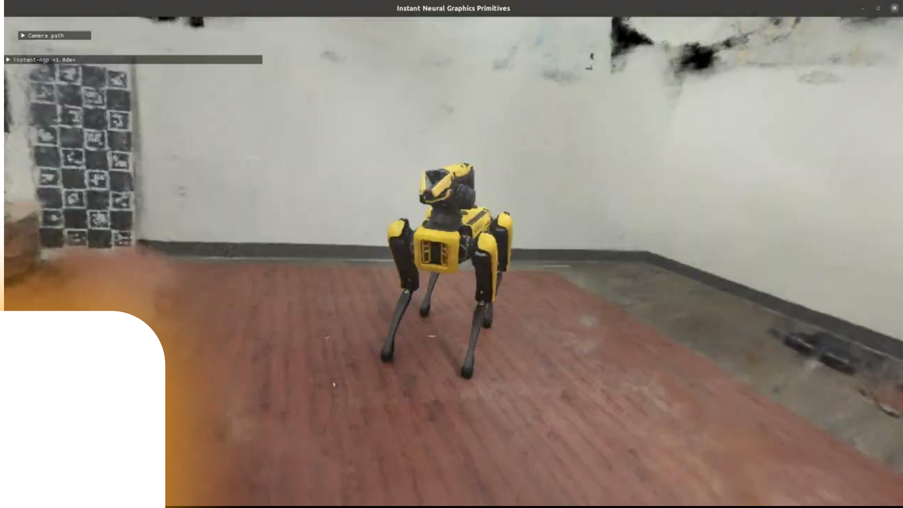
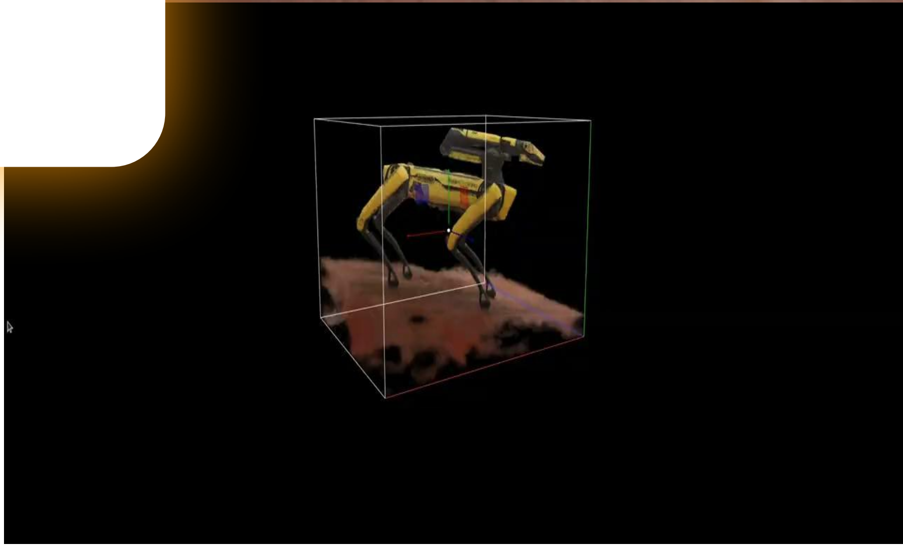
DeepRob

[Student] Lecture 17
by Michael Andrev, Chen Hu, Sean Coffey
Implicit Surfaces, Geometry, NeRF
University of Michigan and University of Minnesota



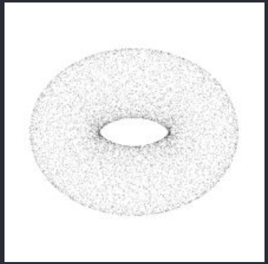


Video credits - Nirmal Raj

Let's go over the basics!

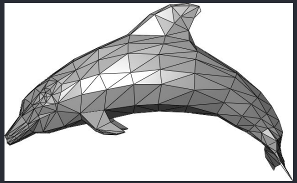
Data Structures of 3D Geometry

1. Pointclouds



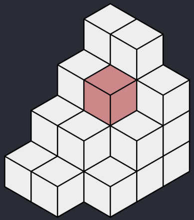
https://commons.wikimedia.org/wiki/File:Point_cloud_torus.gif

1. Meshes



https://en.wikipedia.org/wiki/Polygon_mesh

1. Voxels



https://en.wikipedia.org/wiki/Voxel

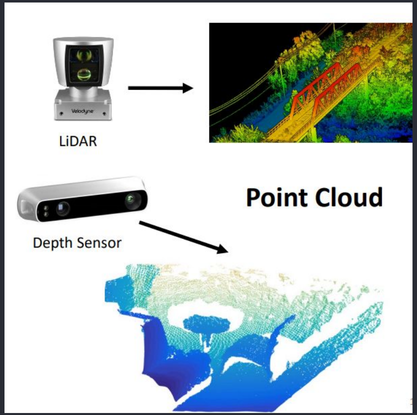
Pointclouds

Point + Cloud = Pointcloud

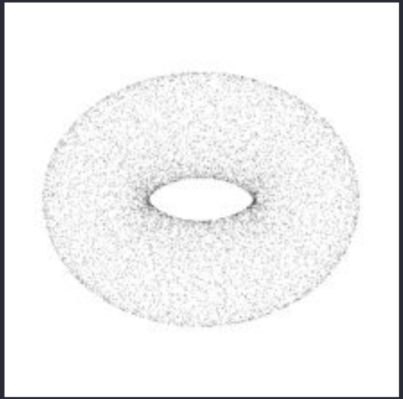
A measurement unit that is represented using x, y, and z coordinates.

An aggregation of many small units of something.

A set of points in a space that represent some 3D shape or object



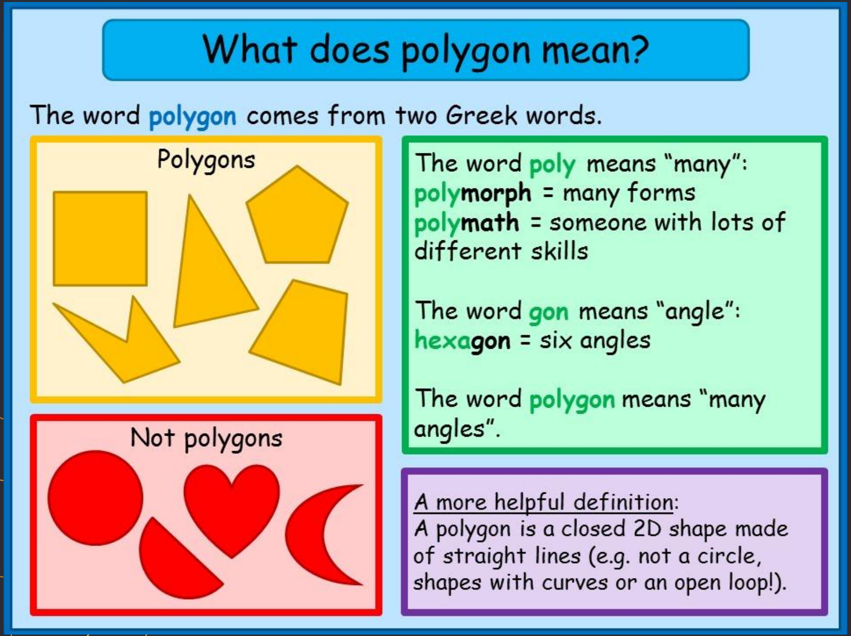
https://geo-matching.com/content/how-to-get-the-best-precision-and-improve-pointcloud-accuracy



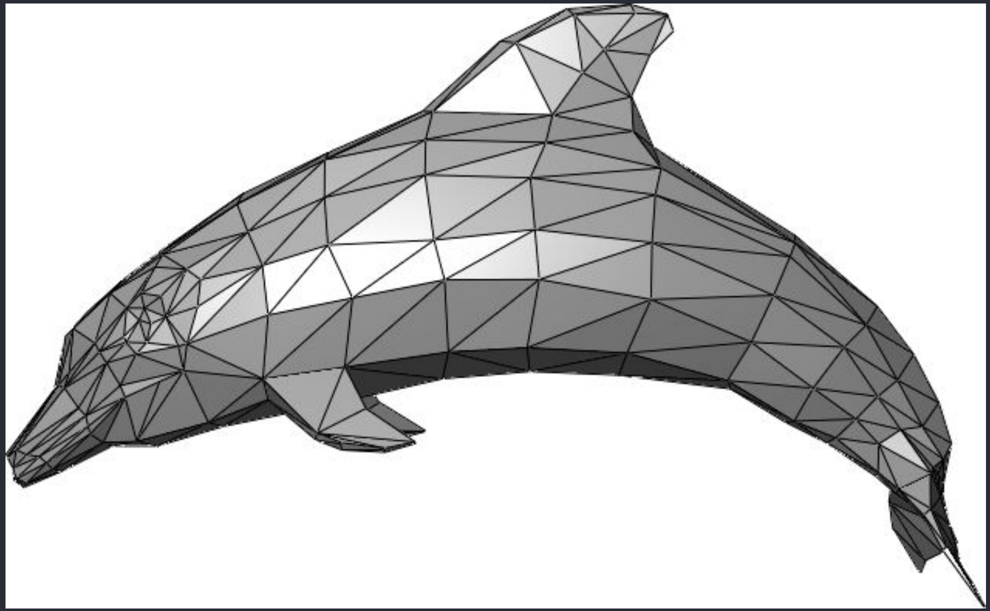
https://commons.wikimedia.org/wiki/File:Point_cloud_torus.gif

Meshes

Meshes (or polygon meshes) are a 3D object that is composed of 1+ polygons



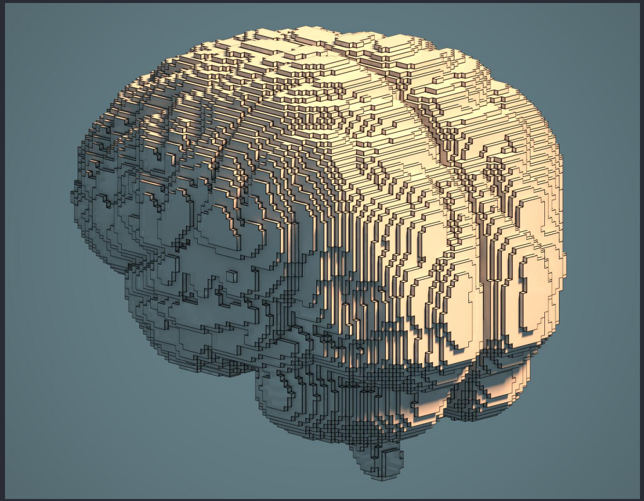
https://www.youtube.com/watch?v=sMlxmItrjWE



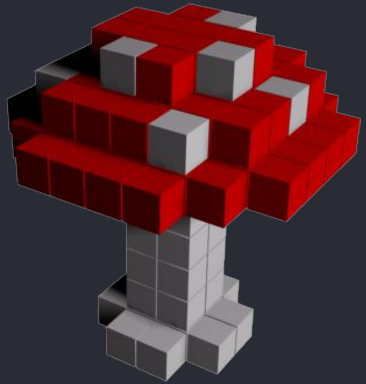
https://en.wikipedia.org/wiki/Polygon_mesh

Voxels

Voxels are 3D pixels represented as cubes.

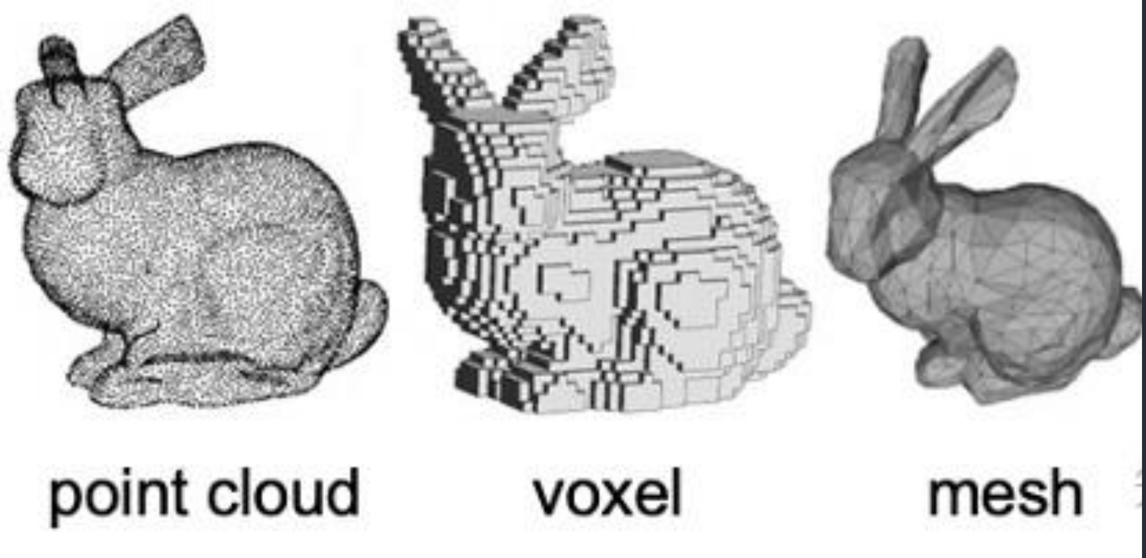


https://www.google.com/url?sa=i&url=https%3A%2F%2Fdon_backos.artstation.com%2Fprojects%2FrR5Y8J&psig=AOvVaw1miitIbCKz2tjcvKOX7XW_&ust=1680288200369000&source=images&cd=vfe&ved=0CA8QjRxqFwoTCNicib_ChP4CFQAAAAAdAAAAABAD



https://blog.spatial.com/the-main-benefits-and-disadvantages-of-voxel-modeling

Summary of Data Structures of 3D Geometry



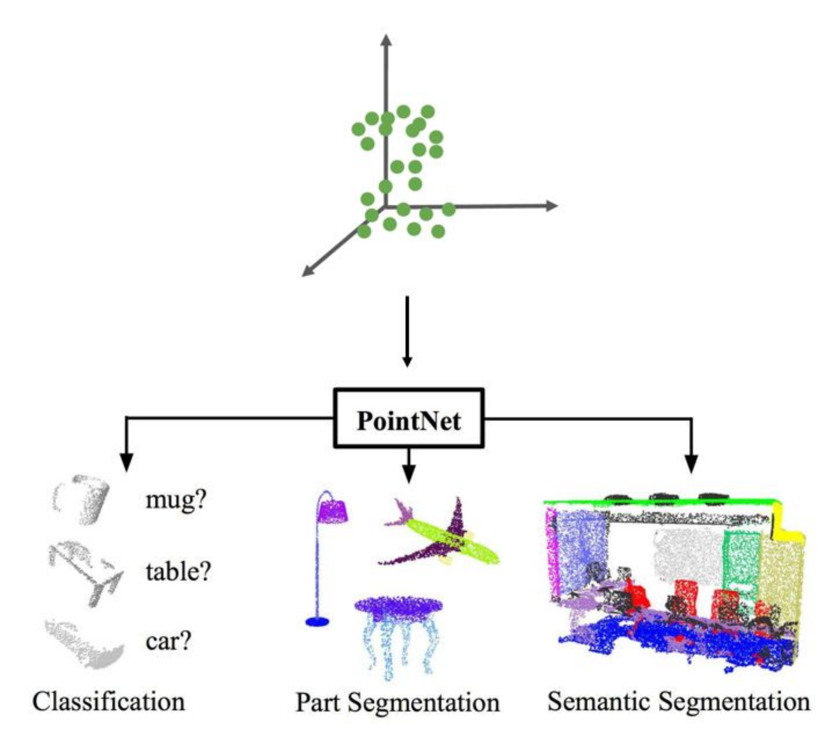
http://graphics.stanford.edu/courses/cs348n-22-winter/

Questions?

