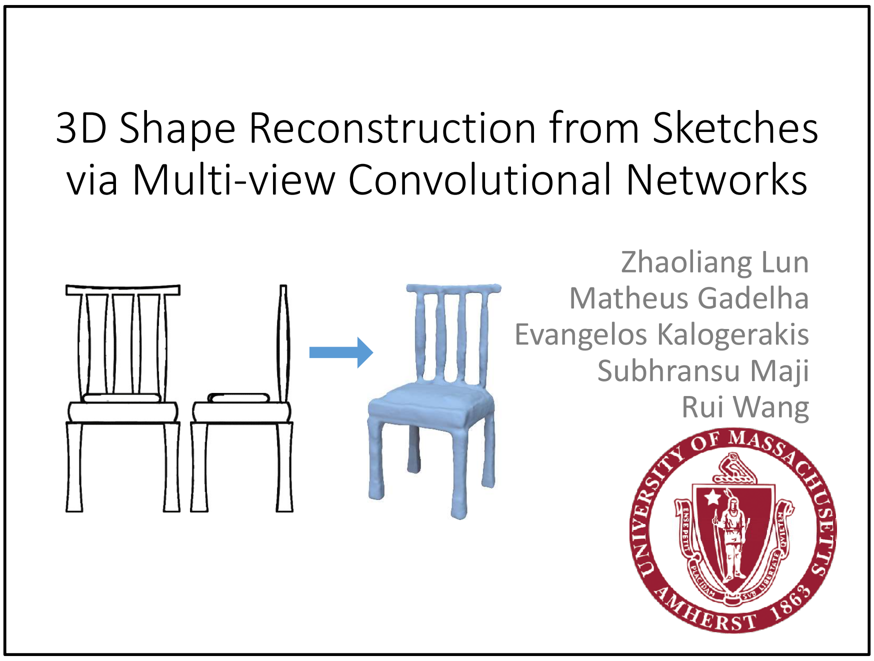


3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks

Zhaoliang Lun
Matheus Gadelha
Evangelos Kalogerakis
Subhransu Maji
Rui Wang

Hello everyone, my name is Zhaoliang Lun. Today I am going to present our paper on reconstructing 3D shapes from sketches using multi-view convolutional networks.

Creating 3D shapes is not easy

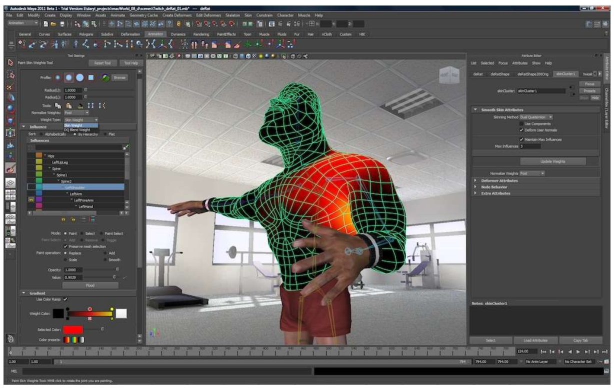
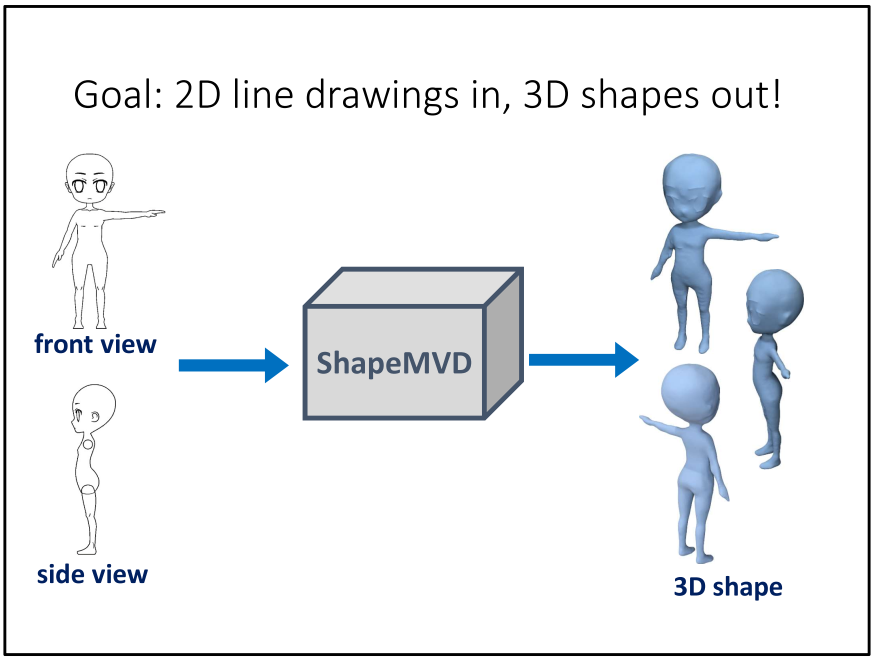


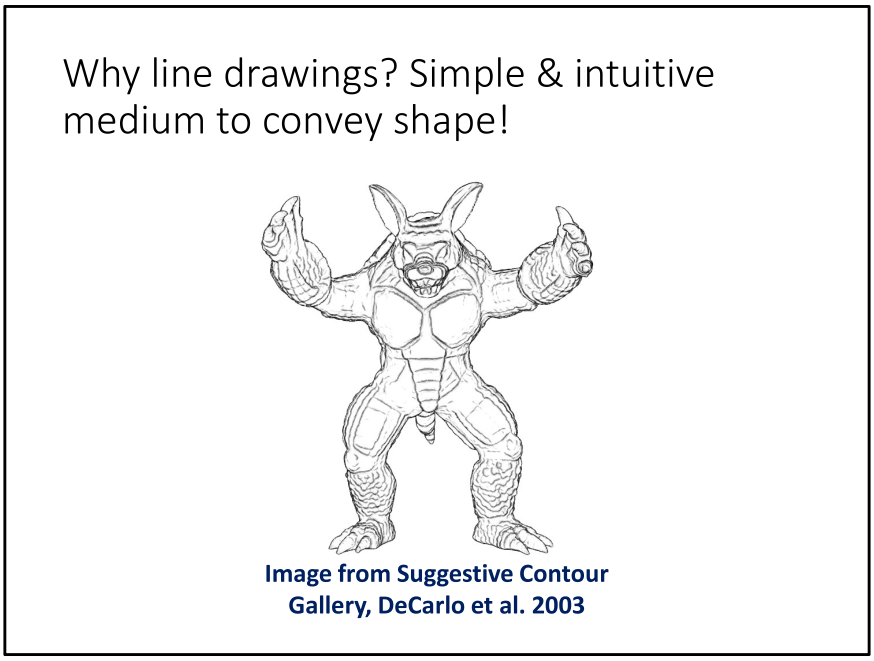
Image from Autodesk 3D Maya

Creating compelling 3D models of shapes is a time-consuming and laborious task, which is often out of reach for users without modeling expertise and artistic skills. Existing 3D modeling tools have steep learning curves and complex interfaces for handling low-level geometric primitives.



Goal: 2D line drawings in, 3D shapes out!

The goal of our project is to make it easy for people to create 3D models. We designed a deep architecture, called Shape Multi-View Decoder, in short ShapeMVD, that takes as input one or multiple sketches in the form of line drawings, such as the ones that you see on the left, and outputs a complete 3D shape that you see on the right.



Why line drawings? Simple & intuitive medium to convey shape!

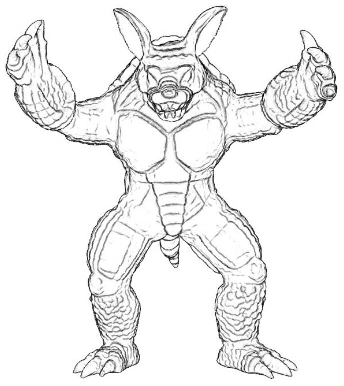
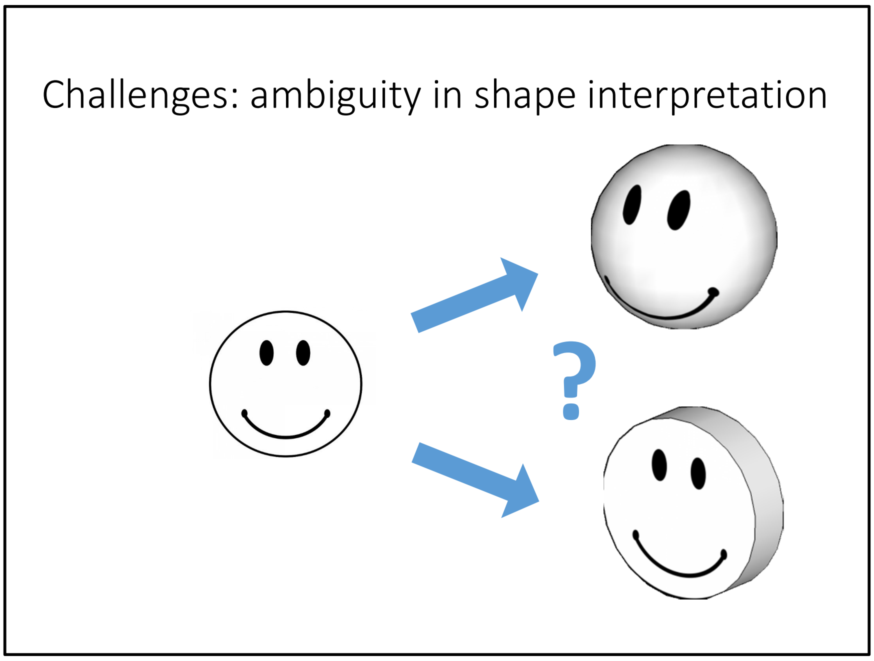


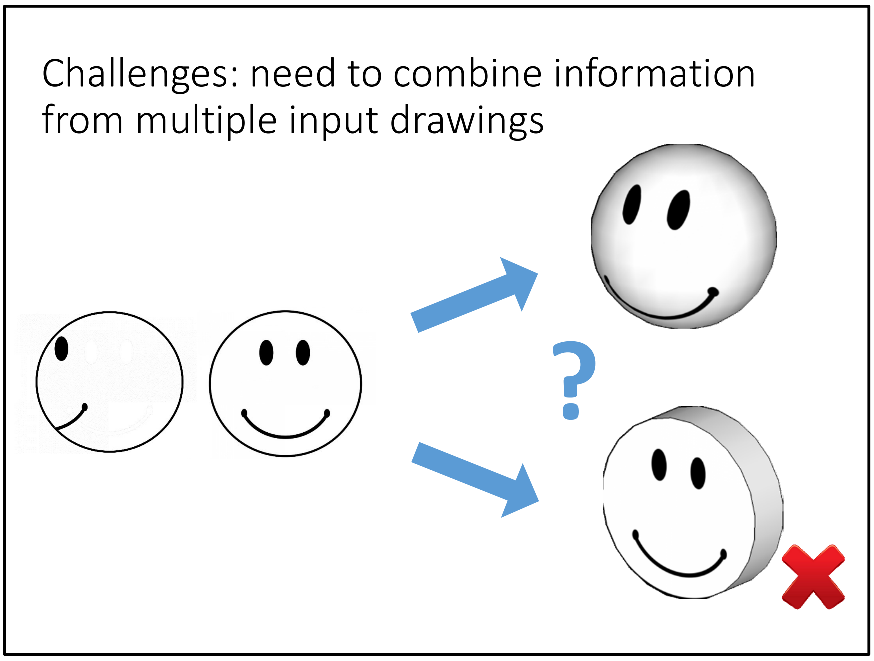
Image from Suggestive Contour Gallery, DeCarlo et al. 2003

Why did we choose line drawings as the input representation? Line drawings is a simple and intuitive medium for artists and casual modelers. By drawing a few silhouettes and internal contours, humans can effectively convey shape. Modelers often prototype their design by using line drawings.



