Perception for Autonomous Systems

Lecture 5 - 3D Point Cloud Processing - Pose Estimation (01/03/2021)

Outline/Content:

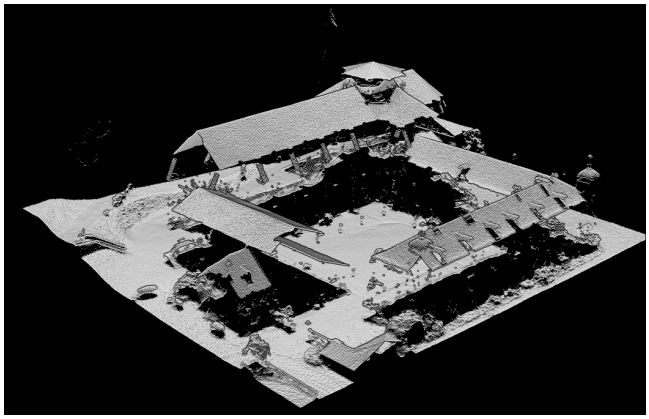
- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?
- Point Cloud Registration
    - Local Alignment
        - Iterative Closest Point (ICP) algorithm
    - Global Alignment
        - 3D Feature Descriptors
        - Spin Images
        - PFH
        - FPFH
    - Random Sample Consensus (RANSAC) in 3D

> Reading Material:
>
> - Book A, Sections 6.1 and 6.2
> - [Paper 1 - ICP] 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.
> - [Paper 2 - Spin Images] 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.
> - [Paper 3 - PFH] 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.
> - [Paper 4 - FPFH] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning Point Cloud Views using Persistent Feature Histograms,” in Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26, 2008.

What is a 3D Point Cloud? Point Cloud Expert: Florent Poux Medium

A point cloud is a set of data points in a three-dimensional coordinate system. These point are spartially defined by [x, y, z]coordinates and often present a surface of a physical asset or object. Reality capture devises obtain the external surface in its three dimensions to generate the point cloud. These are commonly obtained through Photogrammetry, LiDAR (Terrestrial Laser Scanning), Mobile Mapping, Aerial LiDAR, depth sensing, sonar, and more recently deep learning through Generative Adversarial Networks.

Photogrammetry	**Aerial LiDAR**

Photogrammetry	Aerial LiDAR
	

Why do we need 3D Point Clouds? Article

The world is in 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

What is "Pose"

Pose

- the tranformation (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 known 3D object or scene... "(Szelinisk)

Why is 3D Pose Estimation Important? Article

Many applications in Autonomous Systems

- Robot Grasping/Manipulation
- Quality Inspection
- Augmented reality
- Progressive map building (real-time)
- Localization (real-time)

Other applications:

- rescue and recovery
- explore dense forests

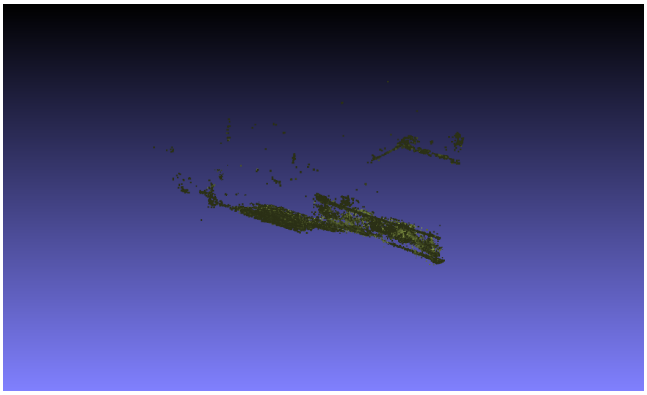
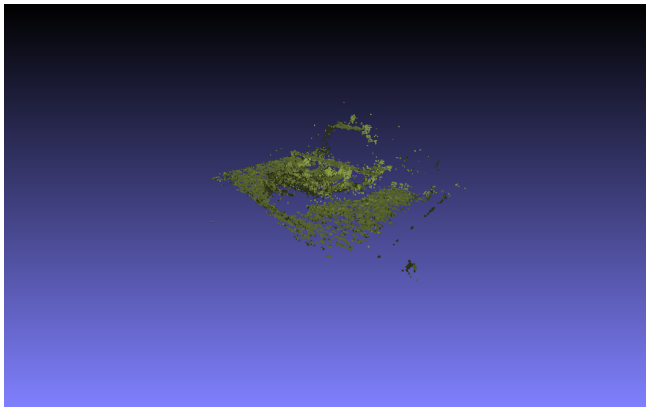
> - Rescue and recovery: Robots or drones with LiDAR sensors can reach dangerous locations and scan or generate useful data helping humans to navigate safely at such risky places and avoid mishaps. Natural disasters, unknown hidden dense forests and dark caves are the best examples of such dangerous locations.
> - Explore dense forests: A sprawling maya network in the guatemala jungle has been discovered with the help of Aerial LiDAR, which mapped the dense forest and discovered subtle sign of the archaelogical wonder.

