ECE4880J – Computer Vision

Guest Lecture on 3D Vision

Prof. He Wang

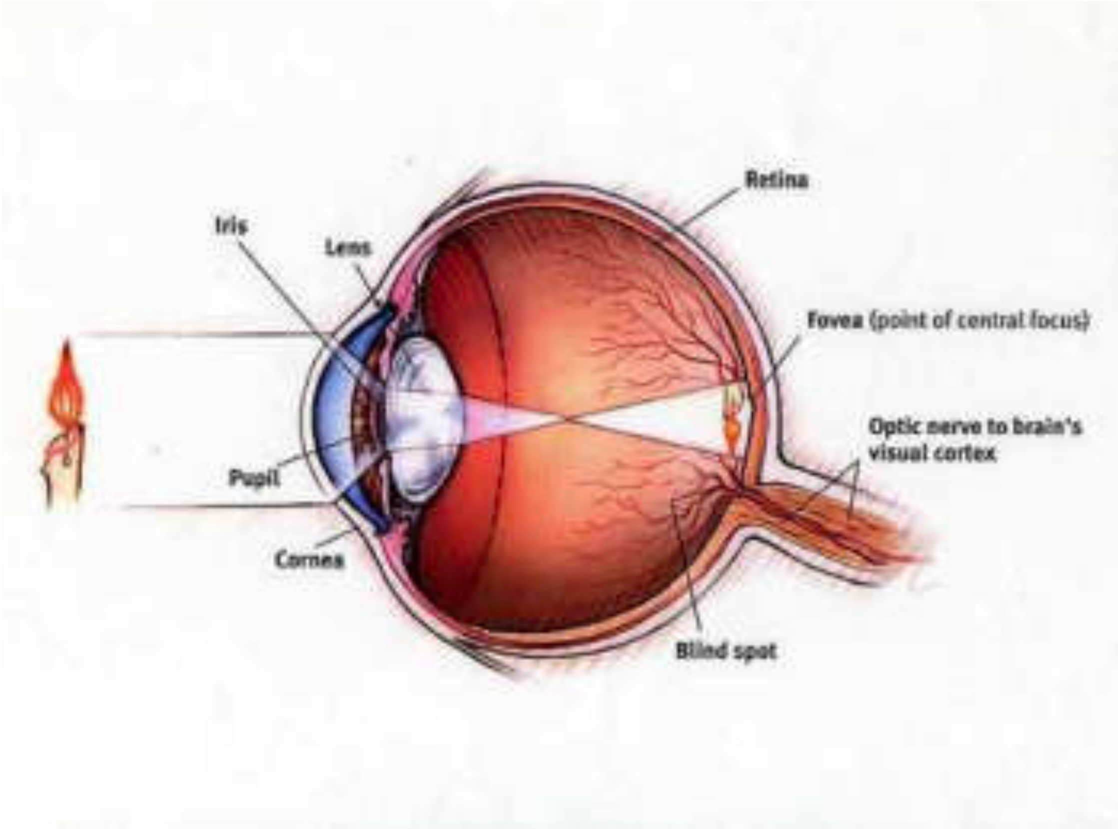
About Me

- 王鹤
- Assistant Professor in Center on Frontiers of Computing Studies (CFCS)
- Joined PKU in September, 2021
- Received Ph.D. from Stanford in 2021
- Received Bachelor from Tsinghua in 2014

- Our lab: *Embodied Perception and InteraCtion (EPIC) Lab*
- Research interest: 3D vision, Robotics
- Homepage: https://hughw19.github.io/

Why 3D Vision?

Monocular Vision



http://thebrucezone.weebly.com/3-image-formation.html

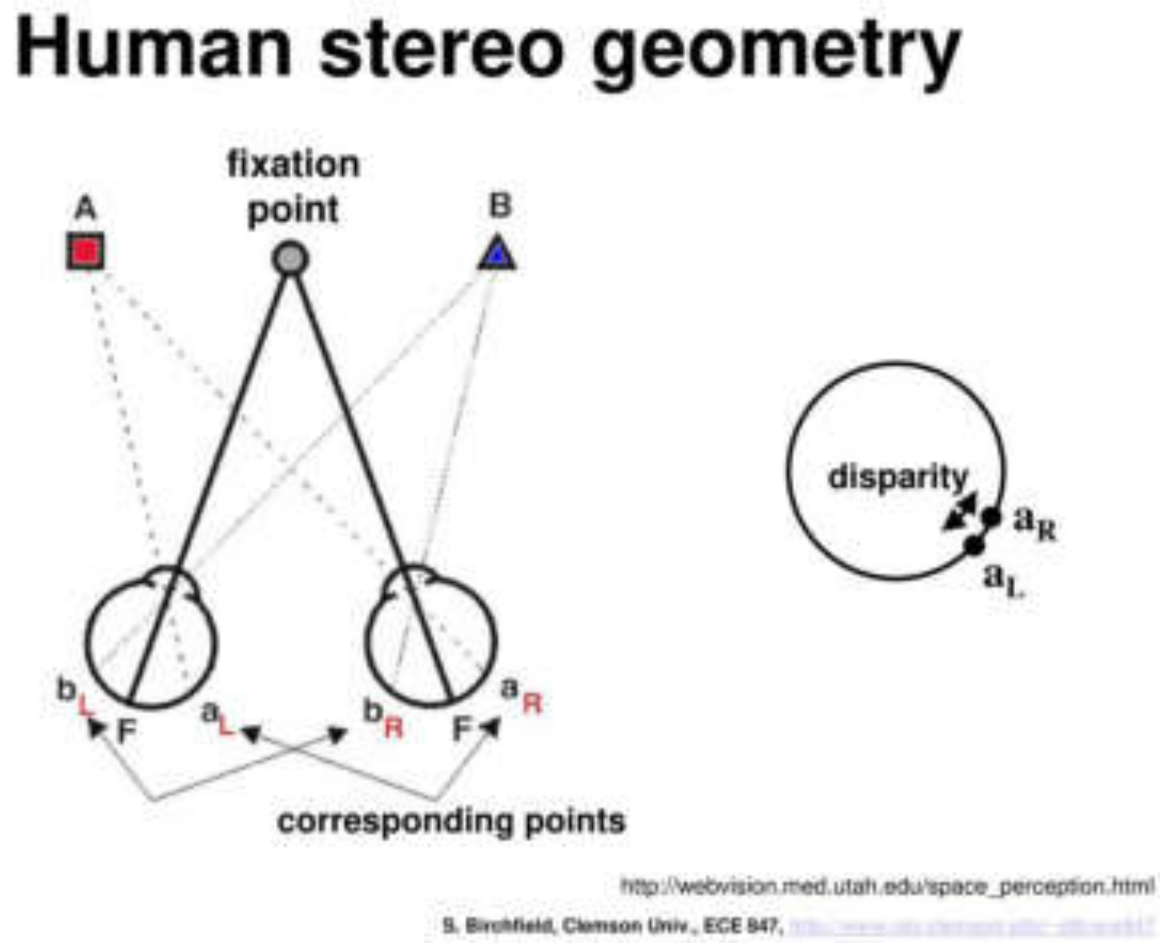
Issue of 2D Vision

- We don’t know true distance between pixels.
- Depth scale ambiguity

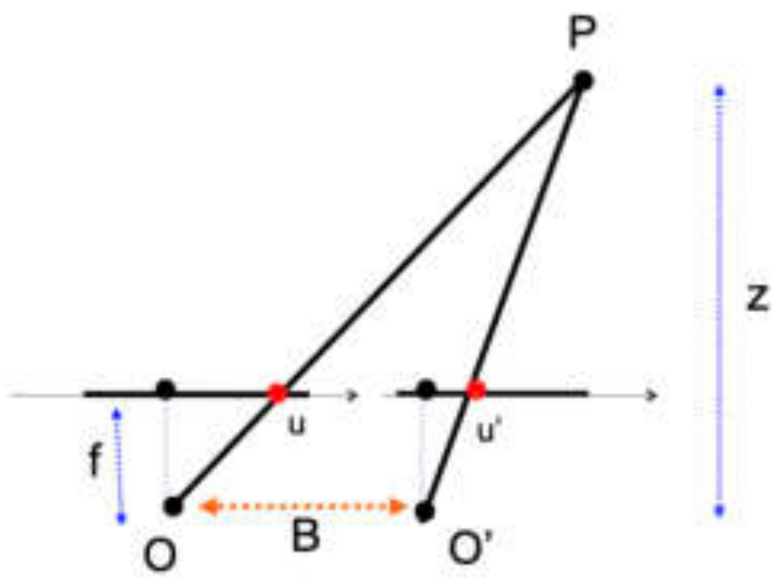
Courtesy slide S. Lazebnik

Binocular Vision and Stereopsis

- Human eyes are binocular.
- Senses distances through stereopsis



Computing Depth



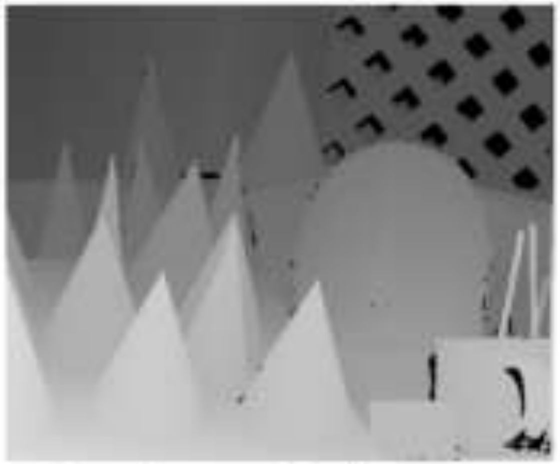
$$u - u' = \frac{B \cdot f}{z} = \text{disparity} \qquad \text{[Eq. 1]}$$

Note: Disparity is inversely proportional to depth

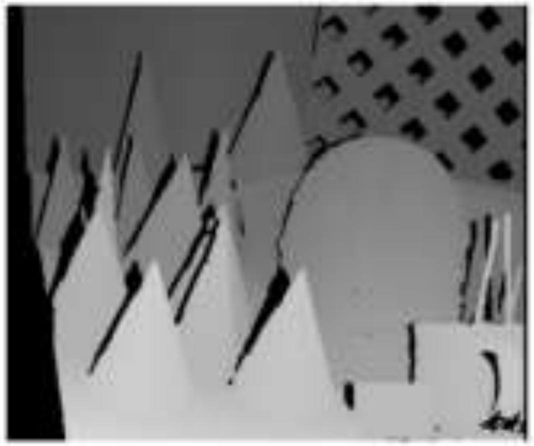
Disparity Maps

Stereo pair

$$u - u' = \frac{B \cdot f}{z}$$



Disparity map / depth map



Disparity map with occlusions

http://vision.middlebury.edu/stereo/

Where are 3D Data from?

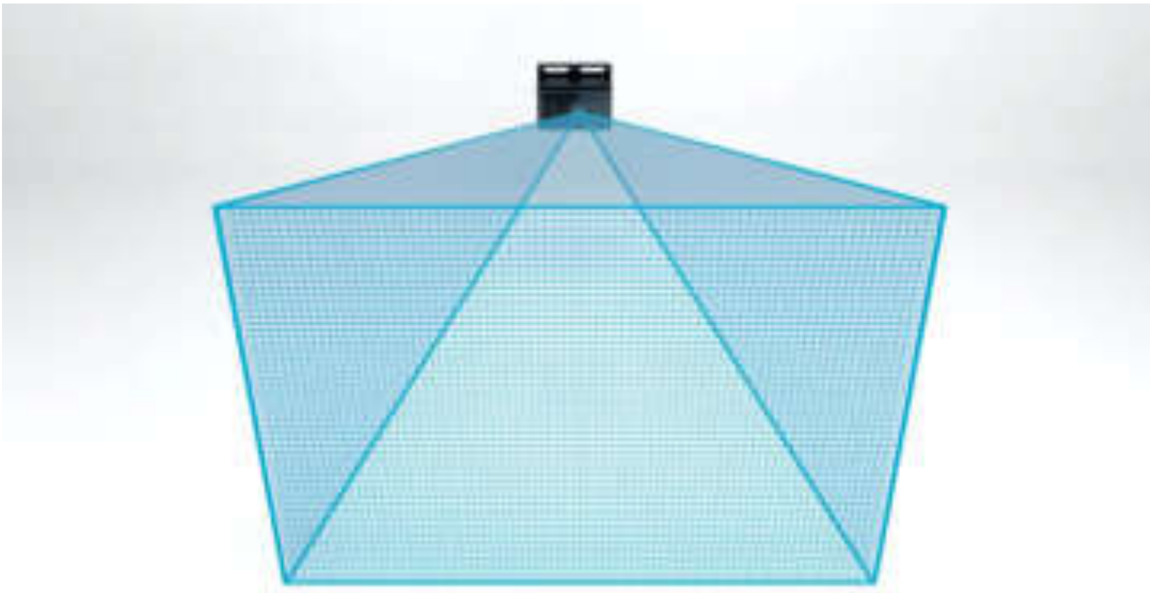
3D Data: from Sensors or Graphics

Real 3D data acquired by 3D sensing

Synthetic 3D data

Depth Sensors

- Depth sensors are a form of 3D range finder
- Measure multi-point distance information across a wide Field-of-View (FoV)



https://www.terabee.com/depth-sensors-precision-personal-privacy

Stereo Sensor

- Compute disparity and turn into depth.

Stereolabs Zed

Intel RealSense

Ensenso

Occipital Structure Core

Stereo Vision

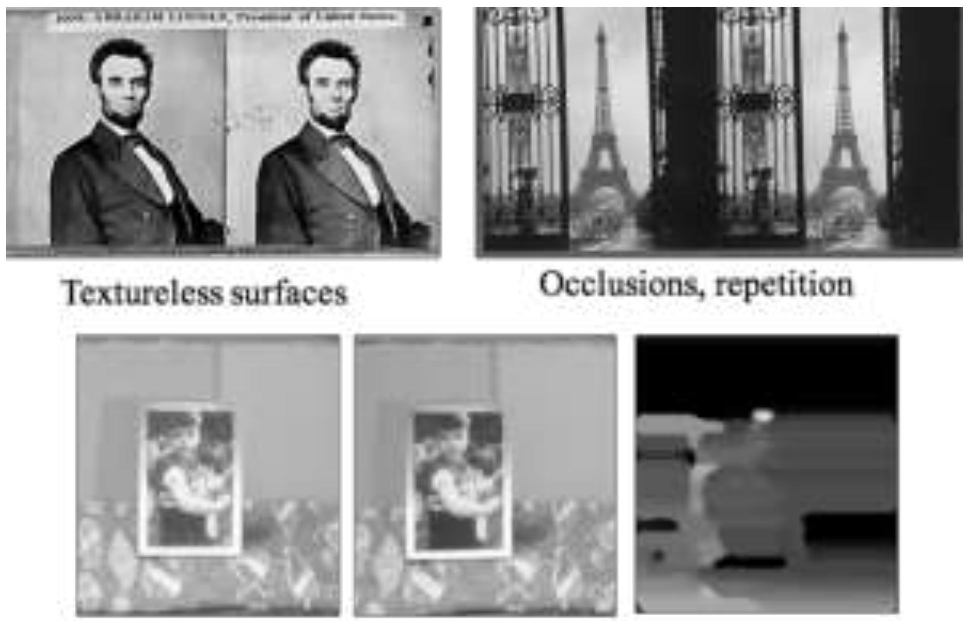
Advantage:

1. Robust to the illumination of direct sunlight
2. Low implementation cost

Drawback:

Finding correspondences along $Image_L$ and $Image_R$ is hard and erroneous

Failure of correspondence search



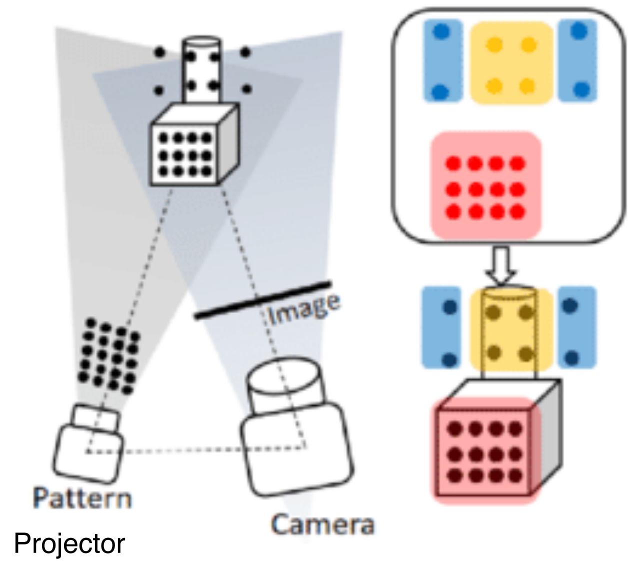
Textureless surfaces — Occlusions, repetition — Non-Lambertian surfaces, specularities

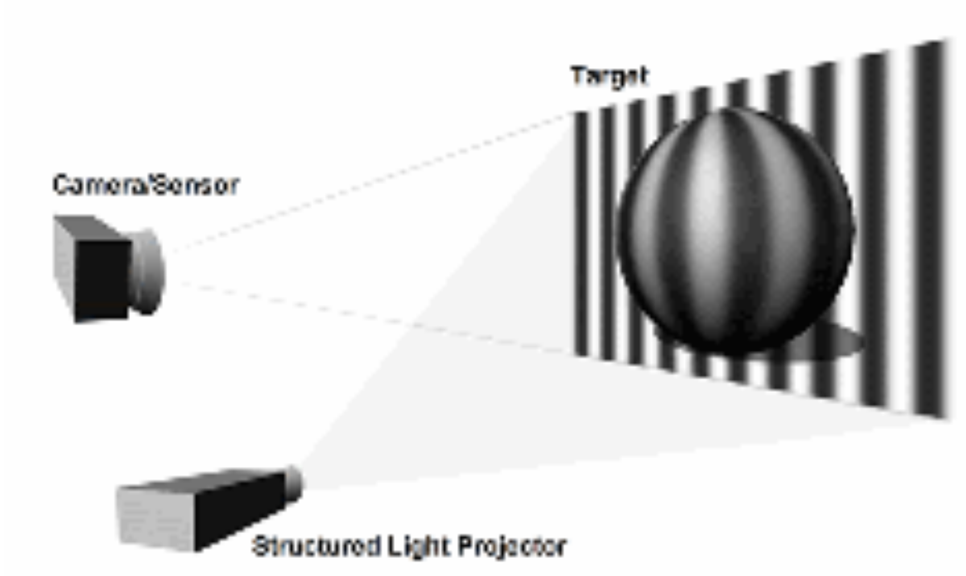
Structured Light

- Belongs to active stereoscopic approaches
- One camera replaced by an infrared projection unit
- Generates a pattern by projecting on the imaged surface

Advantage:
1. Simplify the correspondence problem

Drawback:
1. Near field
2. Indoor





Time-of-Flight

Microsoft Kinect v2 (2013)

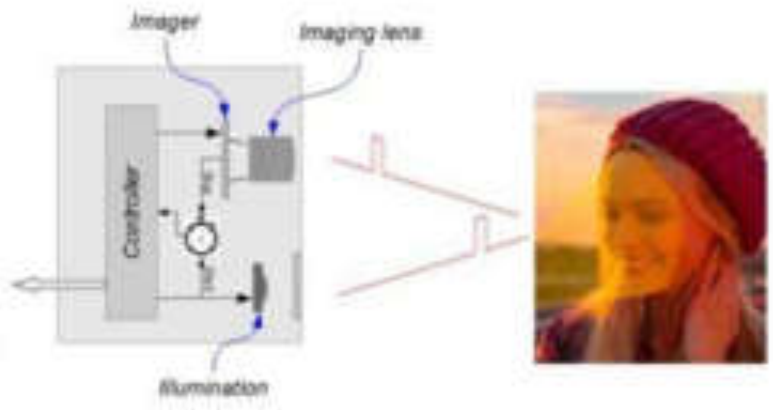
Microsoft Azure Kinect (2020)

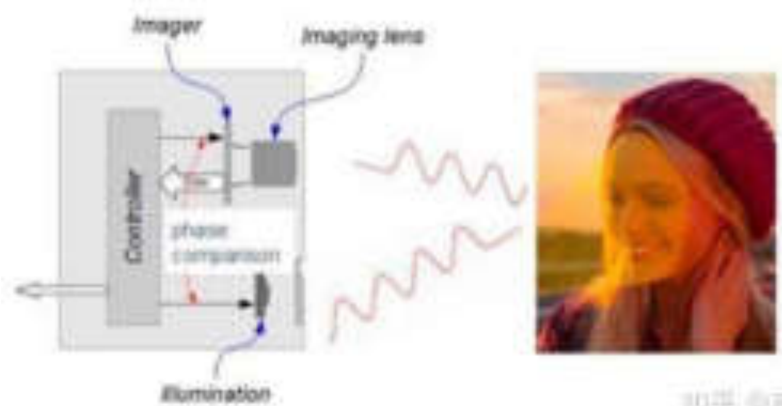
iPad Pro 2019 LiDAR

iToF vs. dToF

- dToF (the future)
  - Direct time-of-flight
  - Pulse wave
  - Long range
  - Theoretically higher precision but currently lower resolution
  - Expensive (needs SPAD)