PointNet and PointNet++

PointNet

1. **End-to-end learning** for irregular point data
2. **Unified** framework for various tasks



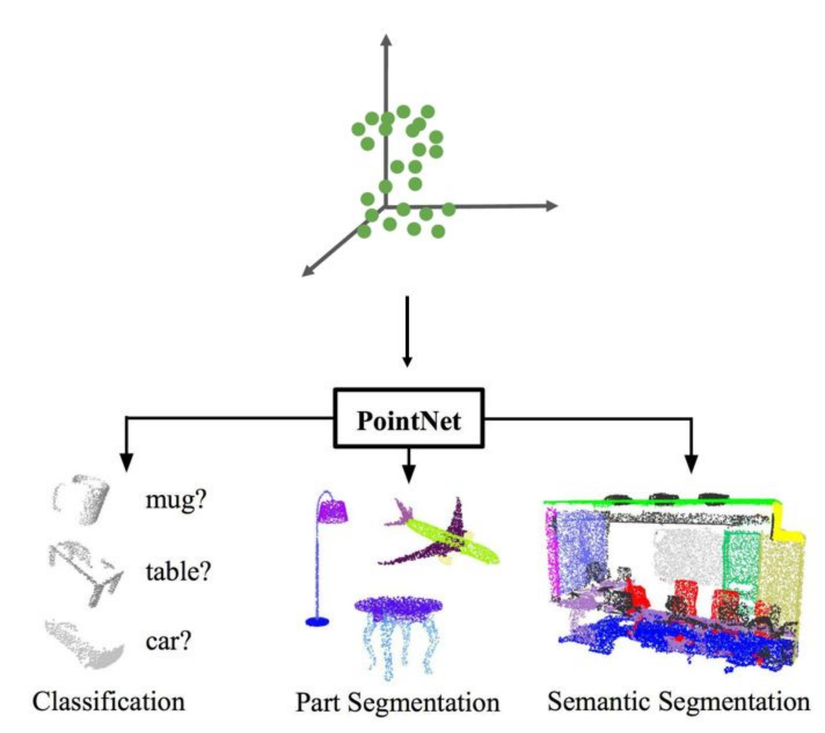
Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

PointNet

1. **End-to-end learning** for irregular point data
2. **Unified** framework for various tasks

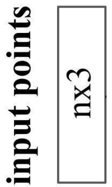
PointNet has to respect key characteristics for points clouds:

1. Point Permutation Invariance
2. Spatial Transformation Invariance
3. Sampling Invariance



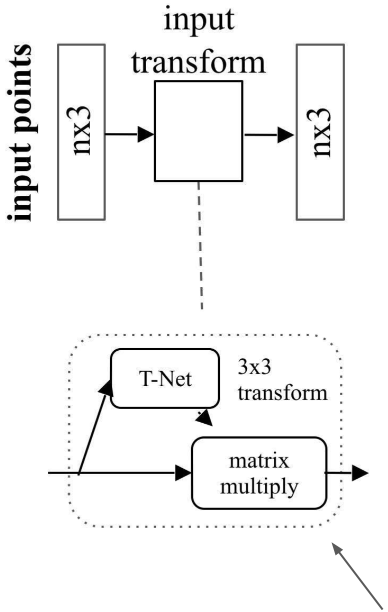
Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

PointNet Classification Network



Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

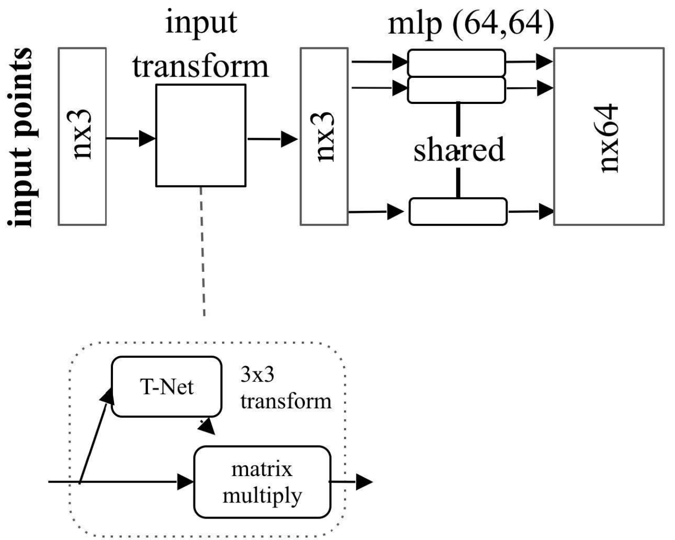
PointNet Classification Network



Encode Spatial Transformation Invariance

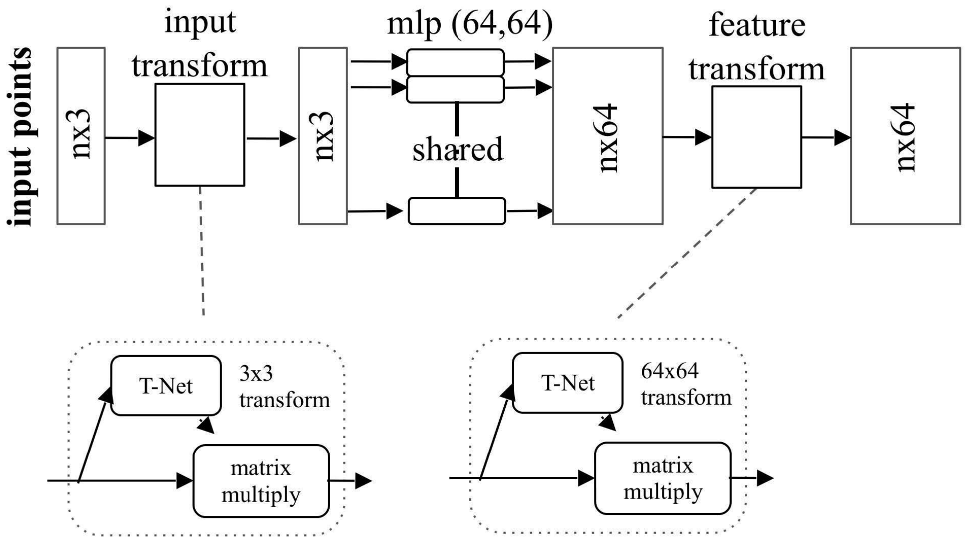
Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

PointNet Classification Network



Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

PointNet Classification Network

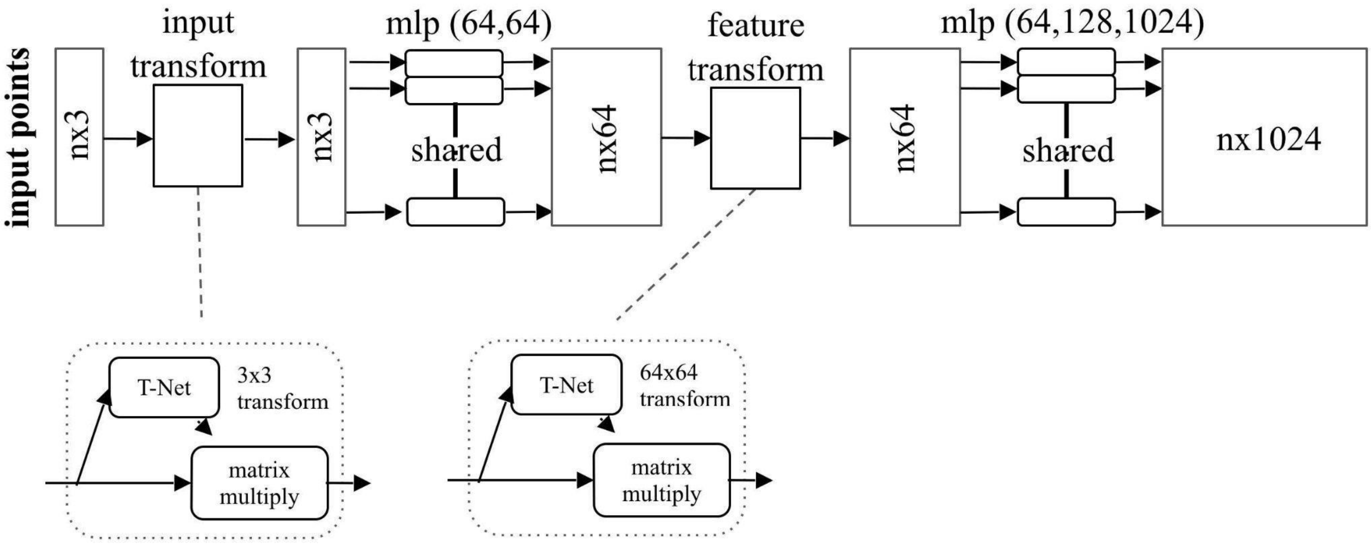


Encode Spatial Transformation Invariance

Regularization loss:
Transform matrix close to orthogonal: $L_{reg} = \|I - AA^T\|_F^2$

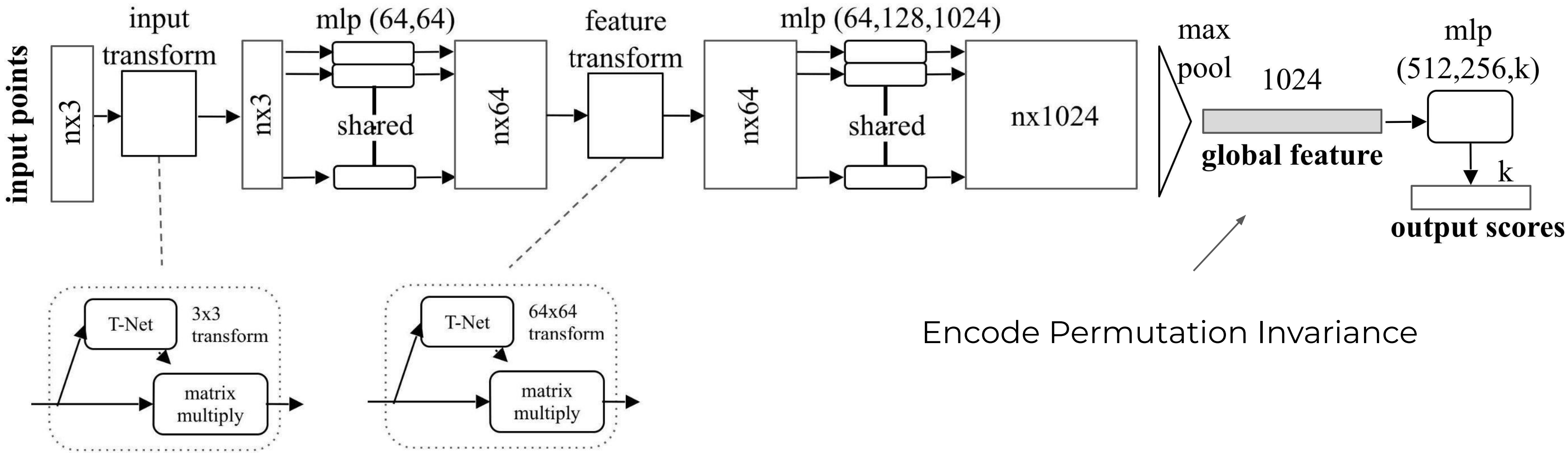
Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

PointNet Classification Network



Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

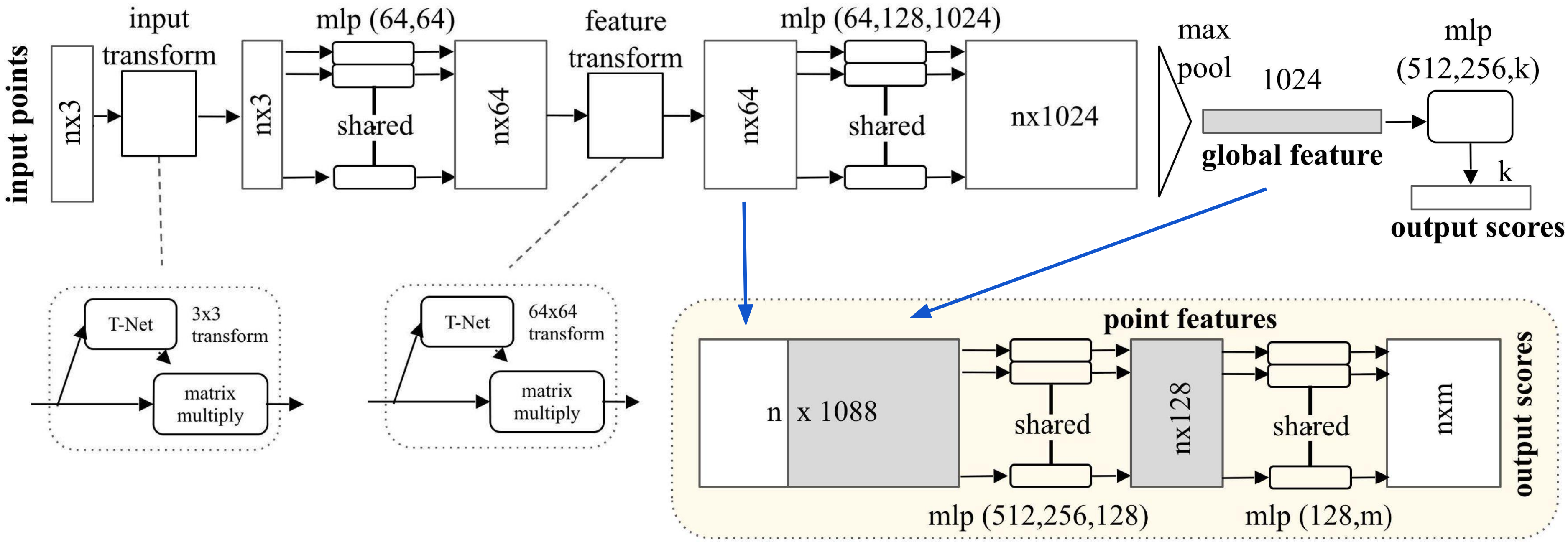
PointNet Classification Network



Encode Permutation Invariance

Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

PointNet Segmentation Network

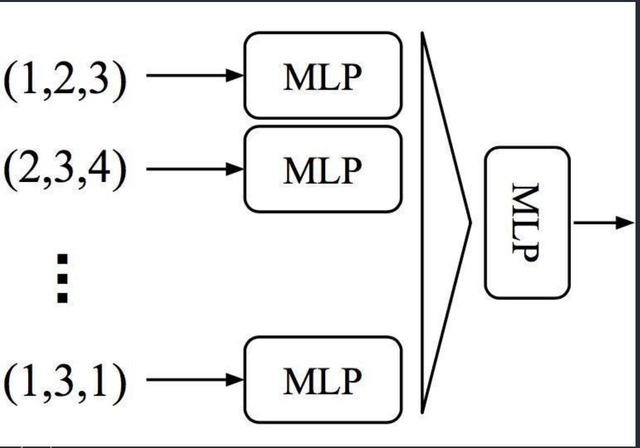


Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

PointNet++

Limitation of PointNet - Global feather learning

1. No local context
2. Limited local invariance

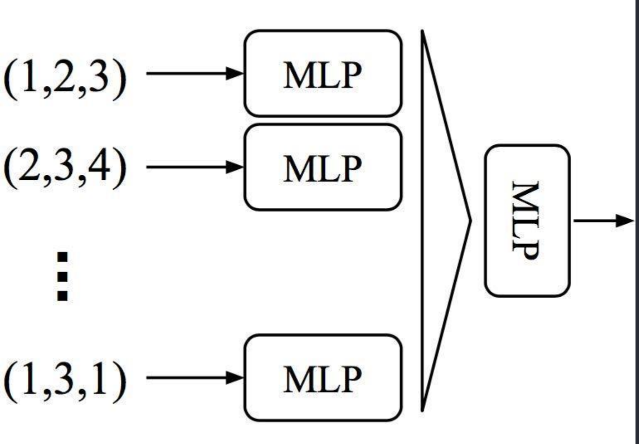


Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

PointNet++

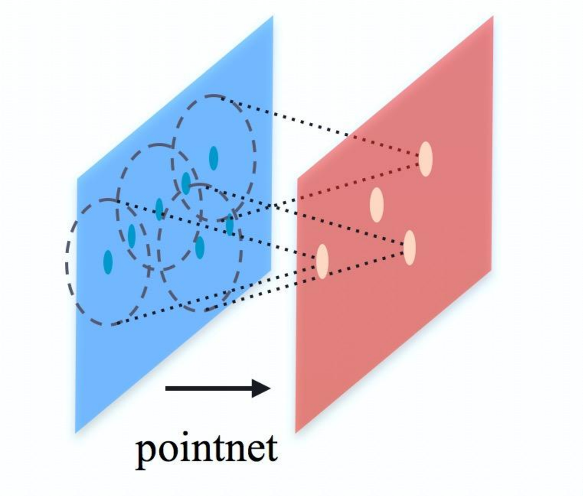
Limitation of PointNet - Global feather learning

