DTU

Lazaros Nalpantidis

3D Point Cloud Processing - Pose Estimation

Outline

- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?

- Point Cloud Registration
  - Local Alignment
    - Iterative Closest Point (ICP) algorithm
  - Global Alignment
    - 3D Feature Descriptors
      - Spin Images
      - PFH
      - FPFH
    - Random Sample Consensus (RANSAC) in 3D

- Summary

Outline

- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?

- Point Cloud Registration
  - Local Alignment
    - Iterative Closest Point (ICP) algorithm
  - Global Alignment
    - 3D Feature Descriptors
      - Spin Images
      - PFH
      - FPFH
    - Random Sample Consensus (RANSAC) in 3D

- Summary

Why do we need 3D Point Clouds?

Why do we need 3D Point Clouds?

- World is 3D
  - Objects are not entirely described by 2D images
  - 3D geometry and shape are often important

- *However,*
  - *– operations in 3D are computationally heavy*
  - *– SIFT/SURF/… do not work in point clouds*

Bonus:
https://github.com/dataarts/radiohead

What is “Pose”?

What is “Pose”?

- Pose
  - the transformation (translation + rotation) needed to map one point cloud (model) to another point cloud (model) of the same (fully or partially) object or scene.

- …or equivalently
  - “…determining a camera’s position relative to a known 3D object or scene…” (Szelinski).

Why is 3D Pose Estimation Important?

Why is 3D Pose Estimation Important?

- Many applications in Autonomous Systems
  - Robot Grasping/Manipulation
  - Quality Inspection
  - Augmented reality
  - Progressive map building

Outline

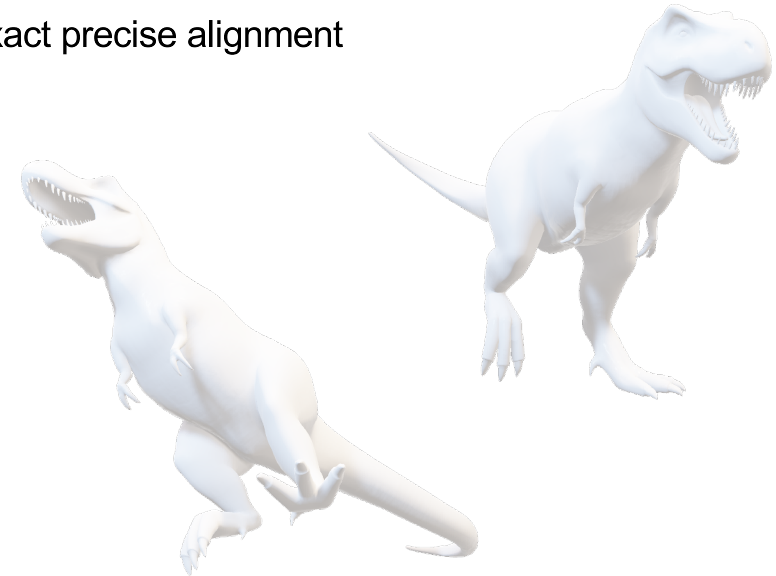
- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?

- <u>Point Cloud Registration</u>
  - – Local Alignment
    - Iterative Closest Point (ICP) algorithm
  - – Global Alignment
    - 3D Feature Descriptors
      - – Spin Images
      - – PFH
      - – FPFH
    - Random Sample Consensus (RANSAC) in 3D

- Summary

Point Cloud Registration

- Generally 2 steps are needed to register 2 given point clouds of the same object (fully or partially)
  - Global alignment
    - The 2 models are ”roughly aligned”
  - Local alignment
    - Starting from a ”rough” initial alignment, find the exact precise alignment



Outline

- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?

- Point Cloud Registration
  - <u>Local Alignment</u>
    - Iterative Closest Point (ICP) algorithm
  - Global Alignment
    - 3D Feature Descriptors
      - Spin Images
      - PFH
      - FPFH
    - Random Sample Consensus (RANSAC) in 3D

- Summary

Local Alignment

- Consider 2 models (point clouds) of the same object (fully or partially) that are almost aligned.
  - What is the difference of their pose?
    …or equivalently…
  - What is the transformation that can fully align them?

