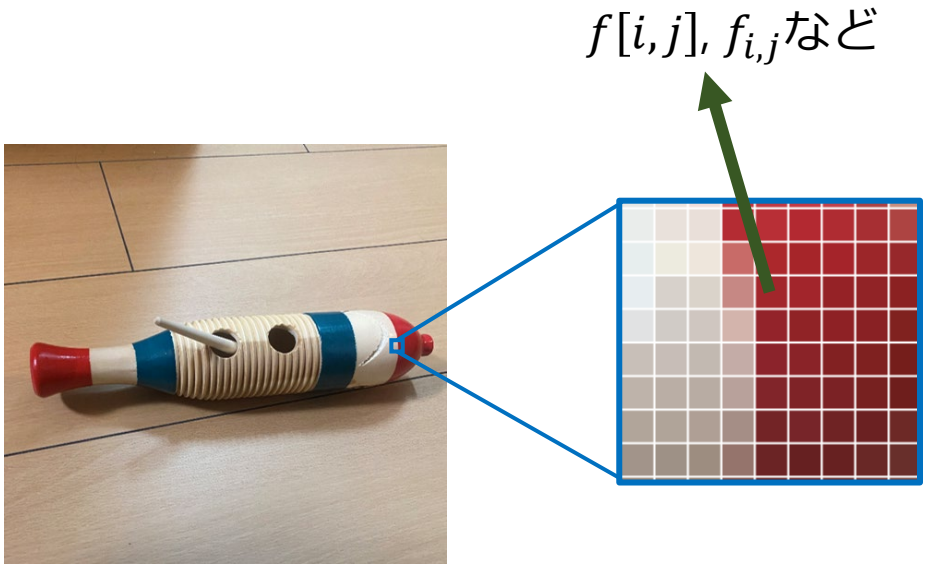
Neural Fieldの紹介

大阪大学 千葉

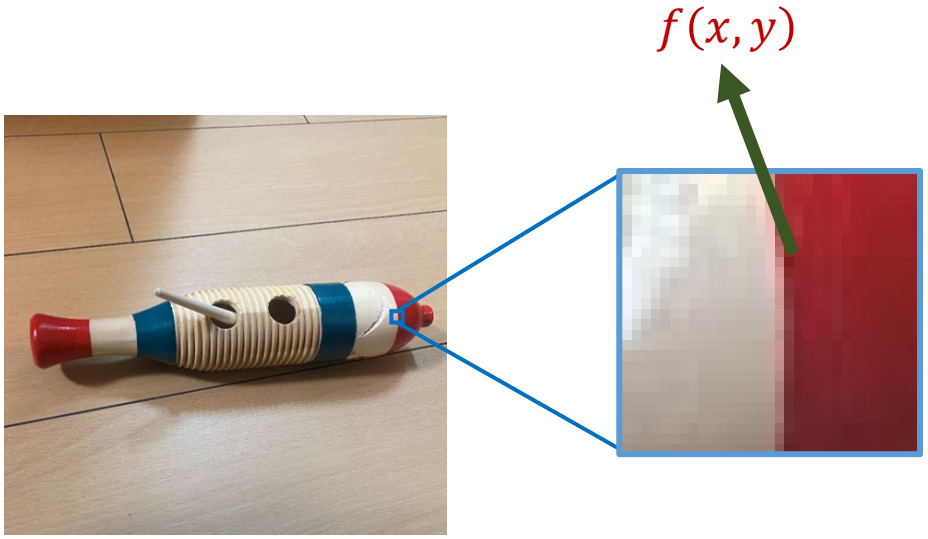
Neural Fieldとは

空間中の（連続な）**各点xに対応した関数値$f(x)$**を
ニューラルネットワークでモデル化

- 画像などは要素を列挙した離散な表現
- 関数として**連続な表現**として学習させる



画像：離散な表現



Neural Field：連続な関数で表現

Neural Fieldの登場した経緯

2016-2018年頃

ニューラルネットワークによる3Dデータ処理が発展（ボクセル，点群，メッシュなど）

　これらの表現には種々の限界（後述）があった

2019年

精緻な三次元形状を記述できる＆
ニューラルネットワークと相性のよい手法
→ **DeepSDF**（等）が提案された

2020年

（別の文脈で）**NeRF**が登場（後述）・発展

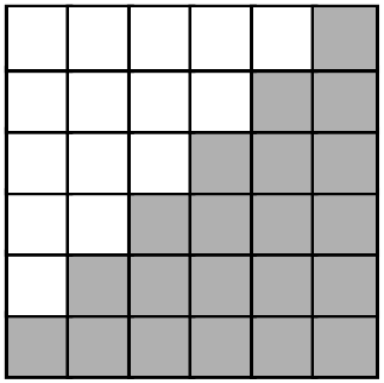
→ これらが徐々に整理され，Neural Field系の技術として発展

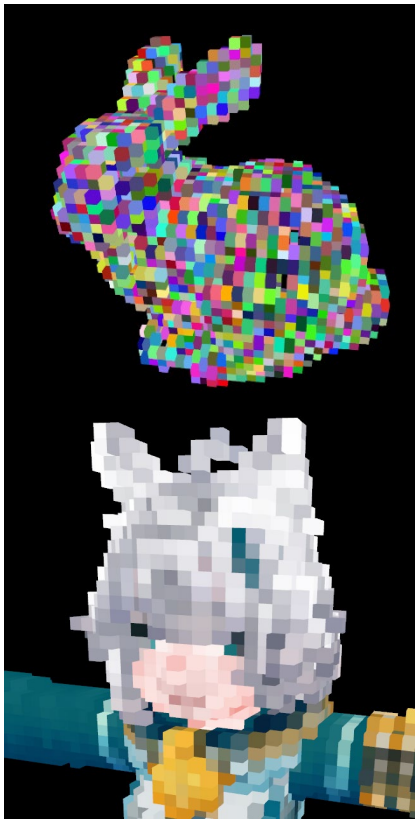
Neural Field以前の3D形状表現

代表的な3D形状表現を紹介

ボクセル

- グリッド上に情報を並べて表現
- 空間解像度の3乗でメモリを消費するため空間解像度が上げにくい





*ボクセルの例

空間座標：**離散**
データ点：**離散**
表面の記述：**あり**

*: L. Mescheder+. Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR2019

Neural Field以前の3D形状表現

代表的な3D形状表現を紹介

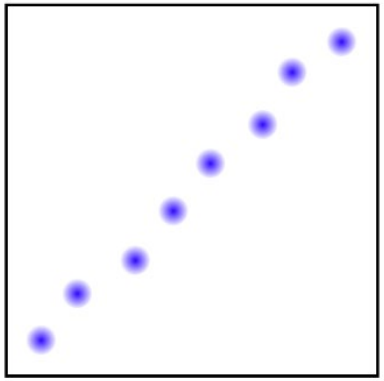
点群

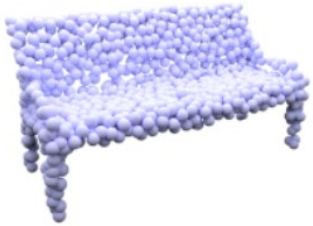
- 三次元点の座標の集合で形状を記述する
- 非グリッドな記述でありメモリ効率よく空間解像度の高い表現
- 点群深層学習が発展
- 明示的に表面は記述していない

空間座標：**連続**

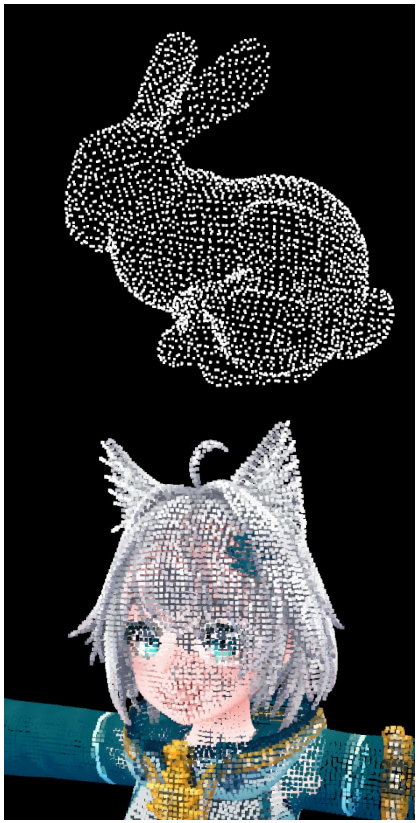
データ点：**離散**

表面の記述：**なし**





*点群の例



*: L. Mescheder+. Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR2019

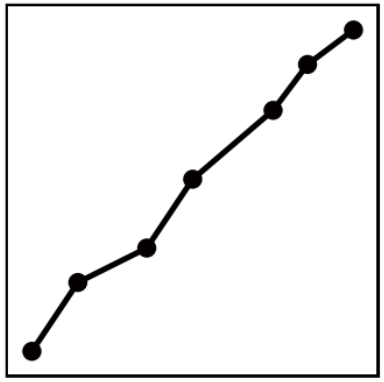
Neural Field以前の3D形状表現

代表的な3D形状表現を紹介

メッシュ

- 頂点・（辺・）面の集合で形状を記述
- 非グリッドな記述
- 頂点の座標が離散なので滑らかな表現は難しい

空間座標：**連続**
データ点：**離散**
表面の記述：**あり**



*メッシュの例

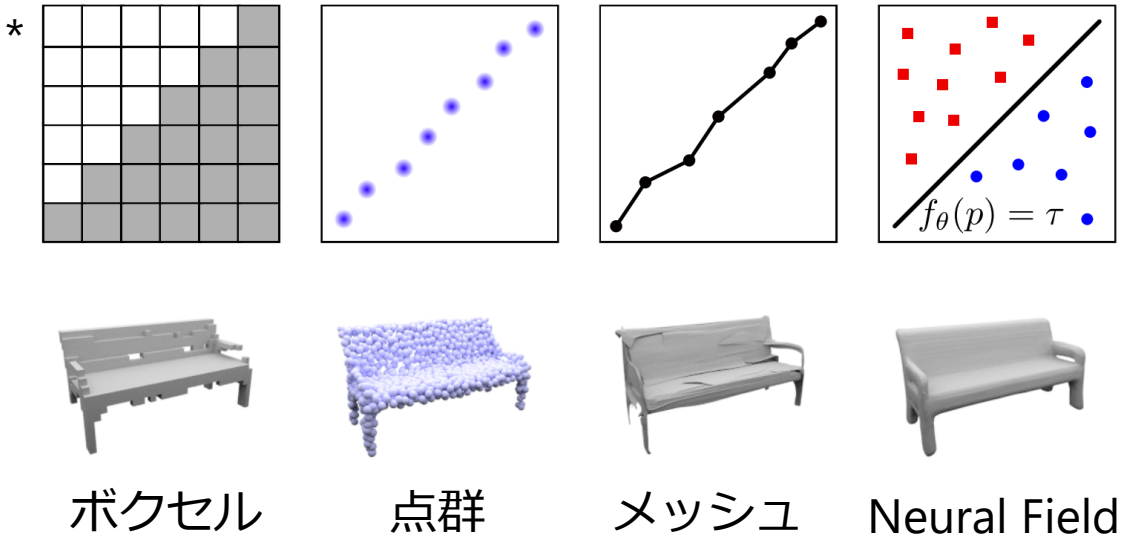
*: L. Mescheder+. Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR2019

Neural Fieldによる表面モデル

ボクセル・点群・メッシュなどは陽なデータ表現
＝直接データ点を保持，記述
　これらの非構造データの処理も面白い課題，今回は省略

Neural Fieldを用いた表面モデル＝
Implicit Surface Representationが登場

空間座標：**連続**
データ点：**連続**
　離散な点について
　好きなだけクエリする
表面の記述：**あり**



*: L. Mescheder+. Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR2019

Neural Fieldによる表面モデル

Neural Fieldを用いた表面モデル

- **DeepSDF**（今回主に紹介）
- Occupancy Network
- IM-Net

がほぼ同時期（すべてCVPR2019）に提案

ニューラルネットワークで陰に表面形状を記述するというアイデアは共通

どのような関数を経由するか・表現の利用方法に差異

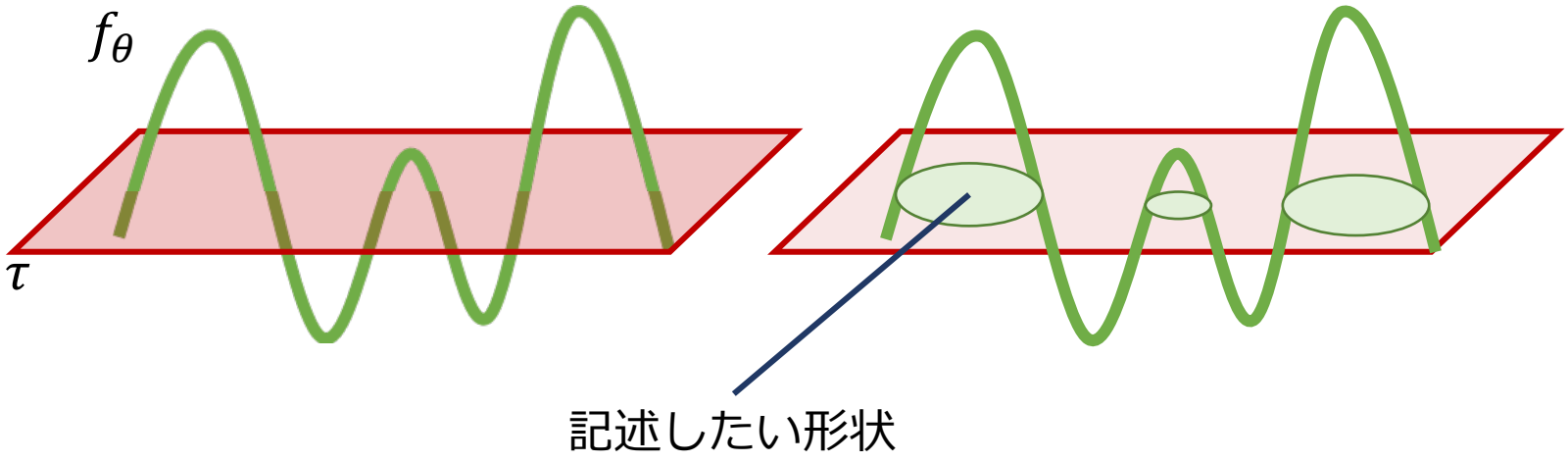
Fieldによる表面形状表現

Fieldによる表面形状の記述とは

空間中の各点での値を返す関数の**等高面で形状を記述**

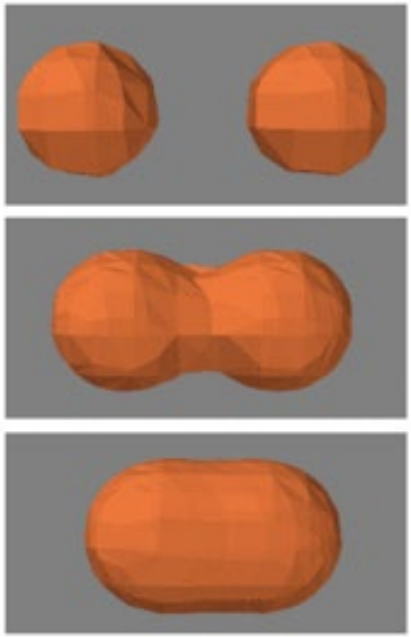
$f(x;\theta) = \tau$

- θ: 形状を表すパラメータ
- x: 座標
- τ: 等高面の高さ



Implicit Function系手法の紹介

- メタボール：濃度分布をもつ球の集合で形状を定義
  等高線で表面形状を表す



- 陰関数表現：メタボールの一般化
  - 内側・外側で符号を変えることで向きを表現（Signed Distance Function: SDF）
  - SDFの勾配方向を計算することで法線も同時に得られる
  - Marching Cubes法などの等値面抽出手法でメッシュを再構成できる

William E. Lorensen+, Marching Cubes: A high resolution 3D surface construction algorithm. SIGGRAPH1987.

