IFT 6113
LEARNING (ON) 3D GEOMETRY

http://tiny.cc/6113/

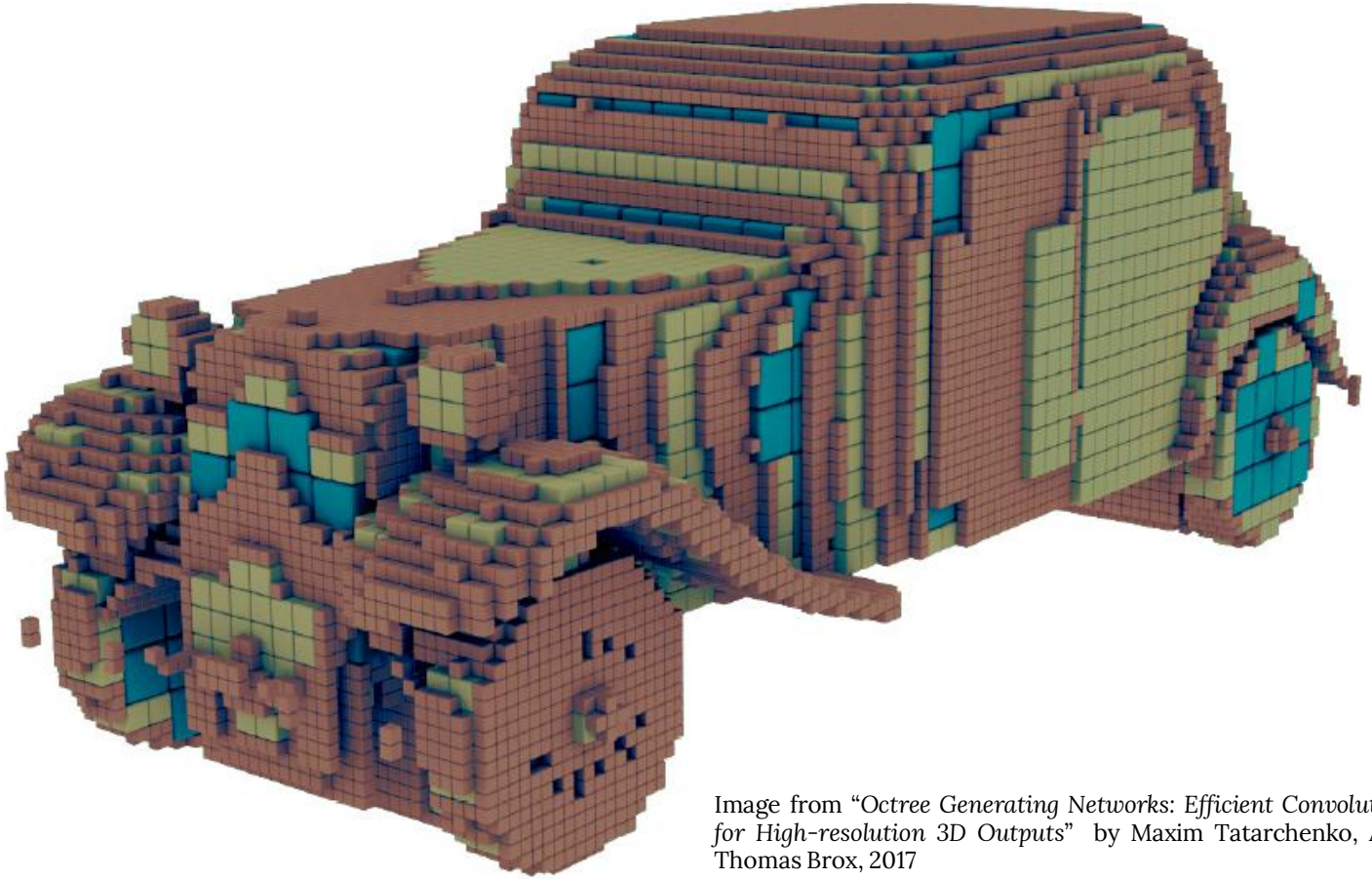


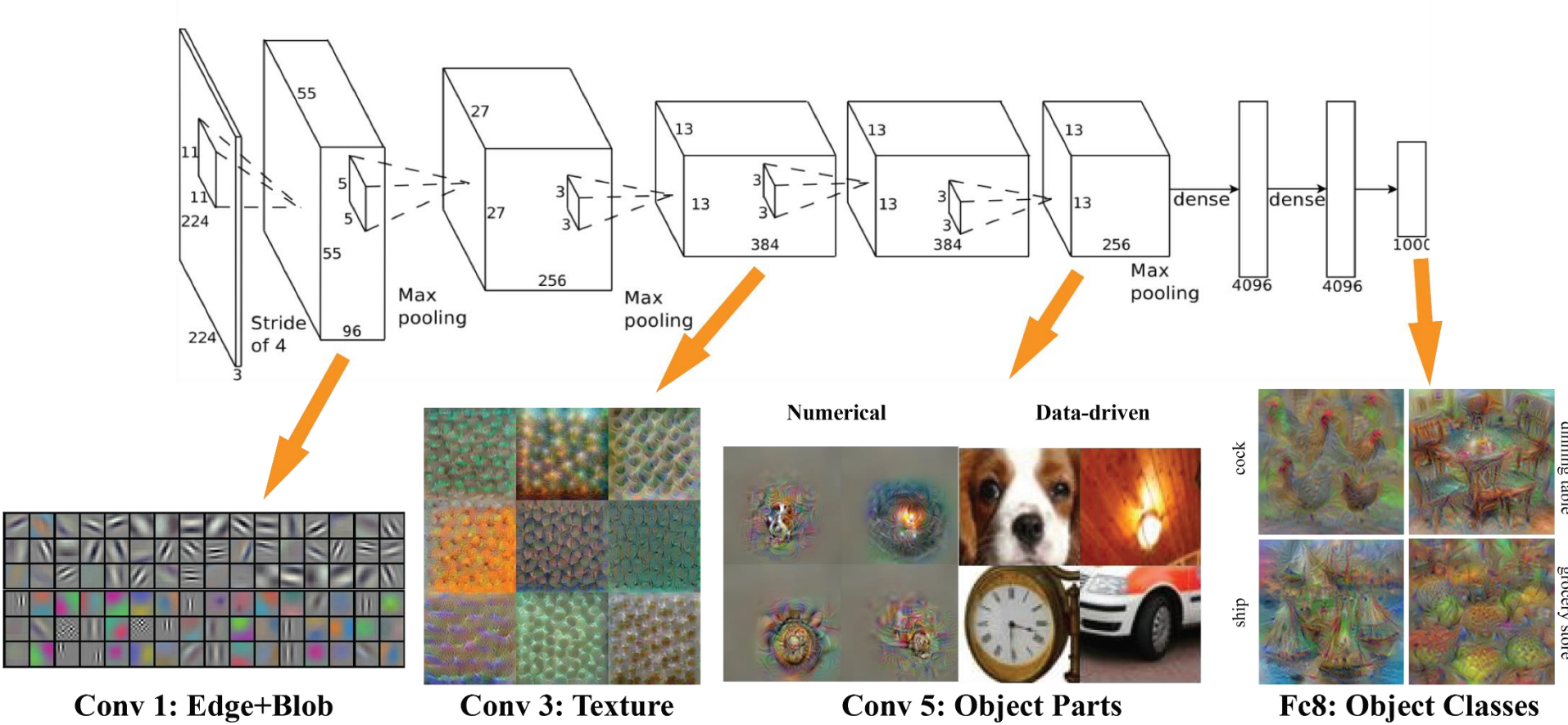
Image from "*Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs*" by Maxim Tatarchenko, Alexey Dosovitskiy, Thomas Brox, 2017

Mikhail Bessmeltsev

Background

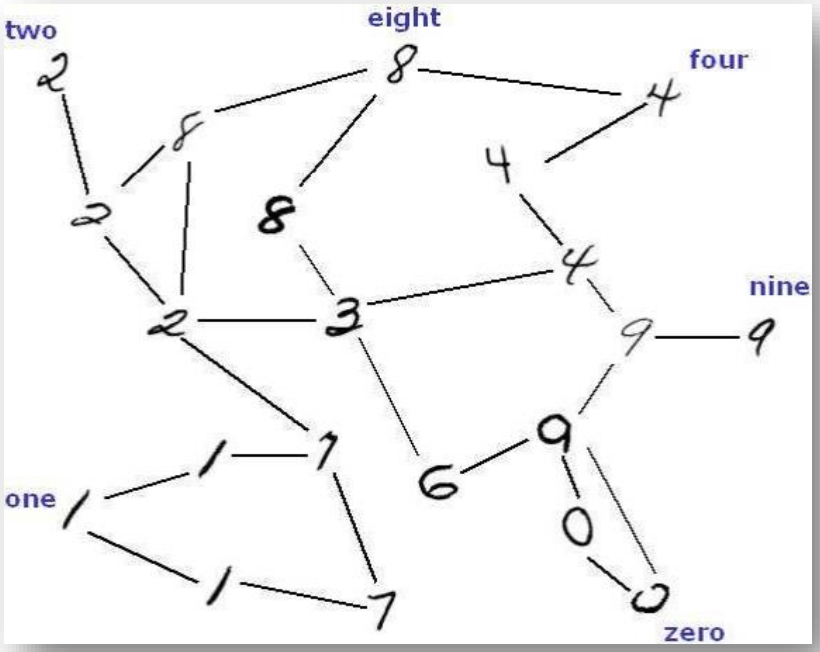
- Neural networks
- CNNs
- Autoencoders

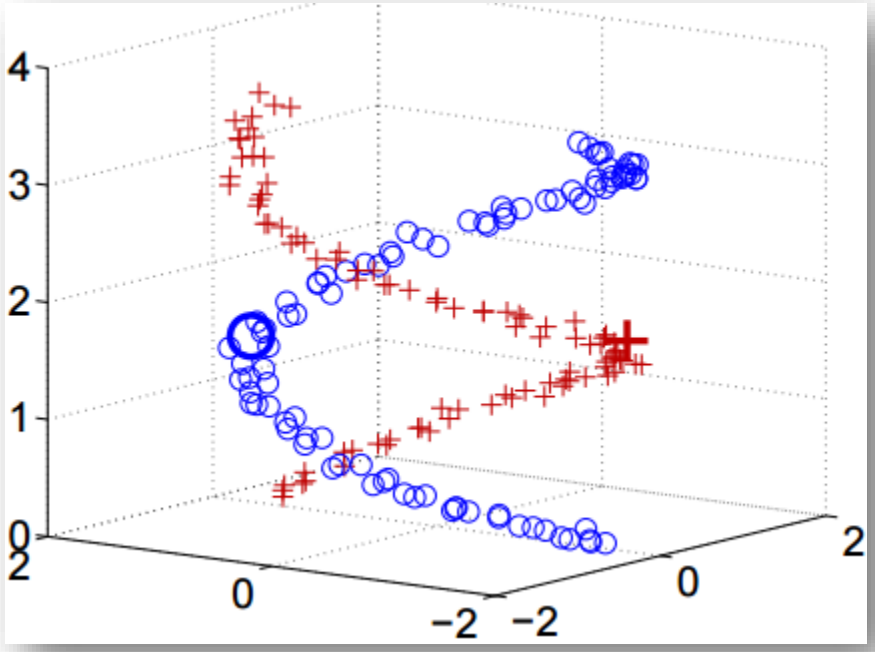
Background: standard CNNs



https://www.cc.gatech.edu/~hays/compvision/proj6/

Semi-Supervised Learning





“Semi-supervised learning using Gaussian fields and harmonic functions”
Zhu, Ghahramani, & Lafferty 2003

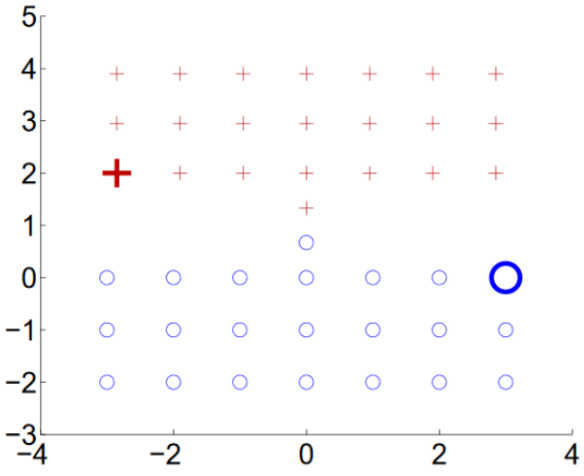
Semi-Supervised Technique

Given: ℓ labeled points $(x_1, y_1), \ldots, (x_\ell, y_\ell); y_i \in \{0, 1\}$

u unlabeled points $x_{\ell+1}, \ldots, x_{\ell+u}; \ell \ll u$

$$\min \frac{1}{2} \sum_{ij} w_{ij} (f(i) - f(j))^2$$

$$\text{s.t. } f(k) \text{ fixed } \forall k \leq \ell$$



Dirichlet energy → Linear system of equations (Poisson)

Related Method

- **Step 1:**
  Build k-NN graph

- **Step 2:**
  Compute p smallest Laplacian eigenvectors

- **Step 3:**
  Solve semi-supervised problem in subspace

"Using Manifold Structure for Partially Labelled Classification"
Belkin and Niyogi; NIPS 2002

Manifold Regularization

$$\text{Regularized learning:} \quad \arg\min_{f \in \mathcal{H}} \frac{1}{\ell} \sum_{i=1}^{\ell} V(f(x_i), y_i) + \gamma \|f\|^2$$

Loss function

Regularizer

Dirichlet energy

$$\|f\|_I^2 := \int \|\nabla f(x)\|^2 \, dx \approx f^\top L f$$

"Manifold Regularization:
A Geometric Framework for Learning from Labeled and Unlabeled Examples"
Belkin, Niyogi, and Sindhwani; JMLR 2006

Examples of Manifold Regularization

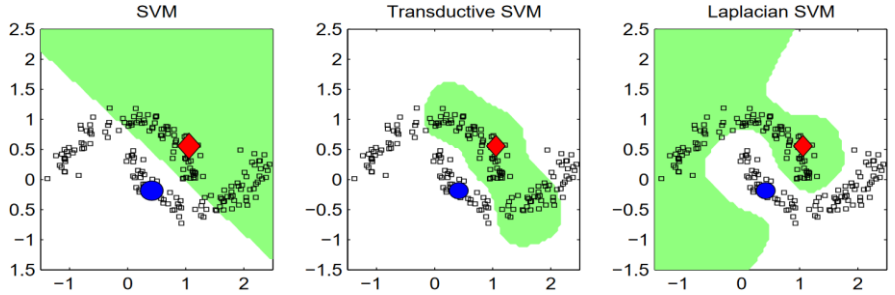
- Laplacian-regularized least squares (LapRLS)

$$\arg\min_{f\in\mathcal{H}} \frac{1}{\ell}\sum_{i=1}^{\ell}(f(x_i)-y_i)^2+\gamma\|f\|_I^2+\text{Other}[f]$$

- Laplacian support vector machine (LapSVM)

$$\arg\min_{f\in\mathcal{H}} \frac{1}{\ell}\sum_{i=1}^{\ell}\max(0,1-y_i f(x_i))+\gamma\|f\|_I^2+\text{Other}[f]$$

"On Manifold Regularization"
Belkin, Niyogi, Sindhwani; AISTATS 2005



Tasks

- 3D reconstruction
- Shape retrieval
- Shape completion
- Shape interpolation
- Shape segmentation

ShapeNet

ShapeNet is an ongoing effort to establish a richly-annotated, large-scale dataset of 3D shapes. We provide researchers around the world with this data to enable research in computer graphics, computer vision, robotics, and other related disciplines. ShapeNet is a collaborative effort between researchers at Princeton, Stanford and TTIC.

Browse Taxonomy | Search Models

Overview

ShapeNet consists of several subsets:

ShapeNetCore

ShapeNetCore is a subset of the full ShapeNet dataset with single clean 3D models and manually verified category and alignment annotations. It covers 55 common object categories with about 51,300 unique 3D models. The 12 object categories of PASCAL 3D+, a popular computer vision 3D benchmark dataset, are all covered by ShapeNetCore.

ShapeNetSem

ShapeNetSem is a smaller, more densely annotated subset consisting of 12,000 models spread over a broader set of 270 categories. In addition to manually verified category labels and consistent alignments, these models are annotated with real-world dimensions, estimates of their material composition at the category level, and estimates of their total volume and weight.

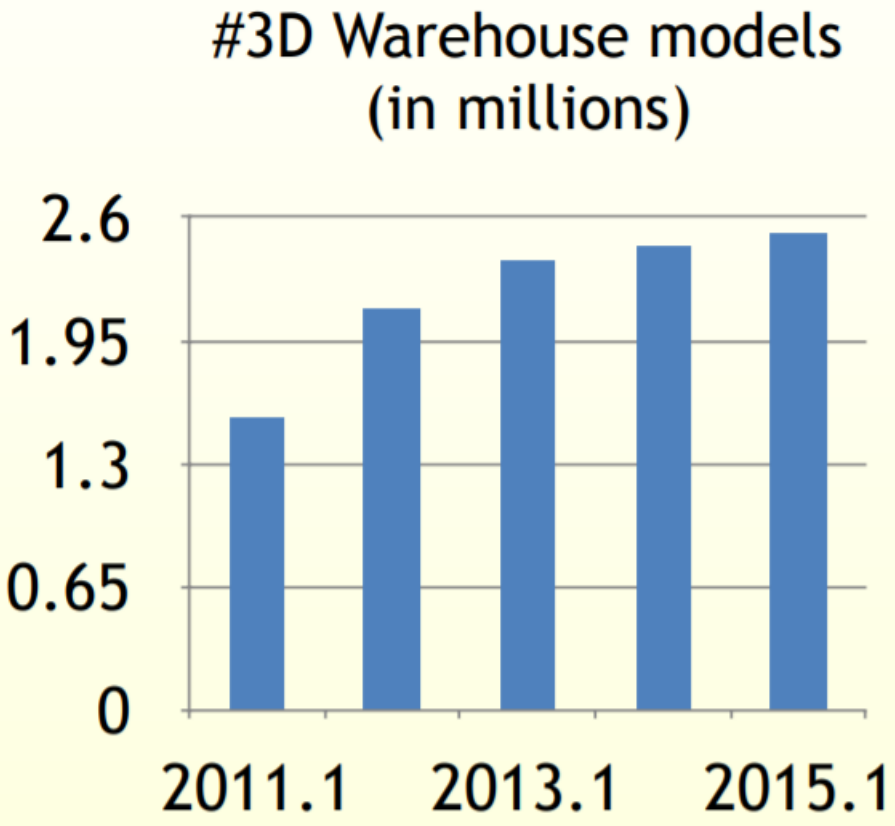
News

March, 2019 We are happy to announce the prelease of PartNet v0. PartNet provides fine-grained, hierarchical part annotations from ShapeNet..

Aug, 2017 We are organizing a ShapeNet challenge at ICCV 2017. More information available at https://shapenet.cs.stanford.edu/iccv17/.

Feb, 2017 We are organizing a large-scale 3D shape retrieval contest as part of the Eurographics 2017 3D Object Retrieval Workshop. More information available www.shapenet.org/shrec17.