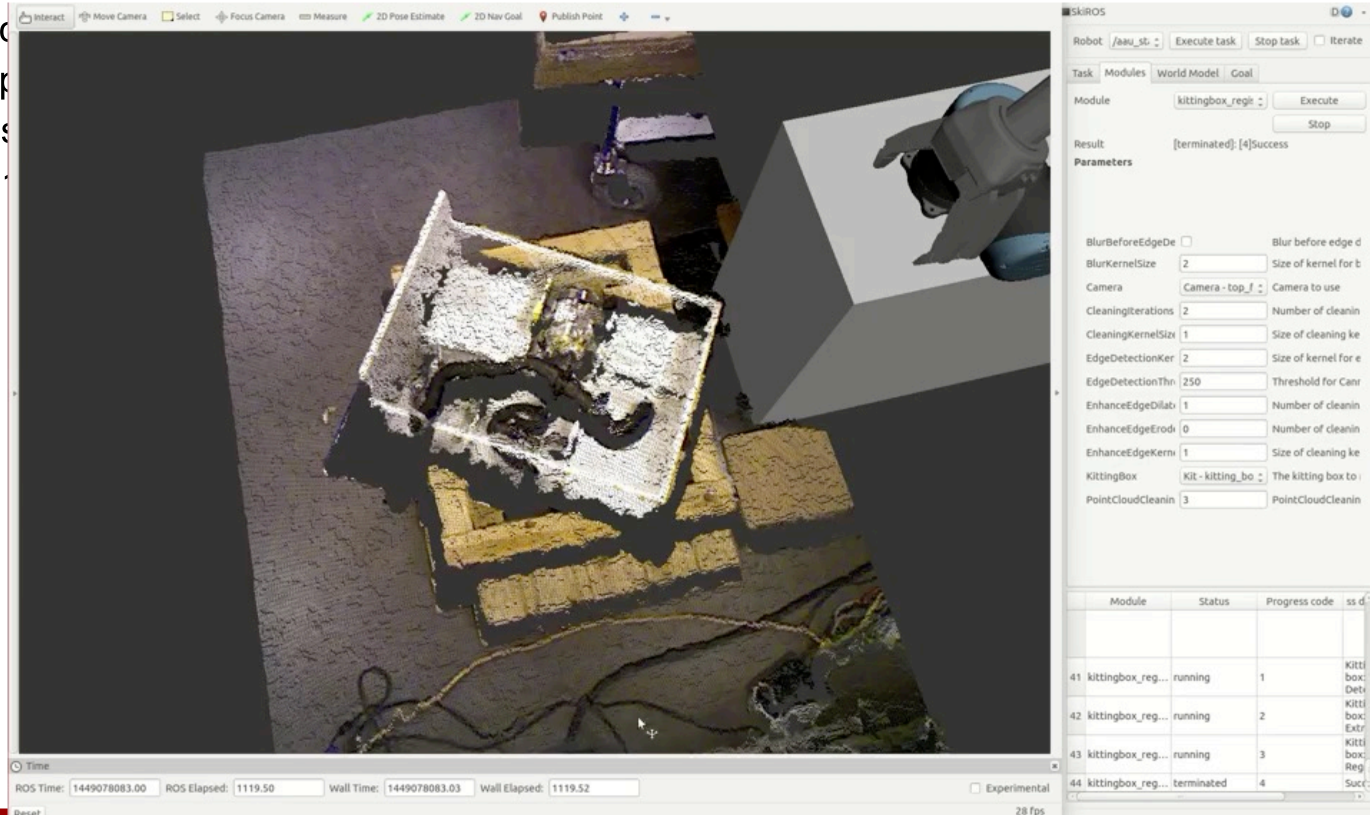
Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each object point *p* in model 1, find the nearest point *q* in model 2
  2. Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, Feb. 1992.

