Automotive Sensors

Point Cloud Processing

Automotive Intelligence Lab.

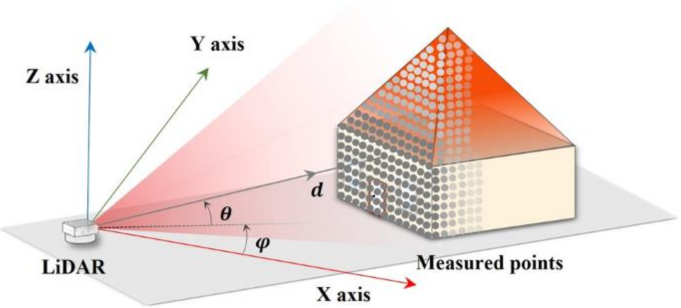
LiDAR Point Cloud

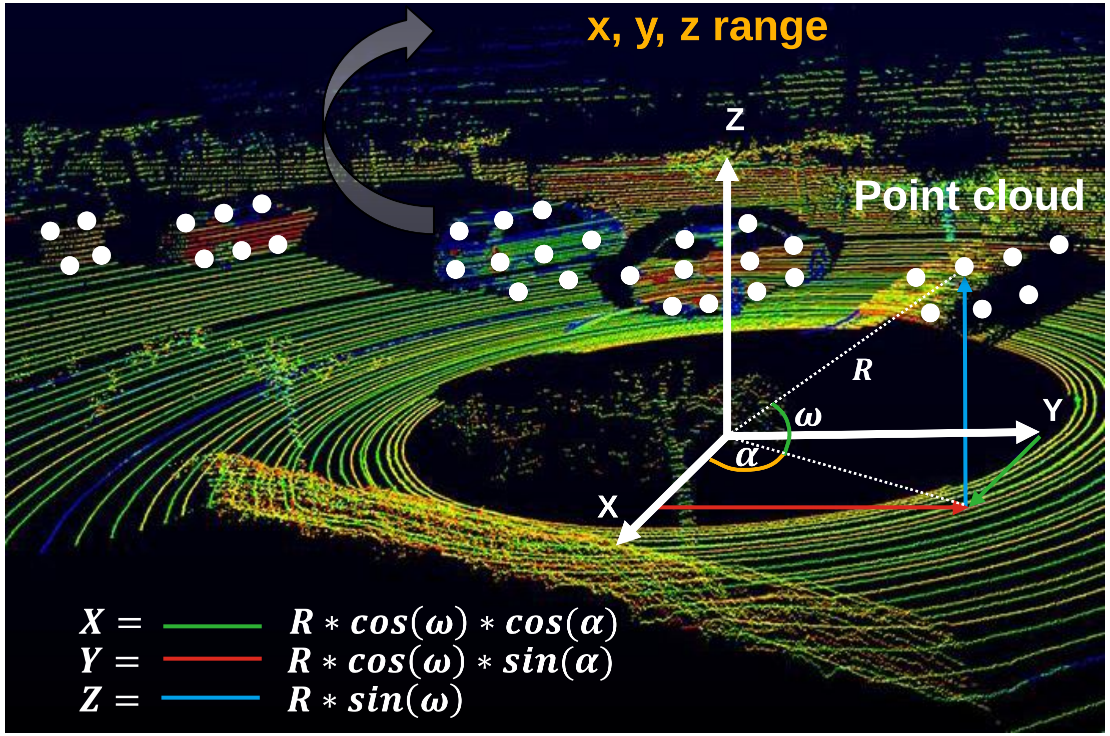
LiDAR Point Cloud

- LiDAR provides the distance (R) with the horizontal (α) and vertical (ω) angles from the lidar center to the object in polar coordinates.

- This information is transformed into X, Y, Z on the LIDAR-centered Cartesian coordinate system through the formula below.
  - $X = R\,cos(\omega)\,cos(\alpha)$
  - $Y = R\,cos(\omega)\,sin(\alpha)$
  - $Z = R\,sin(\omega)$

- A set of these points is called a point cloud and can represent a shape of 3D environment!

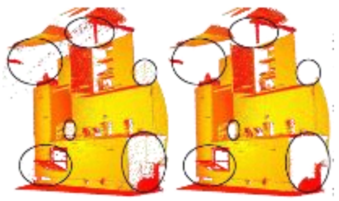




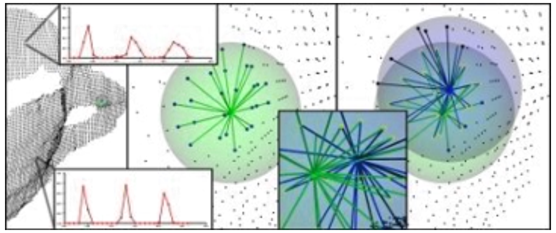
Point Cloud Processing

- Filters
- Features
- Keypoints
- Registration
- KDTree
- Segmentation
- Sample consensus
- Surface

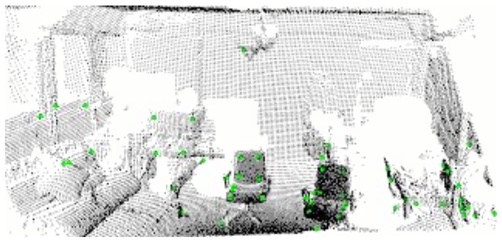
filters



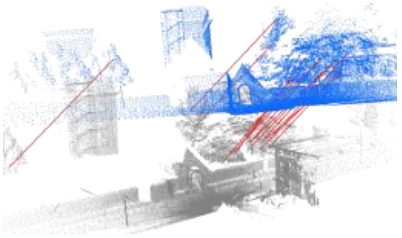
features



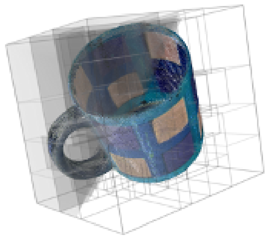
keypoints



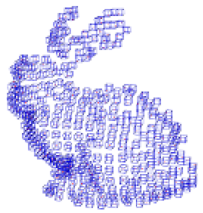
registration



kdtree



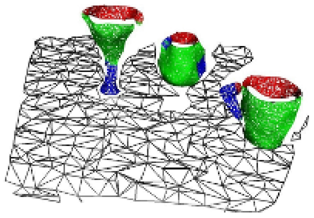
octree



segmentation

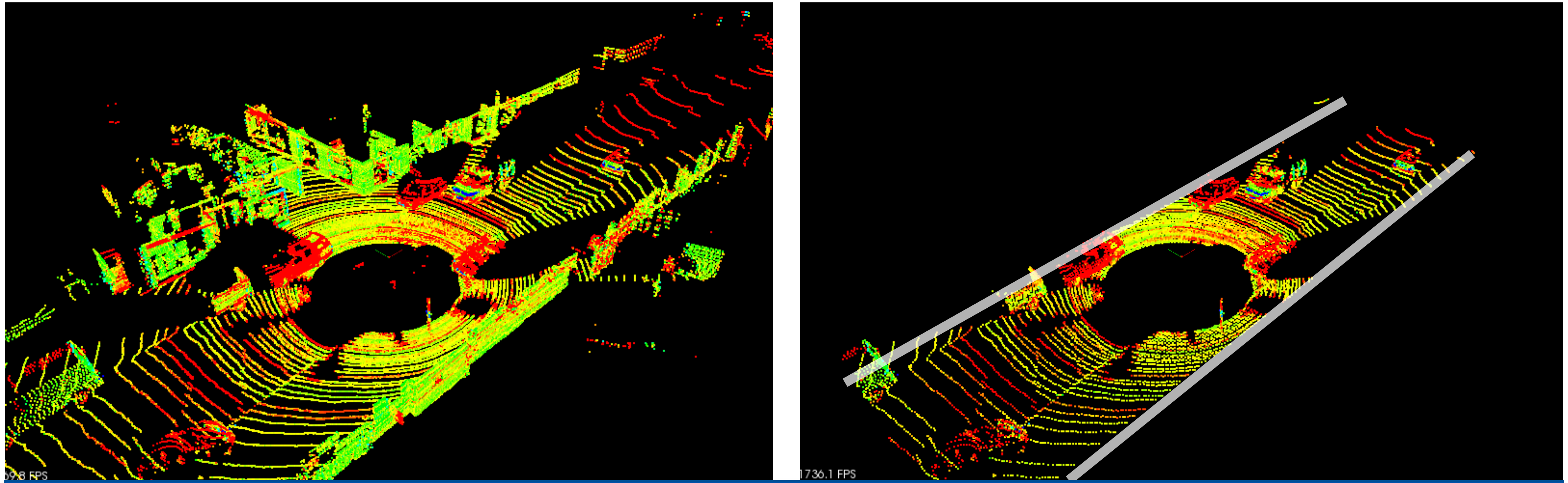
sample_consensus

surface



Filters - ROI Extraction

- **Point cloud data received by lidar can be cut by setting the ROI (Region Of Interest).**
- **Performing to make the data more suitable for subsequent analysis or application.**
  - Setting ROI(Region Of Interest)to reduce the amount of point cloud data computation is also a process of preprocessing

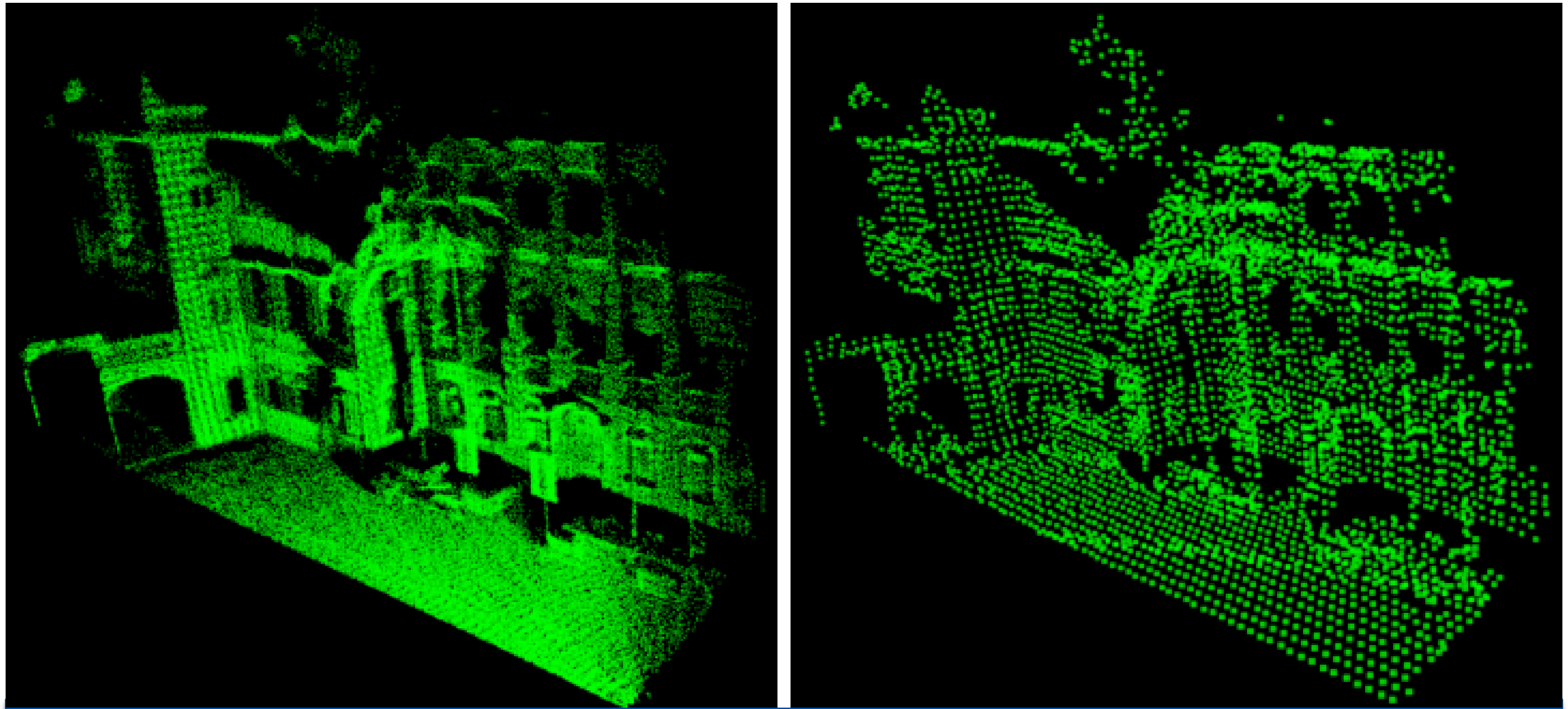


Setting ROI (Region Of Interest)

Filters - Downsampling

Downsampling

- Reducing the number of points in a point cloud
  - Ex) Voxel grid filter, Random sampling



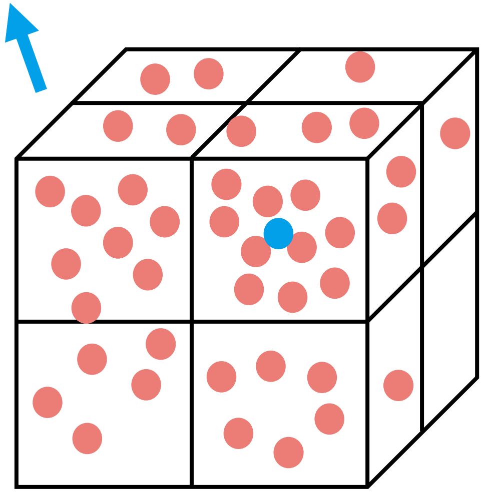
Down Sampling

Downsampling Example (1/2)

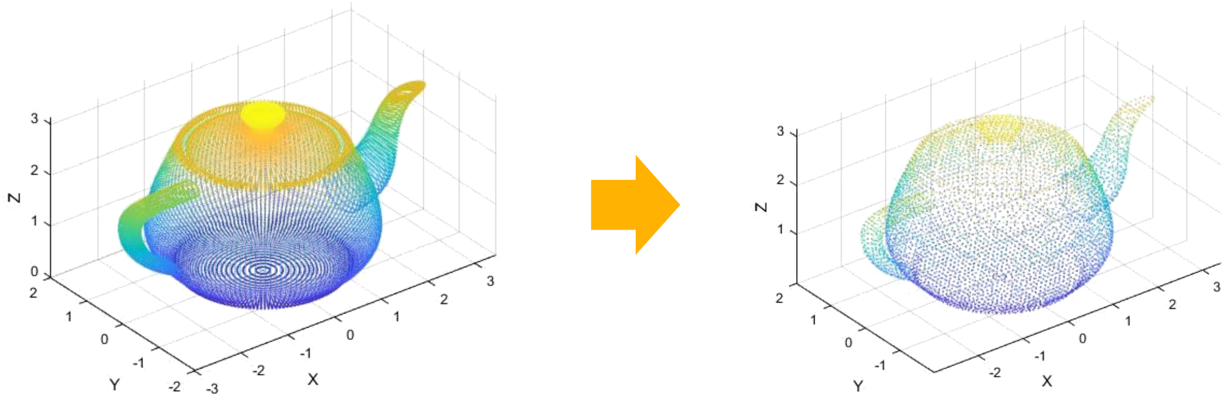
Voxel grid filter

- Frist, divide bounding box into grid boxes.
- Then, merge every point in each grid box.
- The merged points mean the average of locations, colors, and normal of each point.