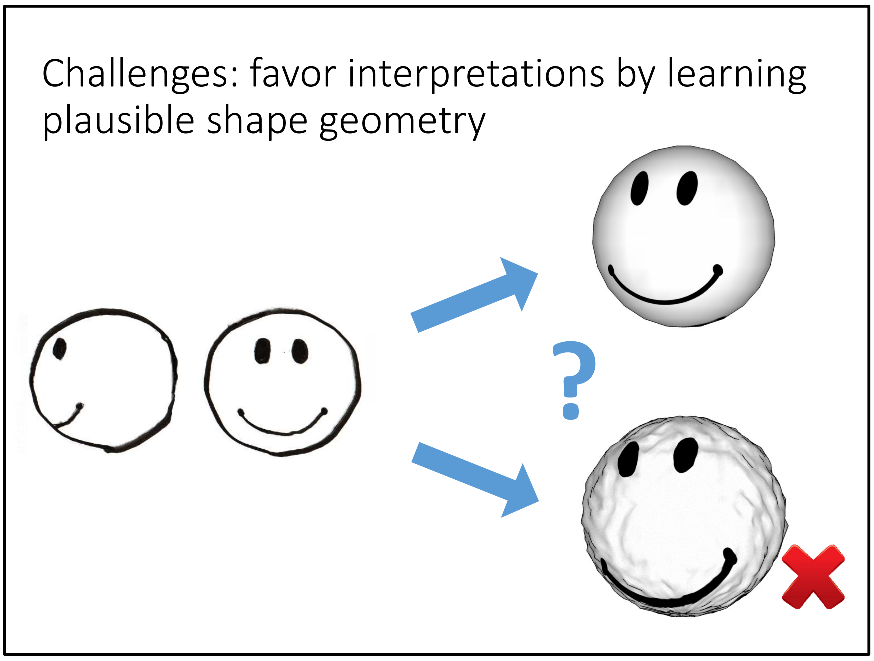
Challenges: ambiguity in shape interpretation

On the other hand, converting 2D sketches to 3D shapes has a number of challenges. First, there is often no single 3D shape interpretation given a single input sketch. For example, given a drawing of a 2D smiley face here, one possible interpretation is a spherical 3D face you see on the top. Another possible interpretation is the button-shape head you see on the bottom.

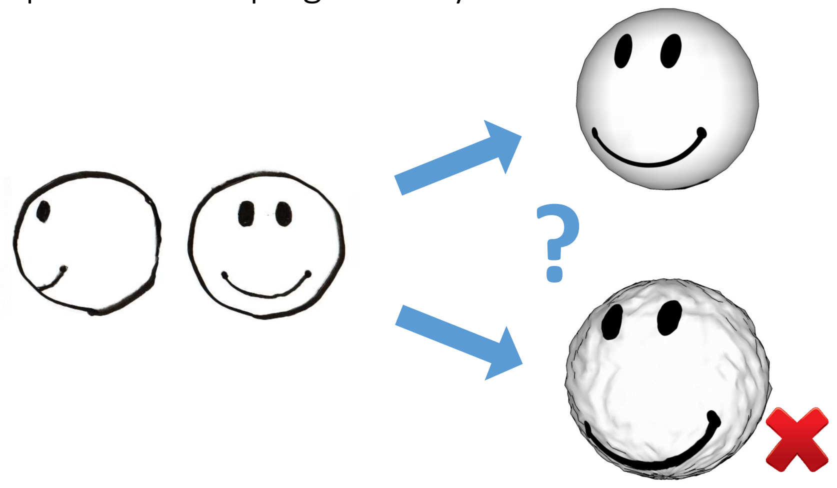


Challenges: need to combine information from multiple input drawings

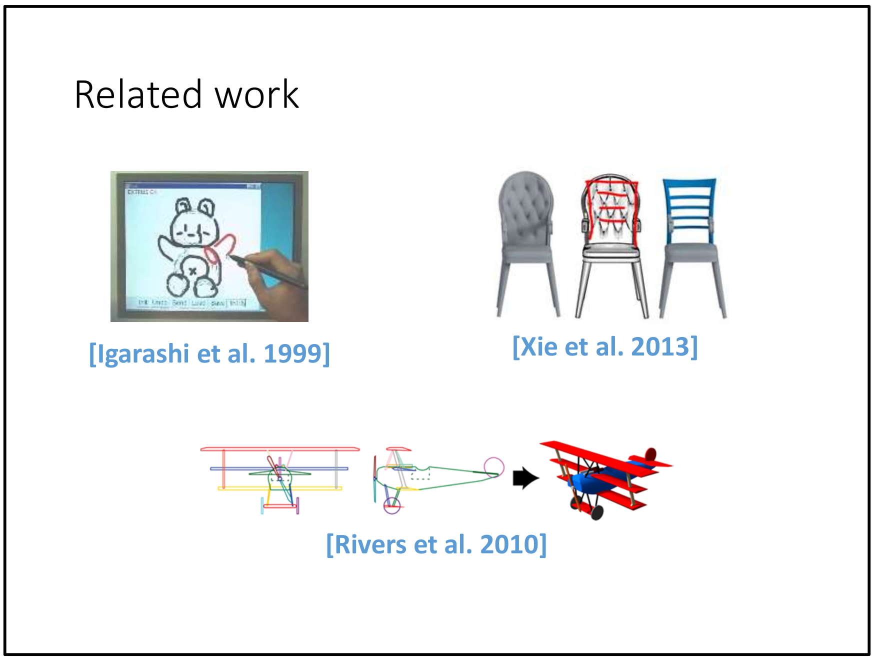
One way to partially disambiguate the output is by using multiple input line drawings from different views. For example, given a second line drawing, the button-shape interpretation becomes inconsistent with the input. A technical challenge here is how to effectively combine information from all input sketches to reconstruct a single, coherent 3D shape as output.



Challenges: favor interpretations by learning plausible shape geometry



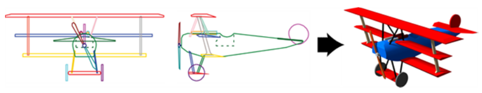
One more challenge is that human drawings are not perfect. The strokes might not be accurate, smooth, or even consistent across views. For example, given these human line drawings, one possible interpretation could be a non-symmetric, noisy head that you see on the bottom. **[CLICK]** A data-driven approach can instead favor more plausible interpretations by learning models of geometry from collections of plausible 3D shapes. For example, if we have a database of 3D symmetric heads, a learning approach would learn to reconstruct symmetric heads and favor the top interpretation, which is far more plausible.



Related work

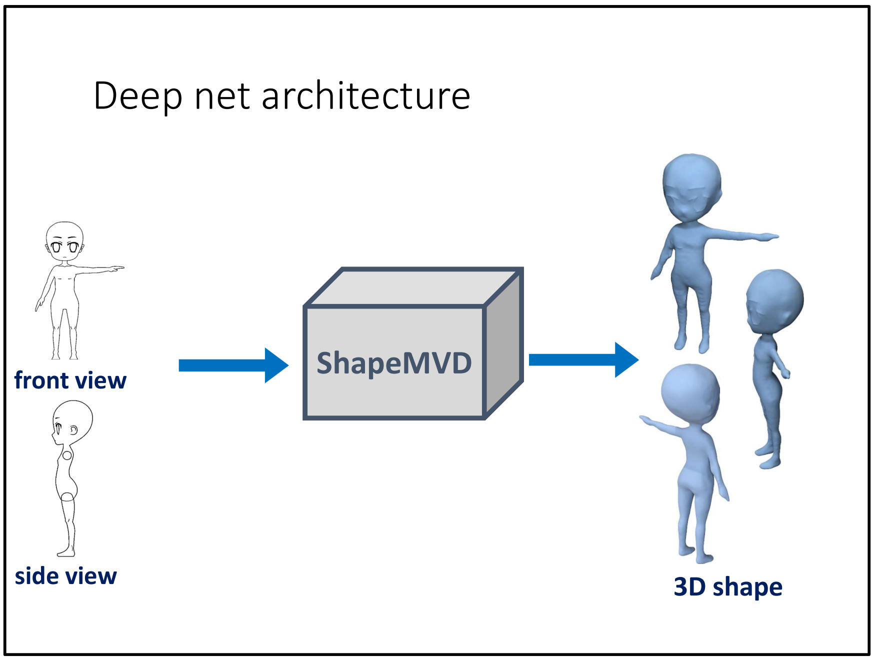
[Igarashi et al. 1999]

[Xie et al. 2013]

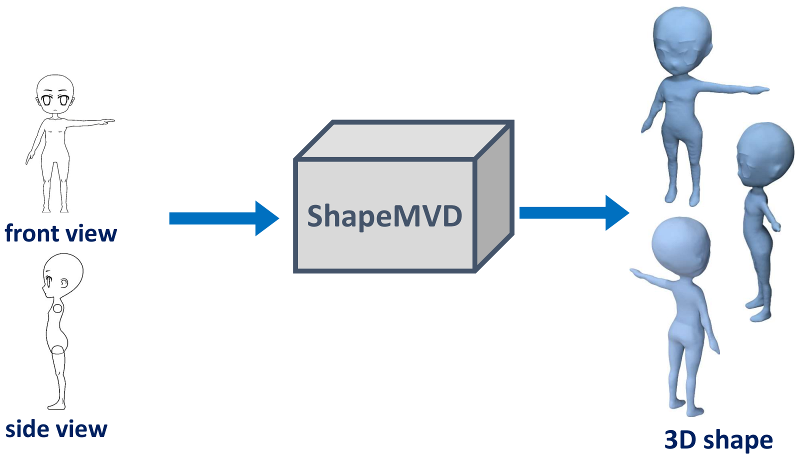


[Rivers et al. 2010]

In the past, researchers tried to tackle this problem primarily through non-learning approaches. However, these approaches were based on hand-engineered pipelines or descriptors, and required significant manual user interaction. We adopt a neural network approach to automatically learn the mapping between sketches and shape geometry.