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each objec
  2. Use all pairs (p
  3. Apply the trans
  4. Repeat steps



Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each object point *p* in model 1, find the nearest point *q* in model 2
  2. Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. **For each object point *p* in model 1, find the nearest point *q* in model 2**
  2. Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. **For each object point *p* in model 1, find the nearest point *q* in model 2**
  2. Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

– How to compare all possible pairs for proximity, in an efficient manner?
  – If there are X points in model 1 and Y points in model 2, then XY distance calculations are required
  – However, we can significantly speed up this search by using k-d trees!

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each object point *p* in model 1, find the nearest point *q* in model 2
  2. **Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2**
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each object point *p* in model 1, find the nearest point *q* in model 2
  2. **Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2**
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

– How to estimate the transformation from model 1 to model 2, given pairs *(p,q)* ?

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each object point p in model 1, find the nearest point q in model 2
  2. **Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2**
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

– How to estimate the transformation from model 1 to model 2, given pairs *(p,q)* ?
– Kabsch Algorithm / Procrustes Analysis:
  – Translate the centroids of both models to the origin of the coordinate system (0,0,0).
    » Compute centroids c_p , c_q of both models
    » Subtract from each point coordinates the coordinates of its corresponding centroid:
    $p' = p - c_p$ and $q' = q - c_c$
  – Compute Covariance Matrix: $C_{pq} = \Sigma p' q'^T$
  – Compute the optimal rotation
    » Calculate the Singular Value Decomposition (SVD) of the covariance matrix: $C_{pq} = USV^T$
    » Calculate rotation matrix: $R = UV^T$
  – Compute the optimal translation: $T = c_q - Rc_p$

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each object point *p* in model 1, find the nearest point *q* in model 2
  2. **Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2**
  3. Apply the transformation to the points of Model 1
  4. Repeat steps 1-3 until convergence/stop criterion is met

  – How to estimate the transformation from model 1 to model 2, given pairs *(p,q)* ?
  – Kabsch Algorithm / Procrustes Analysis

  – QUESTION:
    - Why ICP needs to employ Kabsch algorithm at every iteration?

Local Alignment - ICP

- Iterative Closest Point (ICP)
  1. For each object point *p* in model 1, find the nearest point *q* in model 2
  2. Use all pairs *(p,q)* to estimate the transformation from model 1 to model 2
  3. **Apply the transformation to the points of Model 1**
  4. **Repeat steps 1-3 until convergence/stop criterion is met**

Outline

- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?

- Point Cloud Registration
  - Local Alignment
    - Iterative Closest Point (ICP) algorithm
  - <u>Global Alignment</u>
    - 3D Feature Descriptors
      - Spin Images
      - PFH
      - FPFH
    - Random Sample Consensus (RANSAC) in 3D

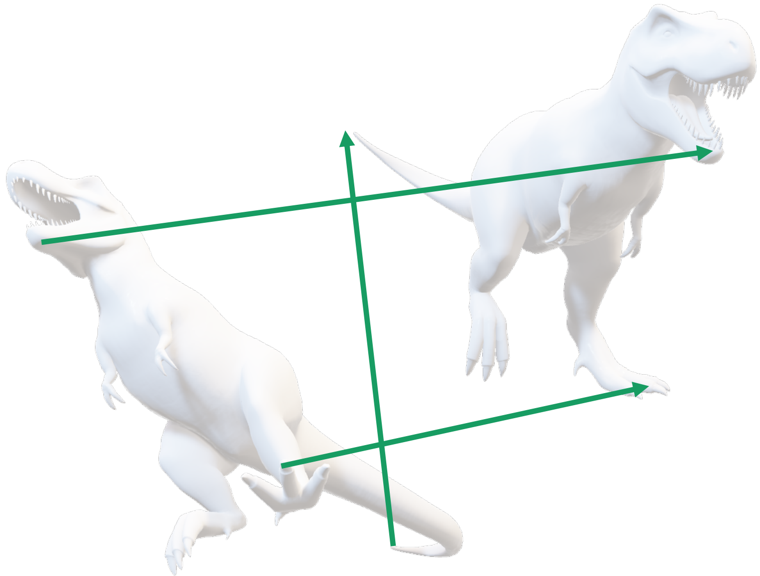
- Summary

Global Alignment

- Consider 2 models (point clouds) of the same object (fully or partially)
  – How can we “roughly” align them?