Wikipedia: メタボール

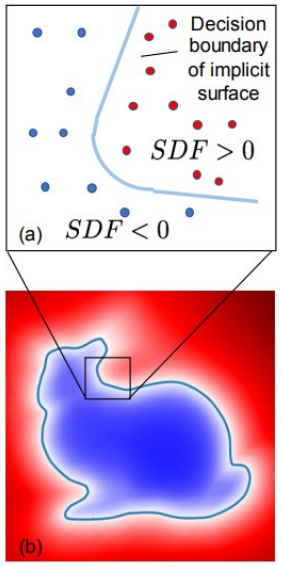
DeepSDFの紹介

形状ごとのSigned Distance Function (SDF)を
ニューラルネットワークで学習

- シンプルなネットワークで三次元形状が記述できる
- Neural Fieldなのでなめらかな表面が表せる

Signed Distance Function (SDF)

$$f(\boldsymbol{x}) \begin{cases} > 0 & \text{（物体の外側）} \\ = 0 & \text{（物体の表面）} \\ < 0 & \text{（物体の内側）} \end{cases}$$



(c)

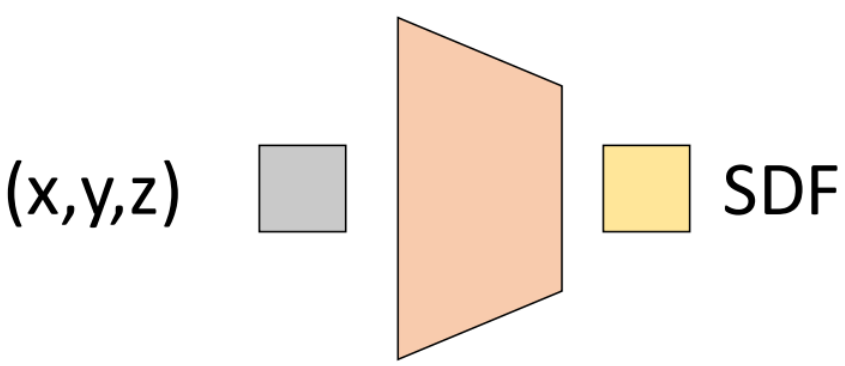
DeepSDFでのモデル

単一形状を記述する場合

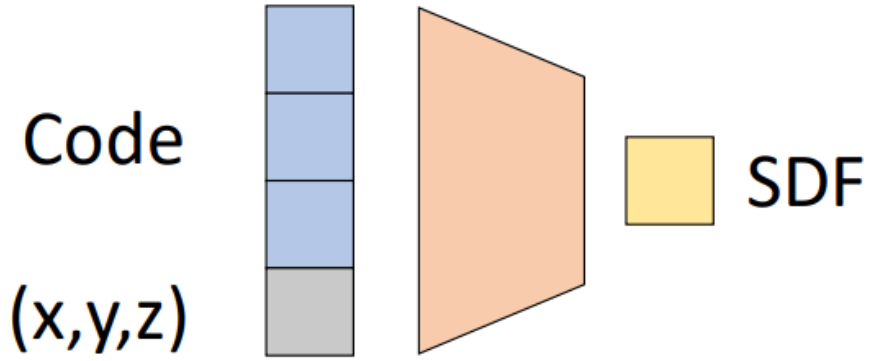
座標を入力，SDFを出力すればよい

複数形状を記述する場合

形状ごとの潜在ベクトル（Code）と座標を入力，SDFを出力する



単一形状の場合

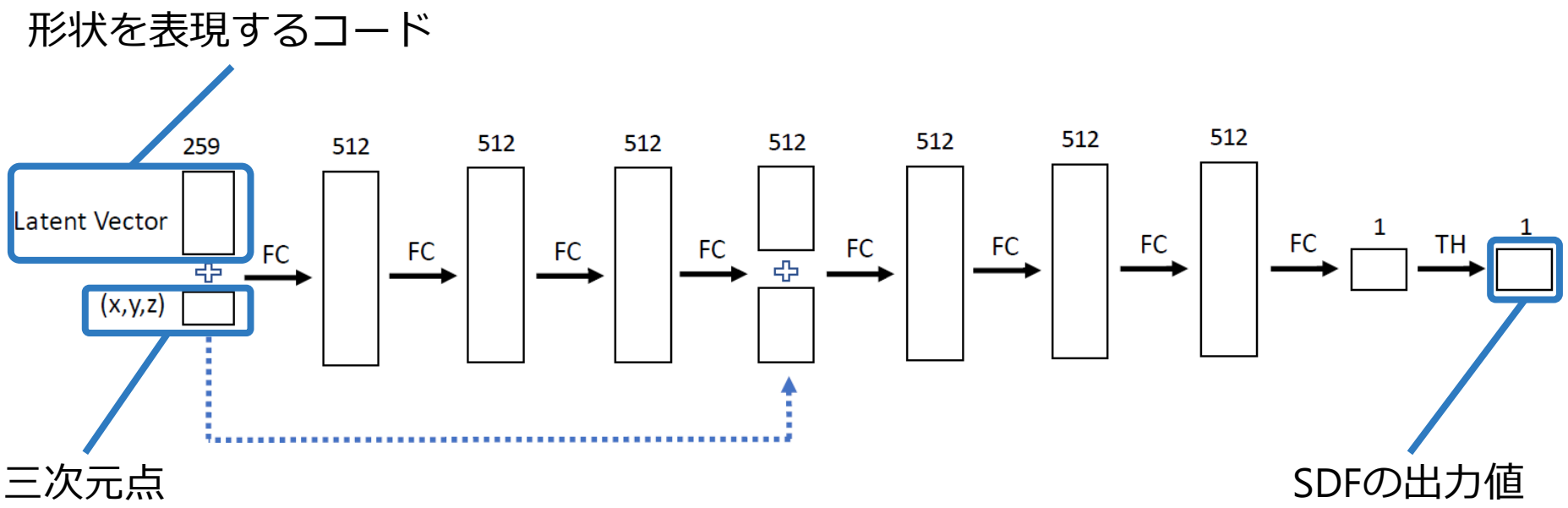


複数形状の場合

ネットワーク構造

ネットワークの構成

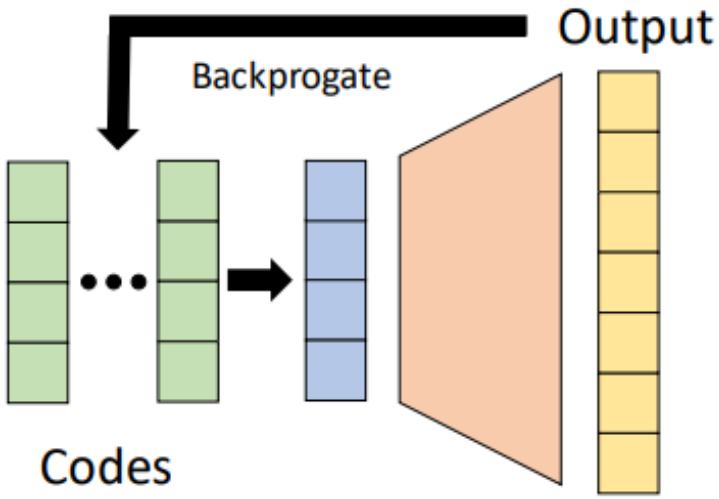
- SDF自体を学習するため，ネットワークの構造はシンプル
- 基本的には単純なMLP



Codeの決め方

オートデコーダー

- 形状ごとにCodeを設定
- デコーダーを同時にCodeも最適化
- 初期値はランダムに与える
- 推論時は与えられたサンプル点の集合から Codeについて最適化，SDFを得る



J. J. Park+. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. CVPR2019.

学習方法

ロス関数

L1ロス with clamp，SDFの値を$[-\delta, \delta]$の範囲に限定

$$\mathcal{L} = |\mathrm{clamp}(f_\theta(x), \delta) - \mathrm{clamp}(s, \delta)|$$

