, reconstructed surfaces, sampled point clouds on surfaces, and point clouds with details.

date our network structure. An interactive sketch-based system will improve appearance as well.

We found that while the point clouds generated closely resemble real cars, the generated mesh may have artefacts around sharp edges. This is due to a well-known point sampling problem in which the system samples points around sharp edges instead of the edges themselves. When performing triangulation to obtain the mesh from the point cloud, there can be some zigzag pattern around those edges. We ignore the problem here since sampling is not a major focus of our paper, but more advanced sampling methods such as (Gauthier and Poulin, 2009) can be employed if needed.

Figure 9 shows the effect of k in lazy learning. Figure 9 shows that our system can reconstruct the car details, and the choice of $k = 5$ generates good quality of point clouds with a high level of detail. In general, a higher k produces smoother results, while a smaller k results in less power to reconstruct shape that fits the target. We also show some failure cases in Figure 10, in which the generated point clouds are either too noisy or not similar to the input sketches. We deduce that this is likely because the car styles related to these input sketches are not common in the database, and the deep learning system does not learn a stable manifold for such samples.

6.2 Training Loss

To quantitatively evaluate the effectiveness of the proposed framework, the plot of different training losses during the training process is illustrated in Figure 7. It indicates that the training process is stable, and the depth, mask and Laplacian losses are small when the

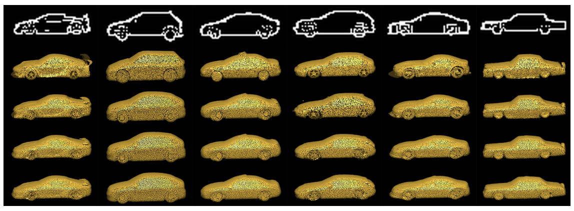


Figure 9: Results generated with different k. From top to bottom: sketches, point clouds with $k = 1$, $k = 3$, $k = 5$ and $k = 7$.

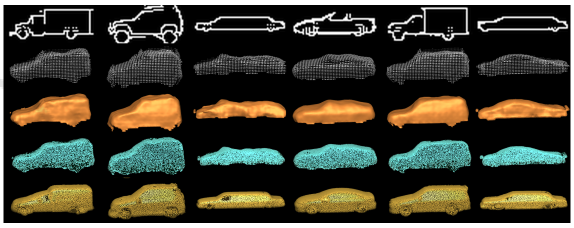


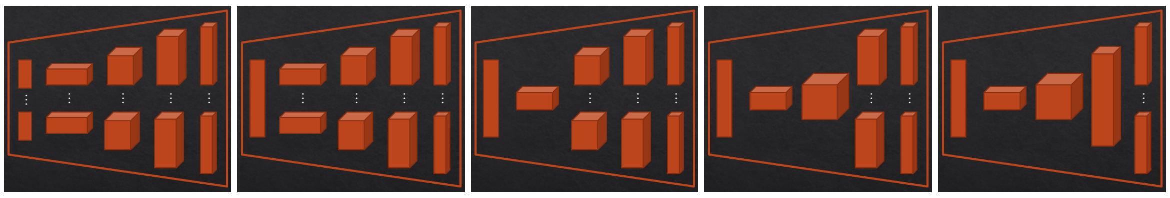
Figure 10: Lower-quality results for sketches that has few similar samples in the database.

system converges. Notice that the KL loss works as a constraint and thus the converging direction is upside-down.

6.3 Ablation Tests on Different Decoder Network Architectures

As explained in Section 4.1, decoders are used for generating depth and mask images in different views for reconstructing the 3D shape of the car from the input sketches. While the images in different views have a different appearance, they are associated with the same underlying 3D shape. This motivated us to share a common layer among the decoders in the network design to preserve the underlying structure and improve the consistency among all synthesized views. In our proposed encoder-decoder network structure (see Figure 5), each decoder consists of five layers. In the ablation test, we vary the number of shared layers in the decoder from 0 (i.e., not sharing any layer) to 4. The different decoder architectures are illustrated in Figure 11.

A wide range of 3D car shapes are reconstructed using different decoder network architectures and the results are illustrated in Figure 12. It can be seen that



(a) No shared layer (b) A one-layer shared (c) Two layers shared (d) Three layers shared (e) Four layers shared

Figure 11: Different decoder network architectures for the ablation test.

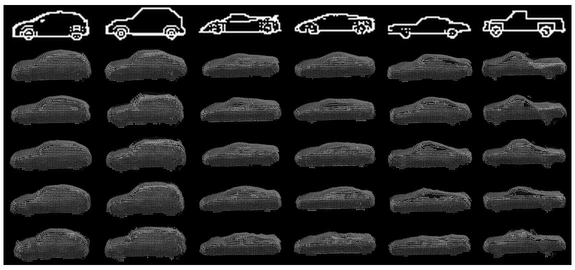


Figure 12: The 3D point clouds reconstructed with different decoder architectures. From top to bottom: input sketches, results of the decoder with no shared layer, sharing the first layer, sharing the first two layers, sharing the first three layers, and sharing first four layers.

our proposed decoder architecture with sharing only the first layer (3rd row in Figure 12) produces the best results in terms of reproducing the car shape with a smooth surface. On the other hand, sharing too many layers (5th and 6th rows in Figure 12) and not sharing any layer (2nd row in Figure 12) result in noisier 3D point clouds, which can be caused the loss of balance between preserving the underlying structure among decoders and refining each view. Sharing the first two layers (3rd row in Figure 12) also produces results with good visual quality, however, the reconstructed 3D shape is less similar to the input sketches when compare with the results obtained from our proposed network. This highlights the optimally of our proposed network design.