1. No local context
2. Limited local invariance



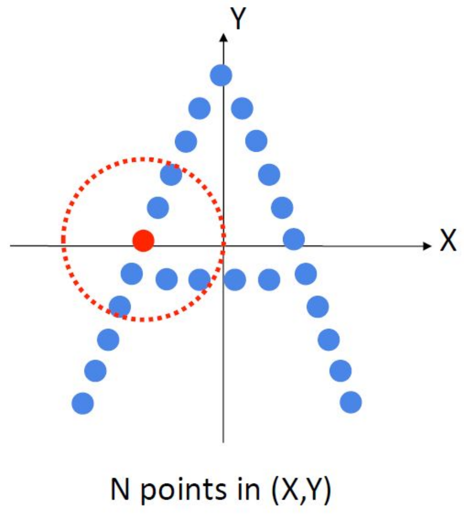
PointNet++: recursively apply pointnet at local regions:

1. Hierarchical feature learning
2. Local translational invariance
3. Permutation invariance



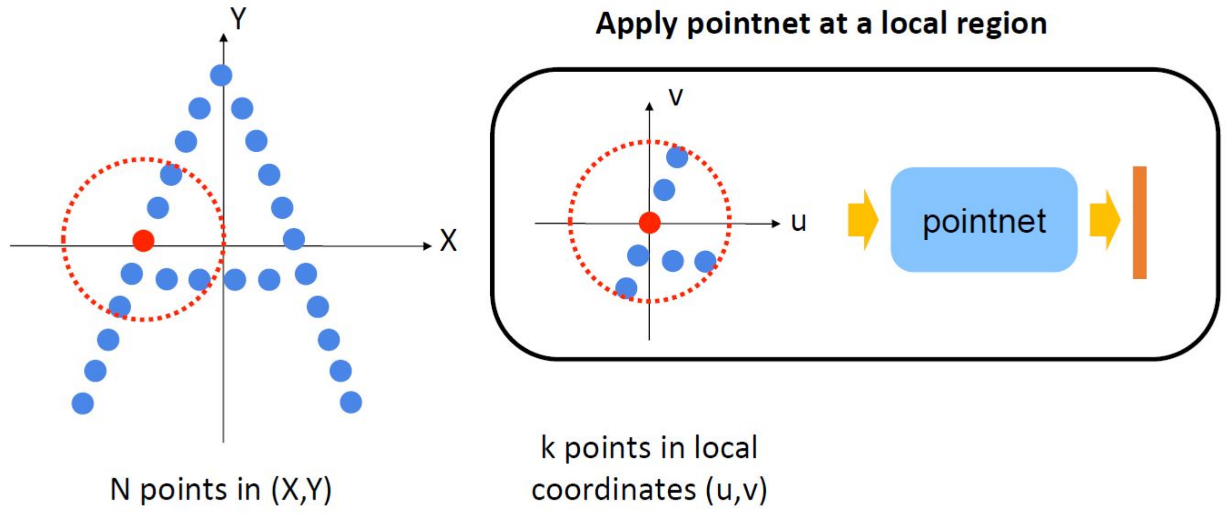
Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

Hierarchical Point Feature Learning



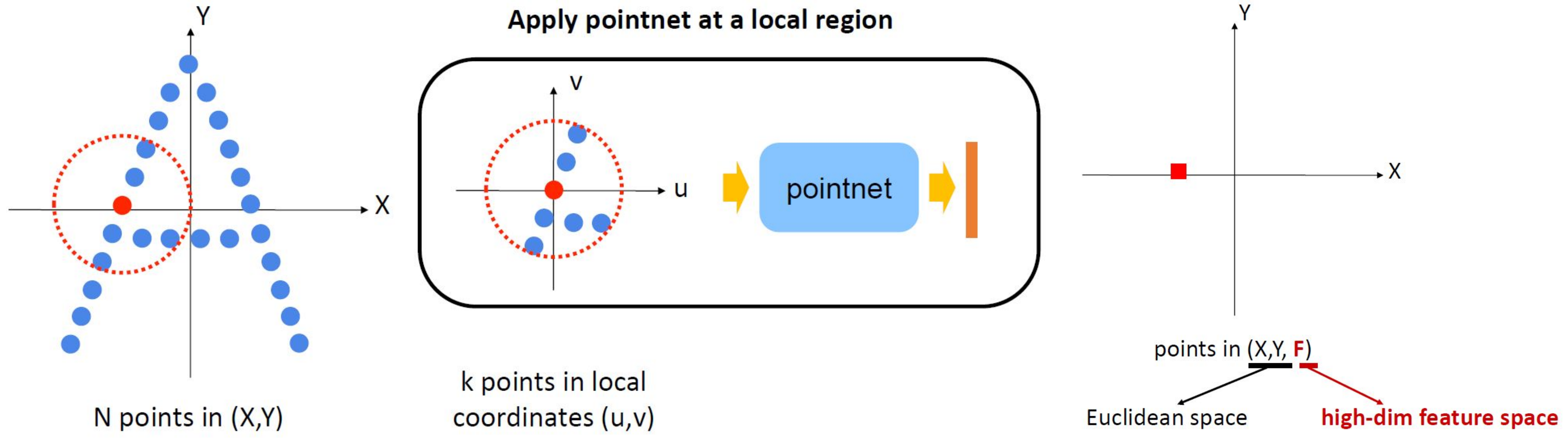
Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

Hierarchical Point Feature Learning



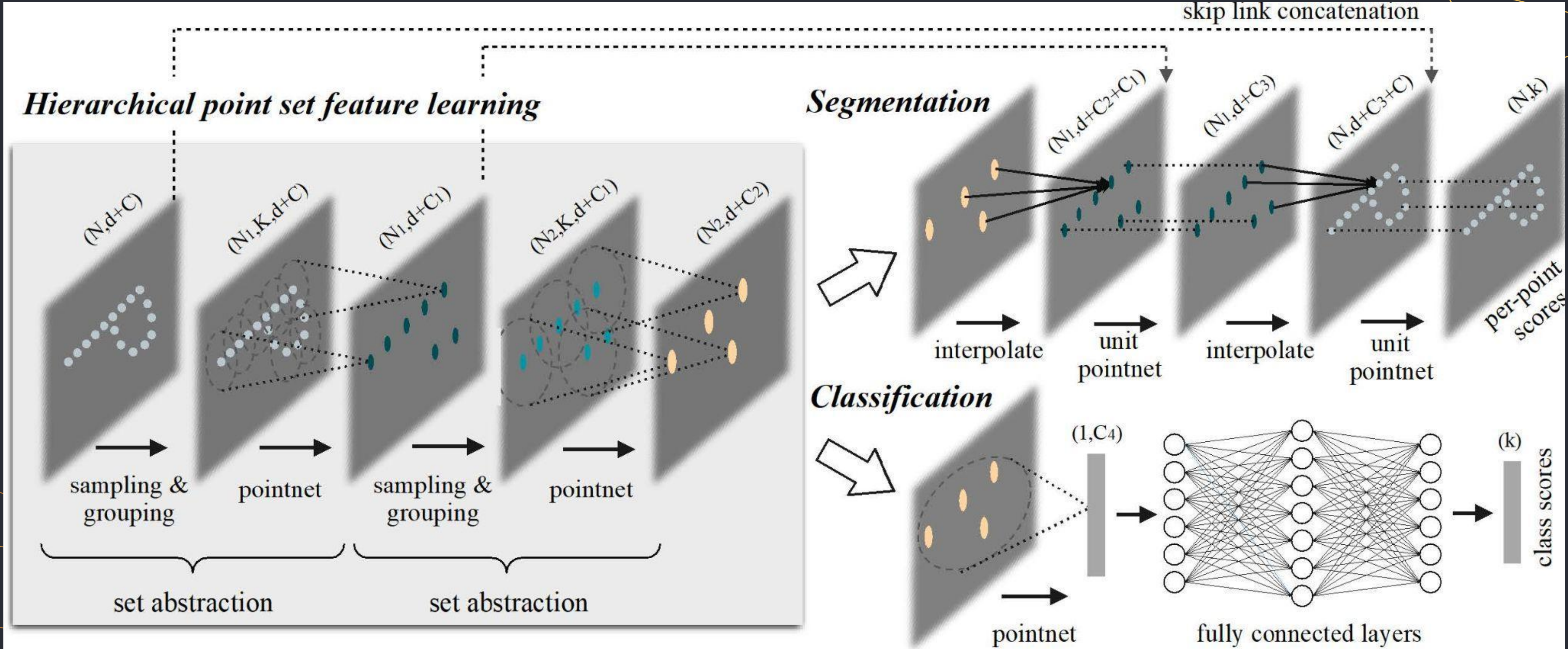
Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

Hierarchical Point Feature Learning



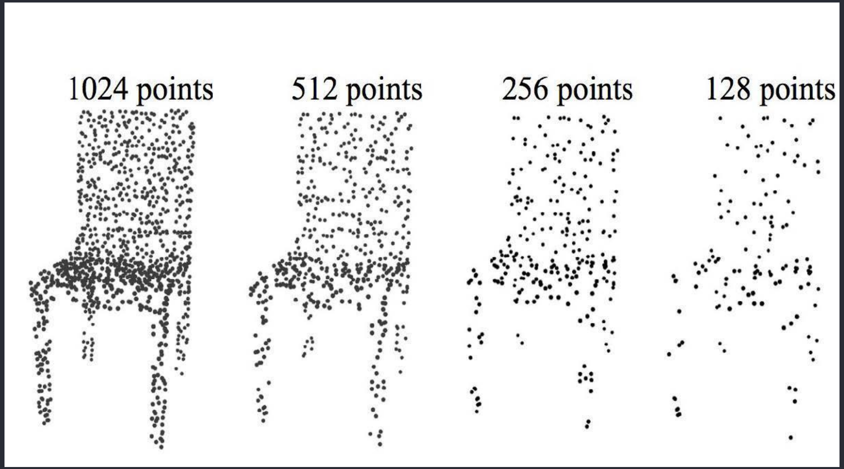
Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

PointNet++

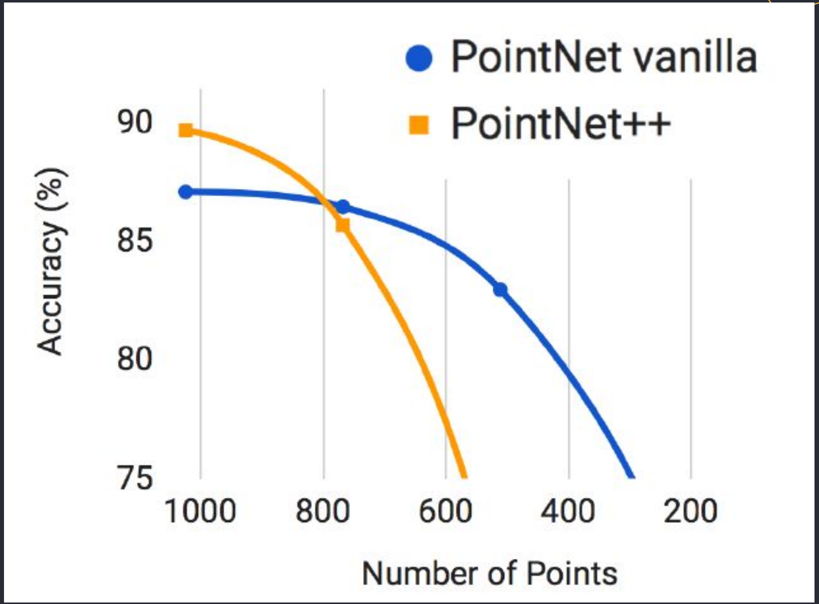


Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

PointNet and PointNet++ Comparison



Density Variation



Qi, Charles Ruizhongtai, et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." Advances in neural information processing systems 30 (2017).

Questions?

3D Reconstruction before NeRF

Reconstruction of 3D scenes and Geometries

The process of capturing the shape and appearance of real objects[1]

Active Methods

- Actively interfere with the reconstructed object, either mechanically or radiometrically (ex laser range finder, 3D ultrasound)[2]

Passive Methods

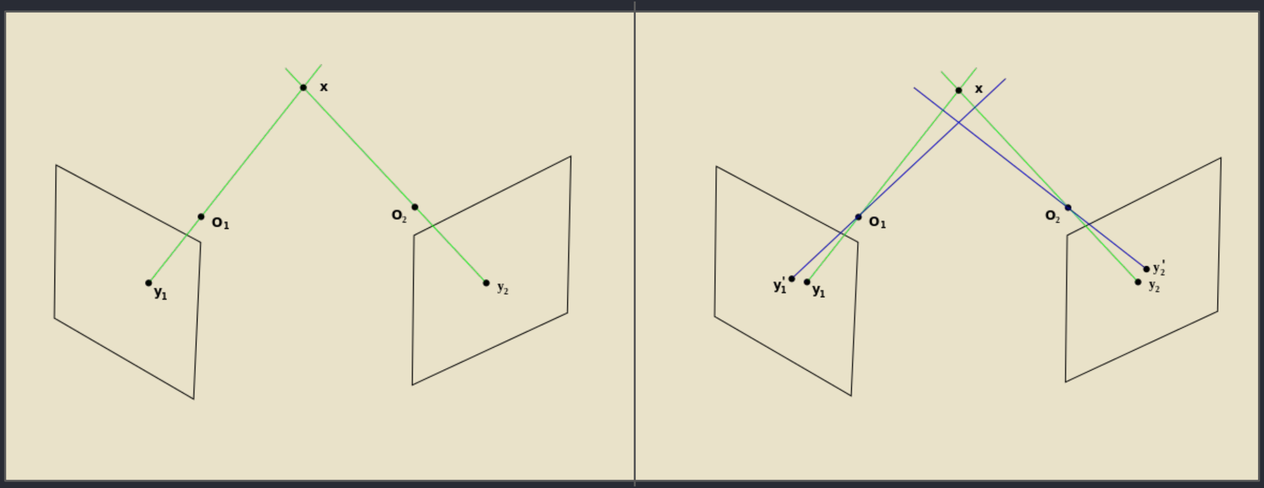
- Measure the radiance reflected or emitted by the object's surface to infer its 3D structure through image understanding[3]

[1]Moons, Theo, Luc Van Gool, and Maarten Vergauwen. "3D reconstruction from multiple images part 1: Principles."
[2]Mahmoudzadeh, Ahmadreza; Golroo, Amir; Jahanshahi, Mohammad R.; Firoozi Yeganeh, Sayna (January 2019). "Estimating Pavement Roughness by Fusing Color and Depth Data Obtained from an Inexpensive RGB-D Sensor"
[3]Buelthoff, Heinrich H., and Alan L. Yuille. "Shape-from-X: Psychophysics and computation Archived 2011-01-07 at the Wayback Machine."

Triangulation

The process of determining a point in 3D space given its projections onto two, or more, images.