- $\mathrm{clamp}(\cdot, \delta)$: 値を$[-\delta, \delta]$に限定
- s: 教師信号

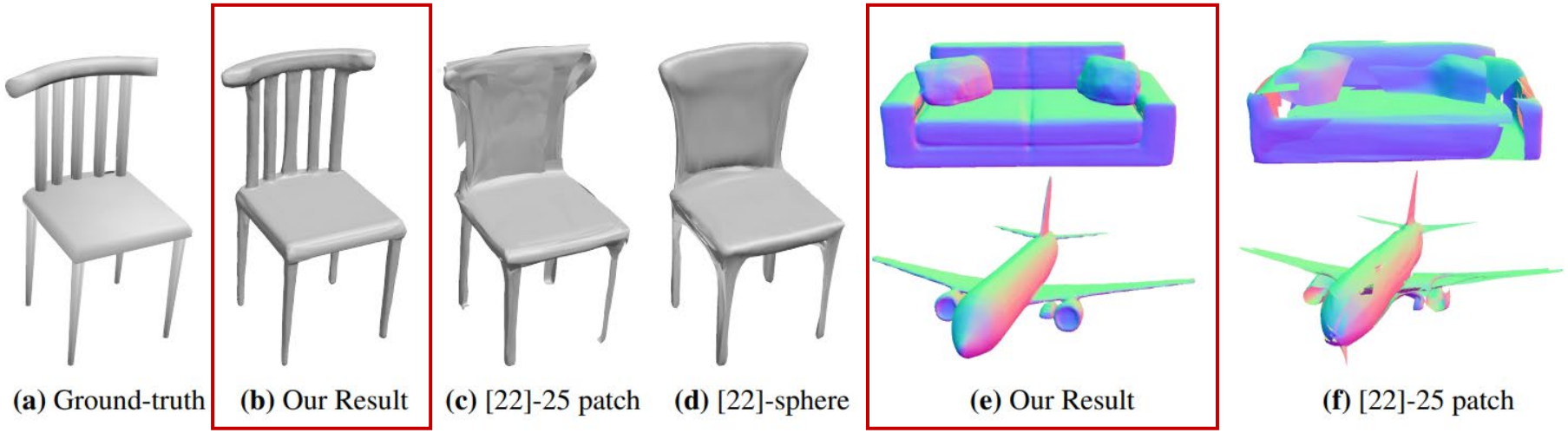
学習する点のサンプリング

- 一つの形状についていくつものサンプル点を与えて学習
- 表面付近を重点的にサンプリング＆
  空間全体でサンプリング

出力の例

既存手法との比較

良好な再構成結果が得られている

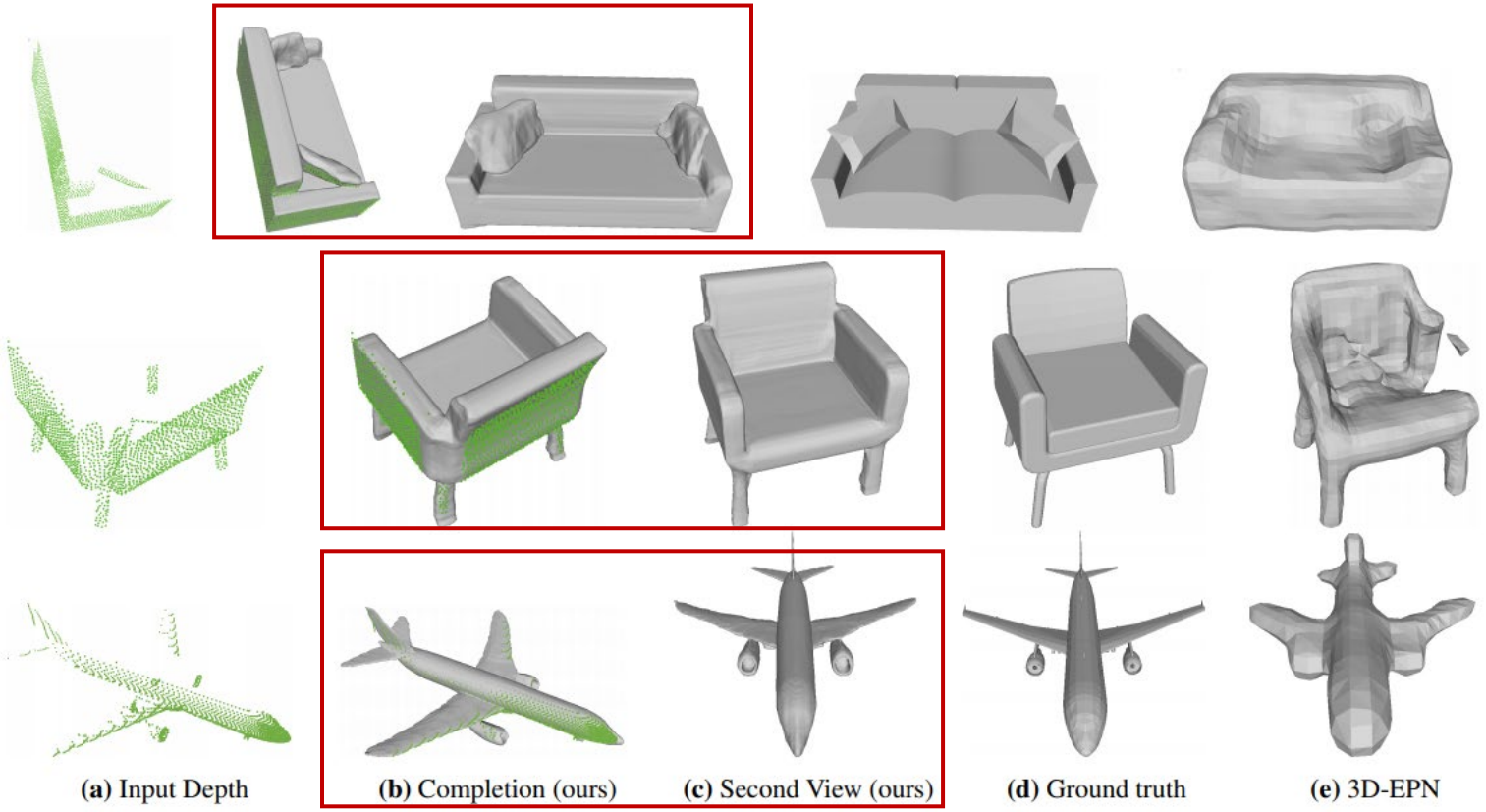


J. J. Park+. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. CVPR2019.

出力の例

片面点群から全体形状の補完

見えていない部分も再現している

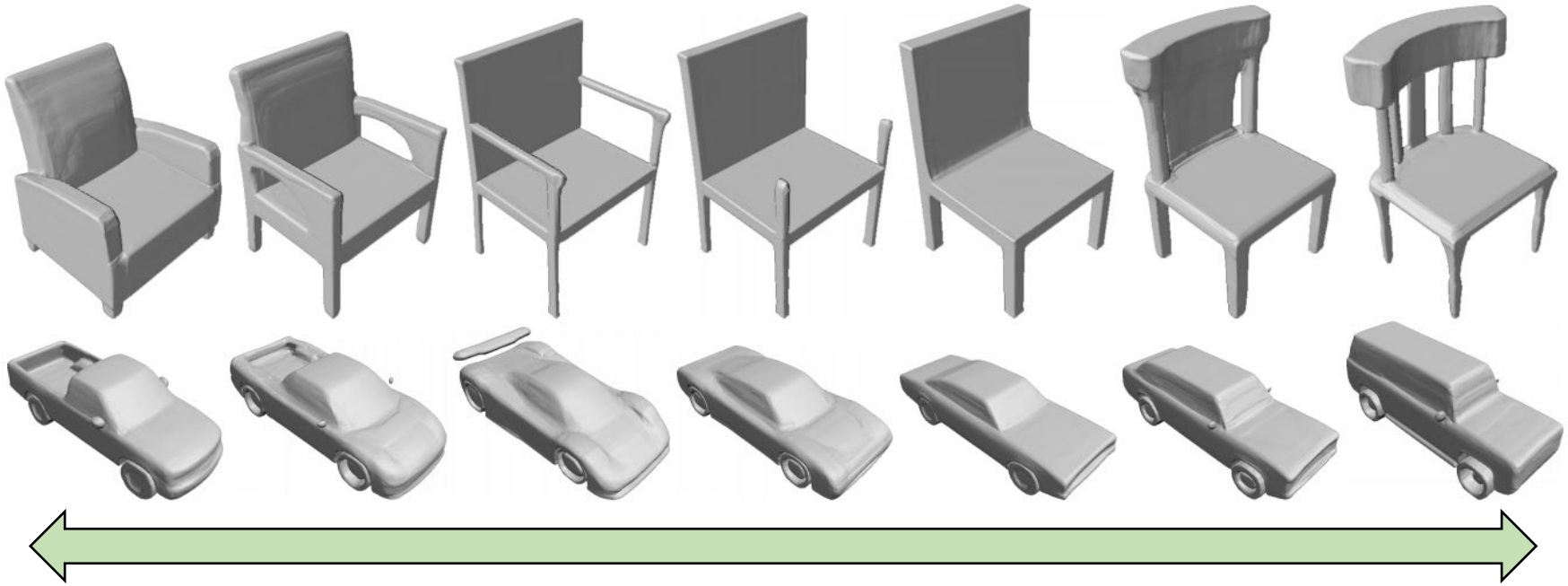


J. J. Park+. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. CVPR2019.

出力の例

形状補間の例

なめらかに補間できている

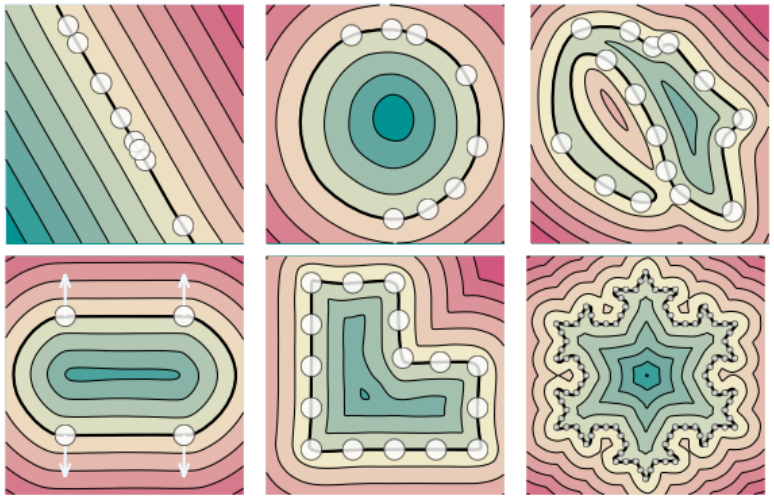


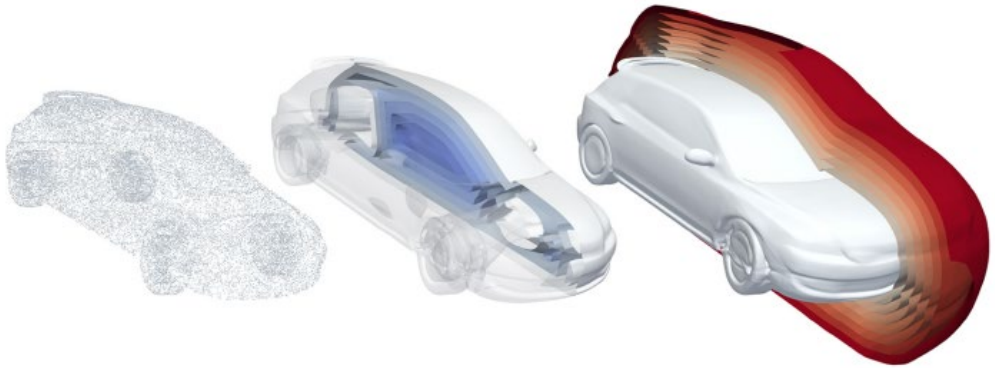
J. J. Park+. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. CVPR2019.

表面点群のみからSDFを学習

Implicit Geometric Regularization [A.Gropp+, ICML2020]

- 表面点群（＝Implicit Functionのゼロ面上の点）からImplicit FunctionによりSDFを学習
- 適切な初期化と空間中での勾配に関する制約（SDFの勾配があらゆる点で1になる）を用いて学習





A. Gropp+. Implicit Geometric Regularization for Learning Shapes. ICML2020.

他の表面モデル

Occupancy Network

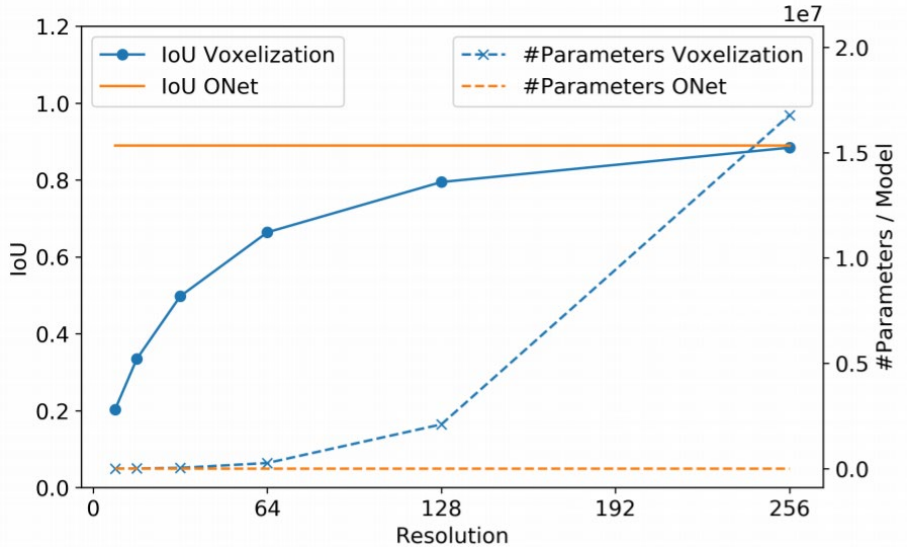
- SDFではなくその点での占有率でモデル化
- 占有しているかどうかの二値分類と考え，Cross-entropy Lossで学習
- エンコーダーと組み合わせ，画像からの3D再構成や低解像度のボクセル表現からの超解像も実現

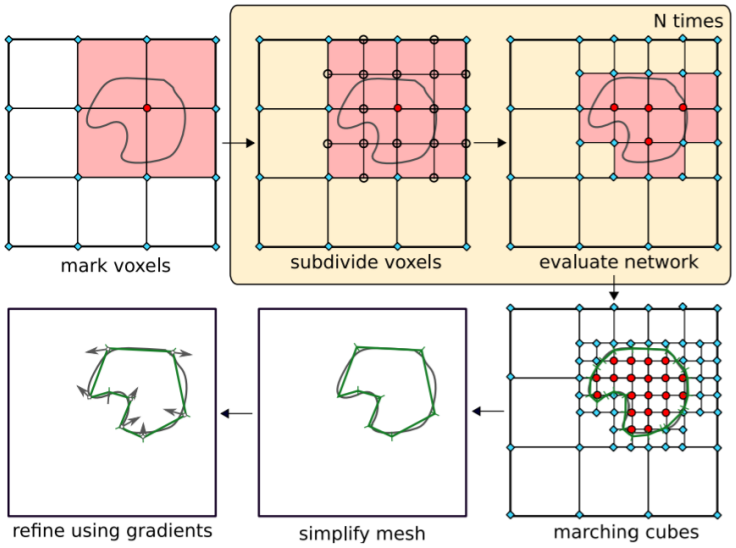
L. Mescheder+. Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR2019

他の表面モデル

Occupancy Network

- Neural Fieldの特徴として，解像度によらずパラメータ数が一定であることを強調
- メッシュとして表面形状をきれいに出力する手法Multiresolution IsoSurface Extractionも提案



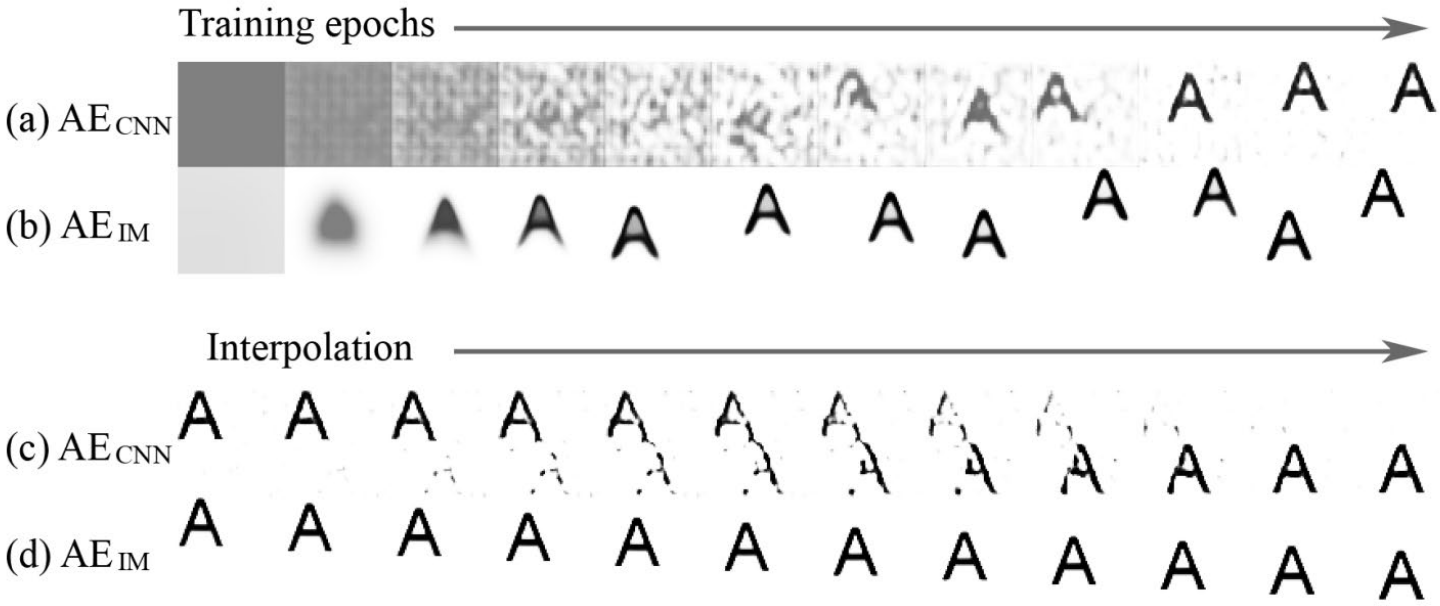


L. Mescheder+. Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR2019

他の表面モデル

IM-Net

- 占有率でモデル化，（重み付き）L2 Lossで学習
- 生成モデルとして使うことに重点
- 学習の過程や潜在ベクトル空間での補間の性質が良い



Z. Chen+. Learning Implicit Fields for Generative Shape Modeling, CVPR2019

表面モデルから体積モデルへ

Neural Fieldのアイデアが3D表現を革新
シンプルなネットワークで複雑な形状を
モデル化できるようになった

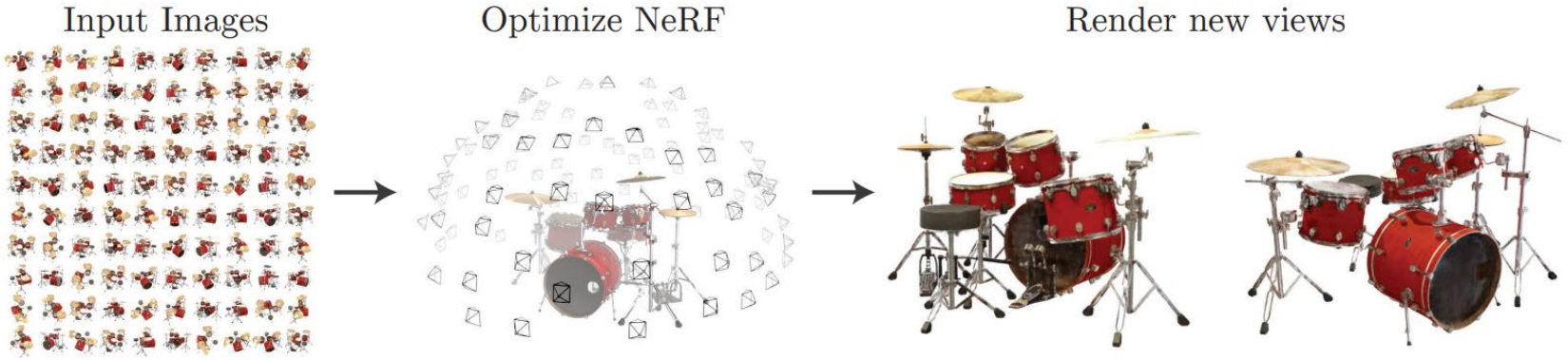
表面モデルのlimitation
- 半透明・構造色を含むシーンには対応していない
- 3Dの教師データが必要

