Learning to Predict 3D Surfaces of Sculptures from Single and Multiple Views

Olivia Wiles[1] · Andrew Zisserman[1]

Received: 28 February 2018 / Accepted: 3 October 2018 / Published online: 22 October 2018

Abstract

The objective of this work is to reconstruct the 3D surfaces of sculptures from one or more images using a view-dependent representation. To this end, we train a network, `SiDeNet`, to predict the **Si**lhouette and **De**pth of the surface given a variable number of images; the silhouette is predicted at a different viewpoint from the inputs (e.g. from the *side*), while the depth is predicted at the viewpoint of the input images. This has three benefits. First, the network learns a representation of shape beyond that of a single viewpoint, as the silhouette forces it to respect the visual hull, and the depth image forces it to predict concavities (which don't appear on the visual hull). Second, as the network learns about 3D using the proxy tasks of predicting depth and silhouette *images*, it is not limited by the resolution of the 3D representation. Finally, using a view-dependent representation (e.g. additionally encoding the viewpoint with the input image) improves the network's generalisability to unseen objects. Additionally, the network is able to handle the input views in a flexible manner. First, it can ingest a different number of views during training and testing, and it is shown that the reconstruction performance improves as additional views are added at test-time. Second, the additional views do not need to be photometrically consistent. The network is trained and evaluated on two synthetic datasets—a realistic sculpture dataset (*SketchFab*), and ShapeNet. The design of the network is validated by comparing to state of the art methods for a set of tasks. It is shown that (i) passing the input viewpoint (i.e. using a view-dependent representation) improves the network's generalisability at test time. (ii) Predicting depth/silhouette images allows for higher quality predictions in 2D, as the network is not limited by the chosen latent 3D representation. (iii) On both datasets the method of combining views in a global manner performs better than a local method. Finally, we show that the trained network generalizes to real images, and probe how the network has encoded the latent 3D shape.

Keywords Visual hull · Generative model · Silhouette prediction · Depth prediction · Convolutional neural networks · Sculpture dataset

1 Introduction

Learning to infer the 3D shape of complex objects given only a few images is one of the grand challenges of computer vision. Another of the many benefits of deep learning has been a resurgence of interest in this task. Many recent works have developed the idea of inferring 3D shape given a set of classes (e.g. cars, chairs, rooms). This modern treatment of class based reconstruction follows on from the pre-deep learning classic work of Blanz and Vetter (1999) for faces and later for other classes such as semantic categories (Kar et al. 2015; Cashman and Fitzgibbon 2013) or cuboidal room structures (Fouhey 2015; Hedau et al. 2009).

This work extends this area in two directions: first, it considers 3D shape inference from *multiple* images rather than a single one (though this is considered as well); second, it considers the quite generic class of piecewise smooth textured sculptures and the associated challenges.

To achieve this, a deep learning architecture is introduced which can take into account a variable number of views in order to predict depth for the given views and the silhouette at a new view (see Fig. 1 for an overview). This approach has a number of benefits: *first* the network learns how to combine the given views—it is an architectural solution—without using multi view stereo. As a result, the views need not be

Communicated by Tae-Kyun Kim, Stefanos Zafeiriou, Ben Glocker and Stefan Leutenegger.

✉ Olivia Wiles
ow@robots.ox.ac.uk

[1] Department of Engineering Science, University of Oxford, Oxford, UK

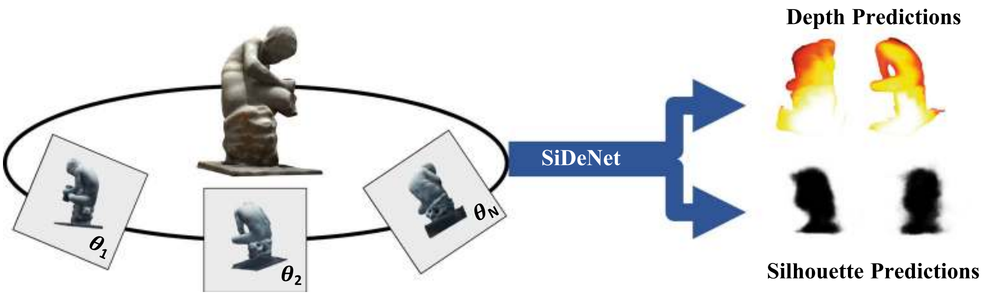

Fig. 1 An overview of SiDeNet. First, images of an object are taken at various viewpoints $\theta_1 \cdots \theta_N$ by rotating the object about the vertical axis. Given a set of these views (the number of which may vary at test time), SiDeNet predicts the depth of the sculpture at the given views and the silhouette at a new view θ'. Here, renderings of the predicted depth at two of the given views and silhouette predictions at new viewpoints are visualised. The depth predictions are rendered using the depth value for the colour (e.g. dark red is further away and yellow/white nearer) (Color figure online)

photometrically consistent. This is useful if the views exhibit changes in exposure/lighting/texture or are taken in different contexts (so one may be damaged), etc. By enforcing that the same network must be able to predict 3D from single and multiple views, the network must be able to infer 3D shape using global information from one view and combine this information given multiple views; this is a different approach from building up depth locally using correspondences as would be done in a traditional multi view stereo approach.

Second, using a view-dependent representation means that the model makes few assumptions about the distribution of input shapes or their orientation. This is especially beneficial if there is no canonical frame or natural orientation over the input objects (e.g. a chair facing front and upright is at 0°). This generalisation power is demonstrated by training/evaluating SiDeNet on a dataset of sculptures which have a wide variety of shapes and textures. SiDeNet generalises to new unseen shapes without requiring any changes.

Finally, as only image representations are used, the quality of the 3D model is not limited by the 3D resolution of a voxel grid or a finite set of points but by the image resolution.

Contributions This work brings the following contributions. *First*, a fully convolutional architecture and loss function, termed *SiDeNet* (Sects. 3, 4) is introduced for understanding 3D shape. It can incorporate additional views at test time, and the predictions improve as additional views are incorporated when both using 2D convolutions to predict depth/silhouettes as well as 3D convolutions to latently infer the 3D shape. Further, this is true without assuming that the objects have a canonical representation unlike many contemporary methods. *Second*, a dataset of complex sculptures which are augmented in 3D (Sect. 5). This dataset demonstrates that the learned 3D representation is sufficient for silhouette prediction as well as new view synthesis for a set of unseen objects with complex shapes and textures. *Third*, a thorough evaluation that demonstrates how incorporating additional views improves results and the benefits of the data augmentation scheme (Sect. 6) as well as that SiDeNet can be used directly on real images. This evaluation also demonstrates how SiDeNet can incorporate multiple views *without* requiring photometric consistency and demonstrates that SiDeNet is competitive or better than comparable state-of-the-art methods for 3D prediction and at leveraging multiple views on both the *Sculptures* and *ShapeNet* datasets. *Finally*, the architecture is investigated to determine how information is encoded and aggregated across views in Sect. 8.[1]

This work is an extension of that described in Wiles and Zisserman (2017). The original architecture is referred to as SilNet, and the improved architecture (the subject of this work) SiDeNet. SilNet learns about the visual hull of the object and is trained on images of a small resolution size to predict the silhouette of the object at again a small resolution size. This is improved in this work, SiDeNet. The loss function is improved by adding an additional term for depth that enforces that the network should learn to predict concavities on the 3D shape (Sect. 3). The architecture is improved by increasing the resolution of the input and predicted image (Sect. 4). The dataset acquisition phase is improved by adding data augmentation in 3D (Sect. 5). These changes are analysed in Sect. 6.

2 Related Work

Inferring 3D shape from one or more images has a long history in computer vision. However, single vs multi-image approaches have largely taken divergent routes. Multi-image approaches typically enforce geometric constraints such that the estimated model satisfies the silhouette and photometric constraints imposed by the given views whereas single image approaches typically impose priors in order to constrain the

[1] Data and resources are available at http://www.robots.ox.ac.uk/~vgg/data/SilNet/.

problem. However, recent deep learning approaches have started to tackle these problems within the same model. This section is divided into three areas: multi-image approaches and single image approaches without deep learning, and newer deep learning approaches which attempt to combine these two problems into one model.

2.1 Multi-image

Traditionally, given multiple images of an object, 3D can be estimated by tracking feature points across multiple views; these constraints are then used to infer the 3D at the feature points using structure-from-motion (SfM), as explained in Hartley and Zisserman (2004). Additional photometric and silhouette constraints can also be imposed on the estimated shape of the object. Silhouette based approaches that attempt to learn the visual hull (introduced by Laurentini 1994) using a set of silhouettes with known camera positions can be done in 3D using voxels (or another 3D representation) or in the image domain by interpolating between views (e.g. the work of Matusik et al. 2000). This is improved by other approaches which attempt to construct the latent shape subject to the silhouette as well as photometric constraints; they differ in how they represent the shape and how they enforce the geometric and photometric constraints (Boyer and Franco 2003; Kolev et al. 2009; Vogiatzis et al. 2003—see Seitz et al. 2006 for a thorough review). The limitation of these approaches is that they require multiple views of the object at test time in order to impose constraints on the generated shape and they cannot extrapolate to unseen portions of the object.

2.2 Single Image

When given a single image, then correspondences cannot be used to derive the 3D shape of the model. As a result, single-image approaches must impose priors in order to recover 3D information. The prior may be based on the class by modelling the deviation from a mean shape. This approach was introduced in the seminal work of Blanz and Vetter (1999). The class based reconstruction approach has continued to be developed for semantic categories (Cashman and Fitzgibbon 2013; Prasad et al. 2010; Vicente et al. 2014; Xiang et al. 2014; Kar et al. 2015; Rock et al. 2015; Kong et al. 2017) or cuboidal room structures (Fouhey 2015; Hedau et al. 2009). Another direction is to use priors on shading, texture, or illumination to infer aspects of 3D shape (Zhang et al. 1999; Blake and Marinos 1990; Barron and Malik 2015; Witkin 1981).

2.3 Deep Learning Approaches

Newer deep learning approaches have traditionally built on the single image philosophy of learning a prior distribution of shapes for a given object class. However, in these cases the distribution is implicitly learned for a specific object class from a single image using a neural network. These methods rely on a large number of images of a given object class that are usually synthetic. The distribution may be learned by predicting the corresponding 3D shape from a given image for a given object class using a voxel, point cloud, or surface representation (Girdhar et al. 2016; Wu et al. 2016; Fan et al. 2016; Sinha et al. 2017; Yan et al. 2016; Tulsiani et al. 2017; Rezende et al. 2016; Wu et al. 2017). These methods differ in whether they are supervised or use a weak-supervision (e.g. the silhouette or photometric consistency as in Yan et al. 2016; Tulsiani et al. 2017). A second set of methods learn a latent representation by attempting to generate new views conditioned on a given view. This approach was introduced in the seminal work of Tatarchenko et al. (2016) and improved on by Zhou et al. (2016), Park et al. (2017).

While demonstrating impressive results, these deep learning methods methods are trained/evaluated on a single or small number of object classes and often do not consider the additional benefits of multiple views. The following approaches consider how to generalise to multiple views and/or the real domain.

The approaches that consider the multi-view case are the following. Choy et al. (2016) use a recurrent neural network on the predicted voxels given a sequence of images to reconstruct the model. Kar et al. (2017) use the known camera position to impose geometric constraints on how the views are combined in the voxel representation. Finally, Soltani et al. (2017) pre-determine a fixed set of viewpoints of the object and then train a network for silhouette/depth from these known viewpoints. However, changing any of the input viewpoints or output viewpoints would require training a new network.

More recent approaches such as the works of Zhu et al. (2017), Wu et al. (2017) have attempted to fine-tune the model trained on synthetic data on real images using the silhouette or another constraint, but they only extend to semantic classes that have been seen in the synthetic data. Novotny et al. (2017) directly learn on real data using 3D reconstructions generated by a SfM pipeline. However, they require many views of the same object and enough correspondences at train time in order to make use of the SfM pipeline.