In order to solve this problem it is necessary to know the parameters of the camera projection function from 3D to 2D for the cameras involved, in the simplest case represented by the camera matrices.[1]

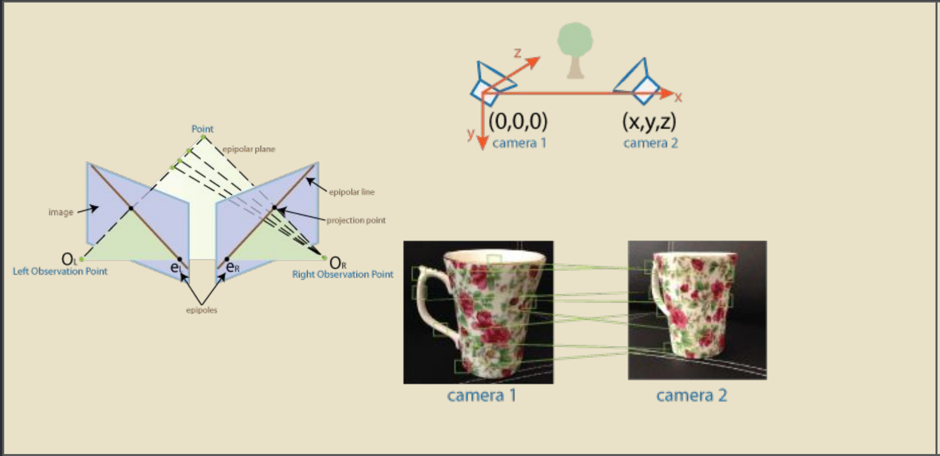


[1]Richard Hartley and Andrew Zisserman (2003). “Multiple View Geometry in computer vision”

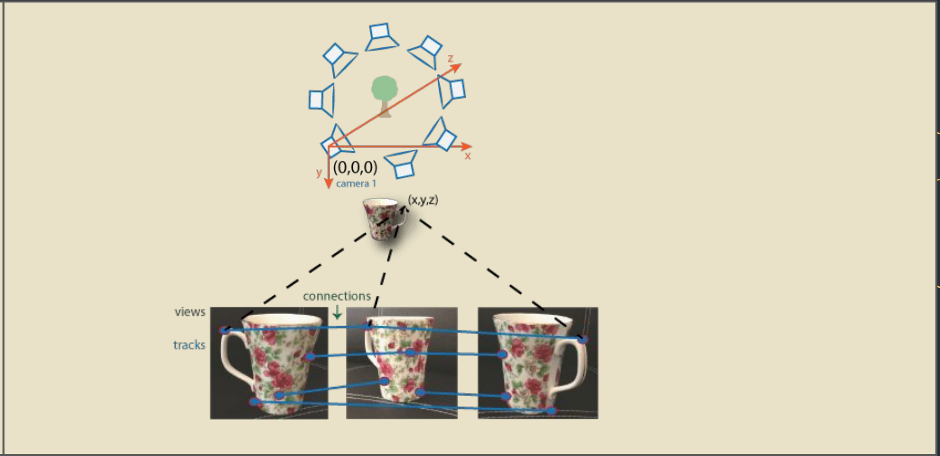
Structure from Motion (SfM)

Structure from motion (SfM) is the process of estimating the 3-D structure of a scene from a set of 2-D images. SfM is used in many applications, such as 3-D scanning , augmented reality, and visual simultaneous localization and mapping (vSLAM)

Structure from Motion from Two Views



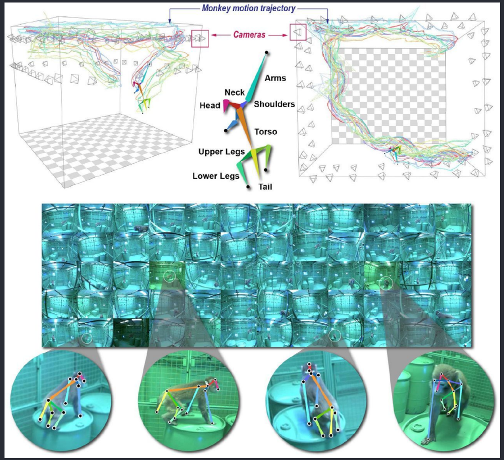
Structure from Motion from Multiple Views



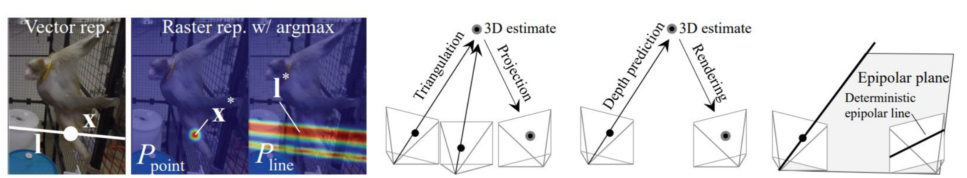
Images from: https://www.mathworks.com/help/vision/ug/structure-from-motion.html

Simultaneous Localization and Mapping (SLAM)

The computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.



Open Monkey Studio



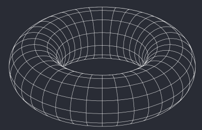
(a) Representation mismatch (b) Triangulation (c) Depth prediction (d) Epipolar line

[1]Yuan Yao, Yasamin Jafarian, & Hyun Soo Park. (2019). “MONET: Multiview Semi-supervised Keypoint Detection via Epipolar Divergence.”

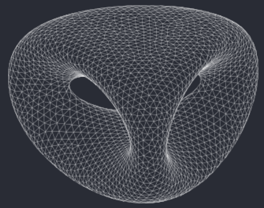
Implicit Surfaces

An implicit surface is the set of zeros of a function of three variables. Implicit means that the equation is not solved for *x* or *y* or *z*.

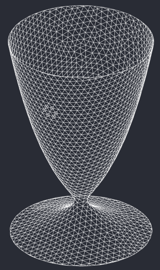
$$\boxed{F(x, y, z) = 0.}$$



$$(x^2 + y^2 + z^2 + R^2 - a^2)^2 - 4R^2(x^2 + y^2) = 0.$$



$$2y(y^2 - 3x^2)(1 - z^2) + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2) = 0$$

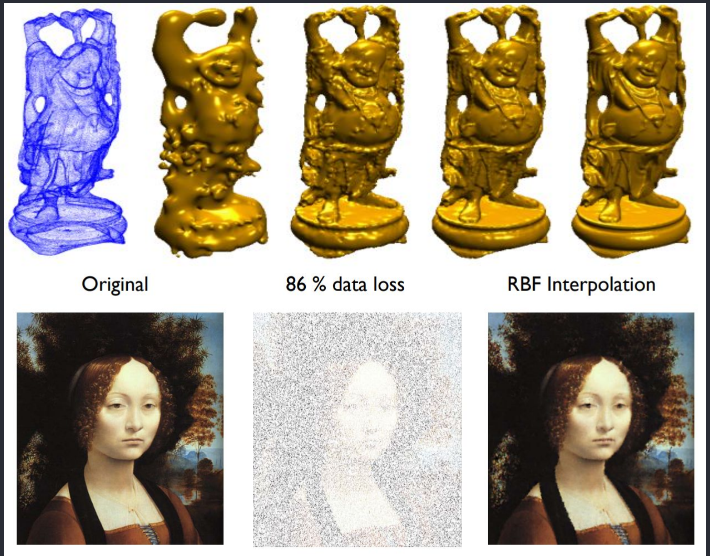


$$x^2 + y^2 - (\ln(z + 3.2))^2 - 0.02 = 0$$

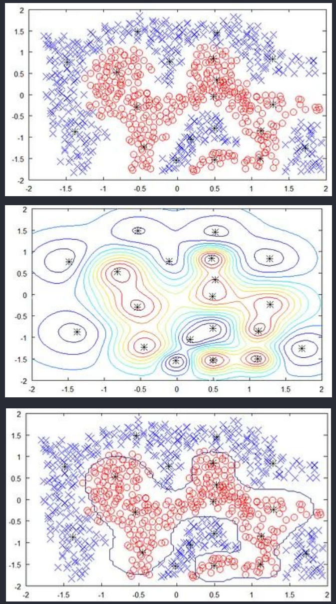
https://en.wikipedia.org/wiki/Implicit_surface

Radial Basis Function

A radial basis function (RBF) is a real-valued function whose value depends only on the distance between the input and some fixed point

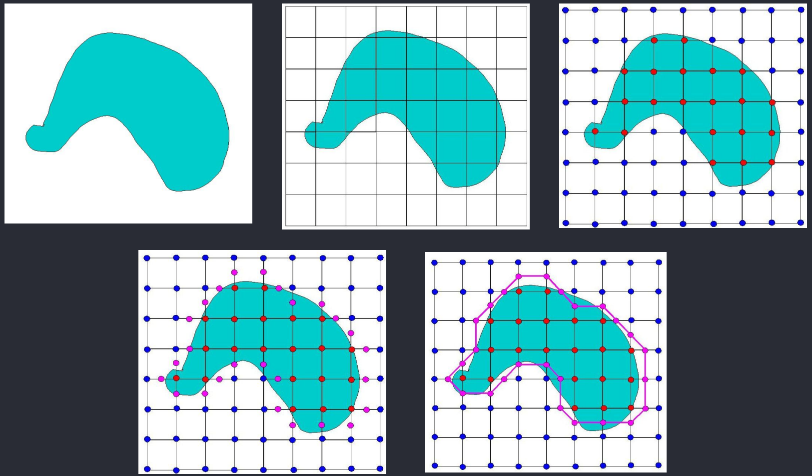


Images from: https://www.cs.jhu.edu/~misha/Fall05/Papers/carr01.pdf



Marching Cubes Algorithm

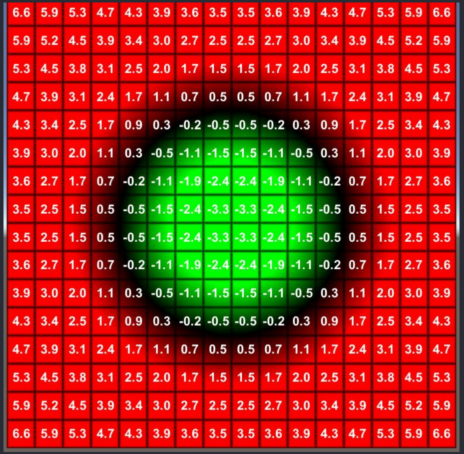
Marching cubes is a computer graphics algorithm, published in the 1987 SIGGRAPH proceedings by Lorensen and Cline, for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field (the elements of which are sometimes called voxels)



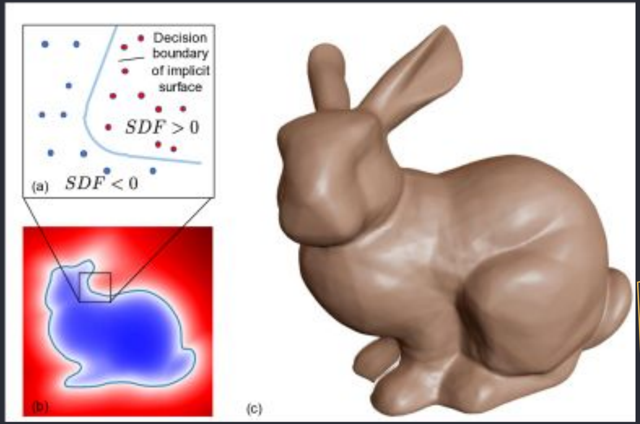
Images from:
https://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html

Signed Distance Function

The signed distance function is a mathematical function that is used to define implicit surfaces.



6.6	5.9	5.3	4.7	4.3	3.9	3.6	3.5	3.5	3.6	3.9	4.3	4.7	5.3	5.9	6.6
5.9	5.2	4.5	3.9	3.4	3.0	2.7	2.5	2.5	2.7	3.0	3.4	3.9	4.5	5.2	5.9
5.3	4.5	3.8	3.1	2.5	2.0	1.7	1.5	1.5	1.7	2.0	2.5	3.1	3.8	4.5	5.3
4.7	3.9	3.1	2.4	1.7	1.1	0.7	0.5	0.5	0.7	1.1	1.7	2.4	3.1	3.9	4.7
4.3	3.4	2.5	1.7	0.9	0.3	-0.2	-0.5	-0.5	-0.2	0.3	0.9	1.7	2.5	3.4	4.3
3.9	3.0	2.0	1.1	0.3	-0.5	-1.1	-1.5	-1.5	-1.1	-0.5	0.3	1.1	2.0	3.0	3.9
3.6	2.7	1.7	0.7	-0.2	-1.1	-1.9	-2.4	-2.4	-1.9	-1.1	-0.2	0.7	1.7	2.7	3.6
3.5	2.5	1.5	0.5	-0.5	-1.5	-2.4	-3.3	-3.3	-2.4	-1.5	-0.5	0.5	1.5	2.5	3.5
3.5	2.5	1.5	0.5	-0.5	-1.5	-2.4	-3.3	-3.3	-2.4	-1.5	-0.5	0.5	1.5	2.5	3.5
3.6	2.7	1.7	0.7	-0.2	-1.1	-1.9	-2.4	-2.4	-1.9	-1.1	-0.2	0.7	1.7	2.7	3.6
3.9	3.0	2.0	1.1	0.3	-0.5	-1.1	-1.5	-1.5	-1.1	-0.5	0.3	1.1	2.0	3.0	3.9
4.3	3.4	2.5	1.7	0.9	0.3	-0.2	-0.5	-0.5	-0.2	0.3	0.9	1.7	2.5	3.4	4.3
4.7	3.9	3.1	2.4	1.7	1.1	0.7	0.5	0.5	0.7	1.1	1.7	2.4	3.1	3.9	4.7
5.3	4.5	3.8	3.1	2.5	2.0	1.7	1.5	1.5	1.7	2.0	2.5	3.1	3.8	4.5	5.3
5.9	5.2	4.5	3.9	3.4	3.0	2.7	2.5	2.5	2.7	3.0	3.4	3.9	4.5	5.2	5.9
6.6	5.9	5.3	4.7	4.3	3.9	3.6	3.5	3.5	3.6	3.9	4.3	4.7	5.3	5.9	6.6



Images from:
https://shaderfun.com/2018/03/25/signed-distance-fields-part-2-solid-geometry/

Questions?

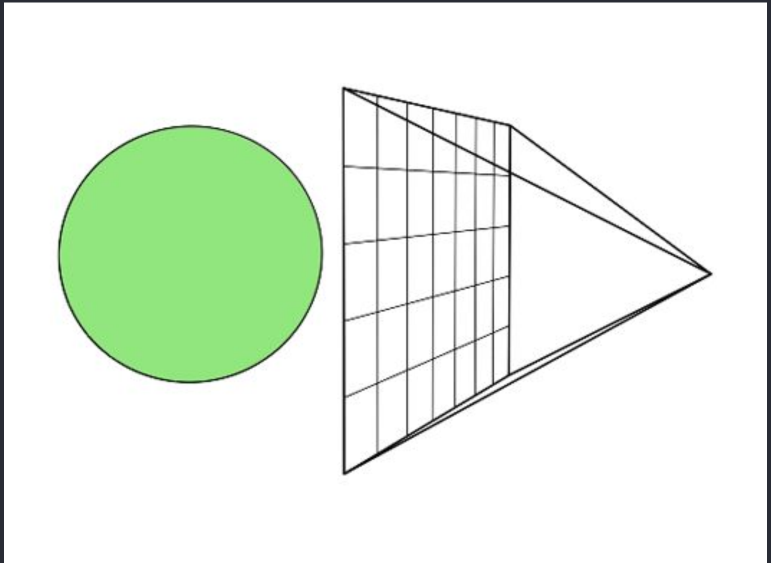
Finally, NeRF… Well… Almost!

Computer Graphic concepts

1. Rendering
2. Volume Rendering
3. Volume Synthesis

Rendering

Generating an image (render) from a 2D/3D model.

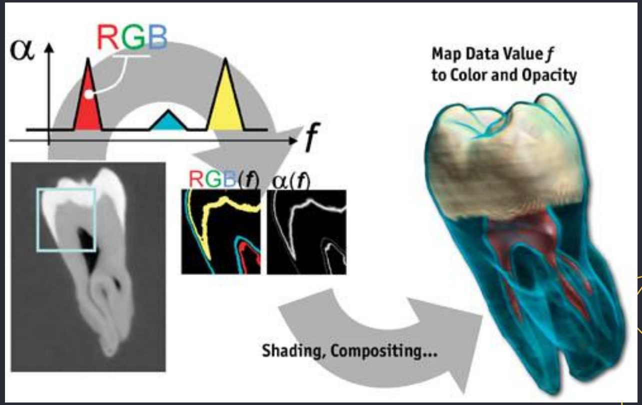


https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/implementing-the-raytracing-algorithm.html

Volume Rendering

Create a 2D projection of 3D voxel data

3D objects —> 2D images



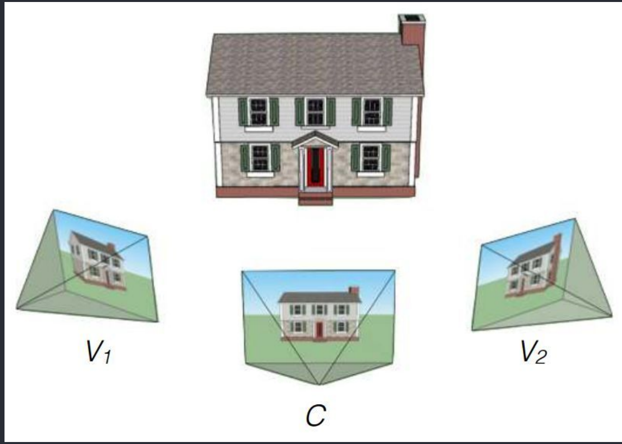
https://www.heavy.ai/technical-glossary/volume-rendering#:~:text=Volume%20rendering%20represents%20a%20collection,MicroCT%20scanner%202D%20slice%20images.

https://www.heavy.ai/technical-glossary/volume-rendering#:~:text=Volume%20rendering%20represents%20a%20collection,MicroCT%20scanner%202D%20slice%20images.