→ **NeRF** (Neural Radiance Field) **の登場**
- 体積（Volumetric）ベースのモデル
- 多視点画像のみから任意視点画像生成を可能に
- 視点依存性にも対応
- （オリジナルのNeRFでは）シーンごとに最適化，DeepSDFのようにCodeで制御しない

NeRFの紹介

NeRF (Neural Radiance Field) [Ben Mildenhall+, ECCV2020]

- 新規視点画像生成（Novel View Synthesis）を行う
  あるシーンの**多視点画像を学習**しておき，
  そのシーンの**任意視点での画像を推定**できるように
- アプローチ
  - Radiance Field （放射輝度場）をNNでモデル化
  - Volume Renderingを深層学習フレームワーク上で計算，シーンごとに勾配ベースで最適化



Radiance FieldとVolume Rendering

Radiance Field

＝ある点・ある視点についての放射輝度

NeRFでは色と密度でモデル化

色：$c(\mathbf{r}(t), \mathbf{d})$ … 色については座標と見る角度$\mathbf{d}$に依存

密度： $\sigma\big(\mathbf{r}(t)\big)$ … 密度は座標にのみ依存

Radiance Fieldについて Volume Renderingすると

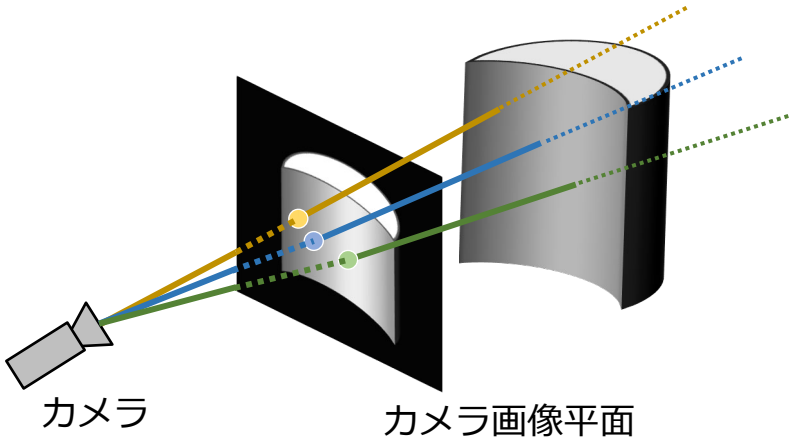
- 見る角度で色が変わるシーンに対応
- 霧・煙などにも対応

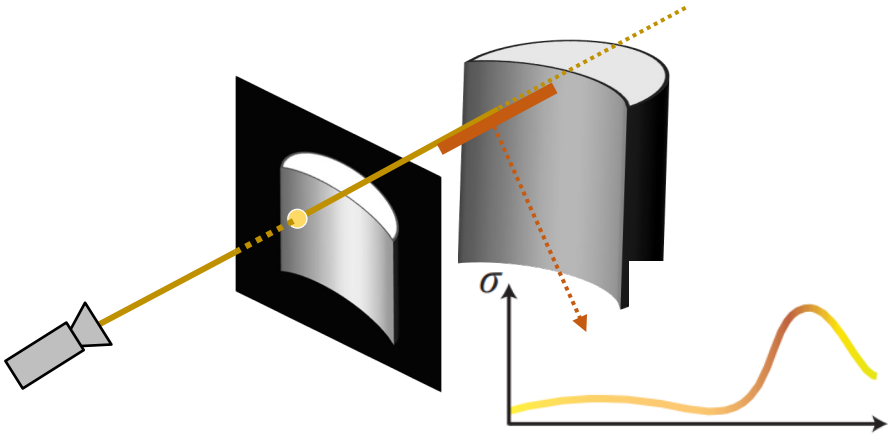
できる

Volume Rendering

画像＝光線に対応した**画素の輝度値を並べた**もの

画素の輝度値
＝光線上の**放射輝度を足し合わせた**もの





Volume Rendering

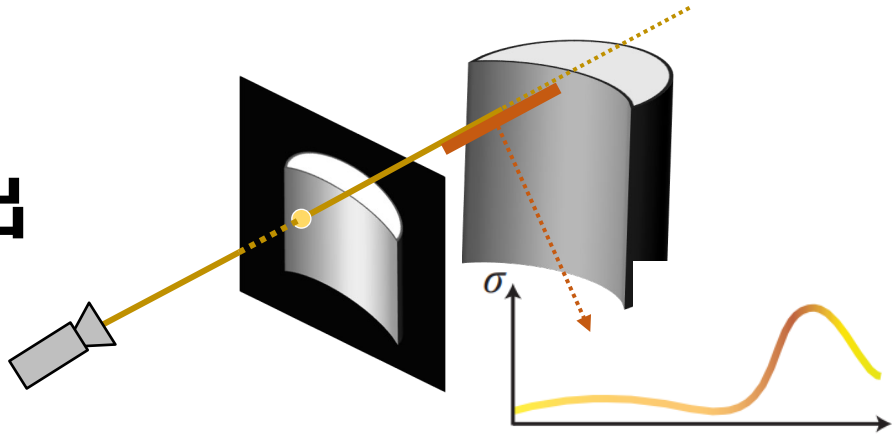
画像＝光線に対応した**画素の輝度値を並べた**もの

画素の輝度値
＝光線上の**放射輝度を足し合わせた**もの

放射輝度（Radiance）の
考え方・・・

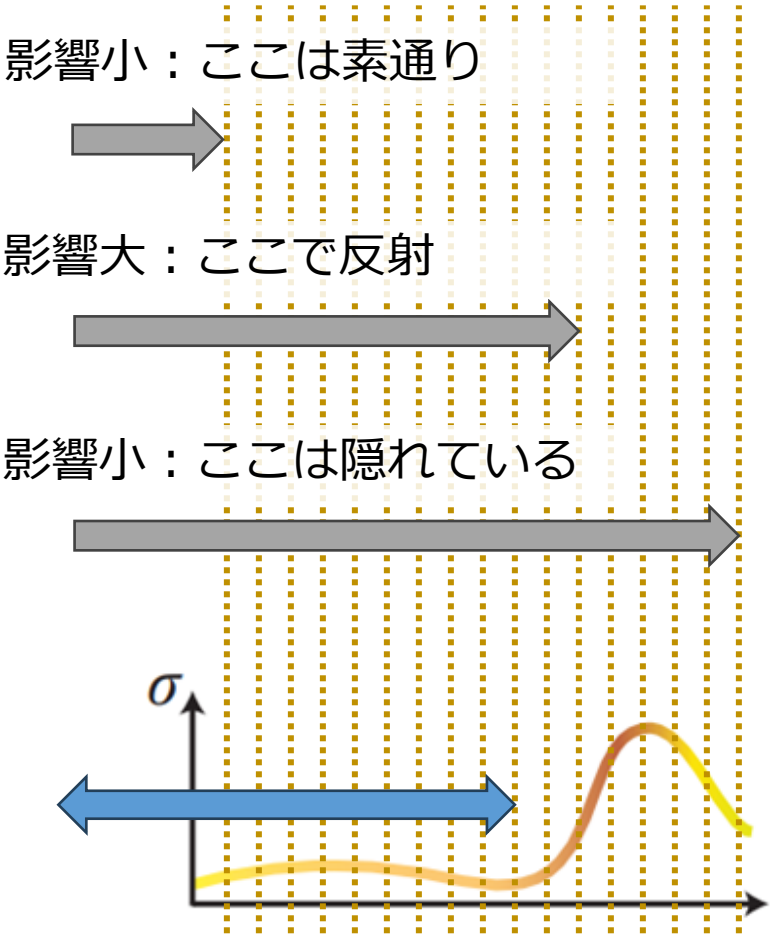
空間中の各点から光が放出

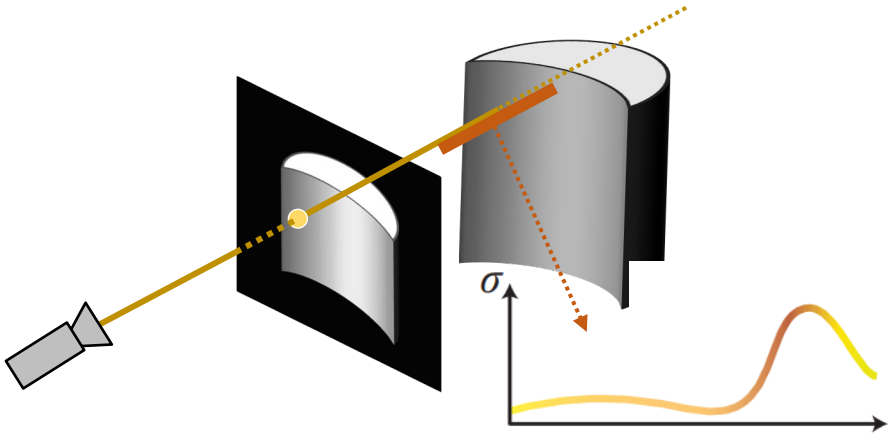
- 方向には依存
- ライティングは固定



Volume Rendering

光線上の**放射輝度を足し合わせた**もの





$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

ある点までの密度の積分：
どれくらい既に隠れているか

Volume Rendering

画像＝光線に対応した**画素の輝度値を並べた**もの

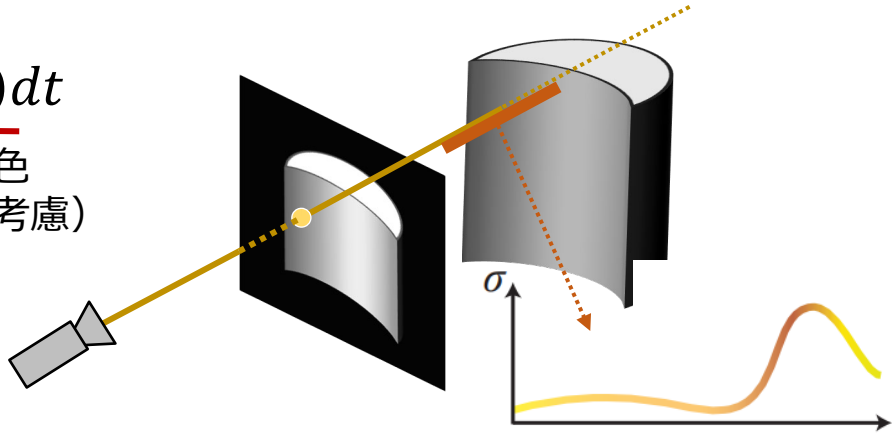
画素の輝度値
＝光線上の**放射輝度を足し合わせた**もの

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\mathbf{r}(t)\big)c(\mathbf{r}(t), \mathbf{d})dt$$

$C(\mathbf{r})$：画素の色
$T(t)$：
$\sigma\big(\mathbf{r}(t)\big)$：その点の密度
$c(\mathbf{r}(t), \mathbf{d})$：その点の色（方向を考慮）

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma\big(\mathbf{r}(s)\big)ds\right)$$

$\sigma\big(\mathbf{r}(s)\big)$：そこまでの密度



これを深層学習フレームワークで実装し
微分可能なボリュームレンダリングを実現，
NeRFの最適化が可能になった

Volume Rendering

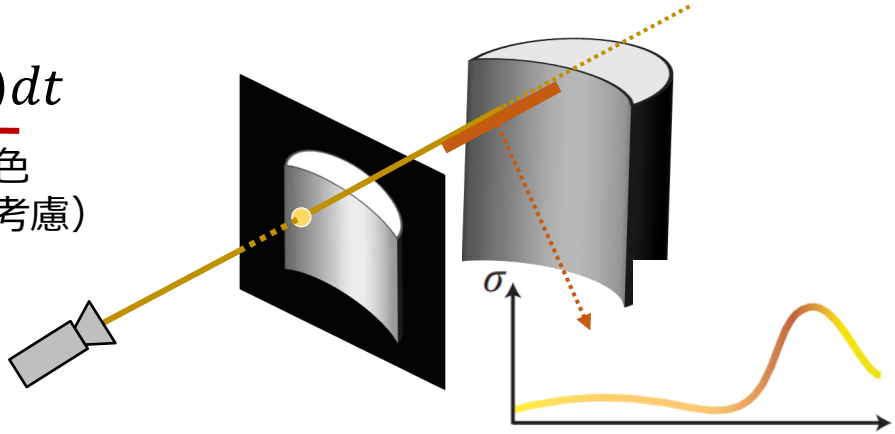
光線上の**放射輝度を足し合わせた**もの

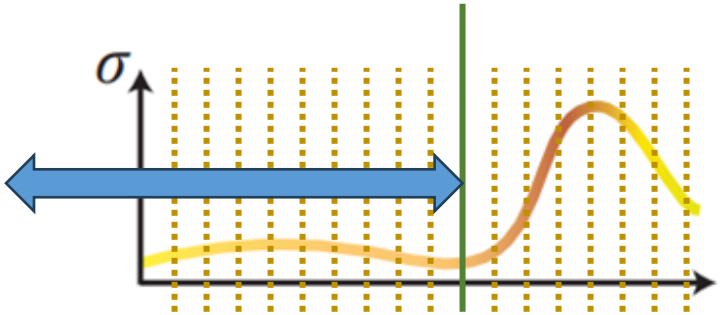
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\mathbf{r}(t)\big)c(\mathbf{r}(t), \mathbf{d})dt$$

画素の色　その点の密度　その点の色（方向を考慮）

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma\big(\mathbf{r}(s)\big)ds\right)$$