This paper improves on previous work in three ways. *First* an image based approach is used for predicting the silhouette and depth, thereby enforcing that the latent model learns about 3D shape without having to explicitly model the full 3D shape. *Second* our method of combining multiple views using a latent embedding acts globally as opposed to locally (e.g. Choy et al. 2016 combine information for subsets of voxels and Kar et al. 2017 combine information along projection rays). Additionally, our method does not require photometric consistency or geometric modelling of the camera movement and intrinsic parameters—it is an architectural solution. In spirit, our method of combining multiple views is more similar to multi-view classification/recognition architectures such as the works of Su et al. (2015), Qi et al. (2016). *Third* a new *Sculptures* dataset is curated from SketchFab (2018) which exhibits a wide variety of shapes from many semantic classes. Many contemporary methods train/test on ShapeNet core which contains a set of semantic classes. Training on class-specific datasets raises the question: to what extent have these architectures actually learnt about shape and how well will they generalise to unseen objects that vary widely from the given class (e.g. as an extreme how accurately would these models reconstruct a tree when trained on beds/bookcases). We investigate this on the *Sculptures* dataset.

3 Silhouette and Depth: A Multi-task Loss

The loss function used enforces two principles: first that the network learns about the visual hull, and second that it learns to predict the surface (and thus also concavities) at the given view. This is done by predicting, for a given image (or set of images), the silhouette in a new view and the depth at the given views. We expand on these two points in the following.

3.1 Silhouette

The first task considered is how to predict the silhouette at a new view given a set of views of an object. The network can do well at this task only if it has learned about the 3D shape of the object. To predict the silhouette at a new angle θ', the network must at least encode the visual hull (the visual hull is the volume swept out by the intersection of the back-projected silhouettes of an object as the viewpoint varies). Using a silhouette image has desirable properties: first, it is a 2D representation and so is limited by the 2D image size (e.g. as opposed to the size of a 3D voxel grid). Second, pixel intensities do not have to be modelled.

3.2 Depth

However, using the silhouette and thereby enforcing the visual hull has the limitation that the network is not forced to predict concavities on the object, as they never appear on the visual hull. The proposed solution to this is to use a multi-task approach. Instead of having the learned representation describe only the silhouette in the new view, the representation must learn additionally to predict the depth of the object in the given views. This enforces that the representation must have a richer understanding of the object, as it must model the concavities on the object as opposed to just the visual hull (which using a silhouette loss imposes). Using a depth image is also a 2D representation, so as with using an image for the silhouette, it is limited by the 2D image size.

4 Implementation

In order to actually implement the proposed approach, the problem is formulated as described in Sects. 4.1 and 4.2 and a fully convolutional CNN architecture is used, as described in Sect. 4.3.

4.1 Loss Function

The loss function is implemented as follows. Given a set of images with their corresponding viewpoints $(I_1, \theta_1), \ldots, (I_N, \theta_N)$ a representation x is learned such that x can be used to not only predict the depth in the given views $d_1, \ldots, d_N$ but also predict the silhouette S at a new viewpoint θ'. Moreover, the number of input views (e.g. N) should be changeable at test time such that as N increases then the predictions $d_1, \ldots d_N, S$ improve.

To do this, the images and their corresponding viewpoints are first encoded using a convolutional encoder f to give a latent representation fv_i. The same encoder is used for all viewpoints giving $f(I_1, \theta_1), \ldots, f(I_N, \theta_N) = fv_1, \ldots, fv_N$. These are then combined to give the latent view-dependent representation x. x is then decoded using a convolutional decoder h_{sil} conditioned on the new viewpoint θ' to predict the silhouette S in the new view. Optionally, x is also decoded via another convolutional decoder h_{depth}, which is conditioned on the given image and viewpoints to predict the depth at the given viewpoints—$d_i = h_{depth}(x, I_i, \theta_i)$. Finally, the binary cross entropy loss is used to compare S to the ground truth S^{gt} and the L_1 loss to compare d_i to the ground truth $d_{i_{gt}}$.

4.2 Improved Loss Functions

Implementing the loss functions naively as described in Sect. 4.1 is problematic. First, the depth being predicted is the absolute depth, which means the model must guess the absolute position of the object in the scene. This is inherently ambiguous. Second, the silhouette prediction decoder struggles to model the finer detail on the silhouette, instead focusing on the middle of the object which is usually filled.

As a result, both losses are modified. For the depth prediction, the mean of both the ground truth and predicted depth are moved to 0.

The silhouette loss is weighted at a given pixel $w_{i,j}$ based on the Euclidean distance at that point to the silhouette (denoted as $\text{dist}_{i,j}$):

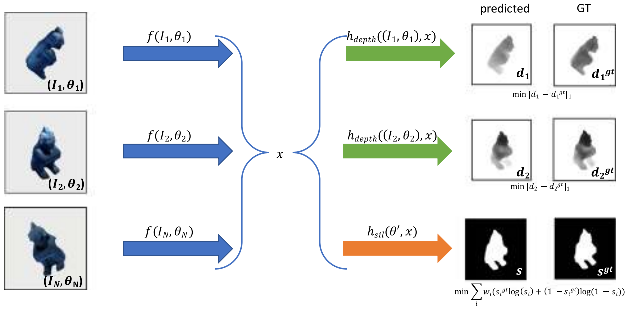

Fig. 2 A diagrammatic explanation of the multi-task loss function used. Given the input images, the images are combined to give a feature vector x which is used by both decoders (denoted in green—depth—and orange—silhouette) to generate the depth predictions for the given views and the silhouette prediction in a new view (Color figure online)

$$w_{i,j} = \begin{cases} \text{dist}_{i,j}, & \text{if dist}_{i,j} \leq T \\ c & \text{otherwise.} \end{cases} \tag{1}$$

In practice $T = 20, c = 5$. The rationale for the fall-off when $\text{dist}_{i,j} > T$ is due to the fact that most of the objects are centred and have few holes, so modelling the pixels far from the silhouette is easy. Using the fall-off incentivises SiDeNet to correctly model the pixels near the silhouette. Weighting based on the distance to the silhouette models the fact that it is ambiguous whether pixels on the silhouette are part of the background or foreground.

In summary, the complete loss functions are

$$\mathcal{L}_{sil} = \sum_{i,j} w_{i,j} \left(S^{gt}_{i,j} \log(S_{i,j}) + \left(1 - S^{gt}_{i,j}\right) \log(1 - S_{i,j}) \right); \tag{2}$$

$$\mathcal{L}_{depth} = \sum_{i=1}^{N} |d_i - d_{i_{gt}}|_1. \tag{3}$$

The loss function is visualised in Fig. 2. Note that in this example the network's prediction exhibits a concavity in the groove of the sculpture's folded arms.

4.3 Architecture

This section describes the various components of SiDeNet, which are visualised in Fig. 3 and described in detail in Table 10. This architecture takes as input a set of images of size 256 × 256 and corresponding viewpoints (encoded as $[\sin\theta_i, \cos\theta_i]$ so that 0°, 360° map to the same value) and generates depth and silhouette images at a resolution of size 256 × 256. SiDeNet takes the input image viewpoints as additional inputs because there is no implicit coordinate frame that is true for all objects. For example, a bust may be oriented along the z-axis for one object and the x-axis for another and there is no natural mapping from a bust to a sword. Explicitly modelling the coordinate frame using the input/output viewpoints removes these ambiguities.

SiDeNet is modified to produce a latent 3D representation in SideNet3D, which is visualised in Fig. 4 and described in Sect. 4.4. This architecture is useful for two reasons. First, it demonstrates that the method of combining multiple views is useful in this scenario as well. Second, it is used to evaluate whether the image representation does indeed allow for more accurate predictions, as the 3D representation necessitates using fewer convolutional transposes and so generates a smaller 57 × 57 silhouette image.

Encoder The encoder f takes the given image I_i and theta θ_i and encodes it to a latent representation $f v_i$. In the case of all architectures, this is implemented using a convolutional encoder, which is illustrated in Fig. 3. The layer parameters and design are based on the encoder portion of the pix2pix architecture by Isola et al. (2017) which is based on the UNet architecture of Ronneberger et al. (2015).

Combination function To combine the feature vectors of each encoder, any function that satisfies the following property could be considered: given a set of feature vectors $f v_i$, the combination function should combine them into a single latent vector x such that for any number of feature vectors, x always has the same number of elements. In particular, an element-wise max pool over the feature vectors and an element-wise average pool are considered. This vector x must encode properties of 3D shape useful for both depth prediction and silhouette prediction in a new view.

Decoder (depth) The depth branch predicts the depth of a given image using skip connections (taken from the corresponding input branch) to propagate the higher details. The exact filter sizes are modelled on the pix2pix and UNet networks.

Decoder (silhouette) The silhouette branch predicts the silhouette of a given image at a new viewpoint θ'. The layers are the same as the decoder (depth) branch without the skip connections (as there is no corresponding input view).

4.4 3D Decoder

For SiDeNet3D, the silhouette decoder is modified to generate a latent 3D representation encoded using a voxel occupancy grid. Using a projection layer this grid is projected to 2D, which allows the silhouette loss to be used to train the network in an end-end manner. This is done as follows. First, the decoder is encoded as a sequence of 3D convolutional transposes which generate a voxel of size $V = 57 \times 57 \times 57$ (please refer to appendix A.1 for the precise details). This box is then transformed to the desired output θ' to give V' using a nearest neighbour sampler as described by Jaderberg et al. (2015). The box is projected to generate the silhouette

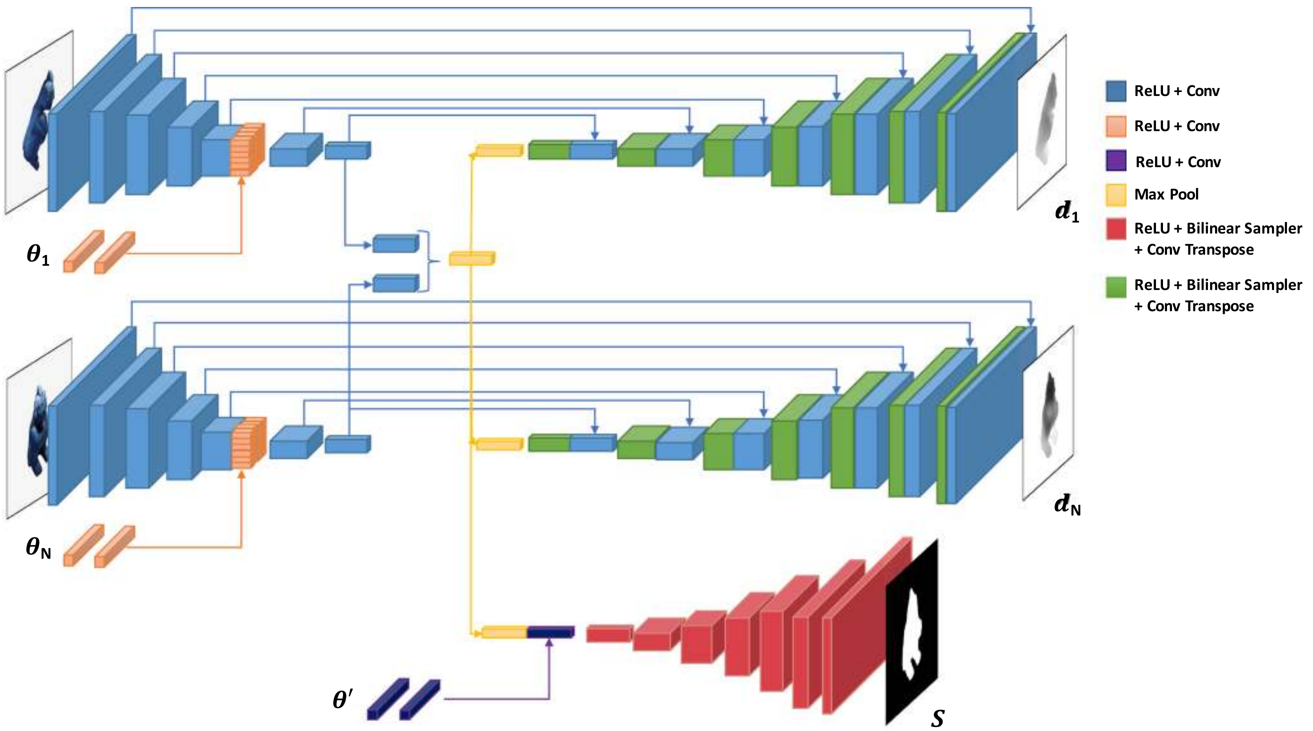

Fig. 3 A diagrammatic overview of the architecture used in SiDeNet. Weights are shared across encoders and decoders (e.g. portions of the architecture having the same colour indicate shared weights). The blue, orange, and purple arrows denote concatenation. The input angles $\theta_1 \cdots \theta_N$ are broadcast over the feature channels as illustrated by the orange arrows. The feature vectors are combined to form x (indicated by the yellow block and arrows). This value is then used to predict the depth at the given views $\theta_1 \cdots \theta_N$ and the silhouette at a new view θ'. The size of x is invariant to the number of input views N, so an extra view θ_i can be added at test time without any increase in the number of parameters. Please see Table 10 for the precise details (Color figure online)