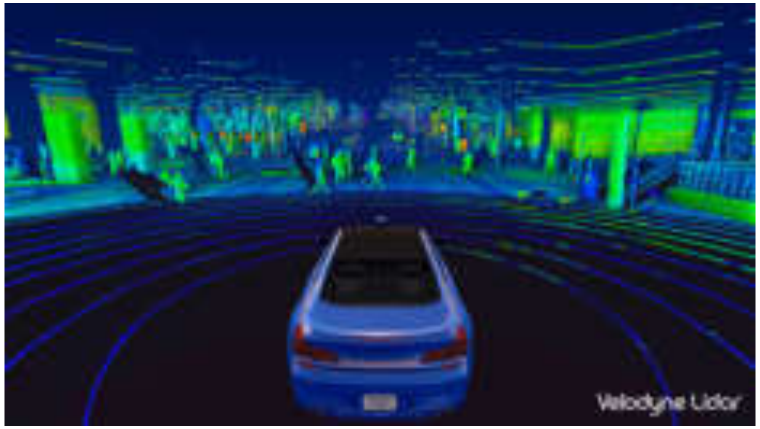
- iToF (Classic 3D imaging)
  - Indirect time-of-flight
  - Sin wave and solve for phase shift
  - Lower range
  - Lower precision but higher resolution
  - Cheaper





LiDAR in Autonomous Driving Cars

- A ToF sensor (mostly dToF) + a rotating scanner
- High laser intensity that supports sensing up to 200 meters and more
- Currently pretty low resolution (32 beams are common)



https://geoslam.com/what-is-lidar/

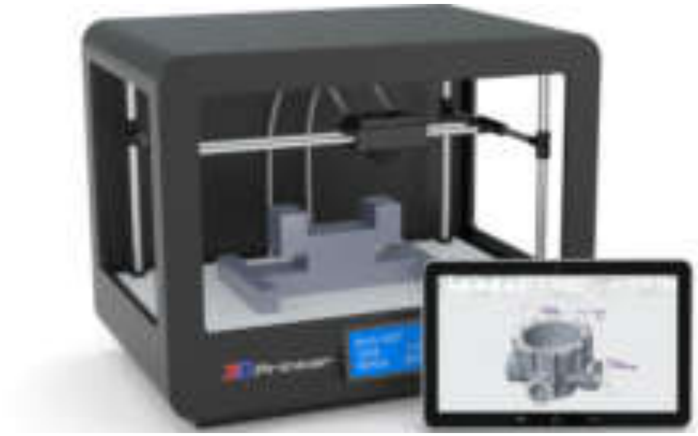
Summary of Different Depth Sensors

CONSIDERATIONS	STEREO VISION	STRUCTURED-LIGHT	TIME-OF-FLIGHT (TOF)
Software Complexity	High	Medium	Low
Material Cost	Low	High	Medium
Compactness	Low	High	Low
Response Time	Medium	Slow	Fast
Depth Accuracy	Low	High	Medium Quickly improving!
Low-Light Performance	Weak	Good	Good
Bright-Light Performance	Good	Weak	Good
Power Consumption	Low	Medium	Scalable
Range	Limited	Scalable	Scalable

CAD Models from Graphics Community

- CAD: computer-aided design models
- Widely used in
  - Graphics applications, including games, movies, animations, etc.
  - 3D printing and fabrications
  - …

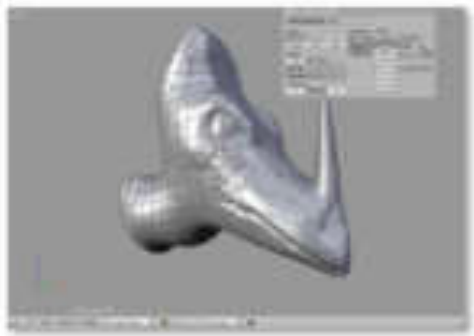
https://www.cadnav.com/3d-models/model-49123.html



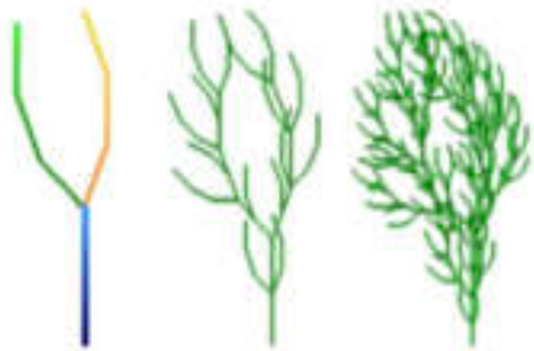
https://www.ptc.com/-/media/Images/CAD-Blog/2018/June/materialise/3d-printer.png?sc_lang=en

How to Obtain CAD Models

- Modeling by designers



- 3D shape synthesis algorithm
  - Procedural modeling
  - Generative models



- Acquired by 3D scans

Synthetic Datasets for 3D Objects

Large-scale Synthetic Objects: ShapeNet

…

ModelNet: absorbed by ShapeNet

Chang et al., **"ShapeNet: An Information-Rich 3D Model Repository"** , *arXiv*
Wu et al., "**3D ShapeNets: A deep representation for volumetric shapes**", *CVPR 2015*
Choi et al., **"A Large Dataset of Object Scans**", *arXiv*

Datasets for Indoor 3D Scenes

Large-scale Synthetic Scenes: SceneNet

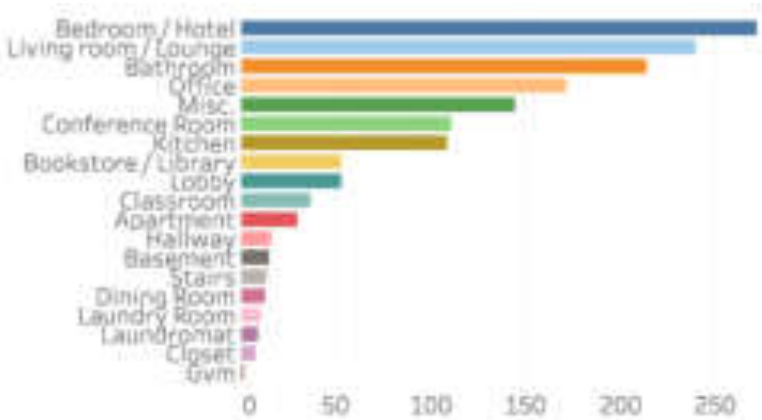
3D meshes
5M Photorealistic Images

Ankur et al., **“Understanding RealWorld Indoor Scenes with Synthetic Data”**, *CVPR 2016*
McCormac et al., **“SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?”**, *ICCV 2017*

Datasets for Indoor 3D Scenes

Large-scale Scanned Real Scenes: ScanNet

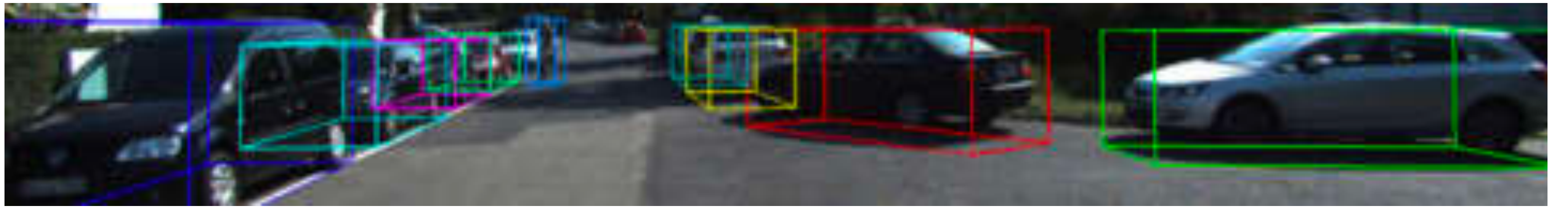
2.5 M Views in 1500 RGBD scans
3D camera poses
surface reconstructions
Instance-level semantic segmentations



Dai et al., "**ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes**", *CVPR 2017*

Datasets for Outdoor 3D Scenes

KITTI: LiDAR data, labeled by 3D bboxes



Semantic KITTI: LiDAR data, labeled per point

Waymo Open Dataset: LiDAR data, labeled by 3D b.boxes

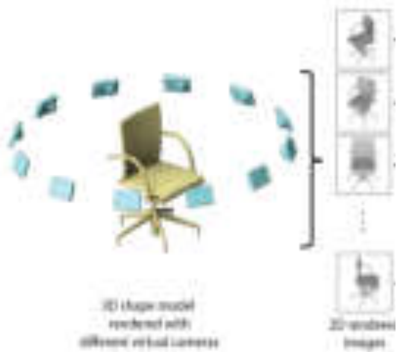
3D Representations

2D Image Representations

$H \times W \times 3$

Multiple 3D Representations

Regular form



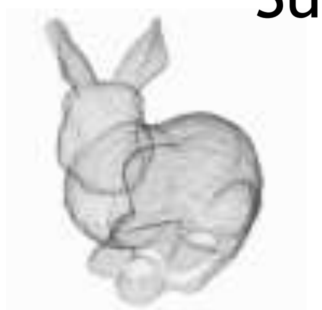
Multi-view images

Depth

Volumetric

Irregular form

Surface Mesh

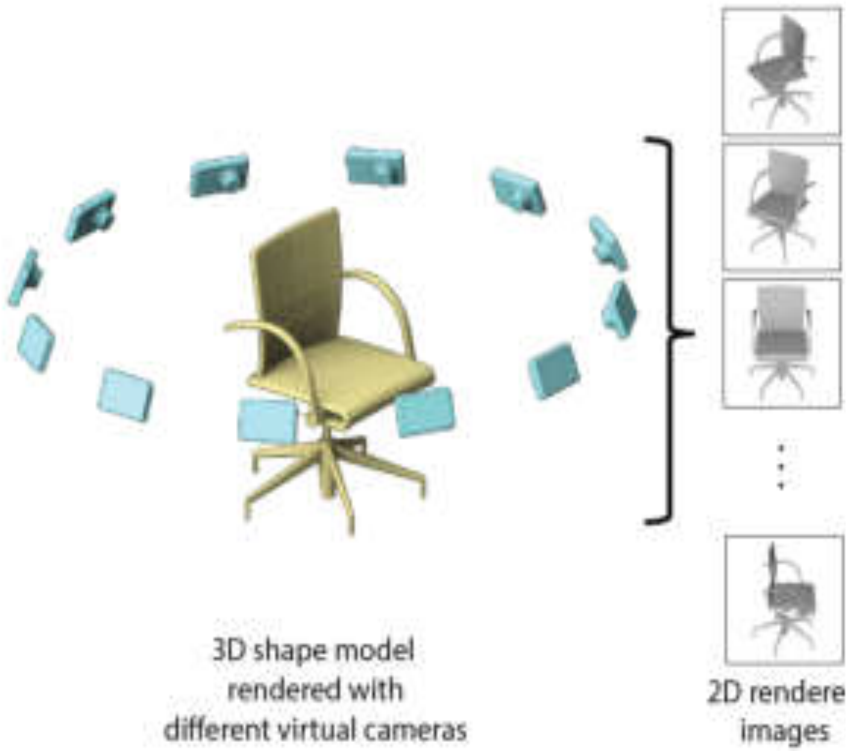


Point Cloud

$$F(x) = 0$$

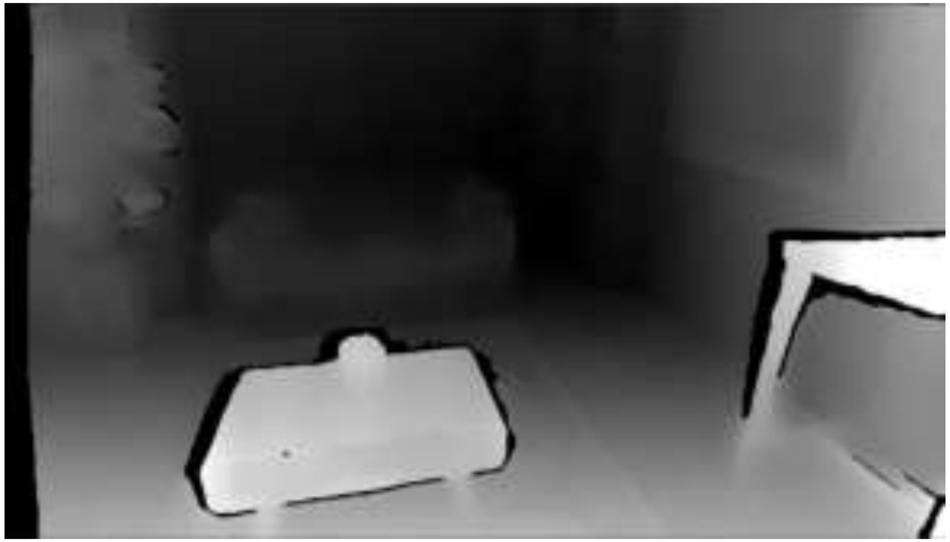
Implicit representation

Multi-View Images



- Multiple images from different viewpoints
- Contain 3D information
- Indirect, not a true 3D representation

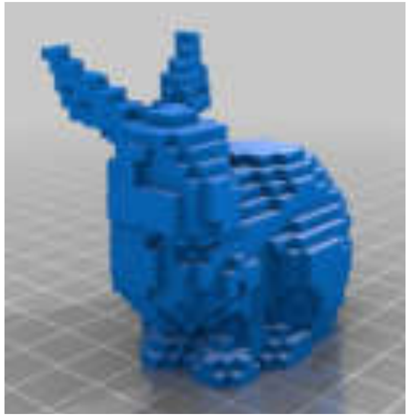
Depth Image



- A single-channel image filled by depth values
- A 2.5D representation

True 3D representation should enable distance measurement between two points.

Voxels

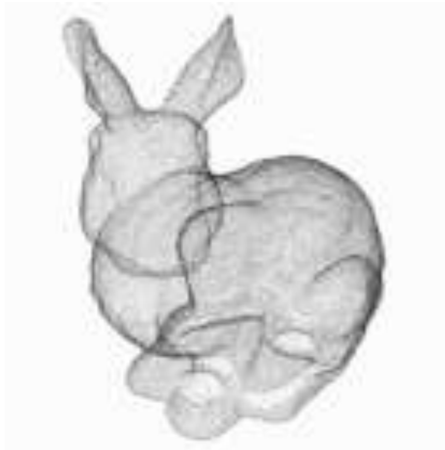


Voxels

- $H \times W \times D$
- Can be indexed
- An expensive geometry representation
- Not a surface representation
  - Where is the surface?
  - How to upsample?

Irregular 3D Representation

Mesh



Point Cloud

$$F(x) = 0$$

Implicit representation

- Irregular representation
- Model the 3D via capturing the surface or something on the surface

Mesh

Surface Mesh

Mesh of
Stanford Bunny

- A piece-wise Linear Surface Representation
- Both a geometry and surface representation

Different Kinds of Mesh in Different Resolutions

Triangle mesh at different resolutions

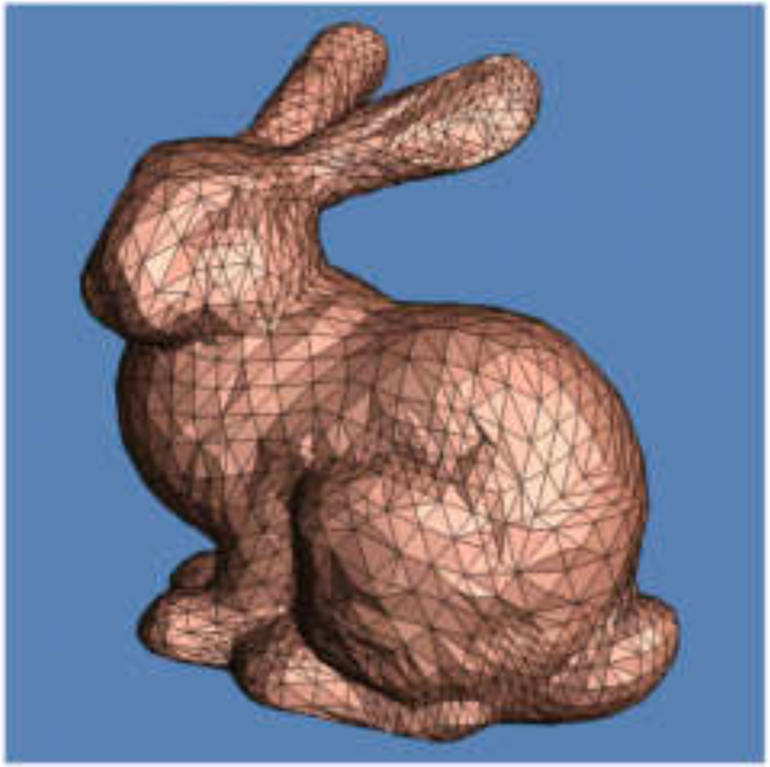
Quad mesh

Triangle Mesh

- Mesh essentially is a graph: {vertex, edge}
- Faces are triangles

$$V = \{v_1, v_2, \ldots, v_n\} \subset \mathbb{R}^3$$
$$E = \{e_1, e_2, \ldots, e_k\} \subseteq V \times V$$
$$F = \{f_1, f_2, \ldots, f_m\} \subseteq V \times V \times V$$

http://graphics.stanford.edu/data/3Dscanrep/stanford-bunny-cebal-ssh.jpg http://www.stat.washington.edu/wxs/images/BUNMID.gif



Data Structure for Mesh

- What information should be stored?
  - Geometry: 3D coordinates
  - Topology
  - Attributes
    - Normal, color, texture coordinates
    - Per vertex, face, edge

Simple Data Structure: Triangle List

- STL format (used in CAD)
- Stored information
  - Face: 3 positions
- No connectivity information

Triangles			
0	x0	y0	z0
1	x1	x1	z1
2	x2	y2	z2
3	x3	y3	z3
4	x4	y4	z4
5	x5	y5	z5
6	x6	y6	z6
...	...	...	...

Indexed Face Set

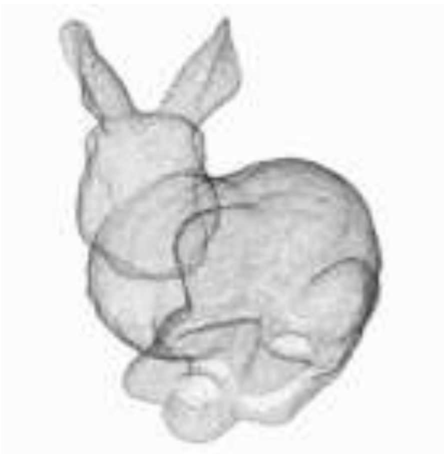
- Used in formats
  - OBJ, OFF, WRL
- Stored information
  - Vertex: position
  - Face: vertex indices
  - Convention is to save vertices in **counter-clockwise order (right hand rule)** for normal direction (pointing out)

Vertices			
v0	x0	y0	z0
v1	x1	x1	z1
v2	x2	y2	z2
v3	x3	y3	z3
v4	x4	y4	z4
v5	x5	y5	z5
v6	x6	y6	z6
...	...	...	...

Triangles			
t0	v0	v1	v2
t1	v0	v1	v3
t2	v2	v4	v3
t3	v5	v2	v6
...	...	...	...

Point Cloud

Point Cloud

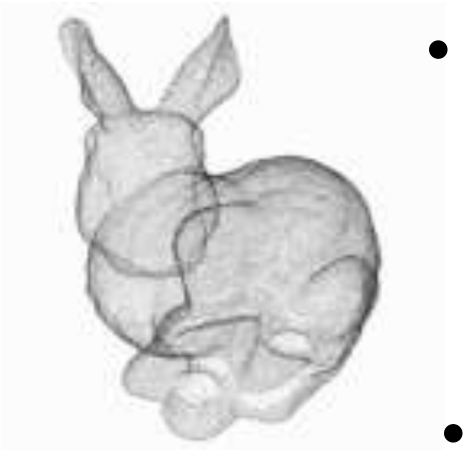


Point Cloud

- N*3
- Irregular and orderless data
- A light-weight geometric representation
  - Compact to store
  - Easy to understand and generally easy to build algorithms

$N = 125$ $N = 250$ $N = 500$ $N = 1000$

Limitations of Point Cloud

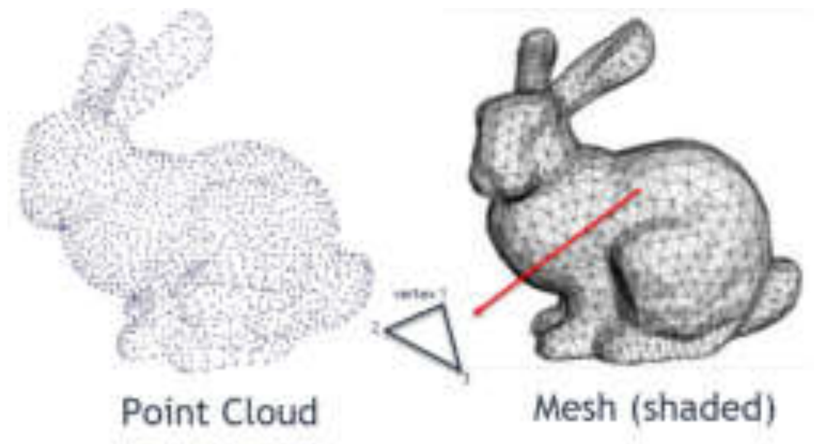


Point Cloud

- Point cloud is not a surface representation
  - where is the surface?
- Point cloud = surface + sampling
  - How to sample point clouds from a mesh surface?