7 SUMMARY AND FUTURE DIRECTIONS

In this paper, we present a system to reconstruct detailed 3D car shapes with a single 2D sketch image. To effectively learn the correlation between 2D sketches and 3D cars, we propose a Variational Autoencoder (VAE) with an intermediate multi-view depth image representation as to the output, and construct the 3D cars as a post-processing step. To ensure the volume and diversity of the training data, we propose a feature-preserving augmentation pipeline to synthesize more car meshes while keeping the shape of important features such as the wheels (Figure 2). Finally, since deep learning has limited capability in representing fine details, we propose a lazy learning algorithm to construct a small subspace-based only on a few relevant database samples for optimizing a car shape with fine-detail features. We show that the system performs robustly in creating cars of substantially different shape and topology, with realistic detailed features included.

Since we are mainly interested in the artistic part of car design, we focus on the exterior shape instead of the internal mechanical parts. In fact, we remove internal vertices when we construct the car database for a more efficient training process. In the future, we would like to research on the engineering aspect of car design, by considering the necessary space to fit in different mechanical parts such as different models of engines.

One future direction is to look into the gap between synthetic and real-world sketches. We utilize a Laplacian filter to synthesize sketch images. While that is an effective method to generate sketches, we observe that real-world images could be different dependent on the users. For example, some users may not draw straight lines or perfect circles. Also, real-world sketches have variations in pen stroke, paper, and colour. We are interested in transfer learning techniques to bridge the gap between synthetic and real-world sketches. We also look forward to fully evaluate the system by introducing different drawing styles of real-world sketch images.

We use multi-view depth images as an intermediate representation in the VAE network. The two major advantages are that we do not need to deal with 3D deep learning, which is memory hungry and complicated to train, as well as we can have more explicit 2D to 2D correlation in the VAE network. Right now, we combine the depth images as a post-processing step. However, it is possible to consider them as a mean of rectifying the output space, and construct extra layers to learn the regression between multi-view depth images and 3D shapes. One future direction is to explore network architectures for this purpose, and introduce more views of depth images in a middle layer of the network for supervision.

We employ lazy learning in reconstructing the details of the cars. Such an approach is robust and effective, but it requires a run-time search on a k-nearest neighbour, which could be time-consuming if the database is large. In the future, we will consider advanced data structures such as k-d trees or landmark-based k-mean clustering to speed up the searching process.

ACKNOWLEDGEMENTS

This project was supported in part by the Royal Society (Ref: IES\R2\181024 and IES\R1\191147) and the Defence and Security Accelerator (Ref: ACC6007422) and JST ACCEL (JPMJAC1602) and JST-Mirai Program (JPMJMI19B2) and JSPS KAKENHI (JP19H01129).

REFERENCES

Akenine-Möllser, T. (2001). Fast 3d triangle-box overlap testing. *Journal of Graphics Tools*, 6(1):29–33.

Amberg, B., Romdhani, S., and Vetter, T. (2007). Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 187–194, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Chai, J. and Hodgins, J. K. (2005). Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696.

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

Charles, R. Q., Su, H., Kaichun, M., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85.

Chen, Q. and Koltun, V. (2017). Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1520.

Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S. (2016). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *The European conference on computer vision (ECCV)*. Springer.

Corsini, M., Cignoni, P., and Scopigno, R. (2012). Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. on Visualization and Computer Graphics*, 18(6):914–924.

Delanoy, J., Aubry, M., Isola, P., Efros, A., and Bousseau, A. (2018). 3d sketching using multi-view deep volumetric prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(21).

Eitz, M., Hildebrand, K., Boubekeur, T., and Alexa, M. (2011). Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE Trans. on Visualization and Computer Graphics*, 17(11):1624–1636.

Fan, H., Su, H., and Guibas, L. J. (2017). A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613.

Gauthier, M. and Poulin, P. (2009). Preserving sharp edges in geometry images. In *Proceedings of Graphics Interface 2009*, GI '09, pages 1–6, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.

Gingold, Y., Igarashi, T., and Zorin, D. (2009). Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph.*, 28(5):148:1–148:9.

Güler, R. A., Neverova, N., and Kokkinos, I. (2018). Densepose: Dense human pose estimation in the wild. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7297–7306.

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

Han, X., Gao, C., and Yu, Y. (2017). Deepsketch2face: a deep learning based sketching system for 3d face and caricature modeling. *ACM Trans. Graph. (TOG)*, 36(4):126.