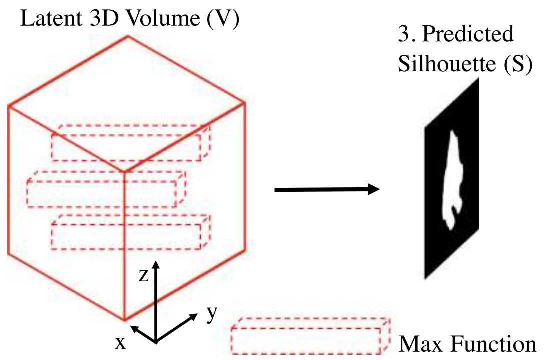

Fig. 4 A diagrammatic overview of the projection in SiDeNet3D. A set of 3D convolutional transposes up-sample from the combined feature vector x to generate the $57 \times 57 \times 57$ voxel (V). This is then projected using a max-operation over each pixel location to generate the silhouette in a new view. Please see Table 10 for a thorough description of the three different architectures

Table 1 Overview of the datasets. Gives the number of sculptures in the train/val/test set as well as the number of views per object

Dataset	Train	Val	Test	# of views
SketchFab	372	20	33	5
SynthSculptures	77	–	–	5
ShapeNet	4744	678	1356	24

in a new view using the max function. As the max function is differentiable, the silhouette loss can be back propagated through this layer and the entire network trained end-to-end.

The idea of using a differentiable projection layer was also considered by Yan et al. (2016), Tulsiani et al. (2017), Gadelha et al. (2016), Rezende et al. (2016). (However, we can incorporate additional views *at test time*.)

5 Dataset

Three datasets are used in this work: a large sculpture dataset of scanned objects which is downloaded from SketchFab (2018), a set of scanned sculptures, and a subset of the synthetic ShapeNet objects (Chang et al. 2015). An overview of the datasets are given in Table 1. Note that unlike our dataset, ShapeNet consists of object categories for which one can impose a canonical view (e.g. that 0° corresponds to a chair facing the viewer). This allows for methods trained on this

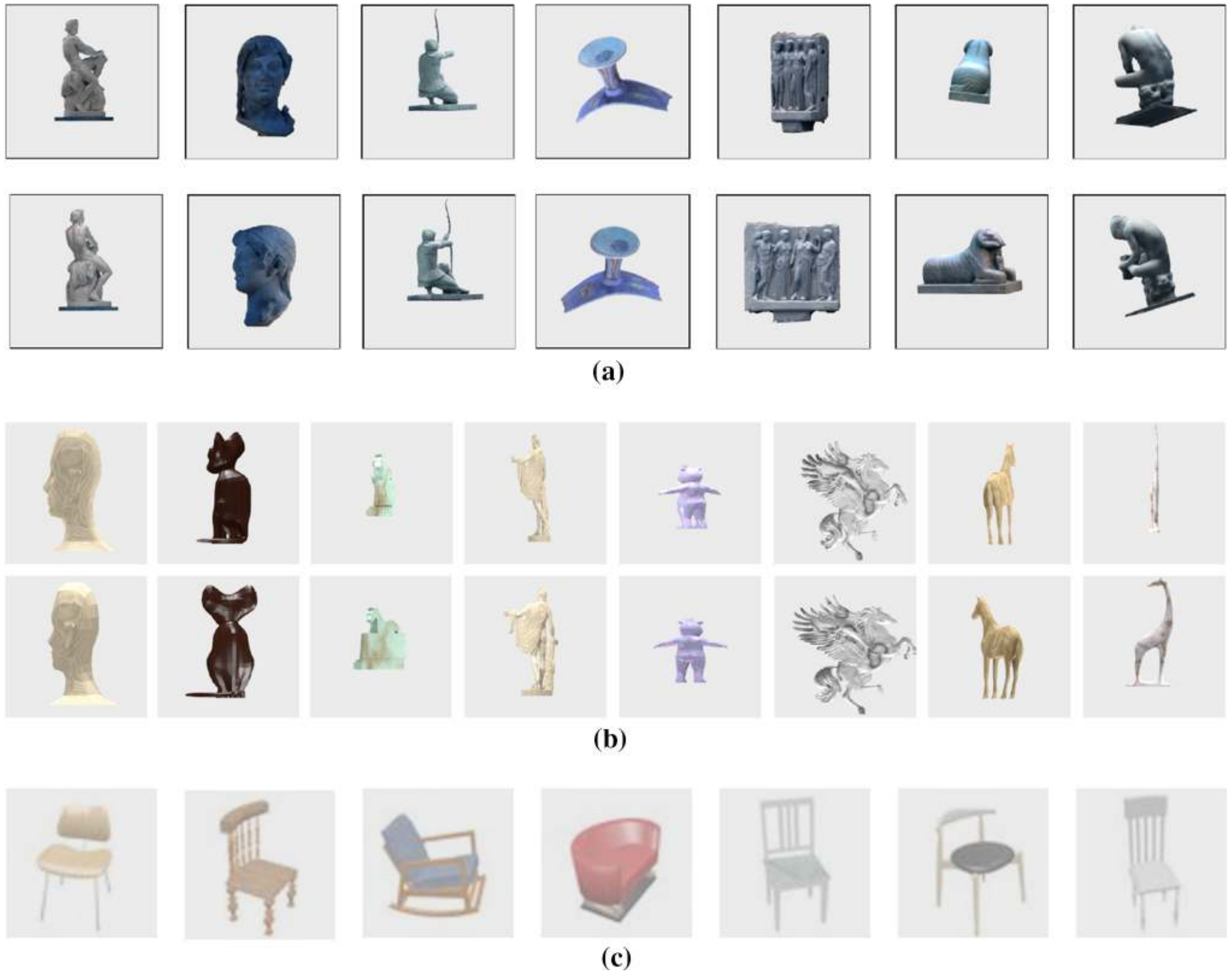

Fig. 5 Sample renderings of the three different datasets. Zoom in for more details. Best viewed in colour. **a** *SketchFab* dataset. Two sample renderings of seven objects. The first three fall into the train set, the rest into the test set. **b** *SynthSculpture* dataset. Sample renderings of eight objects. These samples demonstrate the variety of objects, e.g. toys, animals, etc. **c** *ShapeNet*. Seven sample renderings of the chair subset (Color figure online)

dataset to make use of rotations or transformations relative to the canonical view. However, for the sculpture dataset, this property does not exist, necessitating the need of a view-dependent representation for SiDeNet.

Performing data augmentation in 3D is also investigated and shown to increase performance in Sect. 6.2.

5.1 Sculpture Datasets

SketchFab: sculptures from SketchFab A set of realistic sculptures are downloaded from SketchFab (the same sculptures as used in Wiles and Zisserman 2017 but different renderings). These are accurate reconstructions of the original sculptures generated by users using photogrammetry and come with realistic textures. Some examples are given in Fig. 5a.

SynthSculptures This dataset includes an additional set of 77 sculptures downloaded from TurboSquid[2] using the query *sculpture*. These objects have a variety of realism and come from a range of object classes. For example the sculptures range from low quality meshes that are clearly polygonized to high quality, highly realistic meshes. The object classes range from abstract sculptures to jewellery to animals. Some examples are given in Fig. 5b.

Rendering The sculptures and their associated material (if it exists) are rendered in Blender (Blender Online Community 2017). The sculptures are first resized to be within a uniform range (this is necessary for the depth prediction component of the model). Then, for each sculpture, five images

[2] https://www.turbosquid.com/3d-model/.

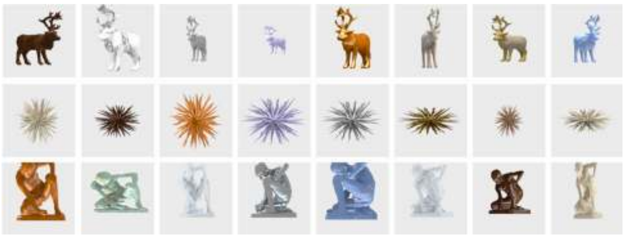

Fig. 6 Seven sample augmentations of three models in the *SynthSculpture* dataset using the 3D augmentation setup described in Sect. 5.1. These samples demonstrate the variety of materials, sizes and viewpoints for a given 3D model using the 3D data augmentation method

of the sculpture are rendered from uniformly randomly chosen viewpoints between 0° and 120° as the object is rotated about the vertical axis. Three light sources are added to the scene and translated randomly with each render. Some sample sculptures (and renders) for *SketchFab* and *SynthSculptures* are given in Fig. 5.

3D augmentation 3D data augmentation is used to augment the two sculpture datasets by modifying the dimensions and material of a given 3D model. The x,y,z dimensions of a model are each randomly scaled from between [0.5, 1.4] of the original dimension. Then a material is randomly chosen from a set of standard blender materials.[3] These materials include varieties of wood, stone, and marble. Finally, the resulting model is rendered from five viewpoints exactly as described above. The whole process is repeated 20 times for each model. Some example renderings using data augmentation for a selection of models from *SynthScultpures* are illustrated in Fig. 6.

Dataset split The sculptures from *SketchFab* are divided at the sculpture level into train, val, test so that there are 372/20/33 sculptures respectively. All sculptures from *SynthSculptures* are used for training. For a given iteration during train/val/test, a sculpture is randomly chosen from which a subset of the 5 rendered views is selected.

5.2 ShapeNet

ShapeNet (Chang et al. 2015) is a dataset of synthetic objects divided into a set of semantic classes. To compare this work to that of Yan et al. (2016), their subdivision, train/val/test split and renderings of the ShapeNet chair subset are used. Their rendered synthetic objects are rendered under simple lighting conditions at fixed 15° intervals about the vertical axis for each object to give a total of 24 views per object. We additionally collect depth maps for each render using the extrinsic/intrinsic parameters of Yan et al. (2016). Some example renderings are given in Fig. 5c. Again at train/val/test time, a sculpture is randomly chosen and a subset of this sculpture's 24 renders is chosen.

6 Experiments

This section first evaluates the design choices: the utility of using the data augmentation scheme is demonstrated in Sect. 6.2, the effect of the different architectures in Sect. 6.3, the multi-task loss in Sect. 6.4, and the effect of the choice of θ' in Sect. 6.8. Second it evaluates the method of combining multiple views: Sect.s 6.5 and 6.6 demonstrate how increasing the number of views at test time improves performance on the *Sculpture* dataset irrespective of whether the input/output views are photometrically consistent. Section 6.7 demonstrates that the approach works on ShapeNet and Sect. 6.9 evaluates the approach in 3D. SiDeNet's ability to perform new view synthesis is exhibited in Sect. 7 as well as its generalisation capability to real images. Finally, the method by which SiDeNet can encode a joint embedding of shape and viewpoint is investigated in Sect. 8.

6.1 Training Setup

The networks are written in pytorch (Paszke et al. 2017) and trained with SGD with a learning rate of 0.001, momentum of 0.9 and a batch size of 16. They are trained until the loss on the validation set stops improving or for a maximum of 200 iterations, whichever happens first. The tradeoff between the two losses – $\mathcal{L} = \lambda_{depth}\mathcal{L}_{depth} + \lambda_{sil}\mathcal{L}_{sil}$—is set such that $\lambda_{depth} = 1$ and $\lambda_{sil} = 1$.