Sampling Strategy: Uniform Sampling

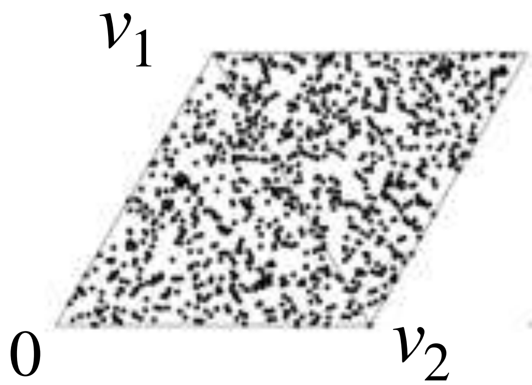
- 1. Compute the areas of each individual face
- 2. Compute the probability of each face and use it as weight
- 3. Independent identically distributed (i.i.d.) sample faces according to the weights
- 4. For each sampled face, uniformly sample from one triangle face



https://static.packt-cdn.com/products/9781788830409/graphics/9ddd542a-fa9b-4410-87c7-cae62c1e49a7.jpg

Uniform Sampling Points in a Triangle

- Special case: for a triangle with one vertex at the origin and the others at positions v_1 and v_2:
- To pick points uniformly distributed inside the triangle, we can do $x = a_1 v_1 + a_2 v_2$, where a_1 and a_2 are uniform variates in the interval $[0,1]$.
- This gives points uniformly distributed in a quadrilateral. The points not in the triangle interior can then be transformed into the corresponding point inside the triangle.

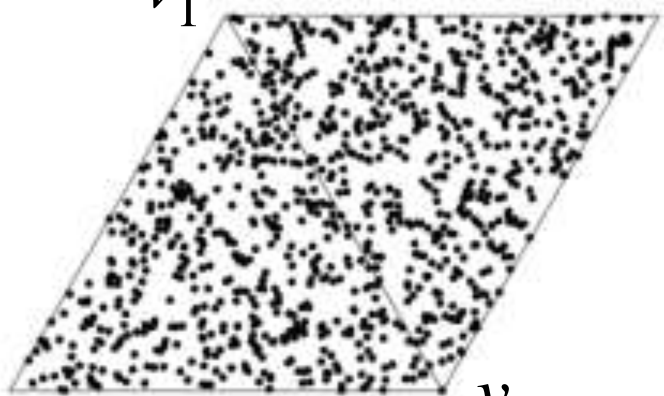


Uniform Sampling Points in a Triangle

- General case: for a triangle with vertices v_1, v_2, v_3:
- $x = v_3 + a_1(v_1 - v_3) + a_2(v_2 - v_3) = a_1 v_1 + a_2 v_2 + (1 - a_1 - a_2) v_3$, where a_1 and a_2 are uniform variates in the interval $[0,1]$

- If $a_1 + a_2 \le 1$, then x will be inside the triangle (or on the edges);
- If $a_1 + a_2 > 1$, then x can be mapped back to the triangle interior via

$$x = (1 - a_1)v_1 + (1 - a_2)v_2 + (a_1 + a_2)v_3$$



Alternative Approach

$$x = (1-\sqrt{r_1})v_1 + \sqrt{r_1}(1-r_2)v_2 + \sqrt{r_1}r_2v_3$$

- Here $r_1, r_2 \sim U(0,1)$.

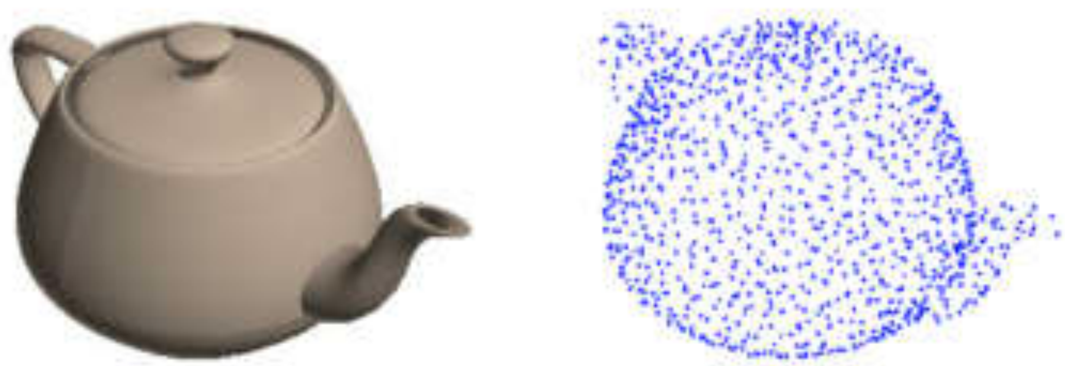
- Proof:
  - If this is true for one triangle, it is true for all triangles, as we can find an affine transformation between them.
  - Use $v_1 = (0,0), v_2 = (1,0), v_3 = (0,1)$
  - Prove x is always inside the triangle.
  - Show that the probability to be within an area of $(0,x) \times (0,y)$ is always $2xy$.

Sampling Strategy: Uniform Sampling

- Usually the easiest to implement
- Issue: Irregularly spaced sampling

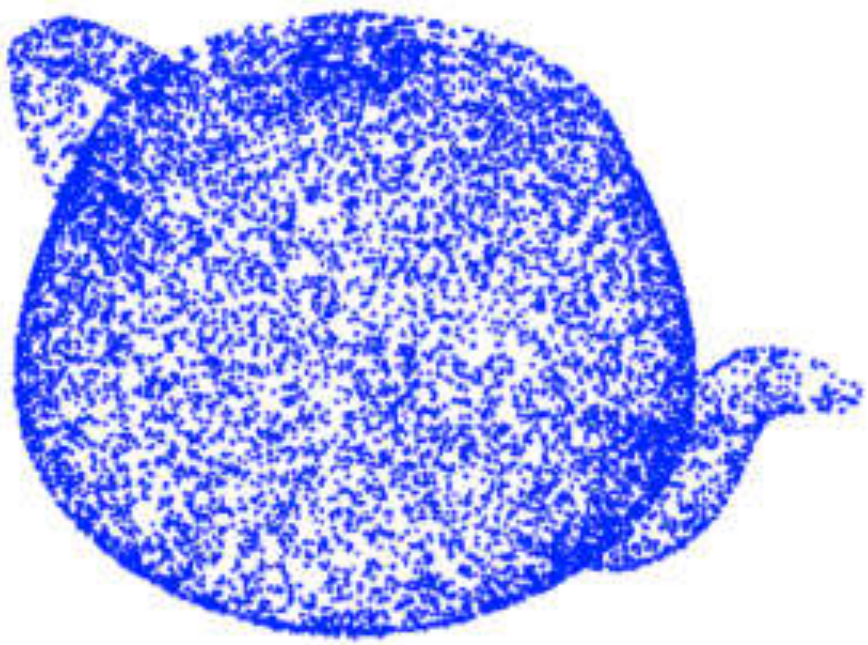
Farthest Point Sampling (FPS)

- Goal: Sampled points are far away from each other
- NP-hard problem
- What is a greedy approximation method?



Iterative Furthest Point Sampling

- Step 1: Over sample the shape by any fast method (e.g., uniformly sample N=10,000 i.i.d. samples)



Iterative Furthest Point Sampling

- Step 2: Iteratively select K points

U is the initial big set of points
$S = \{\}$

add a random point from U to S

for i=1 to K
    find a point $u \in U$ with the largest distance to S
    add u to S

Visualization: Uniform Sampling vs. FPS

- FPS
- Uniform sampling

With the same number of sampled points.

Wang et al. Rethinking Sampling in 3D Point Cloud Generative Adversarial Networks.

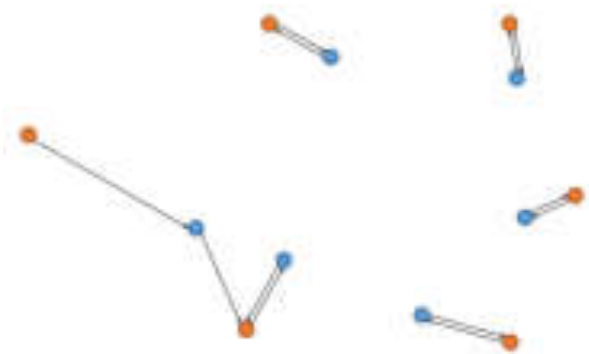
Distance Metrics for Point Cloud

- How to measure the distance between two point clouds?

Distance Metrics for Point Cloud

Chamfer distance We define the Chamfer distance between $S_1, S_2 \subseteq \mathbb{R}^3$ as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$



A Point Set Generation Network for 3D Object Reconstruction from a Single Image, CVPR 2016

Distance Metrics for Point Cloud

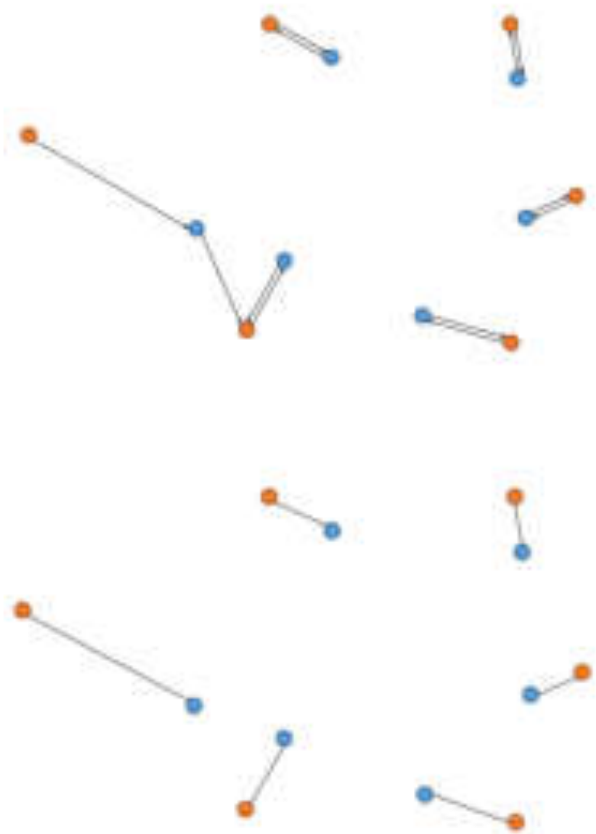
Chamfer distance We define the Chamfer distance between $S_1, S_2 \subseteq \mathbb{R}^3$ as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2$$

Earth Mover's distance Consider $S_1, S_2 \subseteq \mathbb{R}^3$ of equal size $s = |S_1| = |S_2|$. The EMD between A and B is defined as:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

where $\phi : S_1 \to S_2$ is a bijection.



A Point Set Generation Network for 3D Object Reconstruction from a Single Image, CVPR 2016

Distance Metrics for Point Cloud

Chamfer distance We define the Chamfer distance between $S_1, S_2 \subseteq \mathbb{R}^3$ as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

Sum of the closest distances
Insensitive to sampling

Earth Mover's distance Consider $S_1, S_2 \subseteq \mathbb{R}^3$ of equal size $s = |S_1| = |S_2|$. The EMD between A and B is defined as:

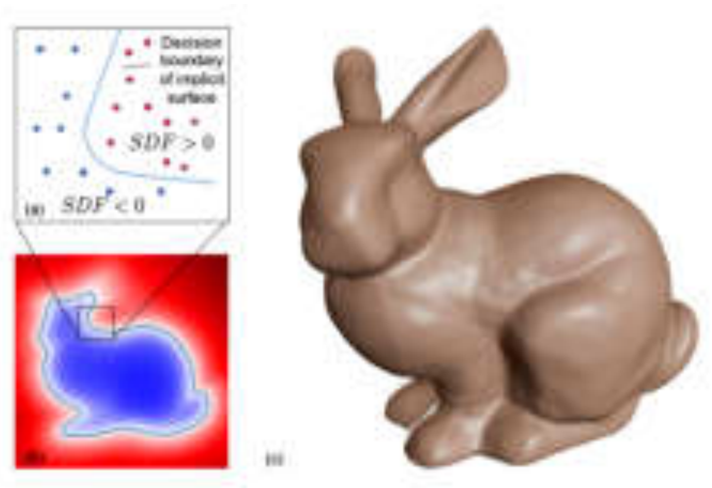
$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$$

where $\phi : S_1 \to S_2$ is a bijection.

Sum of the matched closest distances
Sensitive to sampling

Implicit Field

Implicit Shape

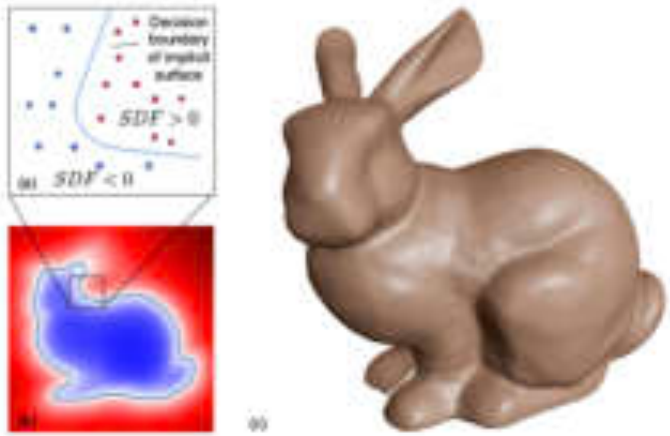


SDF

- Both an implicit geometry and surface representation
- Can convert into mesh
- Signed distance function, unsigned distance function, occupancy network

Signed Distance Function (SDF)

- Interior: $F(x, y, z) < 0$
- Exterior: $F(x, y, z) > 0$
- Surface: $F(x, y, z) = 0$ (zero set, zero iso-surface)
- Example implementation:
  - SDF: $F(x, y, z) =$ distance to the surface

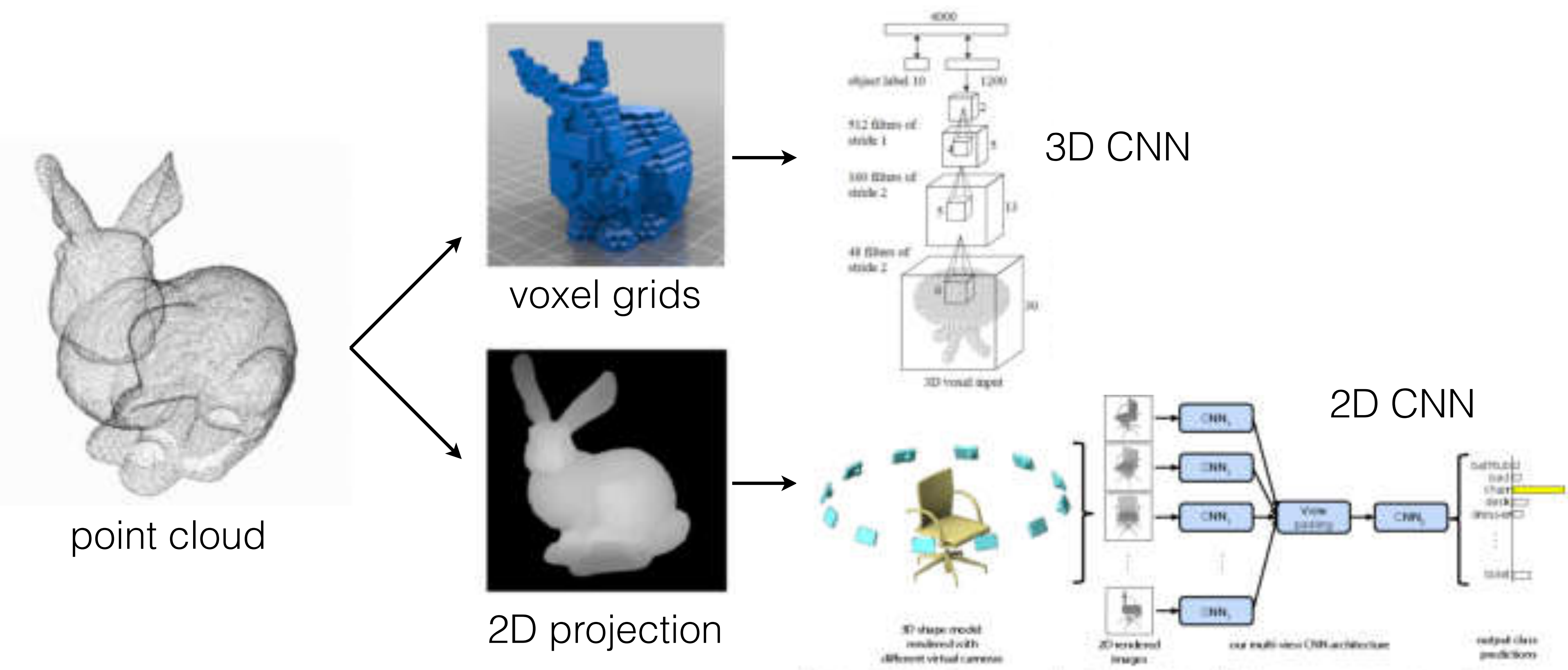


3D Deep Learning

Outline

- Point Networks
  - PointNet
  - PointNet++
- Voxel Networks
- Networks for other representations
  - SDF
  - Mesh

Straightforward Ways of Processing Point Clouds



Regular Image Grid vs. Irregular 3D Point Cloud

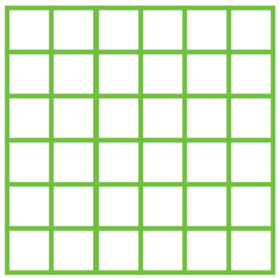
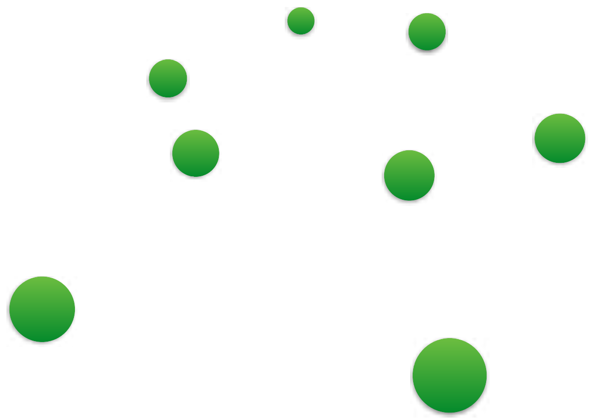


Image grid



Point cloud

Regular Image Grid vs. Irregular 3D Point Cloud

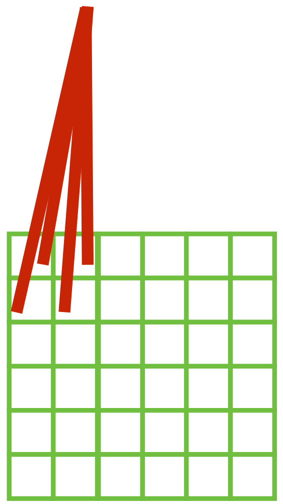
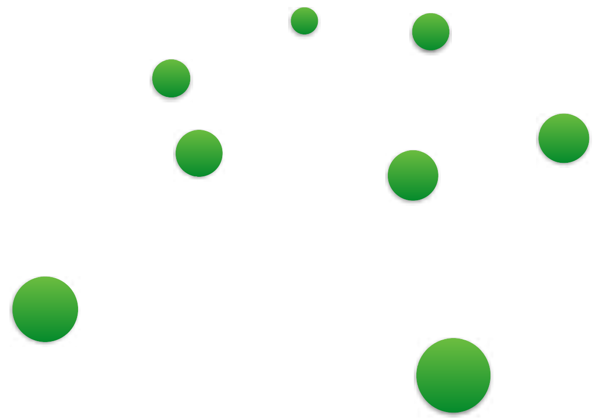


Image grid



Point cloud

Regular Image Grid vs. Irregular 3D Point Cloud

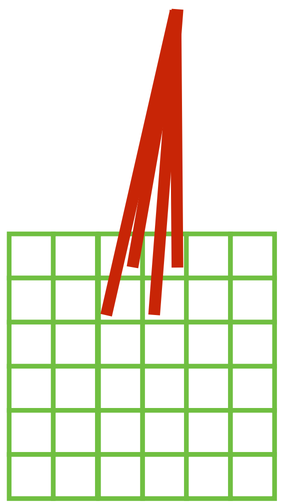
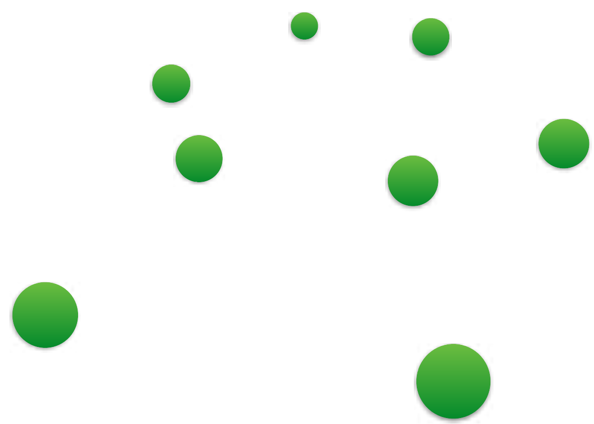


Image grid



Point cloud

Regular Image Grid vs. Irregular 3D Point Cloud

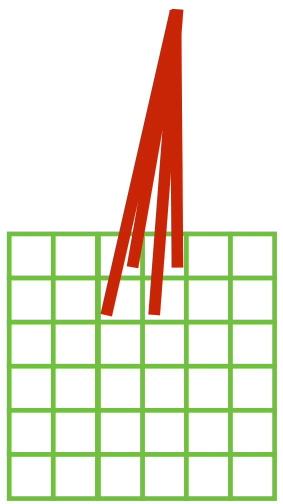
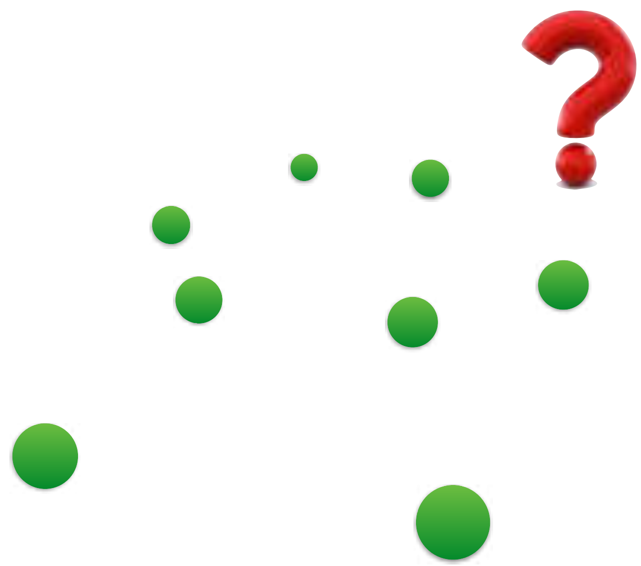