Volume Synthesis

Create a 3D view from a 2D scale

2D images —> 3D objects

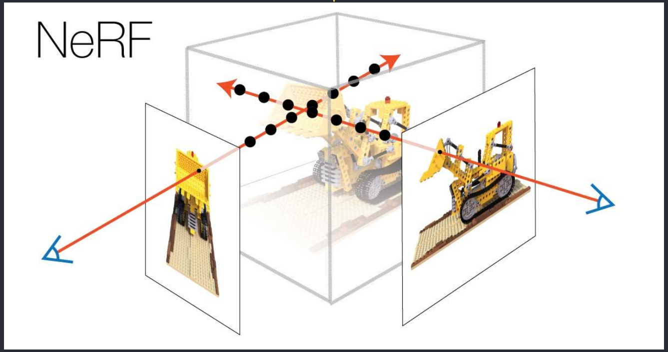


http://graphics.stanford.edu/courses/cs348n-22-winter/

Questions?

Finally, NeRF... For Real!

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



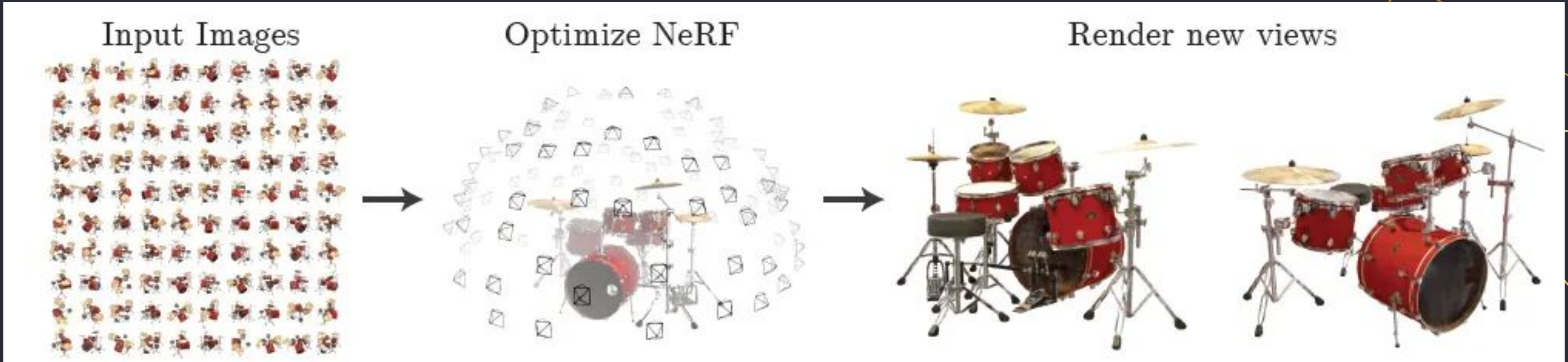
Paper Authors: Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng

https://www.matthewtancik.com/nerf

What is a NeRF?

Technical Definition: Fully-connected neural network that can generate new views of 3D scenes based on a set of 2D images.

*** Think of NeRF as volume rendering + coordinate-based network!**



https://www.matthewtancik.com/nerf

Q. What are we trying to do?

A. We are trying to generate a new picture/view of an object by existing sparse image inputs

Q. How do we do that?

A. Training a neural network which representing the 3D scene

https://www.matthewtancik.com/nerf

NeRF View-Dependent Appearance

https://www.matthewtancik.com/nerf

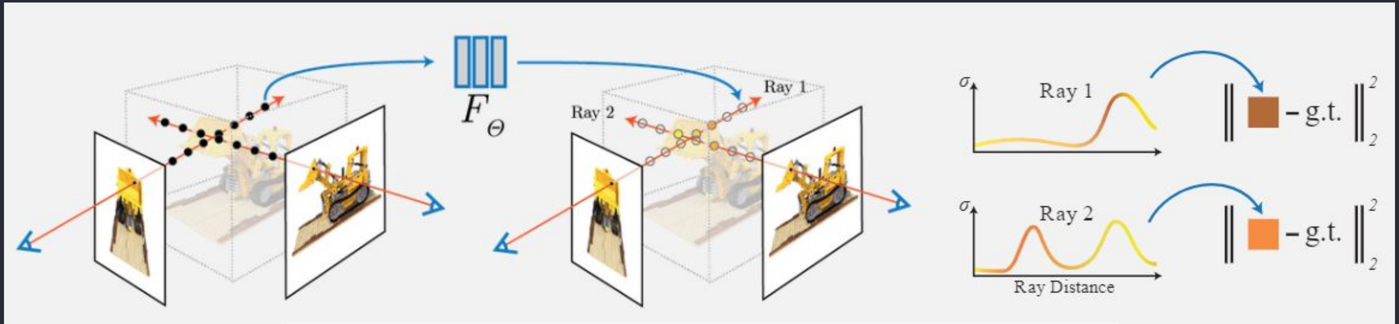
Geometry Visualization

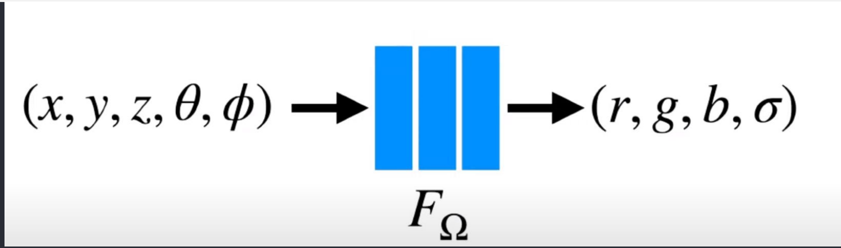
https://www.matthewtancik.com/nerf

Let's dive into the concepts surrounding NeRF!

Brief Overview of NeRF

(a) (b) (c) (d)

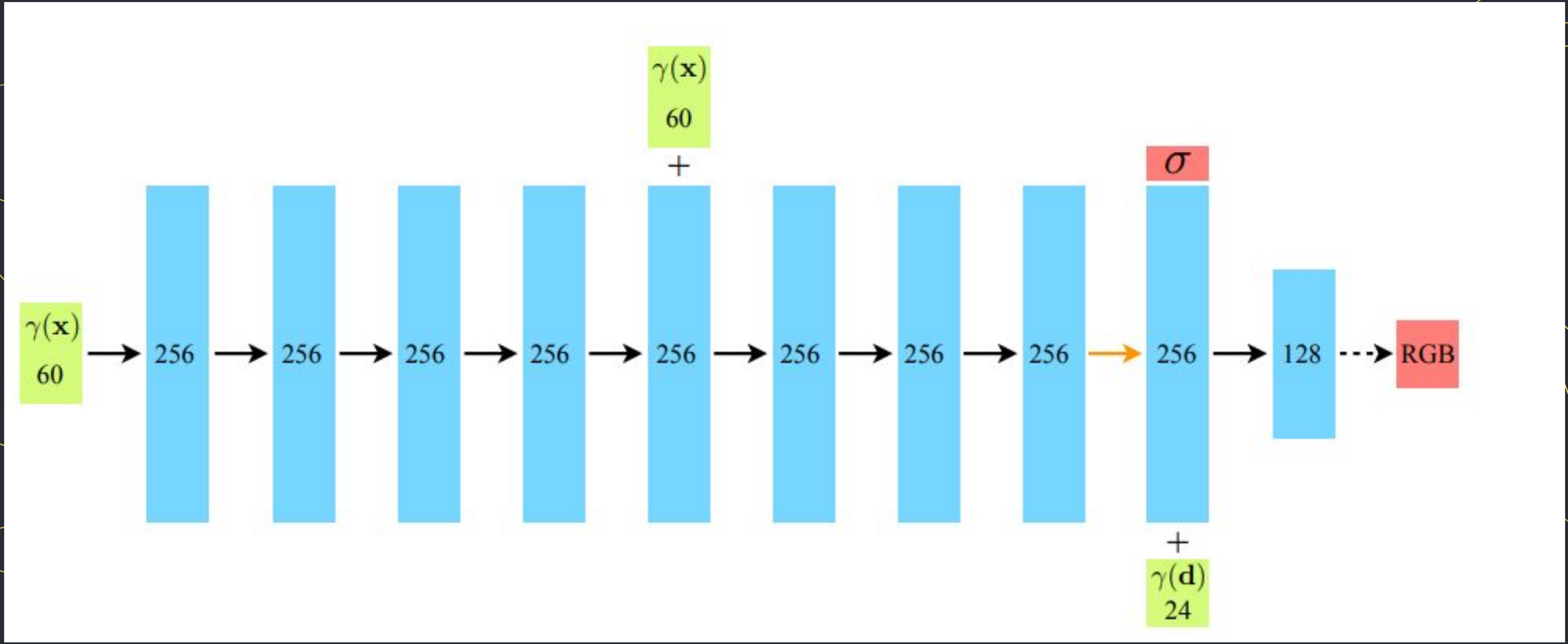




https://www.matthewtancik.com/nerf

https://www.youtube.com/watch?v=WSfEfZ0ilw4

We overfit our MLP to this scene (this is VERY unusual in classical deep learning)!



https://arxiv.org/pdf/2003.08934.pdf

Volume Rendering with Radiance Fields

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Expected color of camera ray

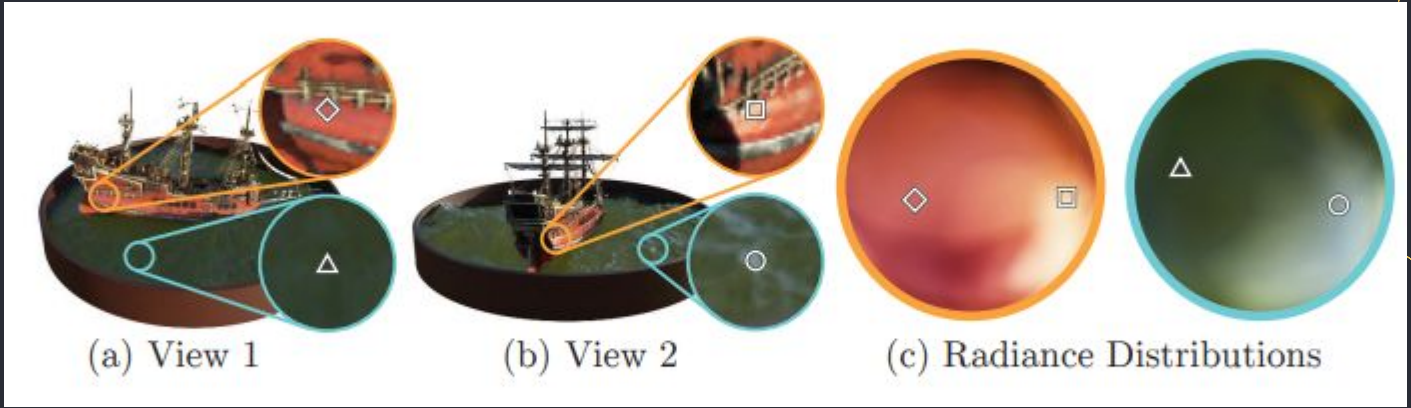
How much light has been blocked up to point t?

Density at the point (in terms of occlusion)

Color at point r(t) from viewing direction d

https://arxiv.org/pdf/2003.08934.pdf

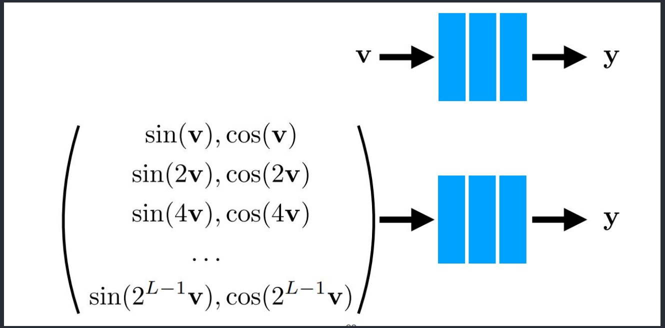
View-Dependent Emitted Radiance



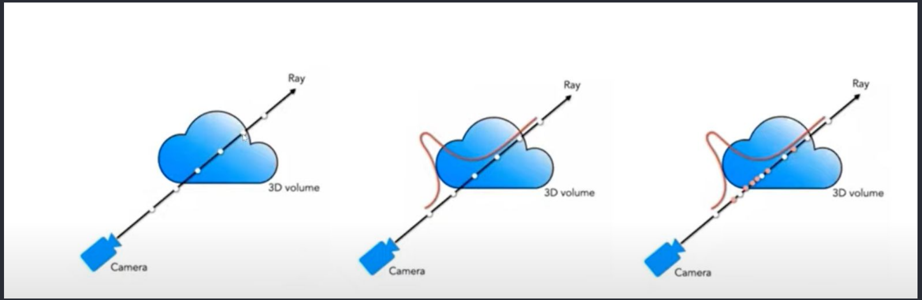
https://arxiv.org/pdf/2003.08934.pdf

Optimizing a Neural Radiance Field

1. Positional Encoding



2. Hierarchical Volume Sampling



https://www.youtube.com/watch?v=WSfEfZ0ilw4

Why do we use Positional Encoding?

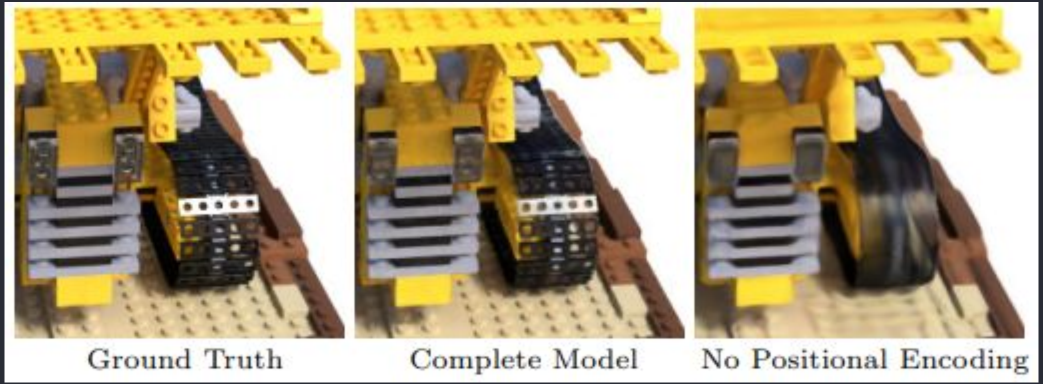
Ground truth image

Neural network output **without** high frequency mapping

Neural network output **with** high frequency mapping

http://graphics.stanford.edu/courses/cs348n-22-winter/

Positional Encoding

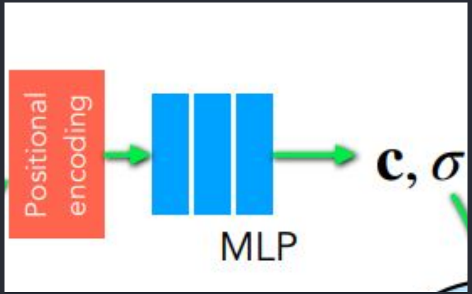


$F_\Theta = F'_\Theta \circ \gamma,$

$\gamma(p) = \left(\sin\left(2^0 \pi p\right), \cos\left(2^0 \pi p\right), \cdots, \sin\left(2^{L-1} \pi p\right), \cos\left(2^{L-1} \pi p\right)\right)$