そこまでの密度





その部分まで見通せる＆
その点で密度が高い
→ その点での放射輝度を採用

密度小さい → 影響小

Volume Rendering

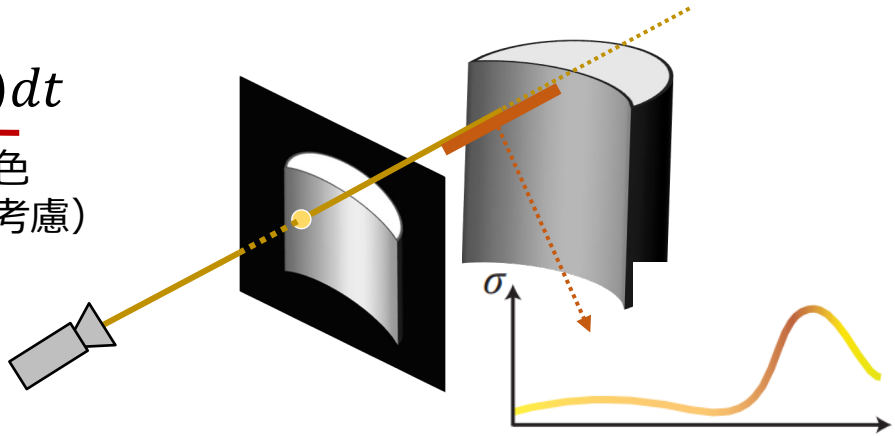
光線上の**放射輝度を足し合わせた**もの

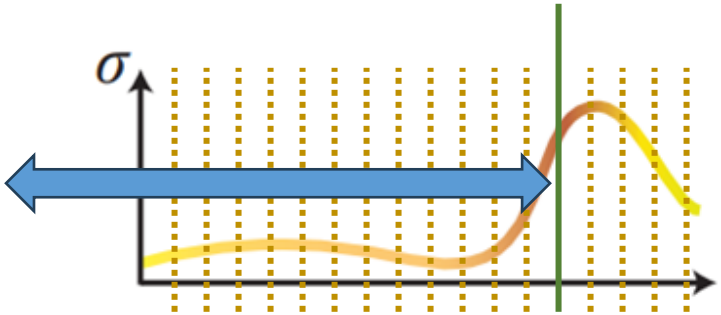
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\mathbf{r}(t)\big)c(\mathbf{r}(t), \mathbf{d})dt$$

画素の色　その点の密度　その点の色（方向を考慮）

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma\big(\mathbf{r}(s)\big)ds\right)$$

そこまでの密度





その部分まで見通せる＆
その点で密度が高い
→ その点での放射輝度を採用

見通せる＆密度大 → 影響大

Volume Rendering

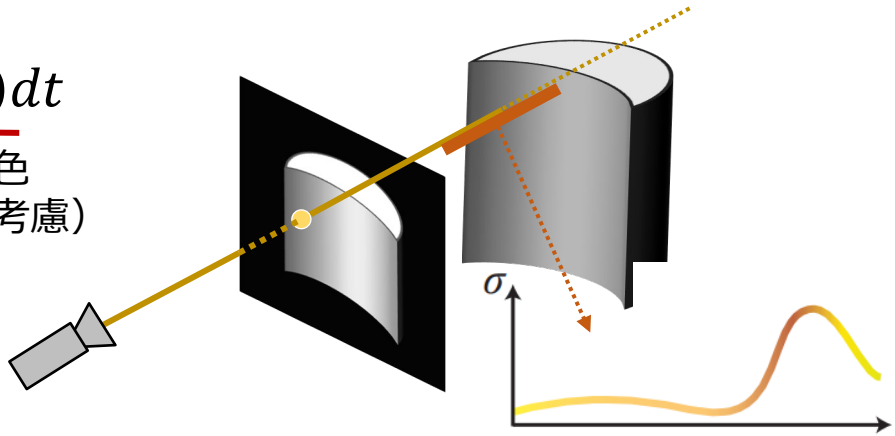
光線上の**放射輝度を足し合わせた**もの

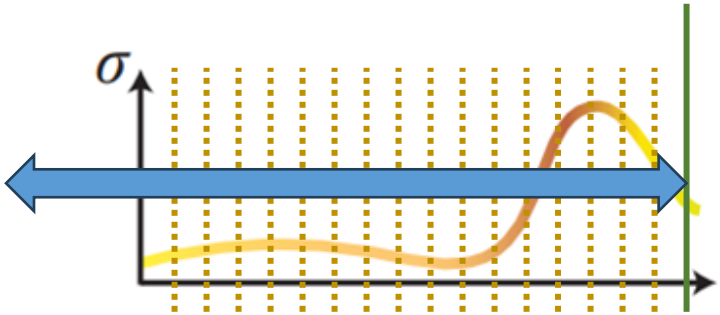
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\mathbf{r}(t)\big)c(\mathbf{r}(t), \mathbf{d})dt$$

画素の色　その点の密度　その点の色（方向を考慮）

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma\big(\mathbf{r}(s)\big)ds\right)$$

そこまでの密度





その部分まで見通せる＆
その点で密度が高い
→ その点での放射輝度を採用

見通せない → $T(t)$が小さい → 影響小

余談：レンダリング方程式

レンダリング方程式

$$L_o(x, \overrightarrow{\omega_o}) = L_e(x, \overrightarrow{\omega_o}) + \int_{S^2} f(x, \overrightarrow{\omega_i}, \overrightarrow{\omega_o}) L_i(x, \overrightarrow{\omega_i}) |\overrightarrow{\omega_i} \cdot \vec{n}| \mathrm{d}\overrightarrow{\omega_i}$$

- L_o: ある点である方向に出てくる光
- L_e: その点でその方向への発光
- L_i: その点でその方向への出力光に関する，
  ある方向からの入力光
- x: 着目点の座標
- $\overrightarrow{\omega_o}$: 光の出る方向
- $\overrightarrow{\omega_i}$: 光の入る方向
- $\vec{n}$: 法線方向
- S^2: 球面全体について

余談：レンダリングでの定式化との関係

レンダリング方程式

$$L_o(x, \overrightarrow{\omega_o}) = L_e(x, \overrightarrow{\omega_o}) + \int_{S^2} f(x, \overrightarrow{\omega_i}, \overrightarrow{\omega_o}) L_i(x, \overrightarrow{\omega_i}) |\overrightarrow{\omega_i} \cdot \vec{n}| \mathrm{d}\overrightarrow{\omega_i}$$

NeRFでは単純化して

$$L_o(x, \overrightarrow{\omega_o}) = NN(x, \overrightarrow{\omega_o})$$

→ 入射光に関して放射輝度が変化しない
（＝ライティングは一定）と仮定

余談：レンダリングでの定式化との関係

ボリュームレンダリング方程式

（数式自体はやたらややこしいので省略）

表面での反射（レンダリング方程式）に加え，

- 光の吸収
- 光の散乱（in scattering / out scattering）
- 発光

の影響を考える.

これを光線にしたがって積分（NeRFでやっていること）すると，観測される輝度値になる

（さらに余談：この文脈で密度σは散乱係数と呼ばれる）

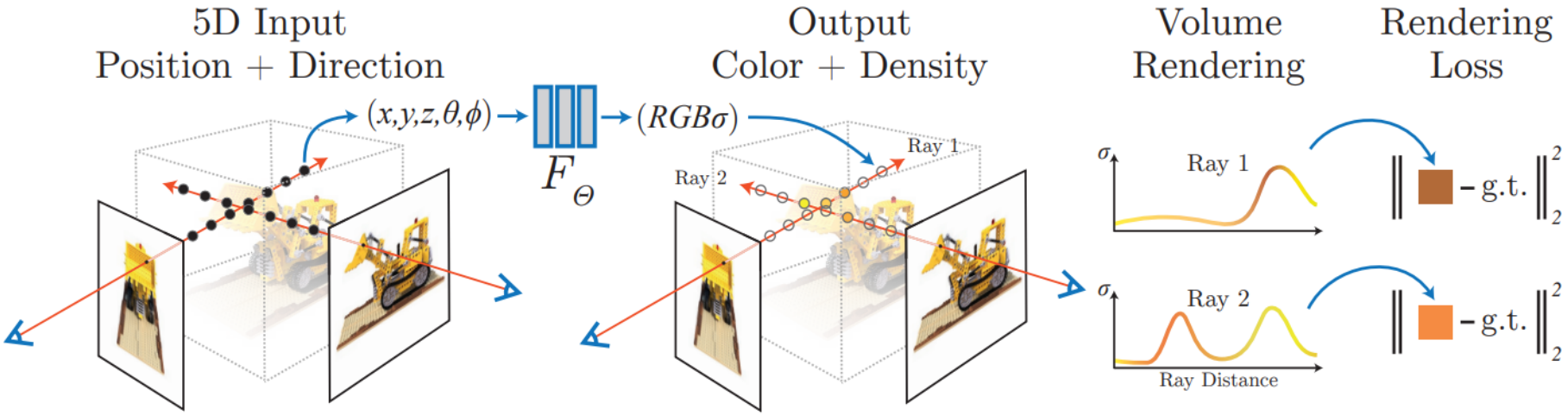
NeRFのモデル化の限界

- シーンについて
  - **単独のシーン**について最適化
  - **複数のシーン**は扱えない
  - **変形を含むシーン**（時系列など）は扱えない
- 光学現象について
  - 密度と放射輝度でモデル化
  - **方向による輝度値変化**（鏡面反射や構造光）は扱える
  - **散乱を含むシーン**（煙など）は扱える
    - ただし実際やってみるといずれもきれいには最適化できない，あくまでモデルとして可能であるの意味
  - **ライティングの変化**は扱えない

