HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Folien zur Vorlesung am 24.06.2025

3D Computer Vision

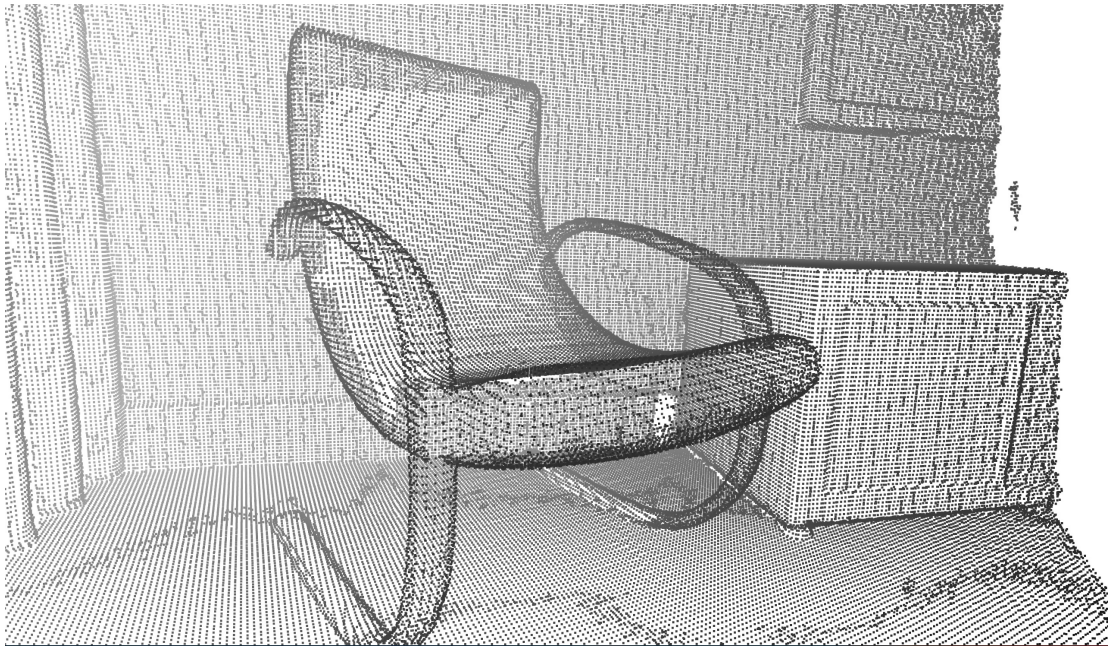
3D DATENSTRUKTUREN

Depth Maps

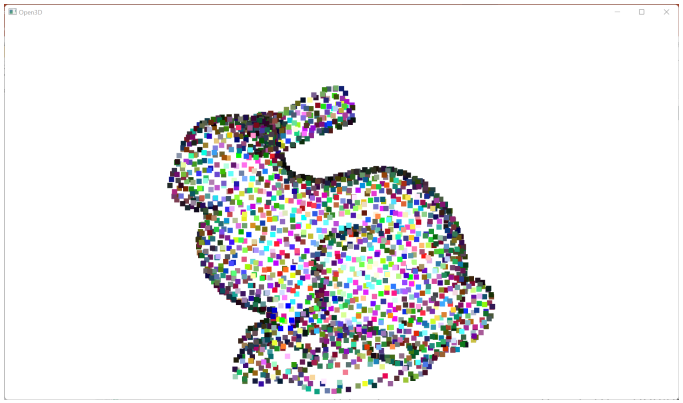
- Also known as **Range Images**
  - pixel values refer to distance
- Advantages
  - Dense representation
  - Gives intuition about occlusion and free space
  - Depth discontinuities are just edges on the image
- Disadvantages
  - Viewpoint dependent, can't merge
  - Doesn't capture physical geometry
  - Need actual 3D locations

3D Representations

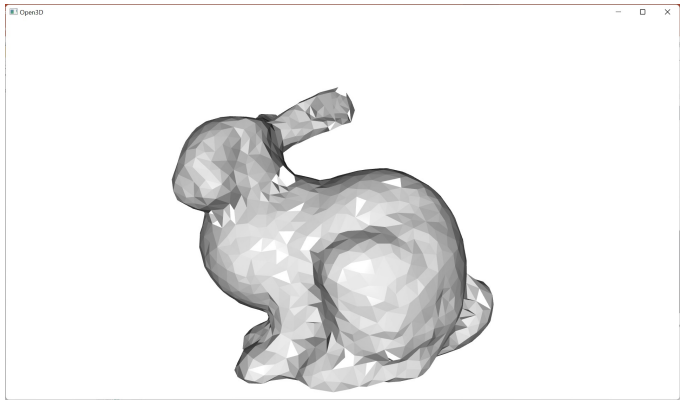
- Take every depth pixel and put it out in the world
- What can this representation tell us?
- What information do we lose?



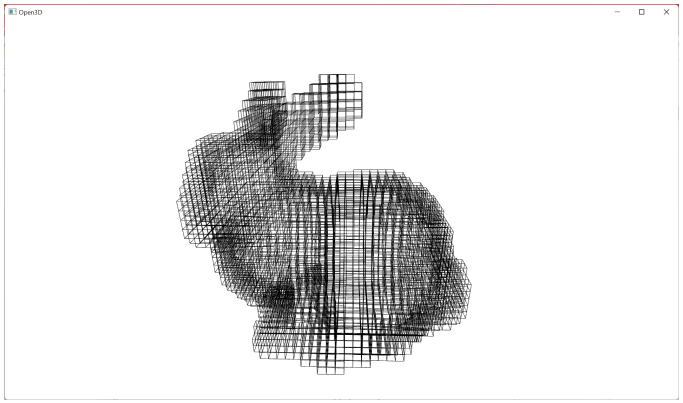
3D Representations



Point Cloud



Mesh



Volumetric

Projected View RGB(D)

...