The "voxel (volume + pixel)" represents a value on a regular grid in 3D space.



Voxel



Original

After Voxel Grid Filtering

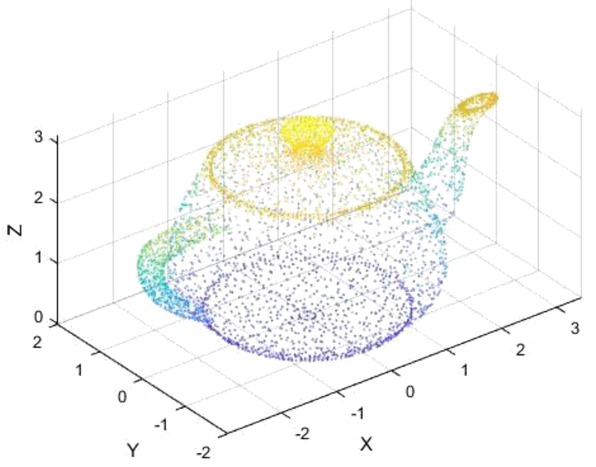
Downsampling Example (2/2)

Non-uniform grid sampling

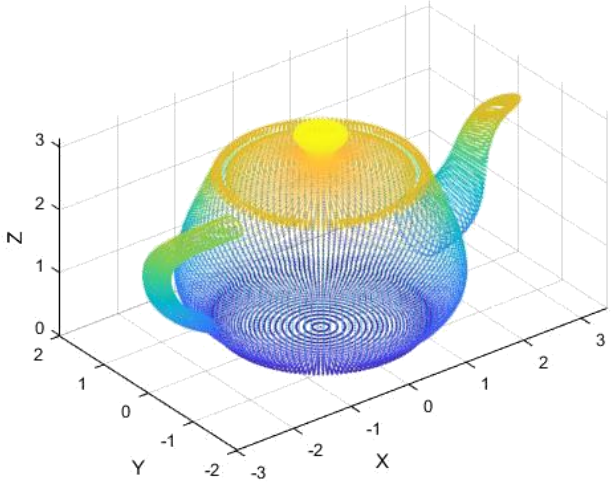
- The normal are computed on the original data prior to down sampling.
- The downsampled output preserves more accurate normal.

Random sampling

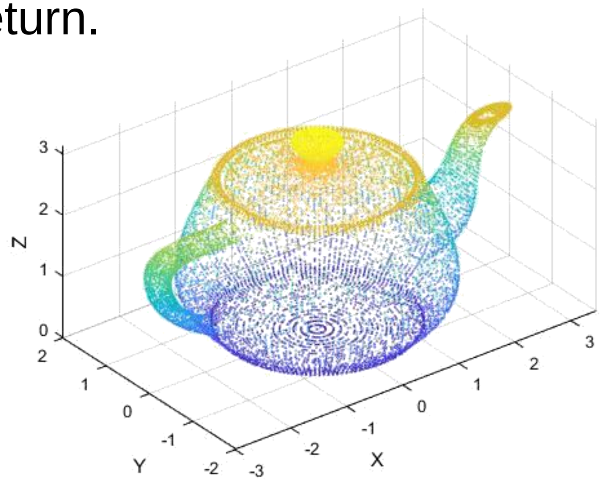
- Downsample randomly with percentage, which means portion of the input for the function to return.



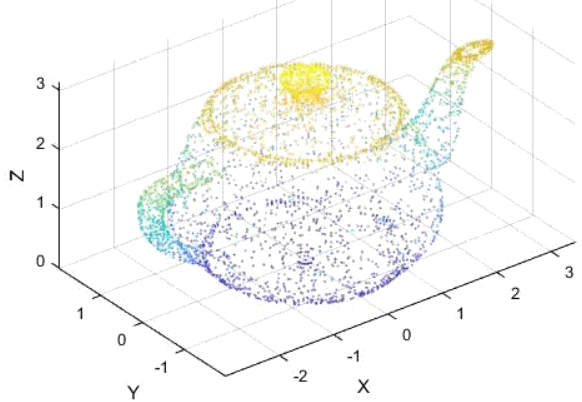
After non-uniform grid sampling



Original



Percentage 0.4 (40%)



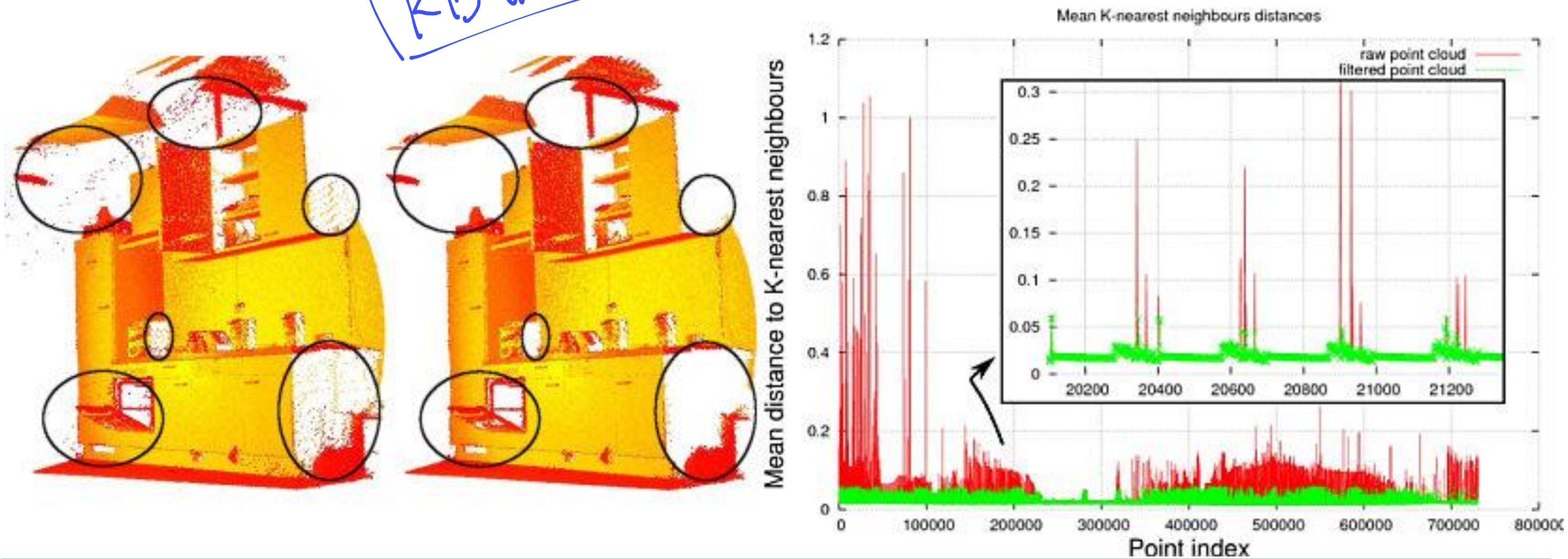
Percentage 0.1 (10%)

Filters – Outlier Removal Filter

Statistical outlier removal filter

- Point cloud scan's irregularities can be addressed by statistically analyzing and removing points that fail to meet specific criteria.
- Removal process evaluates the distances between neighboring points in the dataset.

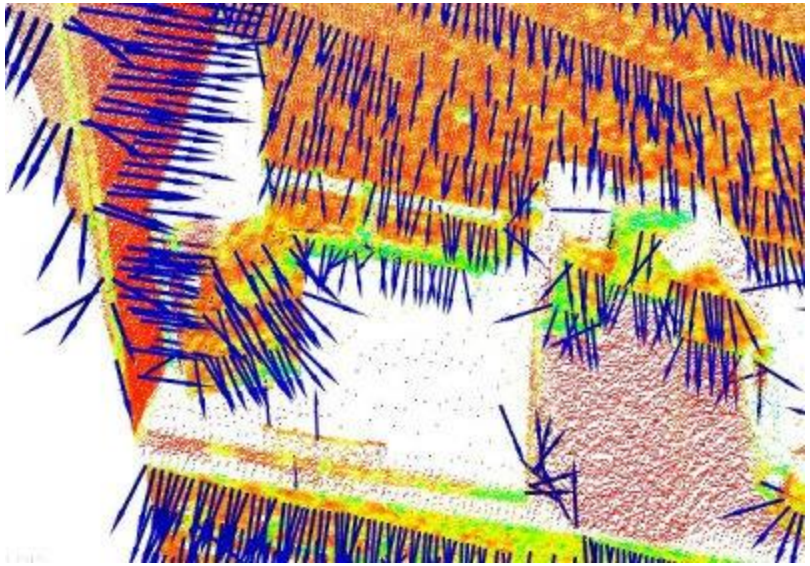
KD Tree

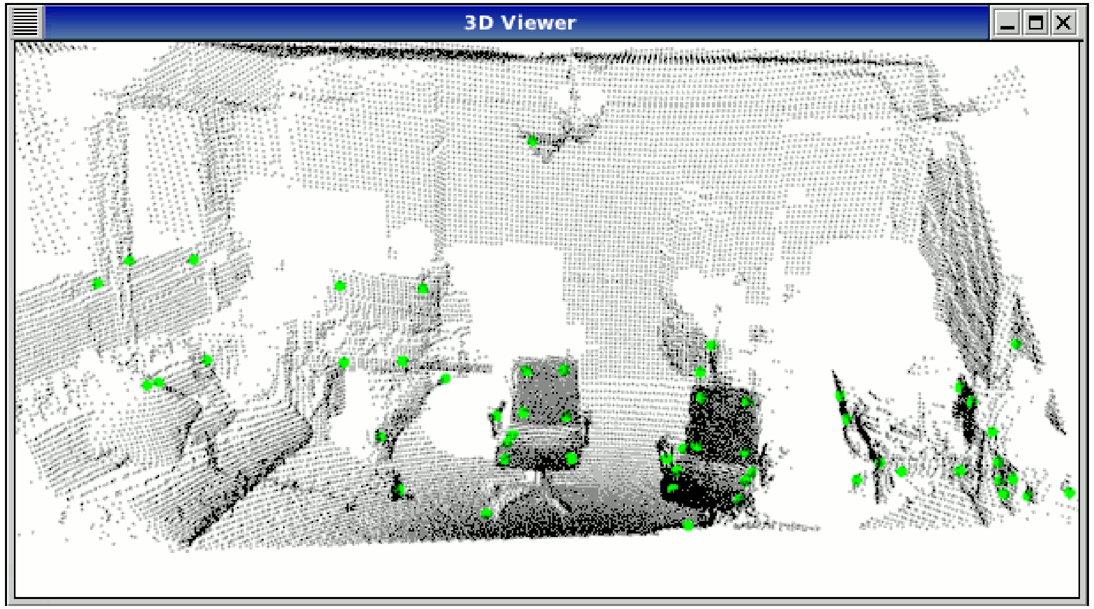


Outlier Removal Filter

Features & Key Points

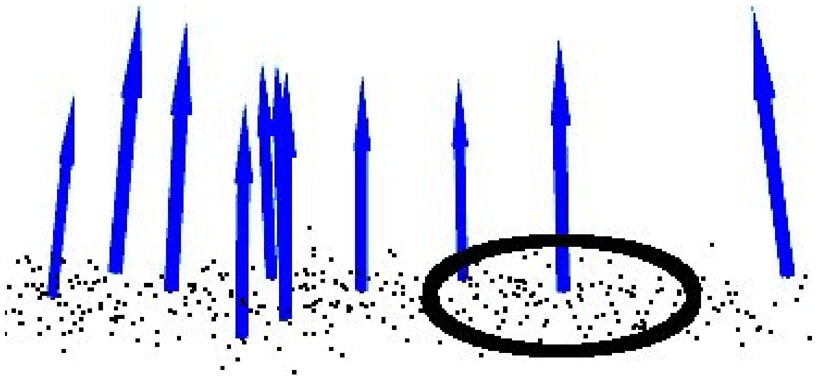
- **3D geometric point features**
  - Describes geometrical patterns based on the information available around the point.
  - Keypoints (also referred to as interest points) are points in a point cloud that are stable, distinctive, and can be identified using a well-defined detection criterion.
- **Available feature**
  - Point Feature Histograms (PFH), Fast Point Feature Histogram (FPFH), Viewpoint Feature Histogram (VFH), Moment of inertia and eccentricity feature, Normal Aligned Radial Feature (NARF)