NeRFのモデルとロス関数

NNによるモデル
- 入力：**位置・光線の角度**
- 出力：**色・密度**

ロス関数：**輝度値を比較**（L2ロス）



B. Mildenhall+. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV2020.

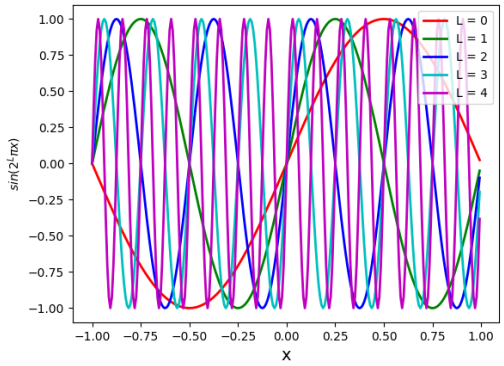
NeRFの工夫

Hierarchical Volume Sampling

粗いレンダリングでどこにあるかを推定してから，
密なレンダリングで重点的にサンプリング

Positional Encoding

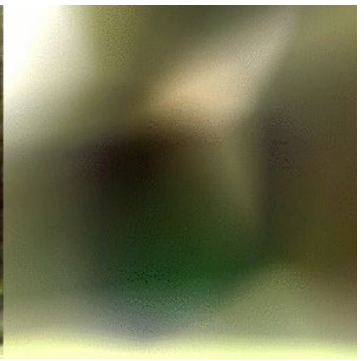
座標・光線方向を高周波・高次元な表現に変換してから
NNの入力とする



2DでのPositional Encodingの例

正解画像

Positional Encodingあり

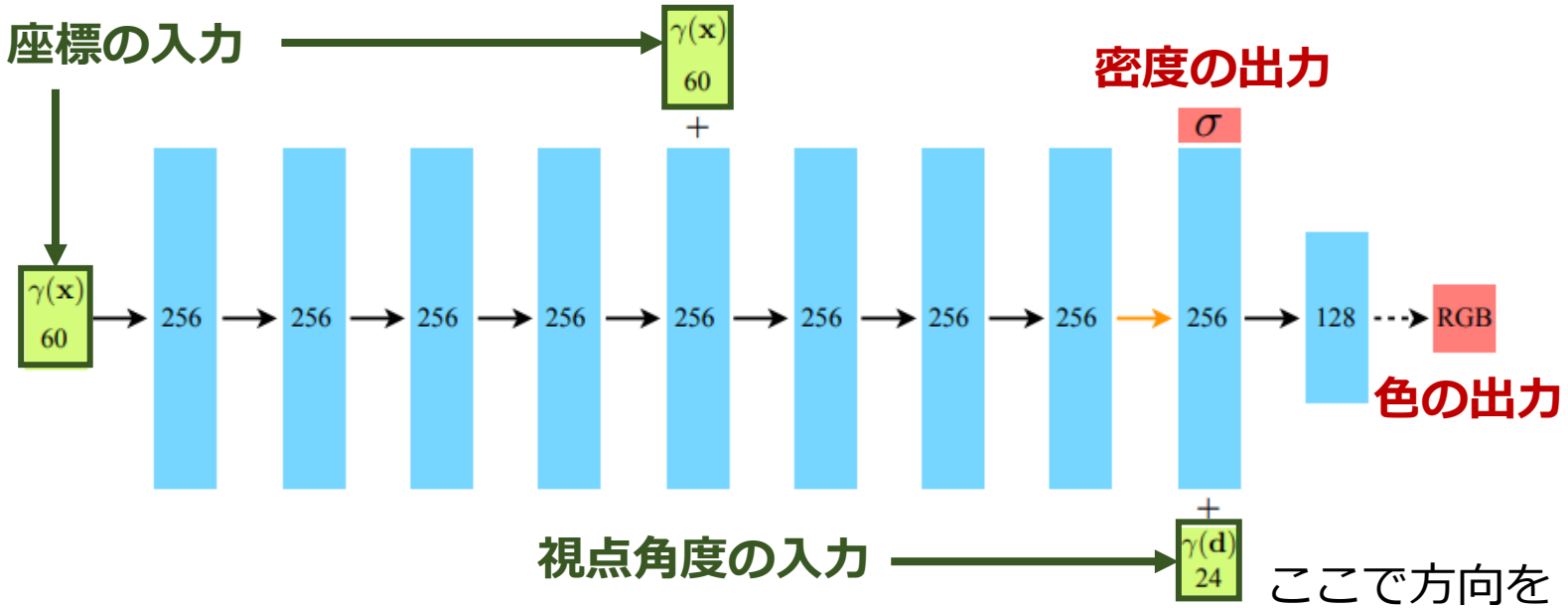


Positional Encodingなし

nerf2D: https://github.com/ankurhanda/nerf2D

ネットワーク構造

シンプルなMLPでRadiance Fieldをモデル化
光線上の各点についてクエリし
Volume Renderingする



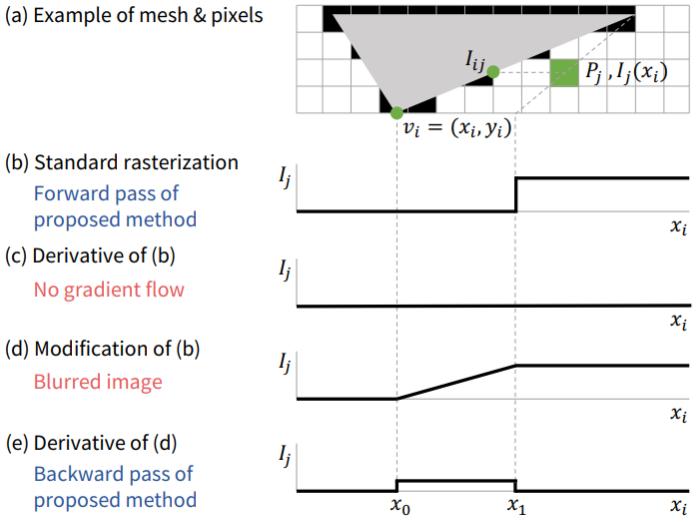
ここで方向を
入れるので，
色のみ視点依存

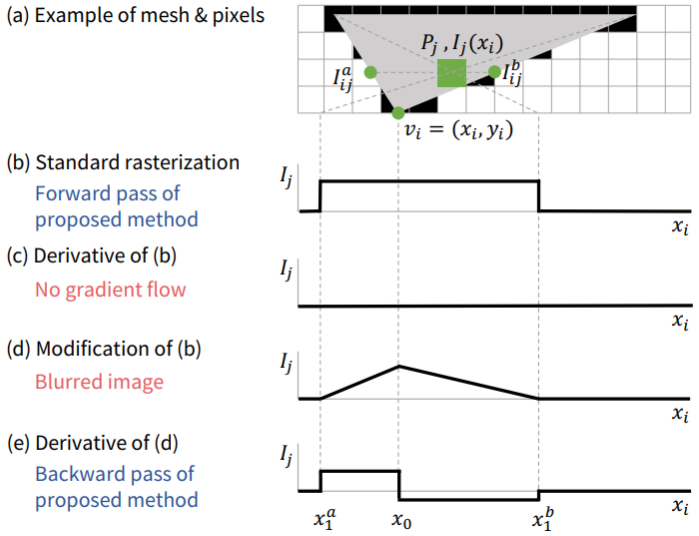
Ben Mildenhall+. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV2020.

別の歴史的背景

メッシュベースの微分可能レンダリング

- Neural 3D Mesh Renderer
- 画像上からパラメータの勾配を計算
  （勾配が通るようにBack Prop.を書き換え）
- メッシュの頂点座標やテクスチャを最適化できるように

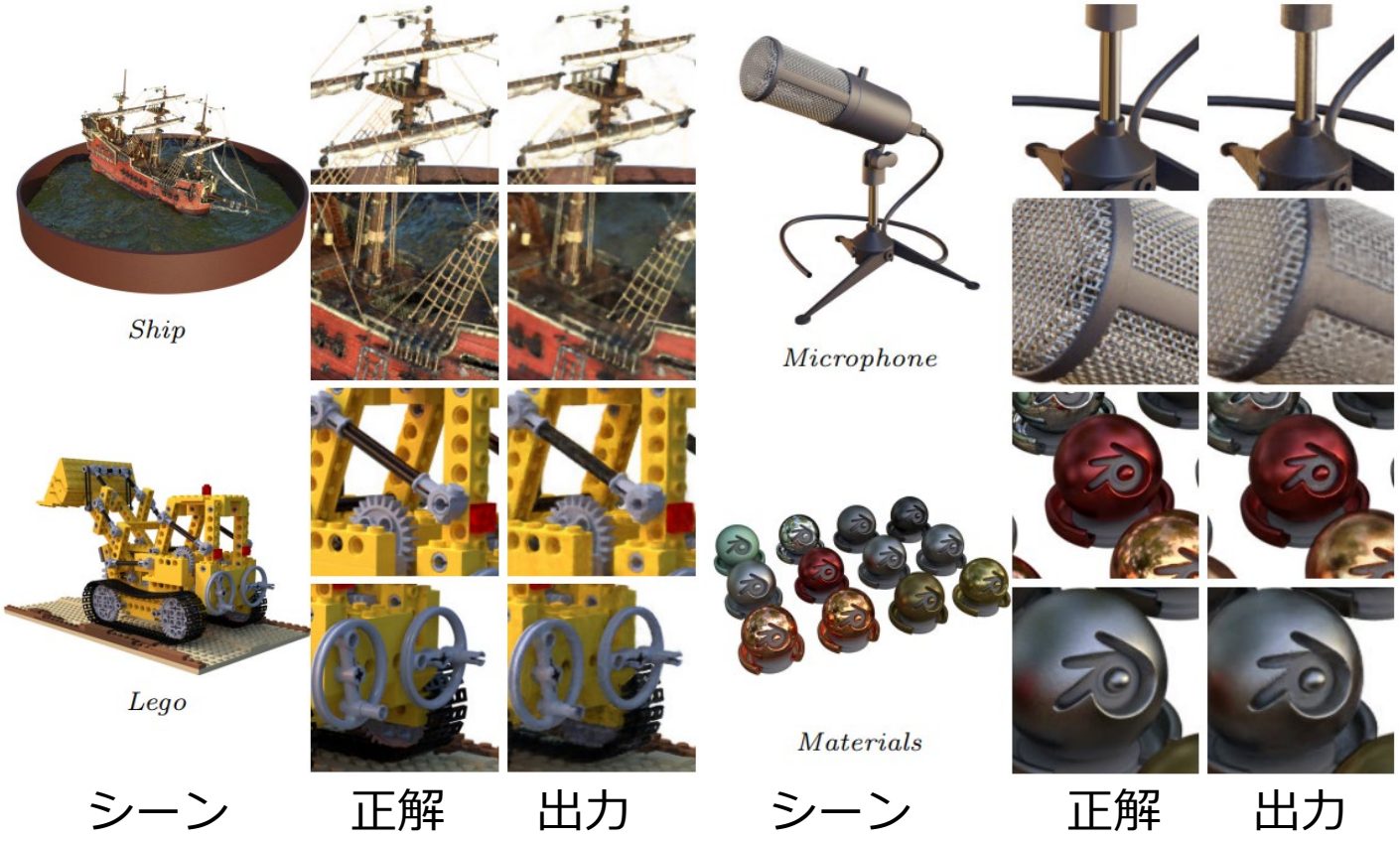




H. Kato+. Neural 3D Mesh Renderer. CVPR2018.

出力の例

細かい箇所まで良好な画像を出力



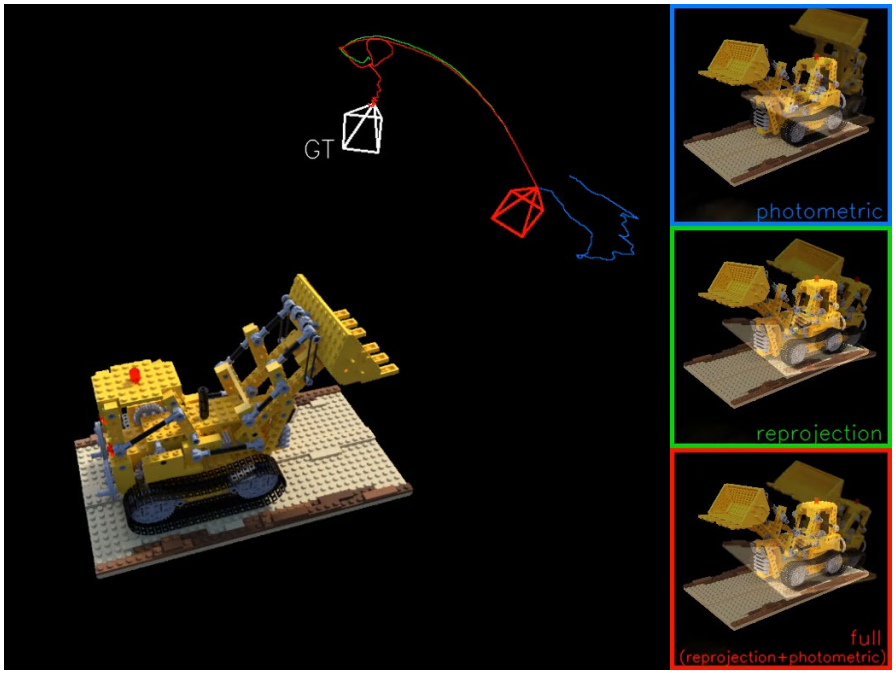
Ben Mildenhall+. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV2020.

カメラ姿勢の最適化

NeRFは微分可能レンダリング

Radiance Fieldだけでなく，ほかのパラメータに微分を伝える（→勾配法で最適化する）ことが可能に

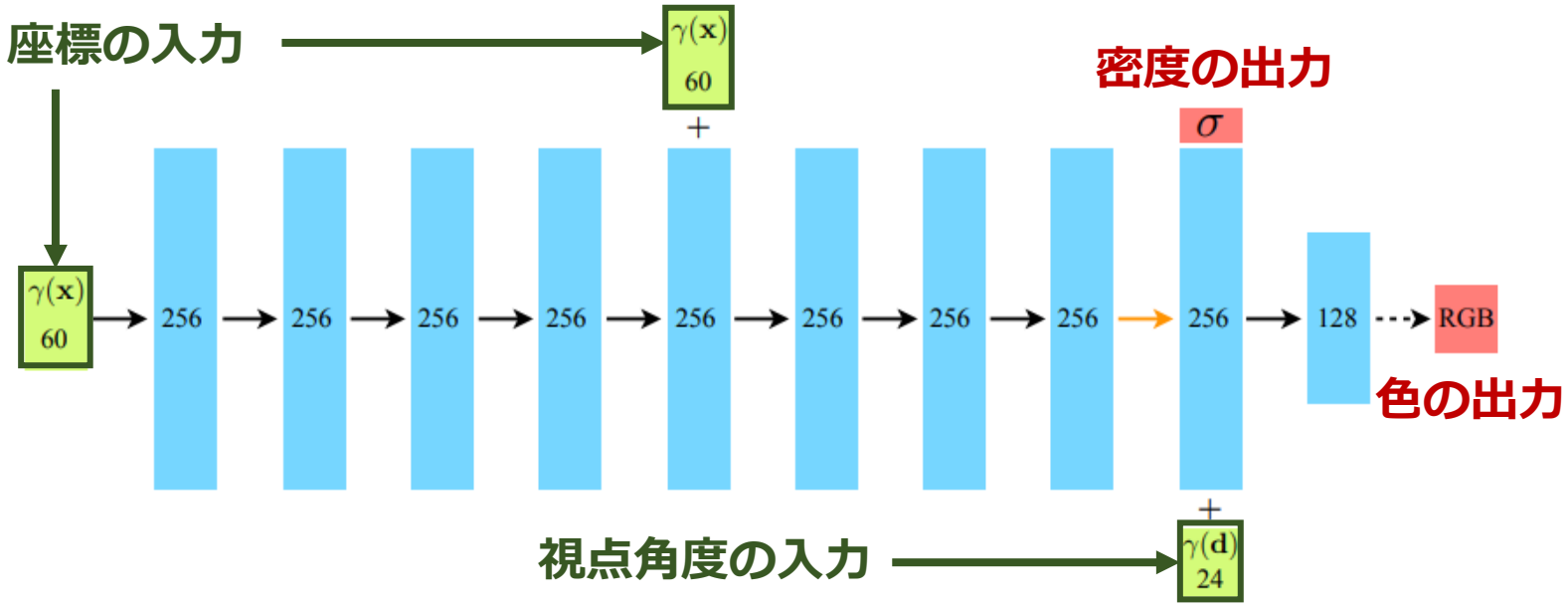
カメラ姿勢推定の例



I. Ueda+. Neural Density-Distance Fields. ECCV2022.

NeRFのネットワーク構造

シンプルなMLPでRadiance Fieldをモデル化
光線上の各点についてクエリし
Volume Renderingする

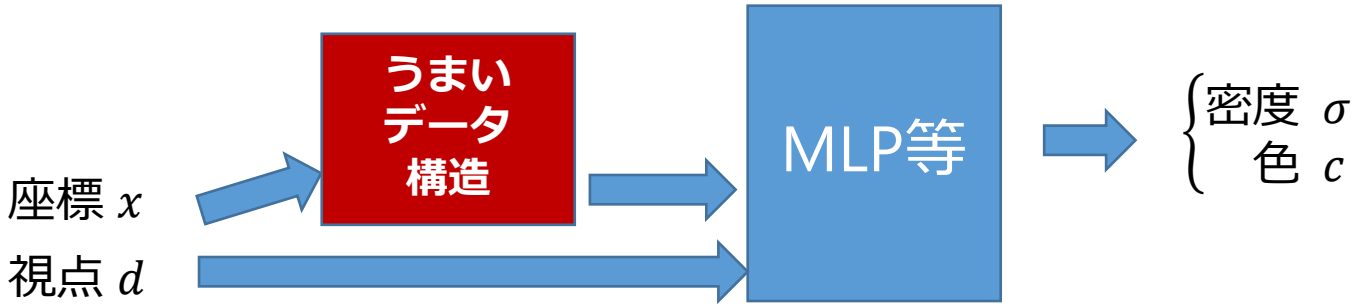


Ben Mildenhall+. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV2020.