Shapes versus Images



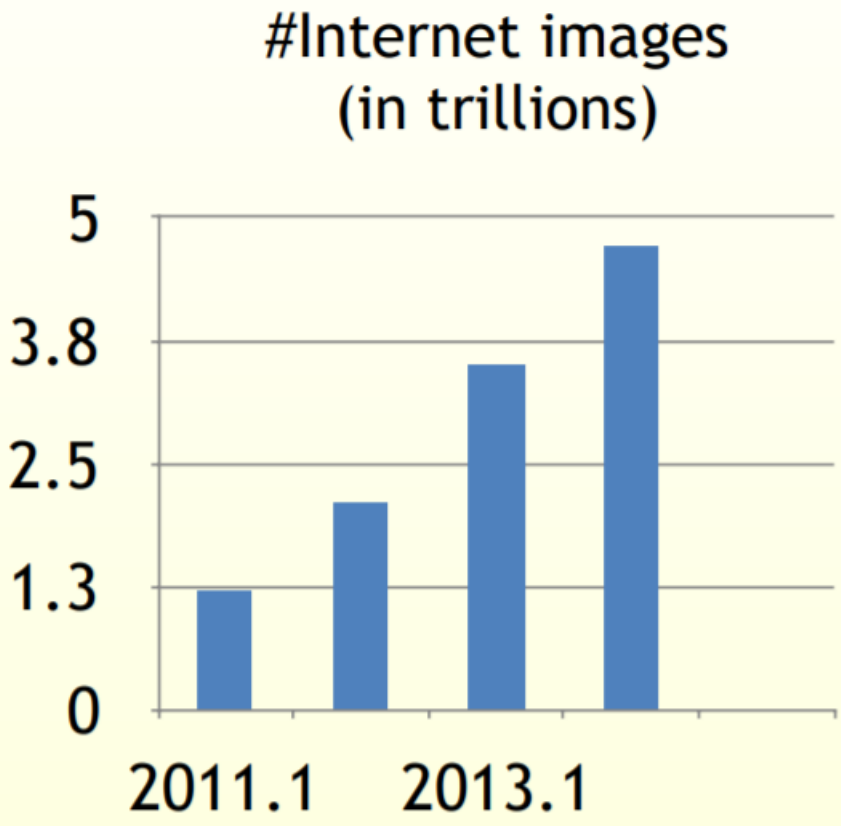
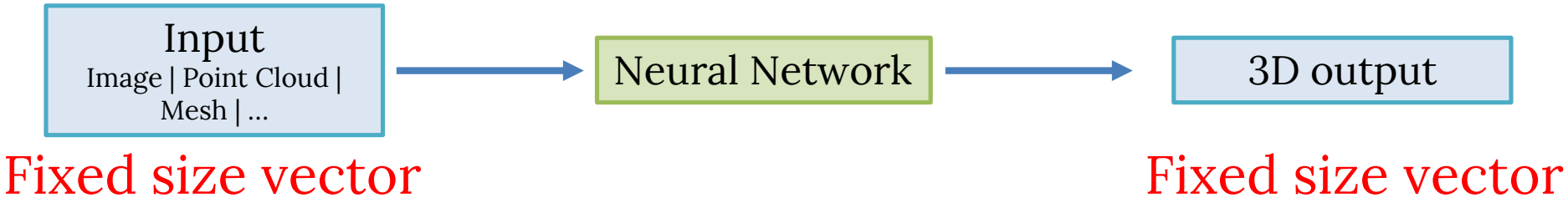
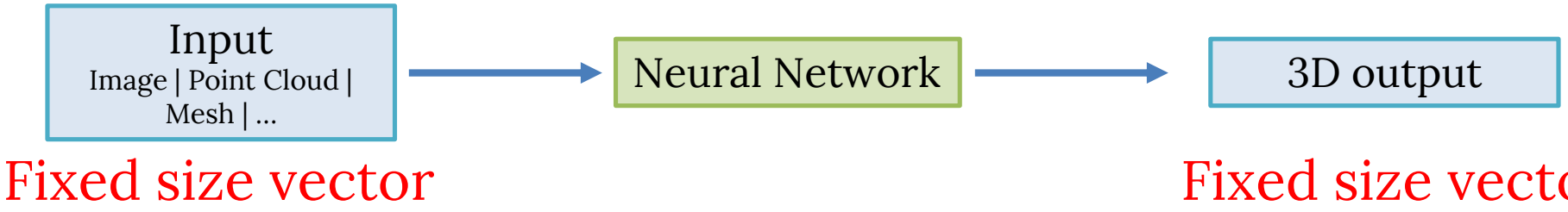


Image credit: Q. Huang, V. Koltun

Example task:
3D reconstruction



3D reconstruction



How to represent 3D geometry?

Latent representation

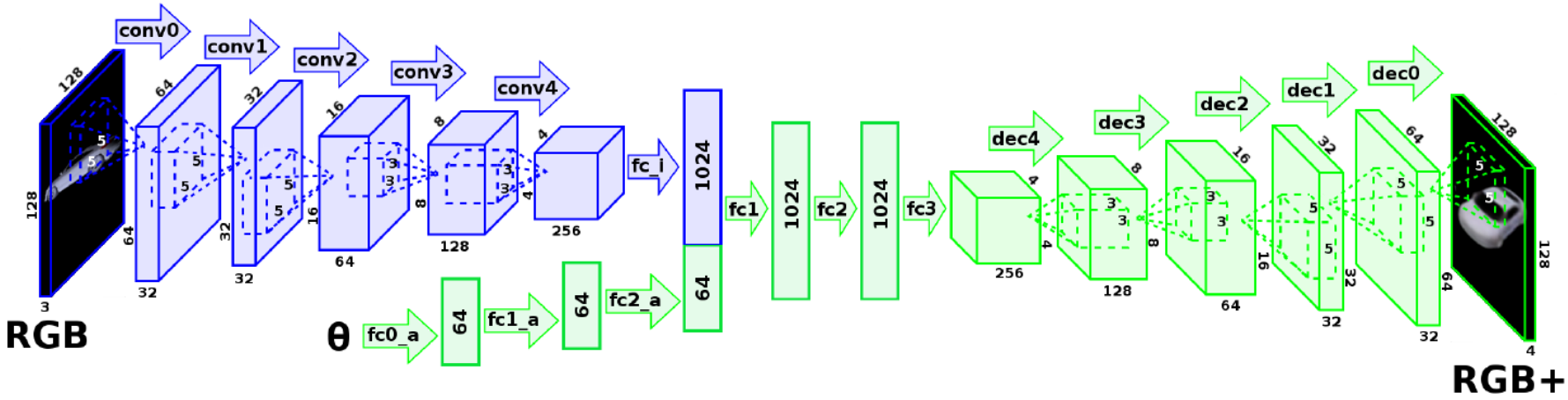


Fig. 2. The architecture of our network. The encoder (**blue**) turns an input image into an abstract 3D representation. The decoder (**green**) processes the angle, modifies the encoded hidden representation accordingly, and renders the final image together with the depth map.

"Multi-view 3D Models from Single Images with
a Convolutional Network" by Maxim Tatarchenko, Alexey Dosovitskiy, Thomas Brox, 2016

Simplest: depth map

3-D Depth Reconstruction from a Single Still Image

Ashutosh Saxena, Sung H. Chung, Andrew Y. Ng

Computer Science Department
Stanford University, Stanford, CA 94305
{asaxena,codedeft,ang}@cs.stanford.edu

Abstract

We consider the task of 3-d depth estimation from a single still image. We take a supervised learning approach to this problem, in which we begin by collecting a training set of monocular images (of unstructured indoor and outdoor environments which include forests, sidewalks, trees, buildings, etc.) and their corresponding ground-truth depthmaps. Then, we apply supervised learning to predict the value of the depthmap as a function of the image. Depth estimation is a challenging problem, since local features alone are insufficient to estimate depth at a point, and one needs to consider the global context of the image. Our model uses a hierarchical, multiscale Markov Random Field (MRF) that incorporates multiscale local- and global-image features, and models the depths and the relation between depths at different points in the image. We show that, even on unstructured scenes, our algorithm is frequently able to recover fairly accurate depthmaps. We further propose a model that incorporates both monocular cues and stereo (triangulation) cues,

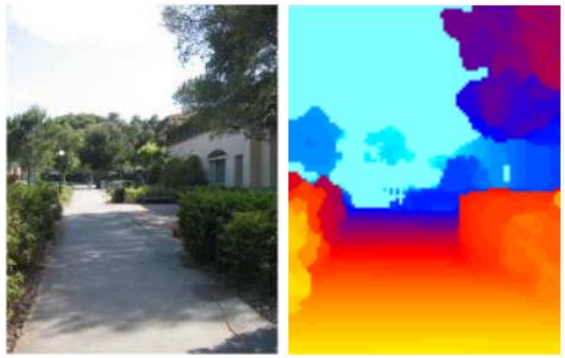


Figure 1: (a) A single still image, and (b) the corresponding (ground-truth) depthmap. Colors in the depthmap indicate estimated distances from the camera.

stereo and monocular cues, most work on depth estimation has focused on stereovision.

Depth estimation from a *single* still image is a difficult task, since depth typically remains ambiguous given only local image features. Thus, our algorithms must take into account the global structure of the image, as well as

3-D Depth Reconstruction from a Single Still Image

Ashutosh Saxena, Sung H. Chung, Andrew Y. Ng

Computer Science Department
Stanford University, Stanford, CA 94305
{asaxena,codedeft,ang}@cs.stanford.edu

Figure 2: The convolutional filters used for texture energies and gradients. The first nine are 3x3 Laws' masks. The last six are the oriented edge detectors spaced at 30^0 intervals. The nine Laws' masks are used to perform local averaging, edge detection and spot detection.

Multi-scale

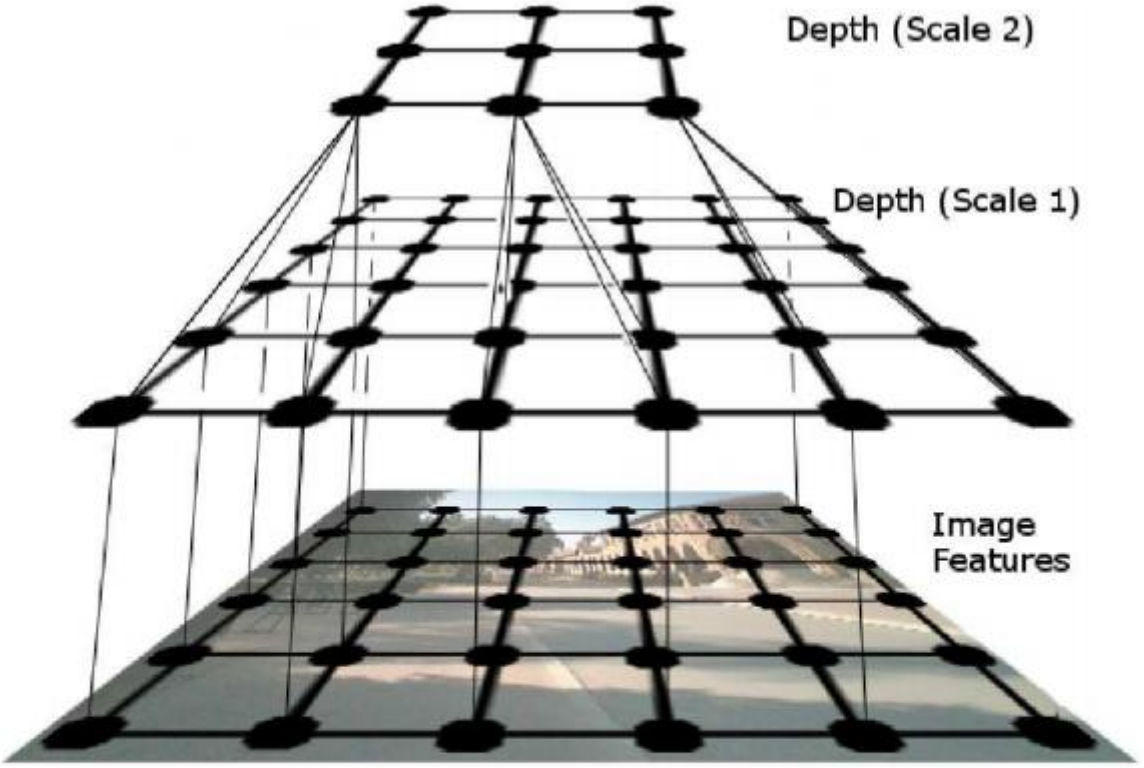


Figure 4: The multiscale MRF model for modeling relation between features and depths, relation between depths at same scale, and relation between depths at different scales. (Only 2 out of 3 scales, and a subset of the edges, are shown.)

Gaussian Markov Random Field

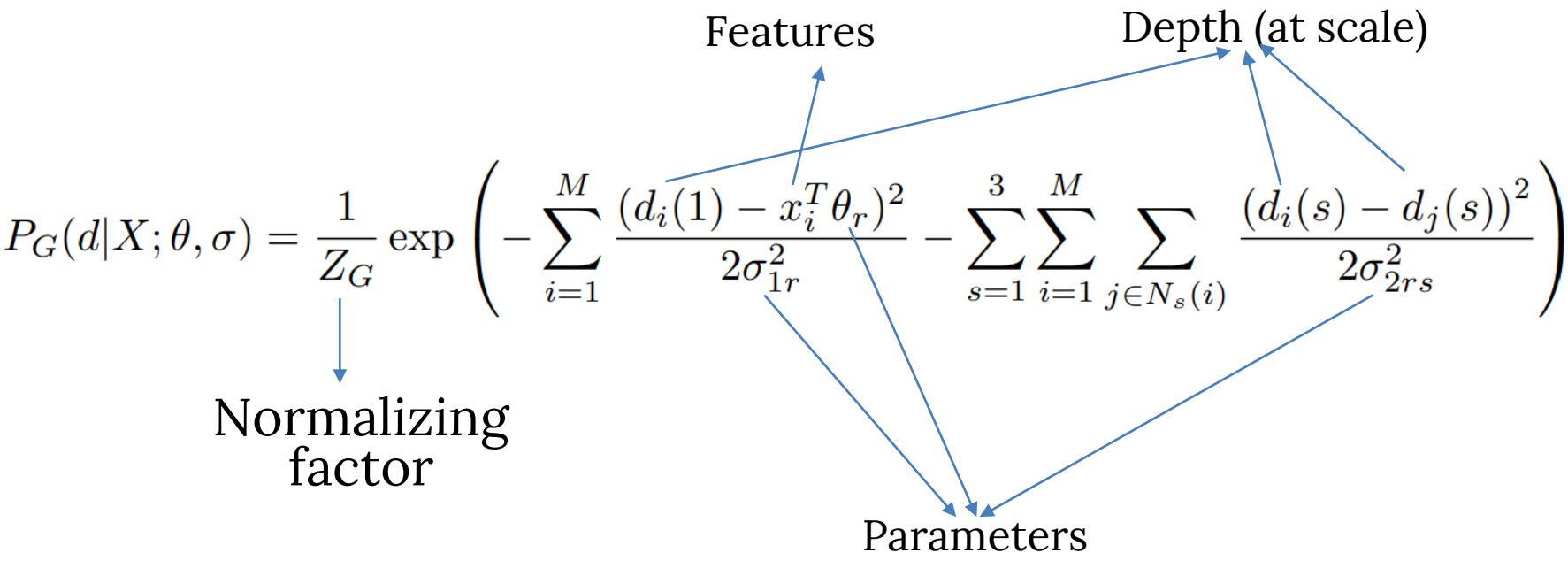
Want: depth ≈ linear function(features)

$$d_i(1) \approx x_i^T \theta$$

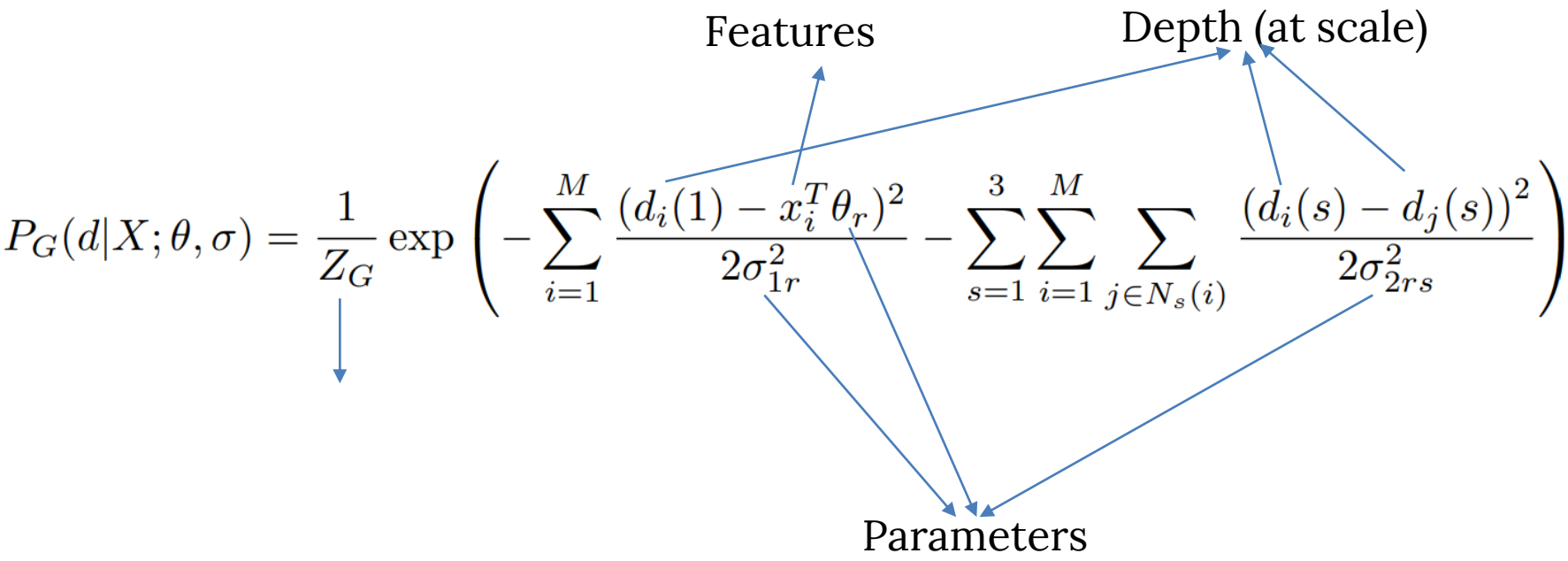
My depth ≈ depth of my neighbors

$$d_i(s) \approx d_j(s)$$

Gaussian Markov Random Field



Gaussian Markov Random Field



Find by minimizing negative log-likelihood of data:

$$\min_{\theta,\sigma} -\log P_G(d|X;\theta,\sigma)$$

Where to get depth maps?

Much easier: stereo images!

https://www.flickr.com/photos/ntenny/46048092055

Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue

Ravi Garg, Vijay Kumar B G, Gustavo Carneiro, Ian Reid

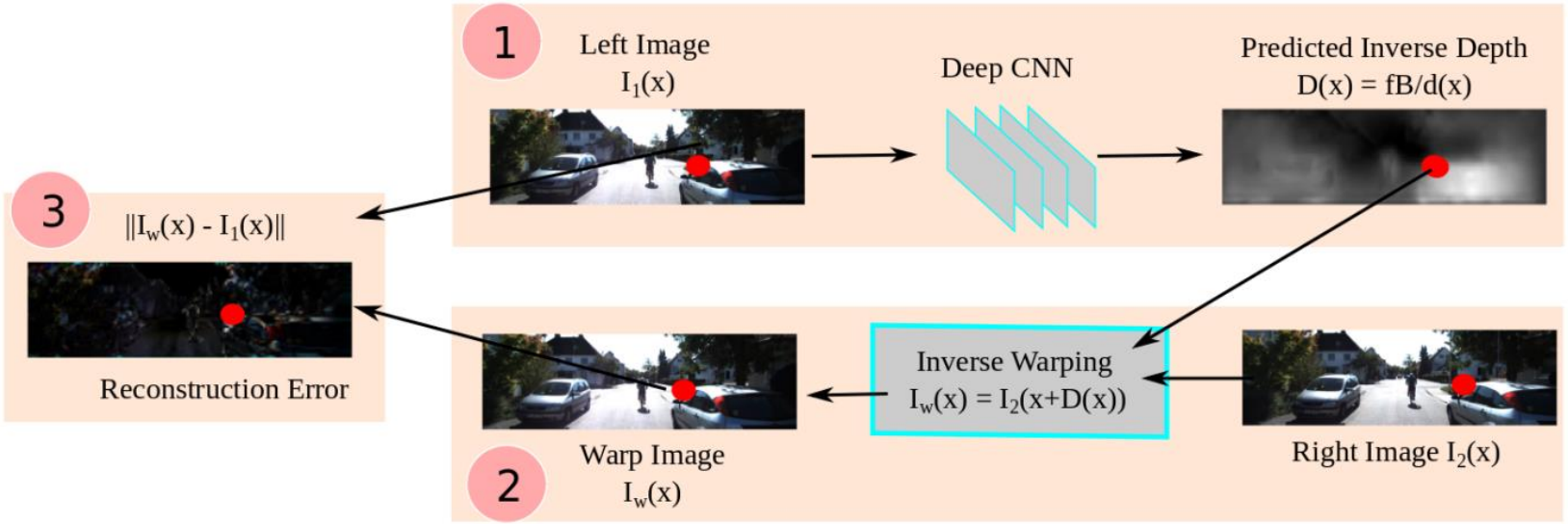
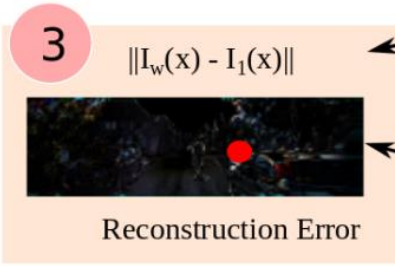


Fig. 1. We propose a stereopsis based auto-encoder setup: the encoder (Part 1) is a traditional convolutional neural network with stacked convolutions and pooling layers (See Figure 2) and maps the left image (I_1) of the rectified stereo pair into its depth map. Our decoder (Part 2) explicitly forces the encoder output to be disparities (scaled inverse depth) by synthesizing a backward warp image (I_w) by moving pixels from right image I_2 along the scan-line. We use the reconstructed output I_w to be matched with the encoder input (Part 3) via a simple loss. For end-to-end training, we minimize the reconstruction loss with a simple smoothness prior on disparities which deals with the

Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue

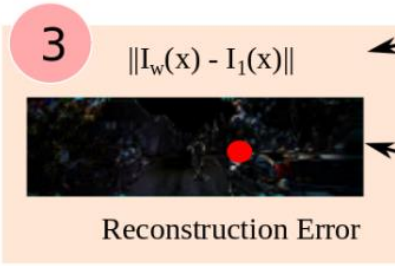
Ravi Garg, Vijay Kumar B G, Gustavo Carneiro, Ian Reid



What if there's a patch of constant color?

Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue

Ravi Garg, Vijay Kumar B G, Gustavo Carneiro, Ian Reid



What if there's a patch of constant color?

Solution: add Dirichlet energy as a regularizer $\|\nabla D(x)\|^2$

Caveat: disparity is low-res

But we need to calculate $I_2(x + D(x))$

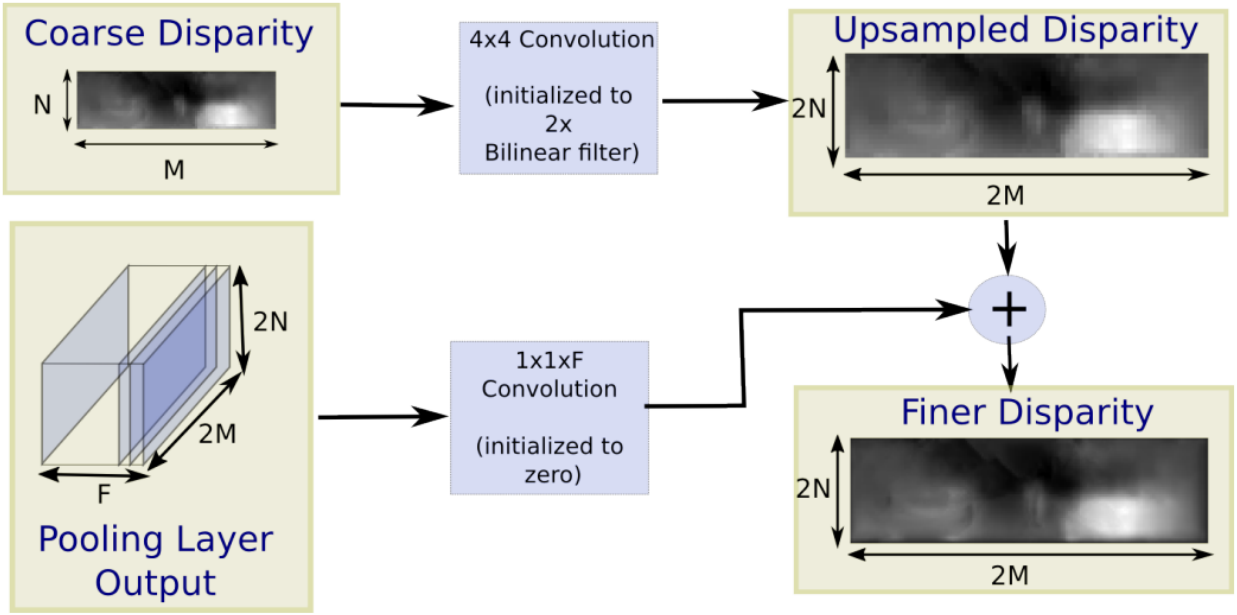


Fig. 2. Coarse-to-fine stereo with CNN with results on a sample validation instance: We adapt the convolution based upsampling architecture proposed in [26] to mimic the coarse-to-fine stereo estimations. Our upsampling filter is initialized with simple bilinear interpolation kernel and we initialize the corresponding pooling layer contribution by setting both bias and 1x1 convolution filter to be zero. The figure shows how features coming from previous layers of the CNN (L3) combined with finer resolution loss function generate better depthmaps at 44×172 from our bilinear upsampled initial estimate of coarser prediction at 22×76.

Hi-res Depth Maps

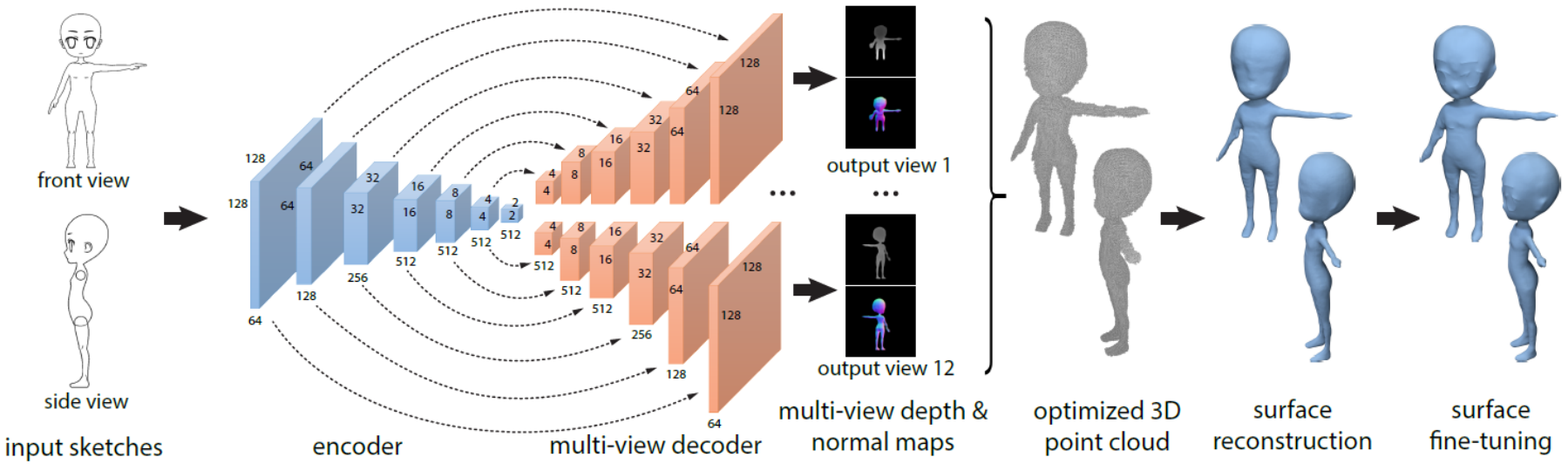


Figure 1. Our method takes line drawings as input and converts them into multi-view surface depth and normals maps from several output viewpoints via an encoder-multi-view-decoder architecture. The maps are fused into a coherent 3D point cloud, which is then converted into a surface mesh. Finally, the mesh can be further fine-tuned to match the input drawings more precisely through geometric deformations.

Predict: depth + normal map → point cloud → mesh

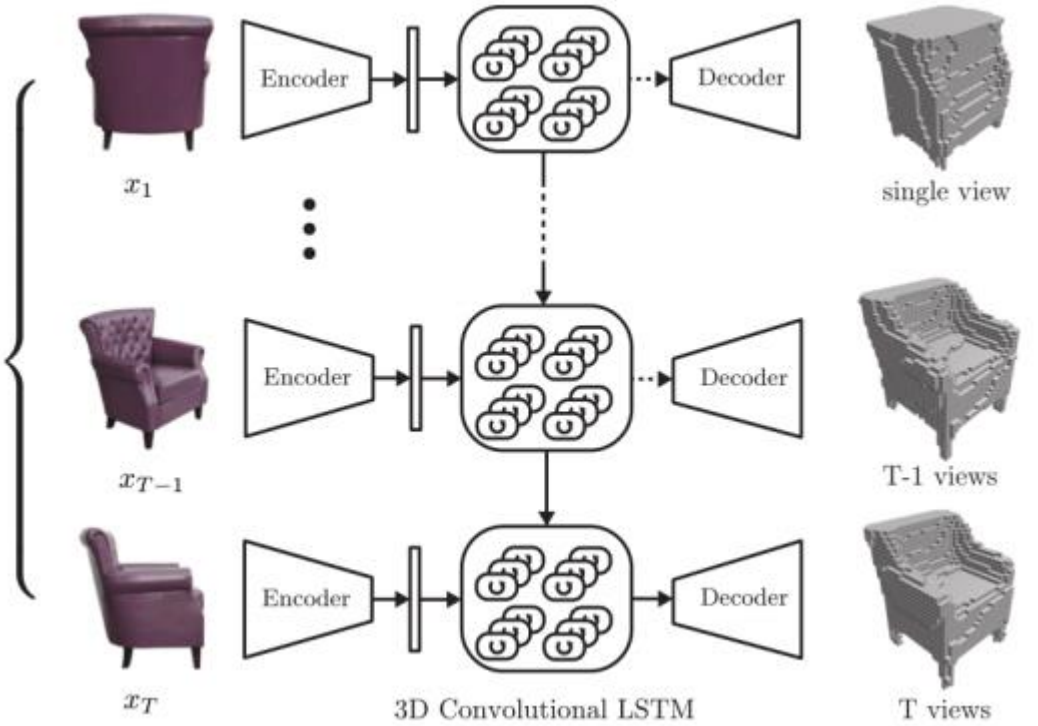
"3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks" by Liu et al., 2017

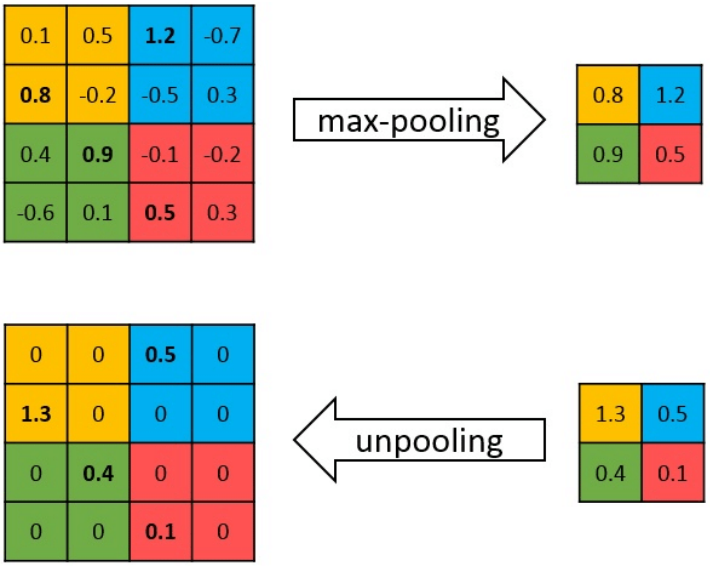
Voxelization

3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction

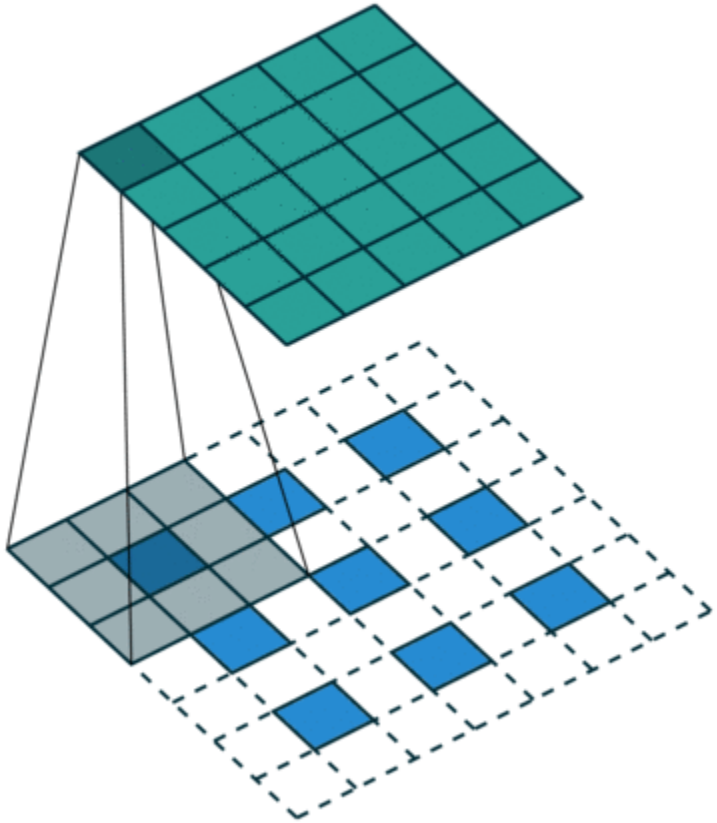
Christopher B. Choy Danfei Xu* JunYoung Gwak*
Kevin Chen Silvio Savarese

Stanford University
{chrischoy, danfei, jgwak, kchen92, ssilvio}@stanford.edu





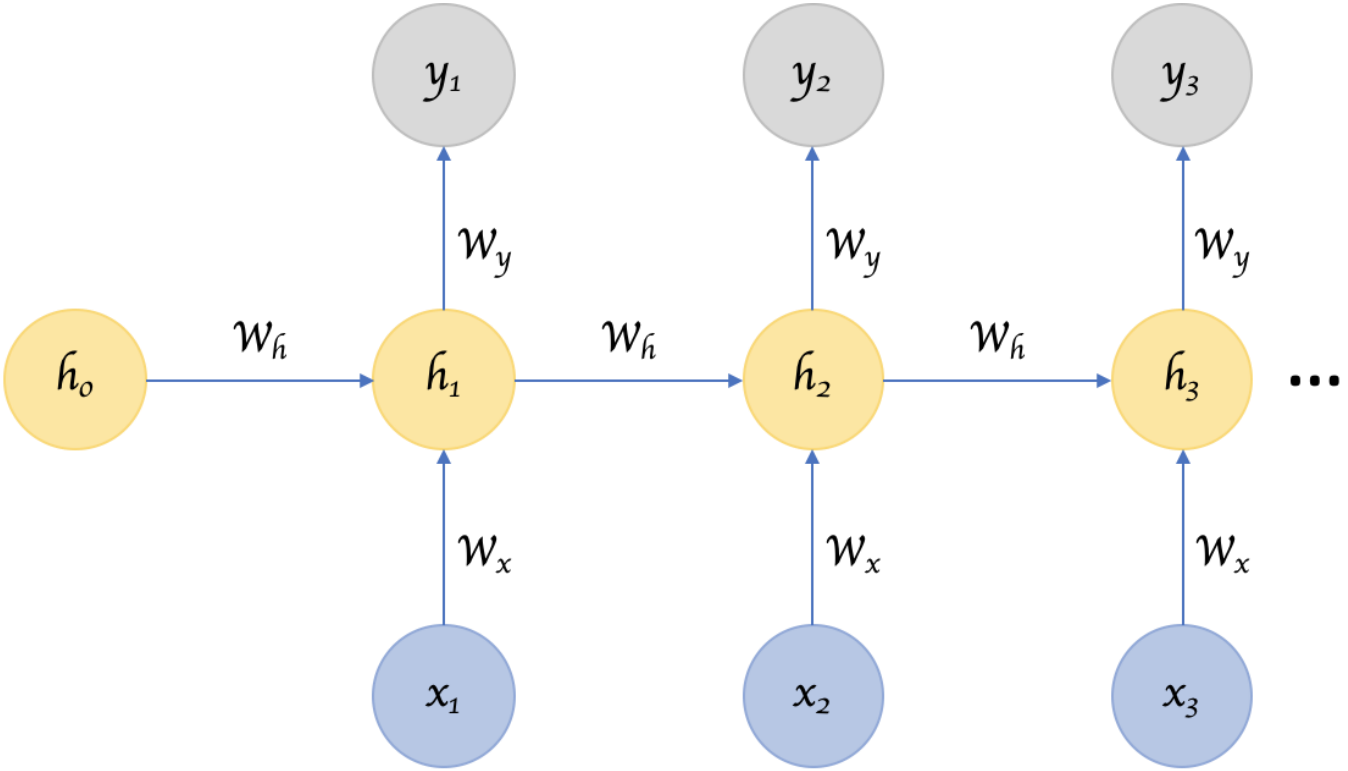
Unpooling



Deconvolution

Accumulate information from all views

Idea: use a recurrent neural network (RNN)

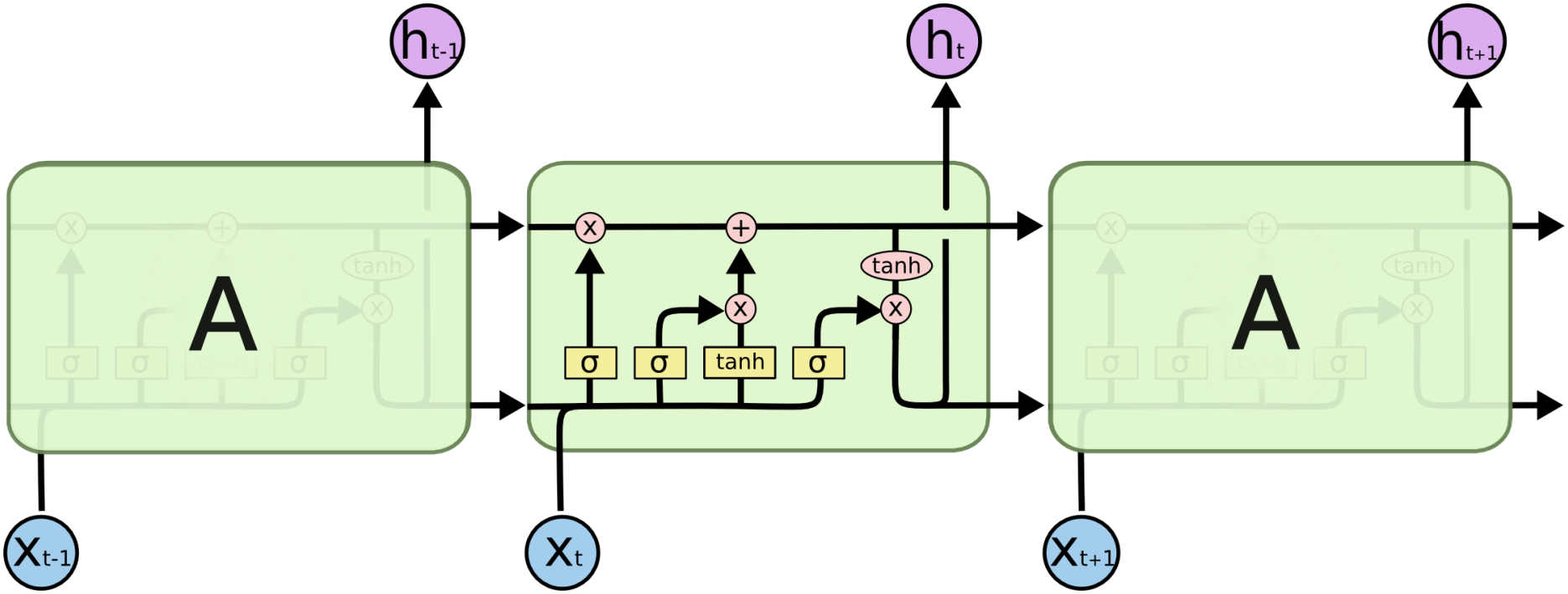


https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Hidden state allows to memorize info