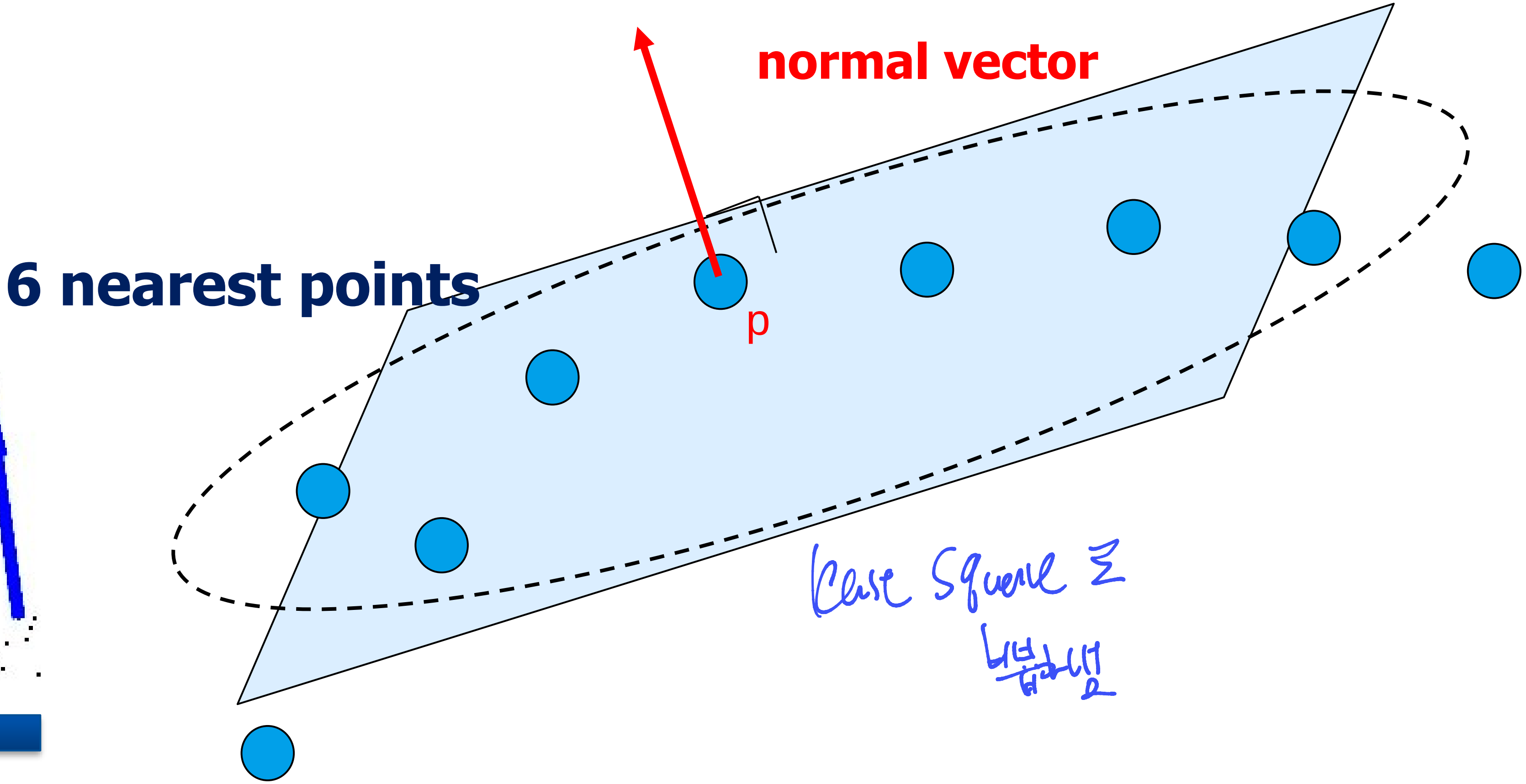


Features - Point Cloud Normal

- **Surface's estimated curvature and normal at a query point p is the most widely used geometric point features.**
  - Curvature and normal are considered local features, as they characterize a point using the information provided by its *k nearest point* neighbors.



Surface Normal

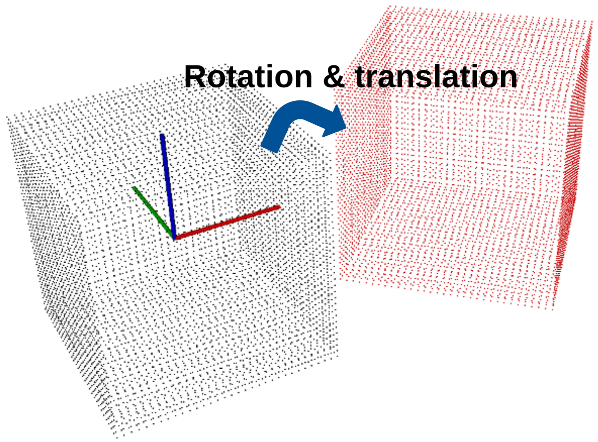


Least Square로

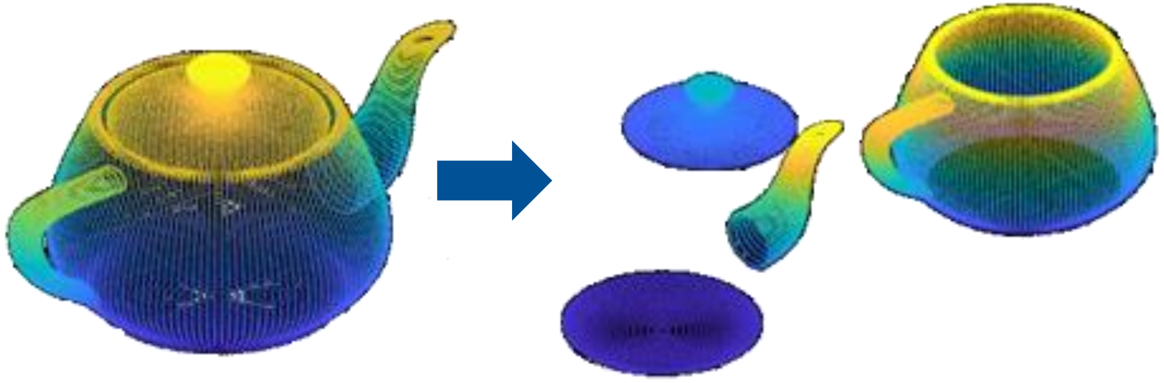
Transformation

- **Representation for relationship between different frames and objects**
  - Point cloud can be moved applying transformation rules.

- **Types of transformation**
  - Linear transformation (affine transformation)
    - Rigid transformation (Rotation & translation)
    - Scaling
    - Reflection
    - Shear
  - Non-linear transformation
    - Displacement of each point



Rigid transformation

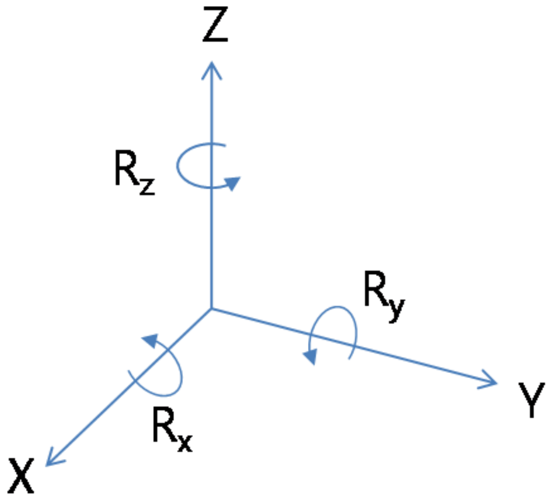


Non-linear transformation

Transformation of Point Clouds

- **Usage of transformation matrix and homogenous coordinates**
  - Point cloud in 3-dimensional space can be transformed by multiplying a transformation matrix.

- **In 3-dimensional space,**
  - The rigid transformation is done by rotation and translation.
  - There are three-axis for rotation and the order of rotation is important.



Coordinate of 3D Space

$$R_x(\theta)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & -\sin\theta\\ 0 & \sin\theta & \cos\theta\end{bmatrix}$$

$$R_z(\theta)=\begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$R_y(\theta)=\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}$$

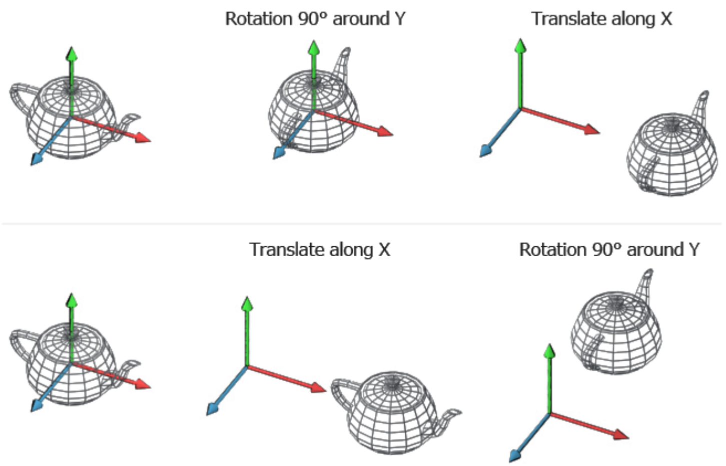
$$R=R_z(\theta_3)R_y(\theta_2)R_x(\theta_1)$$

Rotation Matrix

Rotation | Translation

$$\begin{bmatrix}X'\\ Y'\\ Z'\\ 1\end{bmatrix}=\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\ r_{21} & r_{22} & r_{23} & t_y\\ r_{31} & r_{32} & r_{33} & t_z\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix}$$

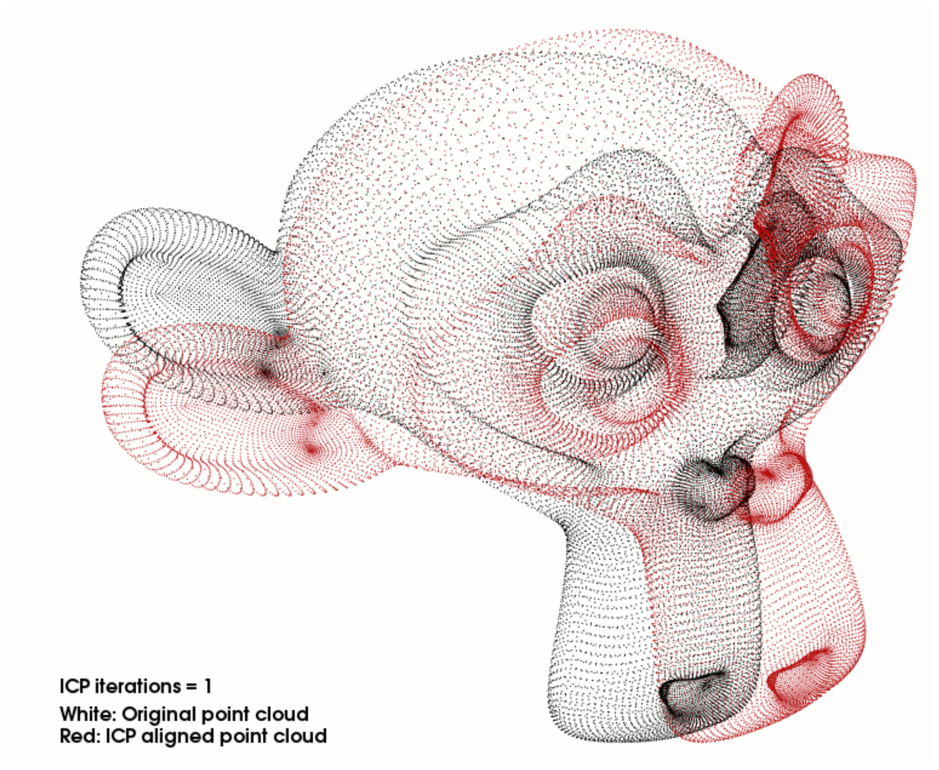
Transformation Matrix

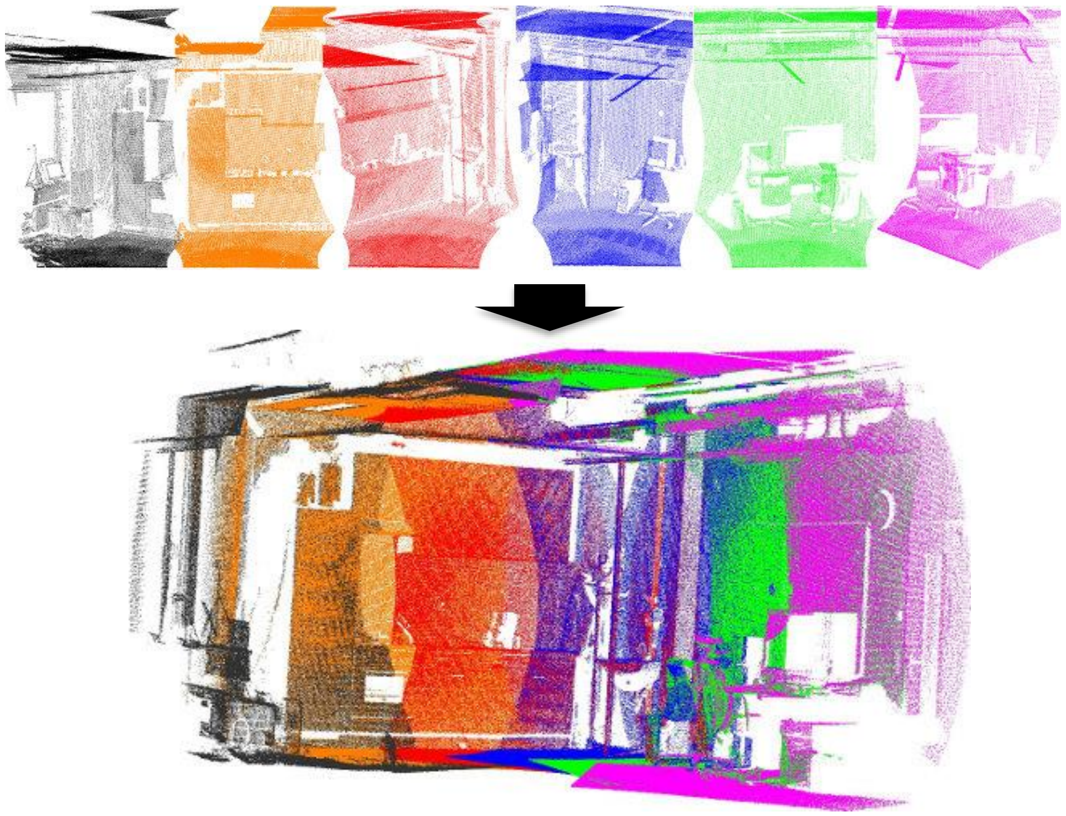


Importance of Transformation Order

Point Cloud Registration

- To find a transformation (rotation R and translation T) that minimizes the distance between two-point clouds.
- Iterative Closest Point (ICP), Normal Distributions Transform (NDT)