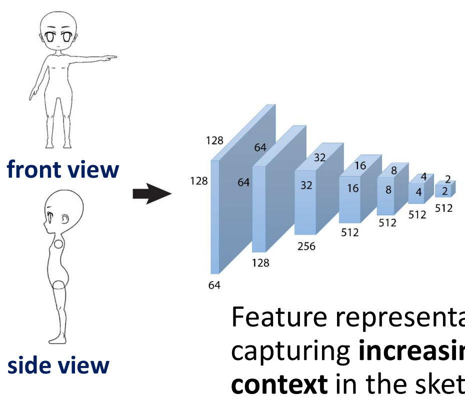


Deep net architecture



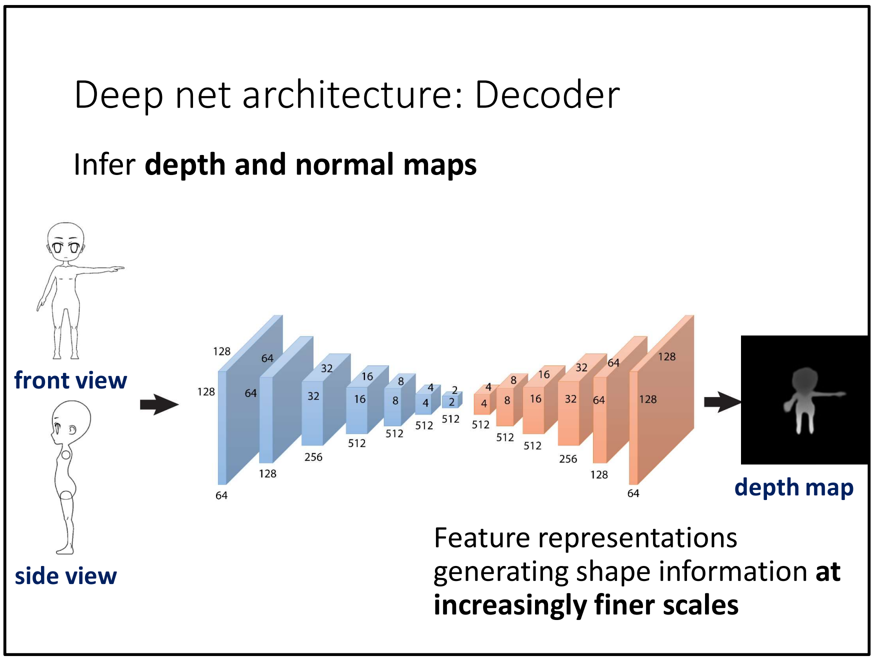
I will now describe our deep network architecture.

Deep net architecture: Encoder



Feature representations capturing **increasingly larger context** in the sketches

First, the line drawings are ordered according to the input viewpoint, and concatenated into an image with multiple channels. This image passes through an encoder with convolutional layers that extract feature representation maps. As we go from the left to the right, the feature maps capture increasingly larger context in the sketch images – the first feature maps encode local sketch patterns, such as stroke edges and junctions, and towards the end, the last map encodes more global patterns, such as what type of character is drawn, or what parts it has.



Deep net architecture: Decoder

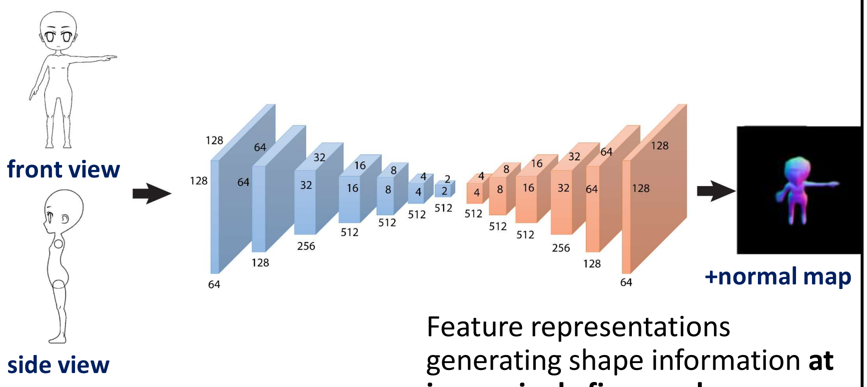
Infer **depth and normal maps**

Feature representations generating shape information **at increasingly finer scales**

The second part of the network is a decoder which has a similar but reversed architecture of the encoder. Going from left to right the feature representations generate shape feature maps at increasingly finer scales. The last layer of the decoder outputs an image that contains the predicted depth, in other words, a depth map for a particular output viewpoint.

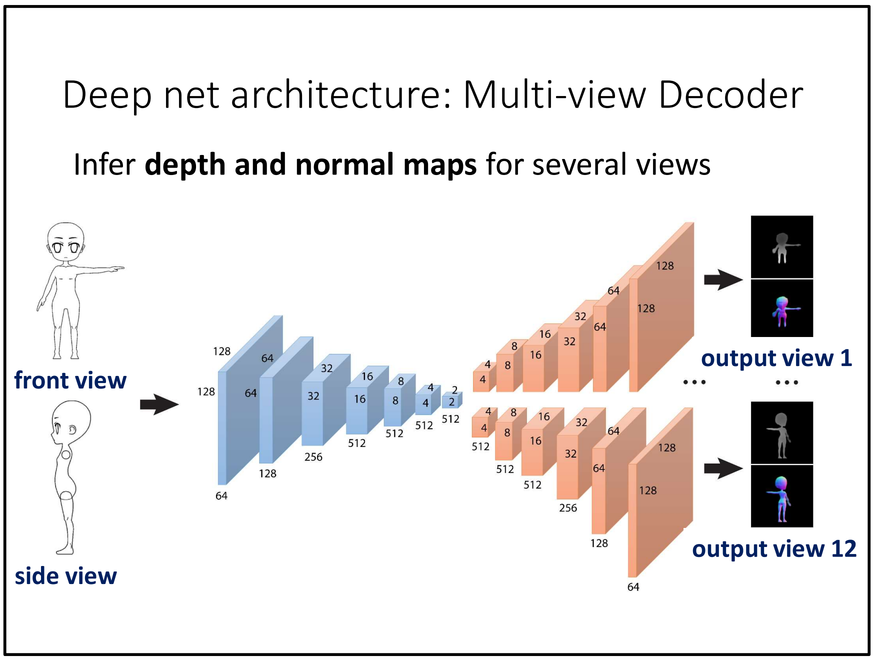
Deep net architecture: Decoder

Infer **depth and normal maps**



Feature representations generating shape information **at increasingly finer scales**

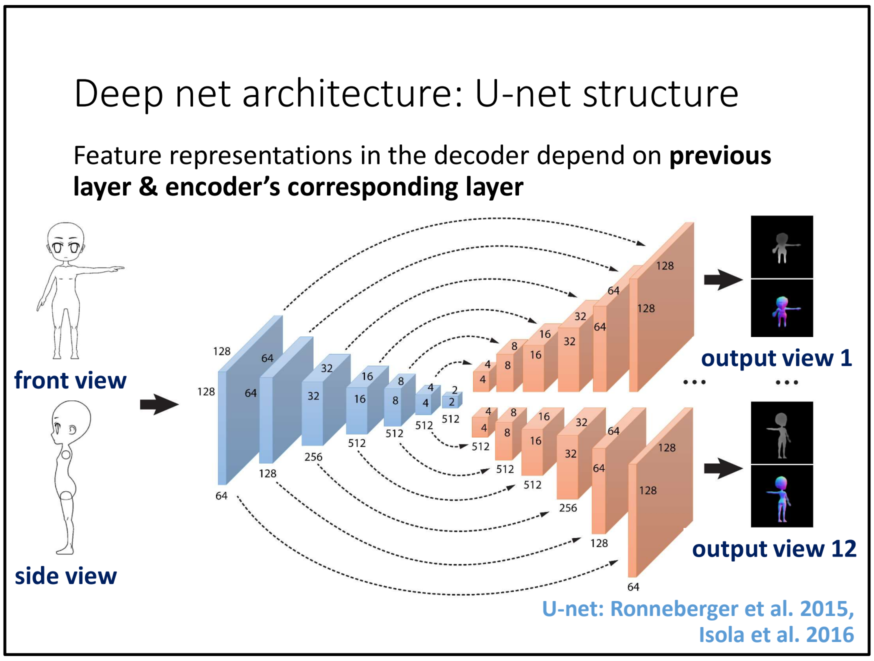
and also one more image that contains predicted normals, in other words a normal map, for that particular output viewpoint. The normals are 3D vectors, encoded as RGB channels in the output normal map.



Deep net architecture: Multi-view Decoder

Infer **depth and normal maps** for several views

One viewpoint is not enough to capture a surface in 3D. Thus, we output multiple depth and normals maps from several viewpoints to deal with self-occlusions. We use a different decoder branch for each output viewpoint. In total we have 12 fixed output viewpoints placed at the vertices of a regular icosahedron.



Deep net architecture: U-net structure

Feature representations in the decoder depend on **previous layer & encoder's corresponding layer**

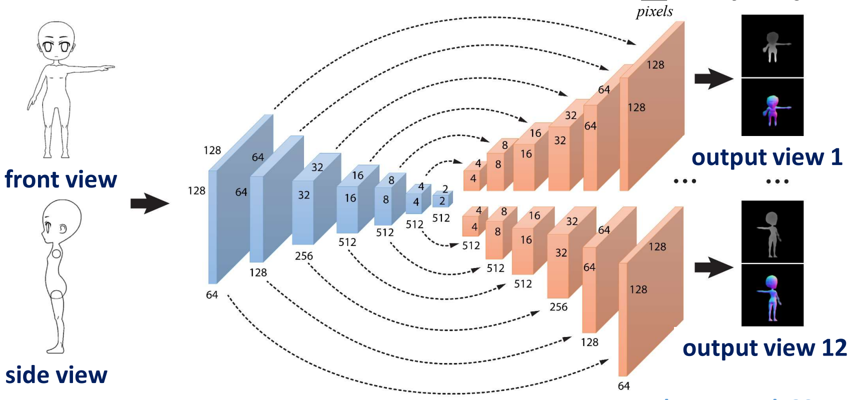
U-net: Ronneberger et al. 2015, Isola et al. 2016

If the decoder relies exclusively on the last feature map of the encoder, then it will fail to reconstruct fine-grained local shape details. These details are captured in the earlier encoder layers. Thus, we employed a U-Net architecture. Each decoder layer processers the maps of the previous layer, and also the maps of the corresponding, symmetric layer from the encoder.