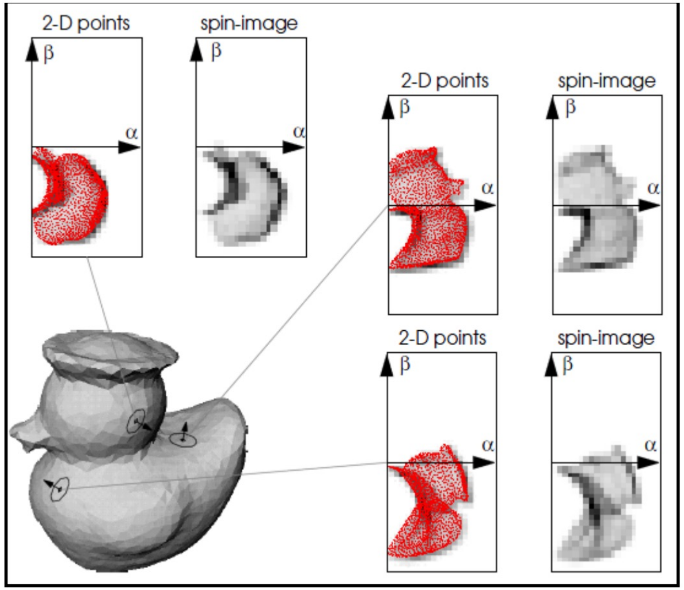
Global Alignment

- Consider 2 models (point clouds) of the same object (fully or partially)
  - How can we "roughly" align them?
    - We cannot use ICP, if the initial poses are too different.
    - Can we use some kind of (3D) "features" and match them?



Global Alignment – 3D Feature Descriptors

- **Spin Images**
  - "Spin" a discretized 2D grid around the surface normal of a point.
  - Accumulate neighboring pixels in the grid bins, while spinning.
  - The descriptor is the 2D grid (image) where each element (pixel) contains the number of accumulated points.



A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 5, pp. 433-449, May 1999.

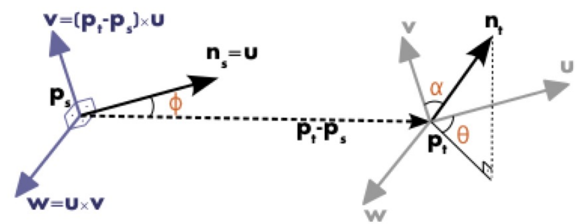
Global Alignment – 3D Feature Descriptors