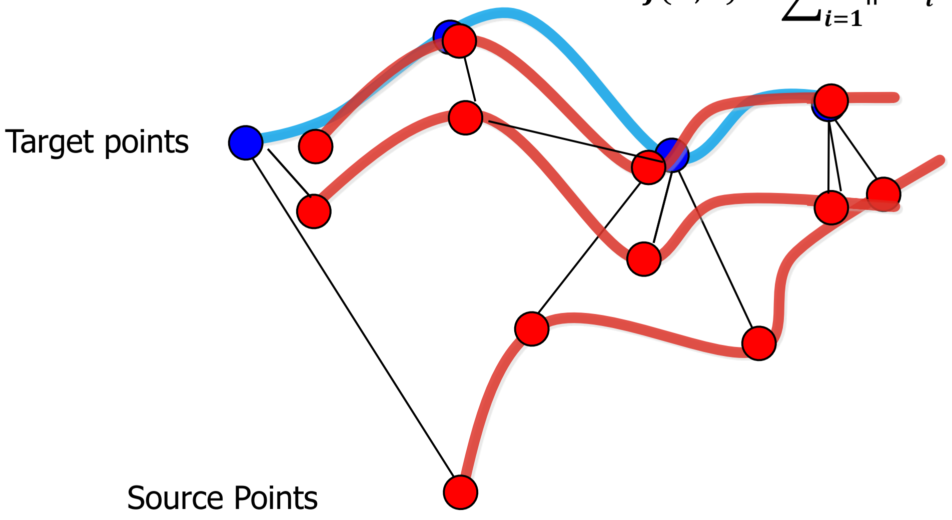


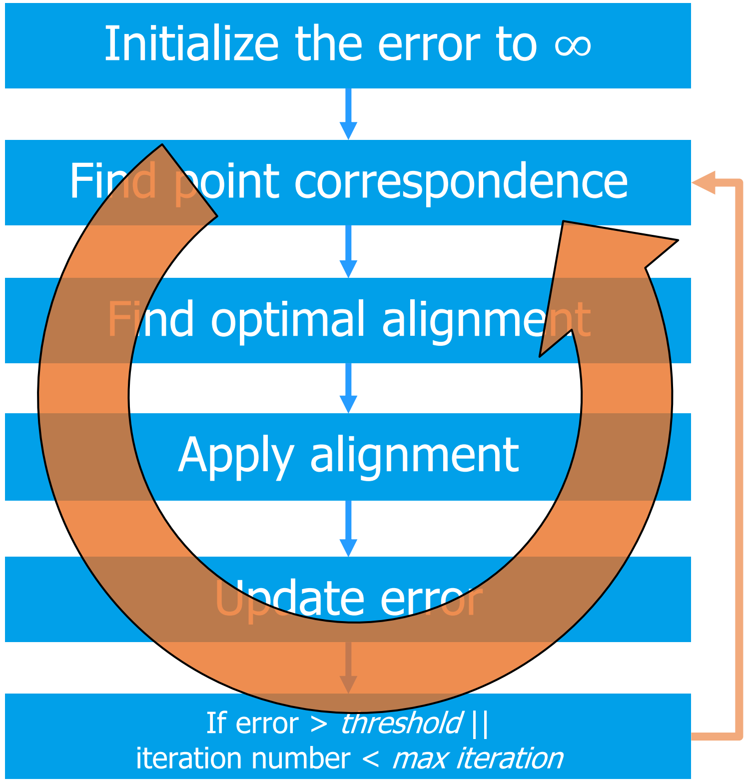
Registration - ICP

ICP(Iterative Closest Point) Algorithm

- An algorithm that minimizes the sum of squared distance between matched points
- Finding the alignment iteratively until it converges
  - ICP performance depends on the point matching method.

$$J(R,T) = \sum_{i=1}^{n} \|RP_i + T - Q_i\|^2$$



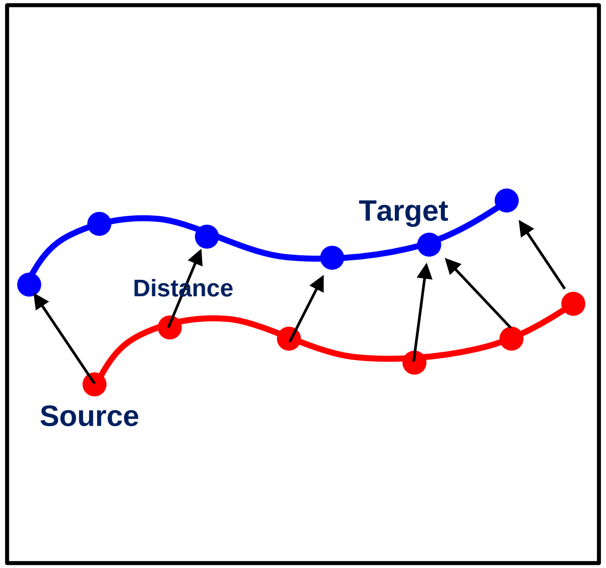


ICP Flow Chart

Registration – Different Type of ICP

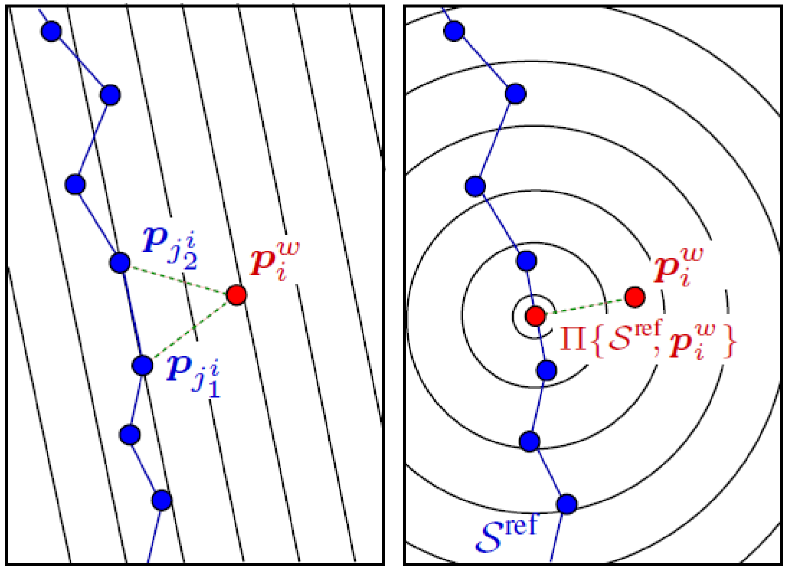
- **The cost function are different respect to error metric for ICP.**
  - Point-to-point metric ICP: Search closes point.
  - Point-to-line metric ICP: Find the closest line by searching two closest points.
  - Point-to-plane metric ICP: Find the plane of the closest point and get the distance between point and plane.

$$\mathbf{T} = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i {d_i^{(\mathbf{T})}}^T d_i^{(\mathbf{T})}$$



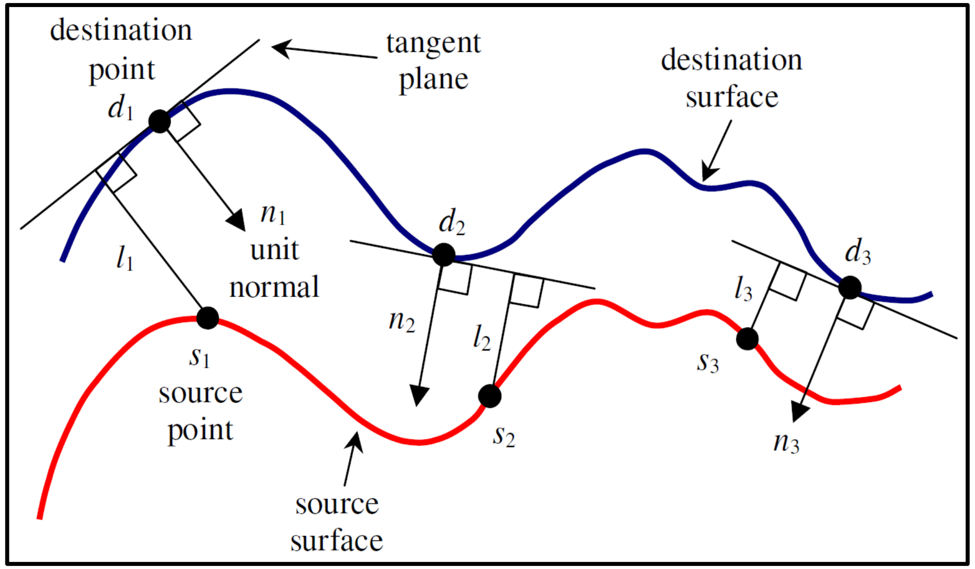
Point-to-point ICP

$$\min_{\boldsymbol{q}_{k+1}} \sum_i \left(\boldsymbol{n}_i^{\mathrm{T}} \left[\boldsymbol{p}_i \oplus \boldsymbol{q}_{k+1} - \Pi\{\mathcal{S}^{\mathrm{ref}}, \boldsymbol{p}_i \oplus \boldsymbol{q}_k\} \right] \right)^2$$



Point-to-line ICP

$$\mathbf{M}_{\mathrm{opt}} = \arg\min_{\mathbf{M}} \sum_i \left((\mathbf{M} \cdot \mathbf{s}_i - \mathbf{d}_i) \bullet \mathbf{n}_i \right)^2$$



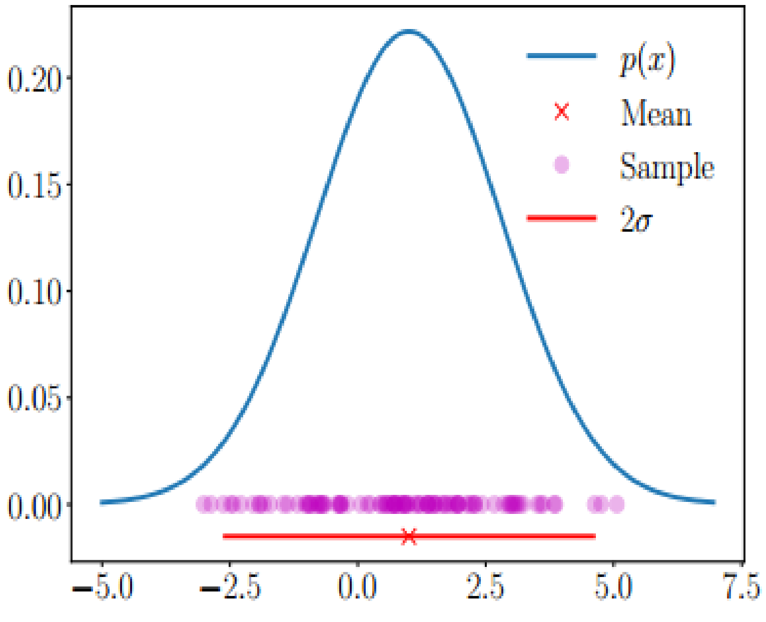
Point-to-plane ICP

Registration – NDT

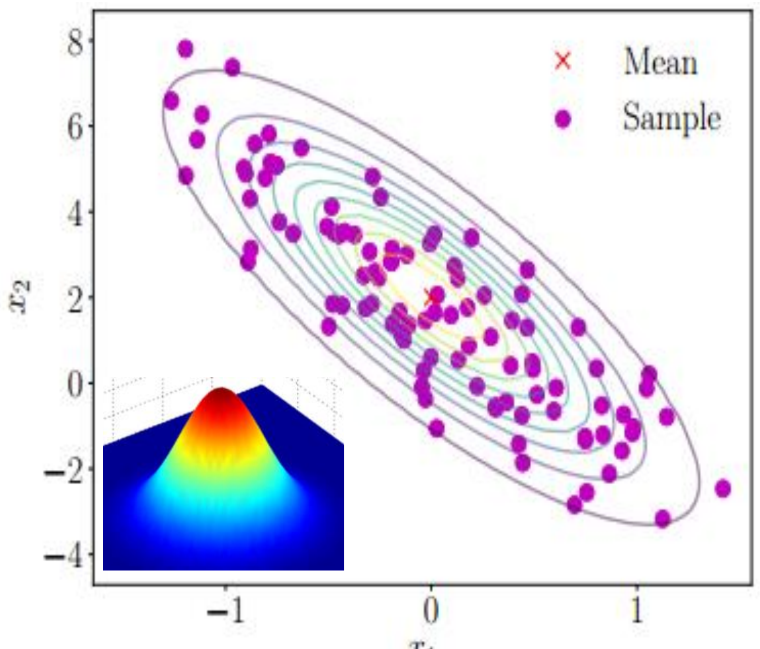
Normal distribution Transformation

- The normal distribution is used to approximate certain data.
- The shape of the normal distribution is determined by mean(μ) and standard deviation(σ).