Training: initial loss

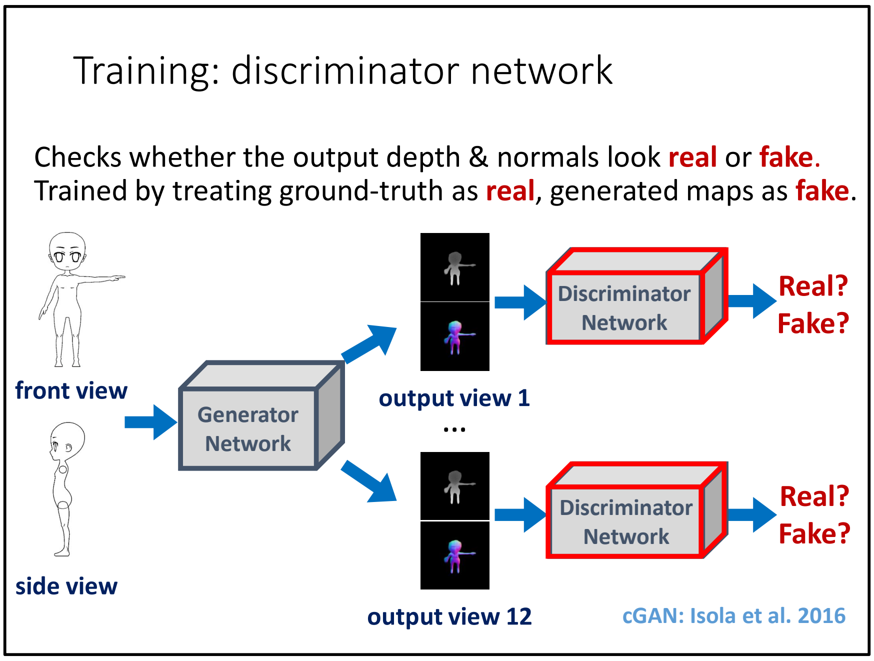
Penalize per-pixel depth reconstruction error: $\sum_{pixels} |d_{pred} - d_{gt}|$

& per-pixel normal reconstruction error: $\sum_{pixels} (1 - n_{pred} \cdot n_{gt})$



U-net: Ronneberger et al. 2015, Isola et al. 2016

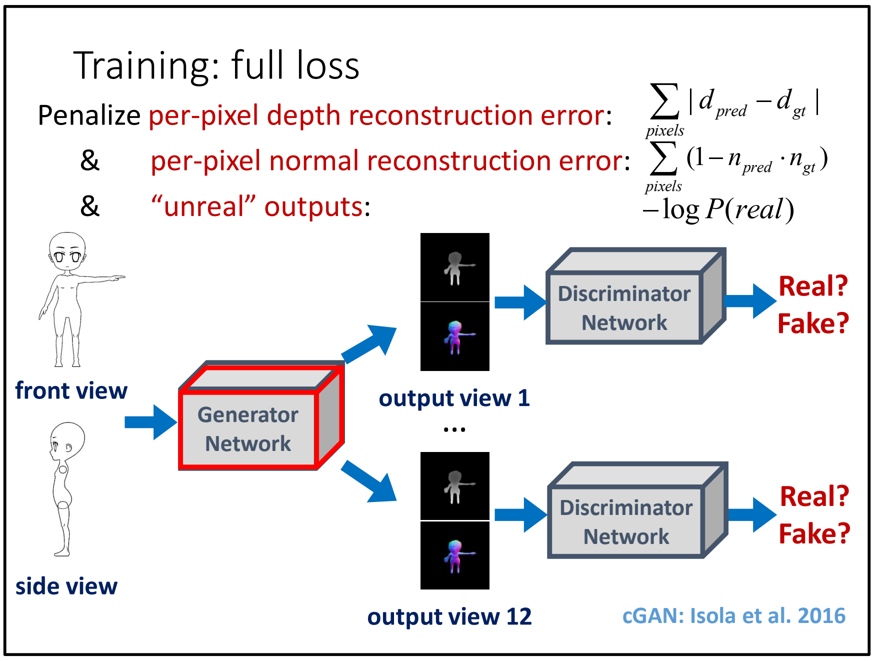
To train this generator network, as a first step, we first employ a loss function that penalizes per-pixel depth and normal reconstruction loss. This loss function, however, focuses more on getting the individual pixel predictions correct, rather than making the output maps plausible as a whole.



Training: discriminator network

Checks whether the output depth & normals look **real** or **fake**.
Trained by treating ground-truth as **real**, generated maps as **fake**.

Therefore, we also train a discriminator network that decides whether the output depth and normal maps, as a whole, look good or bad, in other words, real or fake. The discriminator network is trained such that it predicts ground-truth maps as real, and the generated maps as fake.



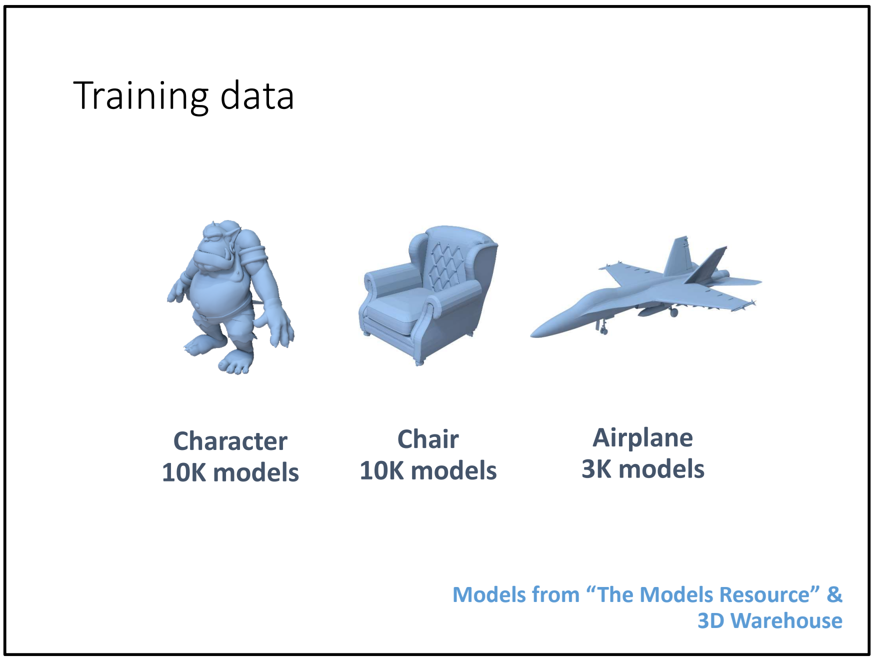
Training: full loss

Penalize per-pixel depth reconstruction error: $\sum_{pixels} | d_{pred} - d_{gt} |$

& per-pixel normal reconstruction error: $\sum_{pixels} (1 - n_{pred} \cdot n_{gt})$

& "unreal" outputs: $-\log P(real)$

At the subsequent steps, the generator network is trained to fool the discriminator. Our loss function is augmented with one more term that penalizes unreal outputs according to the trained discriminator output. Both the generator and discriminator are trained interchangeably. This is also known as the conditional GAN approach.



Training data

Character
10K models

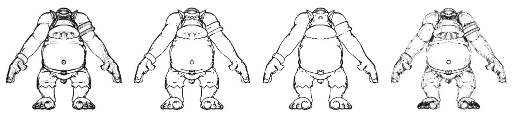
Chair
10K models

Airplane
3K models

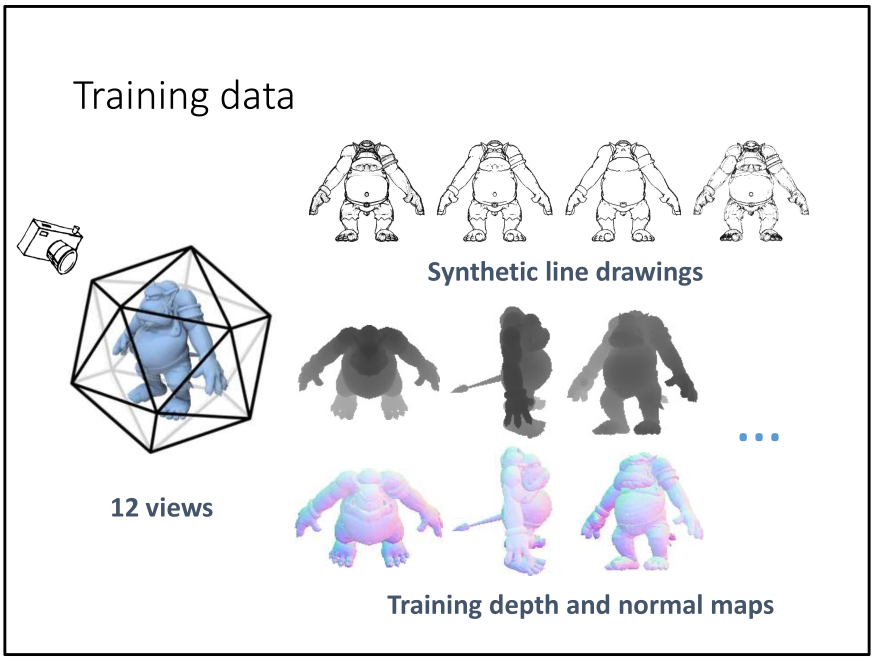
Models from "The Models Resource" &
3D Warehouse

Our architecture is trained per shape category, namely characters, chairs, and airplanes. We have a collection of about 10K characters, 10K chairs, and 3K airplanes.

Training data


Synthetic line drawings

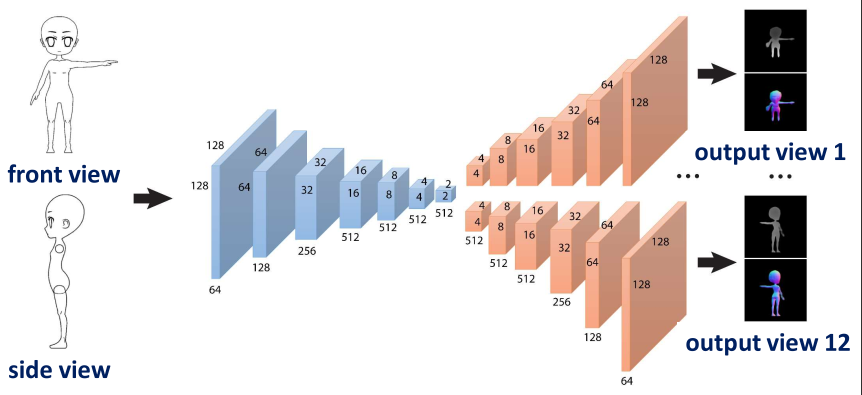
For each training shape, we create synthetic line drawings consisting of a combination of silhouettes, suggestive contours, ridges and valleys.


Training data

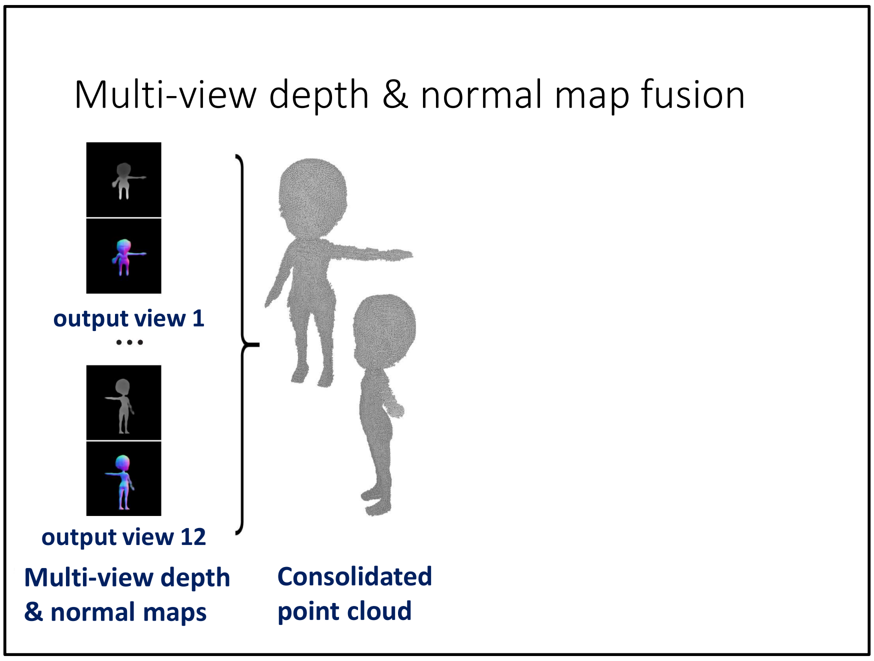
To train our network, we also need ground-truth, multi-view training depth and normal maps. We place each training shape inside a regular icosahedron, then place a viewpoint at each vertex. From each viewpoint we render depth and normal maps.