Compositions of a regular MLP and the mapping to a high-dimensional space

Mapping from R to to R^2L

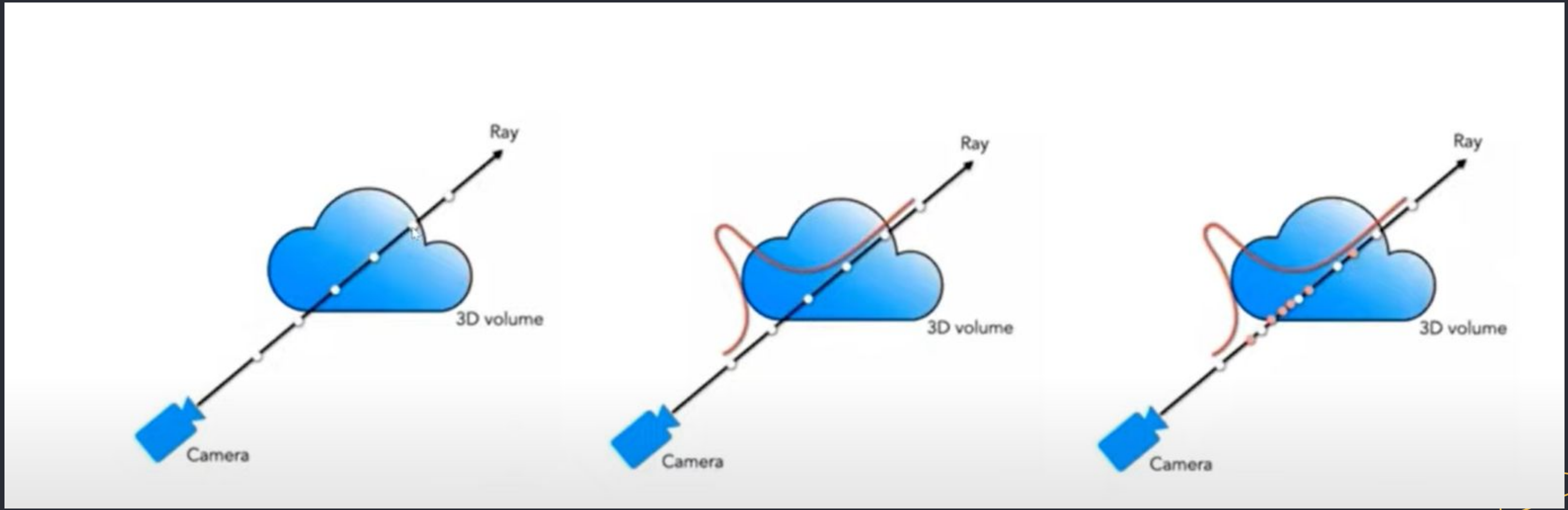


https://arxiv.org/pdf/2003.08934.pdf

http://graphics.stanford.edu/courses/cs348n-22-winter/

Hierarchical Volume Sampling

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \,, \qquad w_i = T_i(1 - \exp(-\sigma_i \delta_i)) .$$

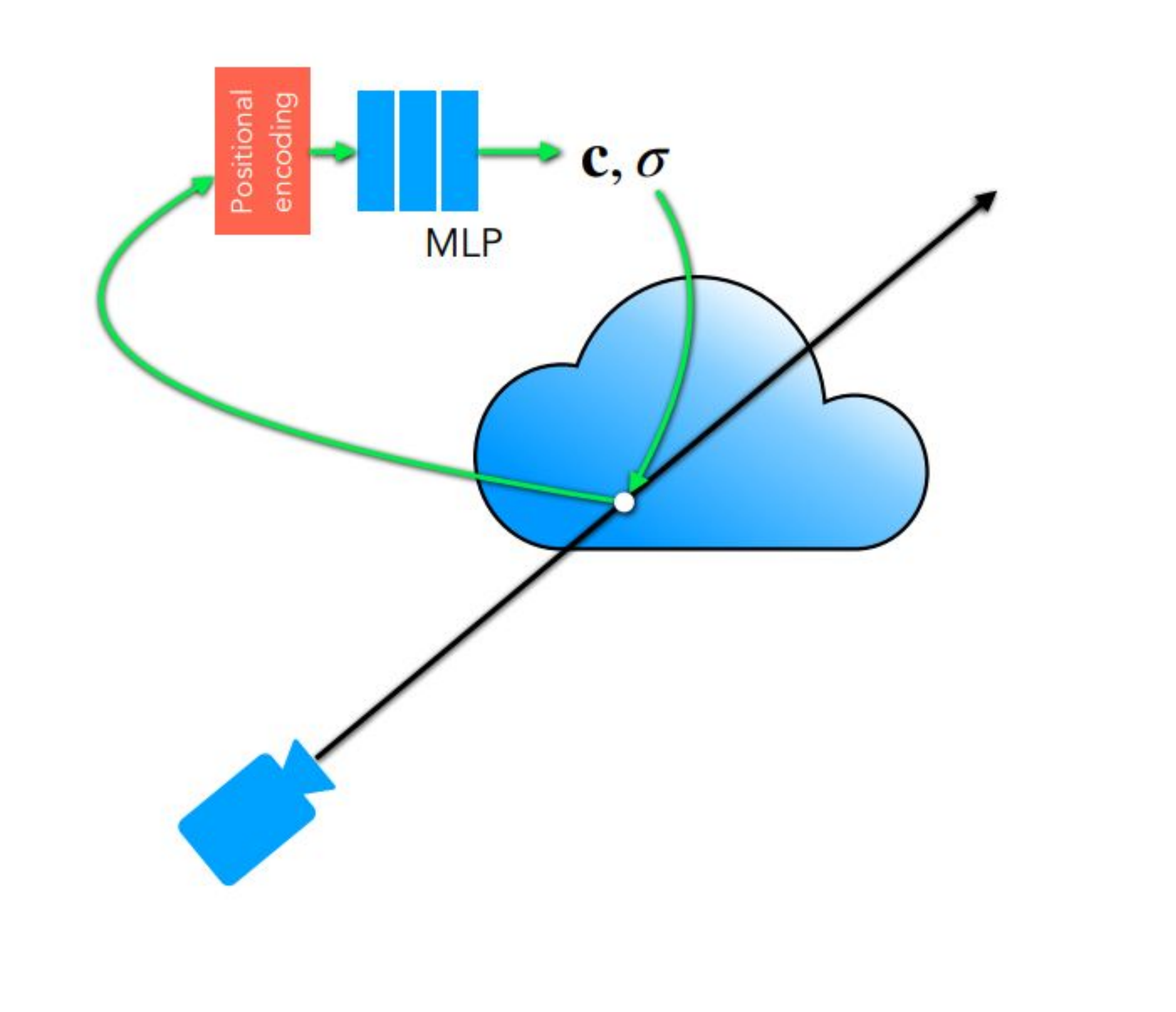


https://arxiv.org/pdf/2003.08934.pdf

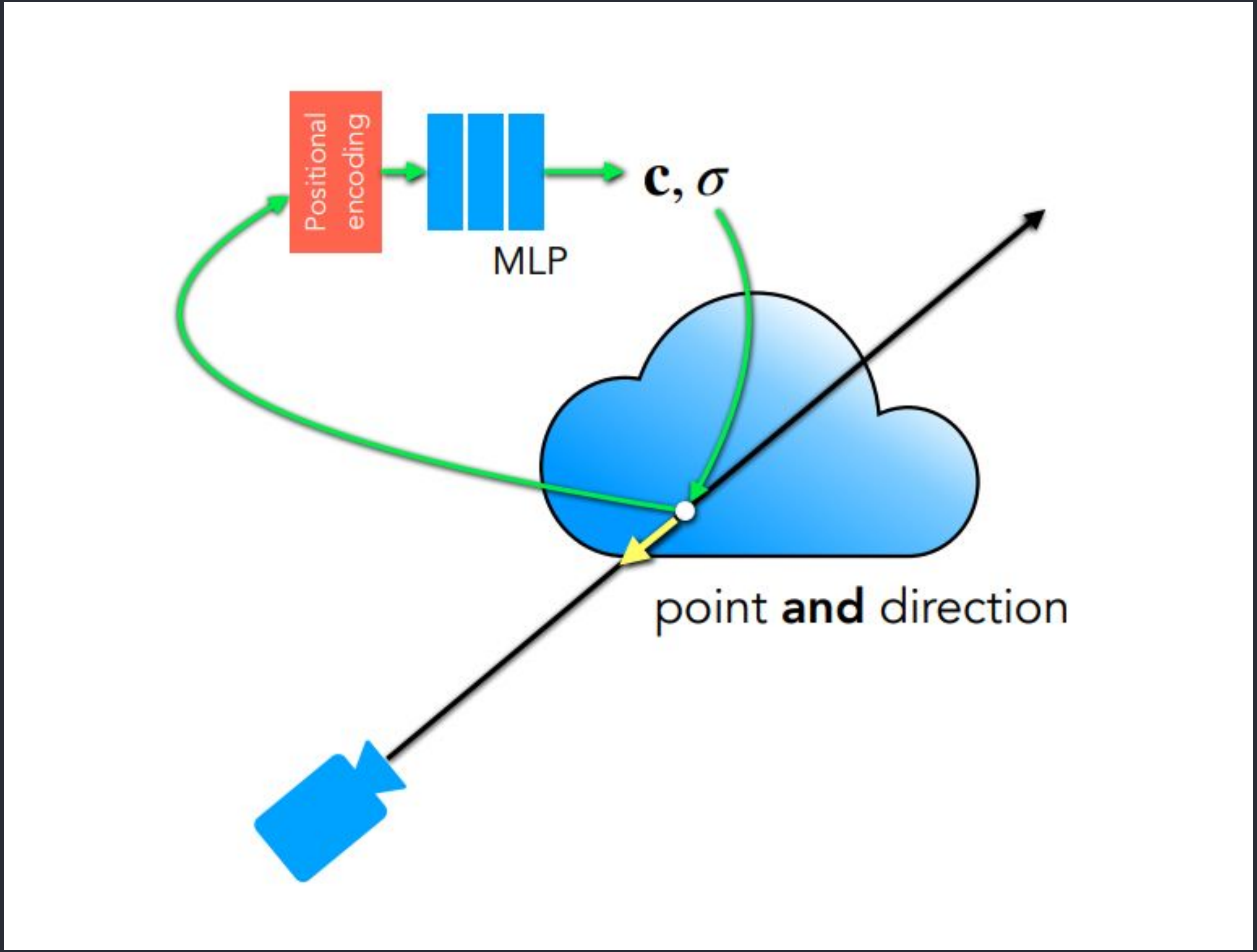
https://www.youtube.com/watch?v=WSfEfZ0ilw4

Phew!!! Let's put everything together!

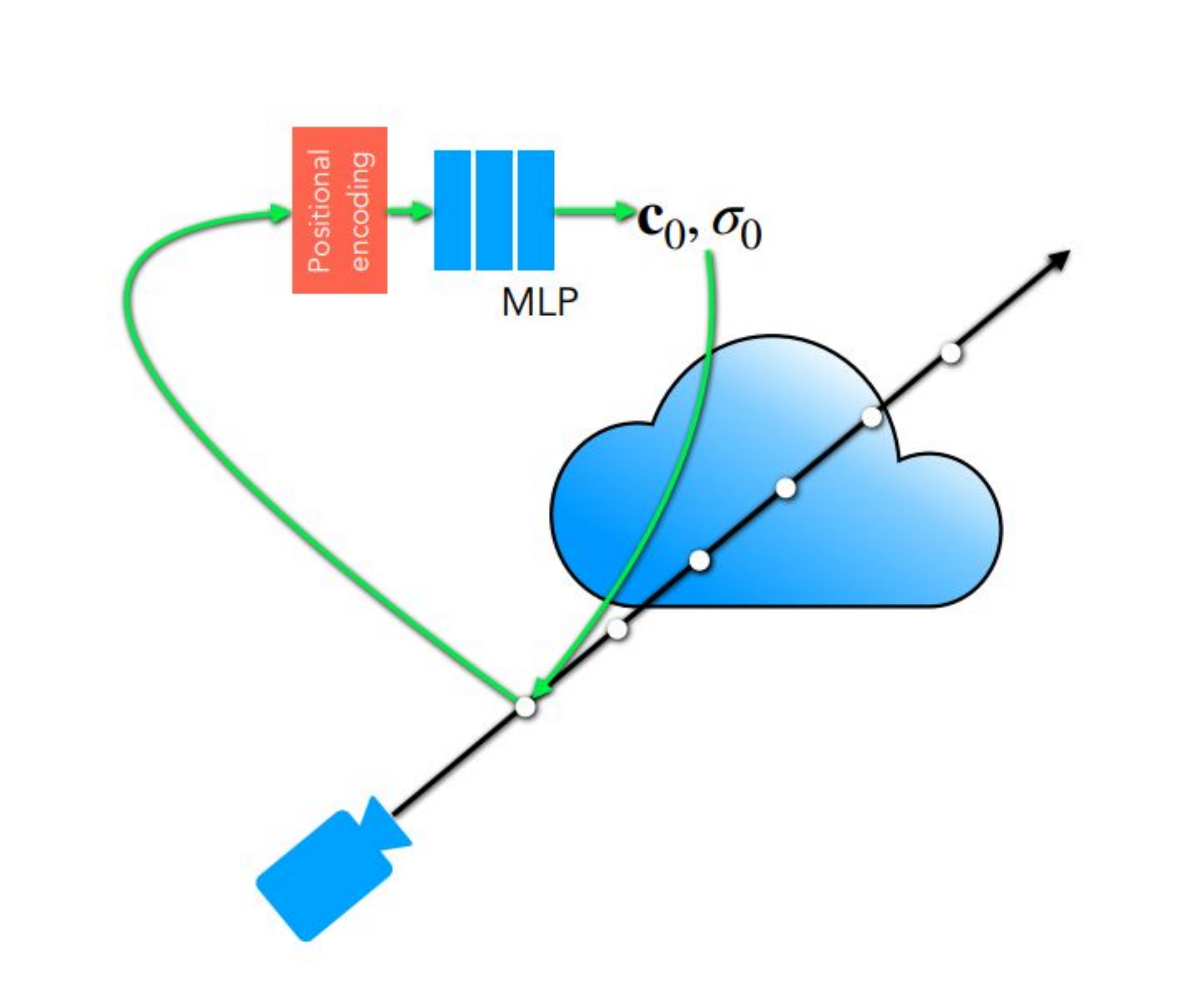
Putting everything together!



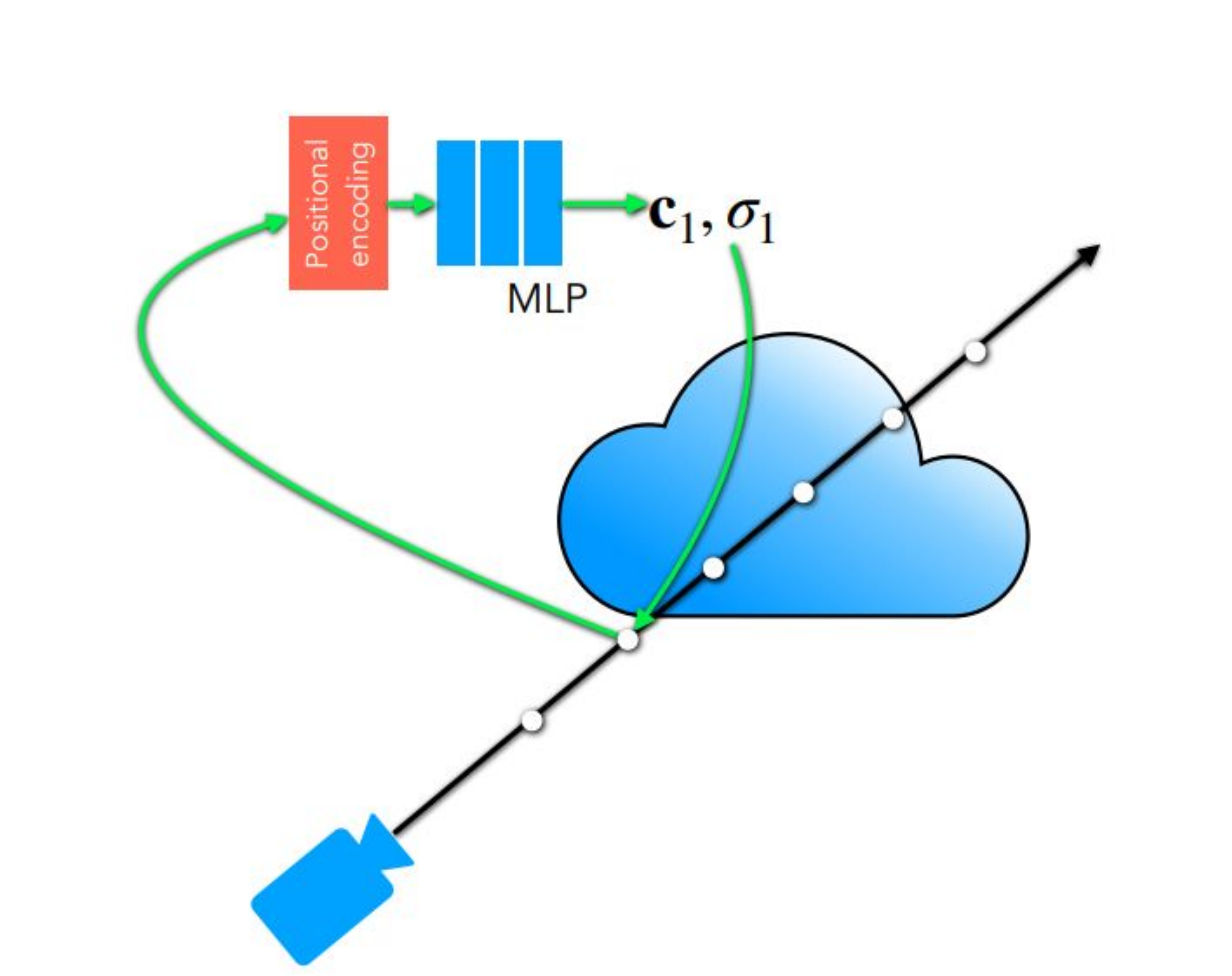
http://graphics.stanford.edu/courses/cs348n-22-winter/