Accumulate information from all views

Idea: use a ~~(RNN)~~ LSTM

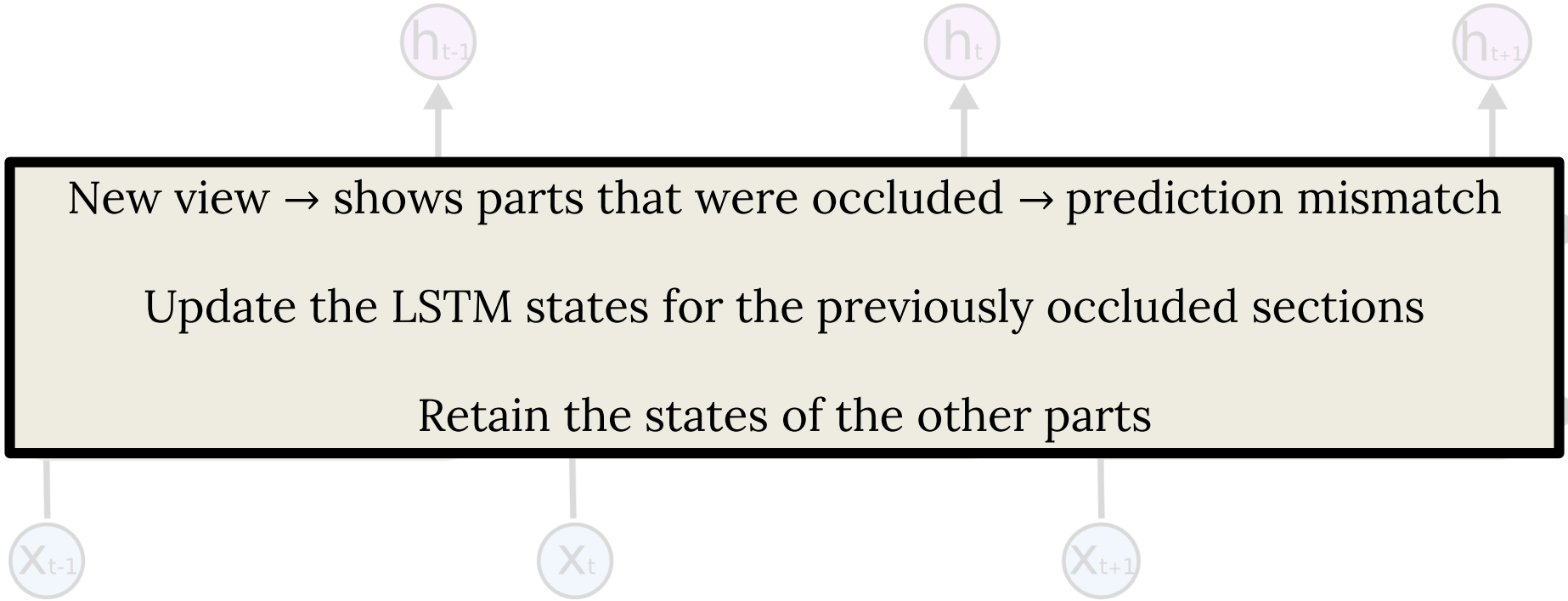


https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0

Hidden state allows to memorize info

Accumulate information from all views

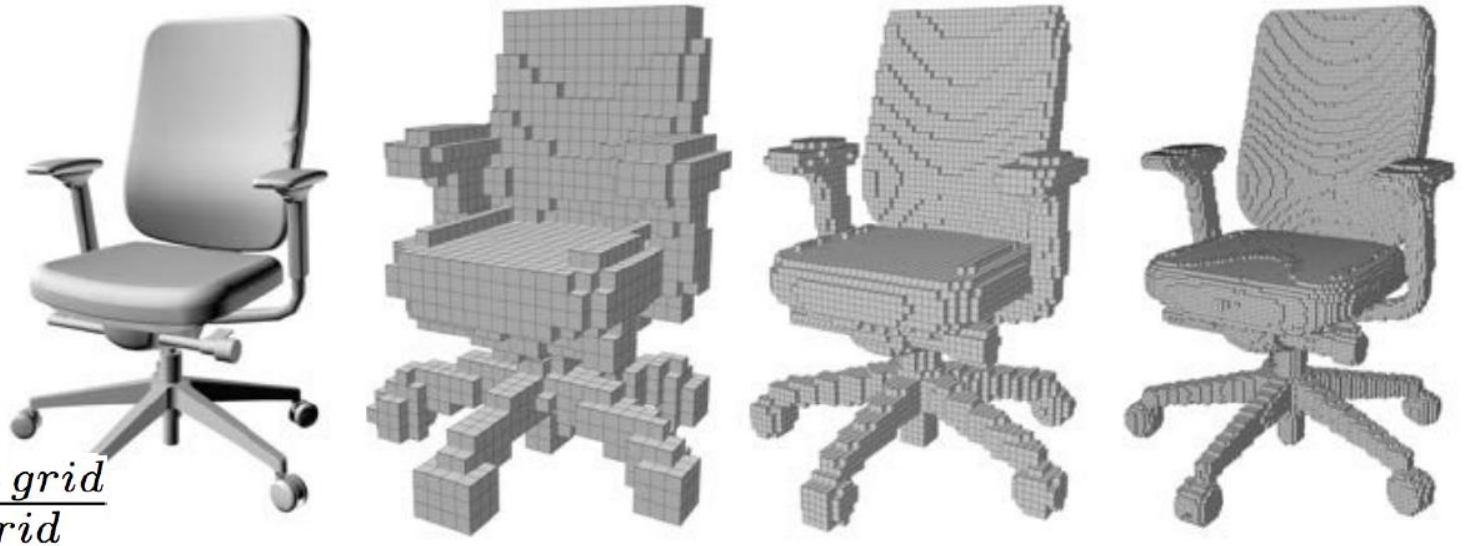
Idea: use a ~~(RNN)~~ LSTM



https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0

Hidden state allows to memorize info

Voxelization: issue



$$\frac{\#occupied\ grid}{\#total\ grid}$$

Occupancy:	10.41%	5.09%	2.41%
Resolution:	32	64	128

Mesh as deformed template

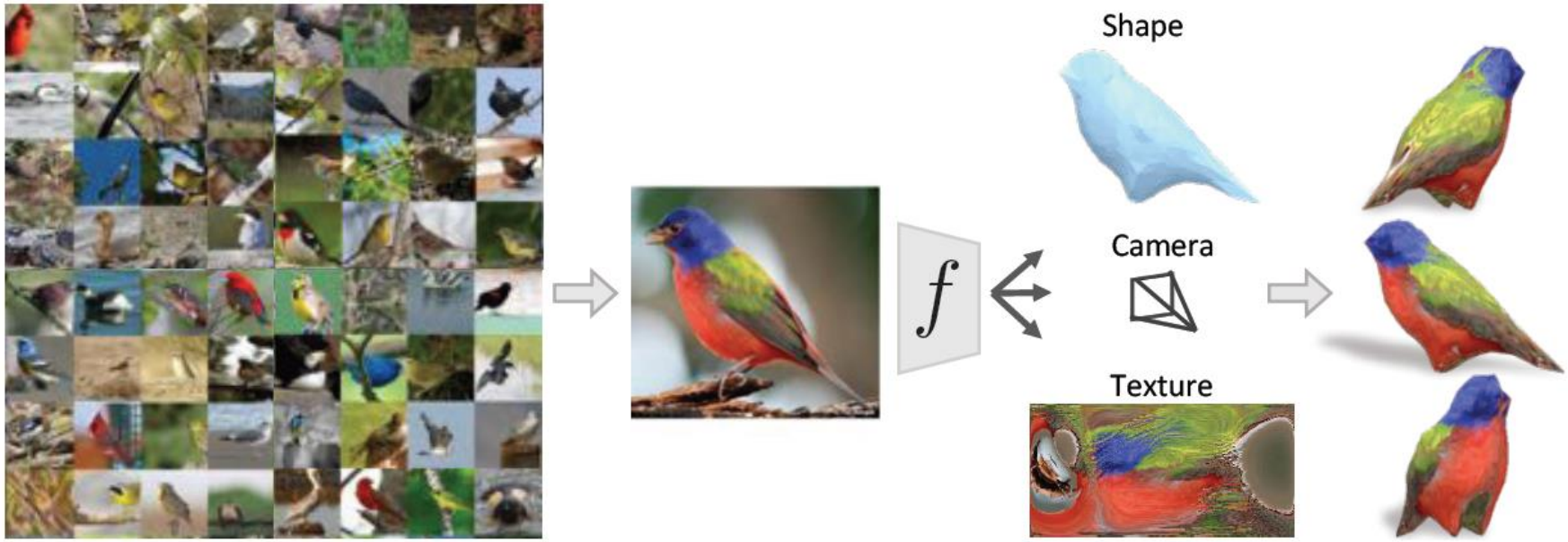


Fig. 1: Given an annotated image collection of an object category, we learn a predictor f that can map a novel image I to its 3D shape, camera pose, and texture.

"Learning Category-Specific Mesh Reconstruction from Image Collections" by Kanazawa et al., 2018

Mesh as deformed template

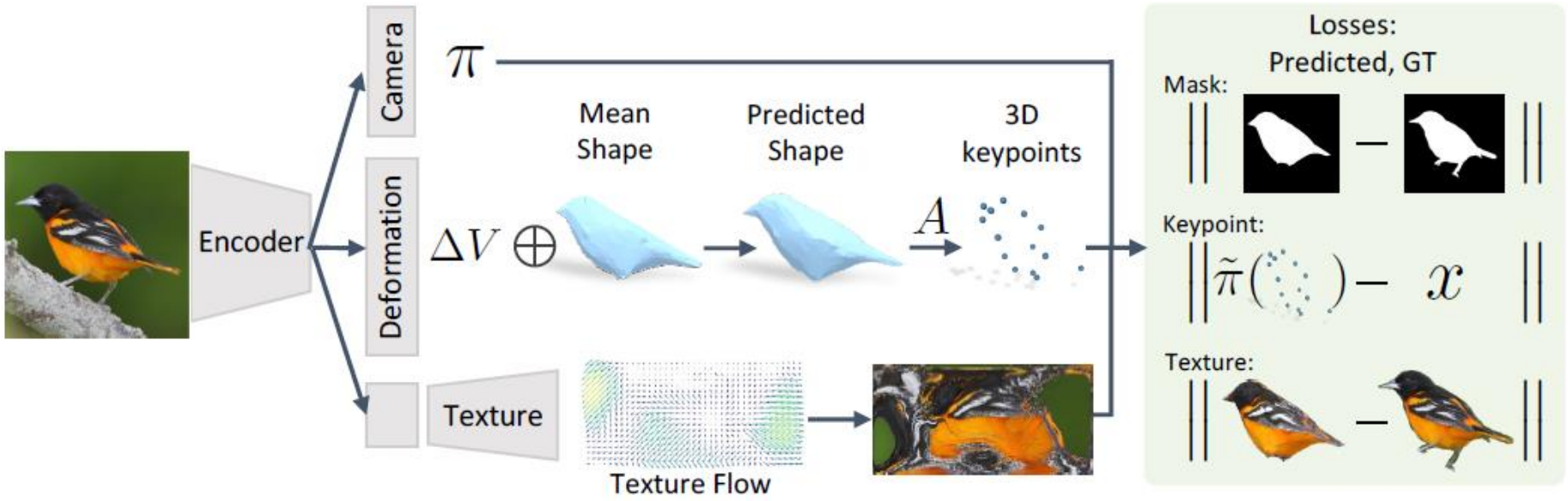


Fig. 2: **Overview of the proposed framework.** An image I is passed through a convolutional encoder to a latent representation that is shared by modules that estimate the camera pose, deformation and texture parameters. Deformation is an offset to the learned mean shape, which when added yield instance specific shapes in a canonical coordinate frame. We also learn correspondences between the mesh vertices and the semantic keypoints. Texture is parameterized as an UV image, which we predict through texture flow (see Section 2.3). The objective is to minimize the distance between the rendered mask, keypoints and textured rendering with the corresponding ground truth annotations. We do not require ground truth 3D shapes or multi-view cues for training.

Differentiable renderers

Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction

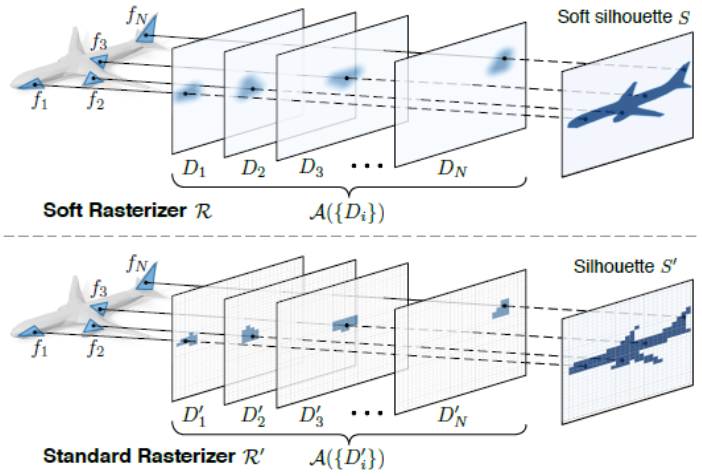
Shichen Liu[1,2], Weikai Chen[1], Tianye Li[1,2], and Hao Li[1,2,3]

[1]USC Institute for Creative Technologies
[2]University of Southern California
[3]Pinscreen

{lshichen, wechen, tli}@ict.usc.edu hao@hao-li.com

Abstract

Rendering is the process of generating 2D images from 3D assets, simulated in a virtual environment, typically with a graphics pipeline. By inverting such renderer, one can think of a learning approach to predict a 3D shape from an input image. However, standard rendering pipelines involve a fundamental discretization step called rasterization, which prevents the rendering process to be differentiable, hence able to be learned. We present the first non-parametric and truly differentiable rasterizer based on silhouettes. Our method enables unsupervised learning for high-quality 3D mesh reconstruction from a single image.



Differentiable renderers

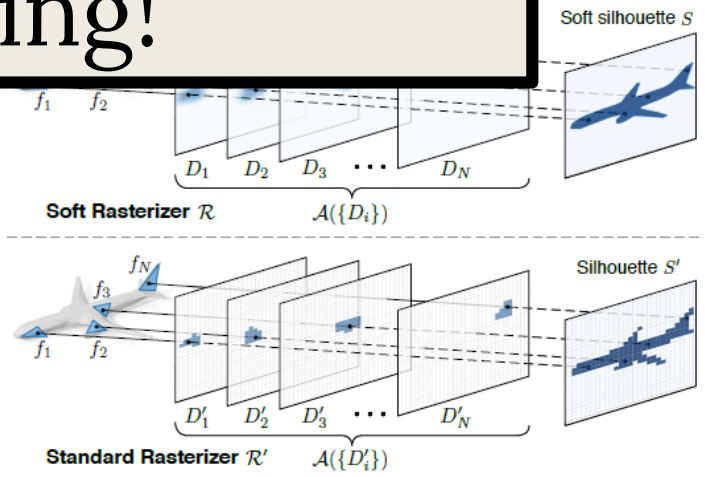
Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction

Shichen Liu[1,2], Weikai Chen[1], Tianye Li[1,2], and Hao Li[1,2,3]

[1]USC Institute for Creative Technologies

Enables **unsupervised** learning!

Rendering is the process of generating 2D images from 3D assets, simulated in a virtual environment, typically with a graphics pipeline. By inverting such renderer, one can think of a learning approach to predict a 3D shape from an input image. However, standard rendering pipelines involve a fundamental discretization step called rasterization, which prevents the rendering process to be differentiable, hence able to be learned. We present the first non-parametric and truly differentiable rasterizer based on silhouettes. Our method enables unsupervised learning for high-quality 3D mesh reconstruction from a single image.

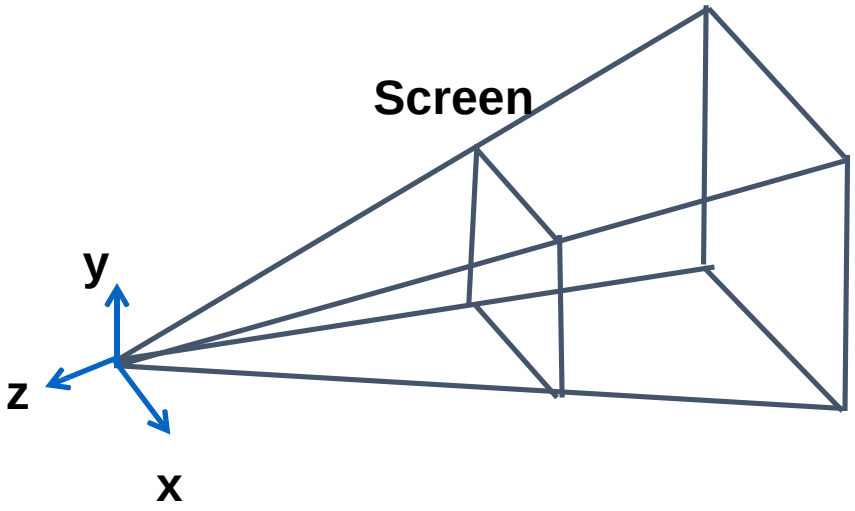


Rendering

= projection + rasterization

Projection

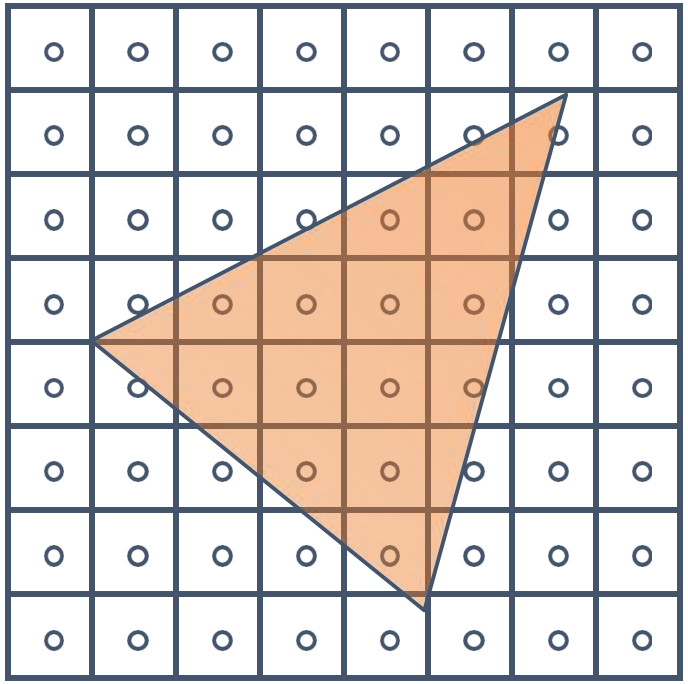
Linear operation



$$P(x, y, z, 1) = \begin{pmatrix} 1 & 0 & 0 & 0 & x \\ 0 & 1 & 0 & 0 & y \\ 0 & 0 & 1 & 0 & z \\ 0 & 0 & 1/d & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix}$$

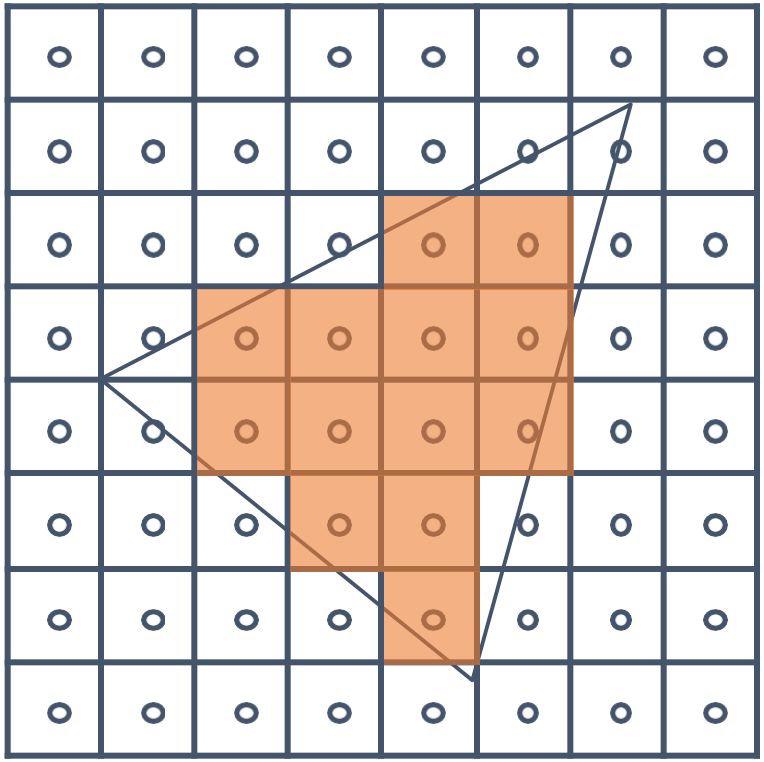
Rasterization

screen is discrete



Rasterization

Discrete/non-differentiable operation



Soft rasterizer

Relax/blur the discretization

~ distance between pixel and triangle?