6.1.1 Evaluation Measure

The evaluation measure used is the intersection over union (IoU) error for the silhouette, L_1 error for the depth error, and chamfer distance for the error when evaluating in 3D. The IoU for a given predicted silhouette S and ground truth silhouette $\bar{S}$ is evaluated as $\frac{\sum_{x,y}(I(S) \cap I(\bar{S}))}{\sum_{x,y}(I(S) \cup I(\bar{S}))}$ where I is an indicator function and equals 1 if the pixel is a foreground pixel, else 0. This is then averaged over all images to give the mean IoU.

The L_1 loss is simply the average over all foreground pixels: $L_1 = \frac{1}{N}\sum_{p_x} |d^{pred}_{p_x} - d^{gt}_{p_x}|_1$ where p_x is a foreground pixel and N the number of foreground pixels. Note that the predicted and ground truth depth are first normalised by subtracting off the mean depth. This is then averaged over the batch. When there are multiple input views, the depth error is only computed for the first view, so the comparison across increasing numbers of views is valid.

[3] https://www.blendswap.com/blends/view/4867.

Table 2 Effect of data augmentation. This table demonstrates the utility of using 3D data augmentation to effectively enlarge the number of sculptures being trained with. *SketchFab* is always used and sometimes augmented (denoted by *Augment*). *SynthSculpture* is sometimes used (denoted by *Used*) and sometimes augmented. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU

SketchFab	*SynthSculpture*		L_1 Depth error	Silhouette IoU
Augment?	Used?	Augment?		
✗	✗	–	0.210	0.643
✓	✗	–	0.202	0.719
✗	✓	✗	0.209	0.678
✓	✓	✓	0.201	0.724

The chamfer distance used is the symmetrized version. Given the ground truth point cloud g and the predicted one p, then the error is $CD = \frac{1}{N}\sum_{i=1}^{N}\min_j |g_i - p_j|^2 + \frac{1}{M}\sum_{i=1}^{M}\min_j |g_j - p_i|^2$.

6.1.2 Evaluation Setup

Unless otherwise stated, the results are for the max-pooling version of SiDeNet, with input/output view size 256 × 256, trained with 2 distinct views, data augmentation of both datasets (Sect. 6.2), $\lambda_{depth} = 1$ and $\lambda_{sil} = 1$, and the improved losses described in Sect. 4.2.

6.2 The Effect of the Data Augmentation

First, the effect of the 3D data augmentation scheme is considered. The results for four methods trained with varying amounts of data augmentation (described in section 5.1) are reported in Table 2 and demonstrate the benefit of using the 3D data augmentation scheme. (These are trained with the non-improved losses.) Using only 2D modifications was tried but not found to improve performance.

6.3 Ablation Study of the Different Architectures

This section compares the performance of SiDeNet$_{57\times57}$, SiDeNet3D, and SiDeNet on the silhouette/depth prediction tasks, as well as using average vs max-pooling. SiDeNet/SiDeNet3D are described in Sect. 4.3. SiDeNet$_{57\times57}$ modifies SiDeNet to generate a 57 × 57 silhouette (for the details for all architectures please refer to "Appendix A.1"). It additionally compares the simple version of the loss functions, described in Sect. 4.1 to the improved version described in Sect. 4.2. Finally the performance of predicting the mean depth value is given as a baseline. See Table 3 for the results.

These results demonstrate that while the difference in the pooling function in terms of results is minimal, our improved loss functions improve performance. Weighting more strongly the more difficult parts of the silhouette (e.g. around the boundary) can encourage the model to learn a better representation.

Finally, SiDeNet$_{57\times57}$ does worse than SiDeNet for both the L_1 loss and the silhouette IoU loss. While in this case the difference is small, as more data is introduced and the predictions become more and more accurate, the benefit of using a larger image/representation is clear. This is demonstrated by the chairs on ShapeNet in Sect. 6.7.

6.4 The effect of using $\mathcal{L}_{depth}$ and $\mathcal{L}_{sil}$

Second, the effect of the individual components of the multi-task loss is considered. The multi-task loss enforces that the network learns a richer 3D representation; the network must predict concavities in order to perform well at predicting depth and it must learn about the visual hull of the object in order to predict silhouettes at new viewpoints. As demonstrated in Table 4, using the multi-task loss does not negatively affect the prediction accuracy as compared to predicting each component separately. This demonstrates that the model is able to represent both aspects of shape at the same time.

Table 3 Ablation study of the different architectures, which vary in size and complexity. $_{basic}$ refers to using the standard L_1 and binary cross entropy loss without the improvements described in Sect. 4.2. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU. The sizes denote the size of the corresponding images (e.g. 256 × 256 corresponds to an output image of this resolution)

Model	Input size	Output size	Pooling?	Improved loss?	Depth $L_{1_{256\times256}}$ error	Silhouette IoU$_{256\times256}$
SiDeNet$_{\text{basic}}$	256 × 256	256 × 256	Max	✗	0.201	0.724
SiDeNet	256 × 256	256 × 256	Max	✓	0.181	0.739
SiDeNet	256 × 256	256 × 256	Avg	✓	0.189	0.734
SiDeNet$_{57\times57_{\text{basic}}}$	256 × 256	57 × 57	Max	✗	–	0.723
SiDeNet$_{57\times57}$	256 × 256	57 × 57	Max	✓	0.195	0.734
SiDeNet3D	256 × 256	57 × 57	Max	✓	0.182	0.733
Baseline: $z = c$	–	–	–	–	0.223	–

Table 4 Effect of the multi-task loss. This table demonstrates the effect of the multi-task loss. As can be seen, using both losses does not negatively affect the performance of either task. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU

Loss function	λ_{depth}	λ_{sil}	Depth L_1 error	Silhouette IoU
Silhouette and depth	1	1	0.181	0.739
Silhouette	–	–	–	0.734
Depth	–	–	0.178	–

Table 5 Effect of incorporating additional views at test time. This architecture was trained with one, two, or three views. These results demonstrate how additional views can be dynamically incorporated at test time and results on both depth and silhouette measures improve. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU

Pooling?	# Views (train)	# Views (test)	L_1 Depth error	Silhouette IoU
Max	1	1	0.206	0.702
Max	1	2	0.210	0.712
Max	1	3	0.209	0.716
Max	2	1	0.204	0.694
Max	2	2	0.181	0.739
Max	2	3	0.170	0.751
Avg	2	1	0.197	0.715
Avg	2	2	0.192	0.725
Avg	2	3	0.189	0.732
Max	3	1	0.198	0.706
Max	3	2	0.172	0.753
Max	3	3	0.162	0.766

Some visual results are given in Figs. 11 and 12. Example (b) in Fig. 12 demonstrates how the model has learned to predict concavities, as it is able to predict grooves in the relief.

6.5 The effect of increasing the number of views

Next, the effect of increasing the number of input views is investigated with interesting results.

For SiDeNet, as with SilNet, increasing the number of views improves results over all error metrics in Table 5. Some qualitative results are given in Fig. 7. It is interesting to note that not only does the silhouette performance improve given additional input views but so does the depth evaluation metric. So incorporating additional views improves the depth prediction for a given view using only the latent vector x.

A second interesting point is that training with more views can predict better than training with fewer numbers of views—e.g. training with three views and testing on one or two views does better than training on two views and testing on two or training on one view and testing on one view. It seems that when training with additional views and testing with a smaller number, the network can make use of information learned from the additional views. This demonstrates the generalisability of the SiDeNet architecture.

6.6 The Effect of Non-photometrically Consistent Inputs

A major benefit of SiDeNet is it does not require photometrically consistent views: provided the object is of the same shape, then the views may vary in lighting or material. While the sculpture renderings used already vary in lighting conditions across different views (Sect. 5), this section considers the extreme case: how does SiDeNet perform when the texture is modified in the input views. To perform this comparison, SiDeNet is tested on the sculpture dataset with a randomly chosen texture for each view (see Fig. 6 for some sample textures demonstrating the variety of the 20 textures). It is then tested again on the same test set but with the texture fixed across all input views. The results are reported in Table 6.

Surprisingly, with no additional training, SiDeNet performs nearly as well when the input/output views have randomly chosen textures. Moreover, performance improves given additional views. The network appears to have learned to combine input views with varying textures without being explicitly trained for this. This demonstrates a real benefit of SiDeNet over traditional approaches—the ability to combine multiple views of an object for shape prediction *without* requiring photometric consistency.

6.7 Comparison on ShapeNet

SiDeNet is compared to Perspective Transformer Nets by Yan et al. (2016) by training and testing on the chair subset of the ShapeNet dataset. The comparison demonstrates three benefits of our approach: the ability to incorporate multiple views, the benefit of our 3D data augmentation scheme, and the benefits of staying in 2D. This is done by comparing the accuracy of SiDeNet's predicted silhouettes to those of Yan et al. (2016). Their model is trained with the intention of using it for 3D shape prediction, but we focus on the 2D case here to demonstrate that using an image representation means that, with the same data, we can achieve better prediction performance in the image domain, as we are not limited by the latent voxel resolution. To compare the generated silhouettes, their implementation of the IoU metric is used: $\frac{\sum_{x,y} I(S_{x,y}) \times \bar{S}_{x,y}}{\sum_{x,y} (I(S_{x,y}) + \bar{S}_{x,y}) > 0.9}$.

Multiple setups for SiDeNet are considered: fine-tuning from the model trained on the sculptures with data augmen-

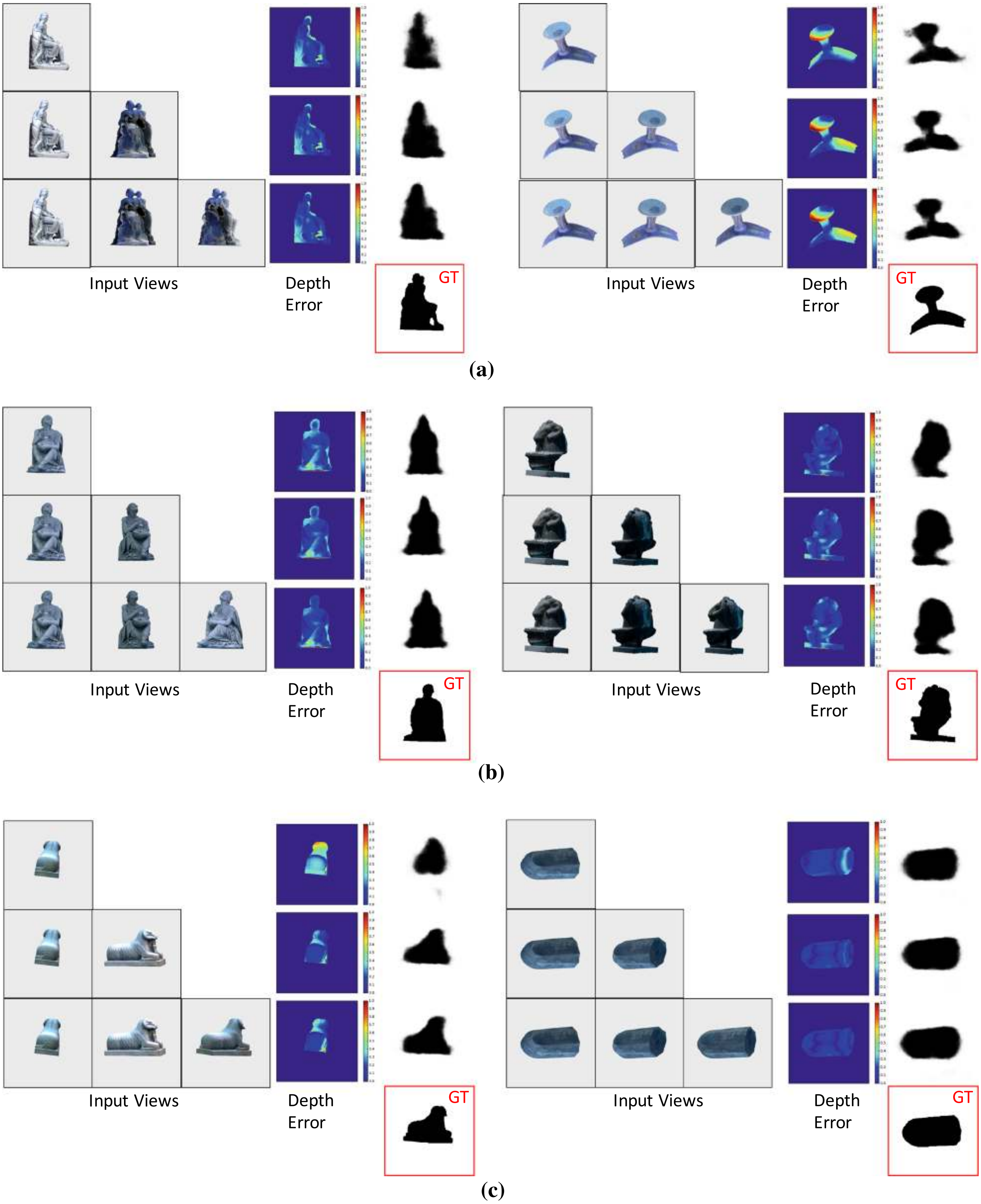

Fig. 7 (**a–c**) Qualitative results for increasing the number of input views on SiDeNet for three different sculptures. SiDeNet's depth and silhouette predictions are visualised as the number of input views is increased. To the left are the input views, the centre gives the depth prediction for the first input view, and the right gives the predicted silhouette for each set of input views. The silhouette in the red box gives the ground truth silhouette. The scale on the side gives the error in depth—blue means the depth prediction is perfectly accurate and red that the prediction is off by 1 unit. (The depth error is clamped between 0 and 1 for visualisation purposes.) As can be seen, performance improves with additional views. This is most clearly seen for the ram in (**c**) (Color figure online)