Test time

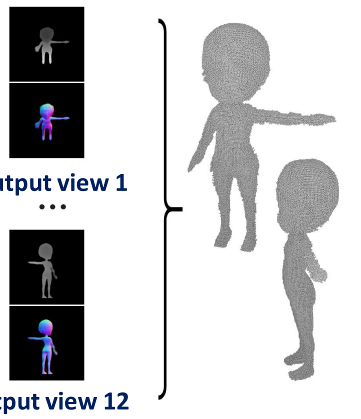
Predict multi-view depth and normal maps!



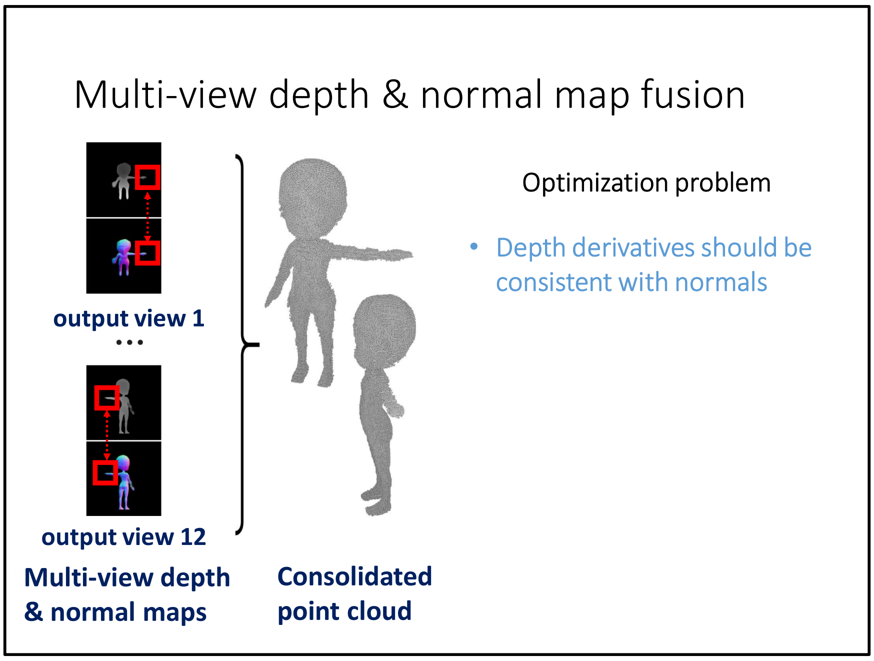
At test time, given the input line drawings, we generate multi-view depth and normal maps based on our learned generator network.



Multi-view depth & normal map fusion



At test time, the output maps are not perfect, meaning that the depths across different viewpoints might not agree on the output surface. The depth derivatives might also be slightly inconsistent with the predicted normals. Thus, we follow an optimization procedure that fuses the depth and normal maps into a coherent point cloud.



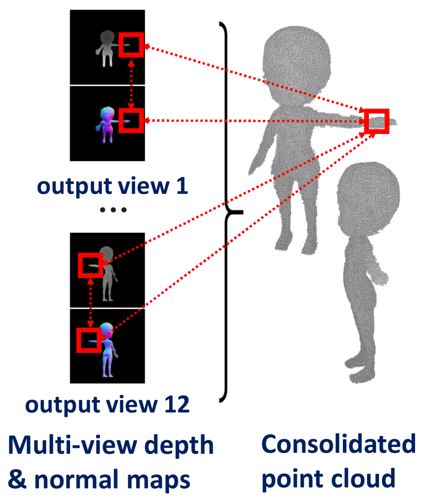
Multi-view depth & normal map fusion

Optimization problem

- Depth derivatives should be consistent with normals

The optimization corrects depths under the following two objectives. First, the depth derivatives, which correspond to surface tangent directions should be as-perpendicular-as-possible to the predicted normals.

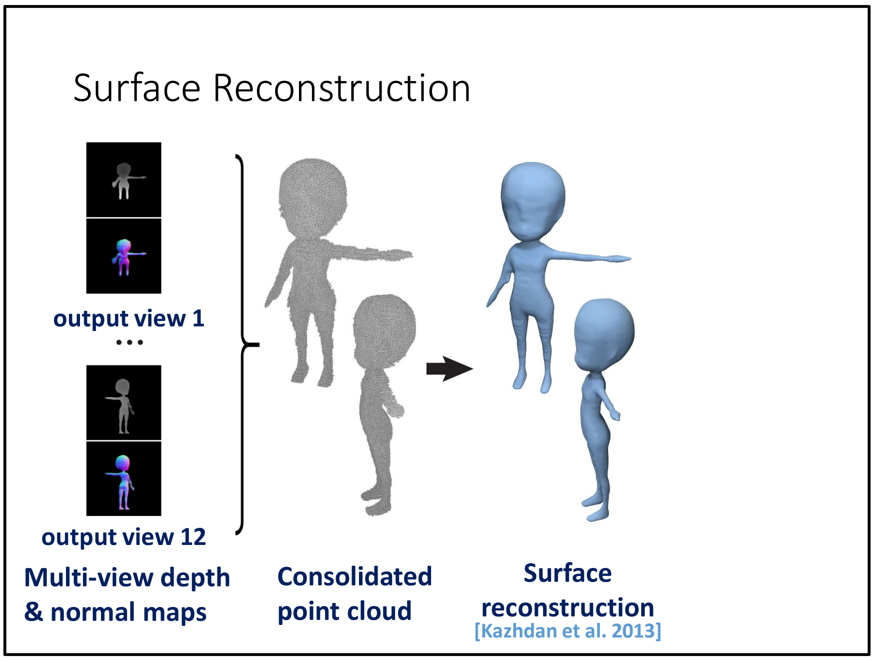
Multi-view depth & normal map fusion



Optimization problem

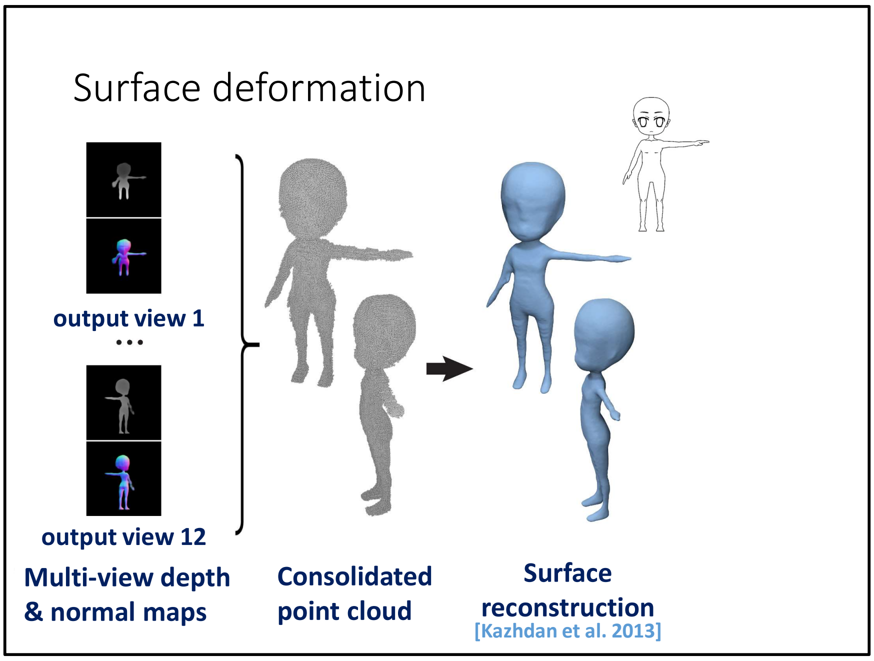
- Depth derivatives should be consistent with normals
- Corresponding depths and normals across different views should agree

Then the depths across different viewpoints should agree. For example, let’s say that we take a pixel from one viewpoint, and map it to a 3D point according to the predicted depth. Then if we project the 3D point onto another viewpoint, then the resulting depth should agree as much as possible with the predicted depth from that viewpoint. This optimization problem can be solved through a linear system – more details in the paper.

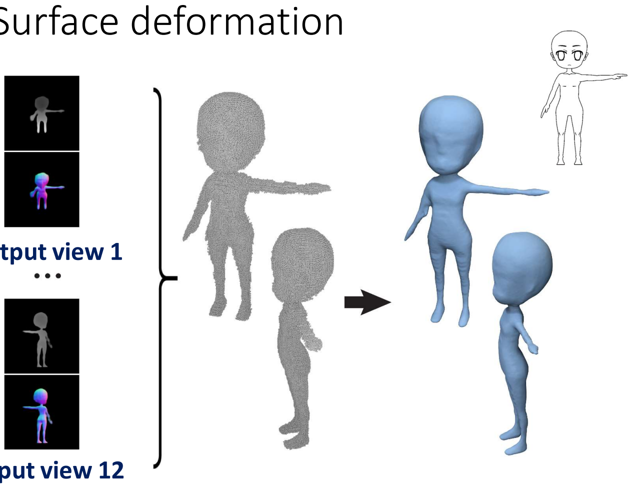


Surface Reconstruction

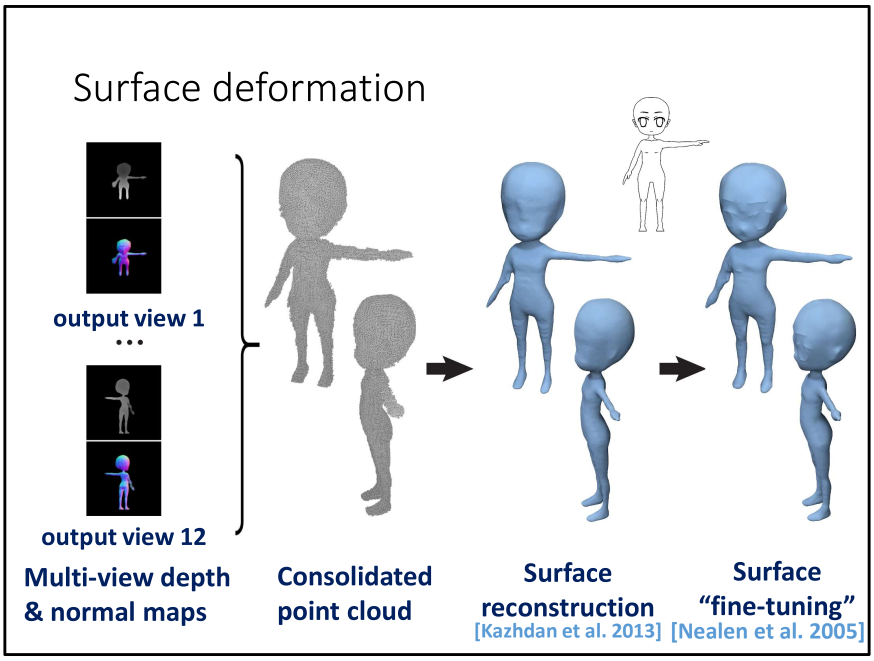
Given the resulting point cloud with normals, we perform surface reconstruction – we use the standard screened Poisson surface reconstruction that yields a polygon mesh.