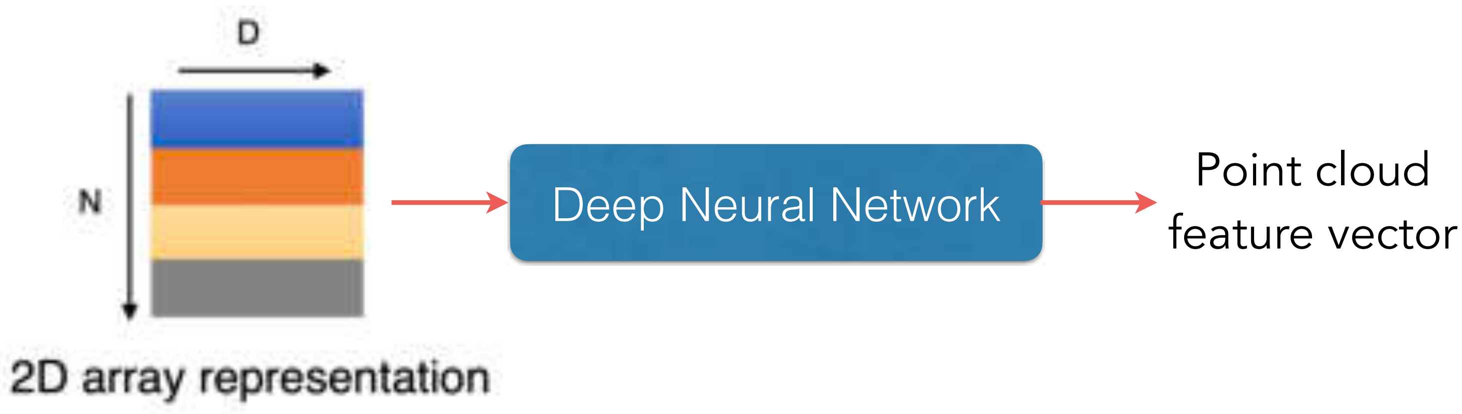
Image grid



Point cloud

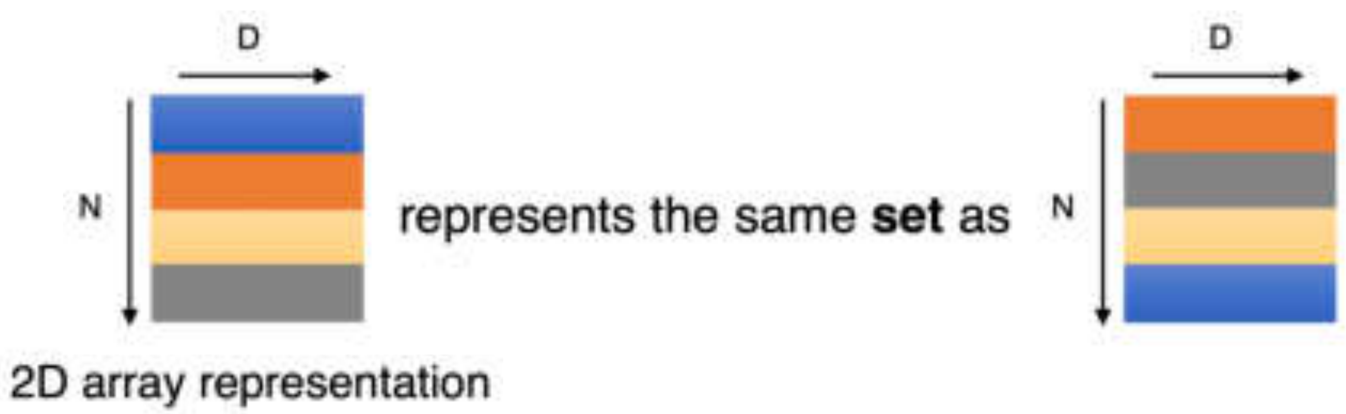
Unordered Inputs

- Point cloud: N orderless points, each represented by a D dim coordinate



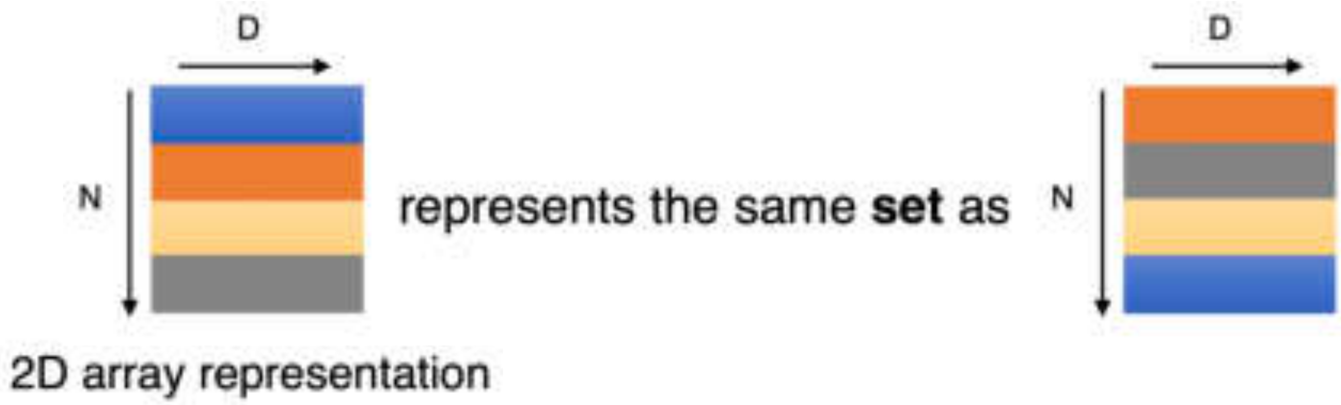
Unordered Inputs

- Point cloud: N orderless points, each represented by a D dim coordinate



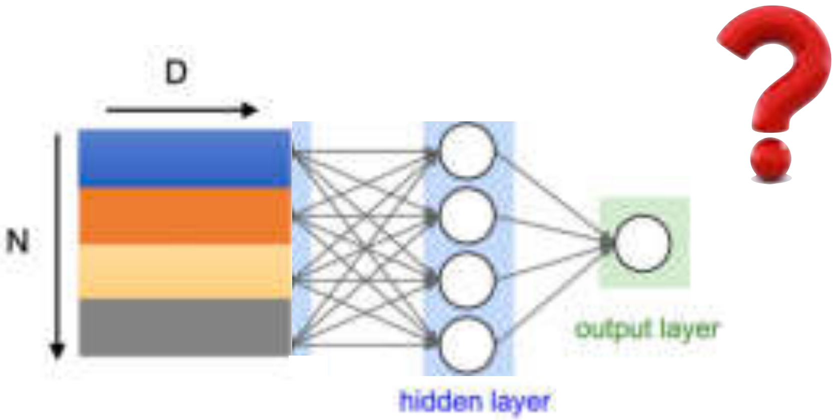
Desired Properties of a Point Cloud Network

- Point cloud: N orderless points, each represented by a D dim coordinate

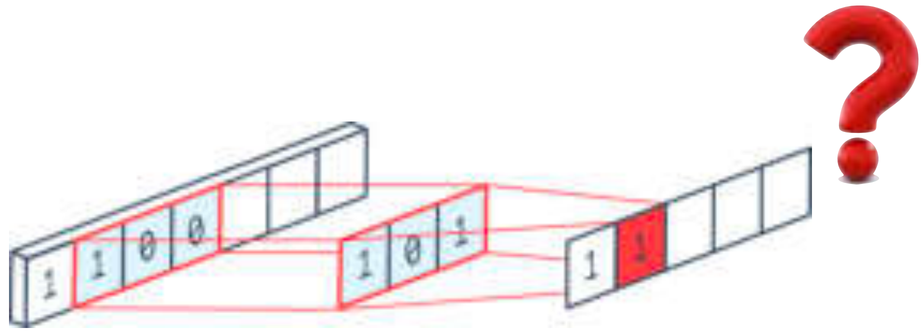


- Deep net needs to be invariant to N! permutations

Permutation Invariance



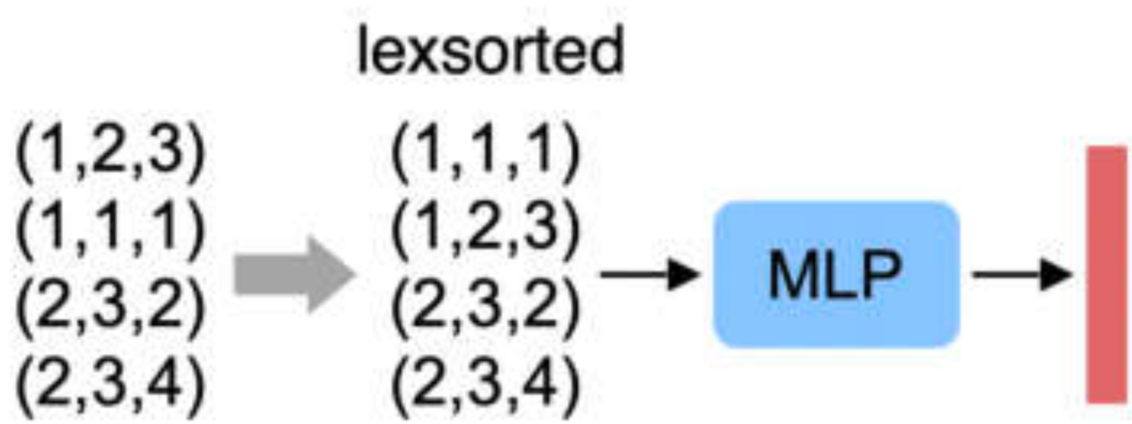
Fully connected network



1D convolutional network

Slide credit: Charles Qi

Permutation Invariance — Sorting?



Not a good idea! Adding one point will change the order dramatically!

Slide credit: Charles Qi

Permutation Invariance: Symmetric Function

$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

Slide credit: Charles Qi

Permutation Invariance: Symmetric Function

$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

Examples:

$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$

$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

$\ldots$

Slide credit: Charles Qi

Permutation Invariance: Symmetric Function

$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

Examples:

$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$

$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

...

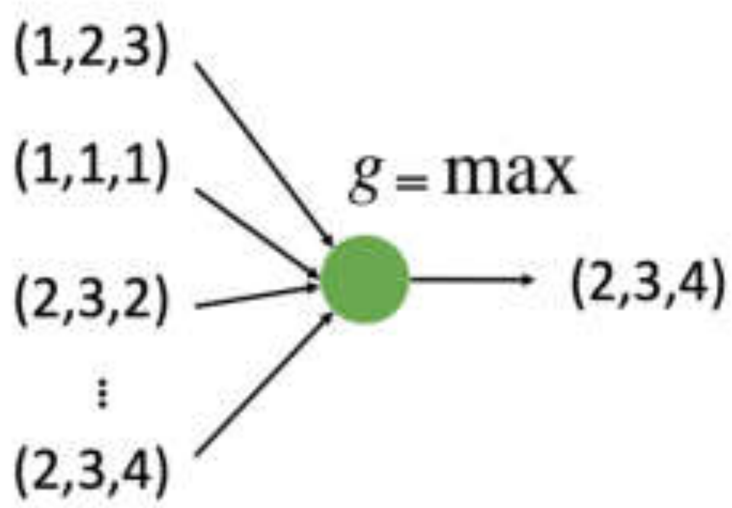
How can we construct a universal family of symmetric functions by neural networks?

Slide credit: Charles Qi

Construct Symmetric Functions by Neural Networks

Simplest form: directly aggregate all points with a symmetric operator g

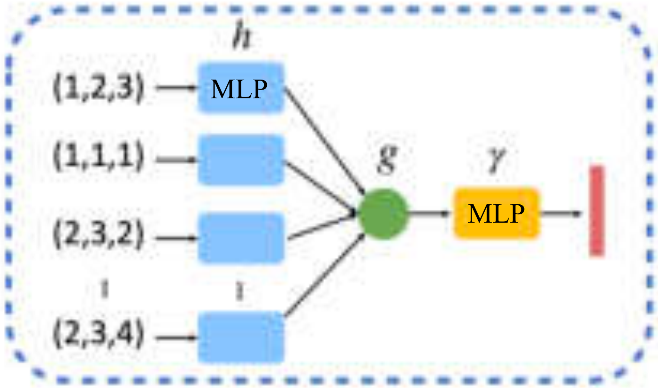
Just discovers simple extreme/aggregate properties of the geometry.



Slide credit: Charles Qi

Construct Symmetric Functions by Neural Networks

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

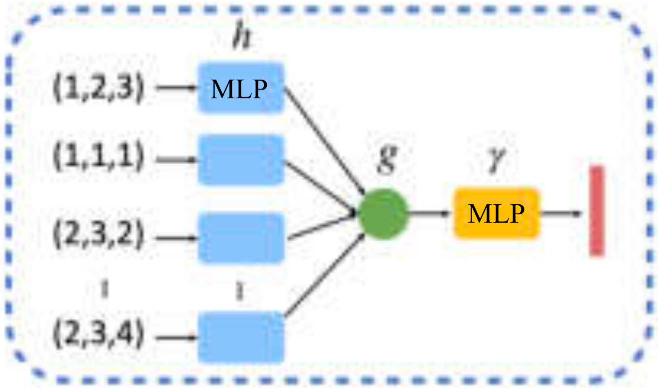


PointNet (vanilla)

Slide credit: Charles Qi

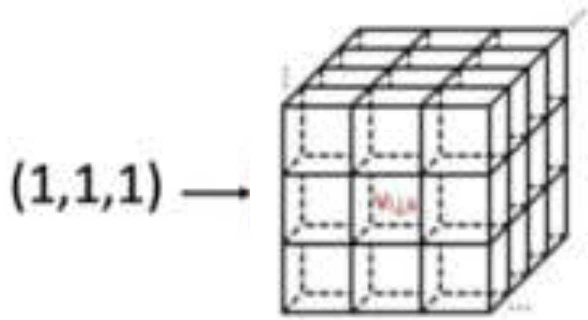
Construct Symmetric Functions by Neural Networks

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



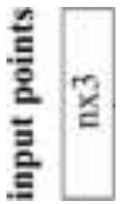
PointNet (vanilla)

- Reflection: assuming g is a max operation, construct a function h where geometric details get kept after applying g.