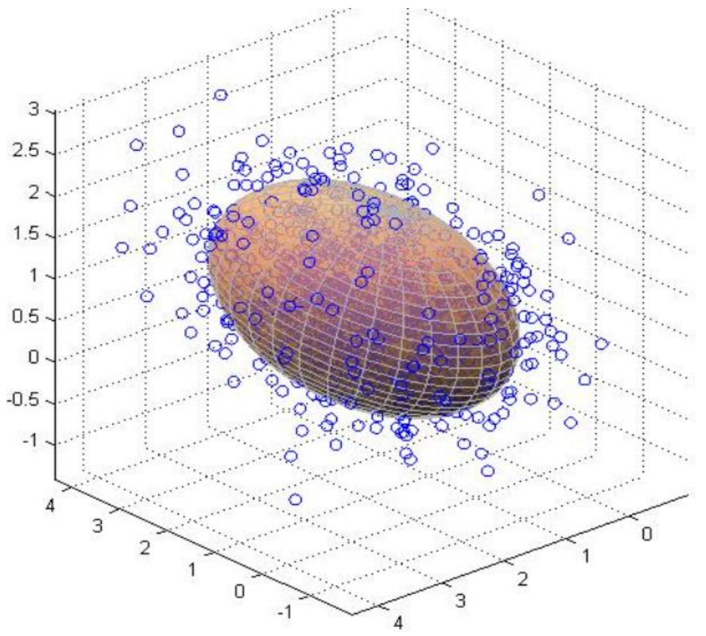
$$X \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$$



1D Normal Distribution of set of Points



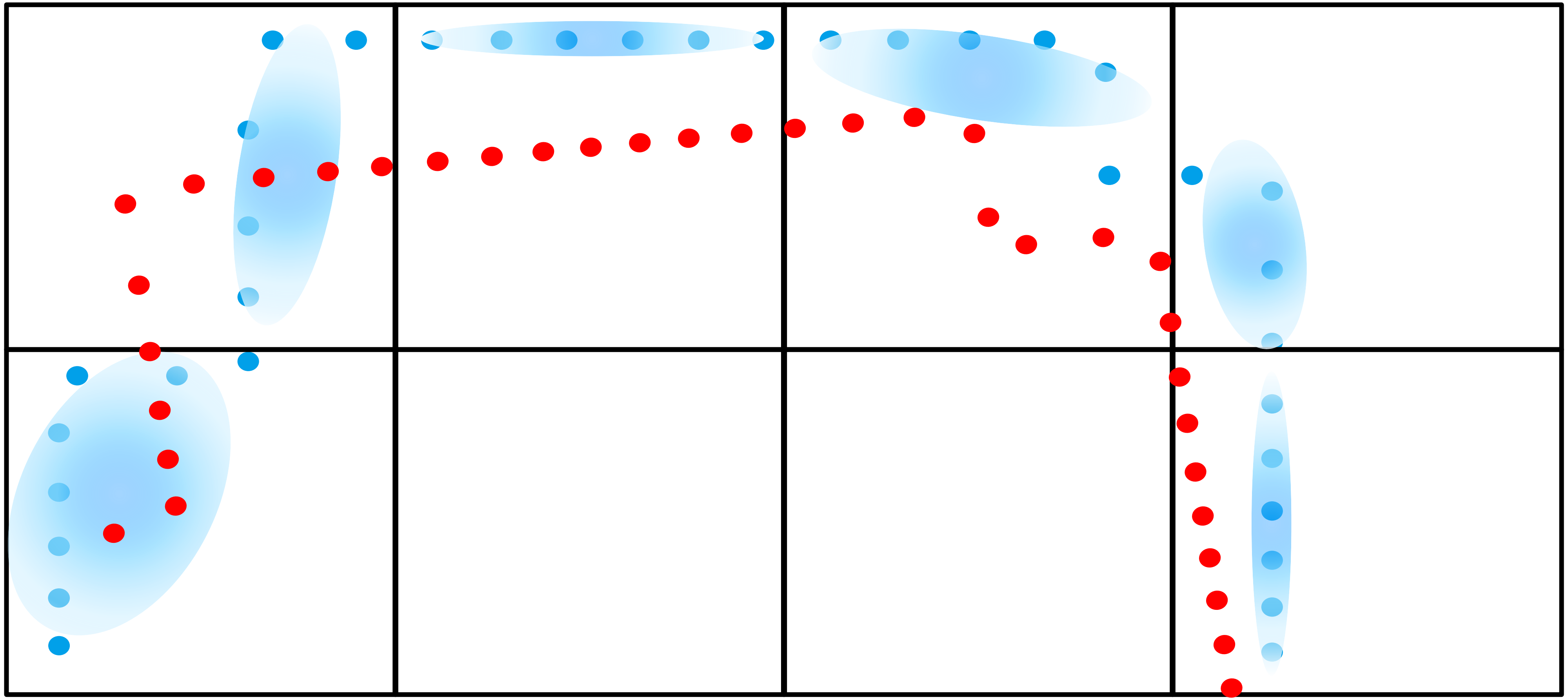
2D Normal Distribution of set of Points



3D Normal Distribution of set of Points

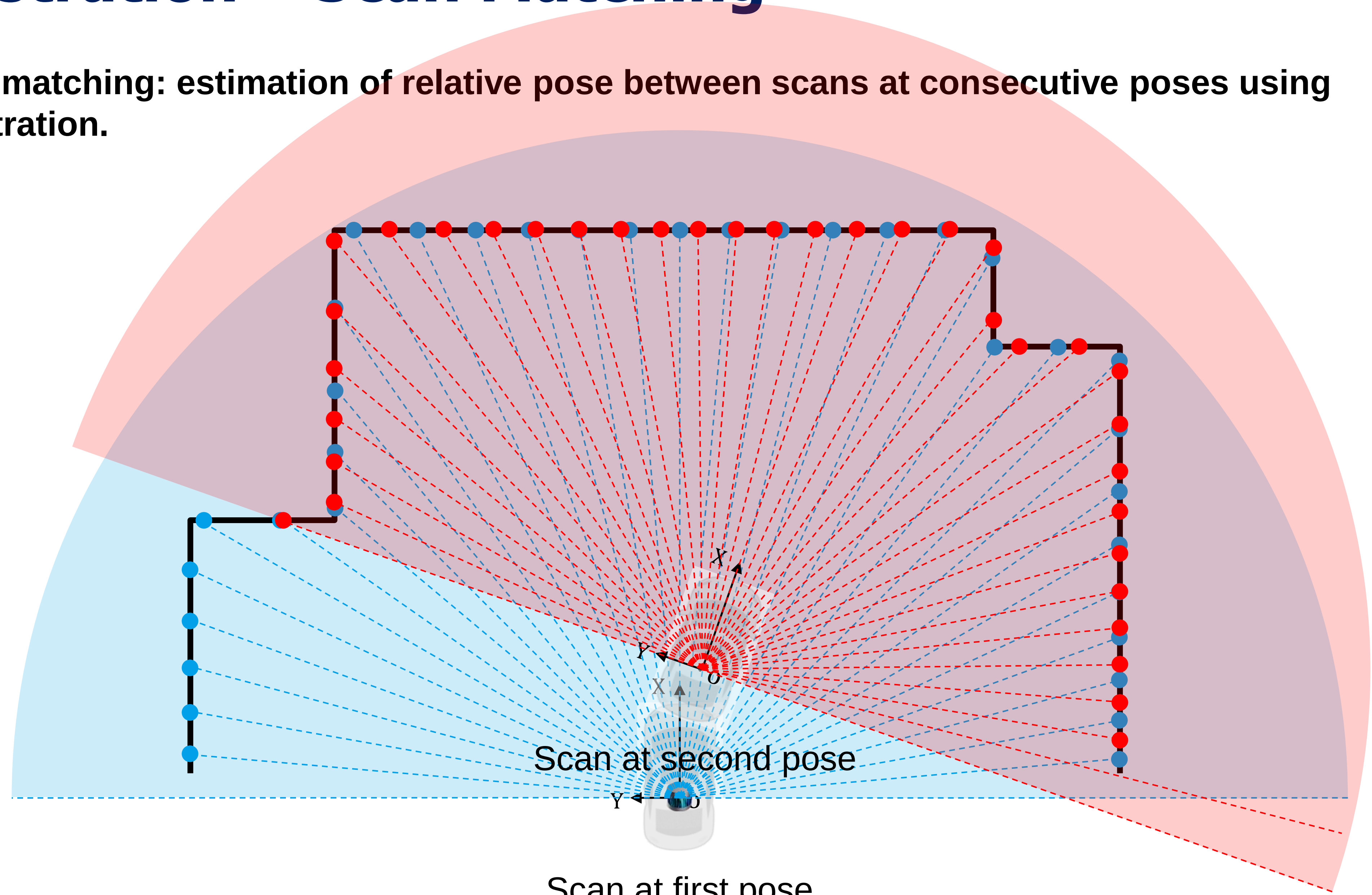
Registration – NDT

The NDT(Normal Distribution Transformation) Algorithm



Registration – Scan Matching

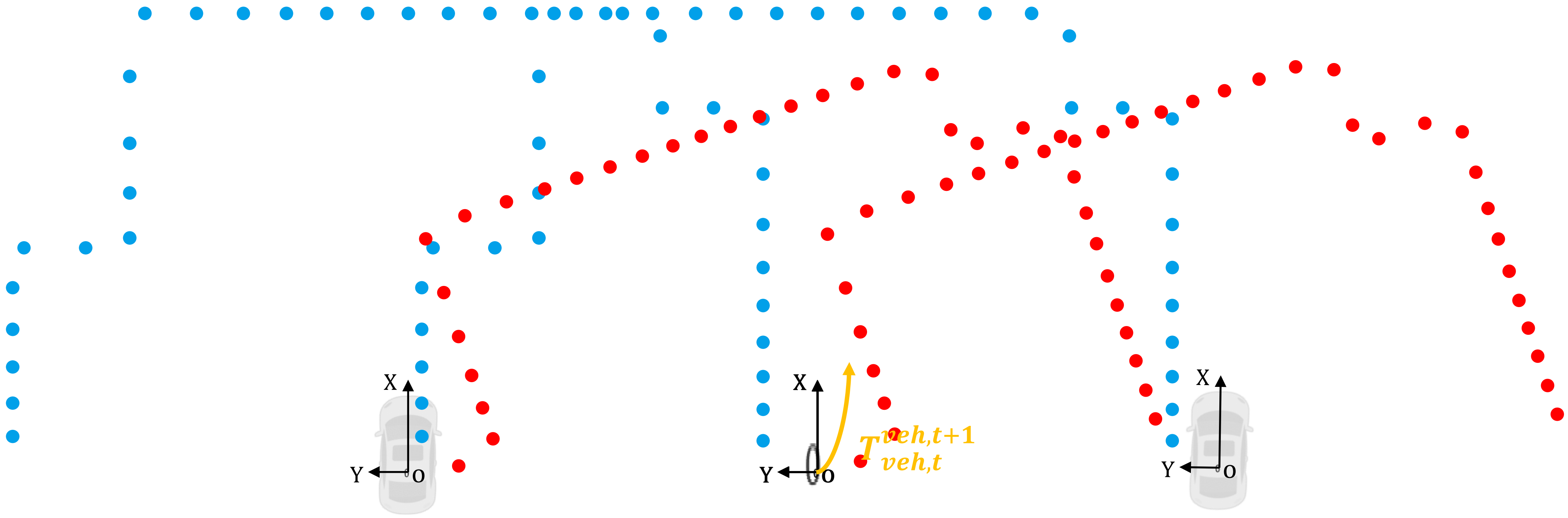
- Scan matching: estimation of relative pose between scans at consecutive poses using registration.