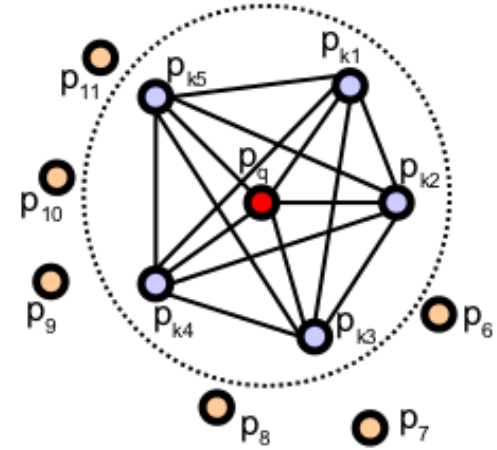
- **PFH (Point Feature Histograms)**

$$\alpha = \arccos\left(v \cdot n_t\right)$$

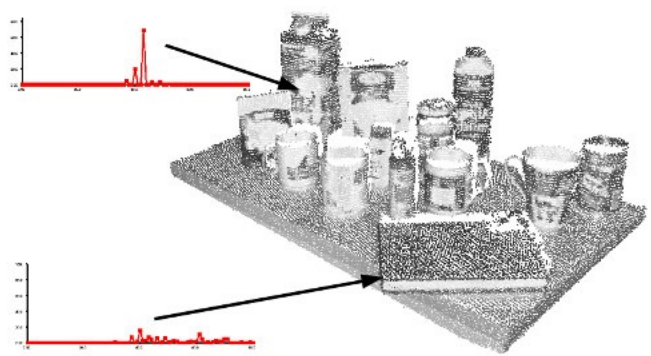
$$\phi = \arccos\left(u \cdot \frac{\left(p_t - p_s\right)}{\left\|p_t - p_s\right\|_2}\right)$$

$$\theta = \arctan\left(w \cdot n_t, u \cdot n_t\right)$$





- Calculate these 3 values for all pairs of points within a radius *r* from the considered point (maybe also their distance *d*)
- The set of all triplets/quadruplets are binned in a histogram – This is the multi-dimensional descriptor



R. B. Rusu, N. Blodow and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," 2009 IEEE International Conference on Robotics and Automation, Kobe, 2009, pp. 3212-3217.

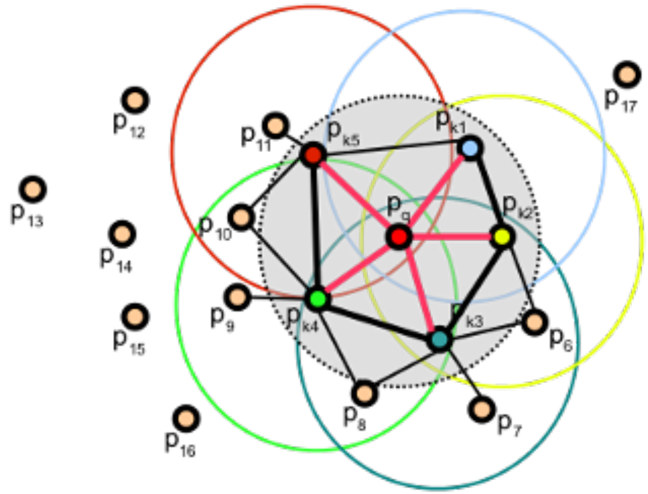
R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning Point Cloud Views using Persistent Feature Histograms,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26, 2008.

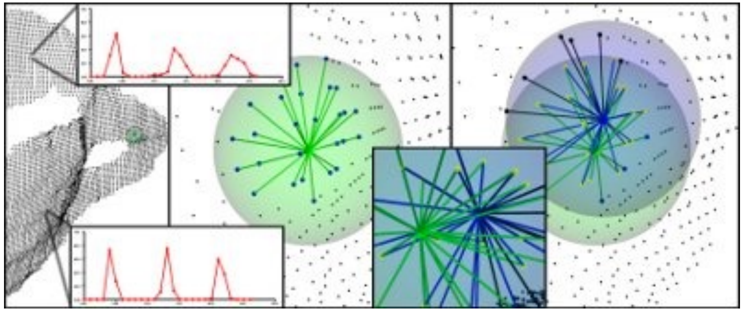
Global Alignment – 3D Feature Descriptors

- **FPFH (Fast Point Feature Histograms)**
  – Simplification/Approximation of the PFH formulation
    - reduces the computational complexity
    - retains most of the discriminative power of PFH.

  – Algorithm:
    - Find all oriented points in a spherical neighborhood of radius *r* around each point (k-d tree)
    - Compute relative angles using surface normals and direction vector from the source point to each neighbor