Henry, J., Shum, H. P. H., and Komura, T. (2012). Environment-aware real-time crowd control. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 193–200, Aire-la-Ville, Switzerland. Eurographics Association.

Henry, J., Shum, H. P. H., and Komura, T. (2014). Interactive formation control in complex environments. *IEEE Trans. on Visualization and Computer Graphics*, 20(2):211–222.

Ho, E. S. L., Shum, H. P. H., Cheung, Y.-m., and Yuen, P. C. (2013). Topology aware data-driven inverse kinematics. *Computer Graphics Forum*, 32(7):61–70.

Igarashi, T., Igarashi, T., and Hughes, J. F. (2006). Smooth meshes for sketch-based freeform modeling. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA. ACM.

Igarashi, T., Igarashi, T., Matsuoka, S., and Tanaka, H. (2007). Teddy: A sketching interface for 3d freeform design. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA. ACM.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Joshi, P. and Carr, N. A. (2008). Repoussé: Automatic inflation of 2d artwork. In *SBM*, pages 49–55. Citeseer.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589.

Kraevoy, V., Sheffer, A., Shamir, A., and Cohen-Or, D. (2008). Non-homogeneous resizing of complex models. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 111:1–111:9, New York, NY, USA. ACM.

Li, C., Pan, H., Liu, Y., Tong, X., Sheffer, A., and Wang, W. (2018a). Robust flow-guided neural prediction for sketch-based freeform surface modeling. In *SIGGRAPH Asia 2018 Technical Papers*, page 238. ACM.

Li, C., Pan, H., Liu, Y., Tong, X., Sheffer, A., and Wang, W. (2018b). Robust flow-guided neural prediction for sketch-based freeform surface modeling. *ACM Trans. Graph.*, 37(6):238:1–238:12.

Li, H., Sumner, R. W., and Pauly, M. (2008). Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum*, 27(5):1421–1430.

Lun, Z., Gadelha, M., Kalogerakis, E., Maji, S., and Wang, R. (2017). 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision (3DV)*, pages 67–77. IEEE.

Nealen, A., Igarashi, T., Sorkine, O., and Alexa, M. (2007). Fibermesh: Designing freeform surfaces with 3d curves. *ACM Trans. Graph.*, 26(3).

Nishida, G., Garcia-Dorado, I., Aliaga, D. G., Benes, B., and Bousseau, A. (2016). Interactive sketching of urban procedural models. *ACM Trans. Graph. (TOG)*, 35(4):130.

Olsen, L., Samavati, F. F., Sousa, M. C., and Jorge, J. A. (2009). Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85 – 103.

Owada, S., Nielsen, F., Nakazawa, K., Igarashi, T., and Igarashi, T. (2006). A sketching interface for modeling the internal structures of 3d shapes. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA. ACM.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5105–5114, USA. Curran Associates Inc.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4.

Schmidt, R., Khan, A., Singh, K., and Kurtenbach, G. (2009). Analytic drawing of 3d scaffolds. *ACM Trans. Graph. (TOG)*, 28(5):149.

Sela, M., Richardson, E., and Kimmel, R. (2017). Unrestricted facial geometry reconstruction using image-to-image translation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1585–1594.

Shao, C., Bousseau, A., Sheffer, A., and Singh, K. (2012). Crossshade: Shading concept sketches using cross-section curves. *ACM Trans. Graph. (SIGGRAPH Conference Proceedings)*, 31(4).

Shen, Y., Henry, J., Wang, H., Ho, E. S. L., Komura, T., and Shum, H. P. H. (2018). Data-driven crowd motion control with multi-touch gestures. *Computer Graphics Forum*, 37(6):382–394.

Shen, Y., Yang, L., Ho, E. S. L., and Shum, H. P. H. (2019). Interaction-based human activity comparison. *IEEE Trans. on Visualization and Computer Graphics*.

Shtof, A., Agathos, A., Gingold, Y., Shamir, A., and Cohen-Or, D. (2013). Geosemantic snapping for sketch-based modeling. *Computer Graphics Forum*, 32(2pt2):245–253.

Shum, H. P. H., Ho, E. S. L., Jiang, Y., and Takagi, S. (2013). Real-time posture reconstruction for microsoft kinect. *IEEE Trans. on Cybernetics*, 43(5):1357–1369.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2017). Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *The IEEE International Conference on Computer Vision (ICCV)*.

Thorne, M., Burke, D., and van de Panne, M. (2004). Motion doodles: An interface for sketching character motion. *ACM Trans. Graph.*, 23(3):424–431.

Turquin, E., Wither, J., Boissieux, L., Cani, M., and Hughes, J. F. (2007). A sketch-based interface for clothing virtual characters. *IEEE Computer Graphics and Applications*, 27(1):72–81.

Umetani, N. (2017). Exploring generative 3d shapes using autoencoder networks. In *SIGGRAPH Asia 2017 Technical Briefs*, SA '17, pages 24:1–24:4, New York, NY, USA. ACM.

Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X. (2017). O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Trans. Graph.*, 36(4):72:1–72:11.

Wang, T., Liu, M., Zhu, J., Tao, A., Kautz, J., and Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional gans. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807.

Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82.