Scan at first pose

Registration – Scan Matching

- **Scan matching: estimation of relative pose between scans at consecutive poses using registration.**



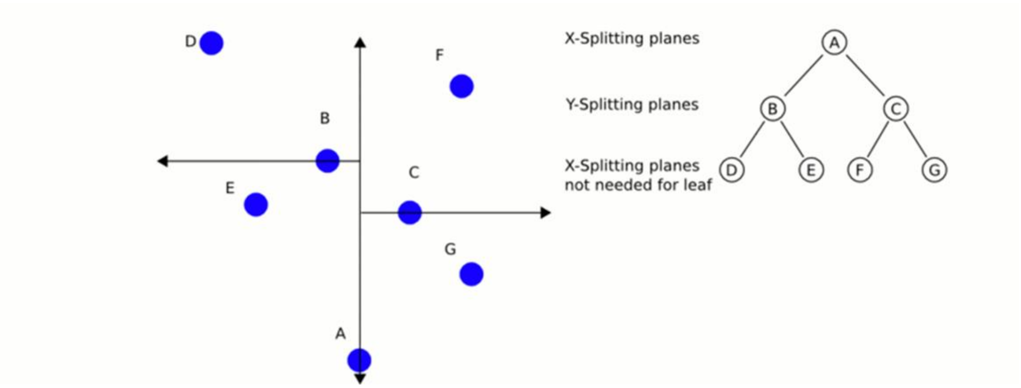
Scan obtained at first pose

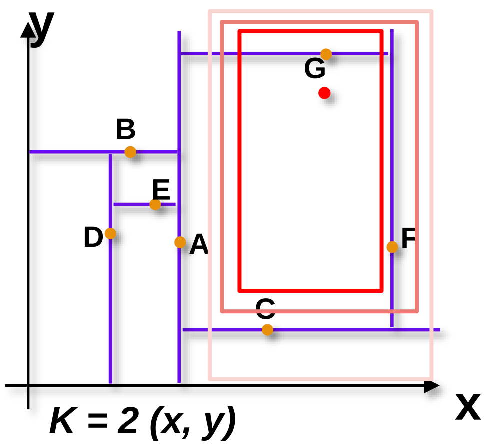
Scans at same coordinate and matching

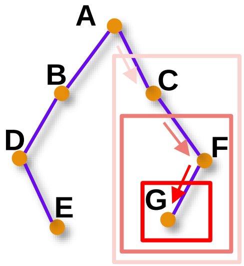
Scan obtained at second pose

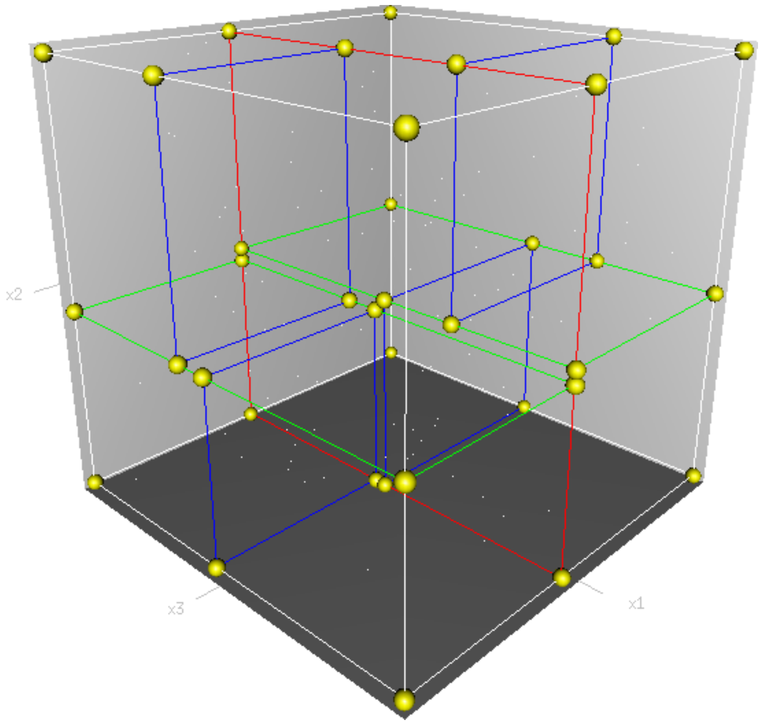
KD Tree

- **Kd-Tree (k-dimensional tree) allows for fast nearest neighbor searches.**
  - **Registration**: KD tree can improve the performance of the registration algorithm by quickly performing nearest neighbor search in this process.
  - **Clustering**: KD trees quickly find the neighbors of each data point, making clustering more efficient.



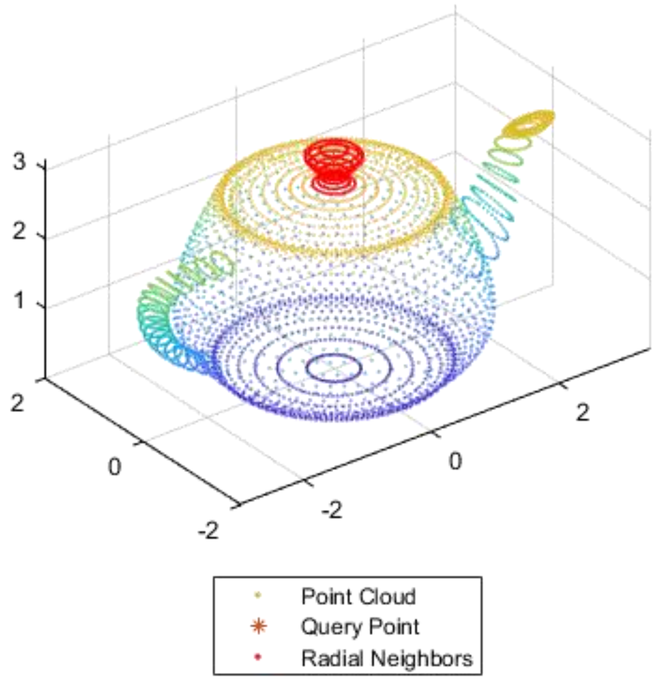




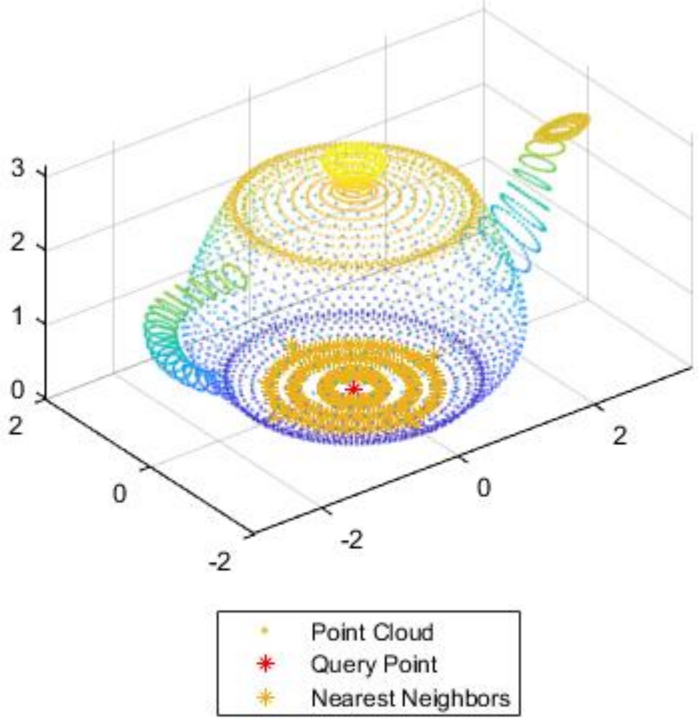


KD Tree-based Search Method

- **findNeighborsInRadius**
  - Function to find neighbors within a radius of a point in the point cloud based on k-d tree
- **findNearestNeighbors**
  - Function to find nearest neighbors of a point in point cloud based on k-d tree



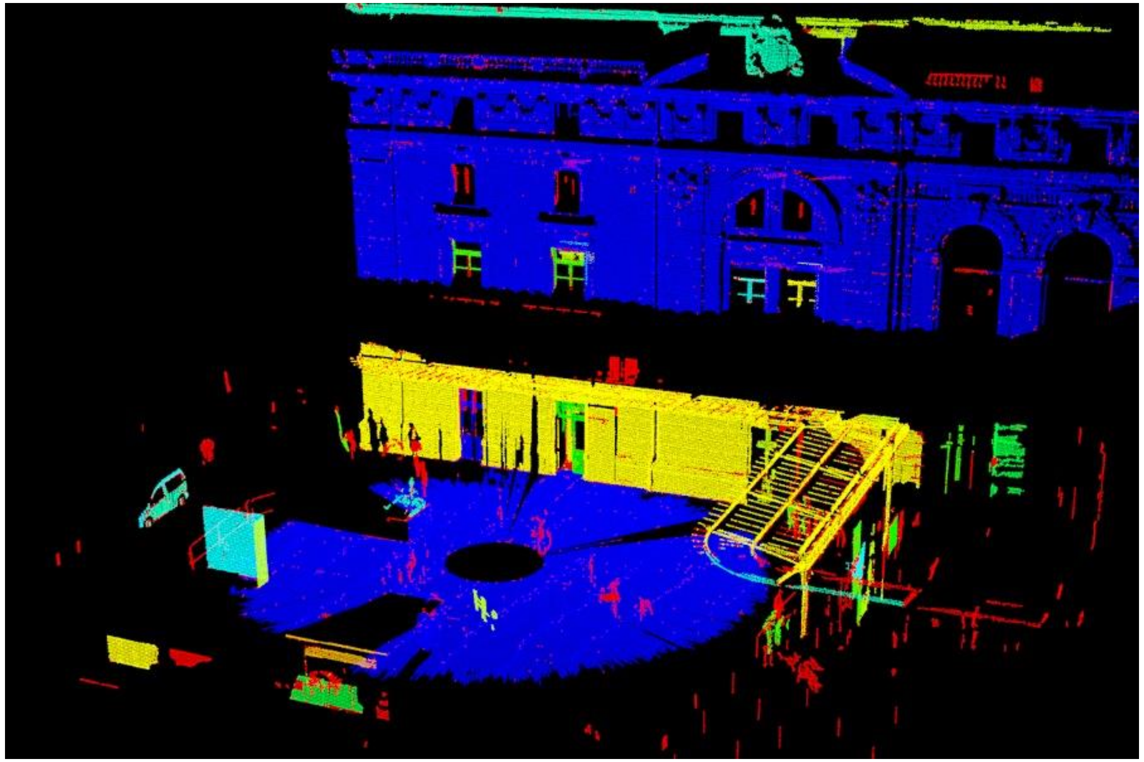
Result of findNeighborsInRadius

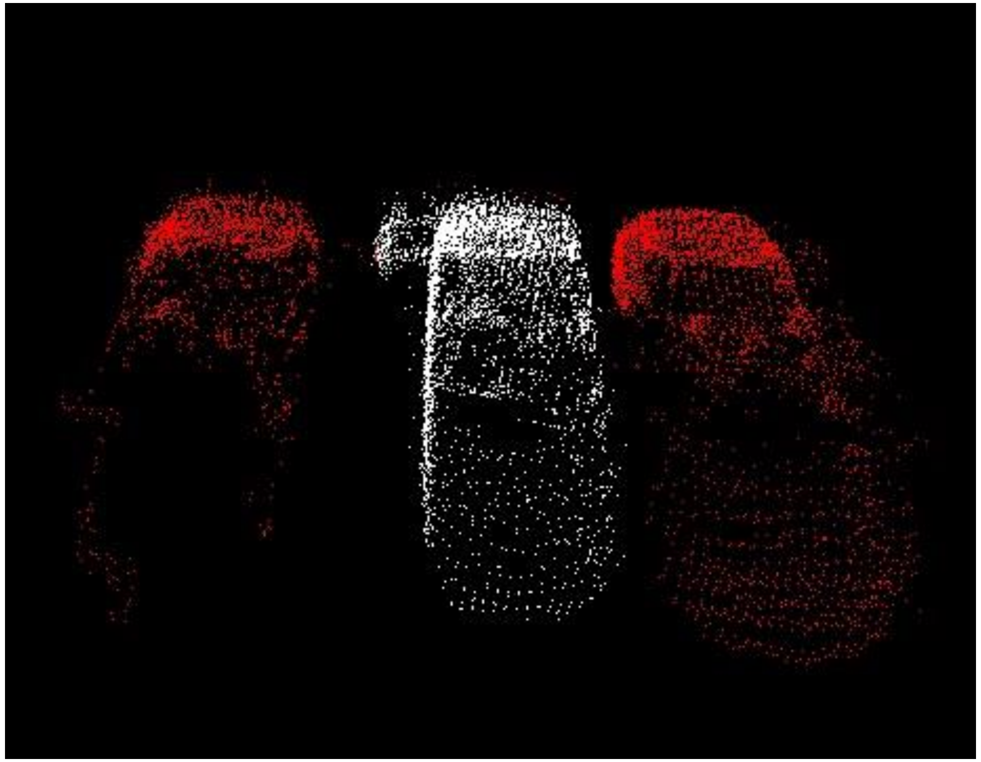


Result of findNearestNeighbors

Segmentation

- **Algorithms for segmenting a point cloud into distinct clusters.**
  - Best suited for processing a point cloud that is composed of several spatially isolated regions.
  - Clustering is often used to break the cloud down into its constituent parts, which can then be processed independently.