NeRFを空間方向に分解する

一つのMLPでモデル化すると重い
→ 部分ごとに分割して情報を保持すれば良い

NeRFの模式図

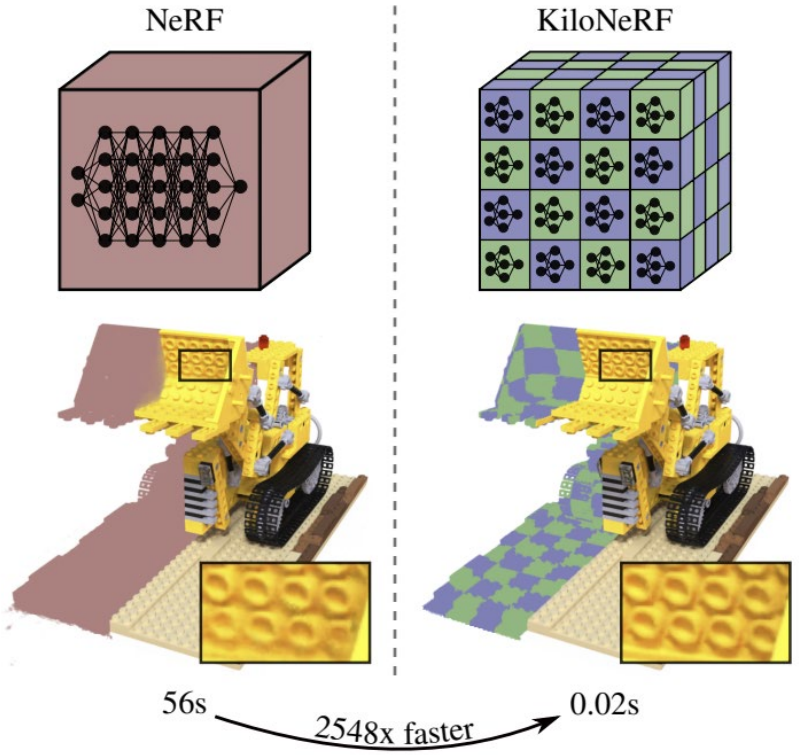


グリッドを用いたRFの模式図

小さいNeRFに分割

NeRFをそのまま**グリッドで部分ごとに分割**

- **kiloNeRF**, **FastNeRF**など
- 各座標でのRadiance Fieldの評価で，毎回全体のMLPを計算しない
- 座標に対応した軽量なMLPのみ評価することで大幅に高速化