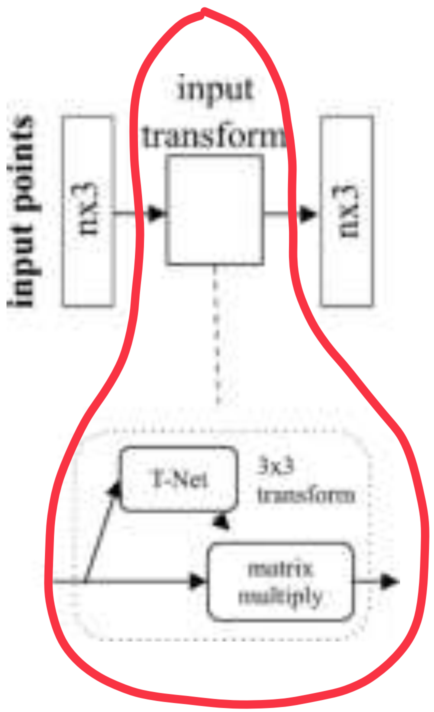


Spatial Hashing Function

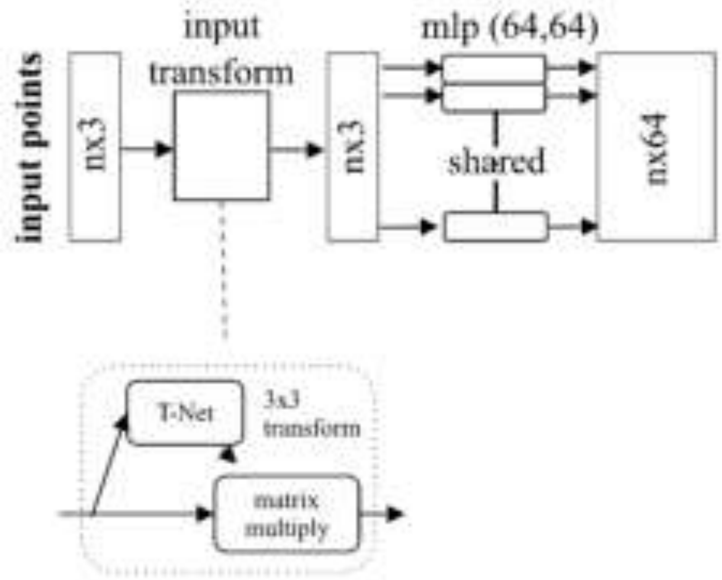
A Detailed Implementation of PointNet



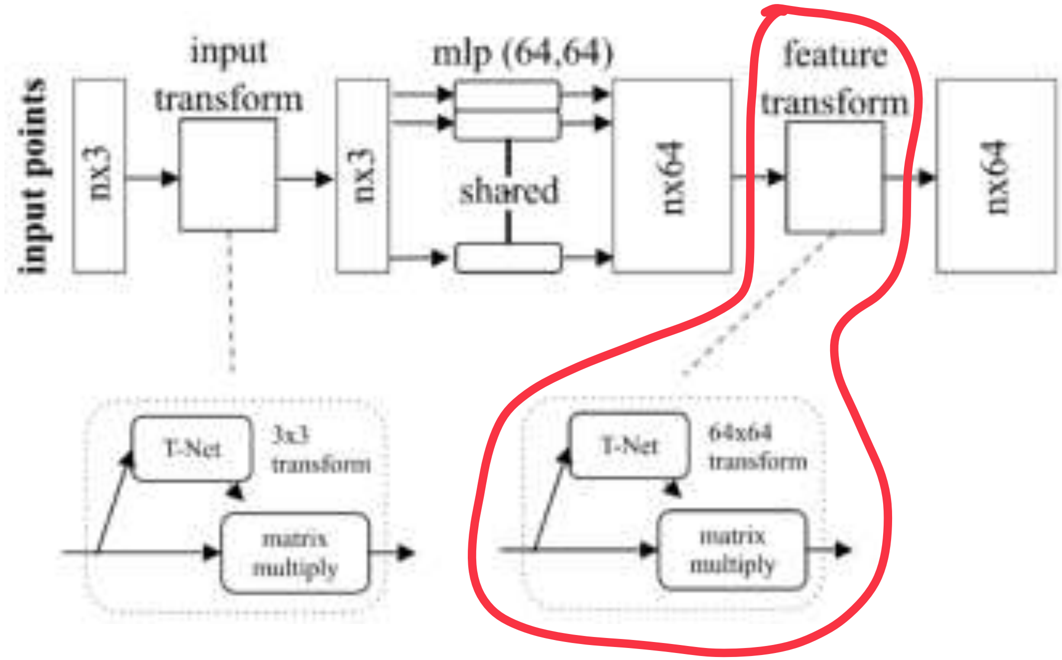
A Detailed Implementation of PointNet



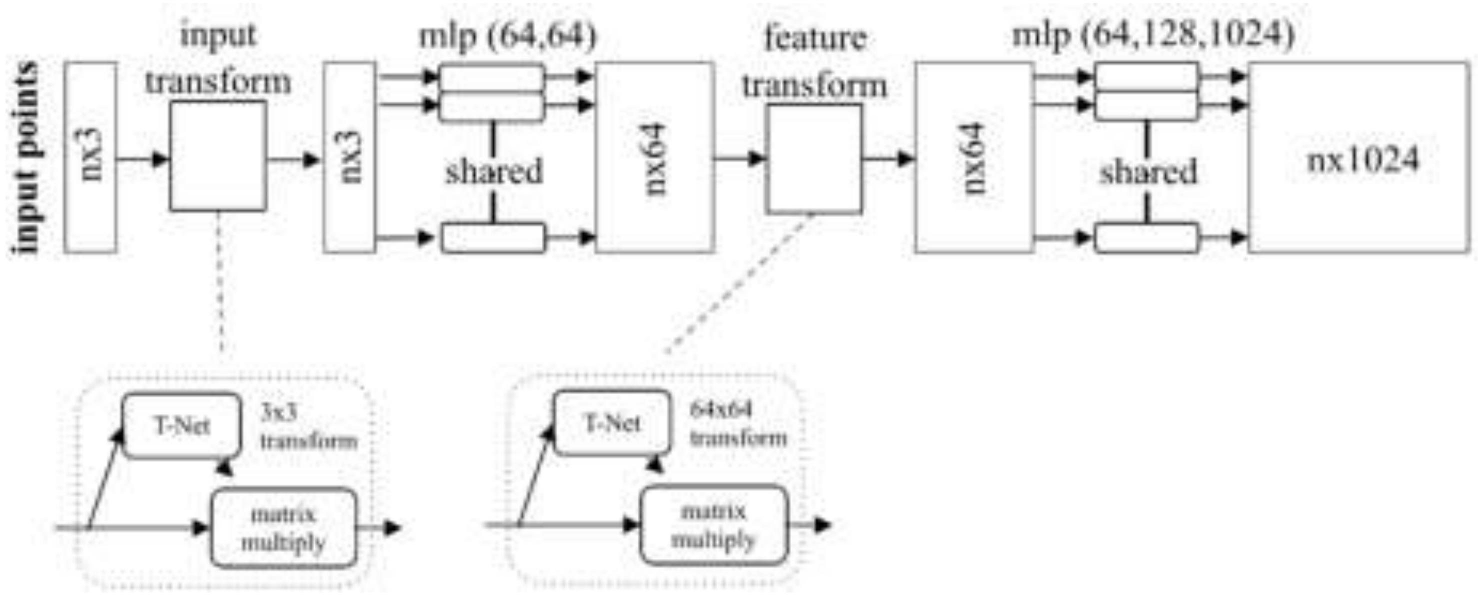
A Detailed Implementation of PointNet



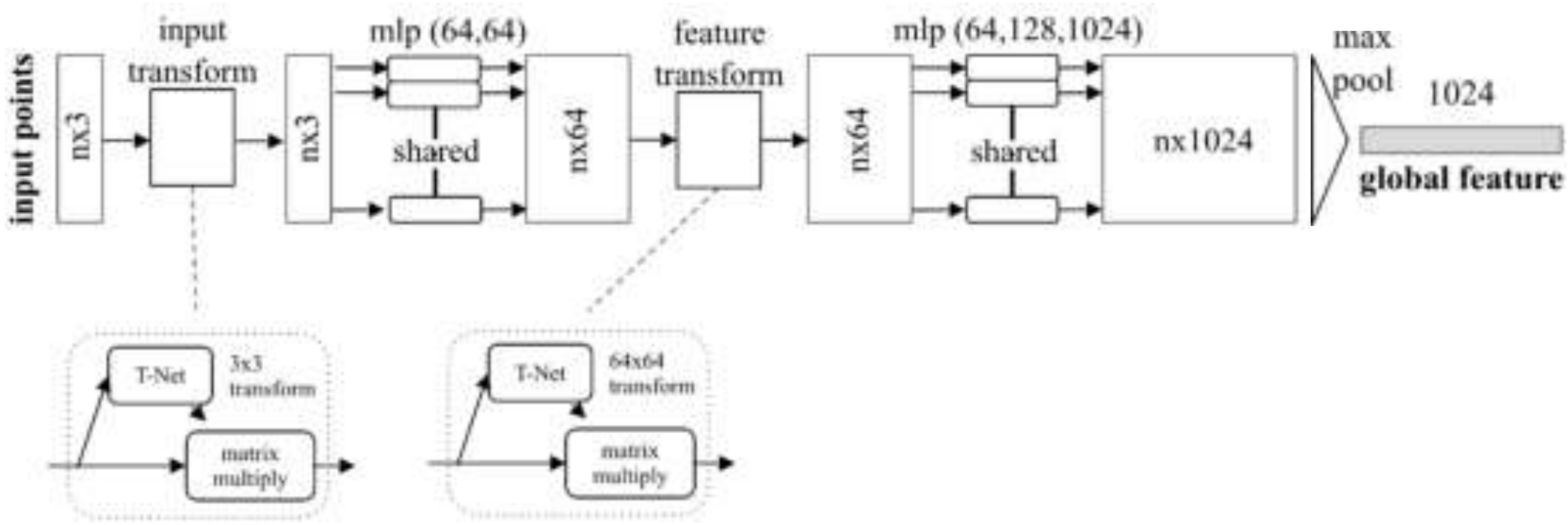
A Detailed Implementation of PointNet



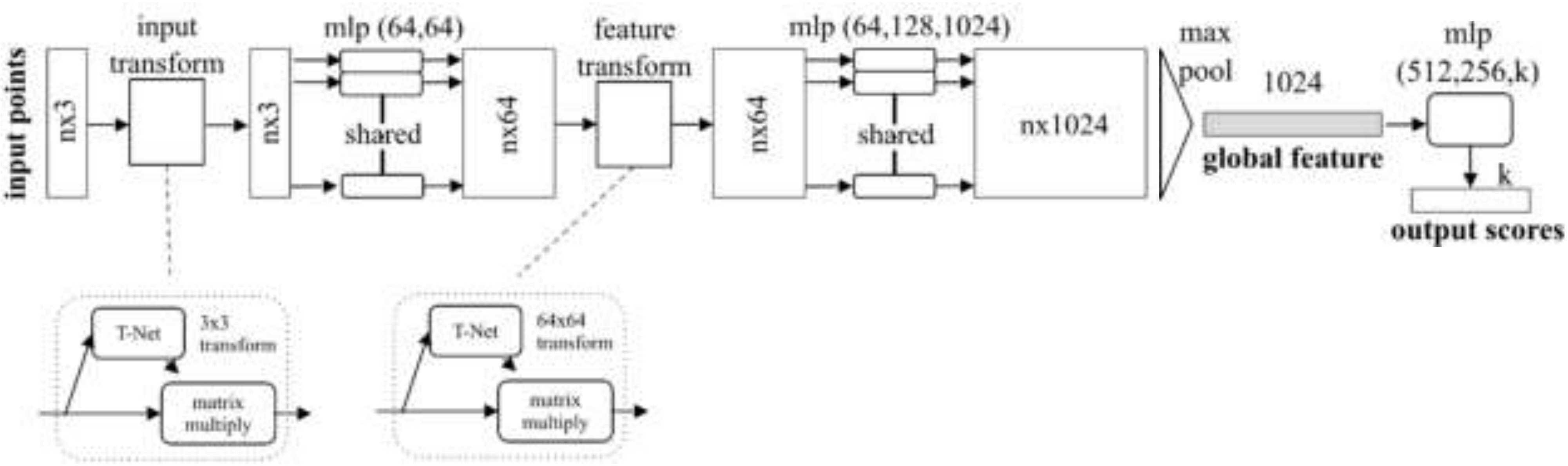
A Detailed Implementation of PointNet



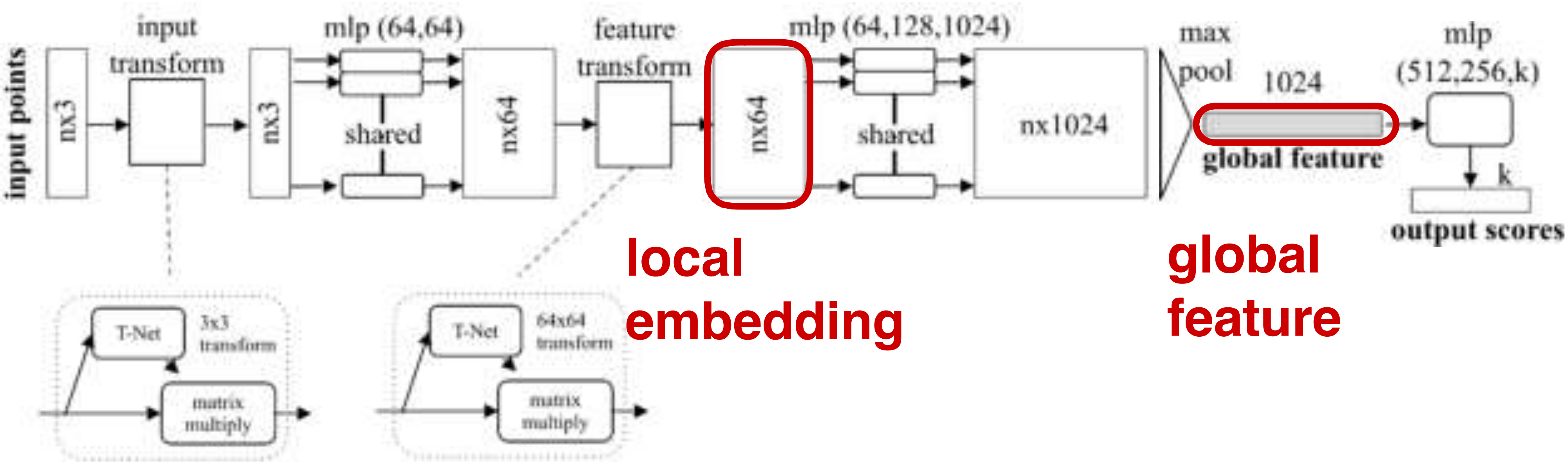
A Detailed Implementation of PointNet



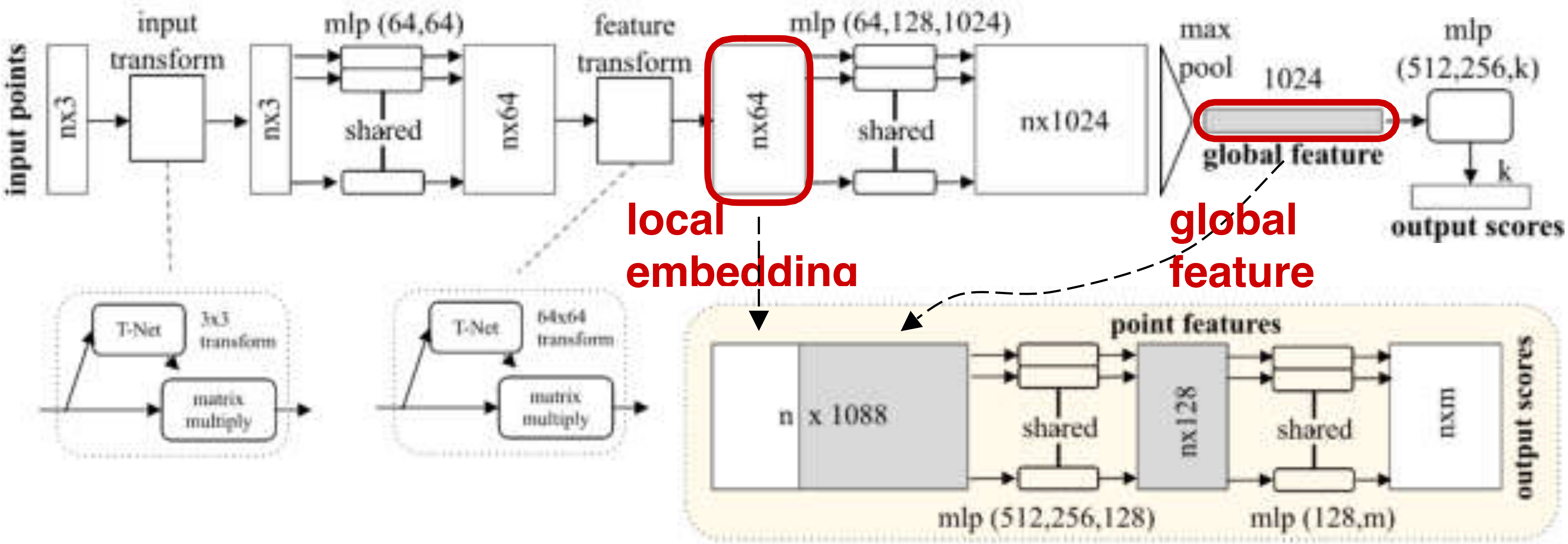
A Detailed Implementation of PointNet



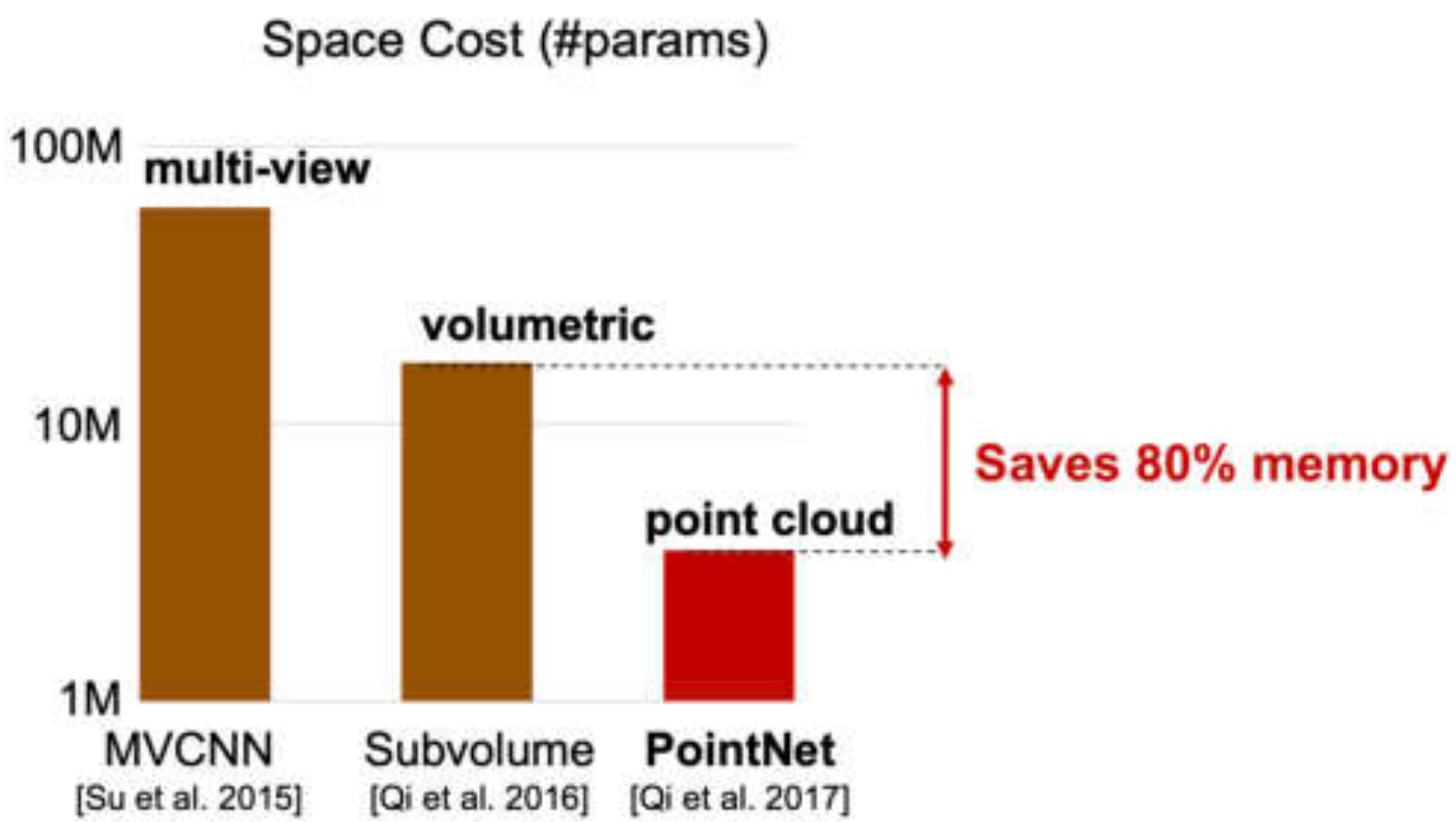
Extension to Segmentation



Extension to Segmentation

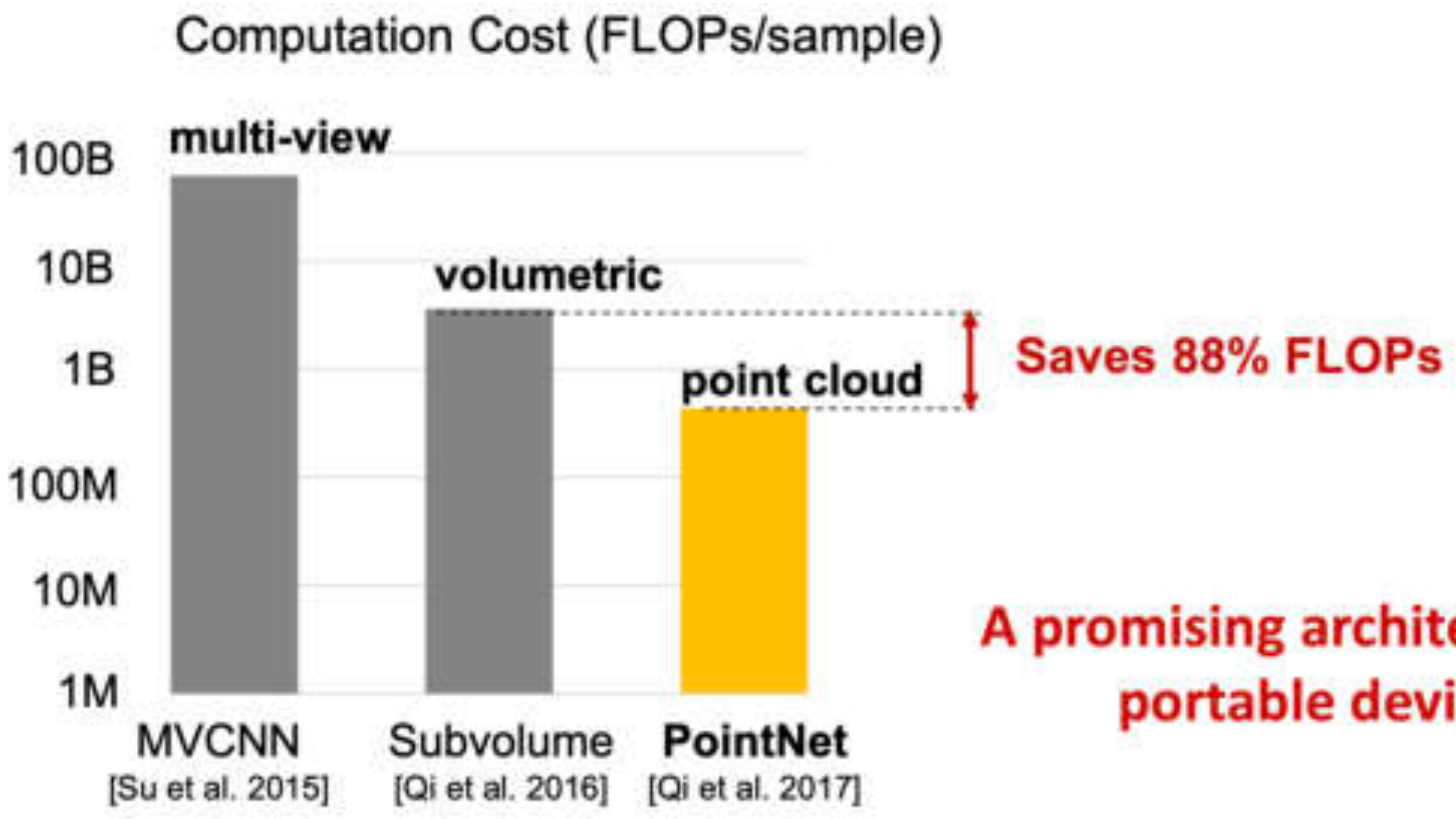


PointNet is Light-Weight and Fast



Slide credit: Charles Qi

PointNet is Light-Weight and Fast

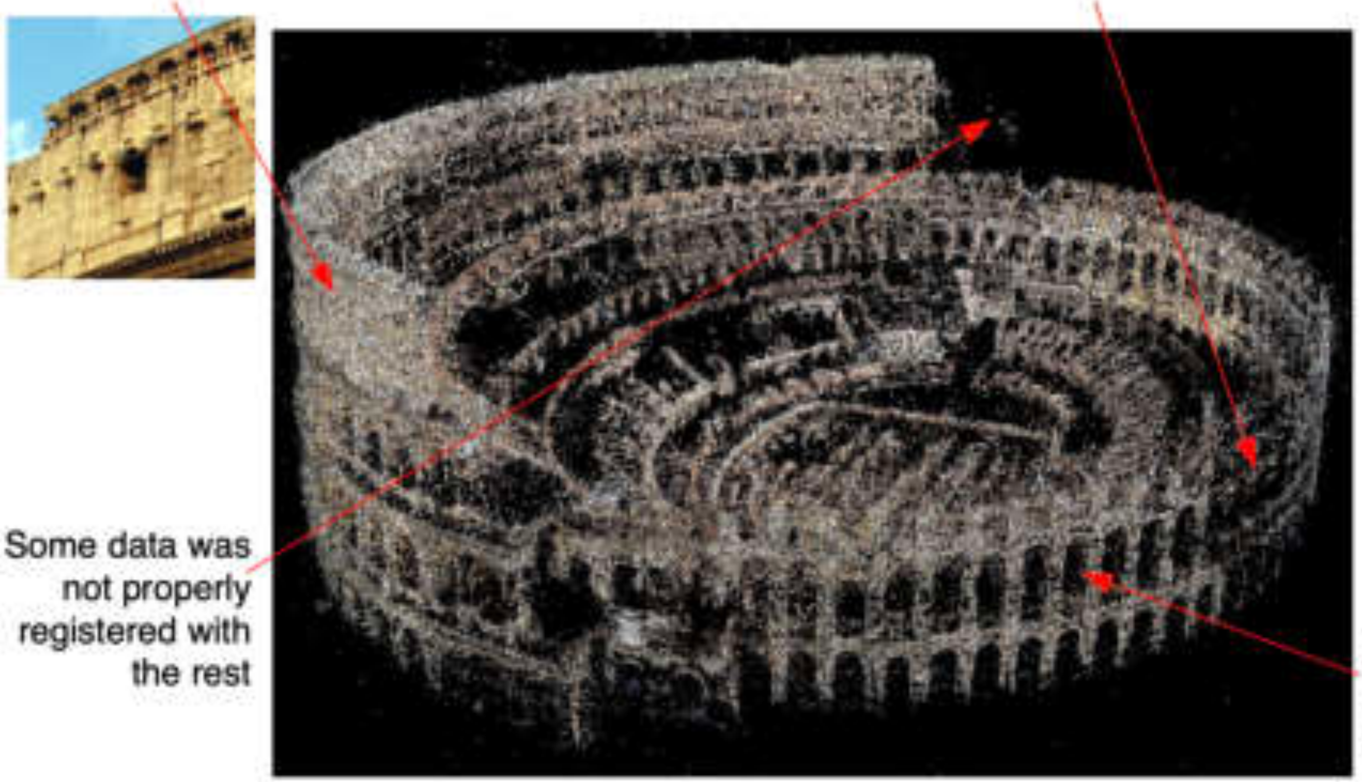


A promising architecture for portable devices!