https://faculty.cc.gatech.edu/~turk/bunny/bunny.html

3D Representations – Signed Distance Functions

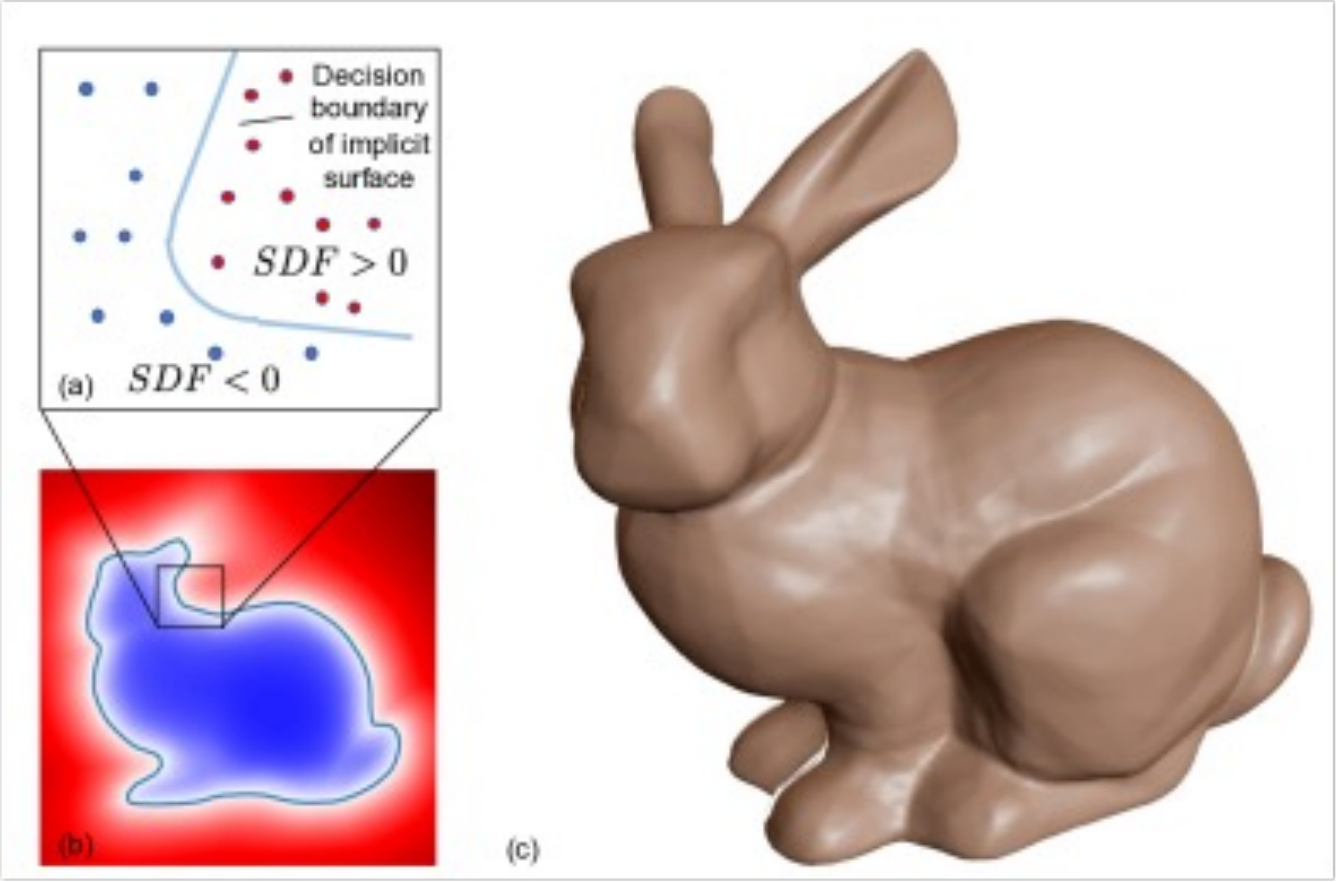


Image from [DeepSDF2019]

3D Representations – NeRF

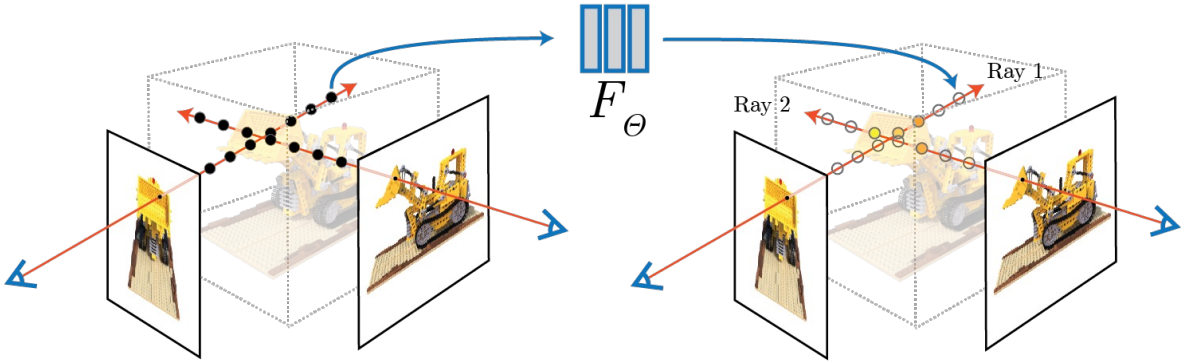
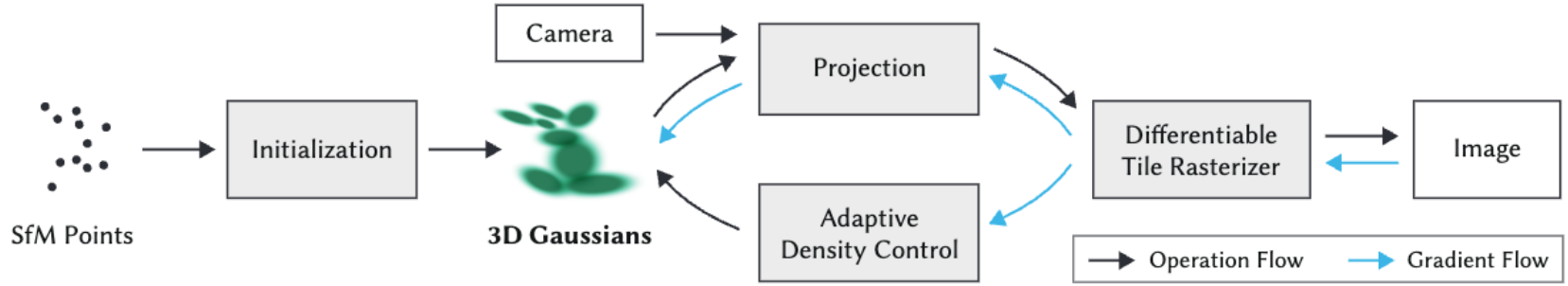


Image and video from NeRF project page, Matthew Tancik

3D Representations – 3D Gaussians



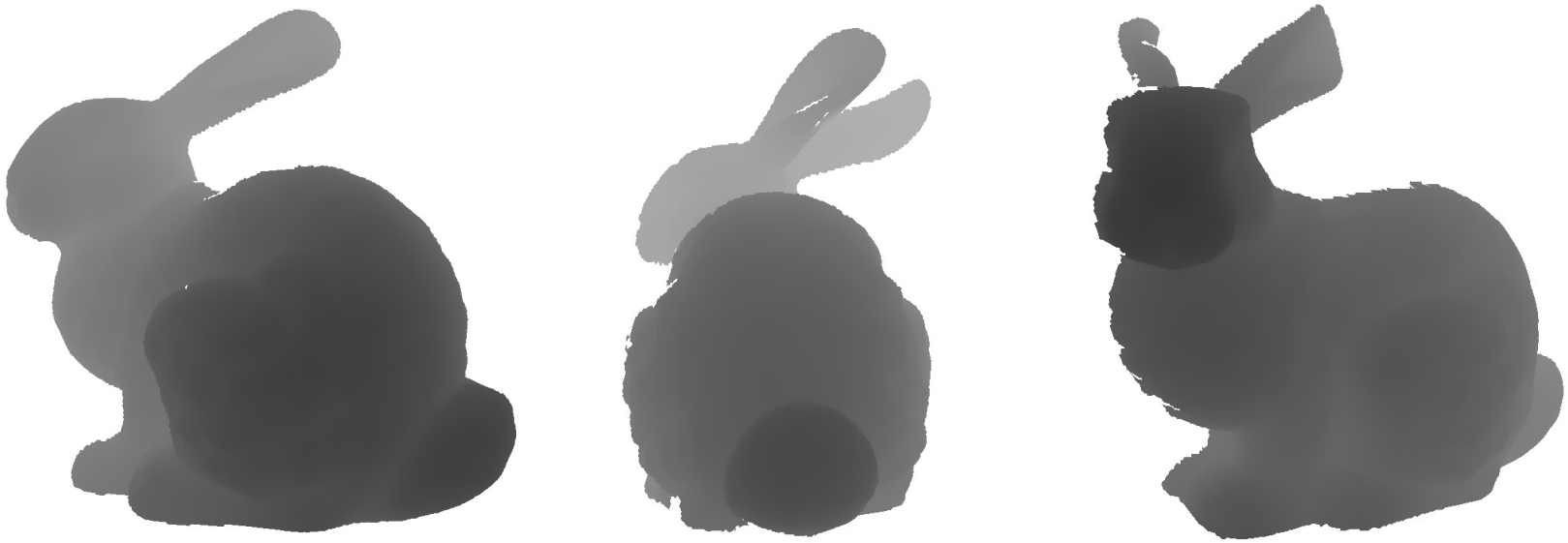
Images from [3DGS project page](), Bernhard Kerbl and Ethan Fuchs

Point clouds

- Advantages
  - Viewpoint independent
  - Captures surface geometry
  - Points represent physical locations
- Disadvantages
  - Sparse representation
  - Lost information about free space and unknown space
  - Variable density based on distance from sensor

Point clouds

- From several depth maps - How can we merge them to one point cloud?