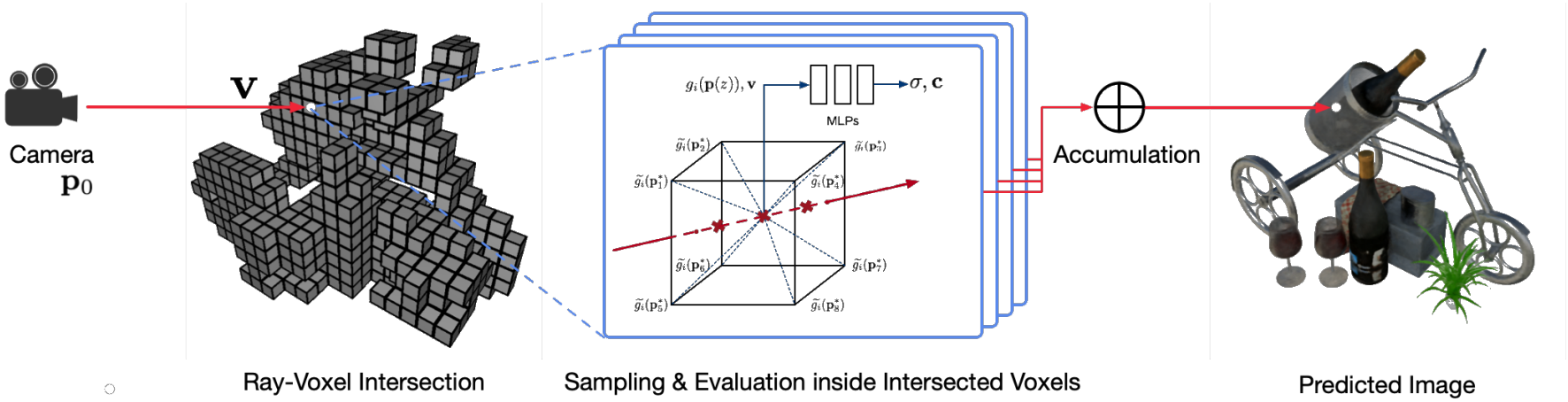


C. Reiser+. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. ICCV2021.

スパースボクセル

ボクセルに分割して**物体のあるところだけ特徴量を割り当てる**

- **NSVF**：物体がある領域に特徴量を割り当て，後段に軽量なMLPを通してRFをモデル化
- 段階的に解像度を上げて物体が存在する部分だけ保持



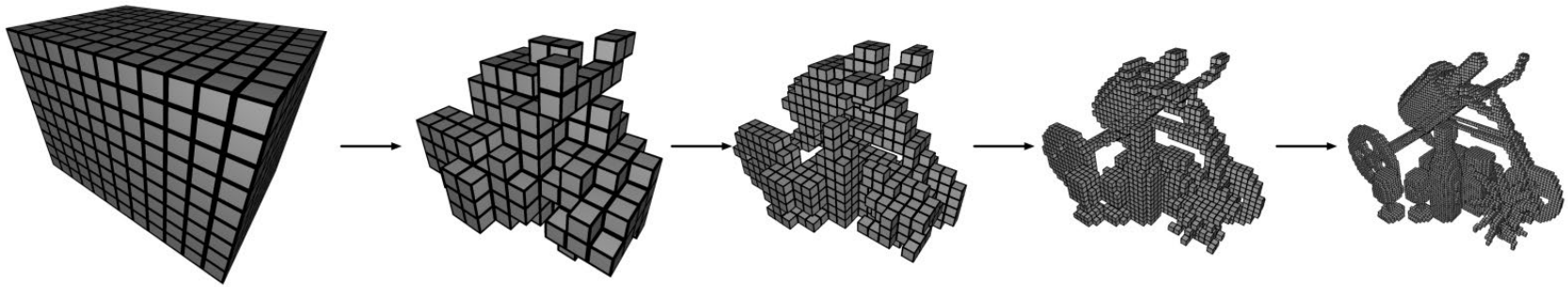
スパースボクセルの例（NSVF）

L. Liu+. Neural Sparse Voxel Fields. NeurIPS2020.

スパースボクセル

ボクセルに分割して**物体のあるところだけ特徴量を割り当てる**

- **NSVF**：物体がある領域に特徴量を割り当て，後段に軽量なMLPを通してRFをモデル化
- 段階的に解像度を上げて物体が存在する部分だけ保持



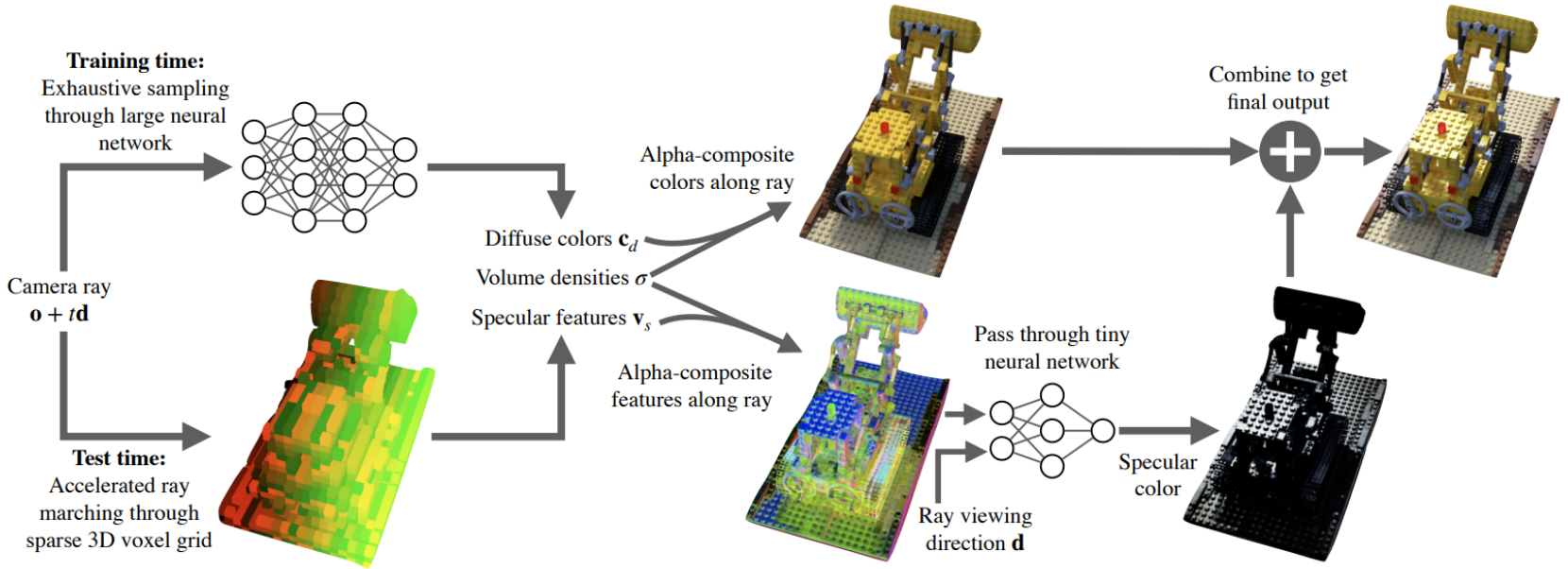
スパースボクセルの例（NSVF）

L. Liu+. Neural Sparse Voxel Fields. NeurIPS2020.

スパースボクセル

ボクセルに分割して**物体のあるところだけ特徴量を割り当てる**

- **SNeRF**：学習済みのNeRFをスパースボクセル表現に変換してレンダリングを高速化

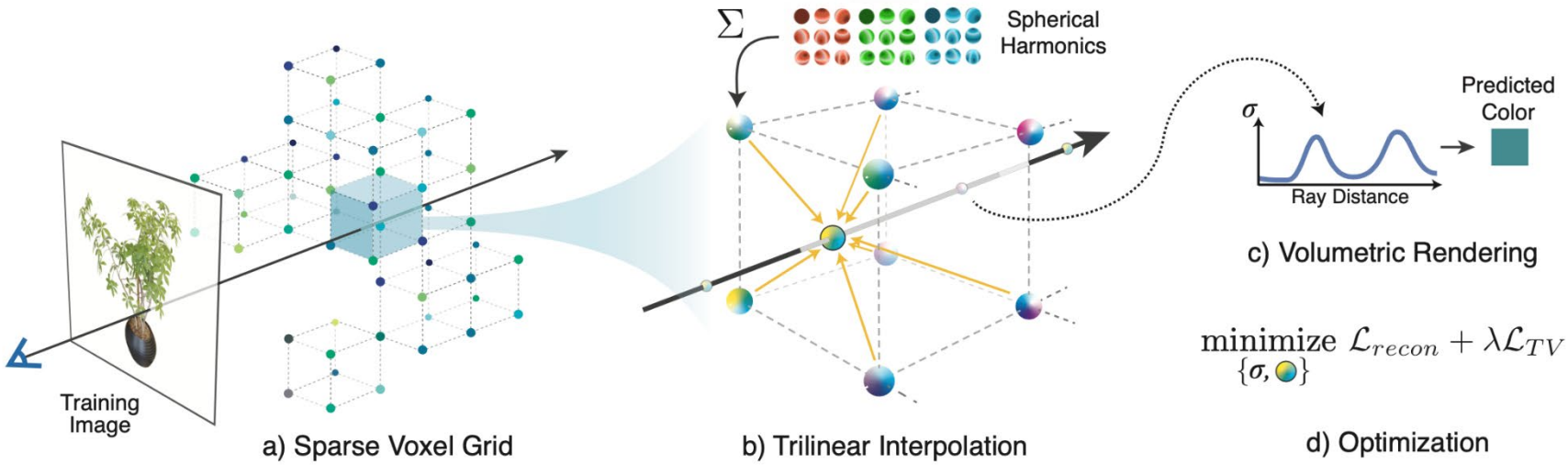


P. Hedman+. Baking Neural Radiance Fields for Real-Time View Synthesis. ICCV2021.

スパースボクセル

ボクセルに分割して**物体のあるところだけ特徴量を割り当てる**

- **Plenoxels**： Radiance Fieldをそのまま
  スパースボクセルとしてモデル化（MLPなし）
- カラーを球面調和関数でモデル化

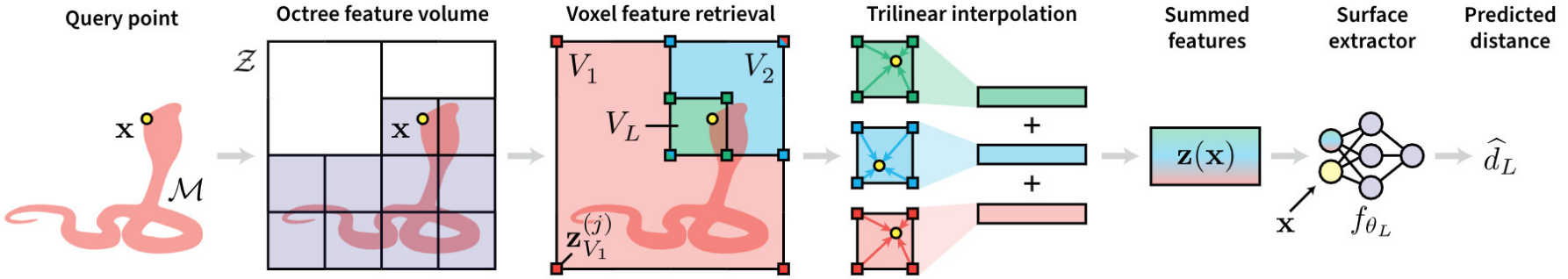


S. Fridovich-Keil & A. Yu+. Plenoxels: Radiance Fields without Neural Networks. CVPR2022.

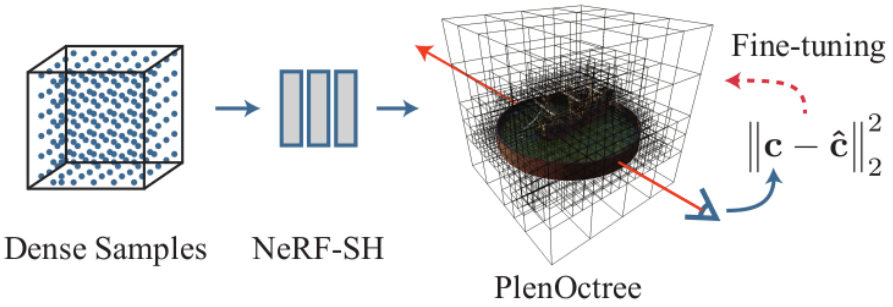
Octree

適応的に切り分けた空間分割で特徴量を
ボクセル上に配置する

- **NGLOD**：各解像度の特徴量の和を利用



- **PlenOctree**: NeRFをOctreeに変換，高速なレンダリング

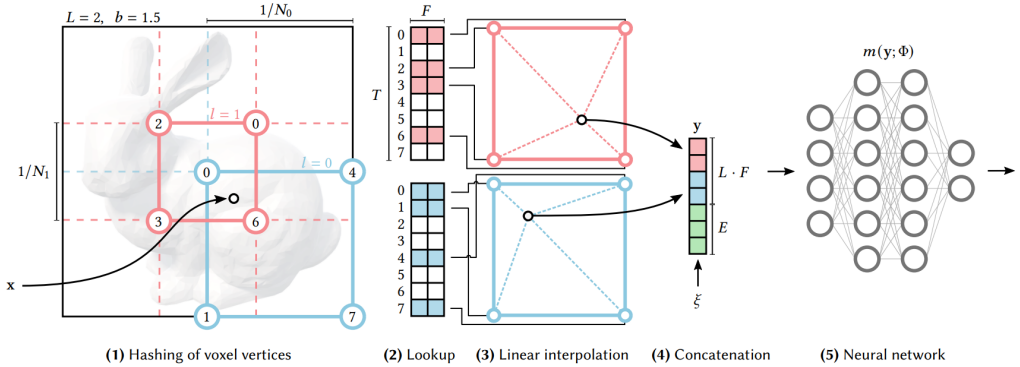


T. Takikawa+. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Surfaces. CVPR2021.

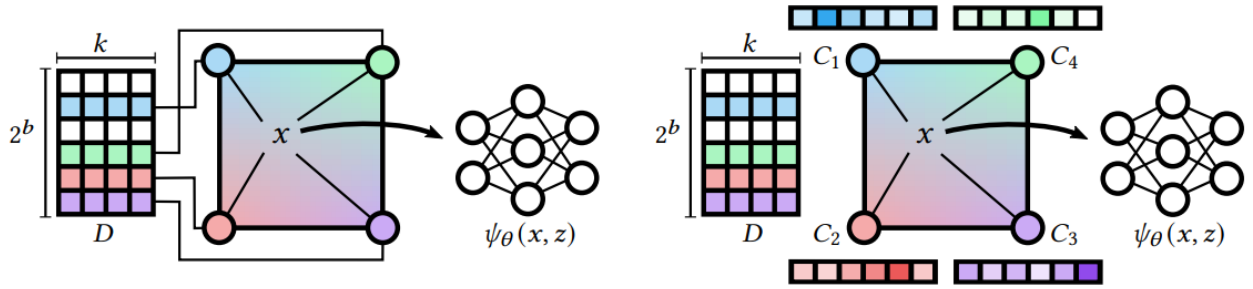
Hash / Codebook

ハッシュ / コードを用いてコンパクトな特徴量表現

- Instant-NGP Multiresolution Hash Encodingで
  ハッシュ化・特徴量をルックアップ



- VQAD: Softmaxで頂点のコードを学習可能にし最適化



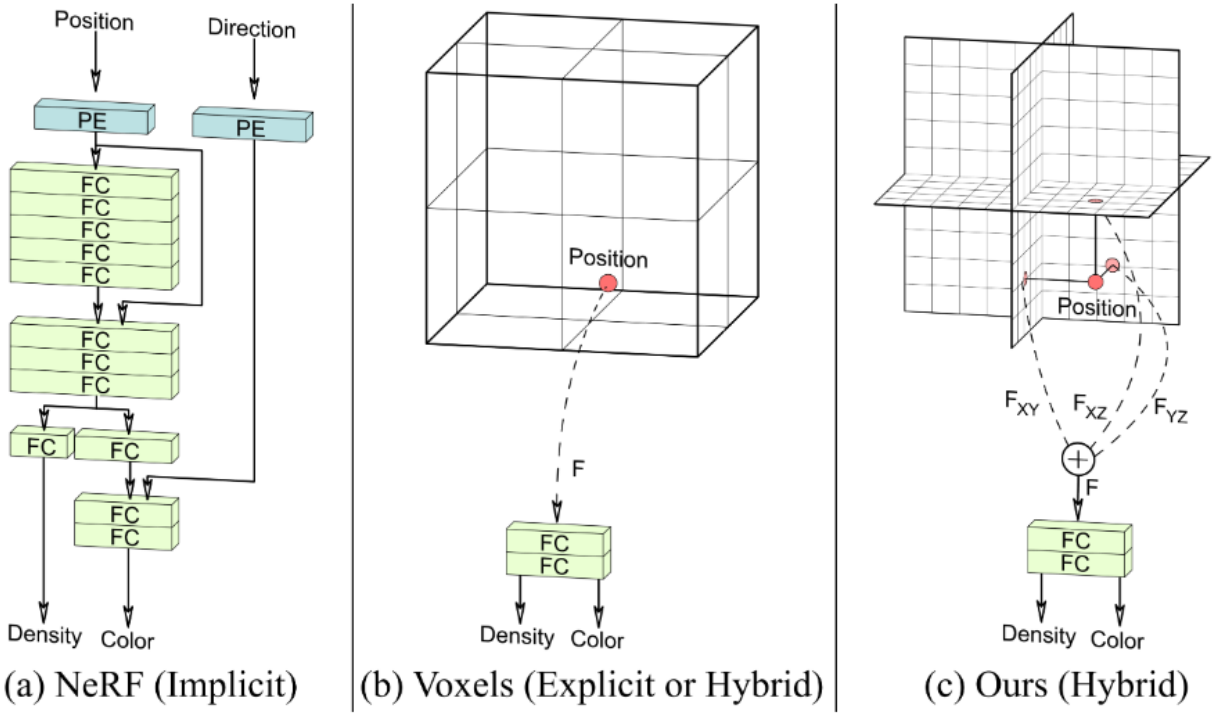
T. Müller+. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. SIGGRAPH2022.
T. Takikawa+. Variable Bitrate Neural Fields. SIGGRAPH2022.

低次元に分解

各面（など）に**分解して再構成**

- **Tri-plane**：各面の特徴量を足し合わせてその点での特徴量を記述，軽量なMLPでRFを出力

Hybrid Representationと命名



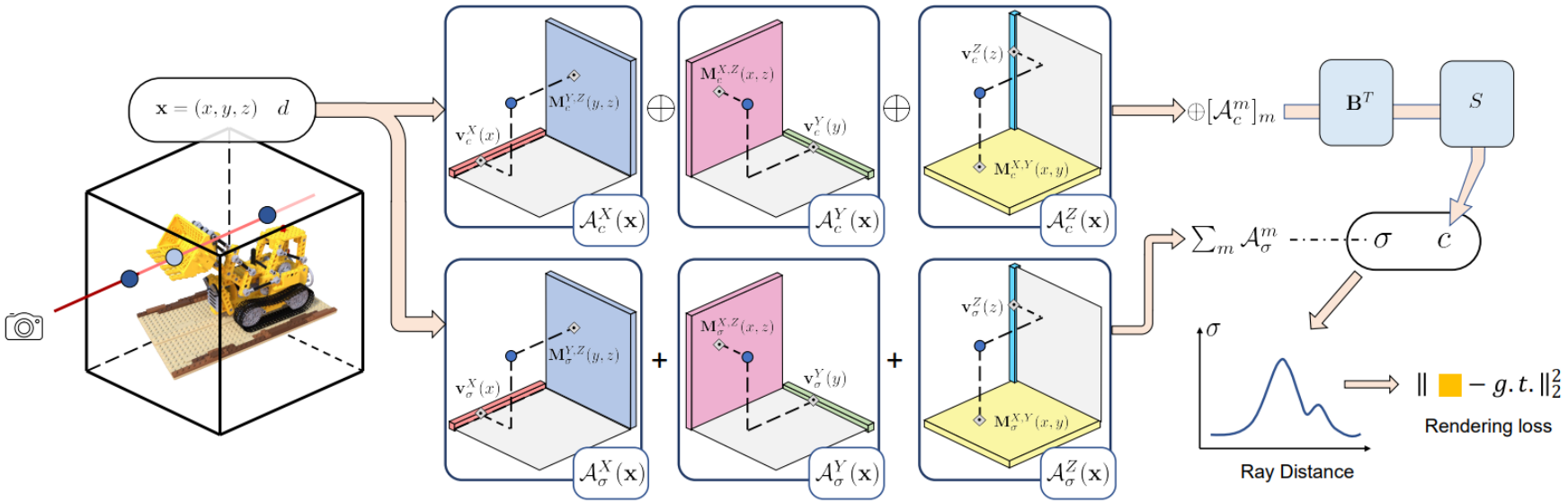
E. R. Chan+. Efficient Geometry-aware 3D Generative Adversarial Networks. CVPR2022.

低次元に分解

各面（など）に分解して再構成

- TensoRF: テンソル分解によって各点のRadiance Fieldをベクトル・行列積の和としてモデル化

  MLPを経由せず，線形変換でRFに変換



A. Chen+. TensoRF: Tensorial Radiance Fields. ECCV2022.

2Dでの応用例

Local Implicit Image Function (LIIF)

- 画像を2DのNeural Fieldで記述
  - 各座標値を入力，輝度値を返すようにモデル化
  - 着目範囲をセルとして与える工夫
- 細かくサンプリングすることで超解像を実現

Y. Chen+. Learning Continuous Image Representation with Local Implicit Image Function. CVPR2021.

Neural Fieldとも考えられる構造

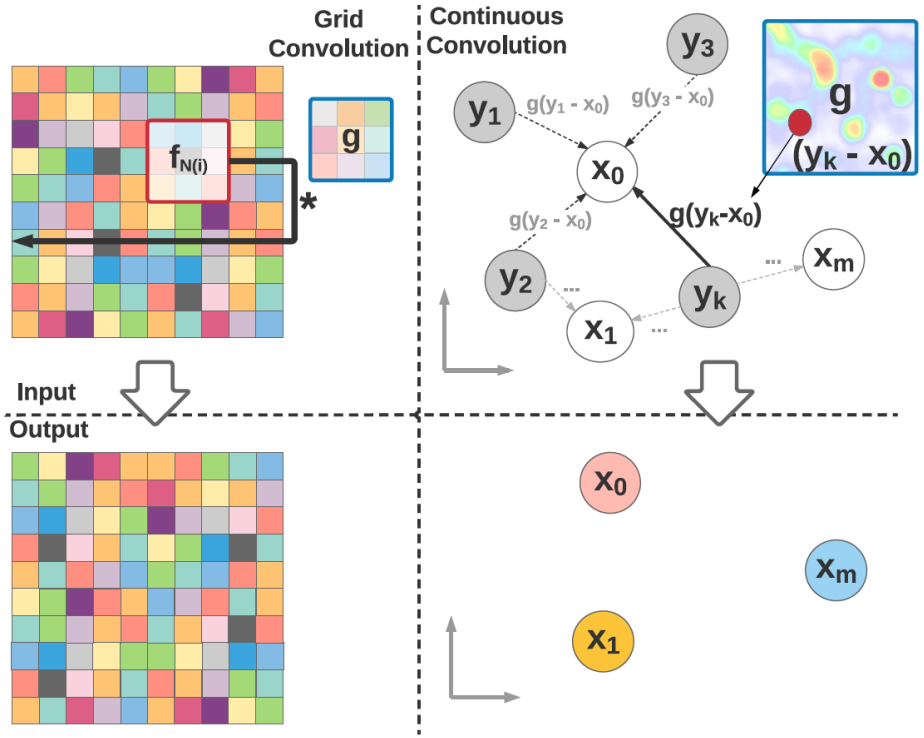
Parametric Continuous Convolutions

非グリッドな環境（点群を想定）での畳み込み

1. 局所領域を選択
2. 注目点との相対座標→その点での重みを出力

相対座標を入力して関数を返すことで連続な表現に

＝Neural Fieldと同じアイデア



Grid Convolution

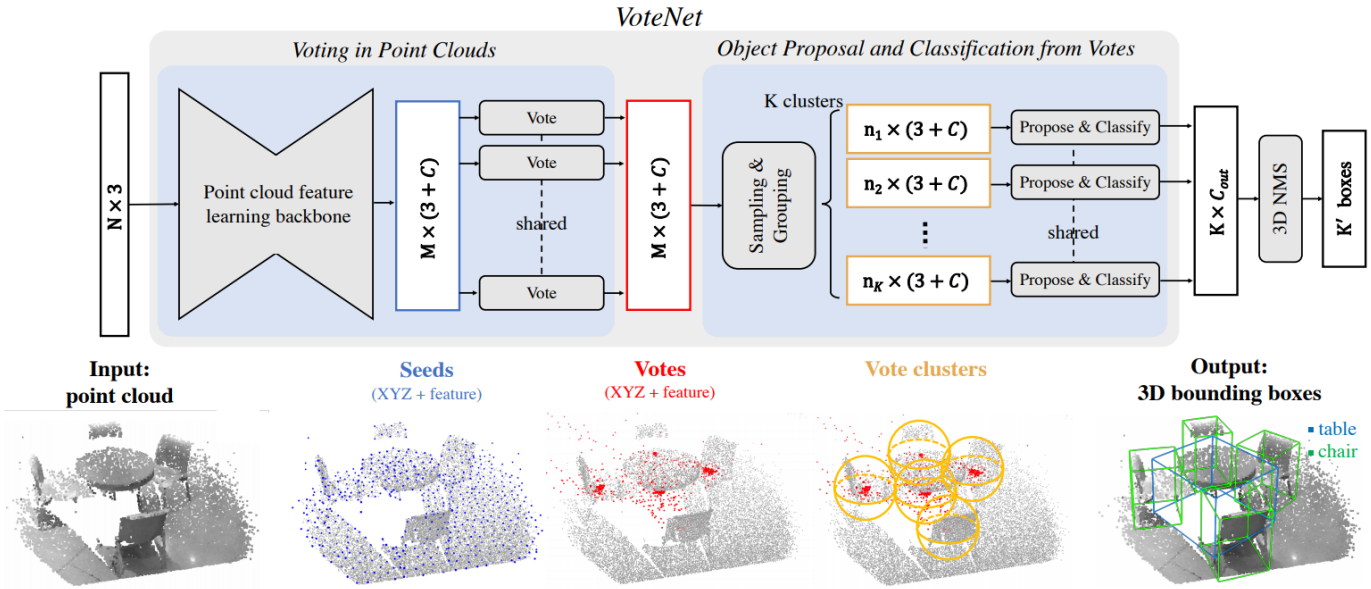
Continuous Convolution

S. Wang+. Deep Parametric Continuous Convolutional Neural Networks. CVPR2018.

Neural Fieldに近いアイデア

点群における物体検出手法 VoteNet

- 点群を入力し，各点が属する物体の中心座標を推定
- 点群上で定義された「物体中心を指すベクトル場」とみなせる
- 空間全体に拡張するとNeural Field



C. R. Qi+. Deep Hough Voting for 3D Object Detection in Point Clouds. ICCV2019.

まとめ

- NeRFに至る研究の背景について，
  Neural Fieldの観点から紹介
  「空間中の各点に対応した関数値」をニューラルネットワークでモデル化
  - 表面モデルのDeepSDF
  - 体積モデルのNeRF
- NeRFが想定する光学的なモデルと，
  Neural Fieldによる実現について紹介
- 特徴量のグリッド表現を紹介
  さまざまな工夫によって高速な学習・推論を実現