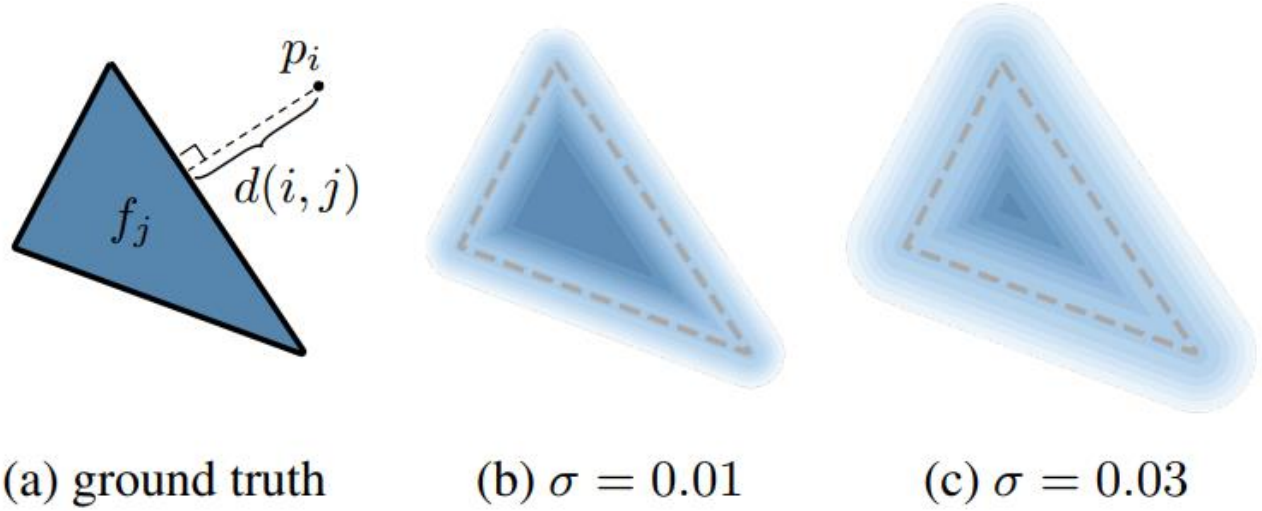
$$D_j^i = sigmoid\left(\delta_{ij} \cdot \frac{d^2(i,j)}{\sigma}\right)$$

+1, if i^{th} pixel is inside j^{th} triangle

Soft rasterizer

Relax/blur the discretization

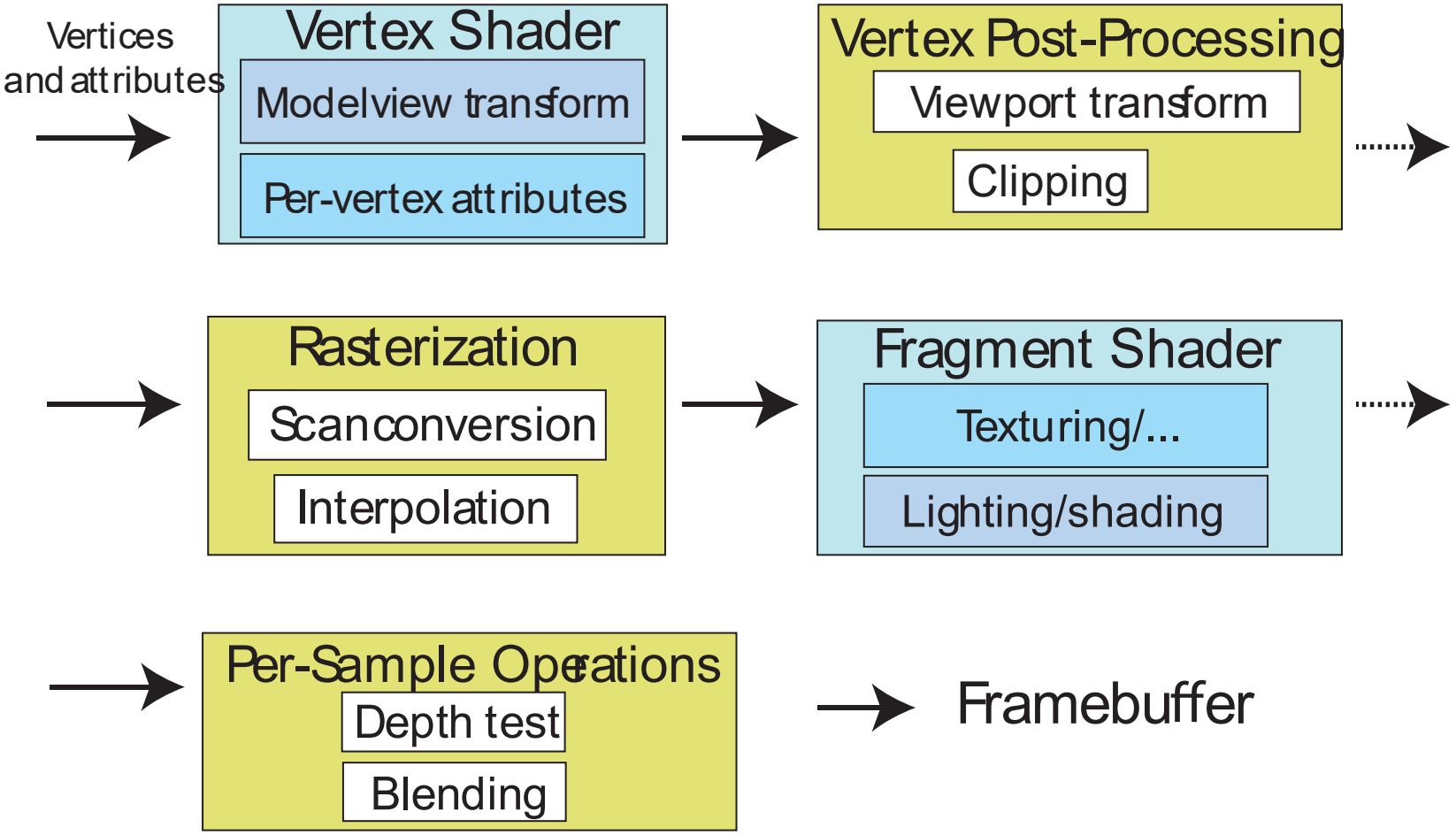
~ distance between pixel and triangle?



(a) ground truth (b) $\sigma = 0.01$ (c) $\sigma = 0.03$

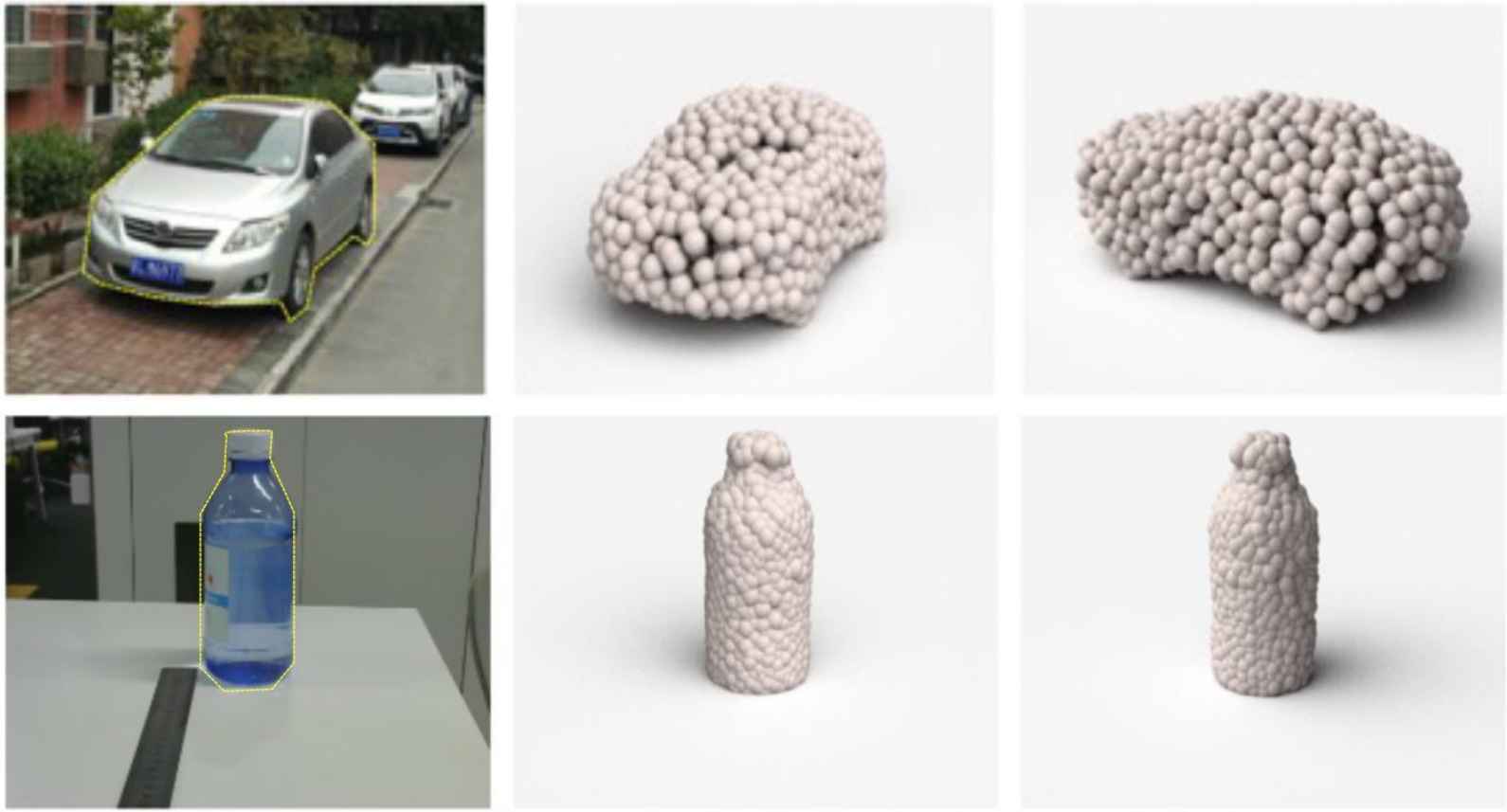
Figure 3: Example probability maps of a single triangle. (a): definition of pixel-to-triangle distance; (b) and (c): probability maps generated with different σ.

Reality check: colors et al.



Sample pipeline

Reconstructing a point cloud



Input

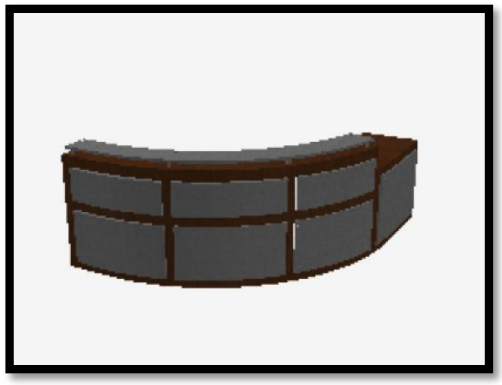
Reconstructed 3D point cloud

A Point Set Generation Network for 3D Object Reconstruction from a Single Image by Fan, Su, Guibas, CVPR 2017

Sample pipeline

Reconstructing a point cloud

TESTING



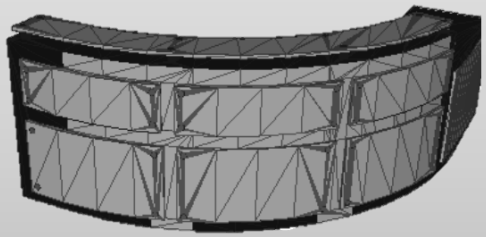
Deep neural network

Point Cloud

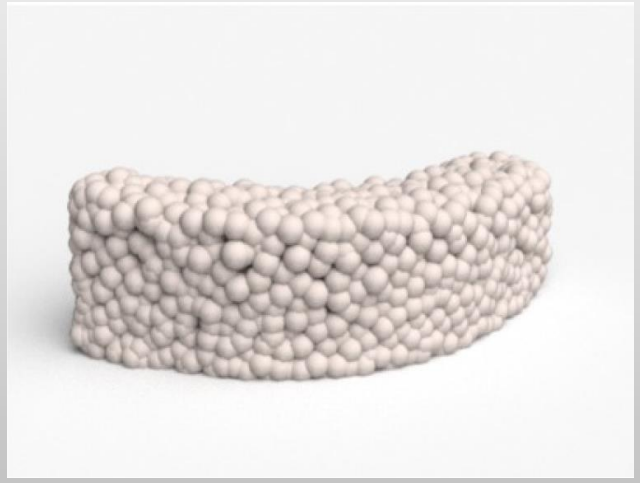
(x_1, y_1, z_1)
(x_2, y_2, z_2)
...
(x_n, y_n, z_n)

TRAINING

Render



Sample



(x'_1, y'_1, z'_1)
(x'_2, y'_2, z'_2)
...
(x'_n, y'_n, z'_n)

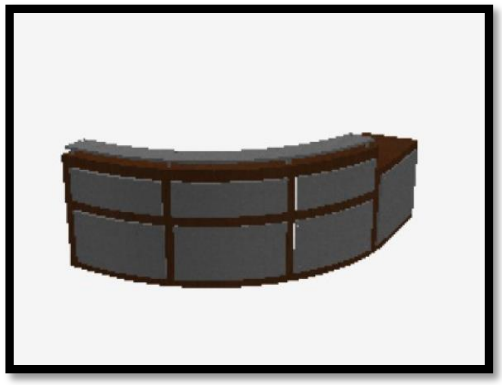
Ground Truth Point Cloud

A Point Set Generation Network for 3D Object Reconstruction from a Single Image by Fan, Su, Guibas, CVPR 2017

Sample pipeline

Reconstructing a point cloud

TESTING



Deep neural network

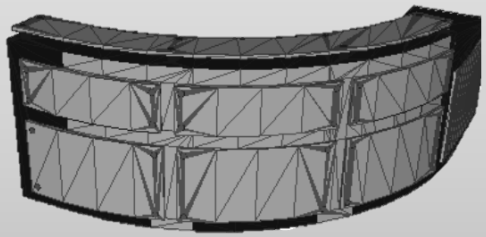
Point Cloud

(x_1, y_1, z_1)
(x_2, y_2, z_2)
...
(x_n, y_n, z_n)

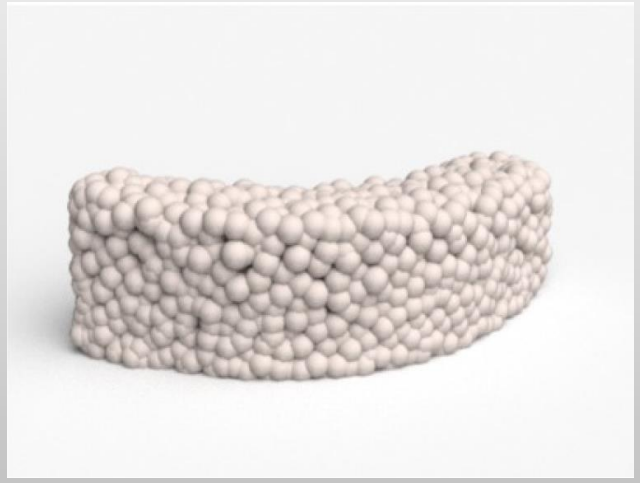
???

TRAINING

Render



Sample



(x'_1, y'_1, z'_1)
(x'_2, y'_2, z'_2)
...
(x'_n, y'_n, z'_n)

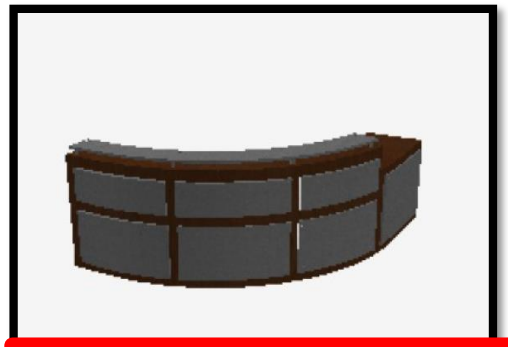
Ground Truth Point Cloud

A Point Set Generation Network for 3D Object Reconstruction from a Single Image by Fan, Su, Guibas, CVPR 2017

Sample pipeline

Reconstructing a point cloud

TESTING



Deep neural network

Point Cloud

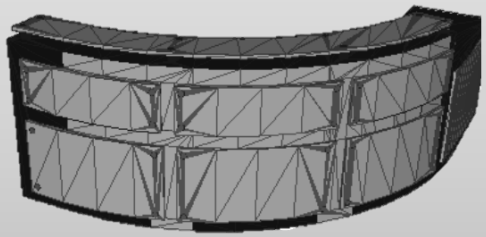
(x_1, y_1, z_1)
(x_2, y_2, z_2)
...
(x_n, y_n, z_n)

How to choose error function/loss?

???

TRAINING

Render



Sample

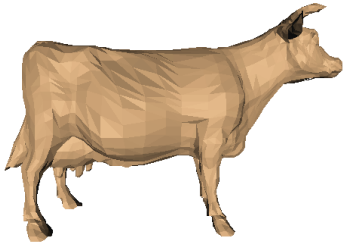
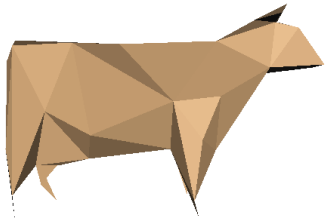
(x'_1, y'_1, z'_1)
(x'_2, y'_2, z'_2)
...
(x'_n, y'_n, z'_n)

Ground Truth Point Cloud

A Point Set Generation Network for 3D Object Reconstruction from a Single Image by Fan, Su, Guibas, CVPR 2017

Recall:

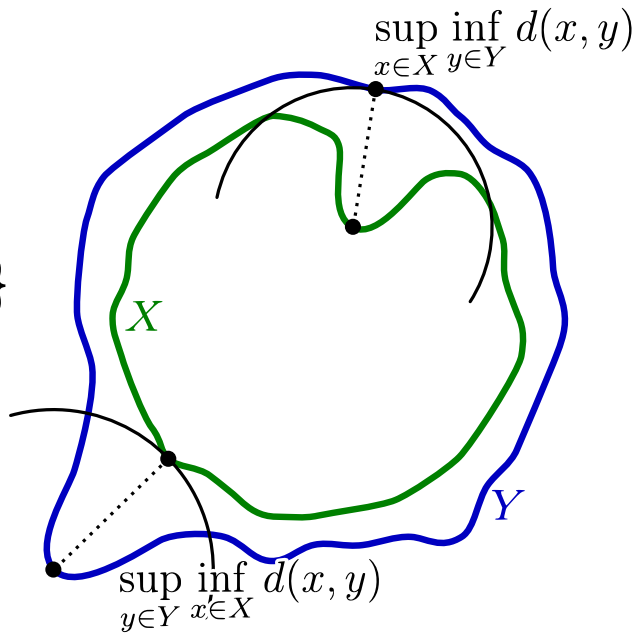
Measuring error

$d($  ,  $) = ?$

How much did we distort the shape?