Slide credit: Charles Qi

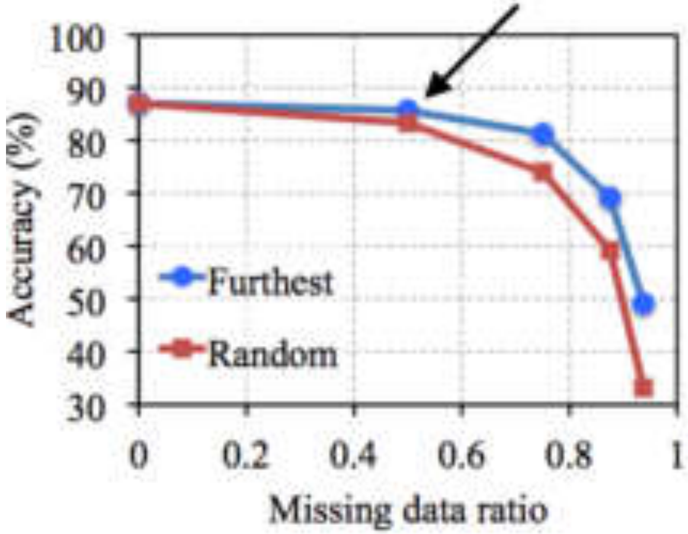
Robustness to Data Corruption

- Many challenges
  - Resolution
  - Occlusion
  - Noise
  - Registration



Robustness to Data Corruption

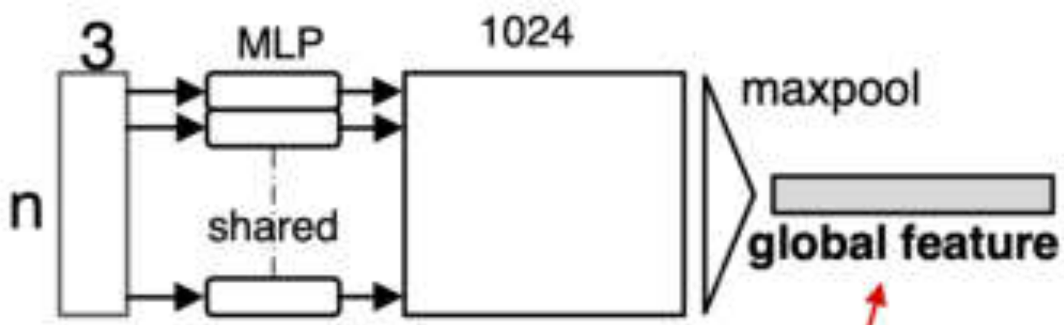
Less than 2% accuracy drop with 50% missing data



dataset: ModelNet40; metric: 40-class classification accuracy (%)

Slide credit: Charles Qi

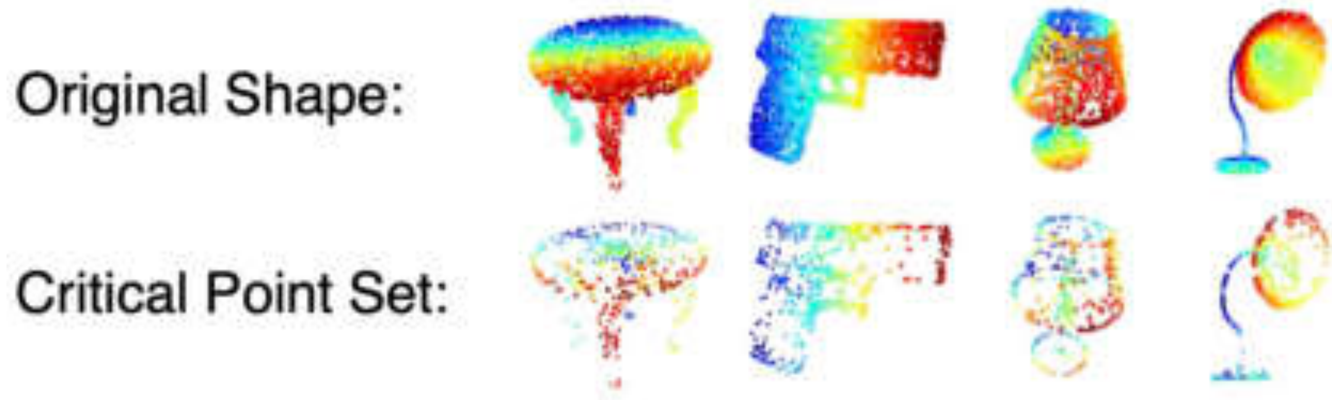
Visualizing Global Point Cloud Features



Which input points are contributing to the global feature?
(critical points)

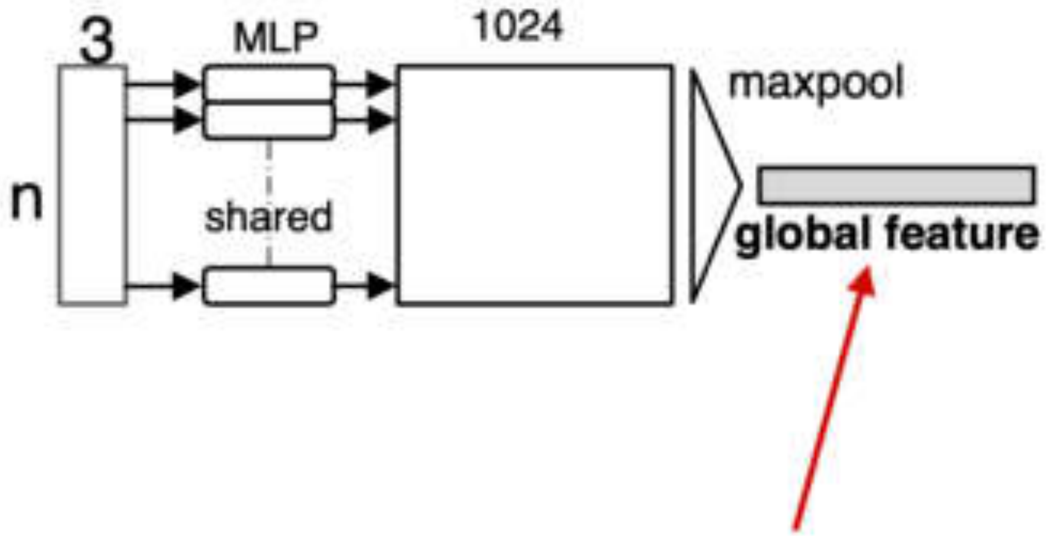
Slide credit: Charles Qi

Visualizing Global Point Cloud Features



Slide credit: Charles Qi

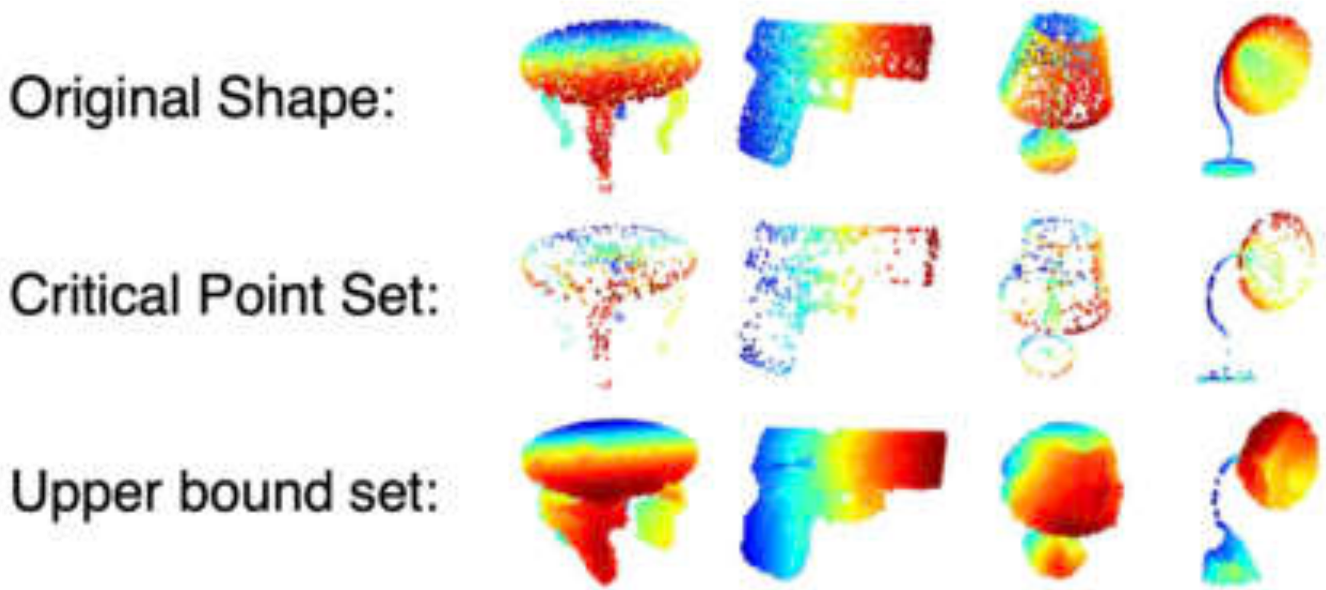
Visualizing Global Point Cloud Features



Which points won't affect the global feature?

Slide credit: Charles Qi

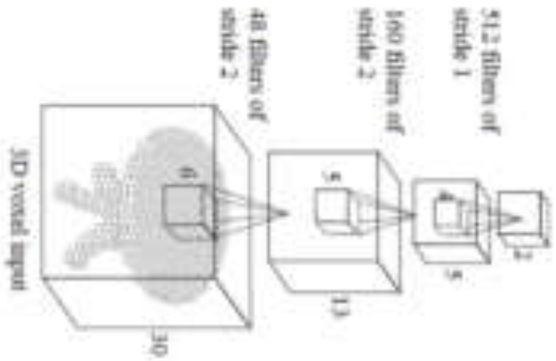
Visualizing Global Point Cloud Features



Slide credit: Charles Qi

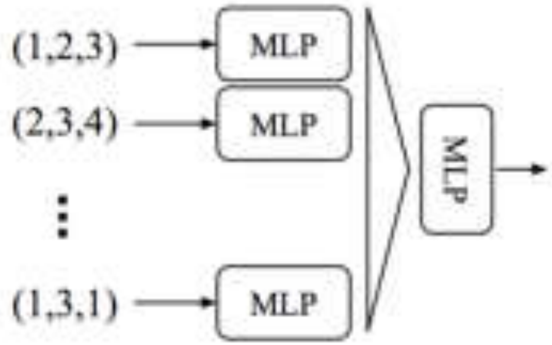
Limitation of PointNet

Hierarchical feature learning
Multiple levels of abstraction



3D CNN (Wu et al.)

Global feature learning
Either one point or all points



PointNet (vanilla) (Qi et al.)

- No local context for each point
- Global feature depends on absolute coordinate. Hard to generalize to unseen scene configurations!

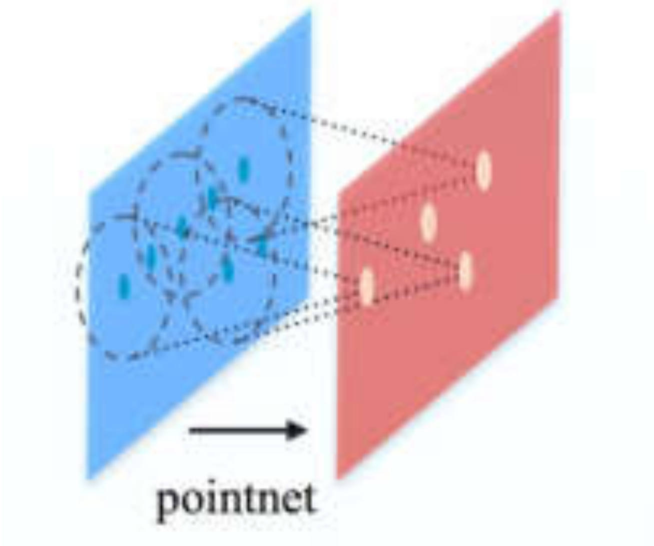
Outline

- Point Networks
  - PointNet
  - PointNet++
- Voxel Networks
- Networks for other representations
  - SDF
  - Mesh

PointNet++

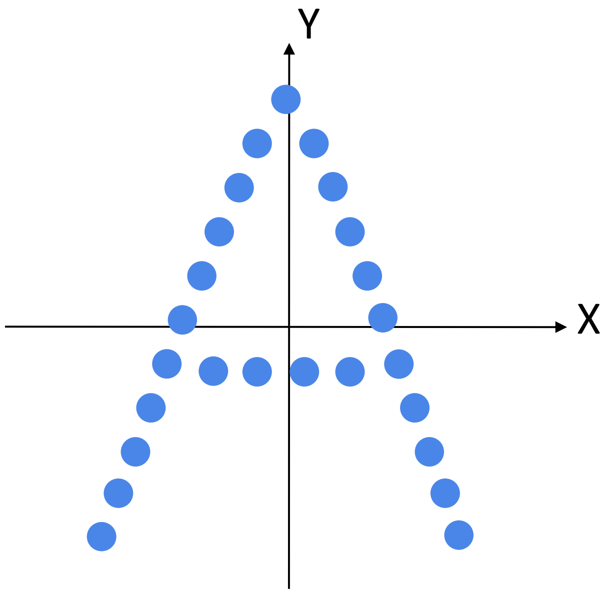
Basic idea: Recursively apply pointnet at local regions.

- ✓ Hierarchical feature learning
- ✓ Local translation invariance
- ✓ Permutation invariance



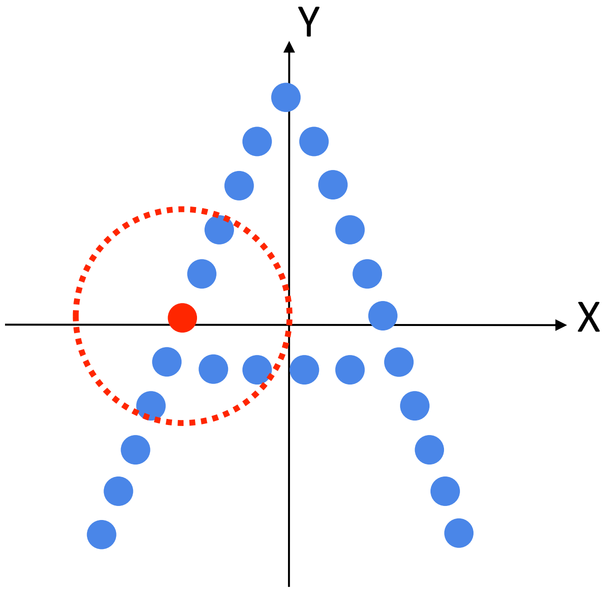
Charles R. Qi, Li Yi, Hao Su, Leonidas Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space (NIPS'17)

Hierarchical Point Feature Learning

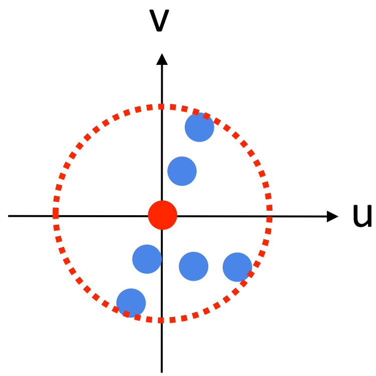


N points in (X,Y)

Hierarchical Point Feature Learning

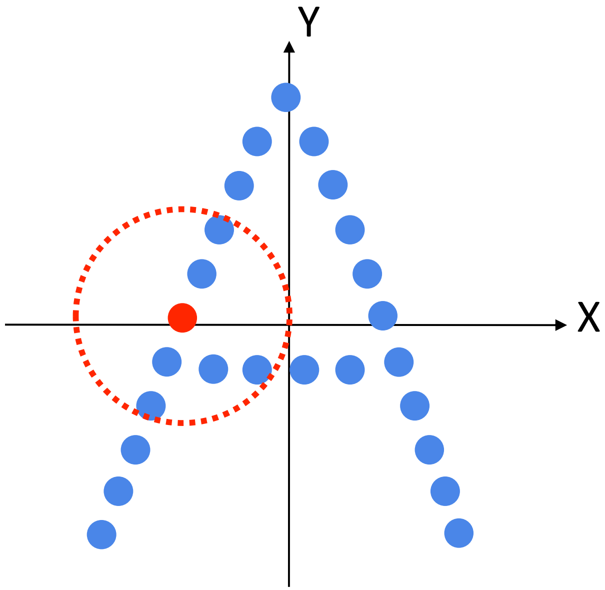


N points in (X,Y)



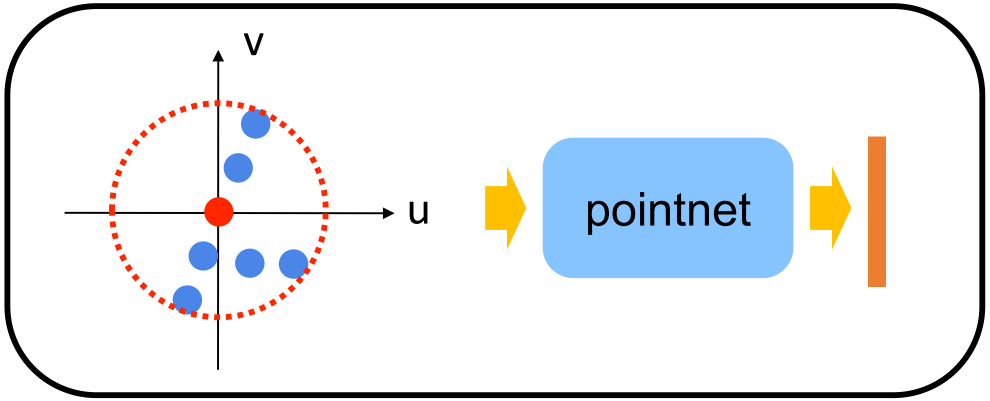
k points in local coordinates (u,v)

Hierarchical Point Feature Learning



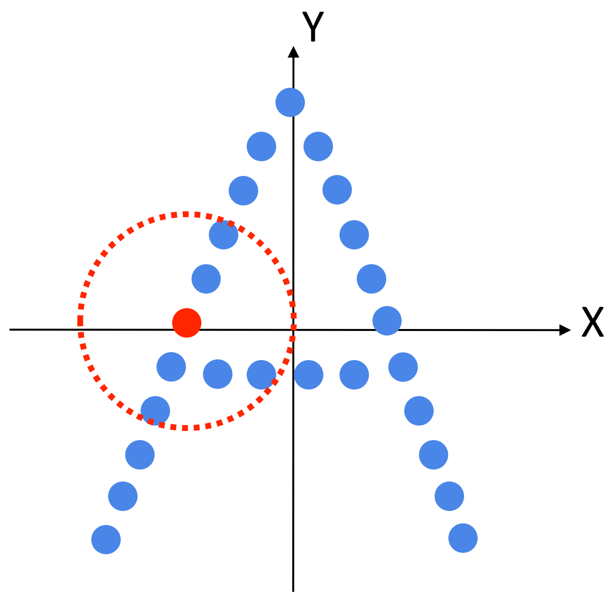
N points in (X,Y)

Apply pointnet at a local region

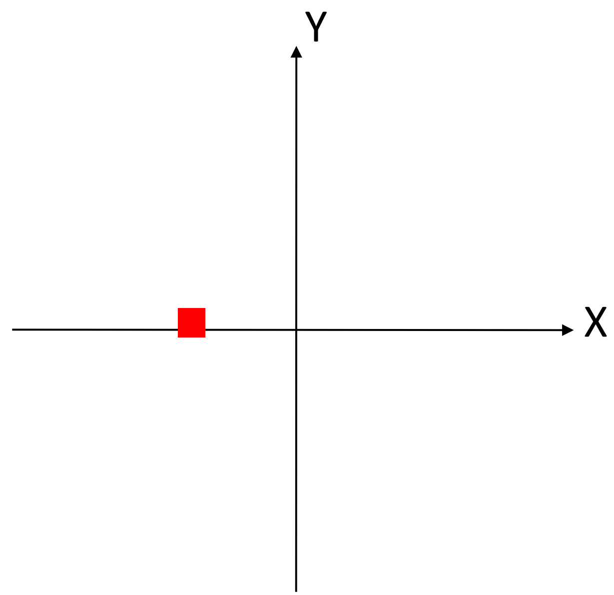


k points in local
coordinates (u,v)

Hierarchical Point Feature Learning



N points in (X,Y)