  – The descriptor is a multi-dimensional histogram



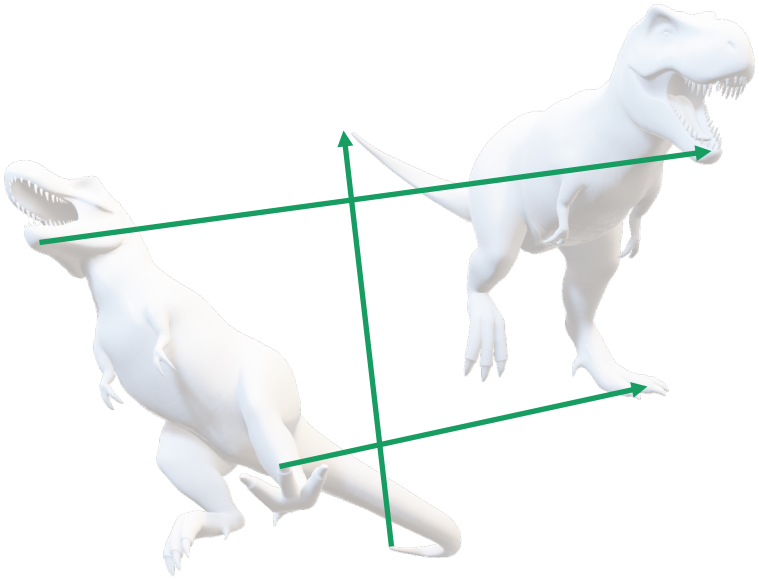


R. B. Rusu, N. Blodow and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," 2009 IEEE International Conference on Robotics and Automation, Kobe, 2009, pp. 3212-3217.

R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning Point Cloud Views using Persistent Feature Histograms,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26, 2008.

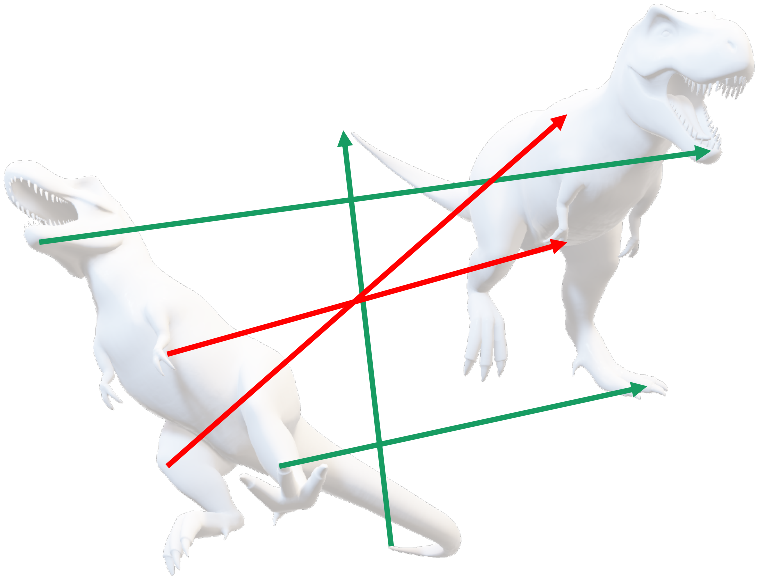
Global Alignment

- Consider 2 models (point clouds) of the same object (fully or partially)
  - How can we “roughly” align them?
    - We cannot use ICP, if the initial poses are too different.
    - Can we use some kind of (3D) “features” and match them?



Global Alignment

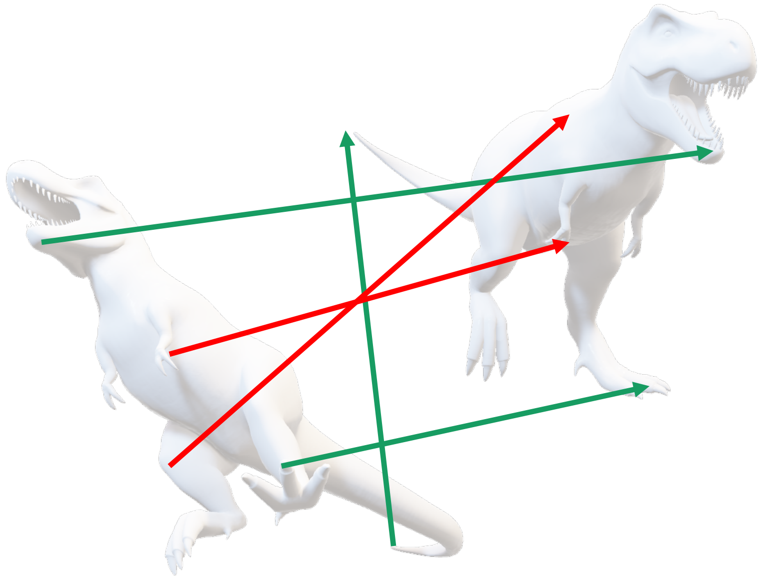
- Matching 3D features generally results in many false-matches
  - Big additional computational cost, brings only small increase in the matching accuracy ☹
  - Need for an algorithm that is robust to outliers.