Option: Hausdorff distance

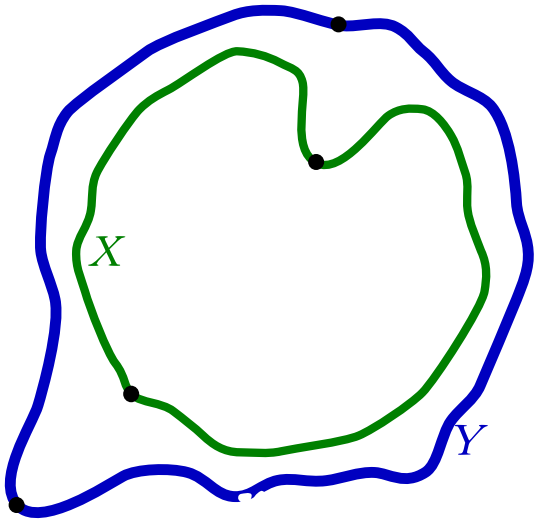
$$d_{H(x,y)} = \max\{\sup_{x \in X} \inf_{y \in Y} d(x,y), \sup_{y \in Y} \inf_{x \in X} d(x,y)\}$$



Loss: Chamfer

Given two surfaces X, Y:

$$d_{Chamfer}(X,Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 + \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2$$



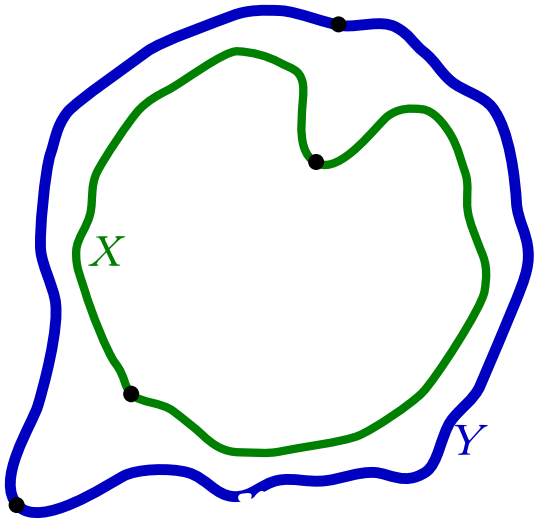
Loss: Wasserstein

aka Earth Mover's Distance (EMD)

$$d_{EMD}(X, Y) = \min_{\phi: X \to Y} \sum_{X} \|x - \phi(x)\|$$

ϕ is a bijection

Find optimal strategy of transporting
blue *sand* into green *hole*



Loss: Wasserstein

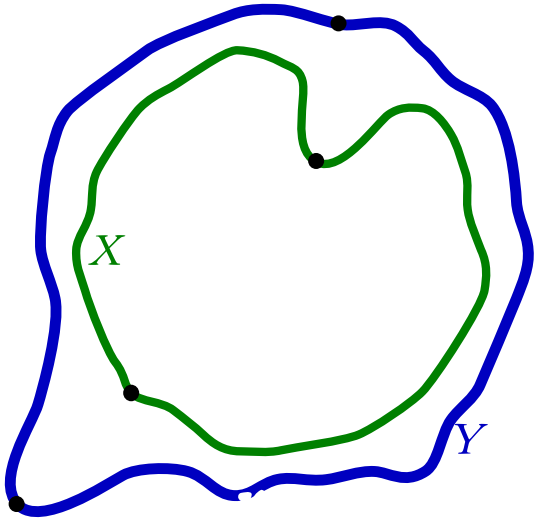
aka Earth Mover's Distance (EMD)

Doesn't have to be Euclidean

$$d_{EMD}(X, Y) = \min_{\phi: X \to Y} \sum_{X} \|x - \phi(x)\|$$

ϕ is a bijection

Find optimal strategy of transporting blue *sand* into green *hole*



Deformable Meshes

AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation

Thibault Groueix[1*], Matthew Fisher[2], Vladimir G. Kim[2], Bryan C. Russell[2], Mathieu Aubry[1]
[1]LIGM (UMR 8049), École des Ponts, UPE, [2]Adobe Research

http://imagine.enpc.fr/~groueixt/atlasnet/

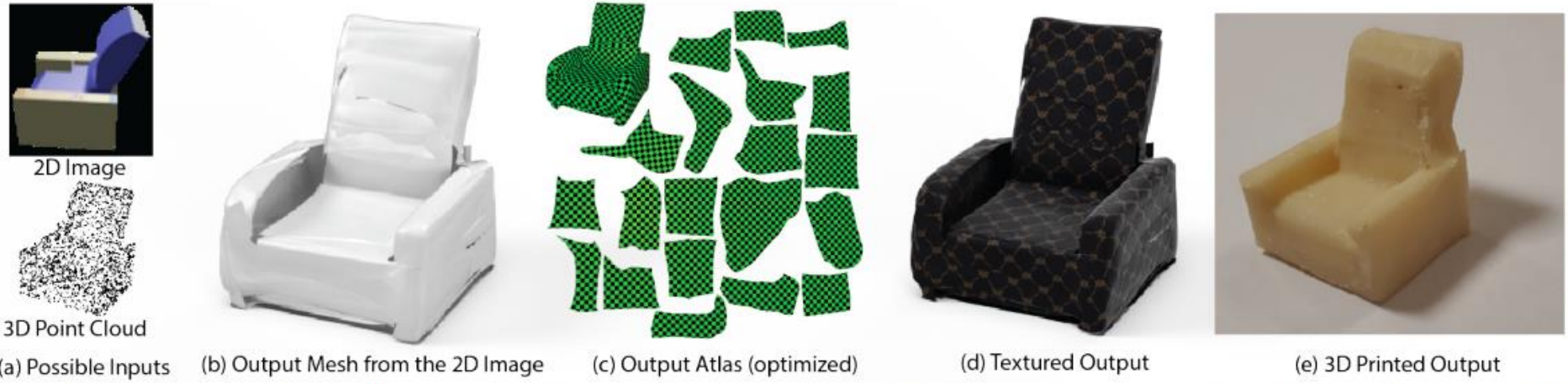


Figure 1. Given input as either a 2D image or a 3D point cloud (a), we automatically generate a corresponding 3D mesh (b) and its atlas parameterization (c). We can use the recovered mesh and atlas to apply texture to the output shape (d) as well as 3D print the results (e).

AtlasNet

Learn $\varphi_\theta : (0,1)^2 \rightarrow S_\theta \subset \mathbb{R}^3$

$$\min_\theta \mathcal{L}(S_\theta, S_{target}) + \lambda \mathcal{R}(\theta)$$

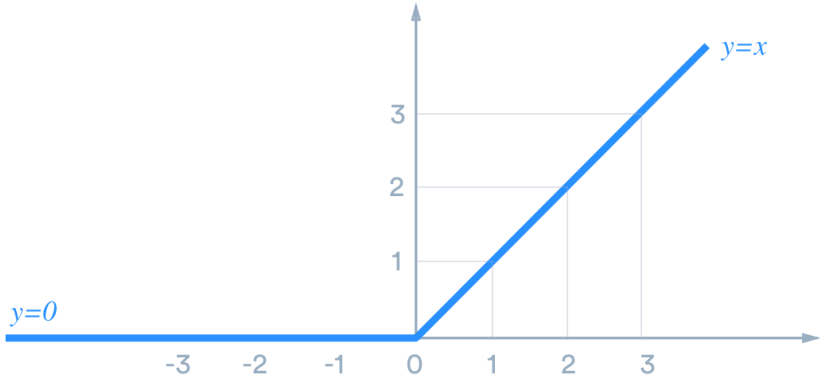
Regularization

Chamfer or Earth-Mover distance

How to represent functions φ_θ?

AtlasNet

$$\text{Learn } \varphi_\theta : (0,1)^2 \to S_\theta \subset \mathbb{R}^3$$
$$\min_\theta \mathcal{L}\left(S_\theta, S_{target}\right) + \lambda \mathcal{R}(\theta)$$



ReLU nonlinearities → Locally affine transformation
→ S_θ is a (locally) 2-manifold

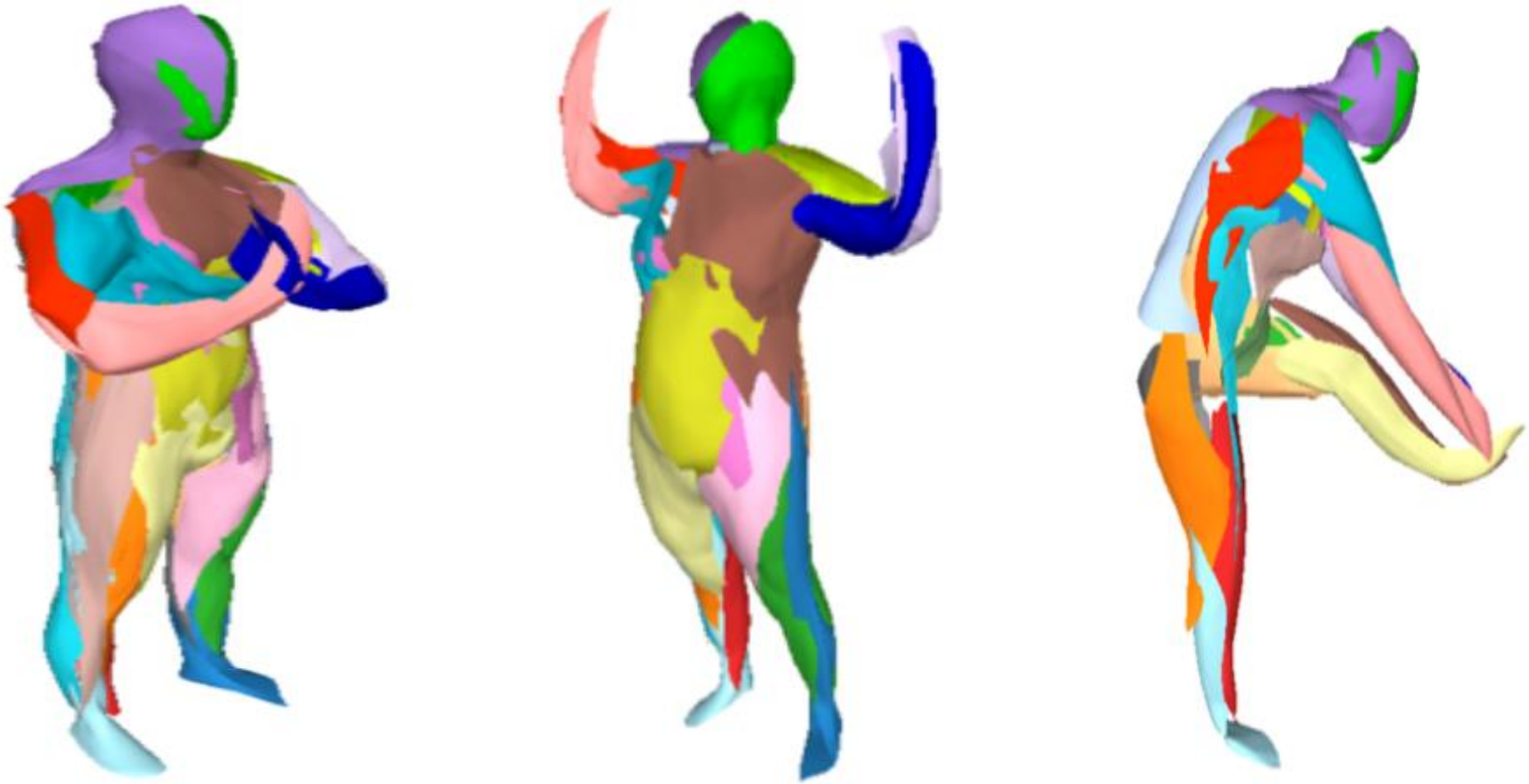
AtlasNet: applications

(a) Shape interpolation.

(b) Shape correspondences.

AtlasNet: limitations

Intersections are hard to detect



More exotic 3D representations

- Signed Distance Function (SDF)

DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

Jeong Joon Park[1,3†] Peter Florence [2,3†] Julian Straub[3] Richard Newcombe[3] Steven Lovegrove[3]

[1]University of Washington [2]Massachusetts Institute of Technology [3]Facebook Reality Labs

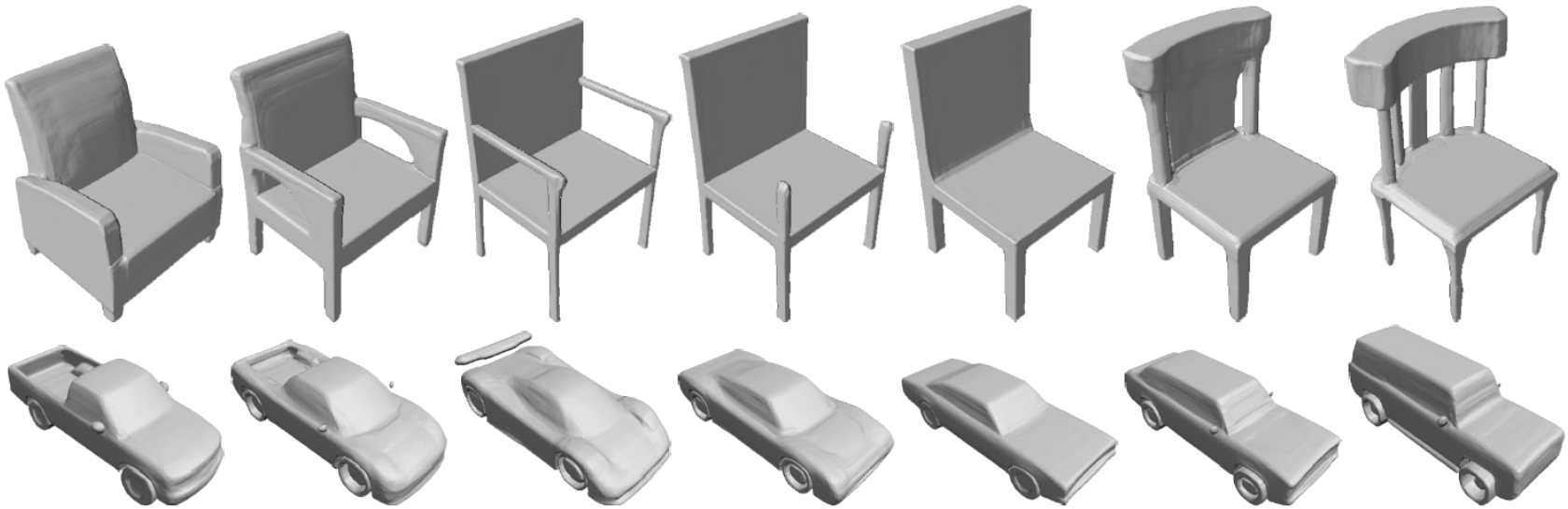


Figure 1: DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.

More exotic 3D representations

- Signed Distance Function (SDF)

DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

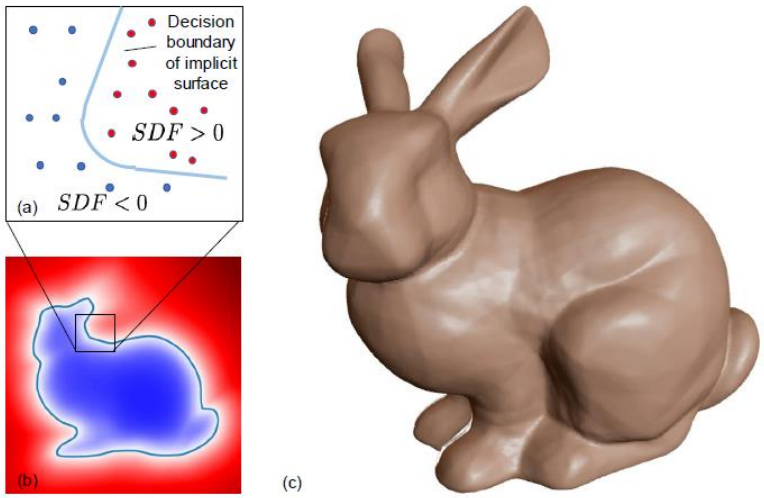


Figure 2: Our DeepSDF representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Note that (b) and (c) are recovered via DeepSDF.

More exotic 3D representations

- Signed Distance Function (SDF)

Gradient of SDF at the surface = normals!

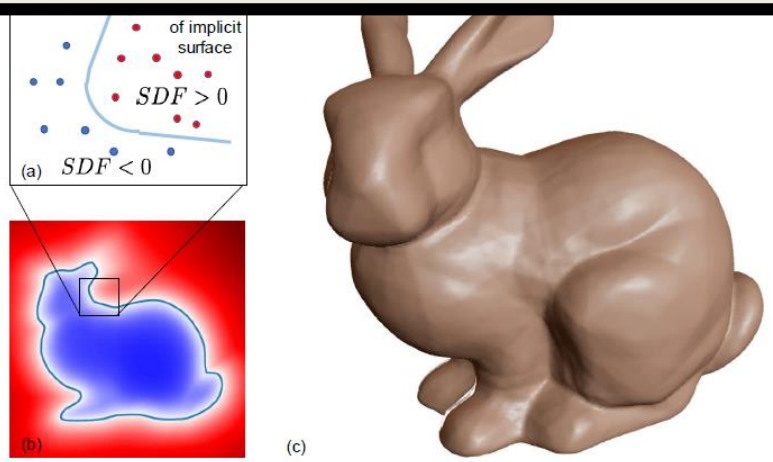
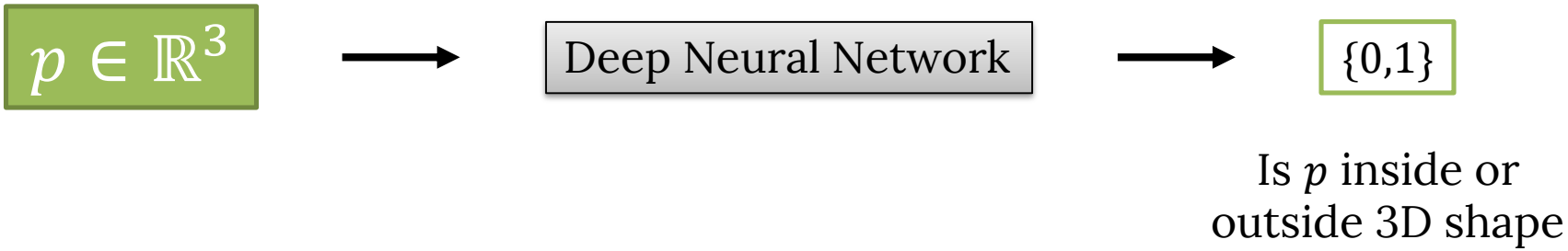


Figure 2: Our DeepSDF representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Note that (b) and (c) are recovered via DeepSDF.

More exotic 3D representations

- Signed Distance Function (SDF)
- Occupancy



"Occupancy Networks: Learning 3D Reconstruction in Function Space" by Mescheder et al., 2018

More exotic 3D representations

- Signed Distance Function (SDF)
- Occupancy
  - Classifier!

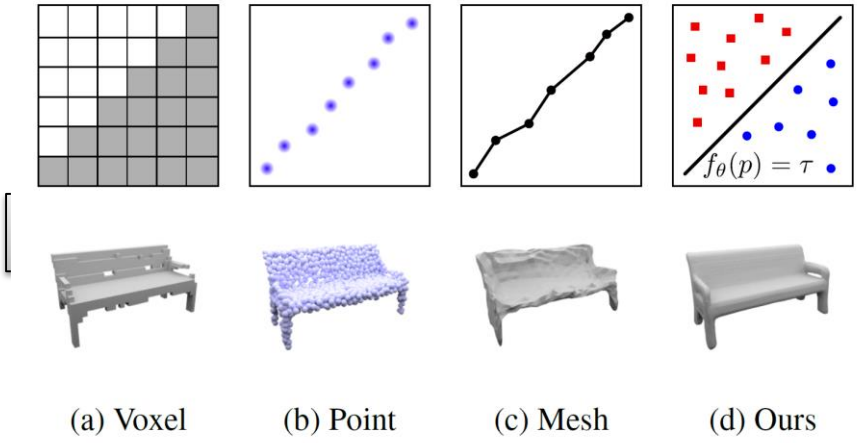


Figure 1: **Overview:** Existing 3D representations discretize the output space differently: (a) spatially in voxel representations, (b) in terms of predicted points, and (c) in terms of vertices for mesh representations. In contrast, (d) we propose to consider the continuous decision boundary of a classifier f_θ (e.g., a deep neural network) as a 3D surface which allows to extract 3D meshes at any resolution.