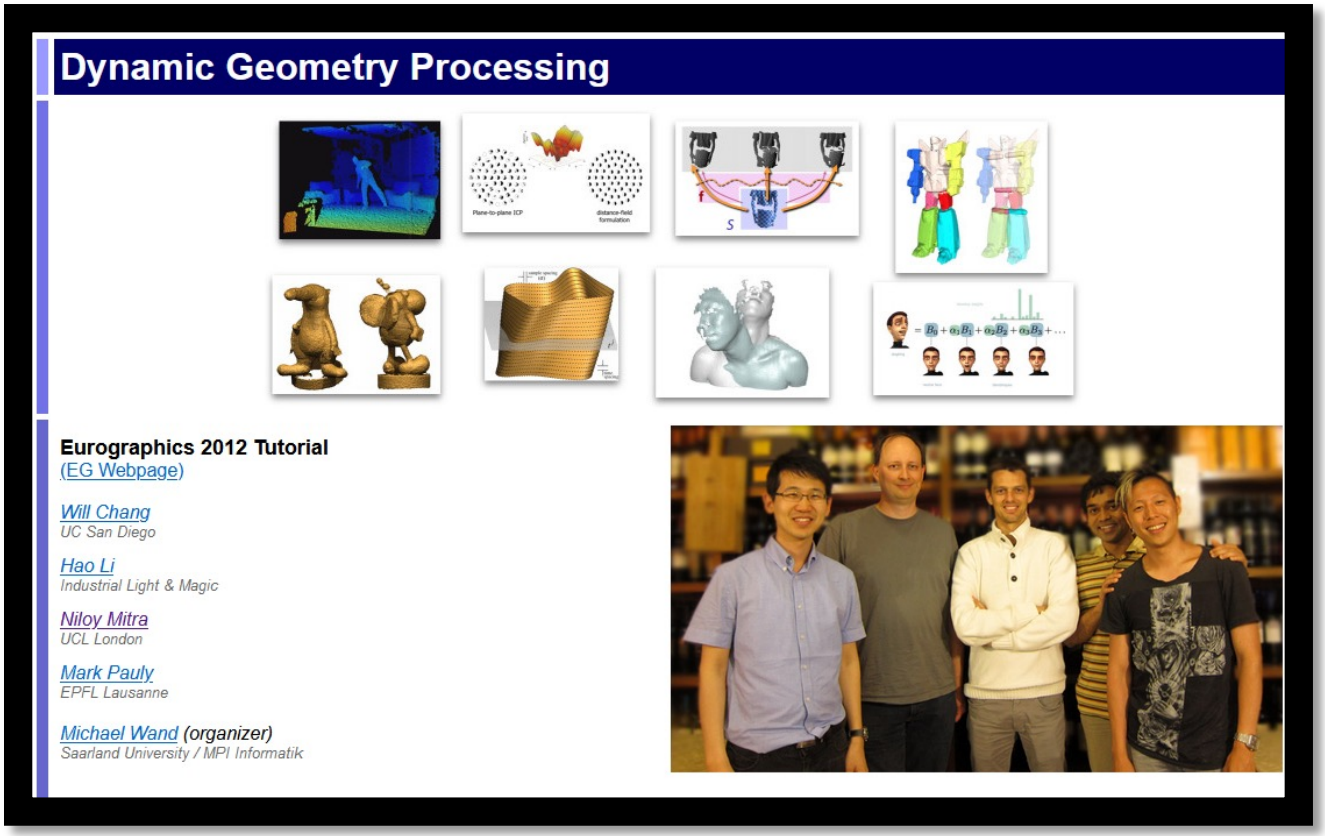


Point clouds

- Using the camera intrinsics, we get a point cloud for each depth map

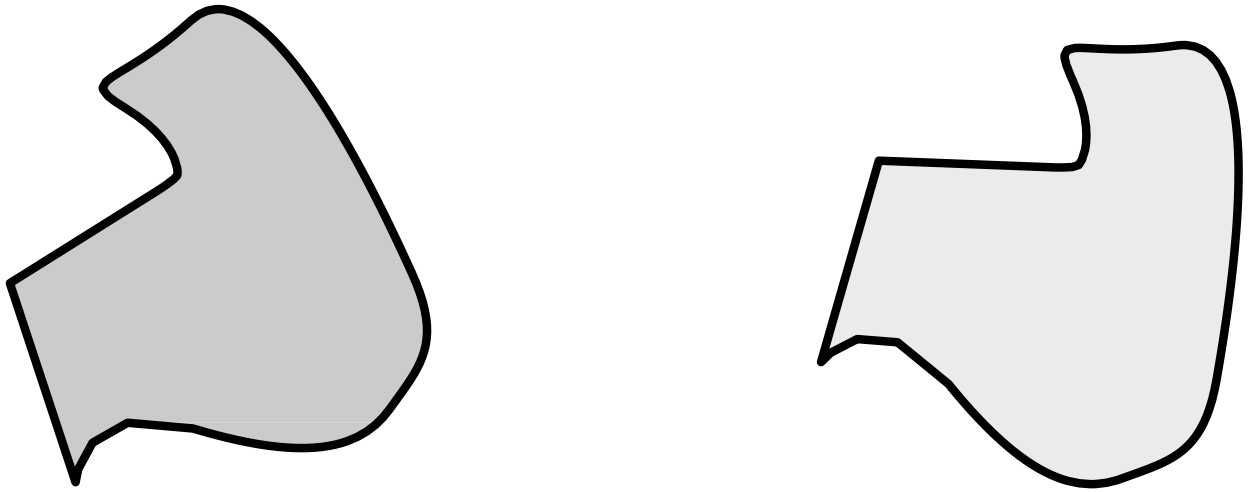
Iterative closest point (ICP)

- The following slides are copied from https://resources.mpi-inf.mpg.de/deformableShapeMatching/EG2012_Tutorial/