Table 6 The effect of using non-photometrically consistent inputs. These results demonstrate that SiDeNet trained with views of an object with the same texture generalises *at runtime* to incorporating views of an object with differing textures. Additional views can be dynamically incorporated at test time and results on both depth and silhouette measures improve. The model is trained with 2 views. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU

Views have the same texture?	# Views (test)	L_1 Depth error	Silhouette IoU
✓	1	0.165	0.739
✓	2	0.142	0.778
✓	3	0.139	0.785
✗	1	0.164	0.738
✗	2	0.143	0.777
✗	3	0.139	0.785

Table 7 Comparison to Perspective Transformer Nets (PTNs) (Yan et al. 2016) on the silhouette prediction task on the chair subset of ShapeNet. Their model is first trained on multiple ShapeNet categories and fine-tuned on the chair subset. SiDeNet is optionally first trained on the Sculpture dataset or trained directly on the chair subset. As can be seen, SiDeNet outperforms PTN given one view and improves further given additional views. These results also demonstrate the utility of various components of SiDeNet: using a larger 256 × 256 image to train the silhouette prediction task and using the improved, weighted loss function. It is also interesting to note that pre-training with the complex sculpture class gives a small boost in performance (e.g. it generalises to this very different domain of chairs). The value reported is the mean IoU metric for the silhouette; higher is better

	Pre-training	Number of views tested with				
		1	2	3	4	5
Yan et al. (2016)	ShapeNet	0.797	–	–	–	–
SiDeNet	Sculptures	0.831	0.845	0.850	0.852	0.853
SiDeNet	–	0.826	0.843	0.848	0.850	0.851
$\text{SiDeNet}_{256\times256_{\text{basic}}}$	–	0.814	0.831	0.835	0.837	0.837
$\text{SiDeNet}_{57\times57_{\text{basic}}}$	–	0.775	0.791	0.795	0.796	0.795

tation (e.g. both in Table 1), with/without the improved loss function and for multiple output sizes. To demonstrate the benefits of the SiDeNet architecture, SiDeNet is trained only with the silhouette loss, so both models are trained with the exact same information. The model from Yan et al. (2016) is fine-tuned from a model trained for multiple ShapeNet categories. The results are reported in Table 7.

These results demonstrate the benefits of various components of SiDeNet, which outperforms Yan et al. (2016). *First*, using a 2D resolution means a much larger image segmentation can be used to train the network. As a result, much better performance can be obtained (e.g. $\text{SiDeNet}_{256\times256_{\text{basic}}}$ has much better performance than $\text{SiDeNet}_{57\times57_{\text{basic}}}$). *Second*, the improved, weighted loss function for the silhouette (Sect. 4.2) improves performance further. *Third*, fine-tuning a model trained with the 3D sculpture augmentation scheme gives an additional small boost in performance. *Finally*, using additional views improves results for all versions of SiDeNet. Some qualitative results are given in Fig. 10.

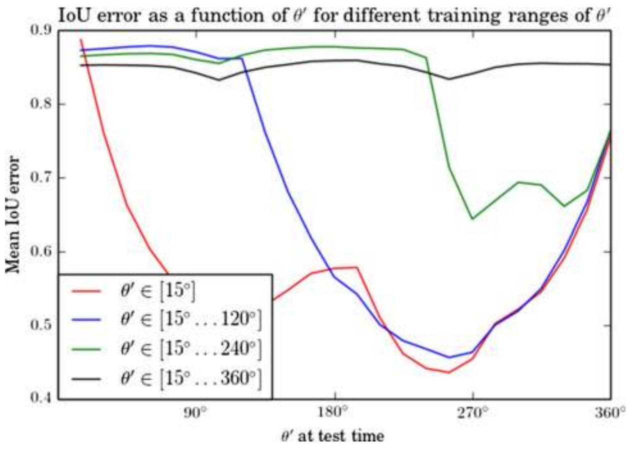

Fig. 8 The effect of varying the range of θ' used at train time on the IoU error at test time (Color figure online)

6.8 The Effect of Varying θ'

In order to see how well SiDeNet can extrapolate to new angles (and there by how much it has learned about the visual hull), the following experiment is performed on ShapeNet. SiDeNet is first trained with various ranges of θ', θ_i. For example if the range is $[15° \cdots 120°]$, then all randomly selected input angles θ_i and θ' are constrained to be within this range during training. At test time, a random chair is chosen and the silhouette IoU error evaluated for each target viewpoint θ' in the full range (e.g. $[15° \cdots 360°]$), but the input angles θ_i are still constrained to be in the constrained range (e.g. $[15° \cdots 120°]$). This evaluates how well the model extrapolates to unseen viewpoints at test time and how well it has learned about shape. If the model was perfect, then there would be no performance degradation as θ' moved out of the constrained range used to train the model. The results are given in Fig. 8. As can be seen (and would be expected), for various training ranges the performance degrades as a function of how much θ' differs from the range used to train the model. The model is able to extrapolate outside of the training range, but the more the model must extrapolate, the worse the prediction.

6.9 Comparison in 3D

We additionally evaluate SiDeNet's 3D predictions and consider the two cases: using the depth maps predicted by SiDeNet and the voxels from SiDeNet3D.

SiDeNet The depth maps are compared to those predicted using the depth map version of Kar et al. (2017) in Table 8.

Table 8 CD (×100) on the *ShapeNet* dataset. The models evaluated on depth predict a depth map which is back-projected to generate a 3D point cloud

Model	Trained with:	Evaluation is on:	Number of views tested with				
			1	2	3	4	6
SiDeNet	Silhouettes + depth	Depth	1.47	0.72	0.62	0.59	0.58
Kar et al. (2017)	Depth	Depth	1.73	0.82	0.71	0.67	0.65

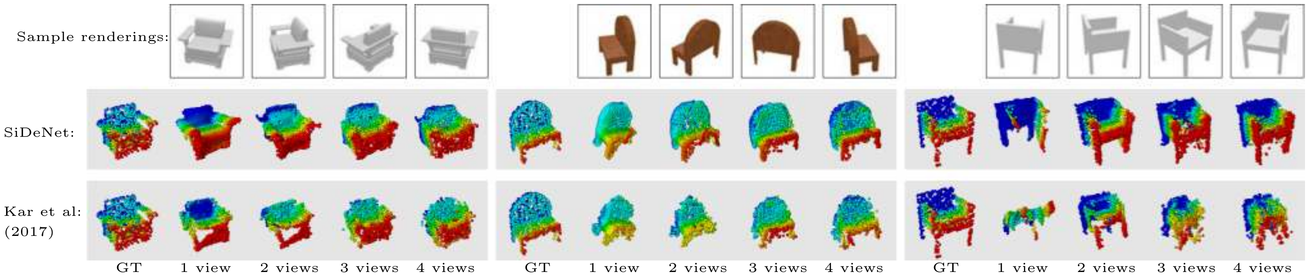

Fig. 9 Comparison of multi-view methods on ShapeNet. Renderings of the given chair are given in the top row, followed by SiDeNet's and Kar et al. (2017)'s predictions. For each chair, for each row, the point clouds from left to right show the ground truth followed by the predictions for one, two, three, and four views respectively. The colour denotes the z value. As can be seen SiDeNet's predictions are higher quality than those of Kar et al. (2017) for these examples

Table 9 CD (×100) on the *Sculptures* dataset. The models evaluated on depth predict a depth map which is back-projected to generate a 3D point cloud. The models evaluated on 3D are compared using the explicitly or implicitly learned 3D

Model	Trained with	Evaluation is on:	Number of views tested with		
			1	2	3
SiDeNet3D	Silhouettes + depth	3D	0.87	0.82	0.81
Kar et al. (2017)	Depth	Depth	2.15	1.38	1.15
Tatarchenko et al. (2016)	Depth	Depth	1.97	–	–
Yan et al. (2016)	Silhouettes	3D	1.26	–	–
Groueix et al. (2018)	3D	3D	1.23	–	–

This comparison is only done on ShapeNet as for the *Sculpture* dataset we found it was necessary to subtract off the mean depth to predict high quality depth maps (Sect. 4.2). However, for ShapeNet there is less variation between the chairs so this is not necessary. As a result SiDeNet is trained with 2 views, the improved silhouette loss but the depth predicted is the absolute depth. The comparison is performed as follows for both methods. For each chair in the test set an initial view is chosen and the depth back-projected using the known extrinsic/intrinsic camera parameters. Then for each additional view, the initial views are chosen by sampling evenly around the z-axis (e.g. if the first view is at 15°, then two views would be at 15°, 195° and three views at 15°, 195°, 255°) and the depth again back-projected to give a point cloud. 2500 points are randomly chosen from the predicted point cloud and aligned using ICP (Besl and McKay 1992) with the ground truth point cloud. This experiment evaluates the method of pooling information in the two methods and demonstrates that SiDeNet's global method of combining information performs better than that of Kar et al. (2017) which combines information along projection rays. Some qualitative results are given in Fig. 9.

SiDeNet3D SiDeNet3D is trained with 2 views and the improved losses. The predicted voxels from the 3D projection layer are extracted and marching cubes used to fit a mesh over the iso-surface. The threshold value is chosen on the validation set. A point cloud is extracted by randomly sampling from the resulting mesh.

SiDeNet3D is compared to a number of other methods in Table 9 for the *Sculpture* dataset. For SiDeNet3D and all baselines models, 2500 points are randomly chosen from the predicted point cloud and aligned with the ground truth point cloud using ICP. The resulting point cloud is compared to the ground truth point cloud by reporting the chamfer distance (CD). As can be seen, the performance of our method improves as the number of input views increases.

Additionally, SiDeNet3D performs better than other baseline methods on the *Sculpture* dataset in Table 9 which demonstrates the utility of explicitly encoding the input viewpoint and thereby representing the coordinate frame of the object. We note again that there is no canonical coordinate frame and the input viewpoint does not align with the output shape, so just predicting the 3D without allowing the network to learn the transformation from the input viewpoint to the 3D (as done in all the baseline methods) leads to poor performance.

Baselines The baseline methods which do not produce point clouds are converted as follows. To convert Yan et al. (2016) to a point cloud, marching cubes is used to fit a mesh over the predicted voxels. Points are then randomly chosen from the extracted mesh. To convert Tatarchenko et al. (2016) to a point cloud, the model is used to predict depth maps at [0°, 90°, 180,° , 270°]. The known intrinsic/extrinsic camera parameters are used to back-project the depth maps. The four point clouds are then combined to form a single point cloud.

7 Generating new views

Finally SiDeNet's representation can be qualitatively evaluated by performing two tasks that require new view generation: rotation and new view synthesis.