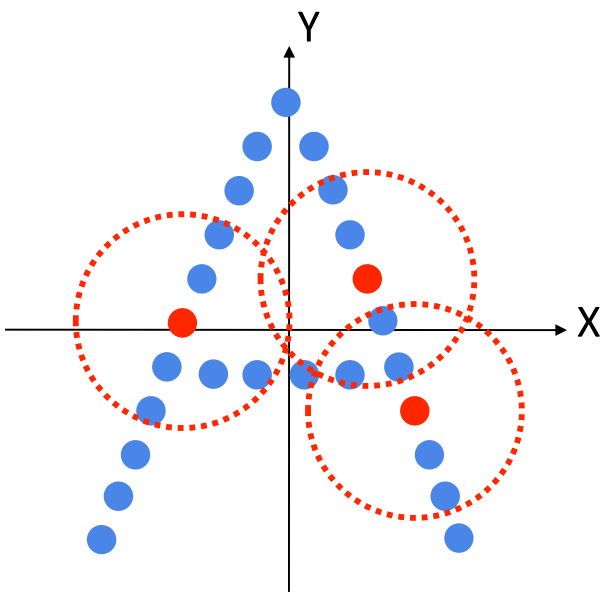


points in (X,Y, **F**)

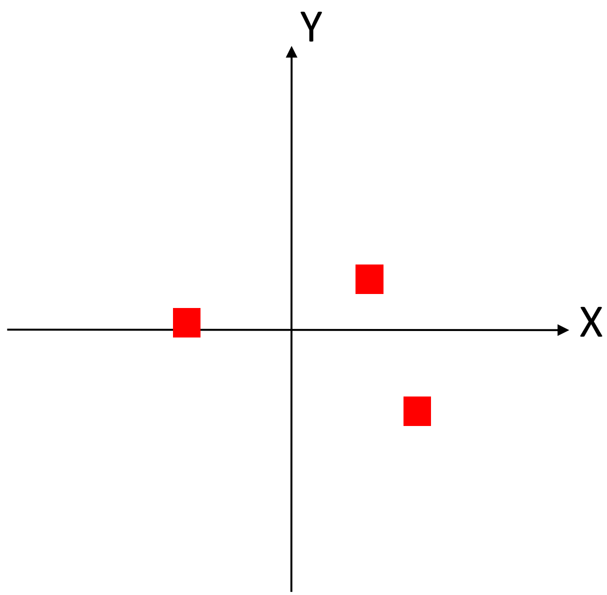
Euclidean space

high-dim feature space

Hierarchical Point Feature Learning

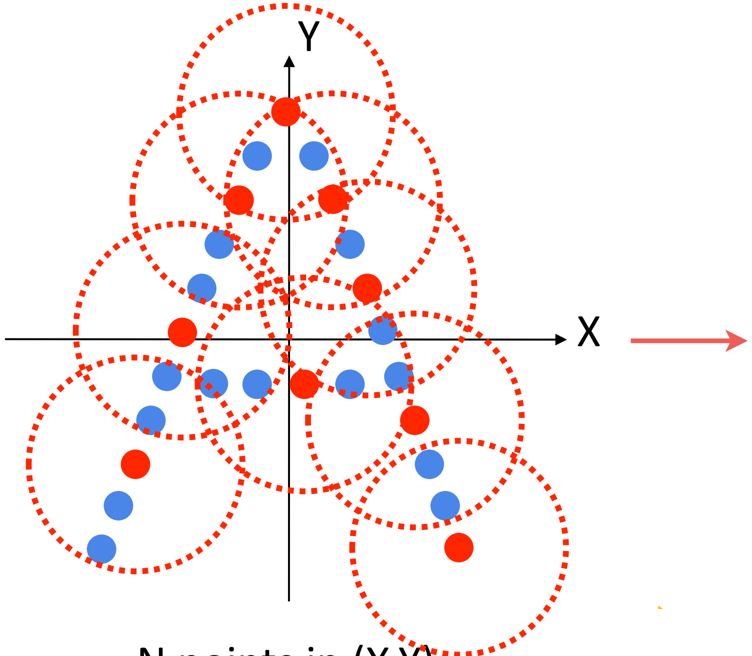


N points in (X,Y)

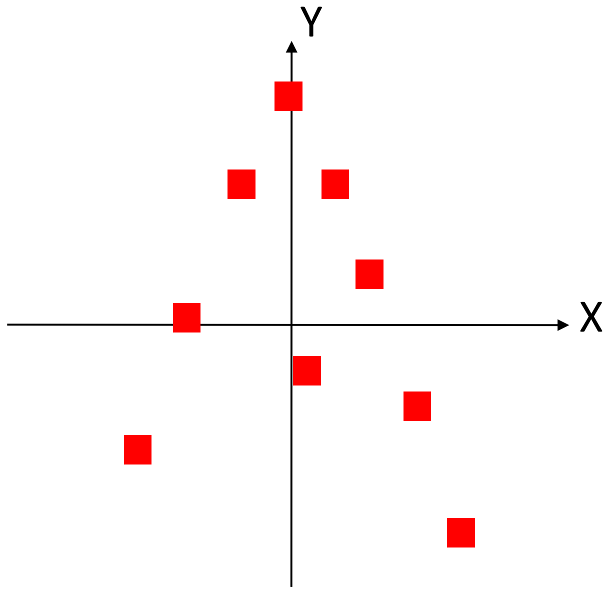


points in (X,Y, **F**)

Hierarchical Point Feature Learning

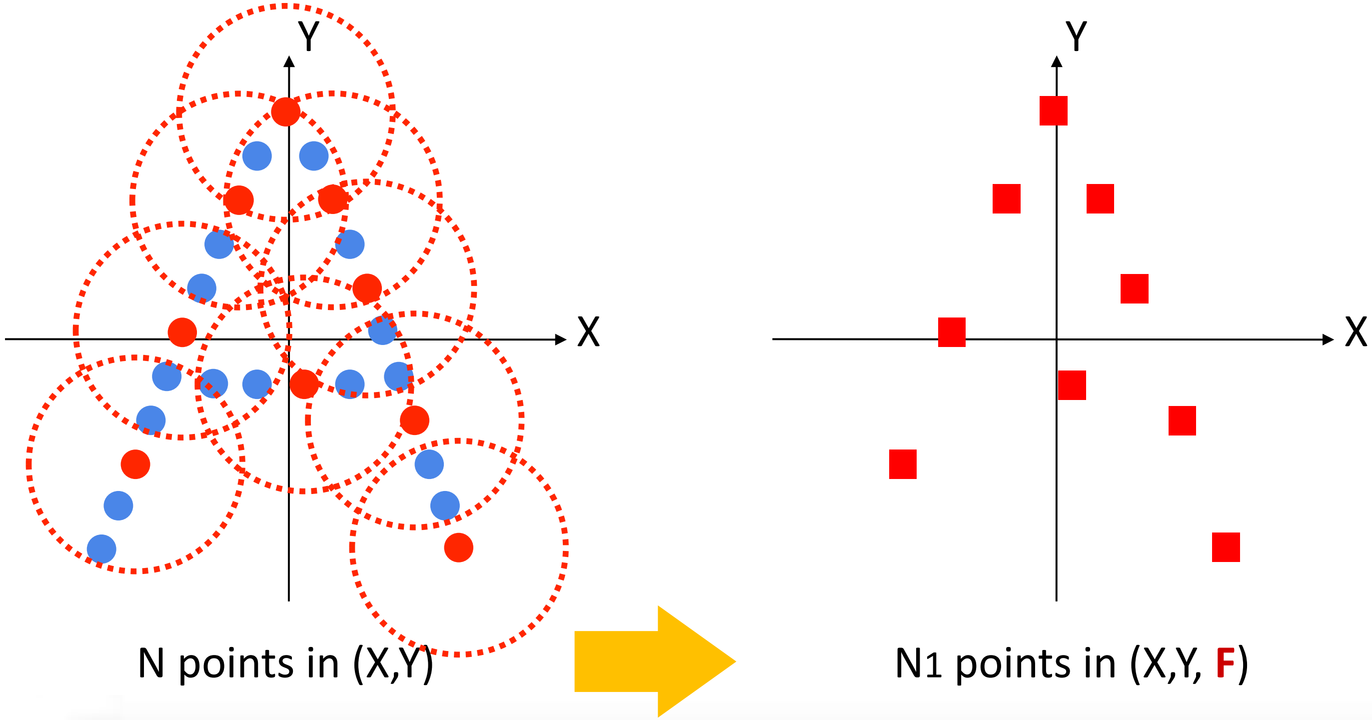


N points in (X,Y)



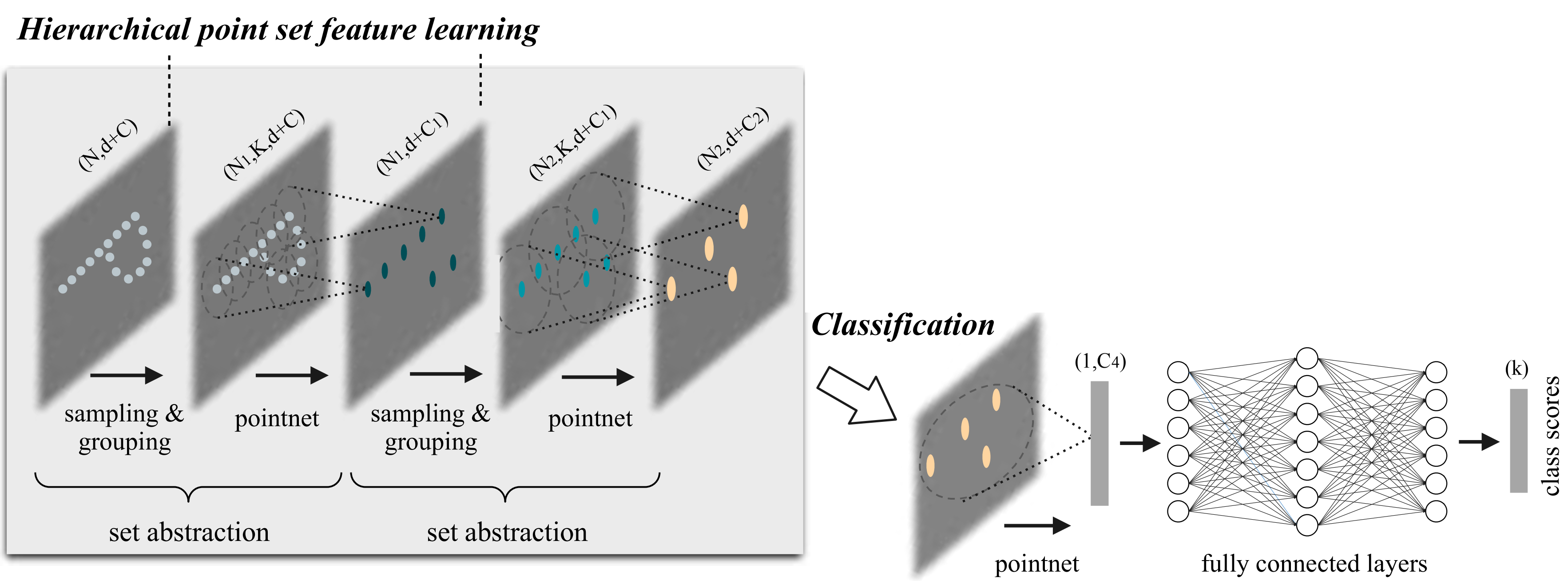
N_1 points in (X,Y, **F**)

Hierarchical Point Feature Learning

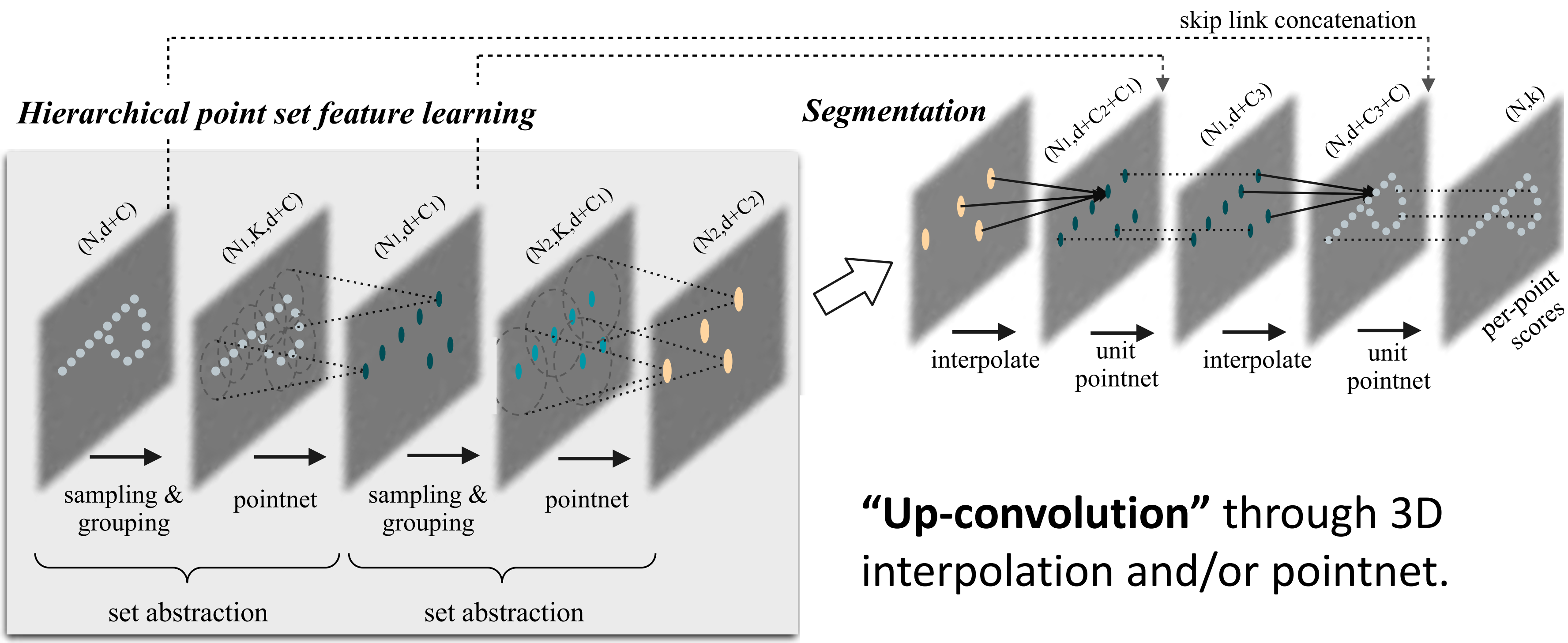


Set Abstraction: farthest point sampling + grouping + pointnet

PointNet++ for Classification



PointNet++ for Segmentation



“Up-convolution” through 3D interpolation and/or pointnet.

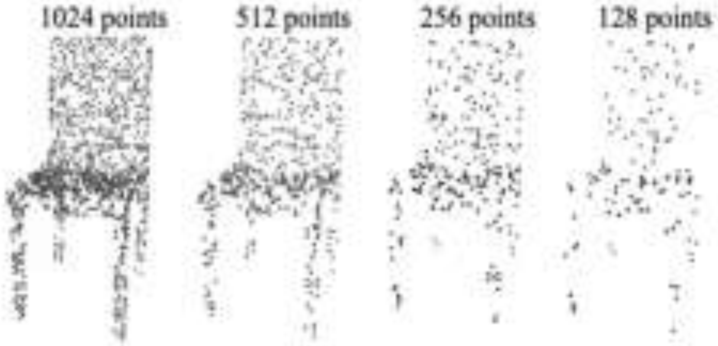
PointNet++: Classificiation

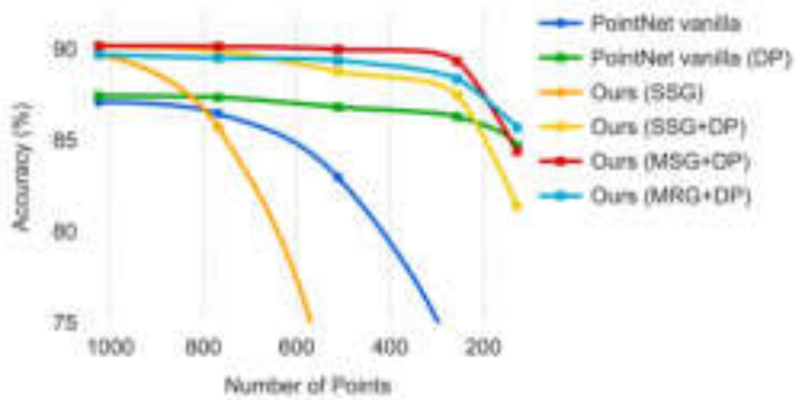
Method	Error rate (%)
Multi-layer perceptron [24]	1.60
LeNet5 [11]	0.80
Network in Network [13]	**0.47**
PointNet (vanilla) [20]	1.30
PointNet [20]	0.78
Ours	0.51

Table 1: MNIST digit classification.

Method	Input	Accuracy (%)
Subvolume [21]	vox	89.2
MVCNN [26]	img	90.1
PointNet (vanilla) [20]	pc	87.2
PointNet [20]	pc	89.2
Ours	pc	90.7
Ours (with normal)	pc	**91.9**

Table 2: ModelNet40 shape classification.





PointNet++: Segmentation

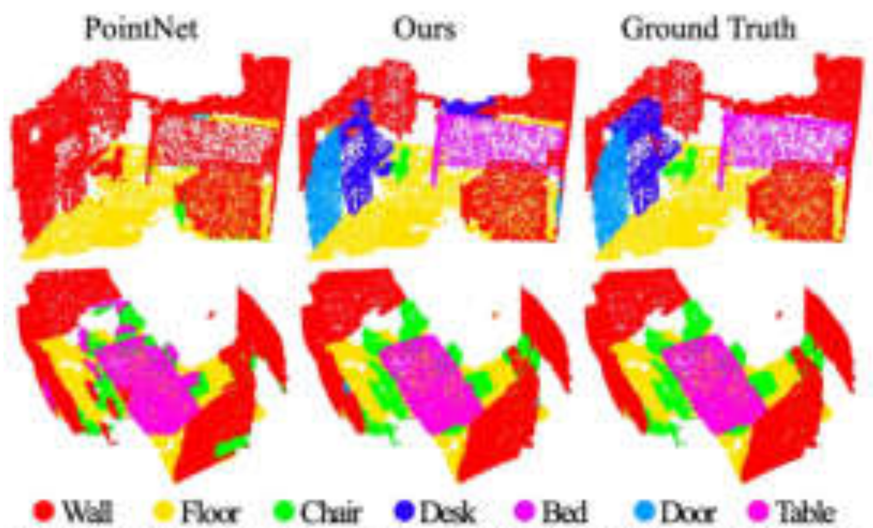


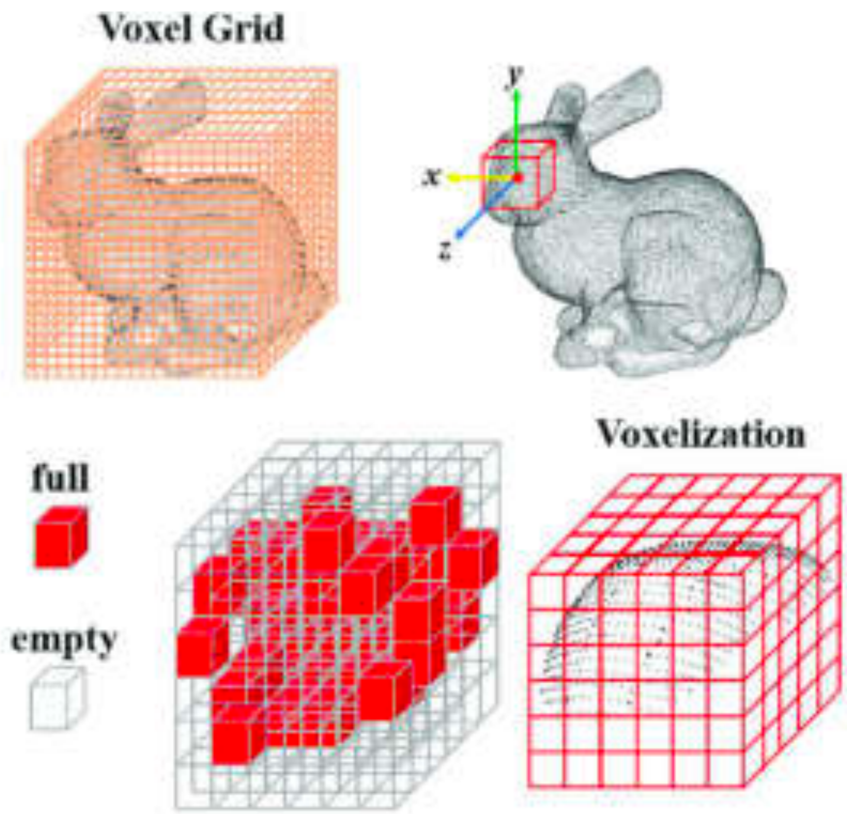
Figure 6: Scannet labeling results. [20] captures the overall layout of the room correctly but fails to discover the furniture. Our approach, in contrast, is much better at segmenting objects besides the room layout.