ICP - Goal

M_1 M_2



$$M_2 = T(M_1)$$

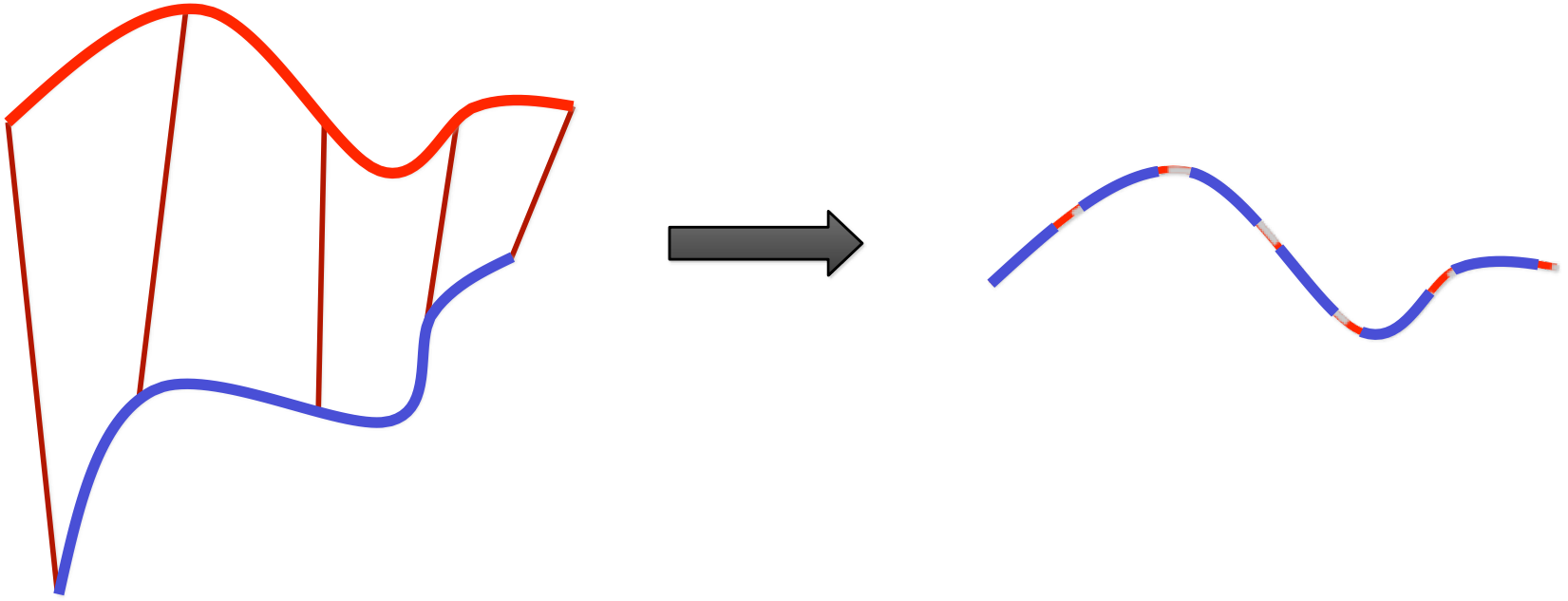
Find T, which is a rigid transform

ICP - Basics

- How many point-pairs are needed to uniquely define a rigid transform?
  - Rigid transform = Rotation + Translation
- Three point-pairs are needed. See [Horn1987] for the math. Translation is recovered by using the centroid of the points, rotation via quaternions.
- How it is found for more points than needed?
  - Translation again from centroids, rotation via SVD. See math.stackexchange.com for the details.

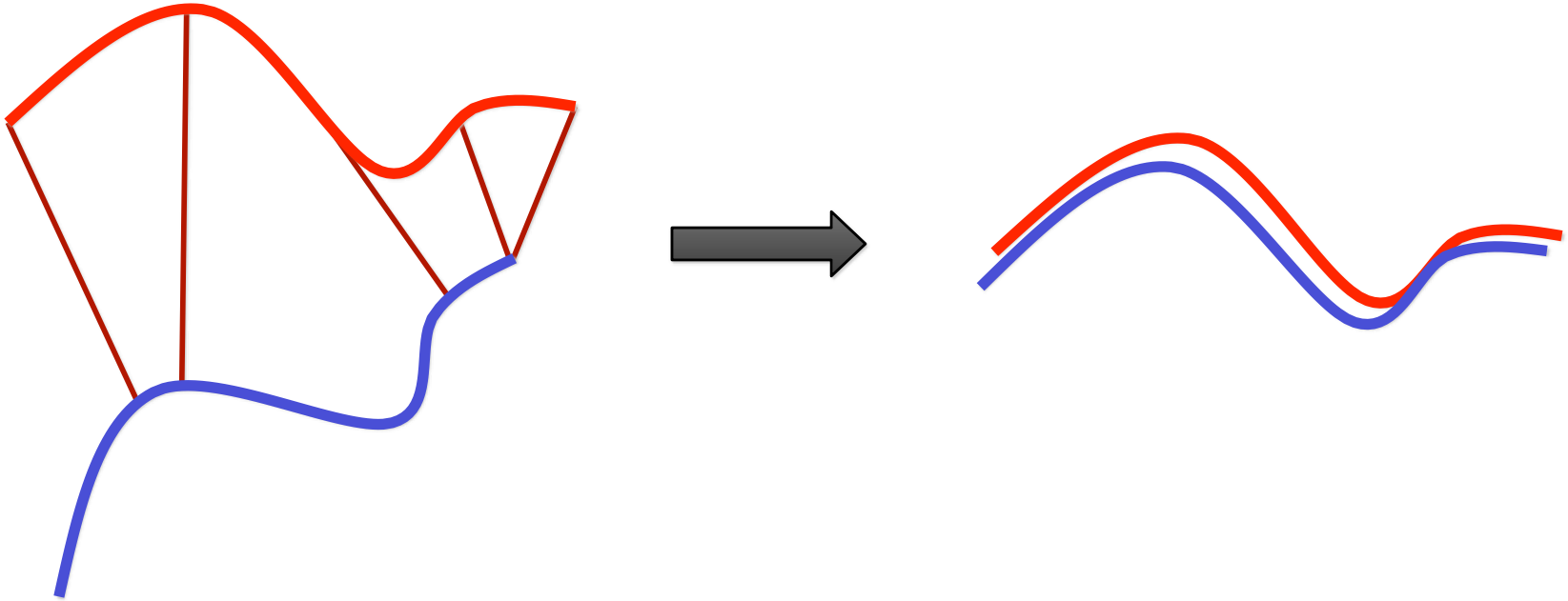
ICP - Goal: Aligning 3D Data

- If correct correspondences are known, you can find correct relative rotation/translation
- Typical use case: correspondences are unknown!



ICP - Problem

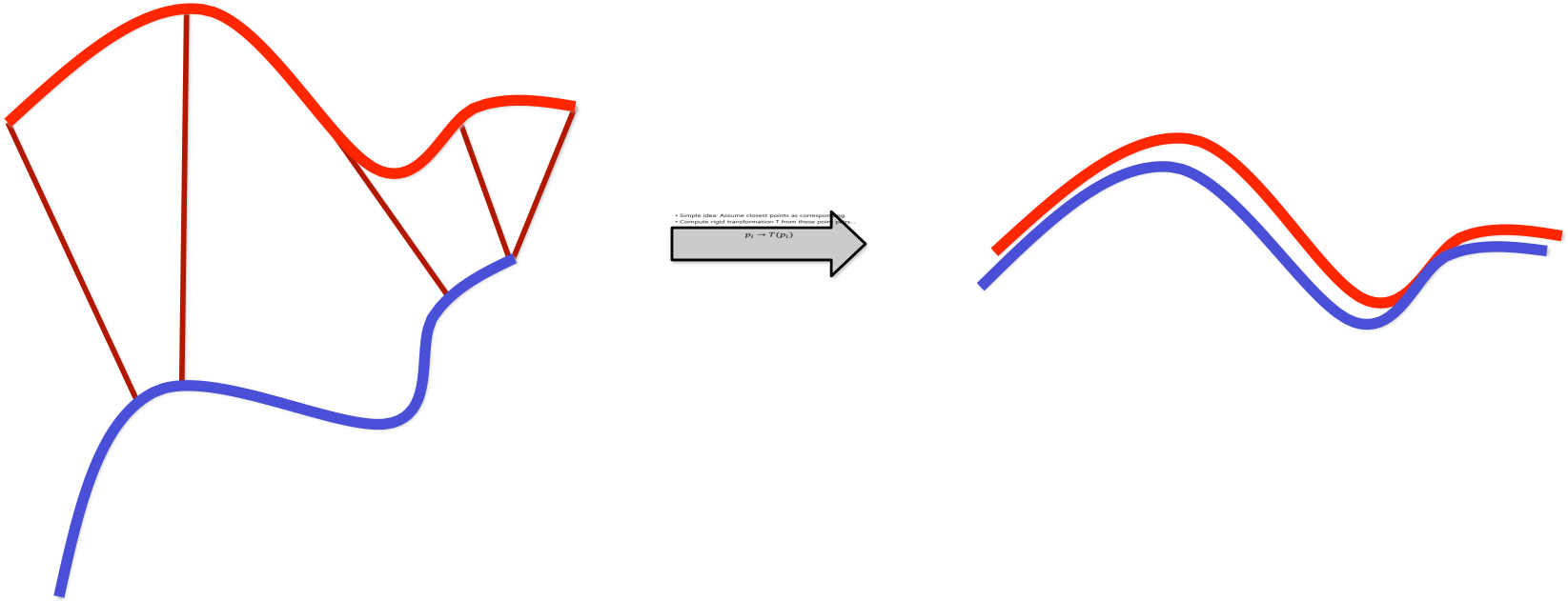
- How to find correspondences:
  - User input?
  - Feature detection?
  - Signatures?