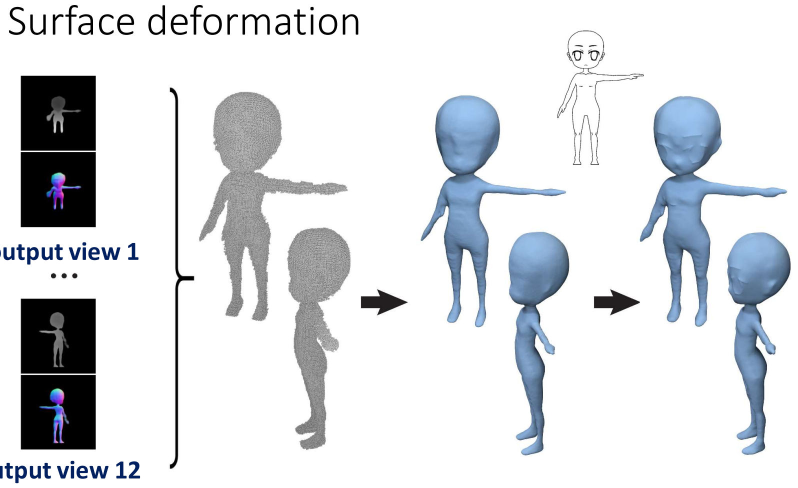
Surface deformation



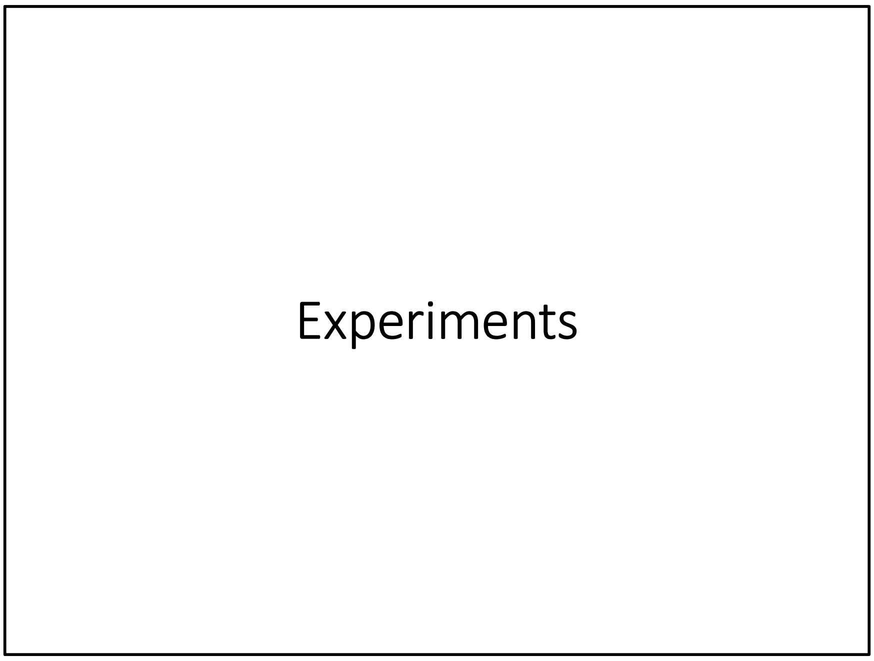
The output mesh tends to lose details in the sketch for various reasons: training is approximate, output resolution is limited to 256x256, or there is no unique surface that is consistent with the input drawings, and so on. To add details, we take each input line drawing.



Surface deformation

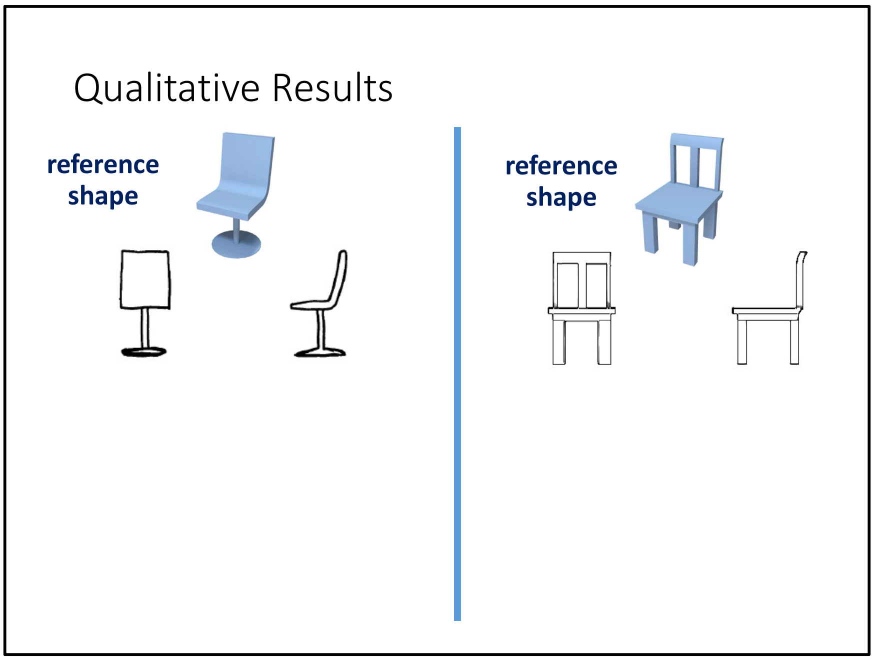


and apply surface deformations, namely laplacian deformations, so that the silhouette, ridges, and valleys of the surface agree with the strokes of the input sketches. We refer you to the paper and previous work on sketch-driven surface deformations for more details.



Experiments

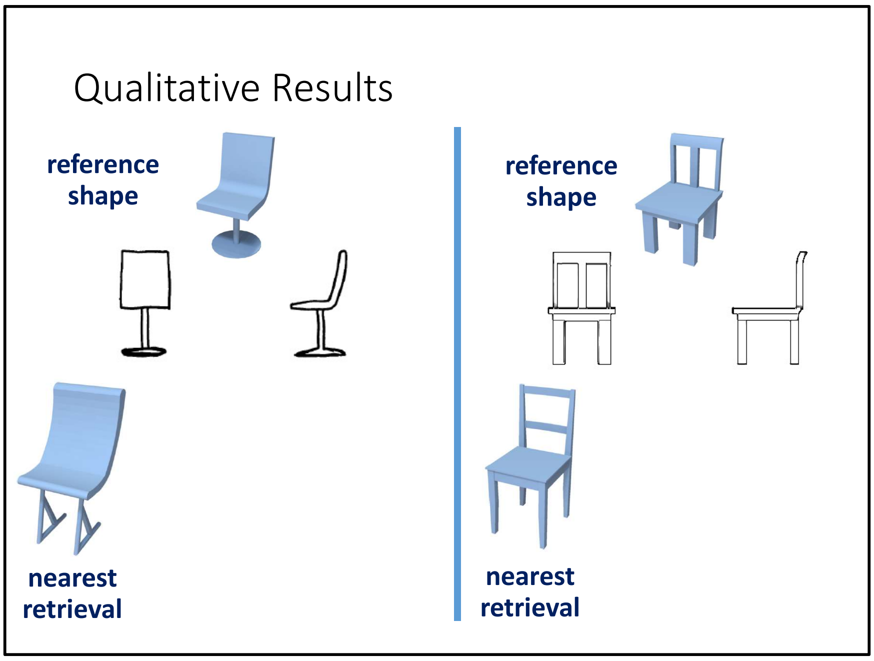
We now discuss our experiments to evaluate our method and alternatives.



Qualitative Results

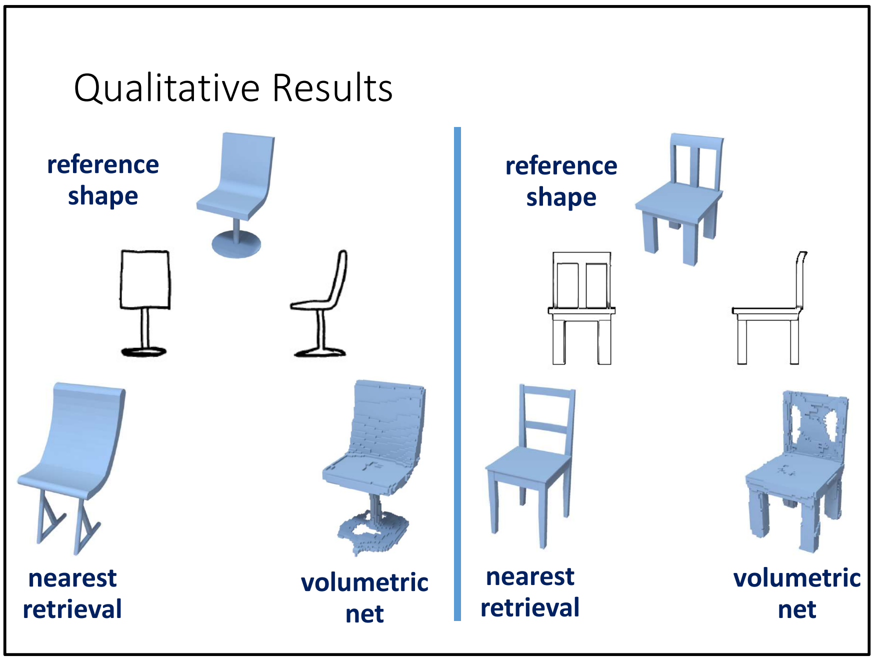
reference shape
reference shape

To evaluate different methods, we showed reference shapes to a few volunteers who participated in an informal user study. We asked them to provide line drawings. The goal of the evaluation is to compare how well reconstructed shapes from line drawings match the reference shapes.



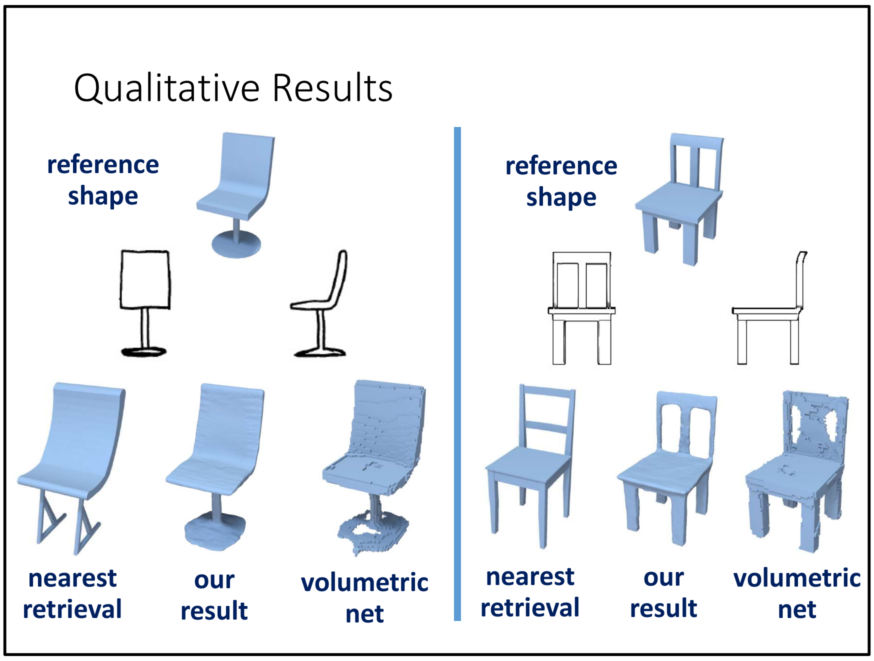
Qualitative Results

A simple baseline method is to recover the nearest training through sketch-based retrieval. Nearest sketch retrieval might find a shape that looks plausible, since it is modeled by an artist. However, it will often not match the input sketch – for example, the back and seat of the retrieved chair are similar to the back and seat of the reference shape, however their legs are different.