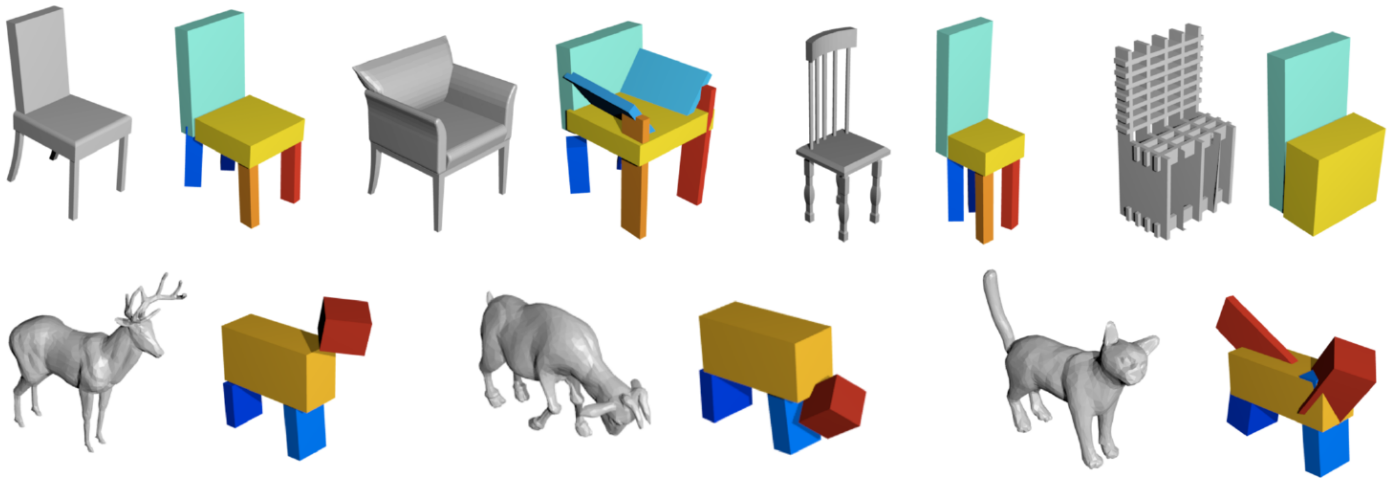
"Occupancy Networks: Learning 3D Reconstruction in Function Space" by Mescheder et al., 2018

More exotic 3D representations

- Signed Distance Function (SDF)
- Occupancy
- Boxes?

Learning Shape Abstractions by Assembling Volumetric Primitives

Shubham Tulsiani[1], Hao Su[2], Leonidas J. Guibas[2], Alexei A. Efros[1], Jitendra Malik[1]
[1]University of California, Berkeley [2]Stanford University
[1]{shubhtuls, efros, malik}@eecs.berkeley.edu, [2]{haosu, guibas}@cs.stanford.edu



3D shape representations

- Multiple Images
- Voxelization
- Deformation
- Depth
- Point Cloud
- Component-based
- SDF

Deformable Meshes

Multi-chart Generative Surface Modeling

Heli Ben-Hamu Haggai Maron Itay Kezurer Gal Avineri Yaron Lipman
Weizmann Institute of Science

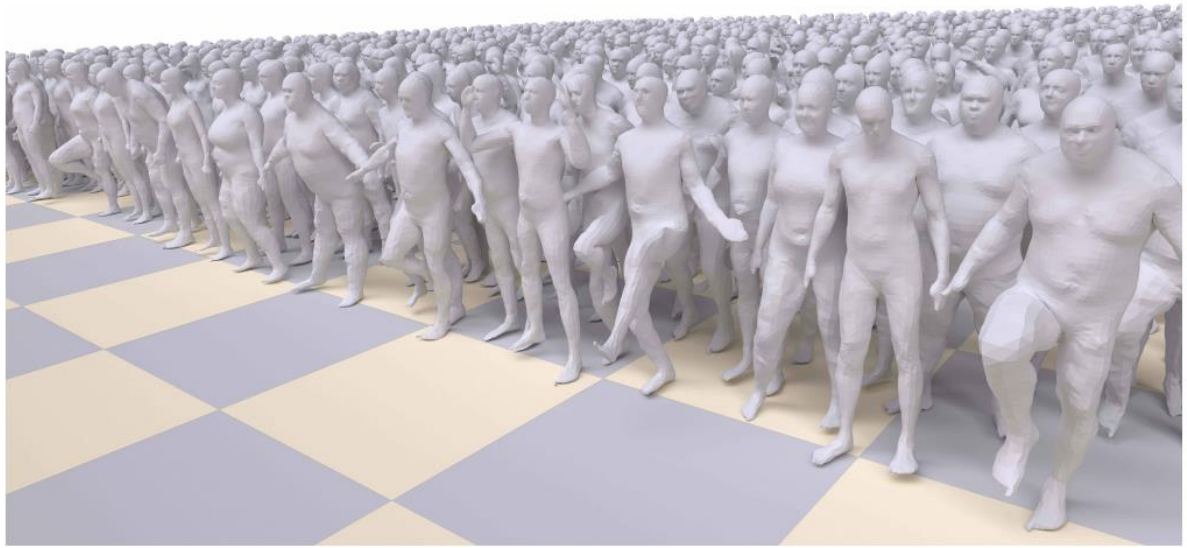


Figure 1: Our method is able to learn shape distribution and generate unseen shapes. This figure shows 1024 human models randomly generated by our method.

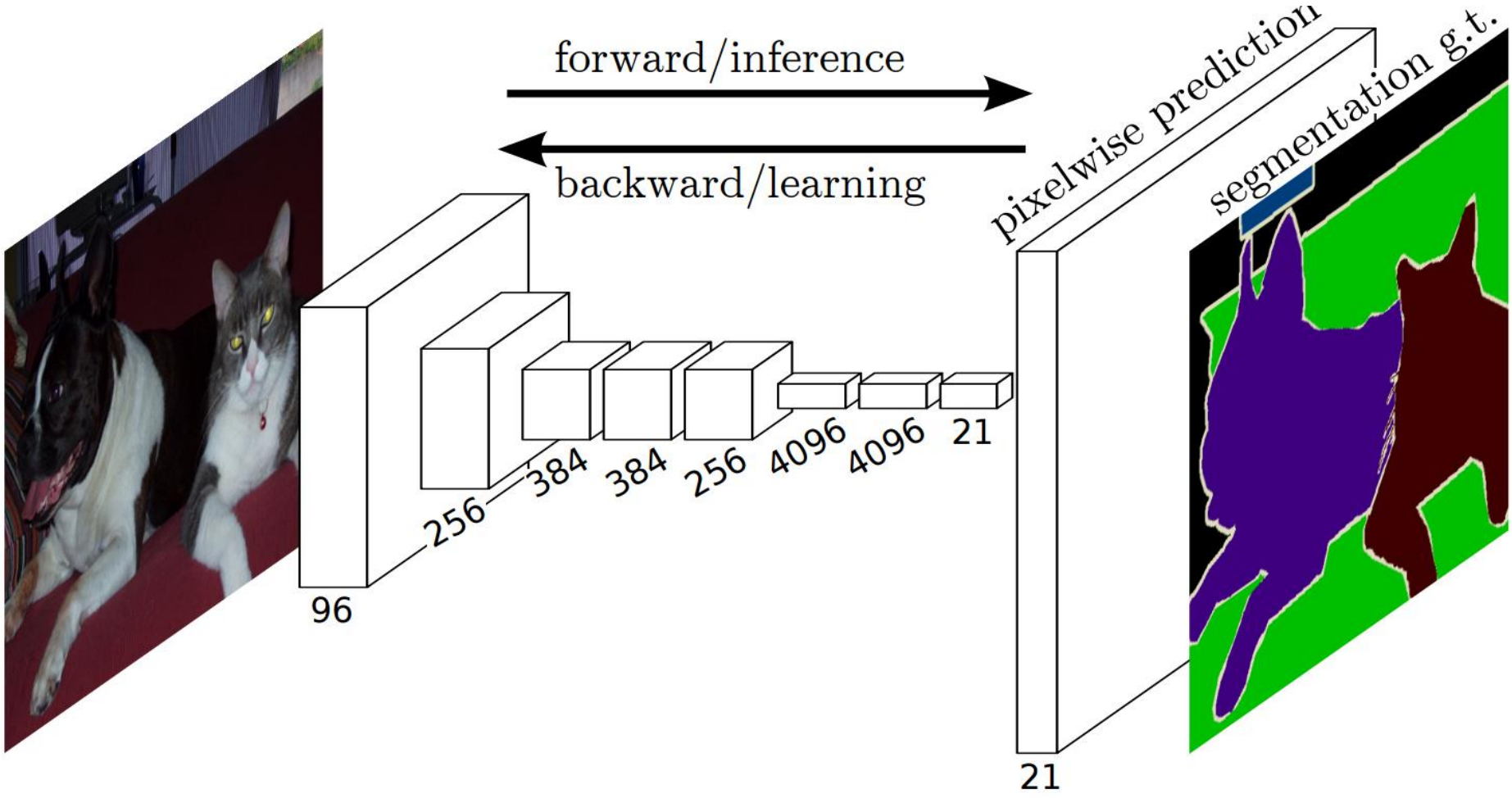
Abstract

This paper introduces a 3D shape generative model based on deep neural networks. A new image-like (*i.e.*, tensor) data representation for genus-zero 3D shapes is devised. It is based on the observation that complicated

the method is demonstrated for the task of anatomic shape generation including human body and bone (teeth) shape generation.

1 Introduction

Convolution for manifolds?



Learning on 3D geometry

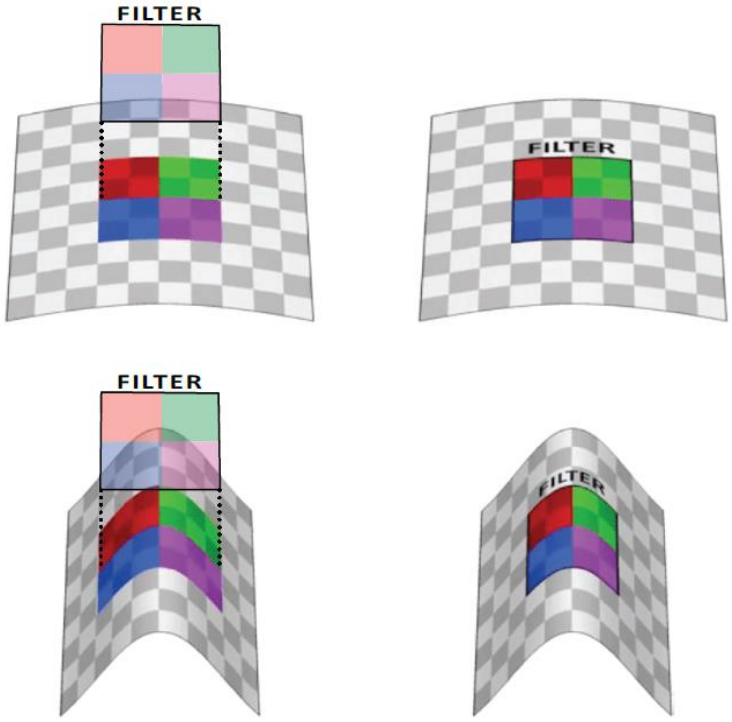


Fig. 5. Illustration of the difference between classical CNN (left) applied to a 3D shape (checkered surface) considered as a Euclidean object, and a geometric CNN (right) applied intrinsically on the surface. In the latter case, the convolutional filters (visualized as a colored window) are deformation-invariant by construction.

From 'Geometric Deep Learning: going beyond Euclidean data' by Bronstein et al. 2017

Convolution Theorem

"Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms"

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

Convolution Theorem

"Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms"

Point-wise multiplication

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

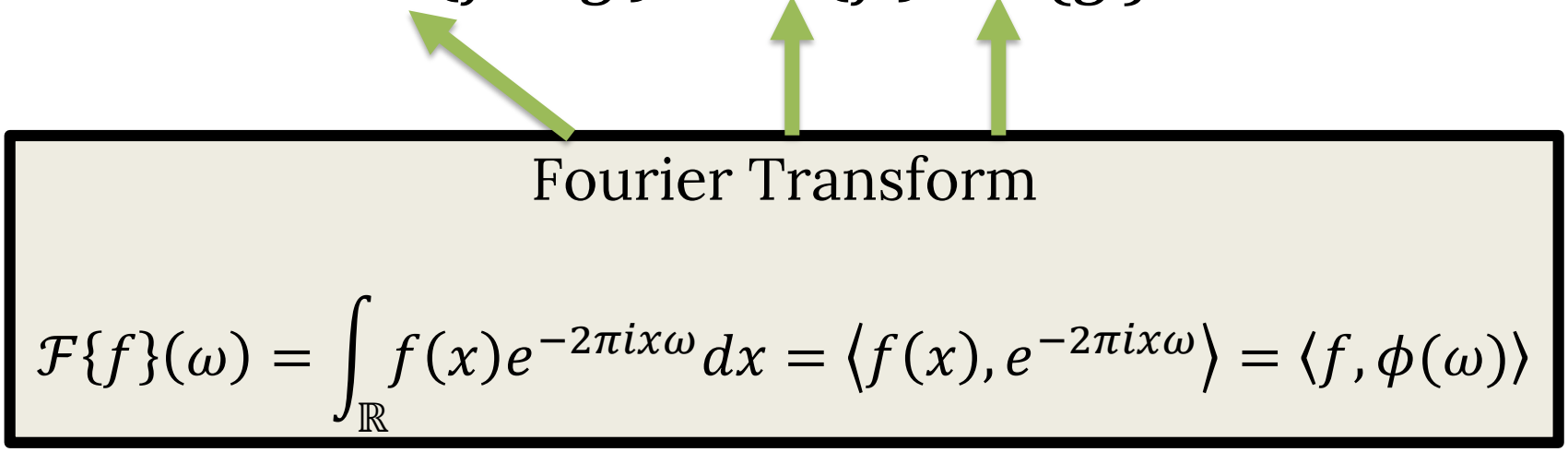
Convolution

$$(f * g)(z) = \int_{\mathbb{R}} f(x)g(z-x)dx$$

Convolution Theorem

"Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms"

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$



Convolution Theorem

"Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms"

Fourier basis

$$(f * g)(x) = \sum_{i \geq 0} \langle f, \phi_i \rangle \langle g, \phi_i \rangle \phi_i(x)$$

Fourier Series

Spectral convolution

Take this theorem as definition

~~**Fourier**~~ Laplacian Eigenbasis

$$(f * g)(x) = \sum_{i \geq 0} \langle f, \phi_i \rangle \langle g, \phi_i \rangle \phi_i(x)$$

Spectral convolution

Take this theorem as definition

~~**Fourier**~~ Laplacian Eigenbasis

$$(f * g)(x) = \sum_{i \geq 0} \langle f, \phi_i \rangle \langle g, \phi_i \rangle \phi_i(x)$$

$$f * g = \Phi(\Phi^T f)(\Phi^T g)$$

$$\Delta = \Phi \Lambda \Phi^T$$

Spectral CNN

Spectral convolutional layer:

$$f_{output} = ReLU(\sum_{l'} \Phi G_{l,l'} \Phi^T f_{input})$$

(diagonal) learnable parameters

J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun.
Spectral networks and locally connected networks on graphs,
Proc. ICLR, 2013.

Spectral CNN

Spectral convolutional layer:

Only makes sense within a category of isometric shapes!

l' (diagonal) learnable parameters

J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun.
Spectral networks and locally connected networks on graphs,
Proc. ICLR, 2013.

Geodesic Convolution

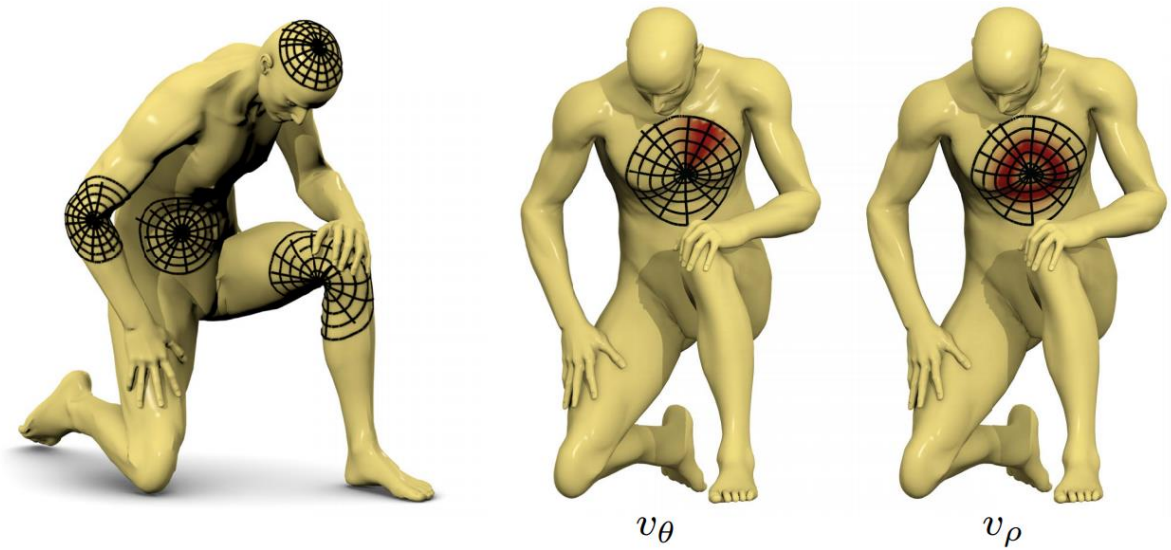
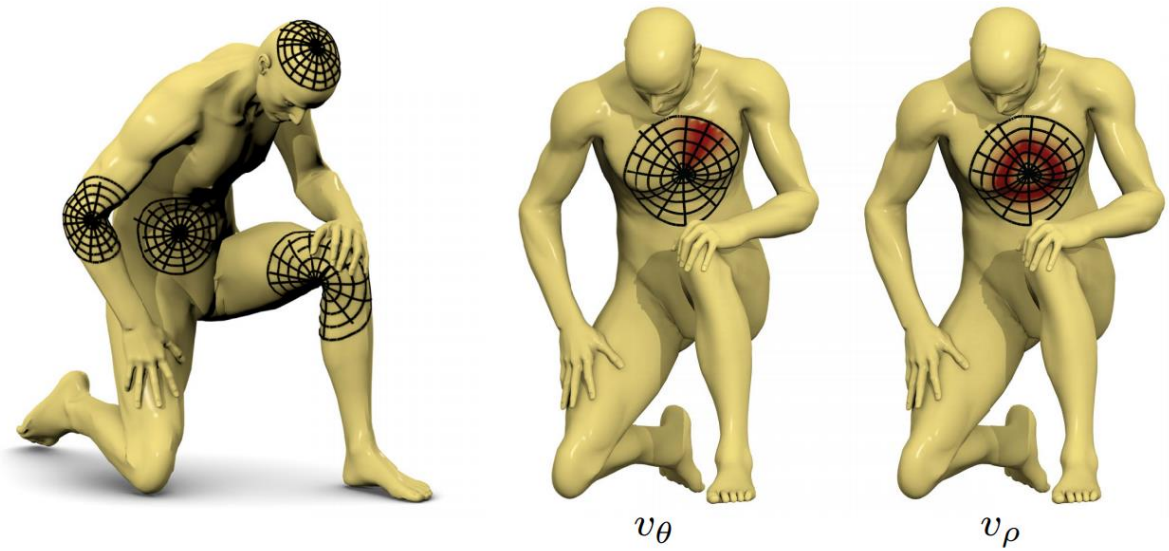


Figure 1: Construction of local geodesic polar coordinates on a manifold. Left: examples of local geodesic patches, center and right: example of angular and radial weights v_θ, v_ρ, respectively (red denotes larger weights).