ICP - Idea

- Simple idea: Assume closest points as corresponding.
- Compute rigid transformation T from these point pairs…

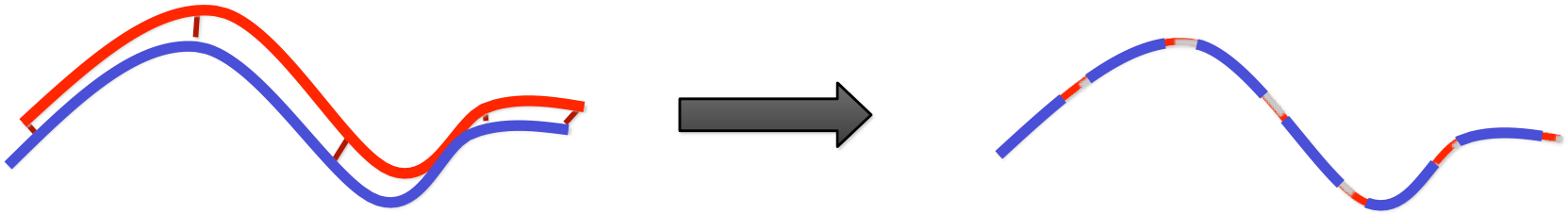
$$p_i \rightarrow T(p_i)$$



ICP - Idea

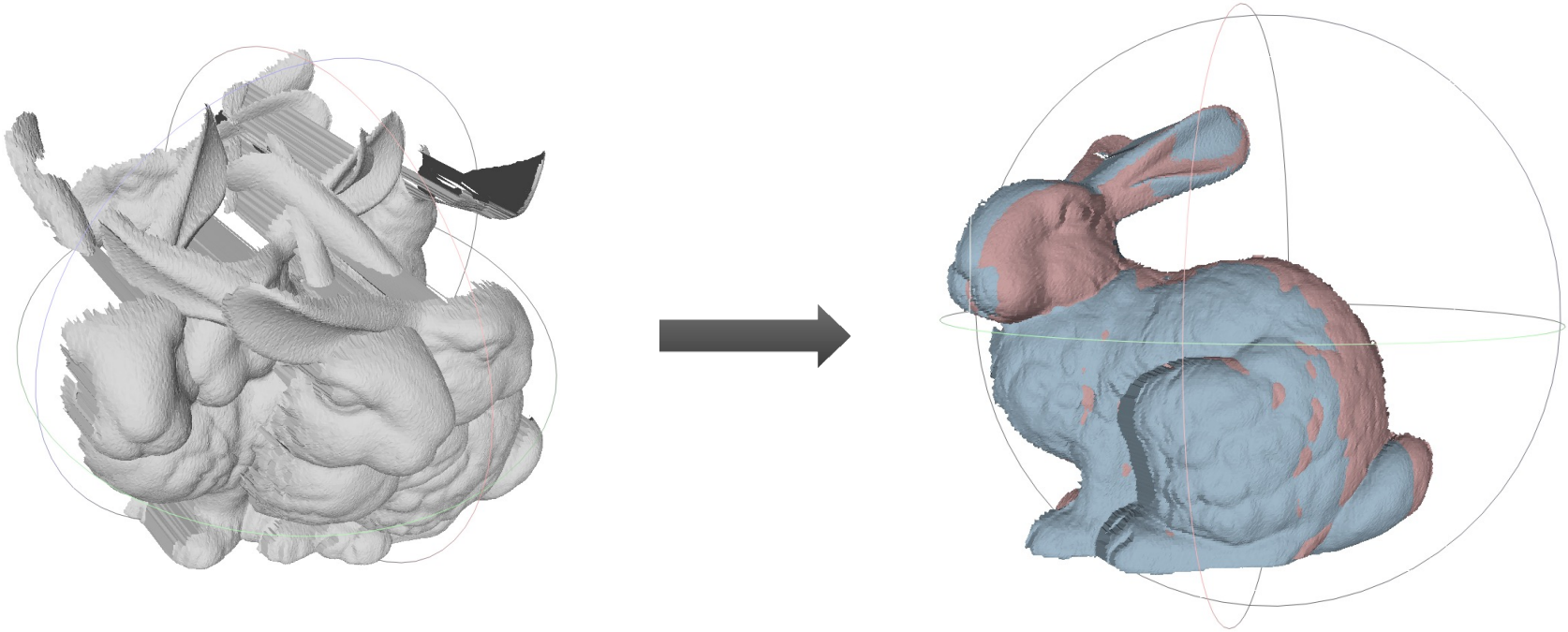
... and iterate to find alignment.

Converges if starting poses are close enough



Iterative Closest Points (ICP) [Besl and McKay 92]

ICP - Demo



Basic ICP

- **Select** (e.g., 1000) random points
- **Match** each to closest point on other scan, using data structure such as *k*-d tree
- **Reject** pairs with distance > *k* times median
- Construct **error function**:

$$E := \sum_i (Rp_i + t - q_i)^2$$

- **Minimize** (e.g. closed form solution in [Horn 87])

ICP - Variants

1. **Selection** of some set of points in one or both meshes.
2. **Matching** these points to samples in the other mesh.
3. **Weighting** the corresponding pairs appropriately.
4. **Rejecting** certain pairs based on looking at each pair individually or considering the entire set of pairs.
5. **Assigning an error metric** based on the point pairs.
6. **Minimizing** the error metric.

Analyze various aspects of performance depending on application:

- Speed
- Stability
- Tolerance of noise and/or outliers
- Maximum initial misalignment

Comparisons of variants can be found in [Rusinkiewicz & Levoy, 3DIM 2001]

ICP - Variants

Closest points are often bad as corresponding points

Matching effectiveness can be improved by restricting match to compatible points

- Compatibility of colors [Godin et al. 94]
- Compatibility of normals [Pulli 99]
- Other possibilities:
  - curvatures, higher-order derivatives, and other local features

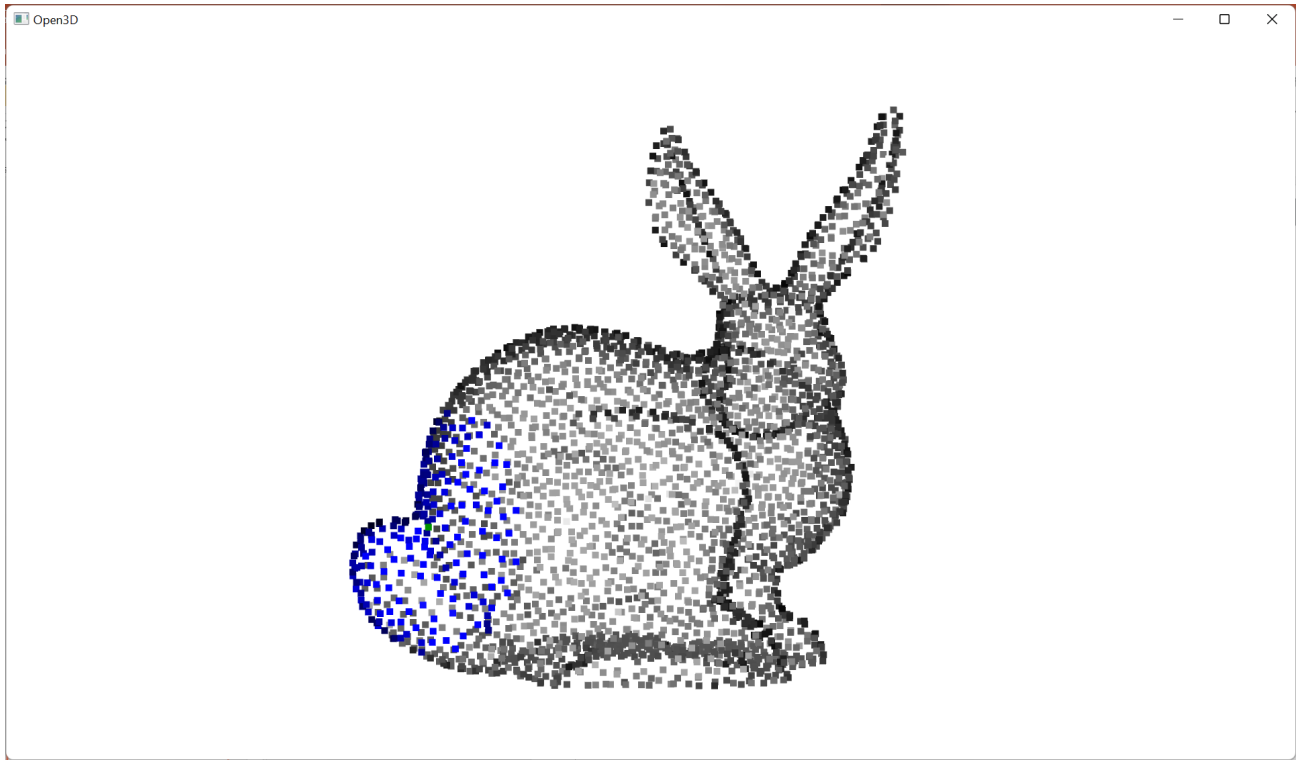
Or: Improvement by fast distance calculations.