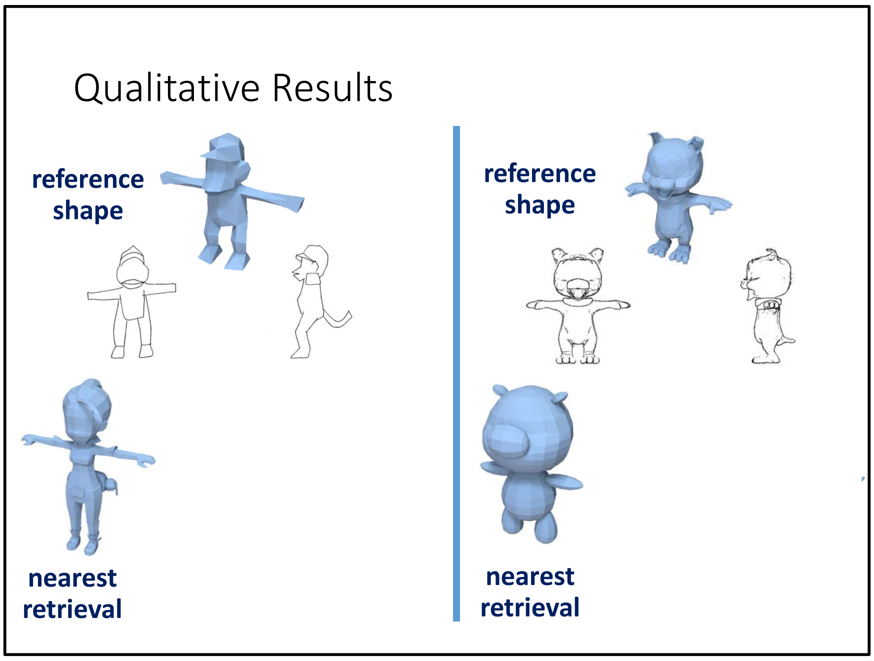
Qualitative Results

Here is the output reconstruction from a method that outputs voxels in a 128x128x128 binary voxel grid. The volumetric method is trained on the same training data, and a loss function that incorporates cross-entropy for voxel prediction. We tried to keep the comparison fair by matching the number of layers and parameters of the volumetric network with the ones in our network, and optimizing all hyper-parameters similarly. The method tends to produce shapes whose topology, part proportions, and structure do not match well with the reference shape.



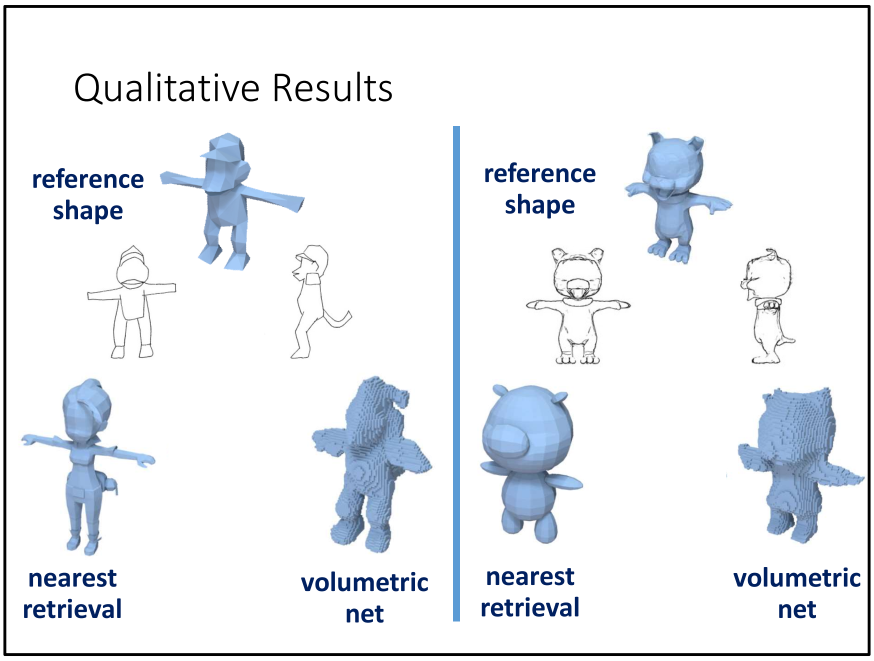
Qualitative Results

Our reconstruction is shown in the middle. Even if our result tends to miss details, or does not have the quality of shapes modeled by artists, our result approximates the reference shape much better in terms of structure, topology, part style and proportions, compared to nearest retrieval or volumetric reconstruction.



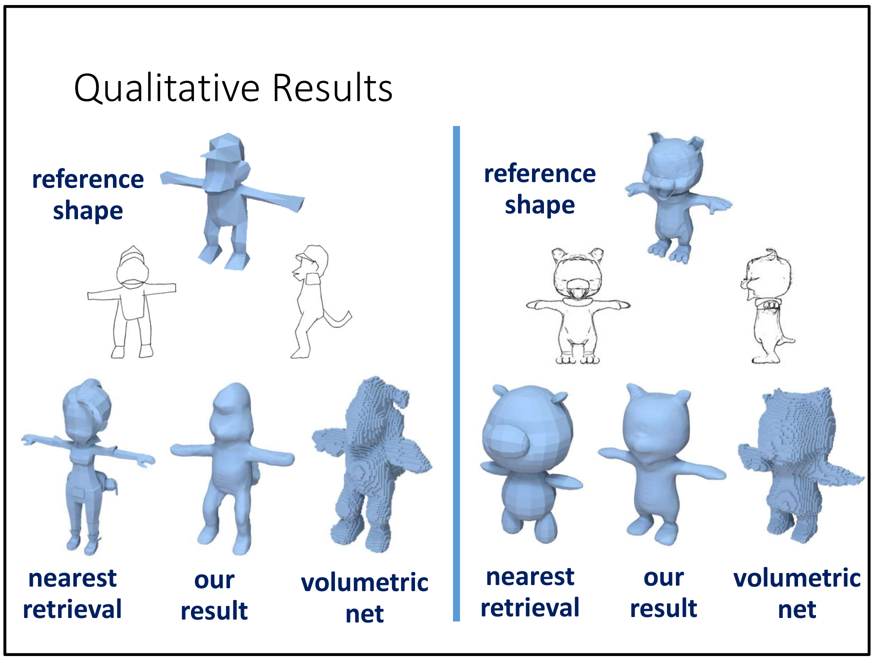
Qualitative Results

These are the results for characters. Again nearest retrieval can give a shape which might not have the parts or style depicted in the input sketches.



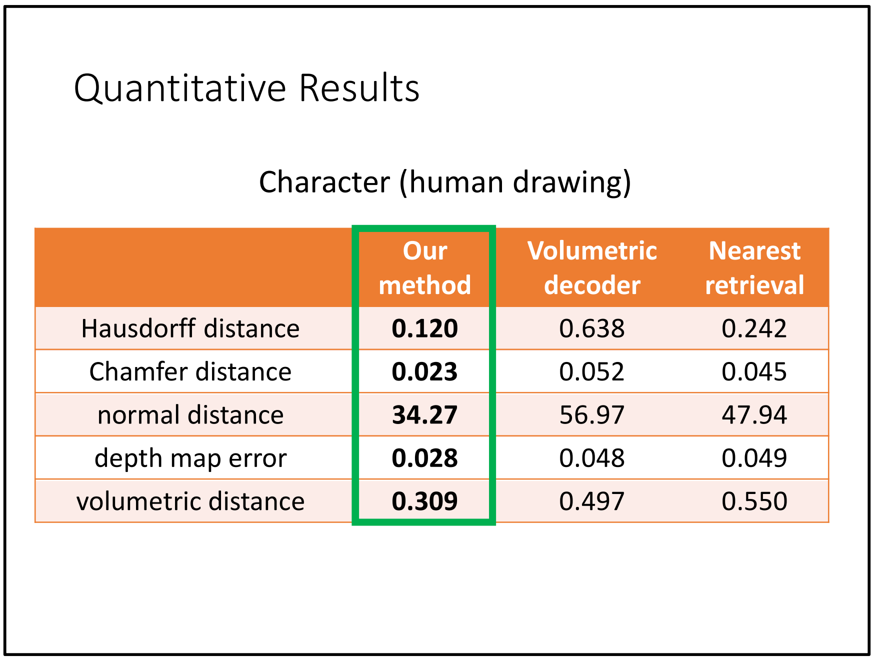
Qualitative Results

The volumetric reconstruction is overly too coarse.



Qualitative Results

Our result captures the parts and overall shape depicted in the input drawings better.

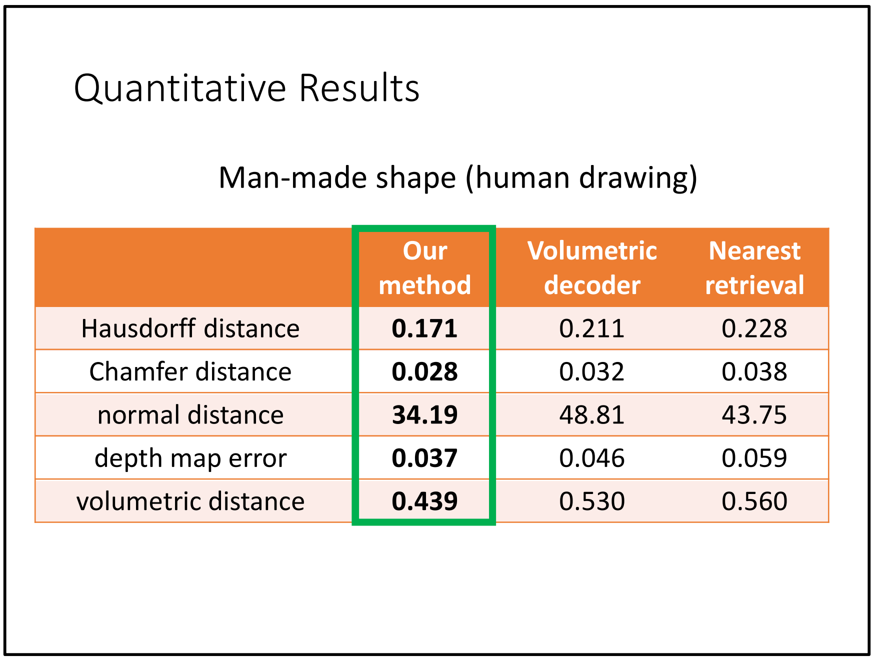


Quantitative Results

Character (human drawing)