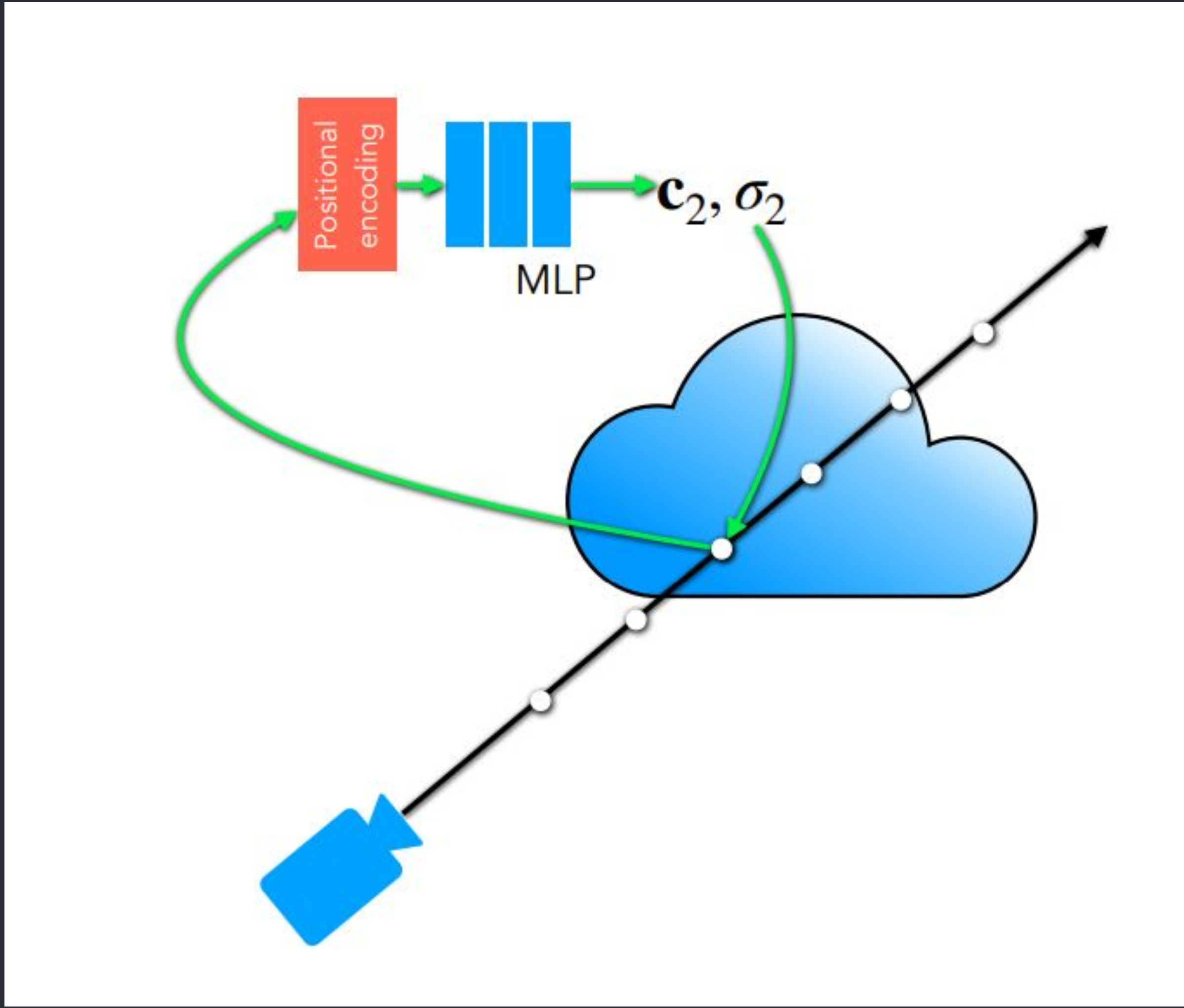
http://graphics.stanford.edu/courses/cs348n-22-winter/



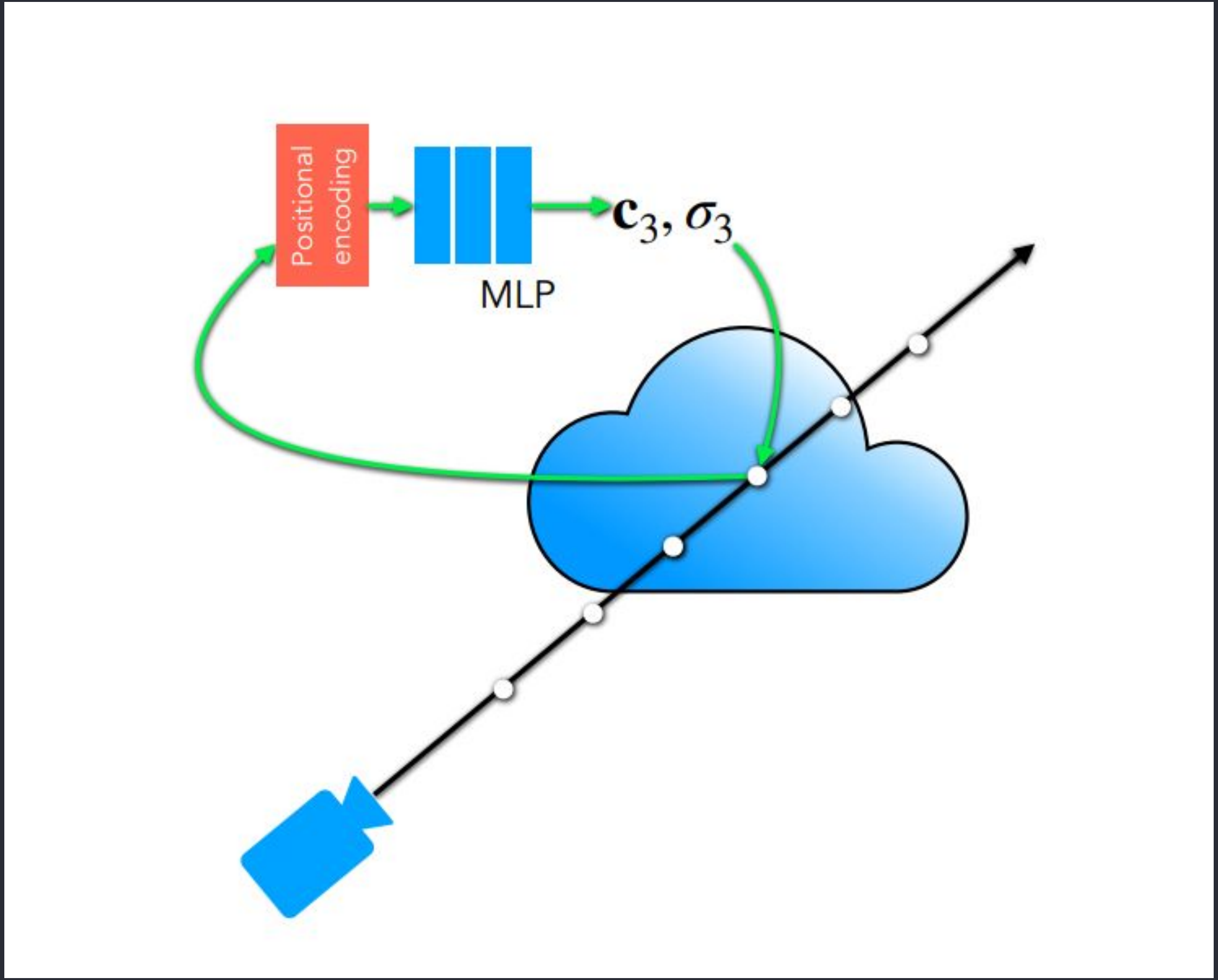
http://graphics.stanford.edu/courses/cs348n-22-winter/



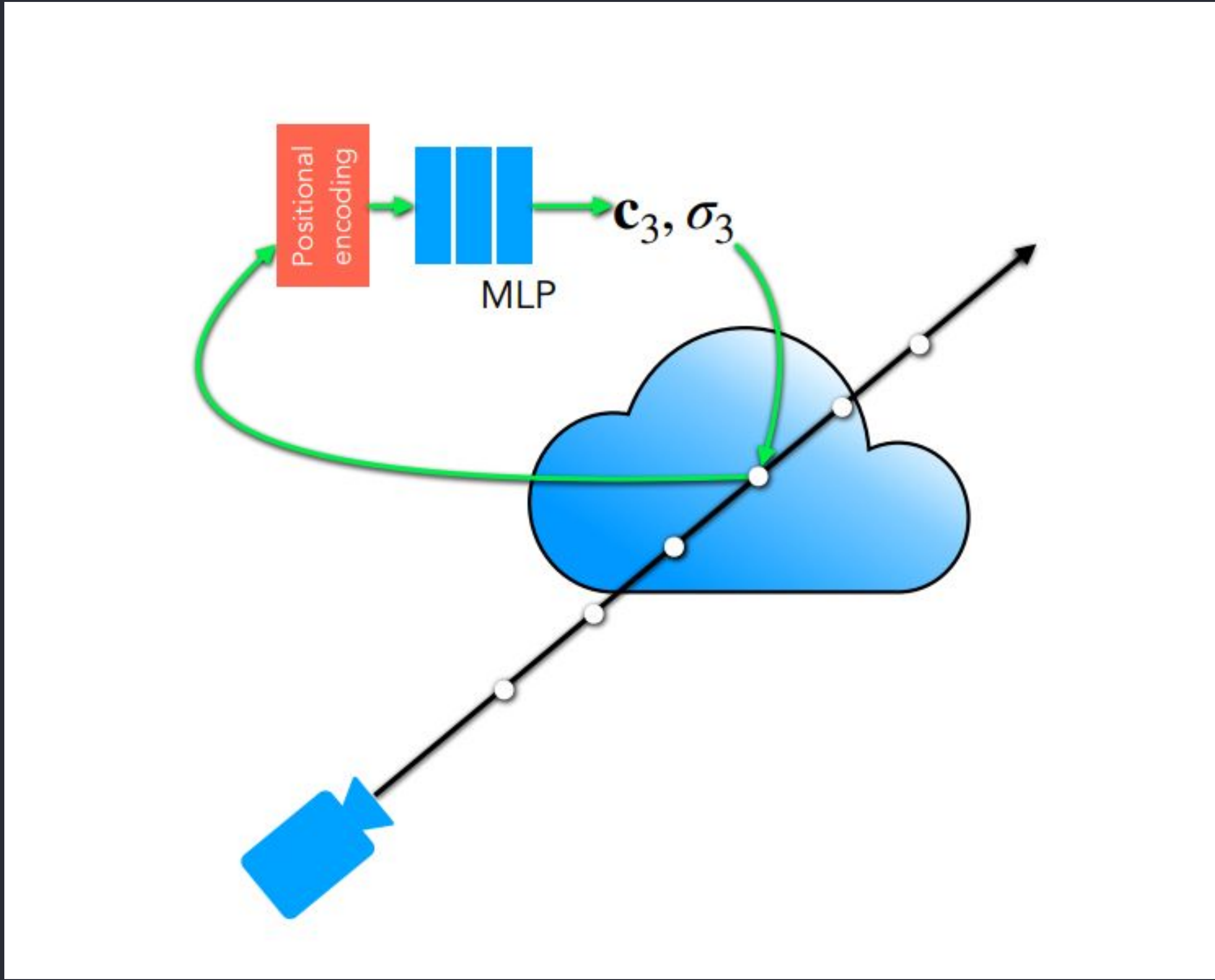
http://graphics.stanford.edu/courses/cs348n-22-winter/



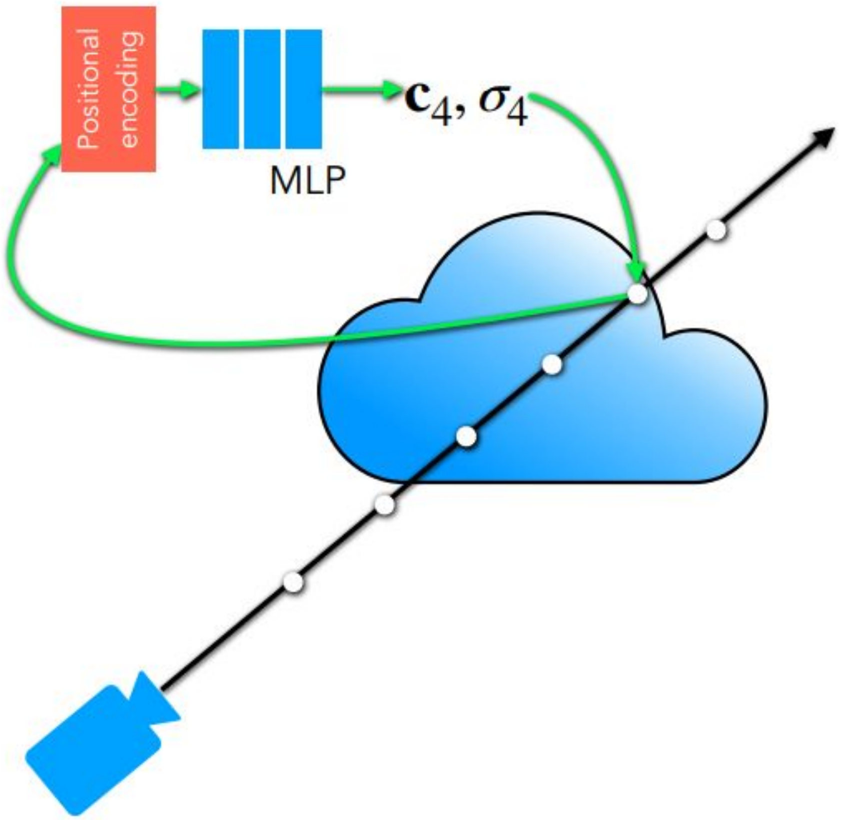
http://graphics.stanford.edu/courses/cs348n-22-winter/