7.1 Rotation

As SiDeNet is trained with a subset of views for each dataset (e.g. only 5 views of an object from a random set of viewpoints in [0°, 120°] for the *Sculpture* dataset and 24 views taken at 15° intervals for ShapeNet), the angle representation can be probed by asking SiDeNet to predict the silhouette as the angle is continuously varied within the given range of viewpoints. Given a fixed input, if the angle is varied continuously, then the output should similarly vary continuously. This is demonstrated in Fig. 10 for both the *Sculpture* and ShapeNet databases.

7.2 New view synthesis

Using the predicted depth, new viewpoints can be synthesised, as demonstrated in Fig. 11. This is done by rendering the depth map of the object using Open3D (Zhou et al. (2018)) as a point cloud at the given viewpoint and at a 45° rotation. At both viewpoints the object is rendered in three ways: using a textured point cloud, relighting the textured point cloud, and rendering the point cloud using the predicted z value.

7.3 Real Images

Finally, the generalisability of what SiDeNet has learned is tested on another dataset of real images of sculptures, curated by Zollhöfer et al. (2015). The images of two sculptures (augustus and relief) are taken. The images are segmented and padded such that the resulting images have the same properties as the *Sculpture* dataset (e.g. distance of sculpture to the boundary and background colour). The image is then input to the network with viewpoint 0°. The resulting prediction is rendered as in Sect. 7.2 at multiple viewpoints and under multiple lighting conditions in Fig. 12. This figure demonstrates that SiDeNet generalises to real images, even though SiDeNet is trained only on synthetic images and for a comparatively small (only $\approx$ 400) sculptures. Moreover these real images have perspective effects, yet SiDeNet generalises to these images, producing realistic predictions.

8 Explainability

This section delves into SiDeNet, attempting to understand how the network learns to incorporate multiple views. To this end, the network is investigated using two methods. The first considers how well the original input images can be reconstructed given the angles and feature encoding x. The second considers how well the original input viewpoints θ_i can be predicted as a function of the embedding x and what this implies about the encoding. This is done for both the max and average pooling architectures.

8.1 Reconstruction

The first investigation demonstrates that the original input images can be relatively well reconstructed given only the feature encoding x and the input views. These reconstructions in Fig. 13 demonstrate that x must hold some viewpoint and image information.

To reconstruct the images, the approach of Mahendran and Vedaldi (2015) is followed. Two images and their corresponding viewpoints, are input to the network and a forward pass computed. Then the combined feature vector x is extracted (so it contains the information from the input views and their viewpoints). The two images are reconstructed, starting from noise, by minimizing a cost function consisting of two losses: the first loss, the $\mathcal{L}_{MSE}$ error, simply says that the two reconstructed images when input to the network, should give a feature vector x' that is the same as x. The second loss, the total variation regulariser $\mathcal{L}_{TV}$ (as in Mahendran and Vedaldi 2015 and Upchurch et al. 2017), states that the reconstructed images should be smooth.

$$\mathcal{L}_{MSE} = \sum_i (x_i - x_i')^2 \tag{4}$$

$$\mathcal{L}_{TV} = \sum_{i,j} \left((I_{i,j+1} - I_{i,j})^2 + (I_{i+1,j} - I_{i,j})^2 \right)^{\beta/2} \tag{5}$$

This gives the total loss $\mathcal{L} = \mathcal{L}_{MSE} + \lambda_{TV} * \mathcal{L}_{TV}$. Here, β, λ_{TV} are chosen such that $\beta = 2$ and $\lambda_{TV} = 0.001$. The

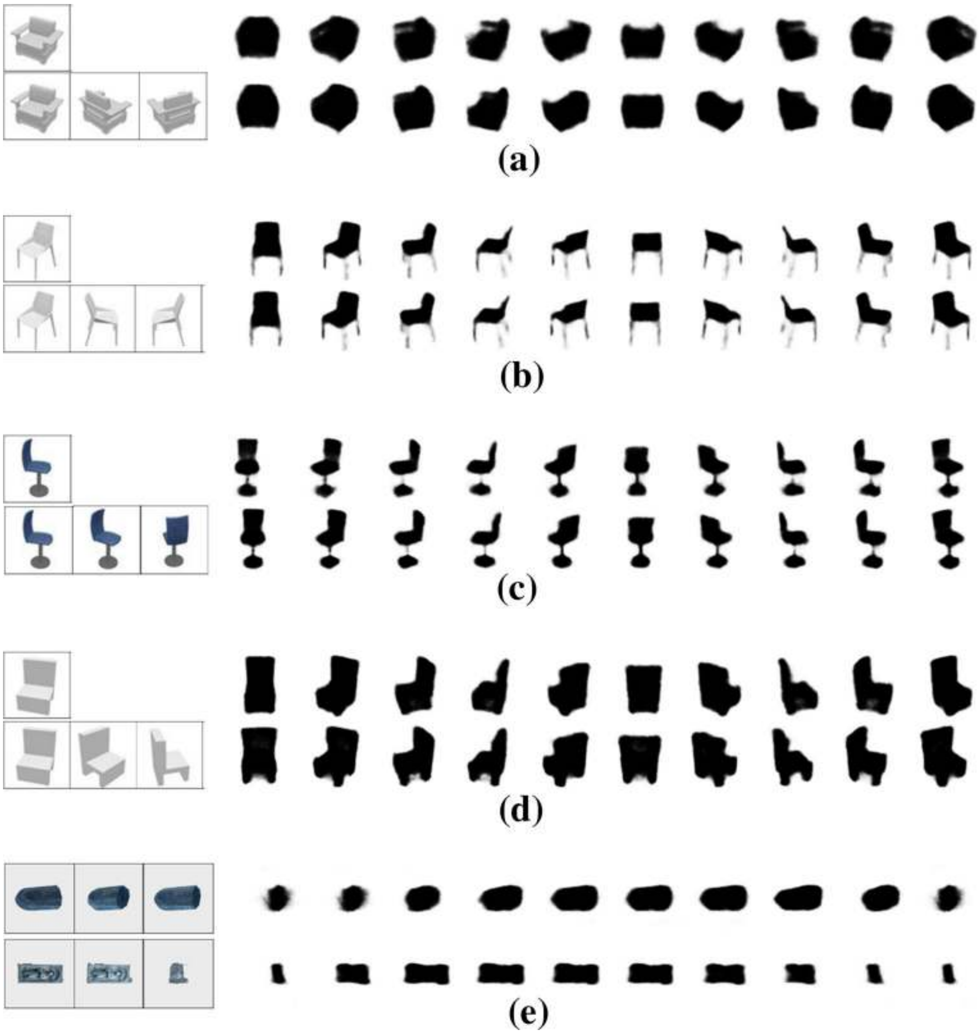

Fig. 10 Qualitative results for rotating an object using the angle embedding of θ'. As the angle θ' is rotated from [0°, 360°] while the input images and viewpoints are kept fixed, it can be seen that the objects rotate continuously for ShapeNet (**a**–**d**) and the *Sculpture* database (**e**). Additionally, the results for ShapeNet improve given additional input views. For example, in (**d**), the base of the chair is incorrectly predicted as solid given one view but correctly predicted given additional views

cost function is optimized using SGD (with momentum 0.975 and learning rate 1, which is decreased by a factor of 0.1 at each 1000 steps).

8.2 Analysis of feature embeddings

In the reconstructions above, it seems that some viewpoint information is propagated through the network, despite the aggregation function. Here, we want to understand precisely how this is done. In order to do so, the following experiment is conducted: how well can the various viewpoints (e.g. $\theta_1 \cdots \theta_N$) be predicted for a given architecture from the embedding x. If the hypothesis—that the embedding x encodes viewpoint—is correct, then these viewpoints should be accurately predicted.

As a result, x is considered to determine how much of it is viewpoint-independent and how much of it is viewpoint-dependent. This is done by using each hidden unit in x to predict the viewpoint θ_1 using ordinary least squares regression (Friedman et al. 2001) (only θ_1 is considered as x is invariant to the input ordering). Training pairs are obtained by taking two images with corresponding viewpoints θ_1 and θ_2, passing them through the network and obtaining x.

The p value for each hidden unit is computed to determine whether there is a significant relation between the hidden unit and the viewpoint. If the p-value is insignificant (i.e. it is large, > 0.05) then this implies that the hidden unit and viewpoint are not related, so it contains viewpoint-independent information (presumably shape information). The number of hidden units with p value less than c, as c is varied, is

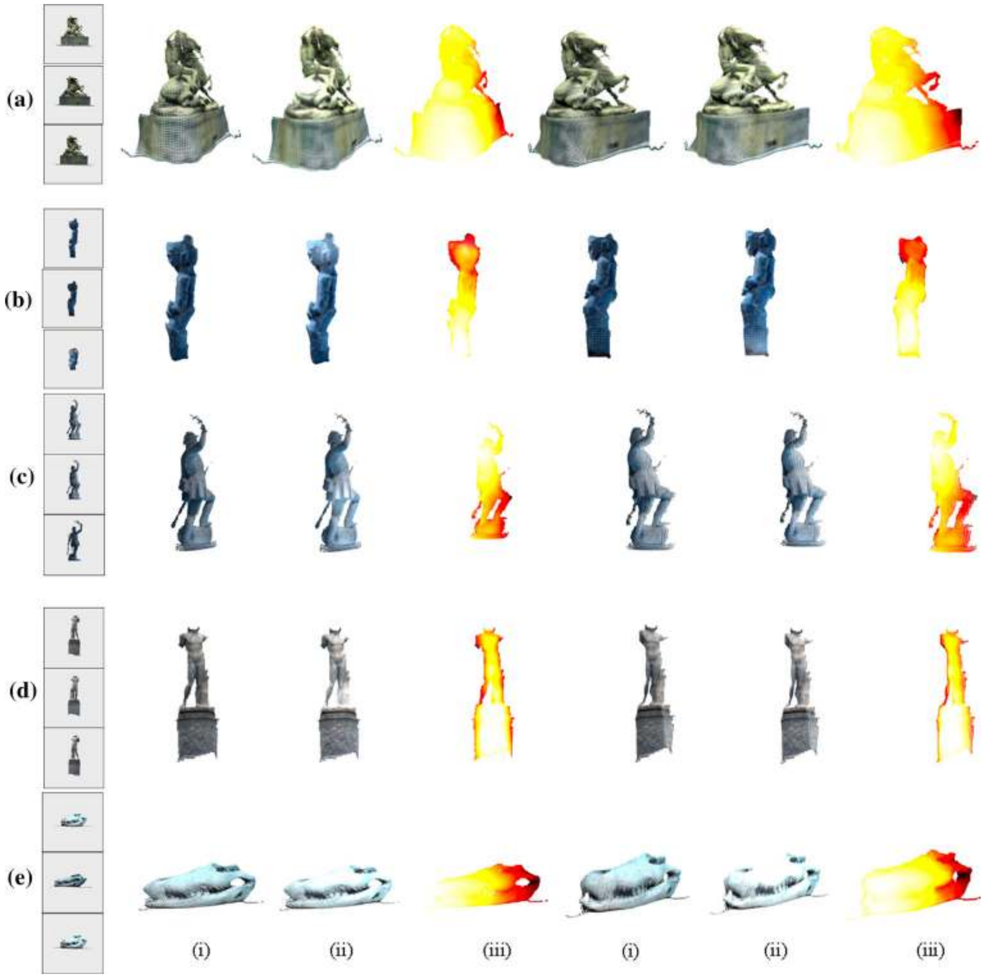

Fig. 11 This figure demonstrates how new views of a sculpture can be synthesised. For each sculpture the input views are shown to the left. The sculpture is then rendered at two viewpoints. At each viewpoint, three renderings are shown: (i) the rendered, textured point cloud, (ii) the point cloud relit and (iii) the depth cloud rendered by using the z-value for the colour (e.g. dark red is further away and yellow/white nearer). Zoom in for details (Color figure online)

visualised in Fig. 14 for both architectures. As can be seen, more than 80% of the hidden units for both architectures are significantly related to the viewpoint.