	Our method	Volumetric decoder	Nearest retrieval
Hausdorff distance	**0.120**	0.638	0.242
Chamfer distance	**0.023**	0.052	0.045
normal distance	**34.27**	56.97	47.94
depth map error	**0.028**	0.048	0.049
volumetric distance	**0.309**	0.497	0.550

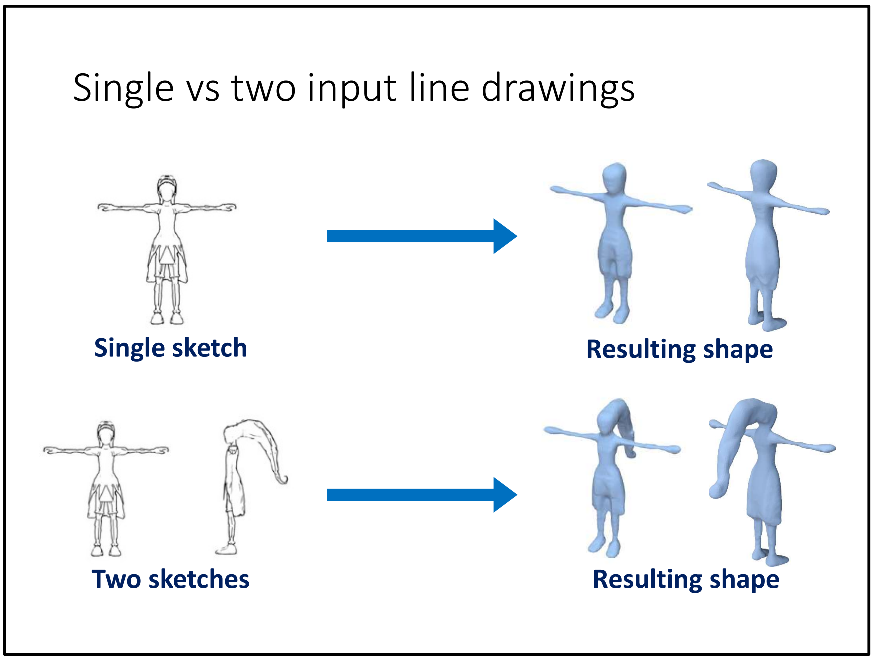
Quantitatively, we can compare the reconstructed shapes and the reference shapes, using various metrics, such as Hausdorff distance, Chamfer distance, angles between normals, depth map error, voxel-based intersection over union. These are the results for character models. According to all metrics, our reconstruction errors are much smaller compared to nearest retrieval and the volumetric network.


Quantitative Results

Man-made shape (human drawing)

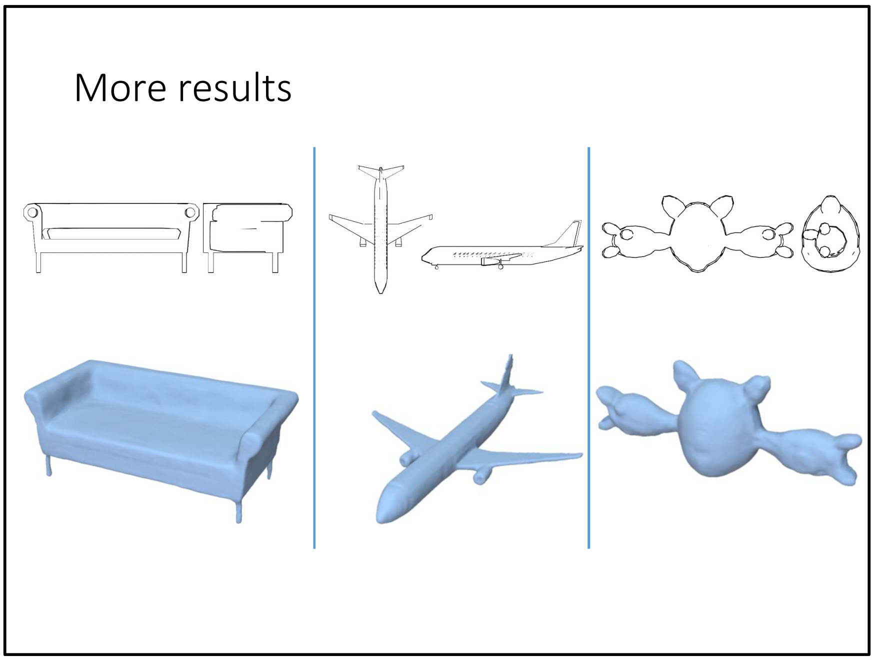
	Our method	Volumetric decoder	Nearest retrieval
Hausdorff distance	**0.171**	0.211	0.228
Chamfer distance	**0.028**	0.032	0.038
normal distance	**34.19**	48.81	43.75
depth map error	**0.037**	0.046	0.059
volumetric distance	**0.439**	0.530	0.560

Here are the results for man-made models. Again we observe the same trend – our methods yields much lower errors.

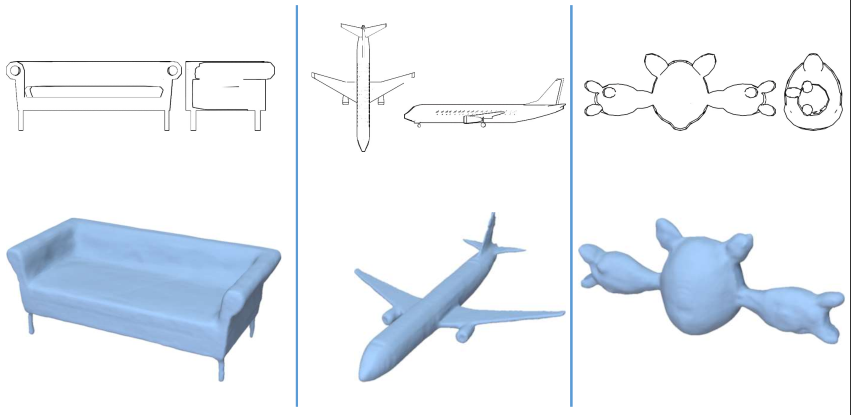


Single vs two input line drawings

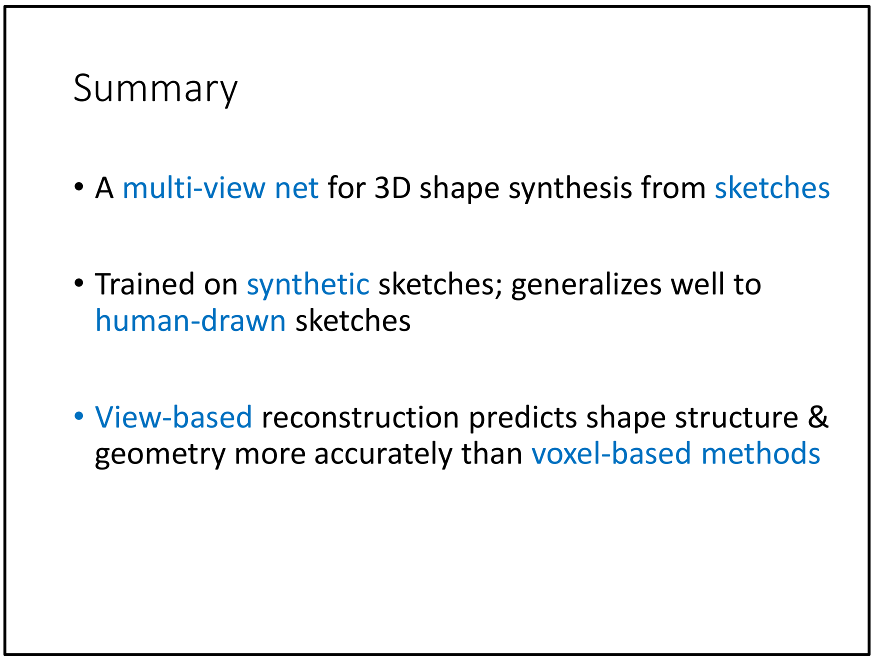
Note that even with a single sketch, our method often outputs a plausible shape – obviously, there is lots of missing information in the input sketch from a single view, for example, the hair ponytail, thus more input sketches help towards creating the desired shape.



More results



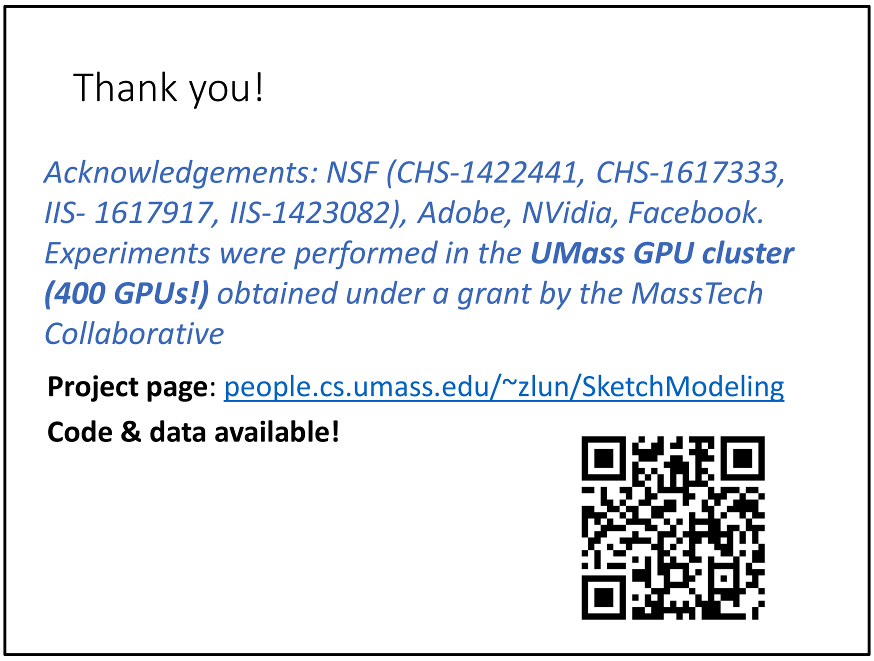
Here we show more results for a sofa, airplane, and a character. As you see here, our results preserve small structures such as thin legs of the sofa, engines of the airplane, or the ears and fingers for the monster.



Summary

- A multi-view net for 3D shape synthesis from sketches
- Trained on synthetic sketches; generalizes well to human-drawn sketches
- View-based reconstruction predicts shape structure & geometry more accurately than voxel-based methods

To summarize, we presented an approach for 3D shape reconstruction from sketches. Our framework is trained on synthetic sketches and we showed that it generalized well to human-drawn sketches. We evaluated our method both qualitatively and quantitatively. Our results indicate that view-based reconstruction is significantly more accurate than a voxel-based reconstruction.



Thank you!

Acknowledgements: NSF (CHS-1422441, CHS-1617333, IIS- 1617917, IIS-1423082), Adobe, NVidia, Facebook. Experiments were performed in the ***UMass GPU cluster (400 GPUs!)*** *obtained under a grant by the MassTech Collaborative*

Project page: people.cs.umass.edu/~zlun/SketchModeling

Code & data available!

Here is a link to our project page. All the codes and data are available to download. Thank you!