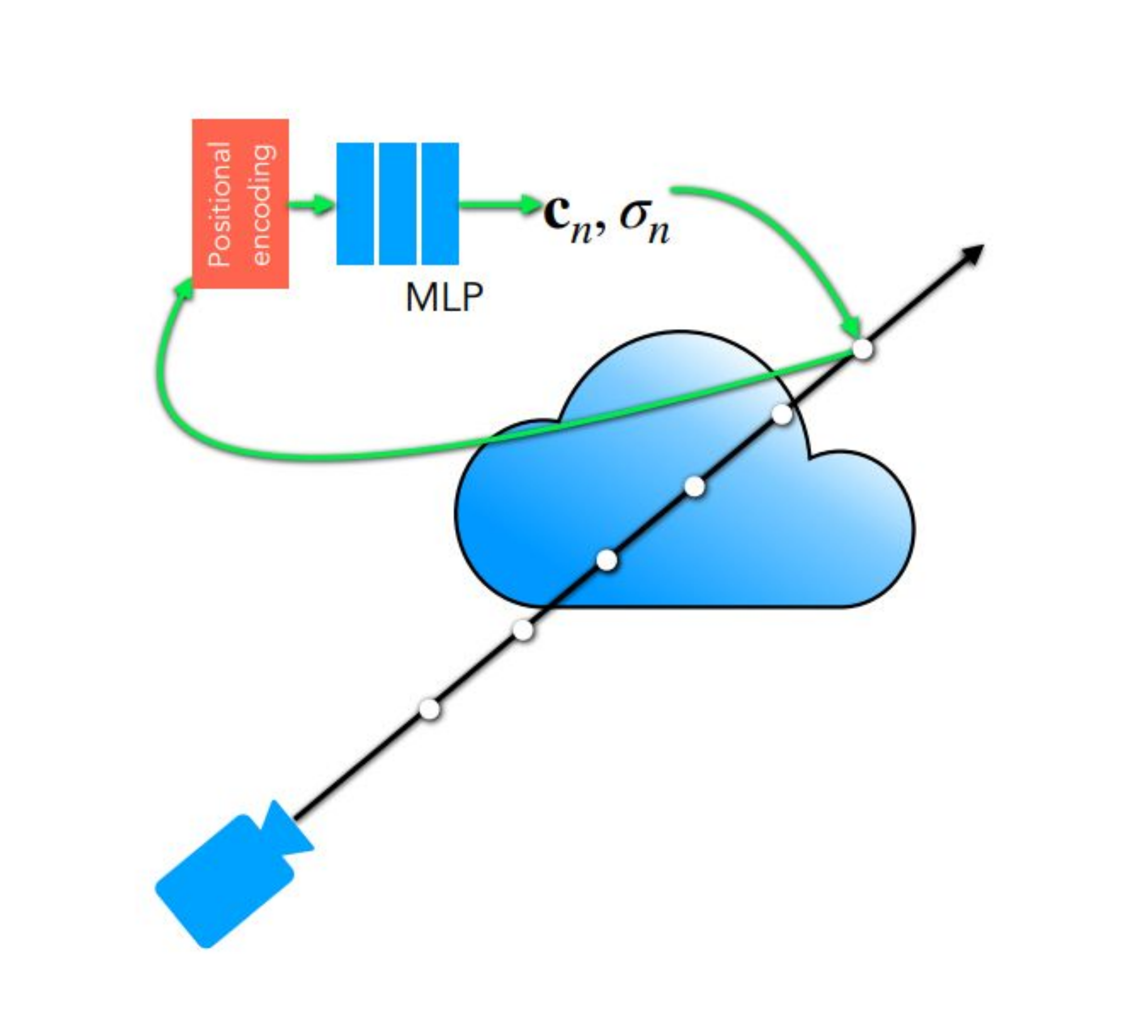
http://graphics.stanford.edu/courses/cs348n-22-winter/



http://graphics.stanford.edu/courses/cs348n-22-winter/



http://graphics.stanford.edu/courses/cs348n-22-winter/



http://graphics.stanford.edu/courses/cs348n-22-winter/

Questions?

It's DATA TIME!!!

Datasets

Realistic Synthetic 360° - 1.56 GB

- Pinecone Images
- Flower Vase Images

Diffuse Synthetic 360° - 1.54 GB

- 8 Objects

https://www.matthewtancik.com/nerf

Data

Dataset	Real or Synthetic Objects	Image Size	Training Images	Testing Images
LLFF	Real	1008 × 756	20 - 62	⅛ of Training Images
Diffuse Synthetic 360	Synthetic	512 × 512	479	1000
Realistic Synthetic 360	Synthetic	512 × 512	479	1000
DeepVoxels	Synthetic	512 × 512	479	1000

NeRF Computation/Memory Requirements

- Memory Requirements for Network Weights: 5 MB
- Training Computation Time: ~15 hours (after 200k iterations)
- Testing Computation Time: between 30 sec and 1 minute
- Training computation time can vary depending on the resolution

Let's talk training!

Training

Generate random seed [if not provided]

```
if args.random_seed is not None:
        print('Fixing random seed', args.random_seed)
        np.random.seed(args.random_seed)
        tf.compat.v1.set_random_seed(args.random_seed)
```

https://github.com/bmild/nerf

Training

Split data into sets (“Train”,"Validation", “Testing”)

```
if args.dataset_type == 'llff':
      images, poses, bds, render_poses, i_test =load_llff_data(args.datadir, args.factor,
                                                          recenter=True)
elif args.dataset_type == 'blender':
      ………..
else:
        ………..

i_train, i_val, i_test= i_split
```

```
def load_blender_data(basedir, half_res=False, testskip=1):
    splits = ['train', 'val', 'test']
    metas = {}
    for s in splits:
        with open(os.path.join(basedir, 'transforms_{}.json'.format(s)), 'r') as fp:
            metas[s] = json.load(fp)

    all_imgs = []
    all_poses = []
    counts = [0]
    for s in splits:
        meta = metas[s]
        imgs = []
        poses = []
        if s=='train' or testskip==0:
            skip = 1
        else:
            skip = testskip

        for frame in meta['frames'][::skip]:
            fname = os.path.join(basedir, frame['file_path'] + '.png')
            imgs.append(imageio.imread(fname))
            poses.append(np.array(frame['transform_matrix']))
        imgs = (np.array(imgs) / 255.).astype(np.float32) # keep all 4 channels (RGBA)
        poses = np.array(poses).astype(np.float32)
        counts.append(counts[-1] + imgs.shape[0])
        all_imgs.append(imgs)
        all_poses.append(poses)

    i_split = [np.arange(counts[i], counts[i+1]) for i in range(3)]

    imgs = np.concatenate(all_imgs, 0)
    poses = np.concatenate(all_poses, 0)

    H, W = imgs[0].shape[:2]
    camera_angle_x = float(meta['camera_angle_x'])
    focal = .5 * W / np.tan(.5 * camera_angle_x)

    render_poses = tf.stack([pose_spherical(angle, -30.0, 4.0) for angle in np.linspace(-180,180,40+1)[:-1]],0)

    if half_res:
        imgs = tf.image.resize_area(imgs, [400, 400]).numpy()
        H = H//2
        W = W//2
        focal = focal/2.

    return imgs, poses, render_poses, [H, W, focal], i_split
```

https://github.com/bmild/nerf

Training

Create Adam Optimizer Object

Initial Learning rate: 5×10^{-4}
{exponentially decays to 5×10^{-5}}
Other hyperparameters set to Adam default (β_1 = 0.9, β_2 = 0.999, and ε= 10−7)

```
# Create optimizer
    lrate = args.lrate
    if args.lrate_decay > 0:
        lrate =
tf.keras.optimizers.schedules.ExponentialDecay(lrate,

decay_steps=args.lrate_decay * 1000, decay_rate=0.1)
    optimizer = tf.keras.optimizers.Adam(lrate)
    models['optimizer'] = optimizer
```

https://github.com/bmild/nerf

Training

Conduct training for N iterations. At each step:

a. Sample random data
b. Make predictions for parameters
c. Computer loss (MSE)
d. Add in the loss for the coarse grained model
e. Apply the gradients

https://github.com/bmild/nerf

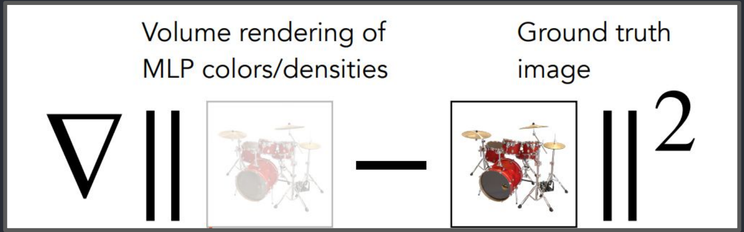
```
N_iters = 1000000
for i in range(start, N_iters):
        # Random from one image
        img_i = np.random.choice(i_train)
        target = images[img_i]


        # Make predictions for color, disparity, accumulated opacity.
        rgb, disp, acc, extras = render(
            H, W, focal, chunk=args.chunk, rays=batch_rays)
        # Compute MSE loss between predicted and true RGB
        img_loss = img2mse(rgb, target_s)
        loss = img_loss
        psnr = mse2psnr(img_loss)

        # Add MSE loss for coarse-grained model
        if 'rgb0' in extras:
            img_loss0 = img2mse(extras['rgb0'], target_s)
            loss += img_loss0
            psnr0 = mse2psnr(img_loss0)

       optimizer.apply_gradients(zip(gradients, grad_vars))
```

Loss Function

The total squared error between the rendered and true pixel colors for both the coarse and fine renderings:

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

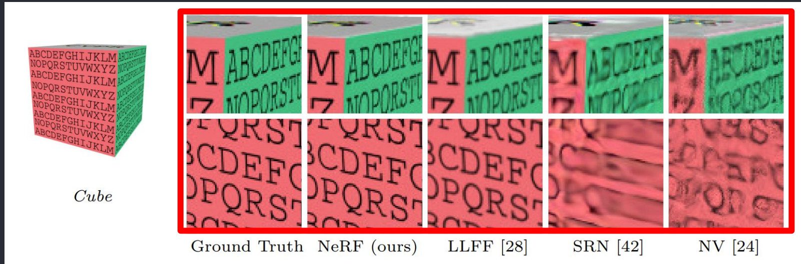
```
def img2mse(x, y): return tf.reduce_mean(tf.square(x - y))
```



https://arxiv.org/pdf/2003.08934.pdf

http://graphics.stanford.edu/courses/cs348n-22-winter/

Visual Results



https://arxiv.org/pdf/2003.08934.pdf

Comparative Results

Method	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

$$MSE = \frac{1}{c*i*j}\sum(I_1 - I_2)^2$$

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

with:

- μ_x the pixel sample mean of x;
- μ_y the pixel sample mean of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1L)^2$, $c_2 = (k_2L)^2$ two variables to stabilize the division with weak denominator;
- L the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

LPIPS: https://arxiv.org/pdf/1801.03924.pdf

https://arxiv.org/pdf/2003.08934.pdf

Ablation Study Results

	Input	#Im.	L	(N_c, N_f)	PSNR↑	SSIM↑	LPIPS↓
1) No PE, VD, H	xyz	100	-	(256, -)	26.67	0.906	0.136
2) No Pos. Encoding	$xyz\theta\phi$	100	-	(64, 128)	28.77	0.924	0.108
3) No View Dependence	xyz	100	10	(64, 128)	27.66	0.925	0.117
4) No Hierarchical	$xyz\theta\phi$	100	10	(256, -)	30.06	0.938	0.109
5) Far Fewer Images	$xyz\theta\phi$	25	10	(64, 128)	27.78	0.925	0.107
6) Fewer Images	$xyz\theta\phi$	50	10	(64, 128)	29.79	0.940	0.096
7) Fewer Frequencies	$xyz\theta\phi$	100	5	(64, 128)	30.59	0.944	0.088
8) More Frequencies	$xyz\theta\phi$	100	15	(64, 128)	30.81	0.946	0.096
9) Complete Model	$xyz\theta\phi$	100	10	(64, 128)	**31.01**	**0.947**	**0.081**

Table 2: An ablation study of our model. Metrics are averaged over the 8 scenes from our realistic synthetic dataset. See Sec. 6.4 for detailed descriptions.

https://arxiv.org/pdf/2003.08934.pdf

Summary

- NeRF = Coordinate Based MLP Network + Volume Rendering
- NeRF Pros:
  - Simple representation
  - Differentiable Rendering Model
- NeRF Cons:
  - Dumb Brute force approach
  - INSANELY SLOW!!!

	Training speed	Rendering speed
Original NeRF	1-2 days	30 sec
KiloNeRF, cached voxels	1-2 days	1/60 sec
Learned voxels	10-15 mins	1/15-1/2 sec
Learned hash maps (Instant NGP)	5 sec - 5 mins	1/60 sec

http://graphics.stanford.edu/courses/cs348n-22-winter/

Questions?

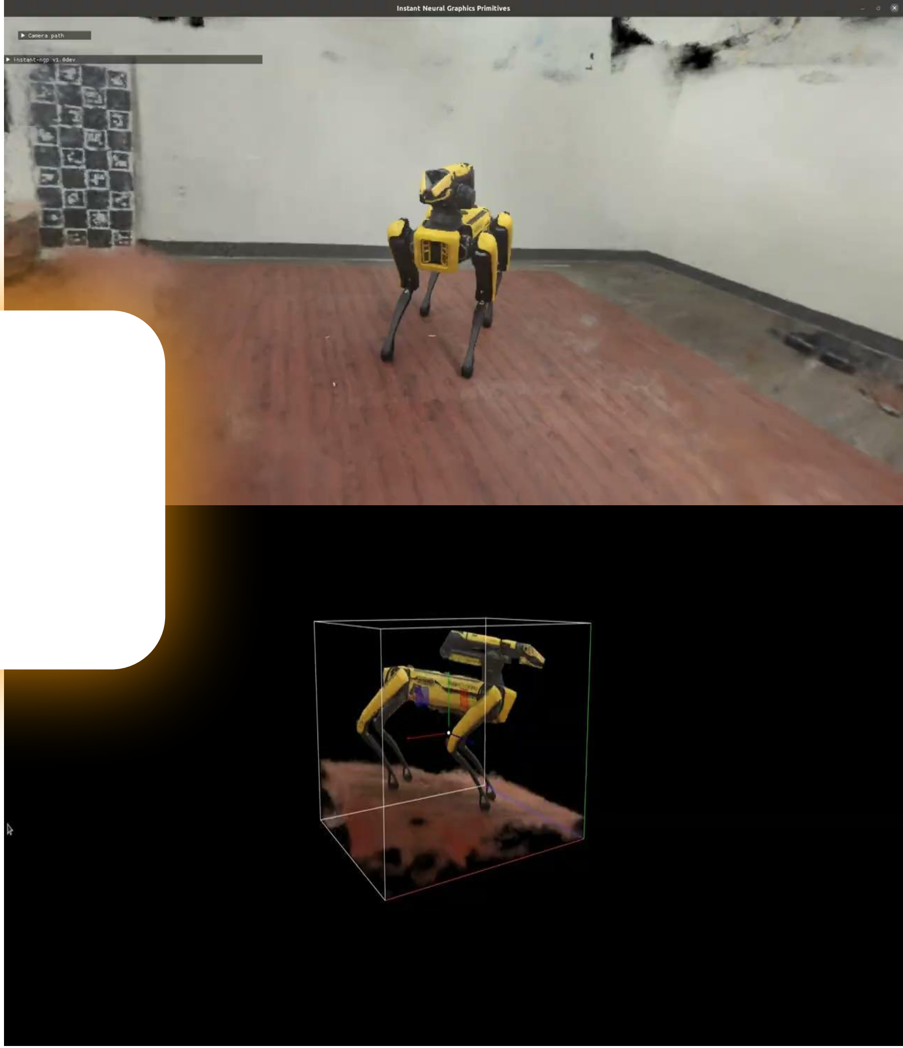
DeepRob

[Student] Lecture 17

by Michael Andrev, Chen Hu, Sean Coffey

Implicit Surfaces, Geometry, NeRF

University of Michigan and University of Minnesota



Video credits - Nirmal Raj

Sean References

Moons, Theo, Luc Van Gool, and Maarten Vergauwen. "3D reconstruction from multiple images part 1: Principles." Foundations and Trends in Computer Graphics and Vision 4.4 (2010): 287-404.

Mahmoudzadeh, Ahmadreza; Golroo, Amir; Jahanshahi, Mohammad R.; Firoozi Yeganeh, Sayna (January 2019). "Estimating Pavement Roughness by Fusing Color and Depth Data Obtained from an Inexpensive RGB-D Sensor". Sensors. 19 (7): 1655. Bibcode:2019Senso..19.1655M. doi:10.3390/s19071655. PMC 6479490. PMID 30959936

Buelthoff, Heinrich H., and Alan L. Yuille. "Shape-from-X: Psychophysics and computation Archived 2011-01-07 at the Wayback Machine." Fibers' 91, Boston, MA. International Society for Optics and Photonics, 1991

Richard Hartley and Andrew Zisserman (2003). “Multiple View Geometry in computer vision”. Cambridge University Press. ISBN 978-0-521-54051-3

Yuan Yao, Yasamin Jafarian, & Hyun Soo Park. (2019). “MONET: Multiview Semi-supervised Keypoint Detection via Epipolar Divergence.”arXiv,1806.00104. https://arxiv.org/pdf/1806.00104.pdf