Finding neighbours in point clouds

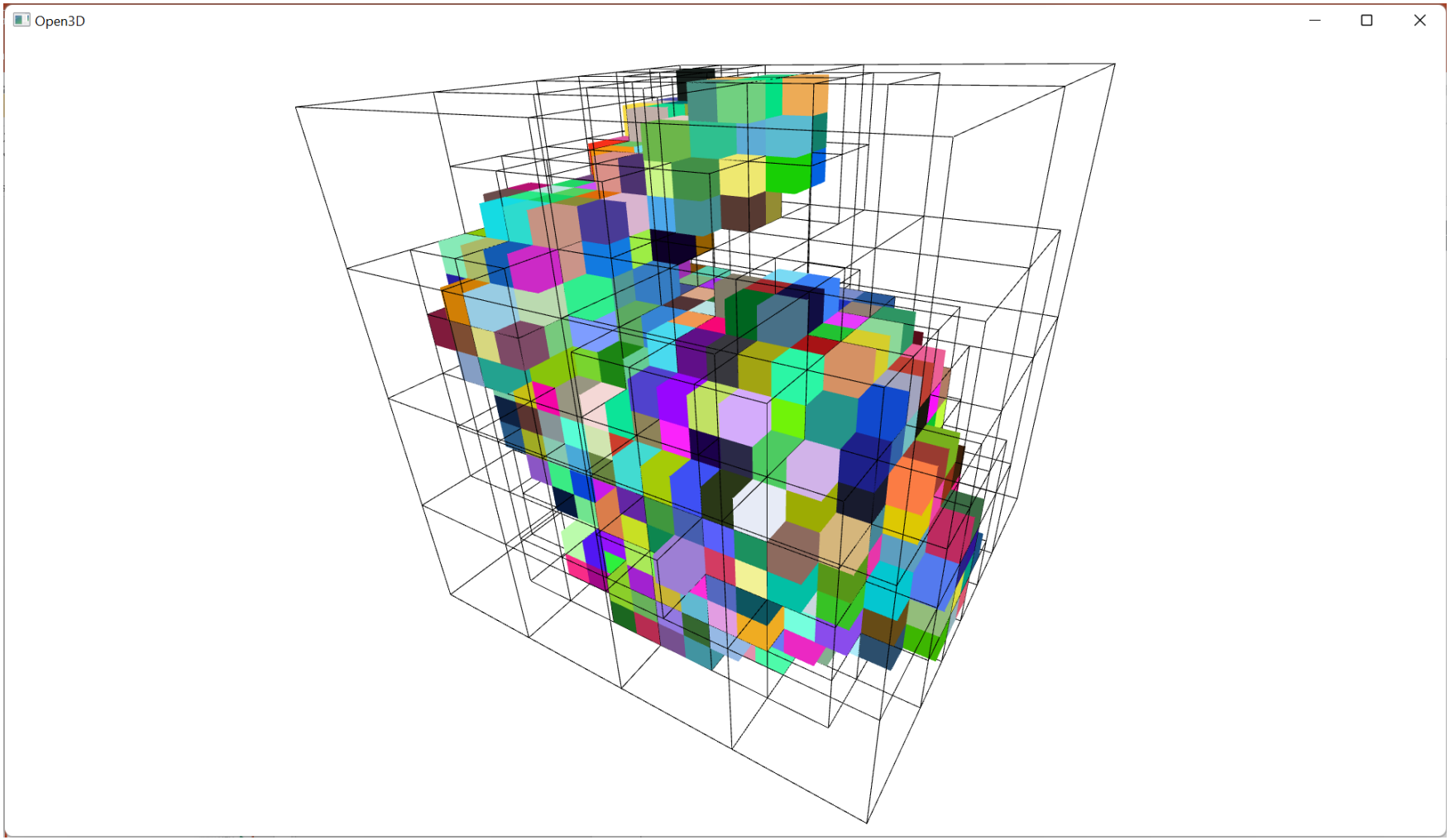
- We need data structures in order to avoid computing the distance between all points.
- General idea:
  - The position of an element in a data structure reflects its position in the coordinate space.

Point clouds

- Neighbour information is lost.
- How can we realize painting some regions of a point cloud?



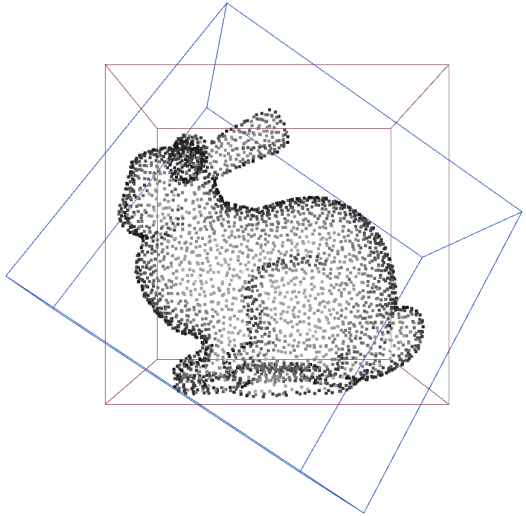
Octree



Octree: Creation

Creating an octree from a set of 3D points:

1. Find/define the bounding 3D volume.
2. Divide the current 3D volume into eight boxes.
3. If any box contains more than one point then divide it further into boxes.
4. Do not divide the box which contains one or zero points in it.
5. Do this process repeatedly until all the boxes contain one or zero point.



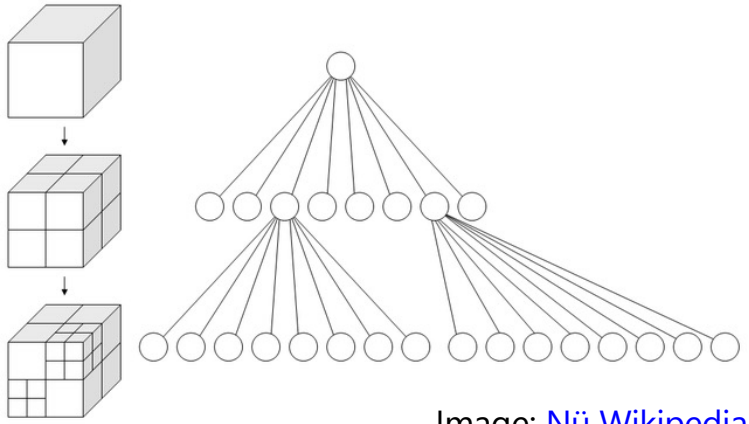


Image: Nü Wikipedia

Source: ParthManiyar

Octree: Insertion

Adding further points to an octree:

1. Check if point is not yet in the tree and inside the 3D volume
2. Start at the root node.
3. Traverse down the tree according to the 3D coordinates of the point until an empty node is found.