Since so many of the hidden units have a significant relation to the viewpoint, they would be expected to vary as a function of the input angle. To investigate this, the activations of the hidden units are visualised as a function of the angle θ_1. For two objects, all input values are kept fixed (e.g. the images and other viewpoint values) except for θ_1 which is varied between 0° and 360°. A subset of the hidden units in x are visualised as θ_1 is varied in Fig. 15. As can be seen, the activation either varies in a seemingly sinusoidal fashion—it is maximised at some value for θ_1 and decays as θ_1 is varied—or it is constant.

Moreover, the activations are not the same if the input images are varied. This implies that the hidden units encode not just viewpoint but also viewpoint-dependent information (e.g. shape—such as the object is tall and thin at 90°). This information is aggregated over all views with either aggregation method. The aggregation method controls whether the most 'confident' view (e.g. if using max) is chosen or all views are considered (e.g. avg). Finally, this analysis demonstrates the utility of encoding the input viewpoints in the architecture. When generating the silhouette and depth at a new or given viewpoint, these properties can be easily morphed into the new view (e.g. if the new viewpoint is at 90° then components nearer 90° can be easily considered with more weight by the model).

8.3 Discussion

In this section, to understand what two versions of SiDeNet—avg and max—have learned, two questions have been posed. How well can the original input images be reconstructed

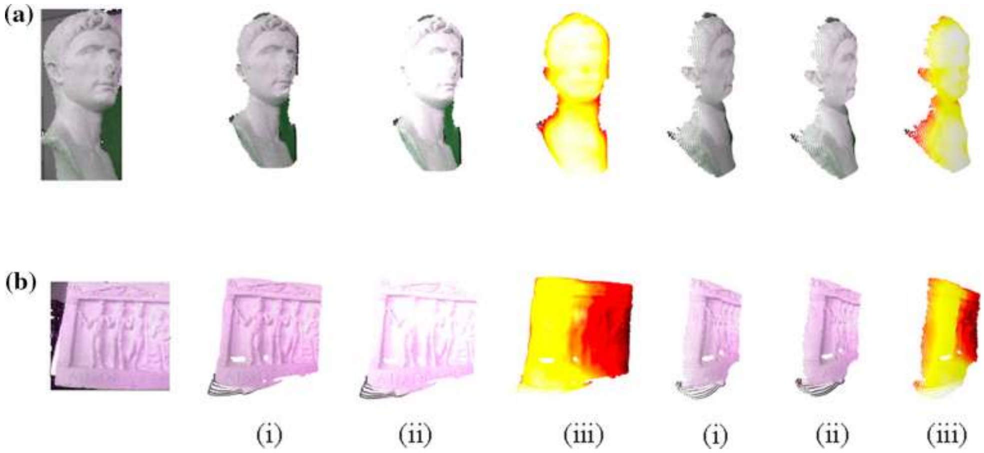

Fig. 12 SiDeNet's predictions for real images. This figure demonstrates how SiDeNet generalises to real images. For each sculpture the input view (before padding and segmentation) is shown to the left. The predicted point cloud is then rendered at two viewpoints. At each viewpoint, three renderings are shown: (i) the rendered, textured point cloud, (ii) the point cloud relit and (iii) the depth cloud rendered by using the z-value for the colour (e.g. dark red is further away and yellow/white nearer). Zoom in for details (Color figure online)

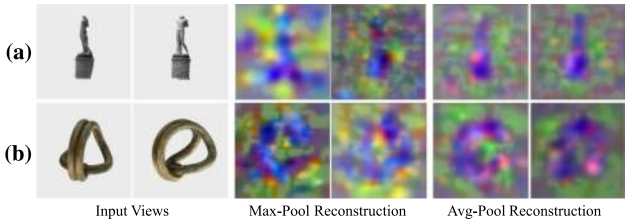

Fig. 13 Reconstruction of the original input images for max/avg pooling architectures. The ability to propagate view and viewpoint information through the network is demonstrated by the fact that the input images can be reconstructed given the latent feature vector and input angles using the approach of Mahendran and Vedaldi (2015)

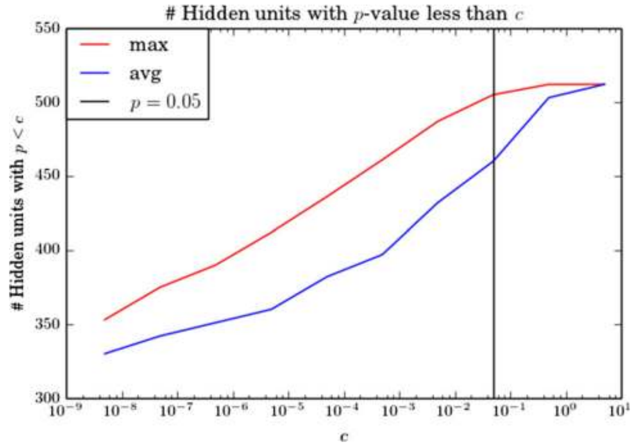

Fig. 14 Visualises the relation between the individual hidden units and the viewpoint. Each hidden unit is used in a separate regression to predict the viewpoint. The p value for each hidden unit is computed and for a given set of values c, the number of hidden units with a p value $< c$ is plotted. This demonstrates that the majority of hidden units in both architectures are correlated with the viewpoint. For the max architecture, 98% of the hidden units have $p < 0.05$ and for the avg pool architecture 90%

from the angles and latent vector x? How is x encoded such that views can be aggregated and that with more views, performance improves? The subsequent analysis has not only demonstrated that the original input views can be reconstructed given the viewpoints and x but has also put forward an explanation for how the views are aggregated: by using the hidden units to encode shape and viewpoint together.

9 Summary

This work has introduced a new architecture SiDeNet for learning about 3D shape, which is tested on a challenging dataset of 3D sculptures with a high variety of shapes and textures. To do this a multi-task loss is used; the network learns to predict the depth for the given views and the silhouette at a new view. This loss has multiple benefits. First, it enforces that the network learns a complex representation of shape, as predicting the silhouette enforces that the network learns about the visual hull of the object and predicting the depth that the network learns about concavities on the object's surface. Second, using an image-based representation is beneficial, as it does not limit the resolution of the generated model; this benefit is demonstrated on the ShapeNet dataset. The trained network can then be used for various applications, such as new view synthesis and can even be used directly on real images.

The second benefit of the SiDeNet architecture is the view-dependent representation and the ability to generalise over additional views at test-time. Using a view-dependent representation means that no implicit assumptions need to be made about the nature of the 3D objects (e.g. that there exists a canonical orientation). Additionally, SiDeNet can leverage

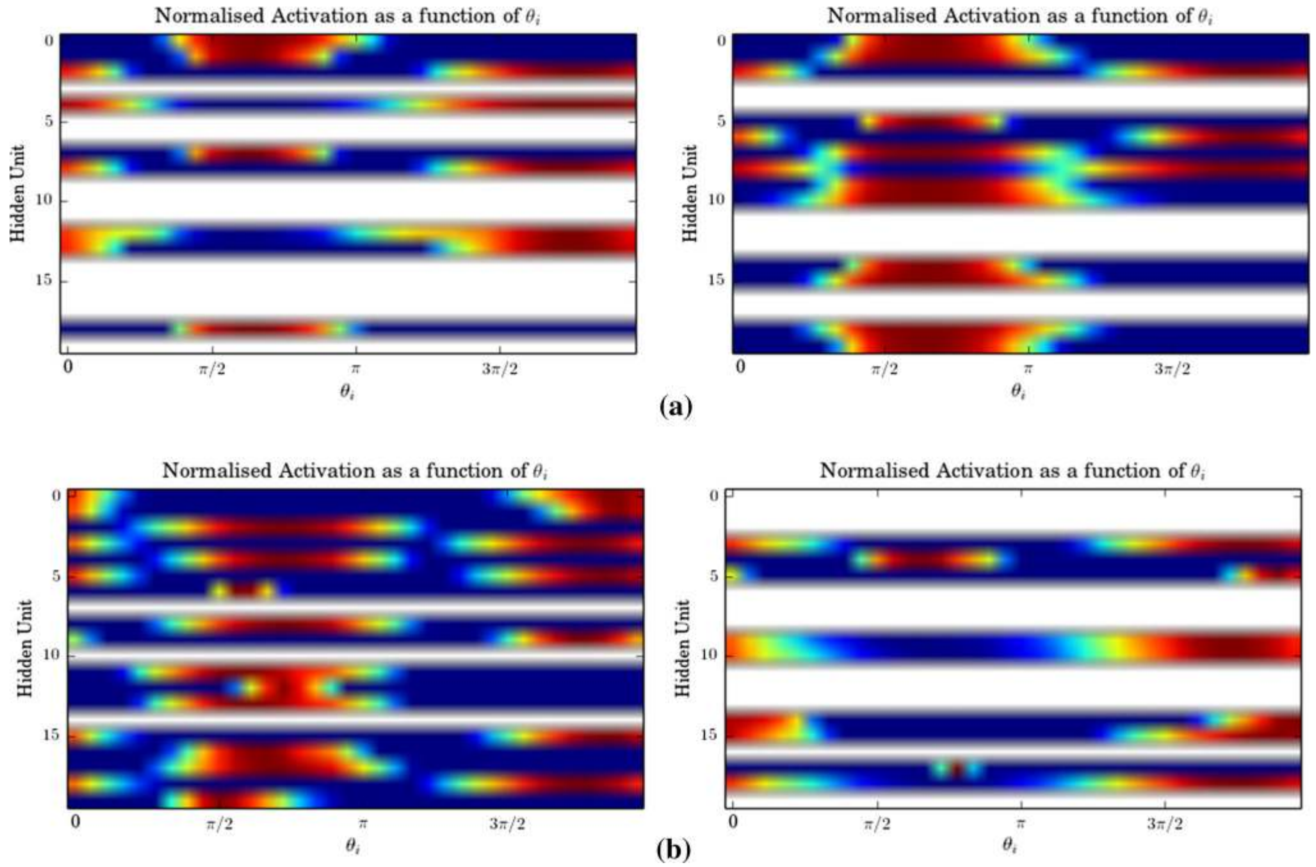

Fig. 15 Visualisation of the activation of hidden units as a function of θ_i for the two architectures. θ_i is varied between 0°, 360° and all other values kept constant. Each hidden unit is normalised to between 0 and 1 over this sequence of θ_i and visualised. This figure demonstrates two things: that the activation is a continuous, smooth function of θ_i or constant (visualised as white in the figure). Second, it demonstrates that the hidden units activated are based on the input views, as they vary from view to view. This implies that the hidden units encode viewpoint dependent information (e.g. object properties and the associated viewpoint). **a** The activation for a subset of hidden units for the avg-pooling architecture for two different sets of input images (left and right). **b** The activation for a subset of hidden units for the max-pooling architecture for two different sets of input images (left and right)

additional views at test time and results (both silhouette and depth) improve with each additional view, *even* when the views are not photometrically consistent.

While the architecture is able to capture a wide variety of shapes and styles as demonstrated in our results, it is most likely that SiDeNet would improve given more data. However, despite the sculpture dataset being small compared to standard deep learning datasets, it is interesting that SiDeNet can be used to boost performance on a very different synthetic dataset of chairs and predict depth, out-of-the-box, on real sculpture images.

Acknowledgements Thank you to Andrew Fitzgibbon and the anonymous reviewers for their useful comments. This work was funded by an EPSRC studentship and EPSRC Programme Grant Seebibyte EP/M013774/1.

A Additional Architectural Details

A.1 2D Architecture

Table 10 gives additional information about the 2D architectures used. There are two variations. The first takes a 256 × 256 architecture and generates silhouette and depth images of size 256 × 256. The second stays in 2D and modifies the silhouette decoder to generate a smaller silhouette of size 57 × 57.