Angular (geodesic given a direction)
and radial bins
(isocurve of geodesic distance → use Fast Marching)

Geodesic Convolution



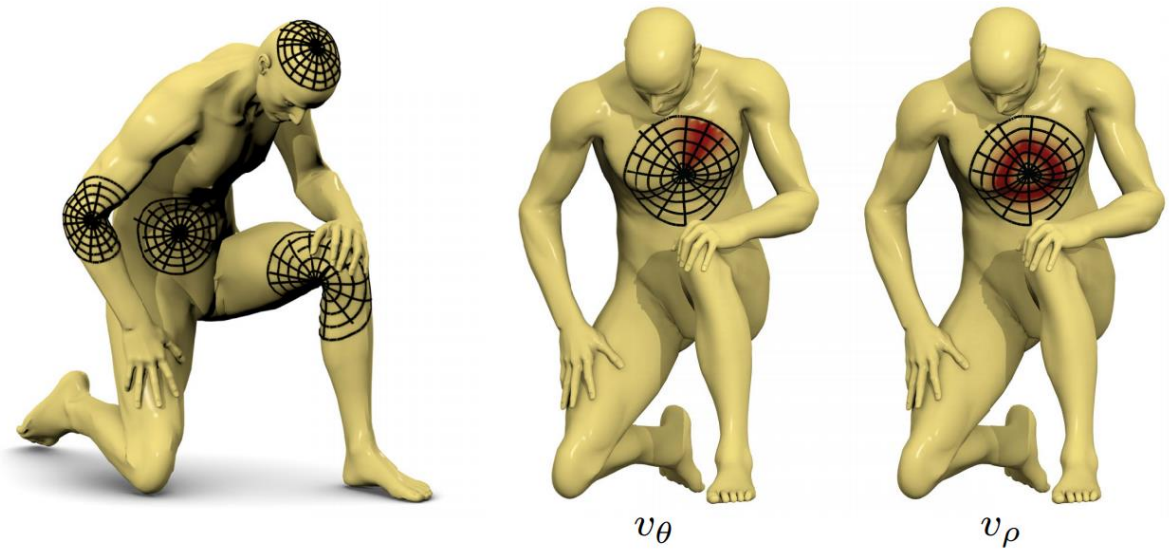
Filters in polar coordinates $a(\theta, r)$

Convolution:

$$(f \star a)(x) = \sum_{\theta, r} a(\theta + \Delta\theta, r)(D(x)f)(r, \theta)$$

Maps to geodesic polar coordinates

Geodesic Convolution



Ambiguity how to rotate the filter →
Try convolutions with all $\Delta\theta$ + angular max pooling

Application: Intrinsic Descriptors

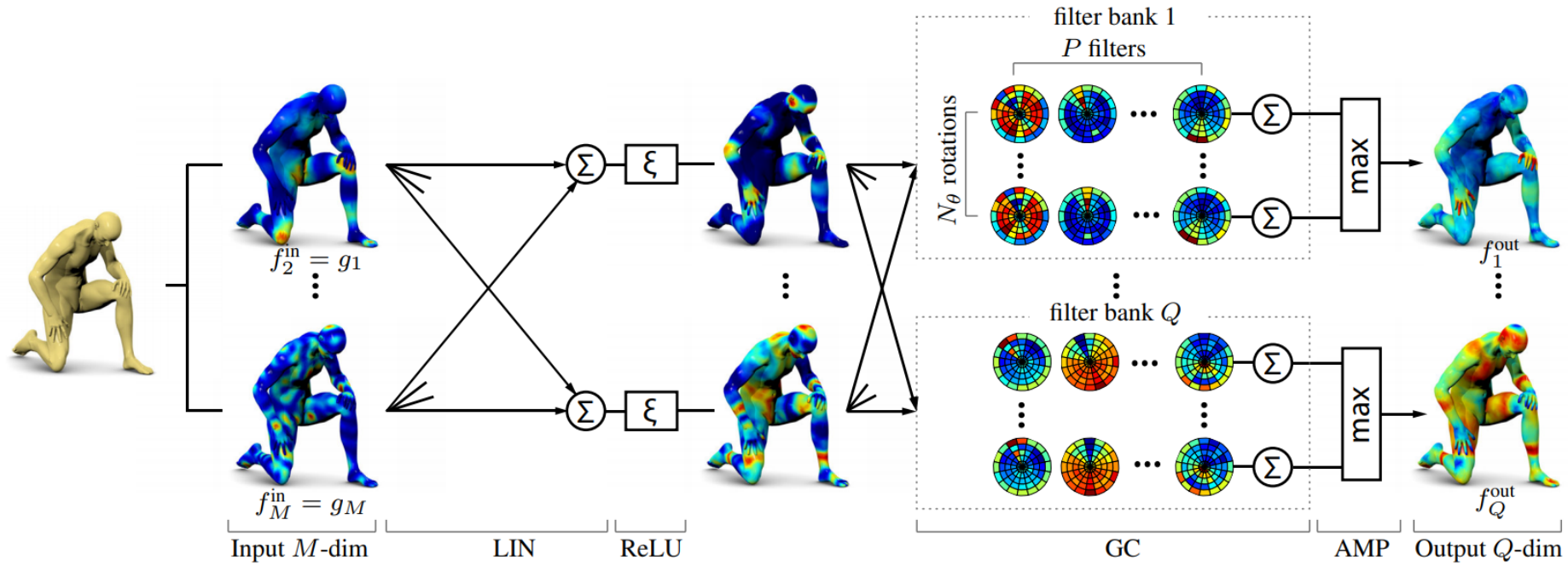


Figure 3: The simple GCNN1 architecture containing one convolutional layer applied to $M = 150$-dimensional geometry vectors (input layer) of a human shape, to produce a $Q = 16$-dimensional feature descriptor (output layer).

Apply to each vertex

input = all eigenvectors of the Laplacian

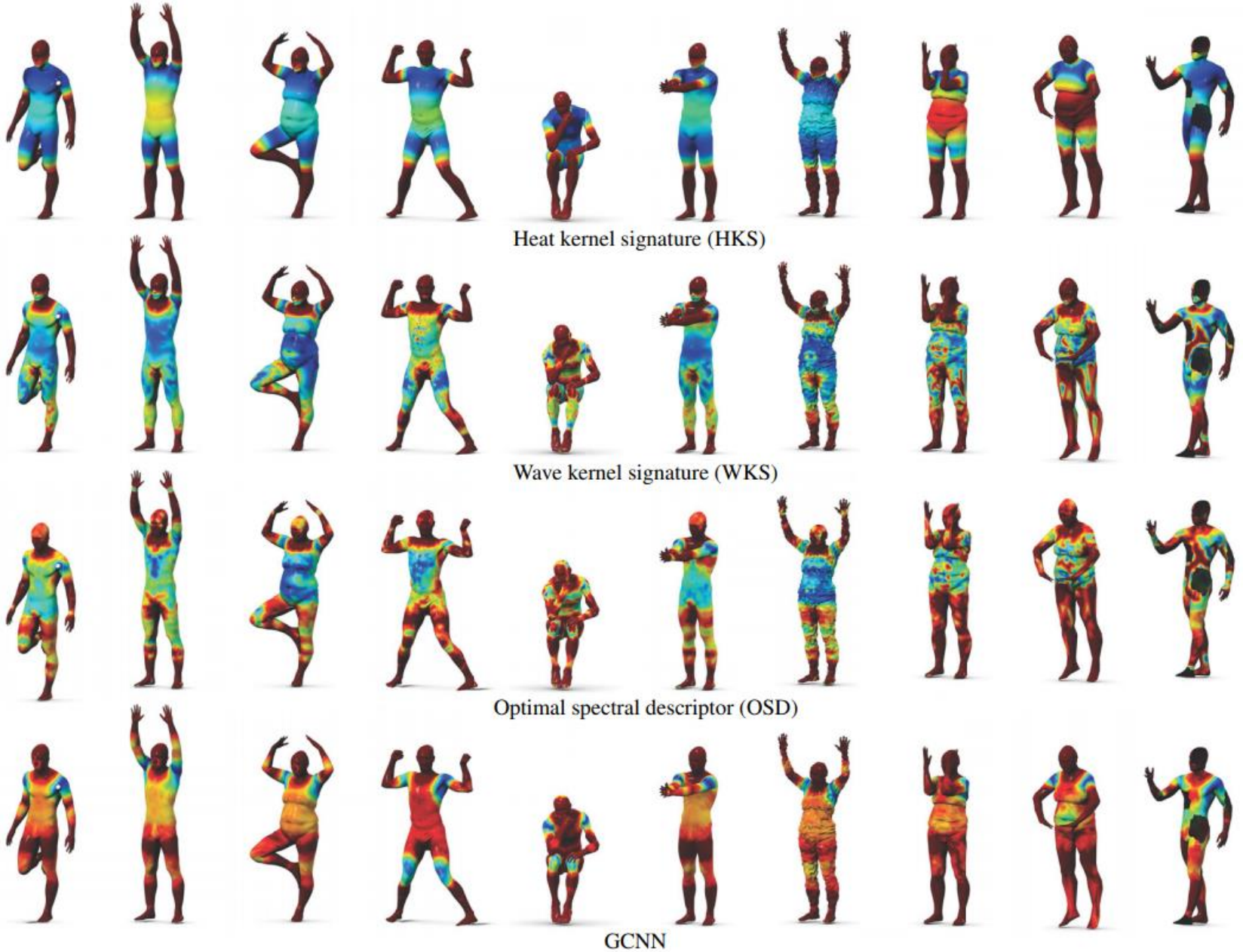


Figure 4: Normalized Euclidean distance between the descriptor at a reference point on the shoulder (white sphere) and the descriptors computed at the rest of the points for different transformations (shown left-to-right: near isometric deformations, non-isometric deformations, topological noise, geometric noise, uniform/non-uniform subsampling, missing parts). Cold and hot colors represent small and large distances, respectively; distances are saturated at the median value. Ideal descriptors would

PointNet

Input = point cloud (fixed # of pts)

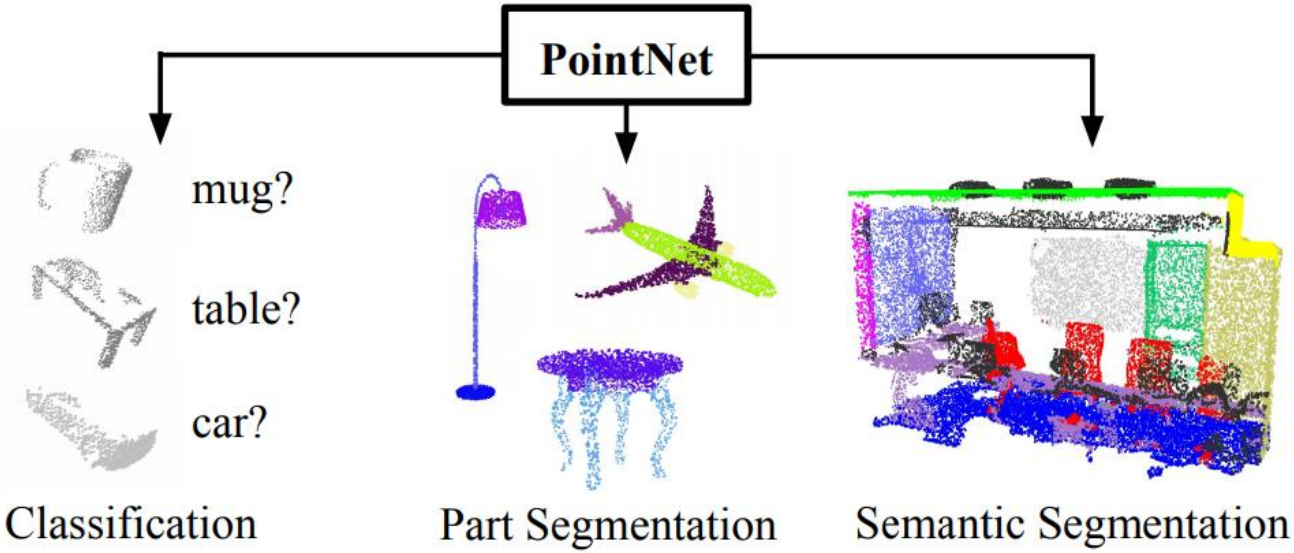


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

Point clouds are…

- Unordered
  - Nothing should change in the output if we change points order
- Sampled from a surface
  - Neighborhood matters
- Invariant under isometries

PointNet

permutation invariance

Symmetric functions:

$$f(x_1, \dots, x_n) = f\left(x_{\pi_1}, \dots, x_{\pi_n}\right)$$

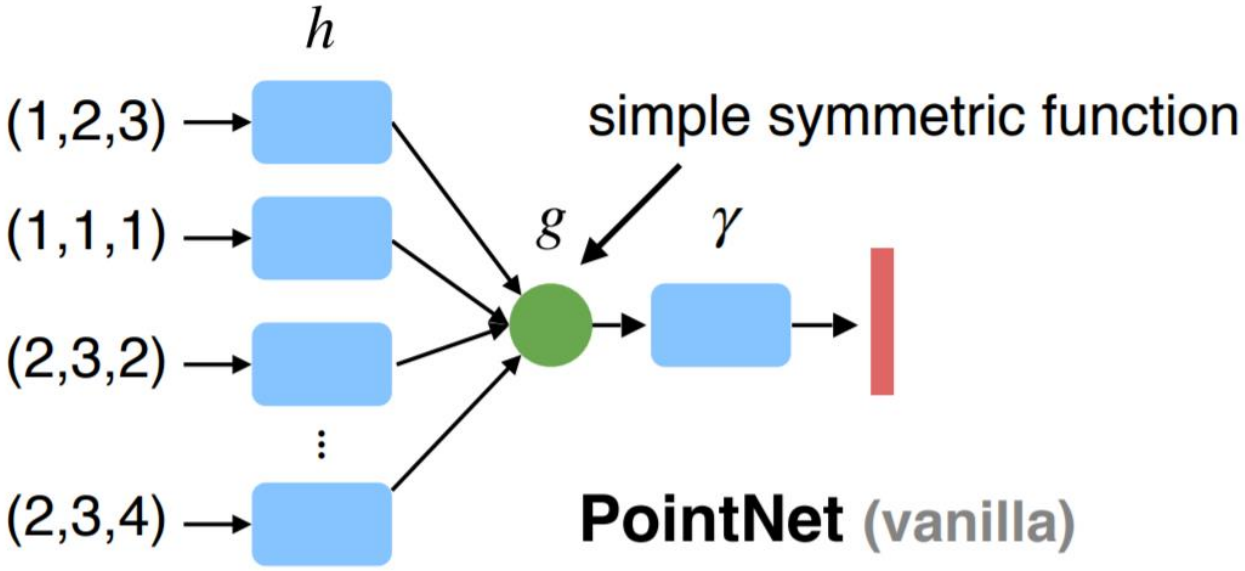
e.g.

$$f(x_1, \dots, x_n) = \sum_i x_i$$

$$f(x_1, \dots, x_n) = \max_i x_i$$

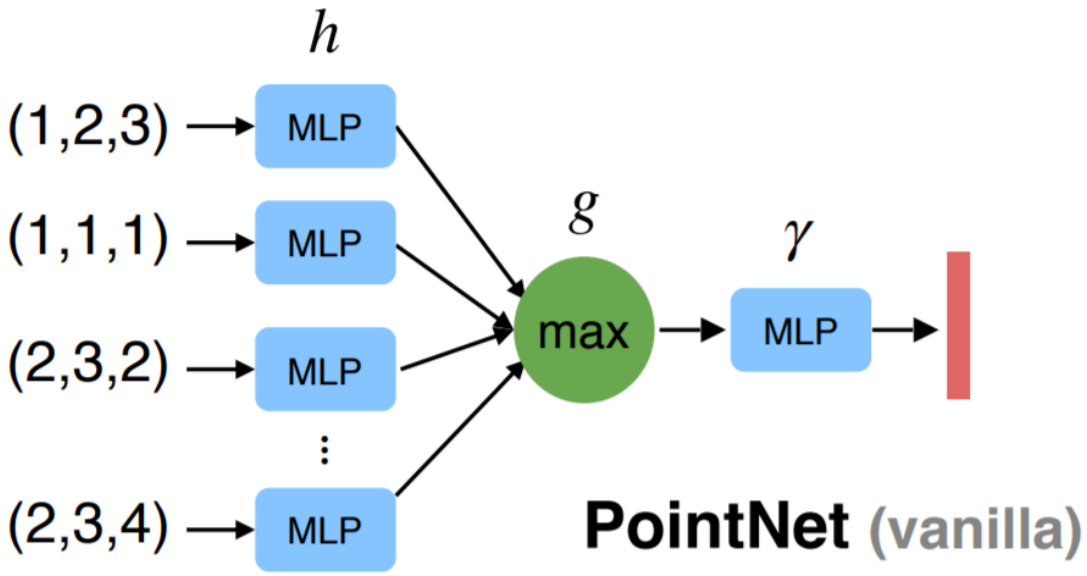
PointNet
permutation invariance

$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric



PointNet
permutation invariance

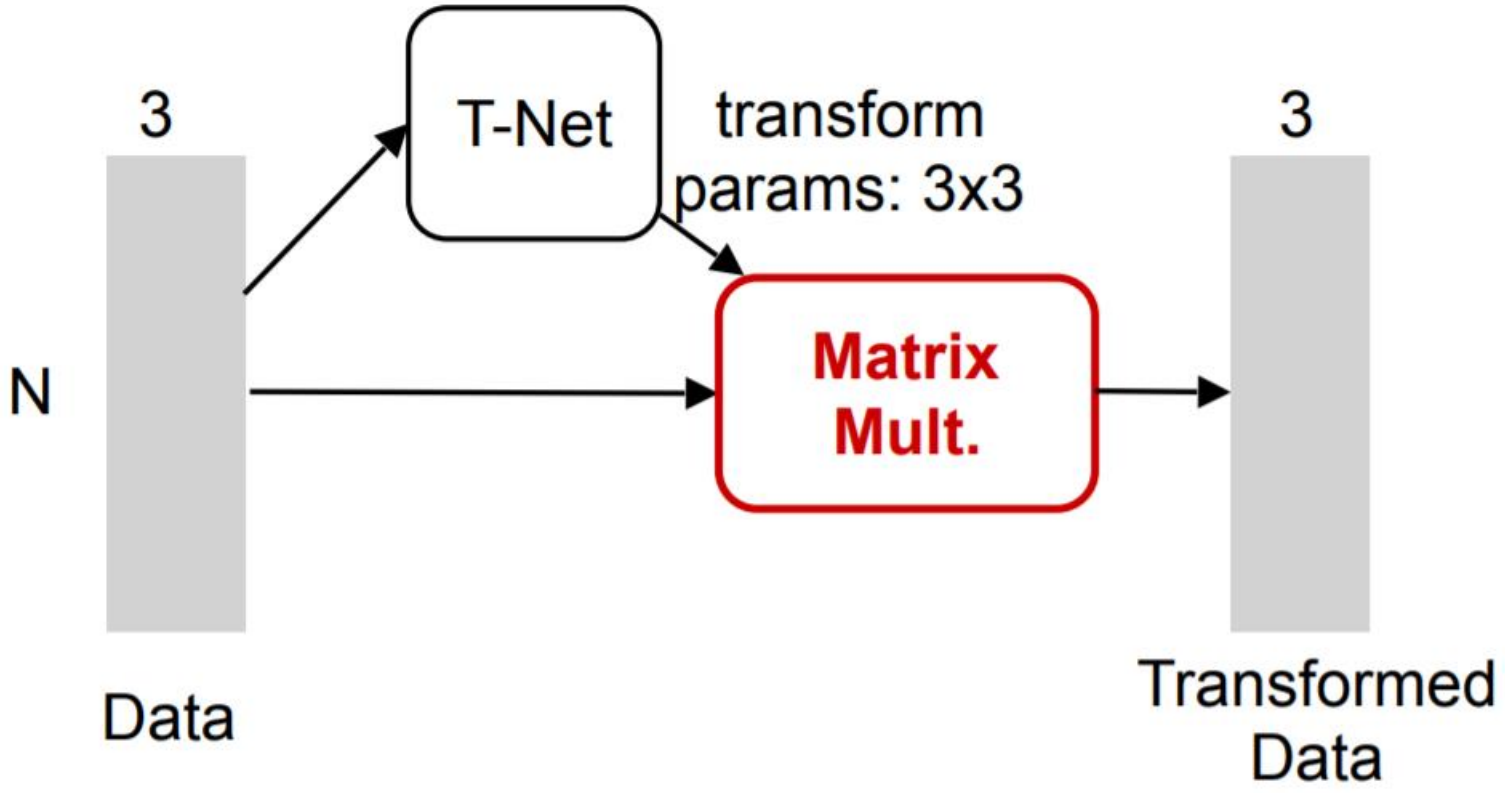
Empirically, we use **multi-layer perceptron (MLP)** and **max pooling**:



PointNet

invariance under rigid transforms

Another tiny net predicting canonical rotation



FİN