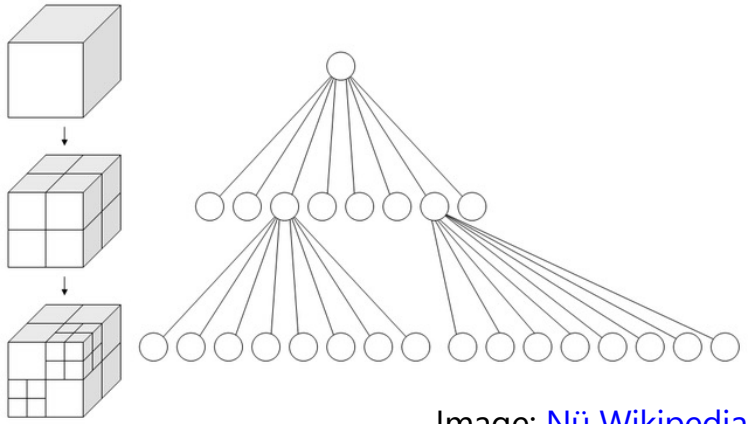


Image: Nü Wikipedia

There is a fixed sub-volume for each point where it can be added to the octree.

Source: ParthManiyar

Octree: Searching

Searching for a point can be done recursively and the point coordinates directly

Start at the root node and traverse down the tree according to the 3D coordinates of the point until the point is found or an empty node is found.

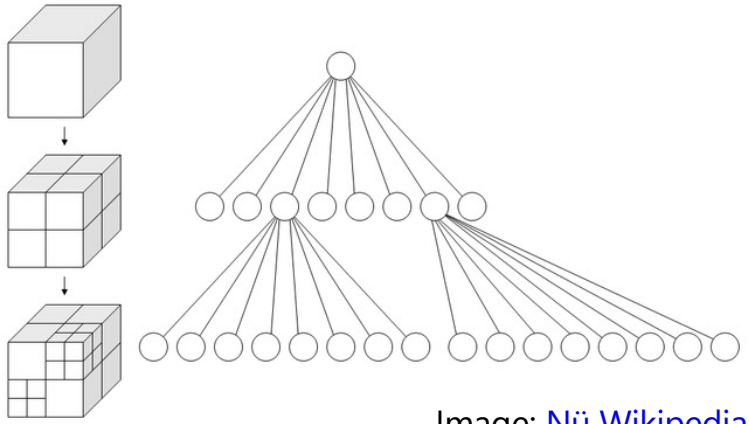
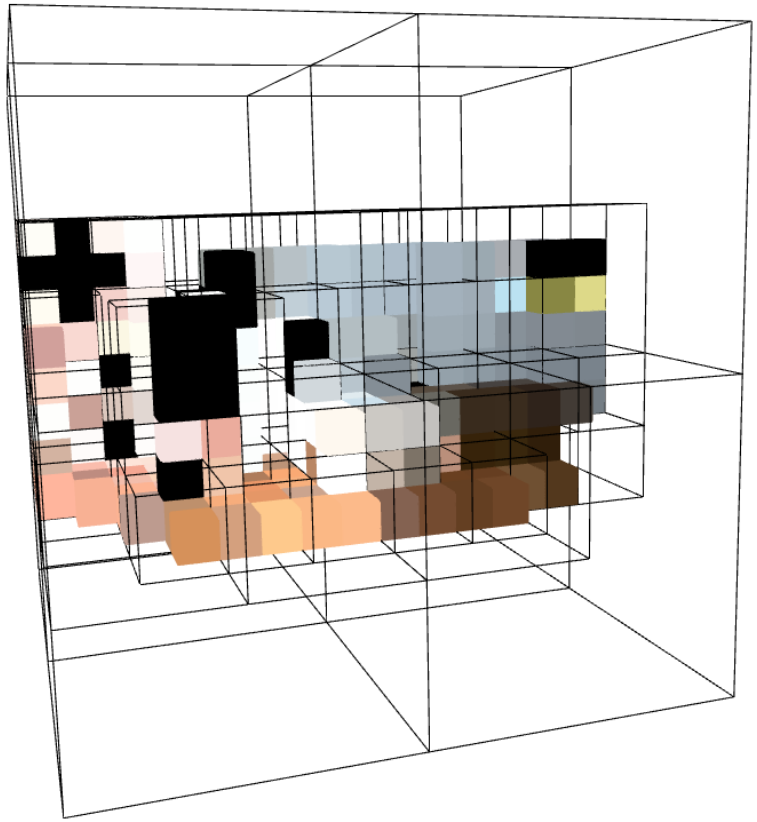


Image: Nü Wikipedia

Source: ParthManiyar

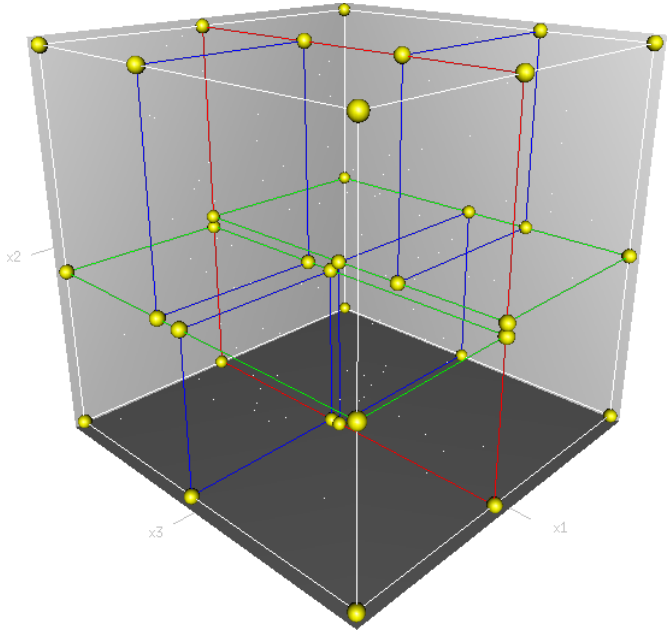
Octree

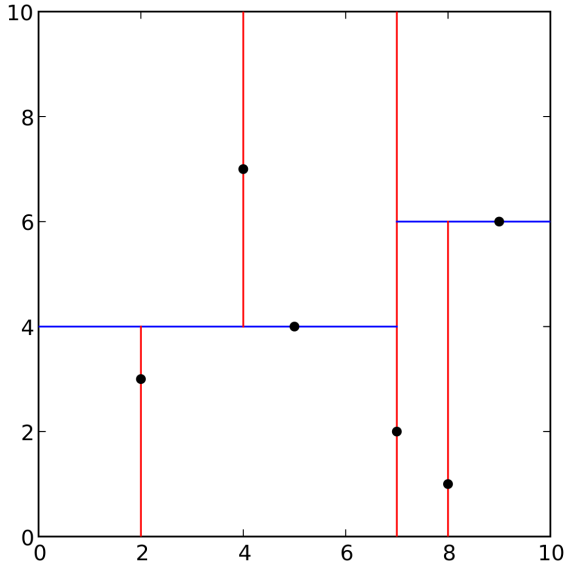


- Depending on the scene, octrees are usually not well balanced.

kD-trees

The *k*-d tree is a binary tree in which *every* node is a *k*-dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts, known as half-spaces.