Table 10 Overview of the different architectures. The colours correspond to Fig. 3. The part in orange corresponds to the angle encoding and the part in blue the image encoding. These are then concatenated at layer **6** by broadcasting the angle encoding across the spatial dimensions of the image tensor to which it is supposed to be concatenated. Layer type *Conv* denotes convolution followed by an *Leaky ReLU (0.2)* layer. Layer type *Upsamp* denotes a sequence of layers: *ReLU*, *Bilinear 2x2 Upsampler*, *Conv*, *BatchNorm*. Layer type *ConvTB* denotes the sequence: *Conv Transpose*, *ReLU*, and *BatchNorm*. Finally, layer type *ConvT* denotes the sequence: *Conv Transpose* and *ReLU*

Layer	Type	Stride / Kernel Size / Padding	Prev. Layer	Img. Size (pre layer)	Img. Size (post layer)
Encoder					
1	Conv	2/4/1	I_i	$3 \times 256 \times 256$	$64 \times 128 \times 128$
2	Conv	2/4/1	1	$64 \times 128 \times 128$	$128 \times 64 \times 64$
3	Conv	2/4/1	2	$128 \times 64 \times 64$	$256 \times 32 \times 32$
4	Conv	2/4/1	3	$256 \times 32 \times 32$	$512 \times 16 \times 16$
5	Conv	2/4/1	4	$512 \times 16 \times 16$	$512 \times 8 \times 8$
A	Conv	1/1	θ_i	$2 \times 1 \times 1$	$32 \times 1 \times 1$
B	Conv	1/1	A	$32 \times 1 \times 1$	$32 \times 1 \times 1$
6	Concat	–	5/B	–	$544 \times 8 \times 8$
7	Conv	2/4/1	6	$544 \times 8 \times 8$	$512 \times 4 \times 4$
8	Conv	2/4/1	7	$512 \times 4 \times 4$	$512 \times 2 \times 2$
9	Conv	2/4/1	8	$512 \times 2 \times 2$	$512 \times 1 \times 1$
Decoder (depth)					
10	UpSamp	1/3/1	x	$512 \times 1 \times 1$	$512 \times 2 \times 2$
11	UpSamp	1/3/1	10/8	$1024 \times 2 \times 2$	$512 \times 4 \times 4$
12	UpSamp	1/3/1	11/7	$1024 \times 4 \times 4$	$512 \times 8 \times 8$
13	UpSamp	1/3/1	12/5	$1024 \times 8 \times 8$	$512 \times 16 \times 16$
14	UpSamp	1/3/1	13/4	$1024 \times 16 \times 16$	$256 \times 32 \times 32$
15	UpSamp	1/3/1	14/3	$512 \times 32 \times 32$	$128 \times 64 \times 64$
16	UpSamp	1/3/1	15/2	$256 \times 64 \times 64$	$64 \times 128 \times 128$
17	UpSamp	1/3/1	16/1	$128 \times 128 \times 128$	$3 \times 256 \times 256$
18	Tanh (×5)	–	17	$3 \times 256 \times 256$	$3 \times 256 \times 256$
Decoder (silhouette) 256 × 256					
C	Conv	1/1	θ_i	$2 \times 1 \times 1$	$32 \times 1 \times 1$
D	Conv	1/1	A	$32 \times 1 \times 1$	$32 \times 1 \times 1$
19	Concat	–	D/x	–	$544 \times 1 \times 1$
20	ConvTB	2/4/1	19	$544 \times 1 \times 1$	$256 \times 4 \times 4$
21	ConvTB	2/4/1	20	$256 \times 4 \times 4$	$128 \times 8 \times 8$
22	ConvTB	2/4/1	21	$128 \times 8 \times 8$	$128 \times 16 \times 16$
23	ConvTB	2/4/1	22	$128 \times 16 \times 16$	$64 \times 32 \times 32$
24	ConvTB	2/4/1	23	$64 \times 32 \times 32$	$64 \times 64 \times 64$
25	ConvTB	2/4/1	24	$64 \times 64 \times 64$	$32 \times 128 \times 128$
26	ConvTB	2/4/1	25	$32 \times 128 \times 128$	$1 \times 256 \times 256$
27	Sigmoid	–	26	$1 \times 256 \times 256$	$1 \times 256 \times 256$
Decoder (silhouette) 57 × 57					
C	Conv	1/1	θ_i	$2 \times 1 \times 1$	$32 \times 1 \times 1$
D	Conv	1/1	A	$32 \times 1 \times 1$	$32 \times 1 \times 1$
19	Concat	–	D/x	–	$544 \times 1 \times 1$
20	ConvT	2/4/1	19	$544 \times 1 \times 1$	$512 \times 4 \times 4$
21	ConvT	2/4/1	20	$512 \times 4 \times 4$	$256 \times 8 \times 8$
22	ConvT	2/5/1	21	$256 \times 8 \times 8$	$128 \times 16 \times 16$
23	ConvT	2/5/1	22	$128 \times 16 \times 16$	$64 \times 32 \times 32$
24	ConvT	2/6/1	23	$64 \times 32 \times 32$	$1 \times 57 \times 57$
25	Sigmoid	–	24	$1 \times 57 \times 57$	$1 \times 57 \times 57$

A.2 3D Decoder

The third architecture modifies the silhouette decoder to generate a latent 3D representation which projects to a silhouette of size 57 × 57 (the encoder is the same as for the 2D architectures). The 3D decoder is composed of the following set of 3D convolutional transposes and ReLU units. ConvT3D(256,3,2) → ReLU → ConvT3D(128,3,2) → ReLU → ConvT3D(64,3,2) → ReLU → ConvT3D(1,4,2). ConvT3D(c,k,s) denotes a 3D convolutional transpose layer with c output channels, a kernel size k and stride s. The resulting 57 × 57 × 57 voxel is finally transformed as described in section 4.4.

References

Barron, J., & Malik, J. (2015). Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*, 1670–1687.

Besl, P., & McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*, 239–256.

Blake, A., & Marinos, C. (1990). Shape from texture: Estimation, isotropy and moments. *Artificial Intelligence*, *45*, 323–380.

Blanz, V., & Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *Proceedings of the ACM SIGGRAPH conference on computer graphics* (pp. 187–194).

Blender Online Community. (2017). *Blender—A 3D modelling and rendering package*. Amsterdam: Blender Foundation, Blender Institute.

Boyer, E., & Franco, J. (2003). A hybrid approach for computing visual hulls of complex objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Cashman, T. J., & Fitzgibbon, A. W. (2013). What shape are dolphins? Building 3D morphable models from 2D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*, 232–244.

Chang, A., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., & Su, H., et al. (2015). Shapenet: An information-rich 3D model repository. arXiv preprint arXiv:1512.03012.

Choy, C., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016). 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In*Proceedings of the European conference on computer vision*.

Fan, H., Su, H., & Guibas, L. (2016). 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*.

Fouhey, D. F., Hussain, W., Gupta, A., & Hebert, M. (2015). Single image 3D without a single 3D image. In *Proceedings of the international conference on computer vision*.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1)., Springer series in statistics New York: Springer.

Gadelha, M., Maji, S., & Wang, R. (2016). 3D shape induction from 2D views of multiple objects. arXiv preprint arXiv:1612.05872.

Girdhar, R., Fouhey, D., Rodriguez, M., & Gupta, A. (2016). Learning a predictable and generative vector representation for objects. In *Proceedings of the European conference on computer vision* (pp. 484–499).

Groueix, T., Fisher, M., Kim, V. G., Russell, B., & Aubry, M. (2018). Atlasnet: A papier-mâché approach to learning 3d surface generation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press. ISBN: 0521540518.

Hedau, V., Hoiem, D., & Forsyth, D. (2009). Recovering the spatial layout of cluttered rooms. In *Proceedings of the international conference on computer vision*.

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

Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial transformer networks. In *Advances in neural information processing systems* (pp 2017–2025).

Kar, A., Häne, C., & Malik, J. (2017). Learning a multi-view stereo machine. In *Advances in neural information processing systems* (pp. 364–375).

Kar, A., Tulsiani, S., Carreira, J., & Malik, J. (2015). Category-specific object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Kolev, K., Klodt, M., Brox, T., & Cremers, D. (2009). Continuous global optimization in multiview 3D reconstruction. *International Journal of Computer Vision*, *84*, 80–96.

Kong, C., Lin, C. H., Lucey, S. (2017). Using locally corresponding cad models for dense 3D reconstructions from a single image. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(2), 150–162.

Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Matusik, W., Buehler, C., Raskar, R., Gortler, S., & McMillan, L. (2000). Image-based visual hulls. In *Proceedings of the ACM SIGGRAPH conference on computer graphics*.

Novotny, D., Larlus, D., & Vedaldi, A. (2017). Learning 3D object categories by looking around them. In *Proceedings of the international conference on computer vision*.

Park, E., Yang, J., Yumer, E., Ceylan, D., & Berg, A. (2017). Transformation-grounded image generation network for novel 3D view synthesis. In *Proceedings of the ieee conference on computer vision and pattern recognition*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. In *Proceedings of NIPS 2017 workshop on Autodiff*.

Prasad, M., Fitzgibbon, A. W., Zisserman, A., & Van Gool, L. (2010). Finding nemo: Deformable object class modelling using curve matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Qi, C. R., Su, H., Niessner, M., Dai, A., Yan, M., & Guibas, L. J. (2016). Volumetric and multi-view CNNS for object classification on 3D data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Rezende, D., Eslami, S. M. A., Mohamed, S., Battaglia, P., Jaderberg, M., & Heess, N. (2016). Unsupervised learning of 3D structure from images. In *Advances in neural information processing systems* (pp. 4997–5005).

Rock, J., Gupta, T., Thorsen, J., Gwak, J., Shin, D., & Hoiem, D. (2015). Completing 3D object shape from one depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of* *the international conference on medical image computing and computer assisted intervention*.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. *Proceedings of the ieee conference on computer vision and pattern recognition* (Vol. 1, pp. 519–528).

Sinha, A., Unmesh, A., Huang, Q., & Ramani, K. (2017). Surfnet: Generating 3D shape surfaces using deep residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Sketchfab. (2018). Sketchfab. Available at: https://sketchfab.com/. Accessed 14 Oct 2018.

Soltani, A. A., Huang, H., Wu, J., Kulkarni, T. D., & Tenenbaum, J. B. (2017). Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the international conference on computer vision*.

Tatarchenko, M., Dosovitskiy, A., & Brox, T. (2016). Multi-view 3D models from single images with a convolutional network. In *Proceedings of the European Conference on computer vision*.

Tulsiani, S., Zhou, T., Efros, A., & Malik, J. (2017). Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Upchurch, P., Gardner, J., Pleiss, G., Pless, R., Snavely, N., Bala, K., & Weinberger, K. (2017). Deep feature interpolation for image content changes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Vicente, S., Carreira, J., Agapito, L., & Batista, J. (2014). Reconstructing Pascal voc. In:*Proceedings of the IEEE conference on computer vision and pattern recognition*.

Vogiatzis, G., Torr, P. H. S., Cipolla, R. (2003). Bayesian stochastic mesh optimization for 3D reconstruction. In *Proceedings of the 14th British machine vision conference, Norwich* (pp. 711–718).

Wiles, O., & Zisserman, A. (2017). Silnet : Single- and multi-view reconstruction by learning from silhouettes. In *Proceedings of the British machine vision conference*.

Witkin, A. P. (1981). Recovering surface shape and orientation from texture. *Artificial Intelligence*, *17*, 17–45.

Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, B., & Tenenbaum, J. (2017). Marrnet: 3D shape reconstruction via 2.5D sketches. In *Advances in neural information processing systems*.

Wu, J., Zhang, C., Xue, T., Freeman, B., & Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in neural information processing systems* (pp. 82–90).

Xiang, Y., Mottaghi, R.,& Savarese, S. (2014). Beyond Pascal: A benchmark for 3d object detection in the wild. In *Proceedings of the IEEE workshop on applications of computer vision*.

Yan, X., Yang, J., Yumer, E., Guo, Y., & Lee, H. (2016). Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision. In *Advances in neural information processing systems*.

Zhang, R., Tsai, P. S., Cryer, J. E., & Shah, M. (1999). Shape-from-shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(8), 690–706.

Zhou, Q. Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. arXiv:1801.09847.

Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A. (2016). View synthesis by appearance flow. In *Proceedings of the European conference on computer vision*

Zhu, R., Galoogahi, H. K., Wang, C., Lucey, S. (2017). Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. In *Proceedings of the international conference on computer vision* (pp. 57–65).

Zollhöfer, M., Dai, A., Innmann, M., Wu, C., Stamminger, M., Theobalt, C., et al. (2015). Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*, *34*, 96.