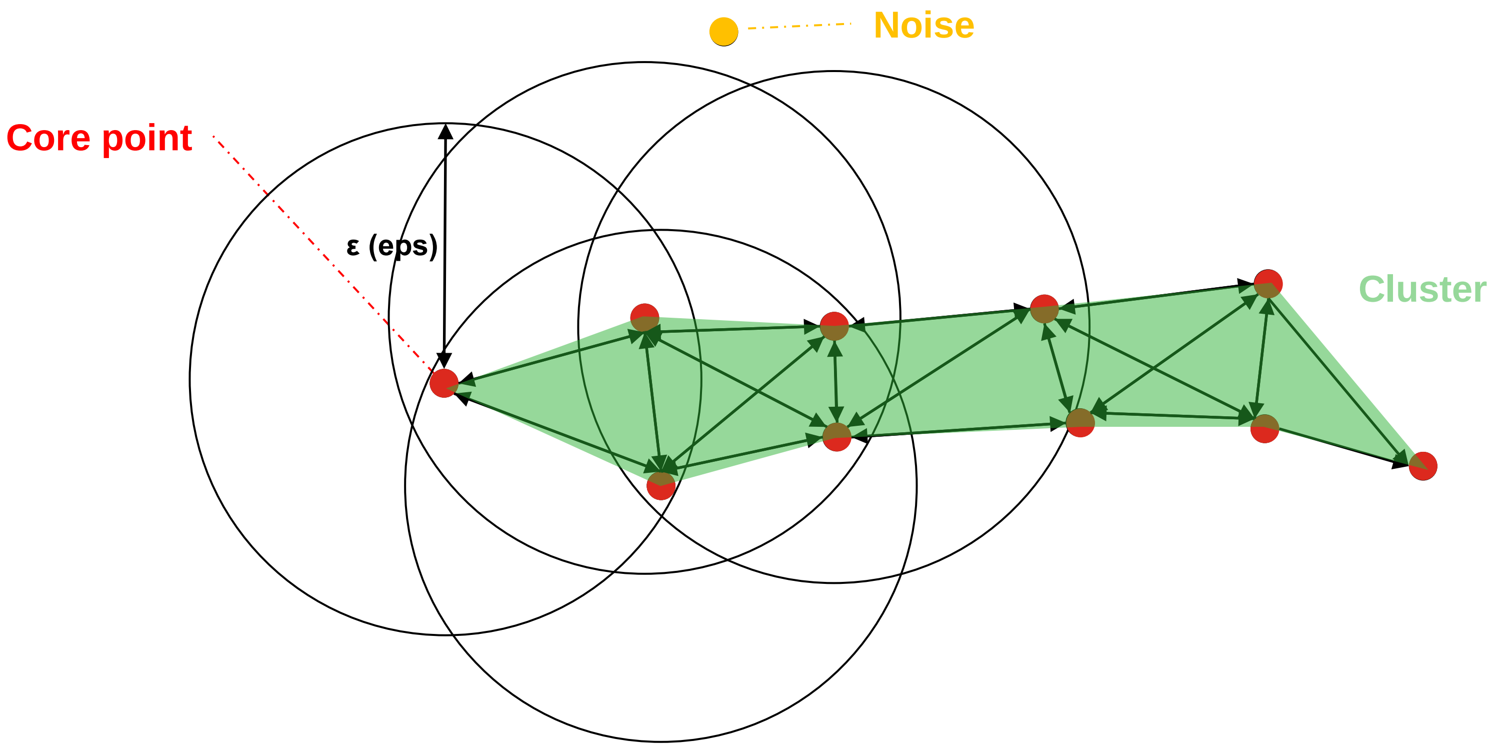


Segmentation

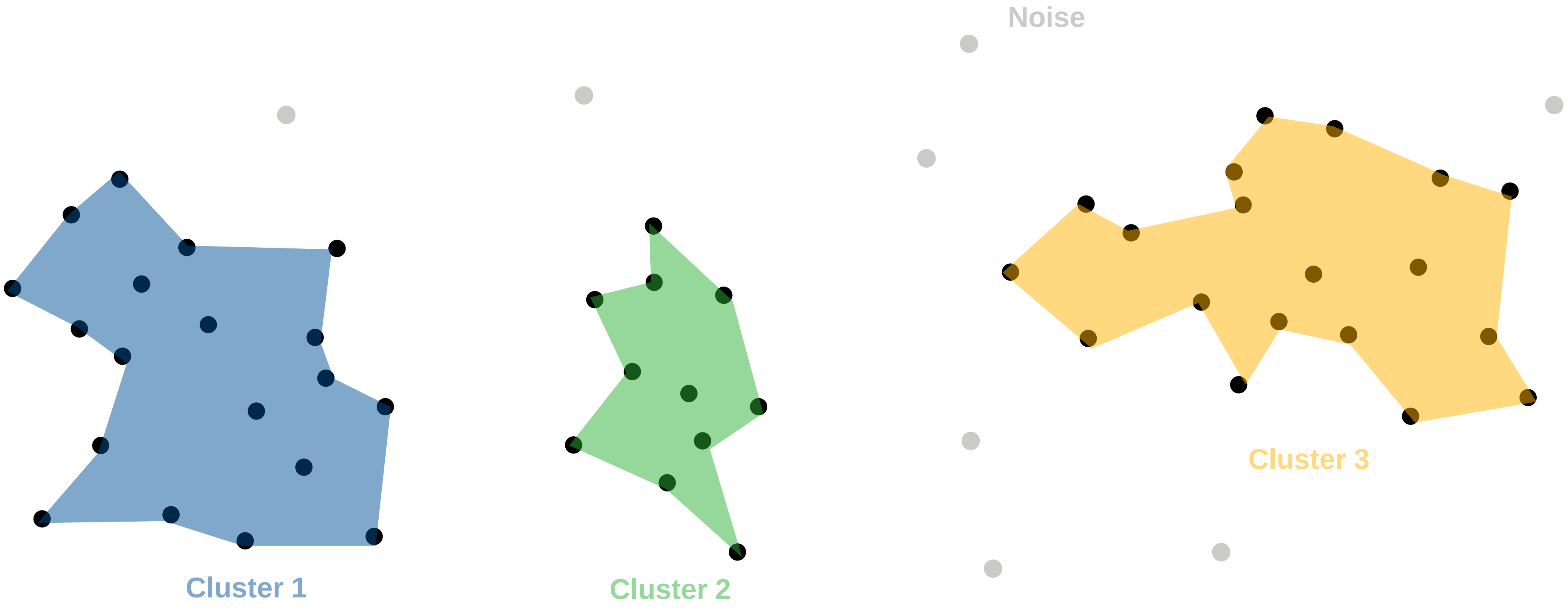
KDTree → DB Scan

Distance-based approach

- ε (eps) required to form a cluster region.
- It starts with an arbitrary starting point that has not been visited.
- This point's ε-neighborhood is retrieved, and if it contains points, a cluster is started.
- Otherwise, the point is labeled as noise.

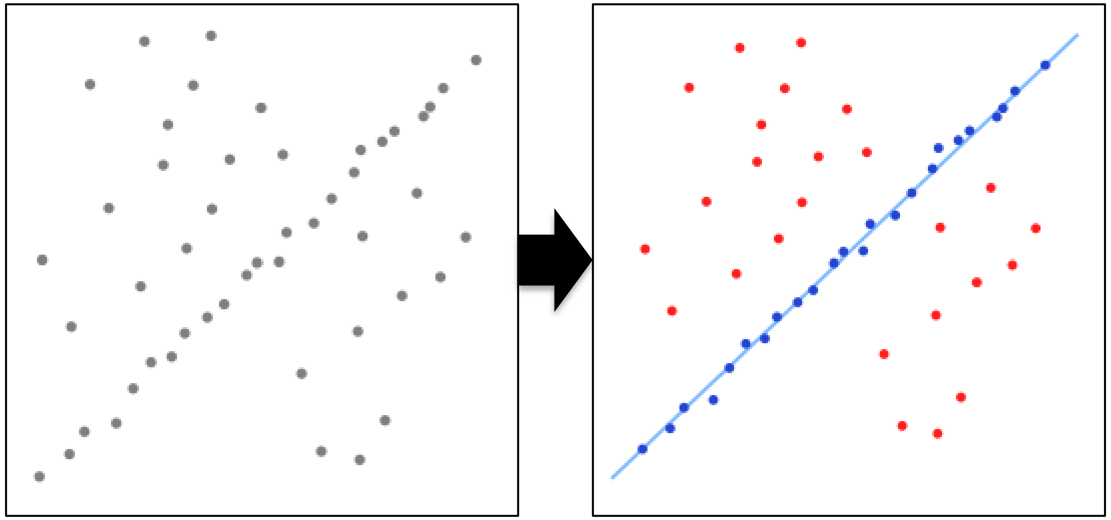


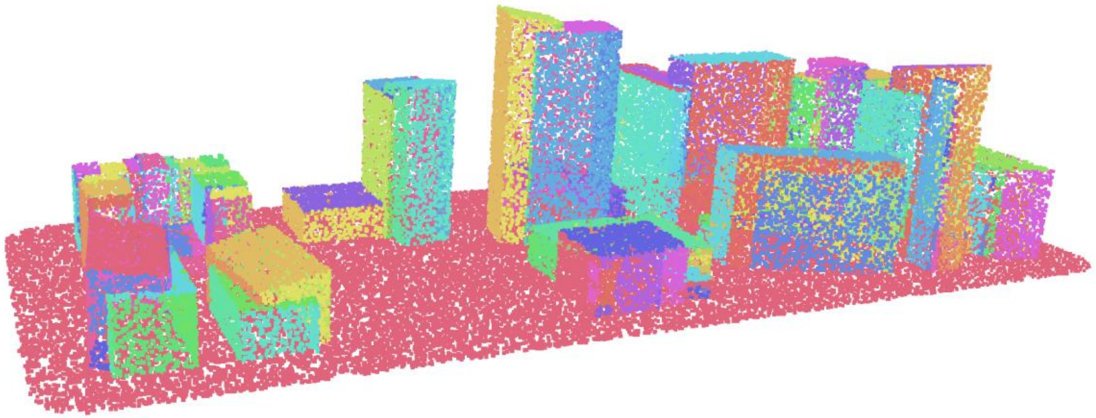
Segmentation



Sample Consensus Algorithm

- **SAmple Consensus (SAC) methods is to detect specific models and their parameters in point clouds.**
  - like RANSAC (RANdom SAmple Consensus)
  - Fitting to lines, planes, cylinders, and spheres.





Plane detection using RANSAC

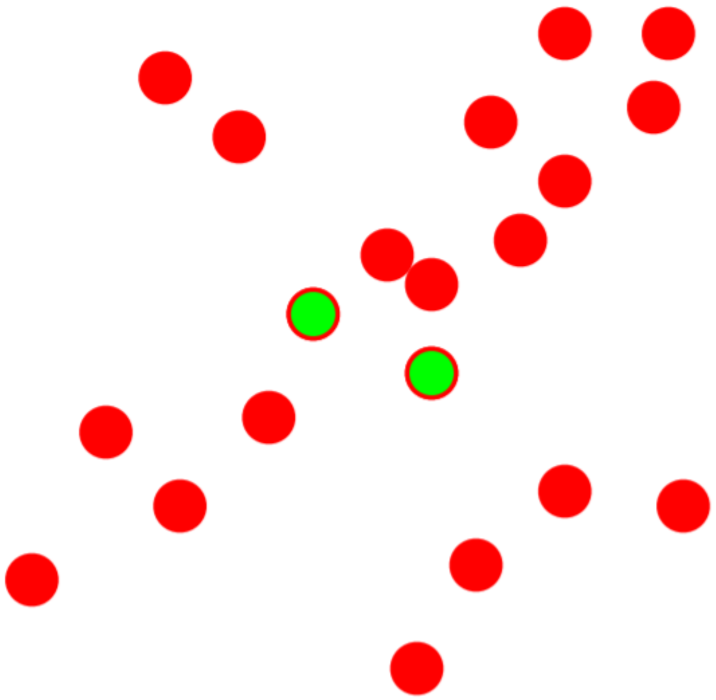
Sample Consensus Algorithm - RANSAC

RANSAC (RANdom SAmple Consensus)

- Using a voting scheme with measurements
  - Repeatedly select sample data randomly and choose **the model with the most consensus, supported by the largest number of data.**

- Example
  - If the model is a straight-line f (x) = ax+b, the number of samples required to determine the model is two or more.

1. Initialize a model score k_{max}.
2. **Randomly select two points p_1, p_2.**
3. Find a f(x), a model equation with two selected points.
4. Calculate the distance between the model f (x) and each point, $r_i = |y_i - f(x_i)|$ (i = 1, 2, ⋯, n) and find the number of points k with $r_i < T$.
5. If k is greater than k_{max}, store the current f(x)
6. Repeat steps 2-5 several(N) times, then return the last saved f(x)



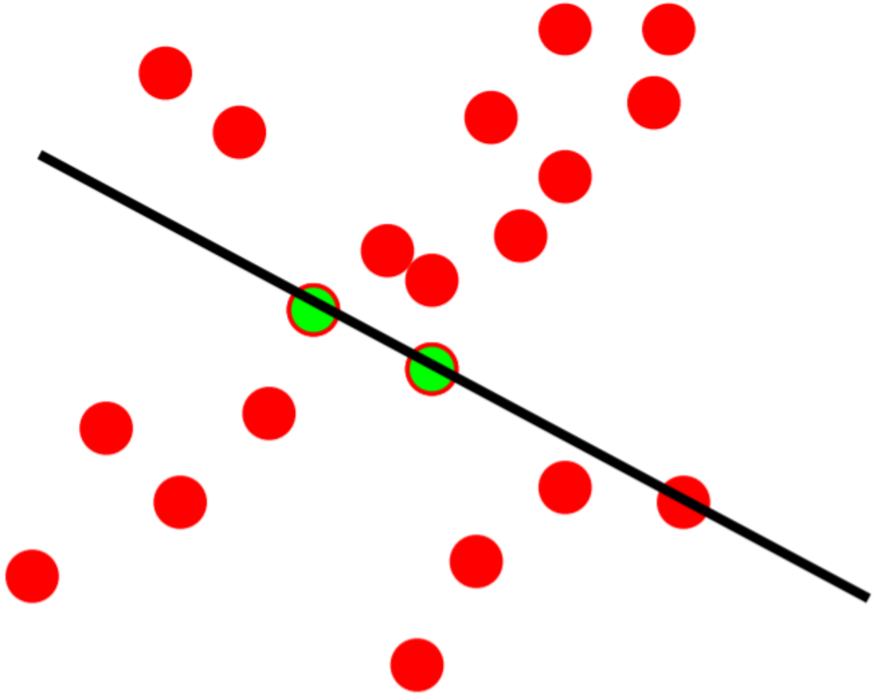
Sample Consensus Algorithm - RANSAC

RANSAC (RANdom SAmple Consensus)

- Using a voting scheme with measurements
  - Repeatedly select sample data randomly and choose **the model with the most consensus, supported by the largest number of data.**

- Example
  - If the model is a straight-line f (x) = ax+b, the number of samples required to determine the model is two or more.