Outline

- Point Networks
  - PointNet
  - PointNet++
- Voxel Networks
- Networks for other representations
  - SDF
  - Mesh

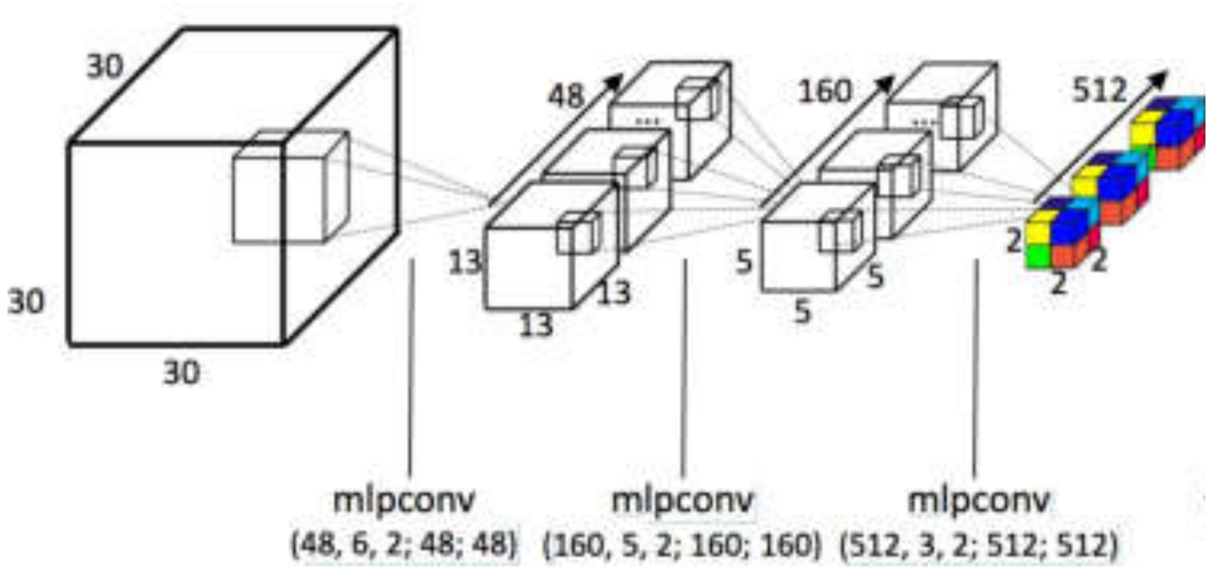
Voxelization

Represent the occupancy of regular 3D grids

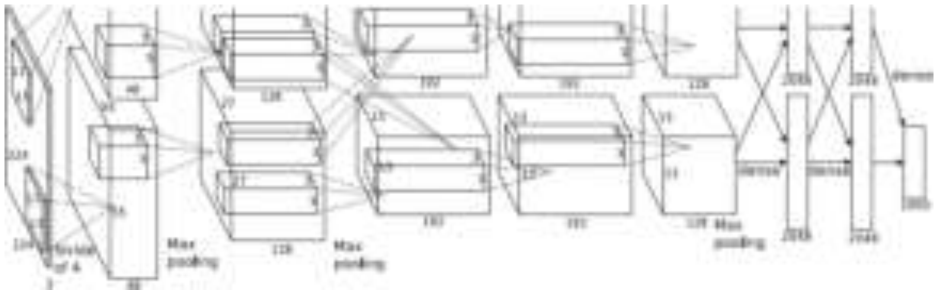


3D CNN on Volumetric Data

3D convolution uses 4D kernels



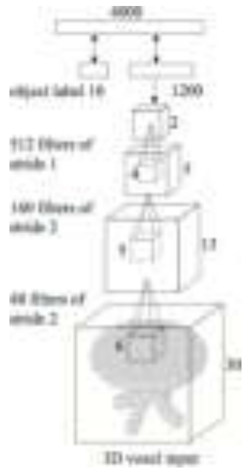
Complexity Issue



AlexNet, 2012

Input resolution: 224x224

224x224=50176

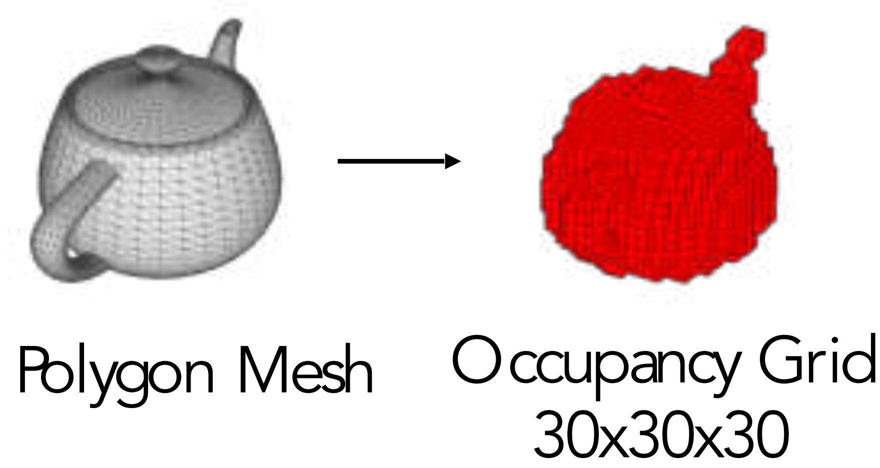


3DShapeNets, 2015

Input resolution: 30x30x30

224x224=27000

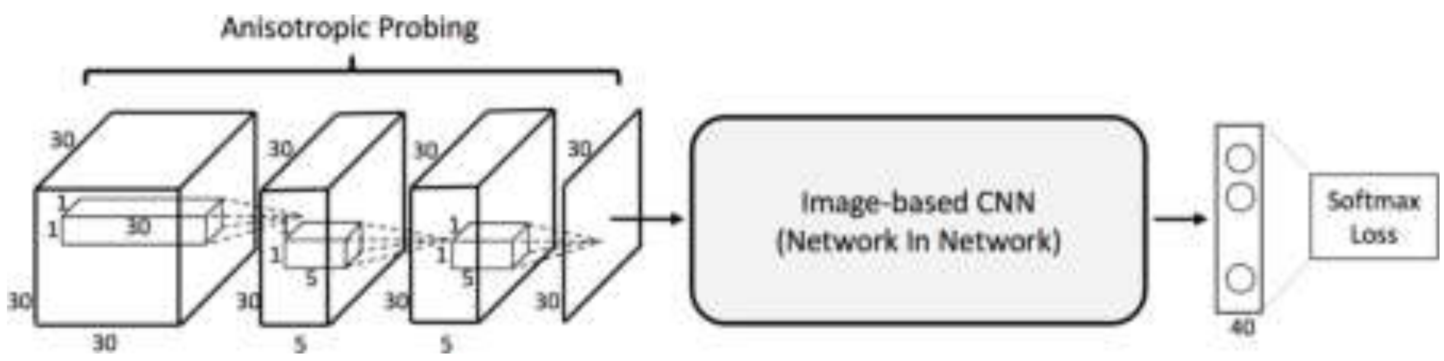
Complexity Issue



Information loss in voxelization

One Idea: Learn to Project

Idea: “X-ray” rendering + Image (2D) CNNs
very low #param, very low computation

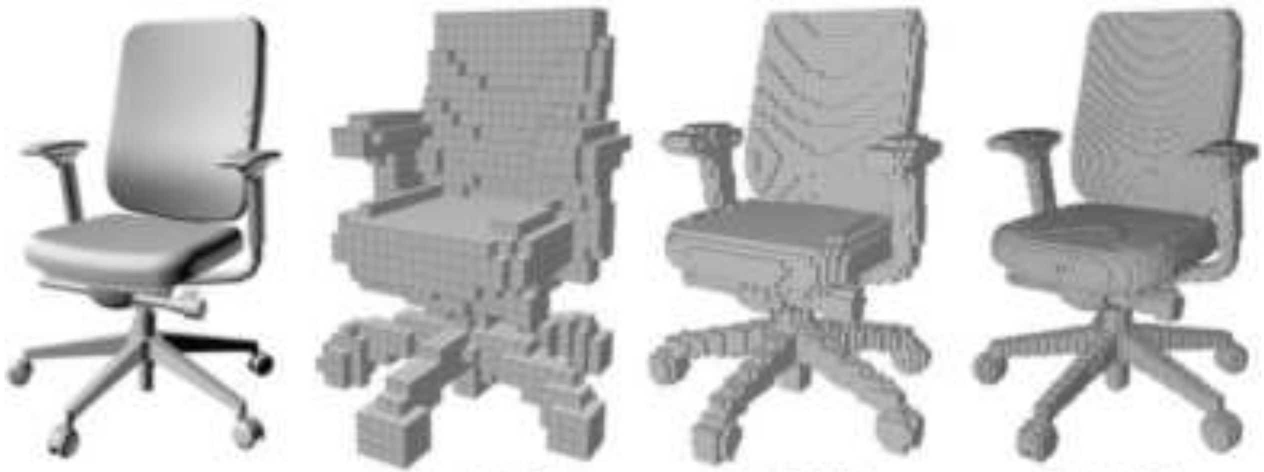


Su et al., “**Volumetric and Multi-View CNNs for Object Classification on 3D Data**”, *CVPR 2016*

Many other works in autonomous driving use **bird’s eye view** for object detection

Slide credit: Hao Su

More Principled: Sparsity of 3D Shapes

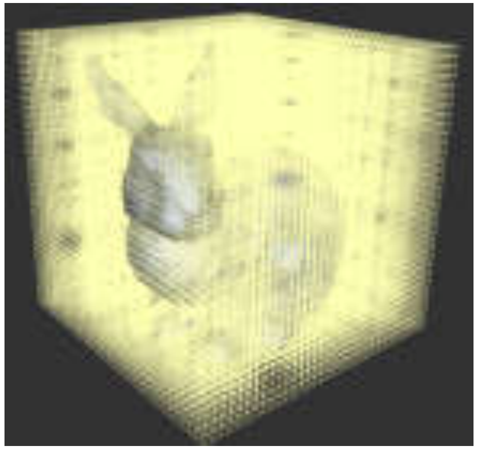


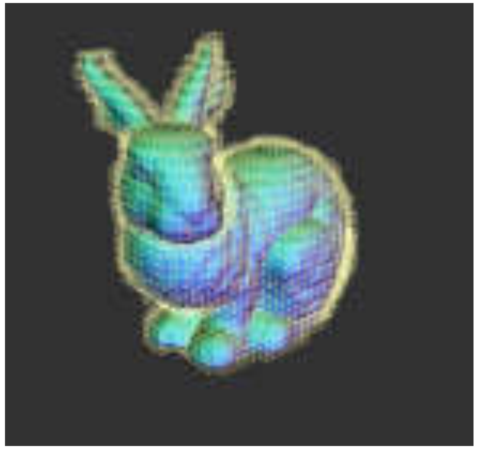
$$\frac{\#occupied\ grid}{\#total\ grid}$$

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

Store only the Occupied Grids

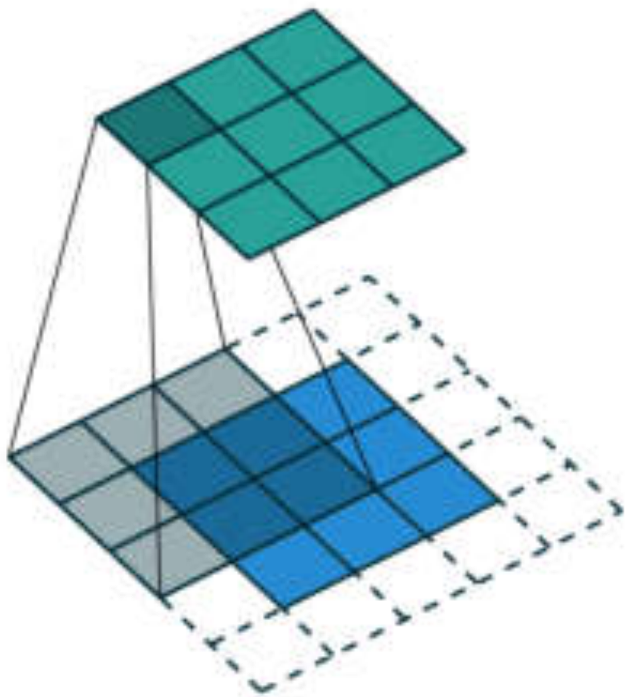
- Store the sparse surface signals
- Constrain the computation near the surface



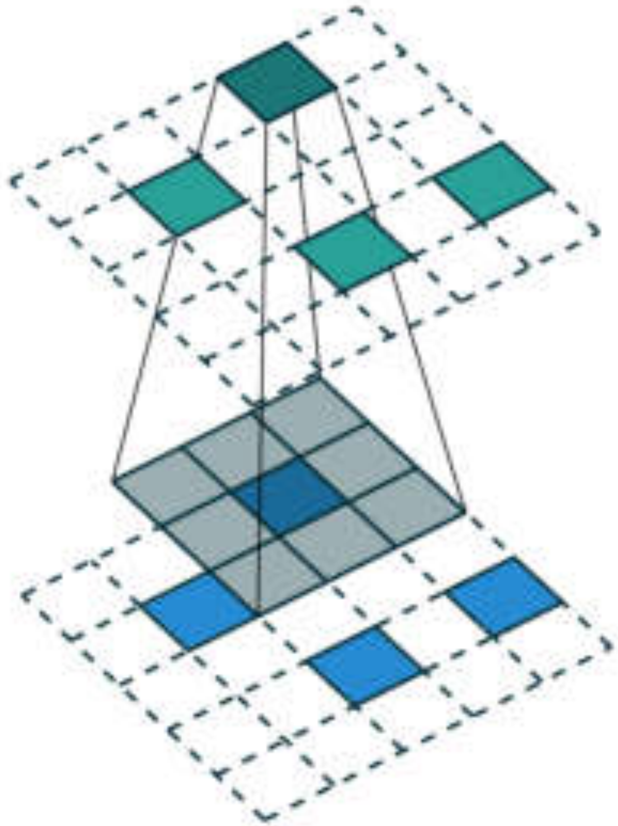


Slide credit: Hao Su

Sparse Convolution



Dense Conv



Sparse Conv

Image credit: Chris Choy

Sparse Convolution

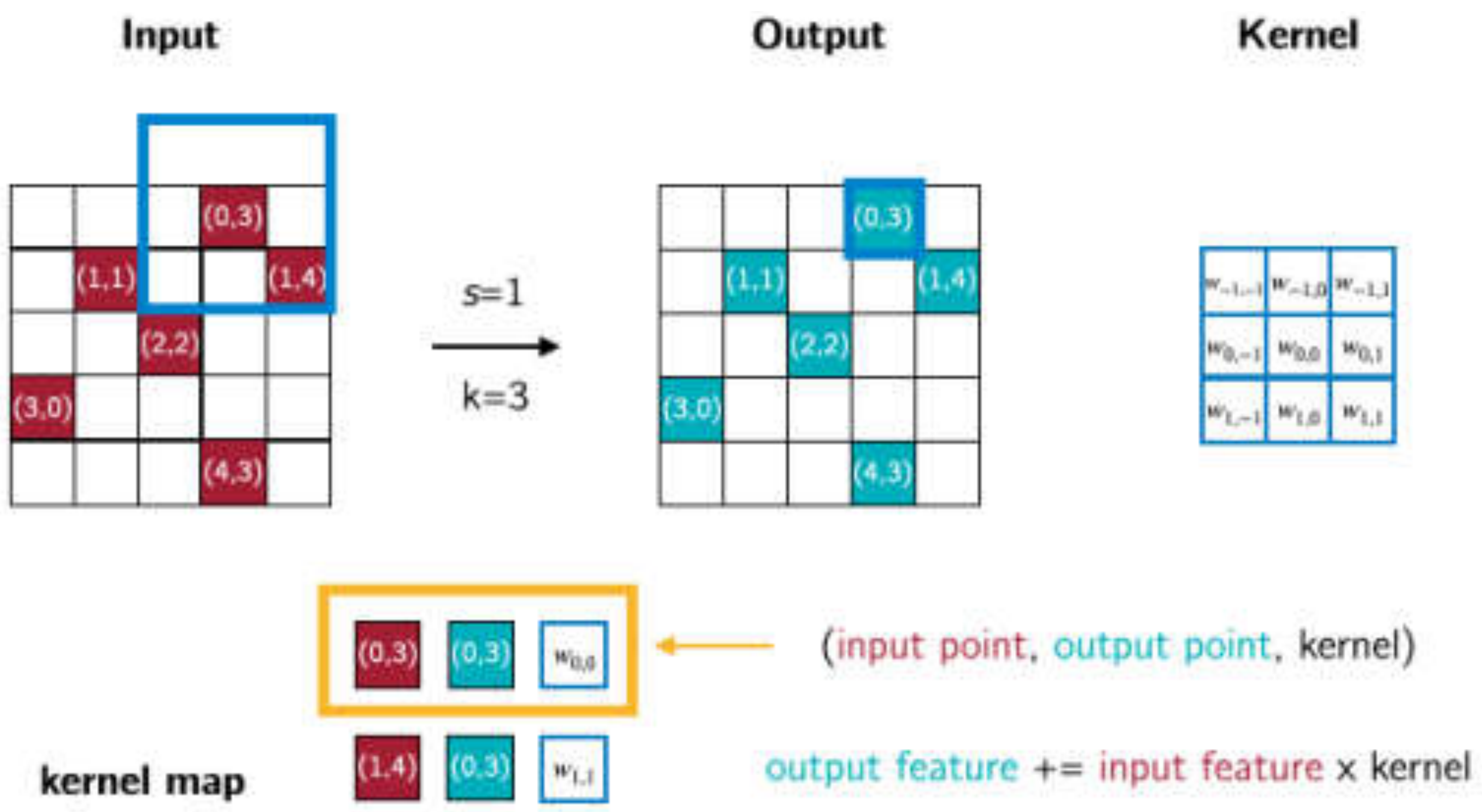


Image credit: torchsparse

Implementation

- SparseConvNet
    - https://github.com/facebookresearch/ SparseConvNet
    - Uses ResNet architecture
    - Takes time to train
- MinkowskiEngine
    -
- TorchSparse
    -
- Tensorflow3D
    -

Summary of Sparse Conv

- Pros:
  - A way higher efficiency than dense conv
  - Regular grid that supports indexing
  - Similarly expressive compared to 2D Conv
  - Translation equivariance similar to 2D Conv

- Cons:
  - Discretization error

Sparse Conv vs. Point Cloud Networks

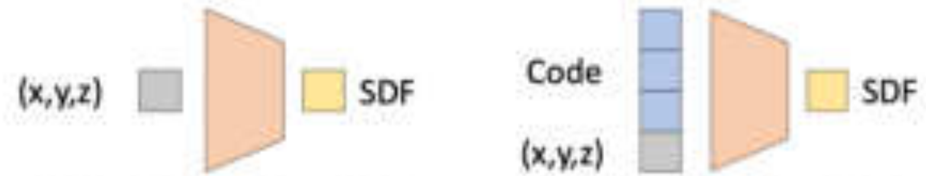
- Sparse Conv:
  - +: Kernels are spatial anisotropic
  - +: More efficient for indexing and neighbor query
  - +: suitable for large-scale scenes
  - -: limited resolutions

- Point cloud networks:
  - +: high resolution
  - +: easier to use and can be the first choice for a quick try
  - -: slightly lower performance
  - -: slower if performing FPS and ball query

Outline

- Point Networks
  - PointNet
  - PointNet++
- Voxel Networks
- Networks for other representations
  - SDF
  - Mesh

Deep SDF

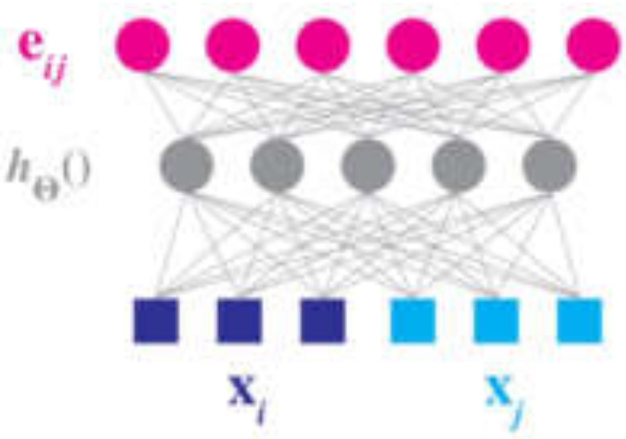


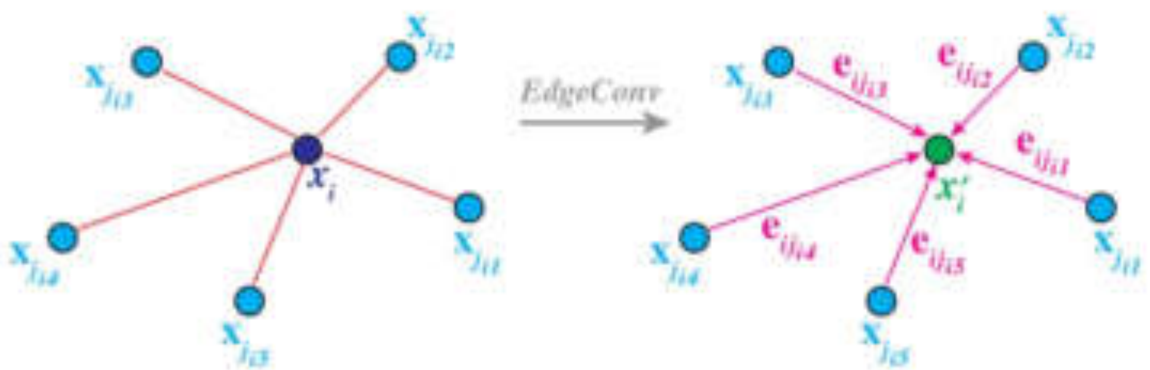
(a) Single Shape DeepSDF (b) Coded Shape DeepSDF

- (a) use the network to overfit a single shape
- (b) use a latent code to represent a shape, so that the network can be used for multiple shapes

Park et al., “DeepSDF: Learning continuous signed distance functions for shape representation.”, CVPR 2019

Convolution on Mesh/Graph





Message passing: The output of EdgeConv at the i-th vertex is thus given by

$$\mathbf{x}'_i = \mathop{\square}_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j). \quad (1)$$

Wang, et.al., Dynamic Graph CNN for Learning on Point Clouds, ToG 2019

Thank you!