By KiwiSunset at the English-language Wikipedia, CC BY-SA 3.0,

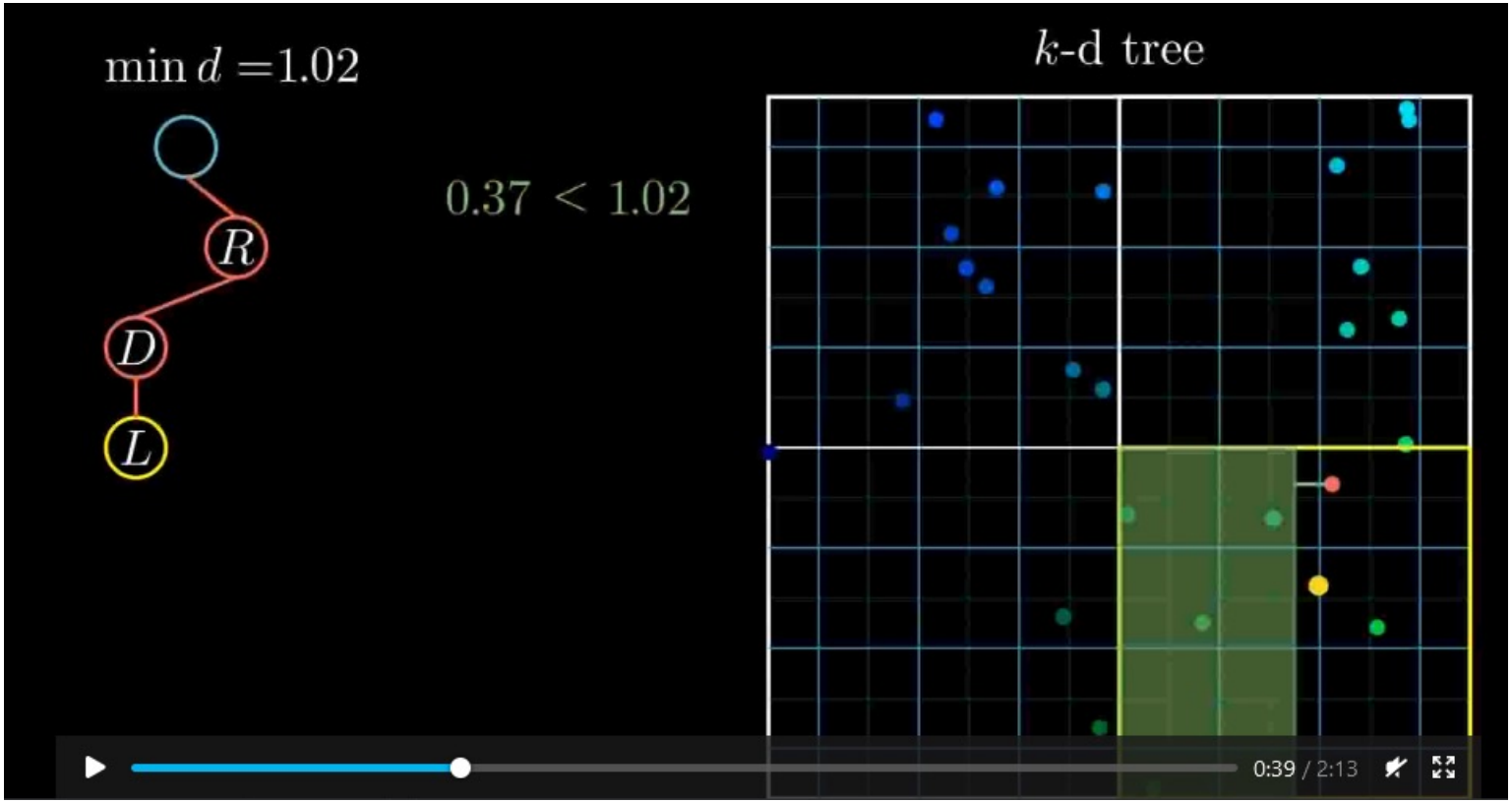
Recommended reading Wikipedia and [CMU, Carl Kingsford]

kD-trees – Nearest neighbour search

- Starting with the root node, the algorithm moves down the tree recursively.
- Once the algorithm reaches a leaf node, it checks the node point and if the distance is better than the "current best", that node point is saved as the "current best".
- The algorithm unwinds the recursion of the tree, performing the following steps at each node:
  - If the current node is closer than the current best, then it becomes the current best.
  - The algorithm checks whether there could be any points on the other side of the splitting plane that are closer to the search point than the current best.
- When the algorithm finishes this process for the root node, then the search is complete.

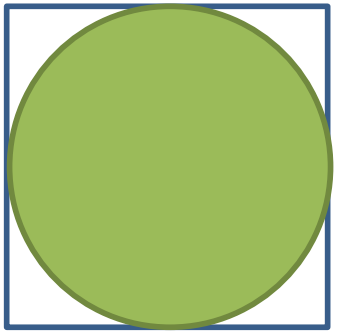
kD-trees – Nearest neighbour search

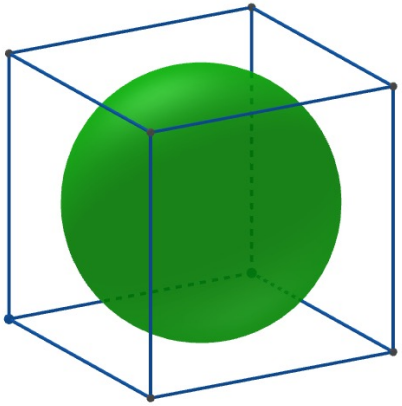
- Video von Wikipedia:
  https://upload.wikimedia.org/wikipedia/commons/4/48/Kdtreeogg.ogv



kD-trees - Limitations

- Degradation in performance with high-dimensional data
  - Curse of dimensionality

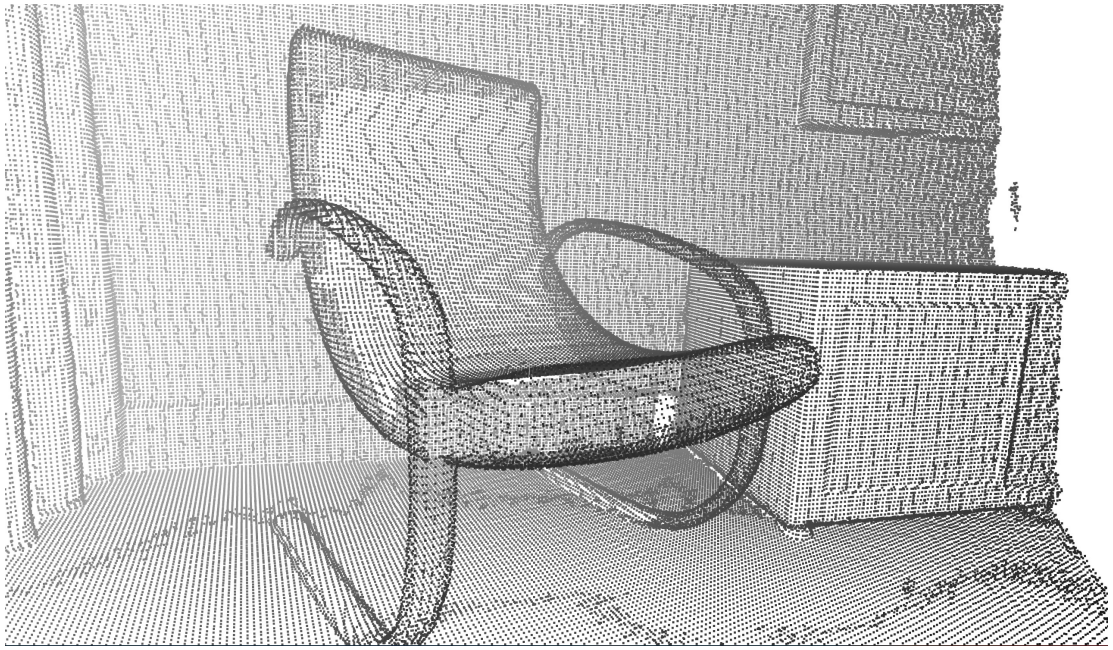




- Degradation in performance when the query point is far from points in the k-d tree

3D Representations

- Computation complexity one factor higher than for images
- Math slightly more complex in 3D than in 2D



3D Representations

2D vs. 3D Perception

Analysis Tools	2D	3D
Representation	Image (u,v)	• Depth image (u,v,d) • Point cloud (x,y,z)
1st order differential geometry	Image gradients	Surface normals
2nd order differential geometry	Second moment matrix	Principle curvature
Corner detection	Harris image	Surface variation
Feature extraction	HOG	• Point Feature Histograms • Spin Images
Geometric model fitting	Hough transform	Clustering + RANSAC
Alignment	SSD window filter	Iterative Closest Point (ICP)

Using point clouds in deep learning

Point cloud is **converted to other representations** before it's fed to a deep neural network

Conversion	Deep Net
Voxelization	3D CNN
Projection/Rendering	2D CNN
Feature extraction	Fully Connected