1. Initialize a model score k_{max}.
2. Randomly select two points p_1, p_2.
3. **Find a f(x), a model equation with two selected points.**
4. Calculate the distance between the model f (x) and each point, $r_i = |y_i - f(x_i)|$ (i = 1, 2, ⋯, n) and find the number of points k with $r_i < T$.
5. If k is greater than k_{max}, store the current f(x)
6. Repeat steps 2-5 several(N) times, then return the last saved f(x)

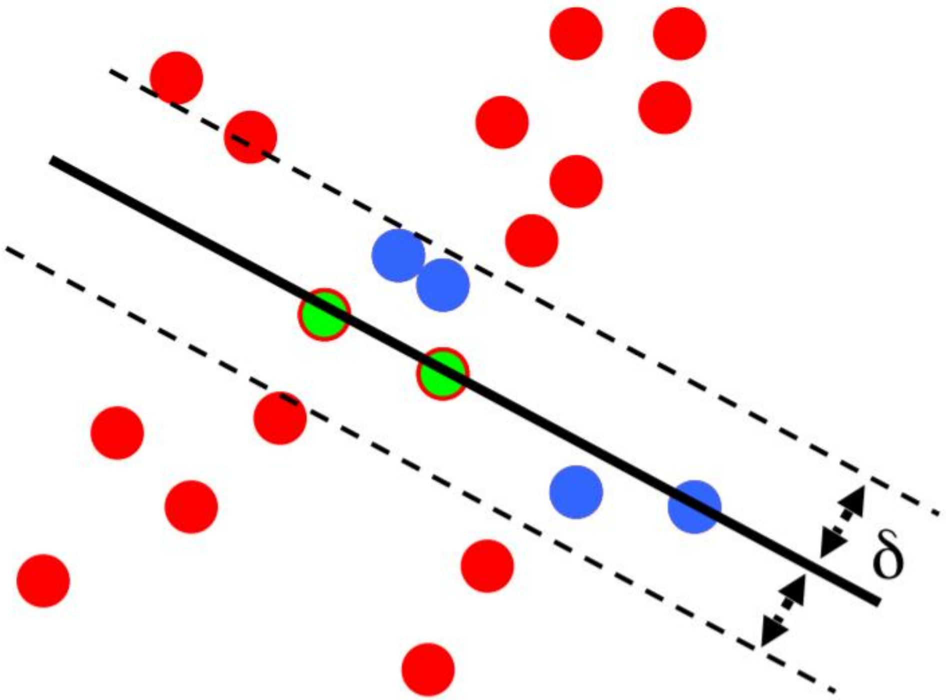


Sample Consensus Algorithm - RANSAC

RANSAC (RANdom SAmple Consensus)

- Using a voting scheme with measurements
  - Repeatedly select sample data randomly and choose **the model with the most consensus, supported by the largest number of data.**
- Example
  - If the model is a straight-line f (x) = ax+b, the number of samples required to determine the model is two or more.

1. Initialize a model score k_{max}.
2. Randomly select two points p_1, p_2.
3. Find a f(x), a model equation with two selected points.
4. **Calculate the distance** between the model f (x) and each point, $r_i = |y_i - f(x_i)|$ (i = 1, 2, ···, n) **and find the number of points k** with $r_i < T$.
5. If k is greater than k_{max}, store the current f(x)
6. Repeat steps 2-5 several(N) times, then return the last saved f(x)



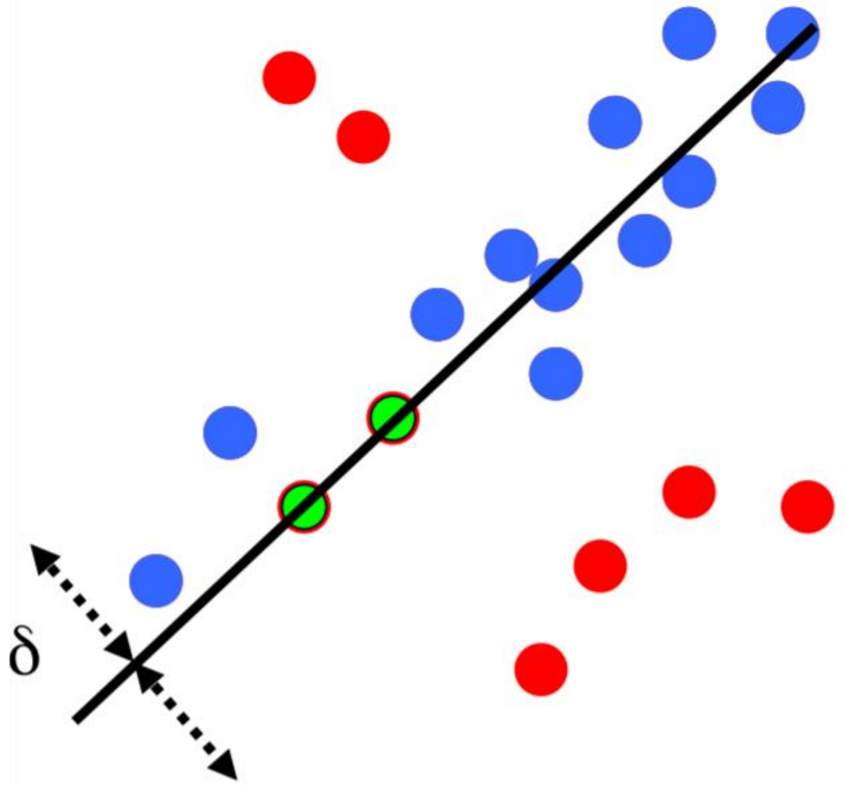
Sample Consensus Algorithm - RANSAC

RANSAC (RANdom SAmple Consensus)

- Using a voting scheme with measurements
  - Repeatedly select sample data randomly and choose **the model with the most consensus, supported by the largest number of data.**

- Example
  - If the model is a straight-line f (x) = ax+b, the number of samples required to determine the model is two or more.

1. Initialize a model score k_{max}.
2. Randomly select two points p_1, p_2.
3. Find a f(x), a model equation with two selected points.
4. Calculate the distance between the model f (x) and each point, $r_i = |y_i - f(x_i)|$ ($i = 1, 2, \cdots, n$) and find the number of points k with $r_i < T$.
5. **If k is greater than k_{max}, store the current f(x)**
6. Repeat steps 2-5 several(N) times, then return the last saved f(x)



Sample Consensus Algorithm - RANSAC

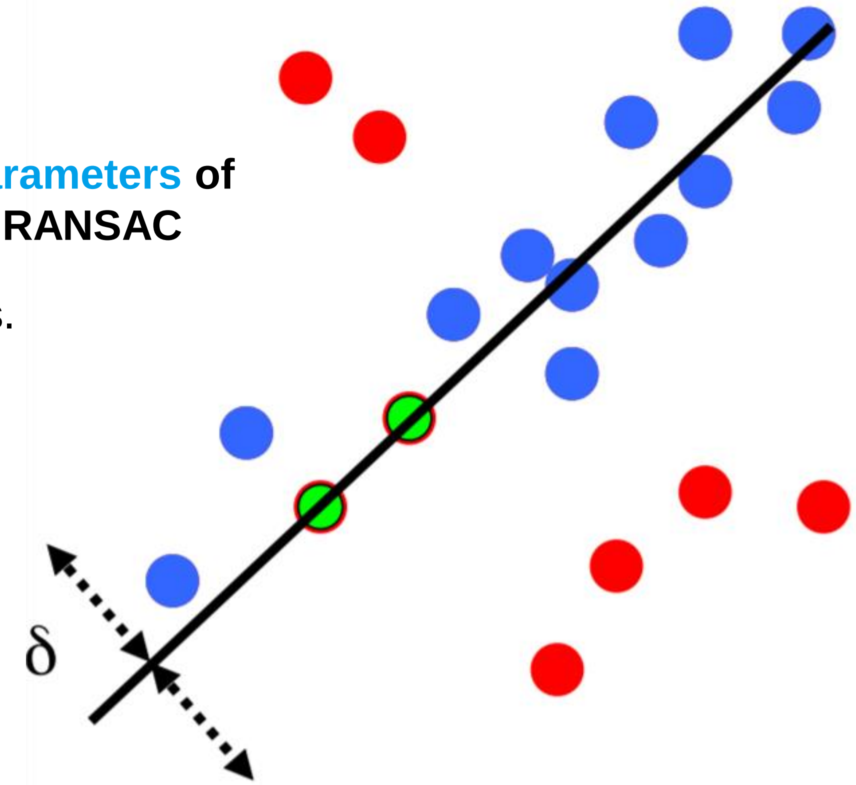
RANSAC (RANdom SAmple Consensus)

- Using a voting scheme with measurements
  - Repeatedly select sample data randomly and choose **the model with the most consensus, supported by the largest number of data.**

- Example
  - If the model is a straight-line f (x) = ax+b, the number of samples required to determine the model is two or more.

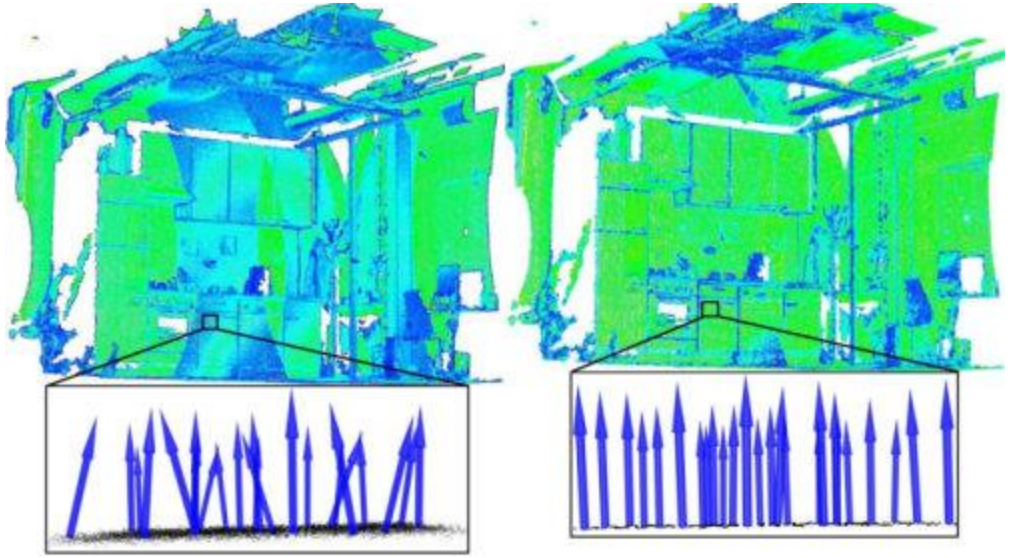
1. Initialize a model score k_{max}.
2. Randomly select two points p_1, p_2.
3. Find a f(x), a model equation with two selected points.
4. Calculate the distance between the model f (x) and each point, $r_i = |y_i - f(x_i)|$ ($i = 1, 2, \cdots, n$) and find the number of points k with $r_i < T$.
5. If k is greater than k_{max}, store the current f(x)
6. Repeat steps 2-5 several(N) times, then return the last saved f(x)

Parameters of RANSAC

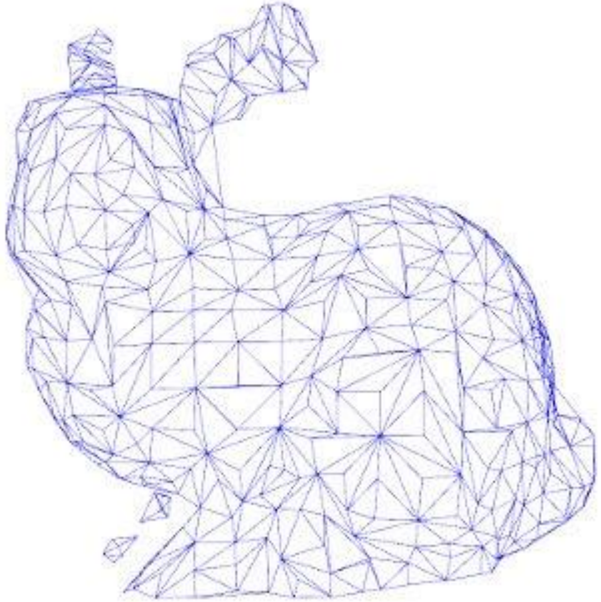


Surface

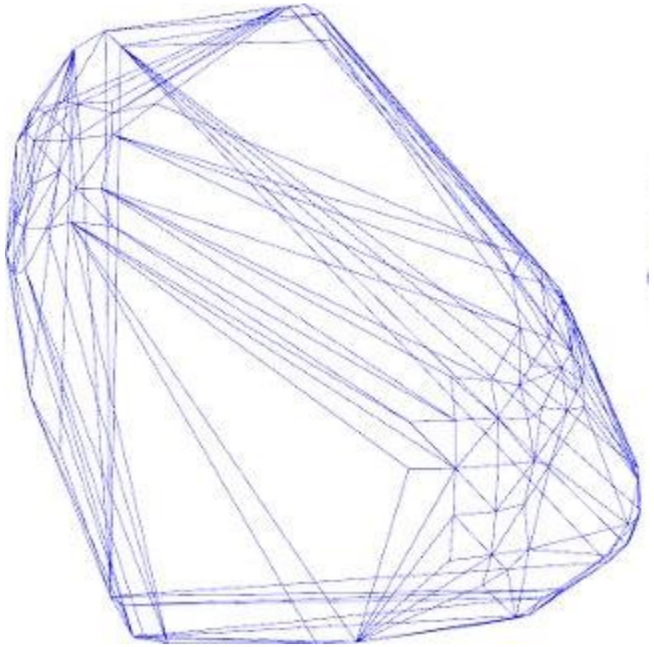
- **Deals with reconstructing the original surfaces from 3D scans**
  - A smoothed/resampled surface with normal
  - A mesh representation
  - Creating convex hull



Smoothed / Resampled Surface with Normal



Mesh Representation



Creating Convex Hull

THANK YOU
FOR YOUR ATTENTION