Global Alignment – RANSAC in 3D

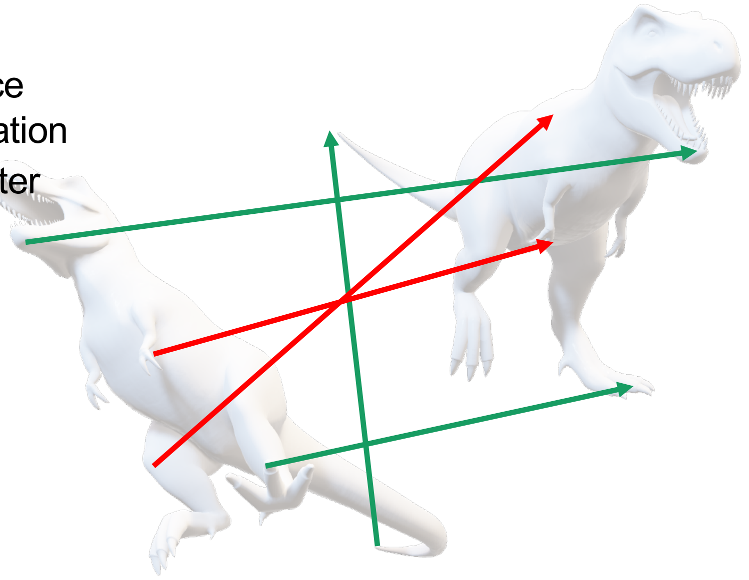
- Matching 3D features generally results in many false-matches
  - Big additional computational cost, brings only small increase in the matching accuracy ☹
  - Need for an algorithm that is robust to outliers.

- RANSAC
  - We can use:
    - Model 1 (point cloud with normals)
    - Model 2 (point cloud with normals)
    - Feature matches (containing many outliers)
  - In order to:
    - Estimate the "rough" pose difference between the 2 models



Global Alignment – RANSAC in 3D

- RANSAC for Pose Estimation
  - Randomly choose 3 pairs of matched points
  - Estimate the relative pose
    - Kabsch/ Procrustes
  - Apply the transformation and assess its “validity”:
    - number of inliers: matched point pairs whose distance became “small” after applying the specific transformation
    - Sum of distances between all matched point pairs after transformation
    - …
  - Keep the transformation with most inliers
  - Repeat random sampling



Outline

- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?

- Point Cloud Registration
  - Local Alignment
    - Iterative Closest Point (ICP) algorithm
  - Global Alignment
    - 3D Feature Descriptors
      - Spin Images
      - PFH
      - FPFH
    - Random Sample Consensus (RANSAC) in 3D

- <u>Summary</u>

Summary

- Point clouds provide the geometry of the scene/object (and might also combine it with color information)
- Pose Estimation very important for many tasks of Autonomous Systems

- Point Cloud Registration:
  1. "Rough" alignment (Global)
     - 3D Feature Descriptors (Spin Images, FPH, FPFH, …)
     - RANSAC for robust model fitting
  2. "Fine-tuning" of alignment (Local)
     - ICP
       - » Kabsch / Procrustes

Source: https://en.wikipedia.org/wiki/Otsu%27s_method

Lazaros Nalpantidis

3D Point Cloud Processing - Pose Estimation