Point Cloud registration Paper

Definition: Point Cloud Registration is a fundamental problem in 3D computer vision and photogrammetry. Given several sets of points in different coordinate systems, the aim of registration is to find the transformation that best aligns all of them into a common coordinate system. Point Cloud Registration plays a significant role in many vision applications such as 3D model reconstruction, cultural heritage management, landslide monitoring and solar energy analysis.

Generally 2 steps are needed to register 2 given point clouds of the same object (fully or partially)

- Global alignment (3D Feature Descriptors)
  - The 2 models are "roughly aligned"
- Local alignment (ICP)
  - Starting from a "rough" initial alignment, find the exact precise alignment

Before Registration	**After Registration**
	

Local Alignment Jupyter-Notebook

Having two scans (point clouds) of the same object P={pi} and Q={qi} we want to find a transformation (rotation R and translation t) to apply to P to match Q as good as possible (alignment).

This leads to the following two questions:

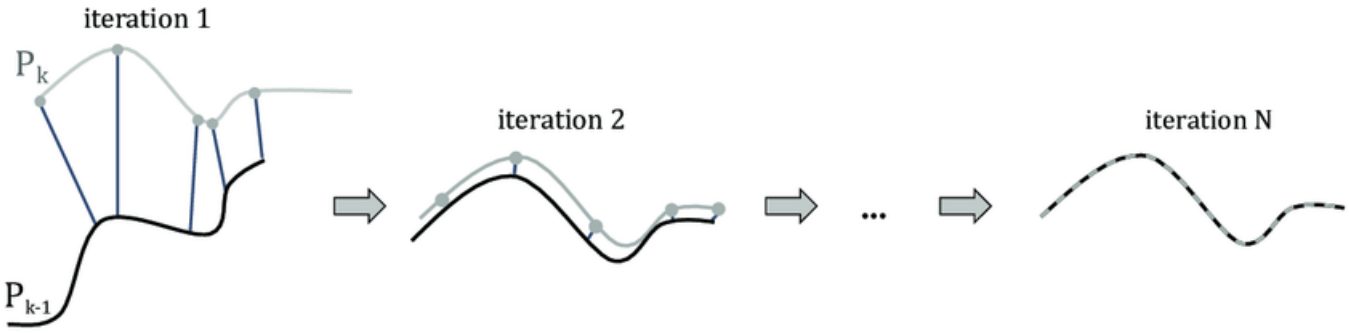
- What is the difference of theirs pose? ... or equivalently..
- What is the transformation that can fully align them?

> Checkout the linked jupyter notebook to see the IPC algorithm hands on

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

Visual representation of the IPC algorithm



Elaborating the bullet points 1 and 2 in greater detail

For each object point p in model 1, find the nearest point q in model 2: How to compute all possible pairs for proximity?

Simple k-nearest neighbor approach, is being used to find correspondences (similair points) in the two point clouds. However, this approach can be computationally heavy and slow. A better and more suitable approach is K-d trees.

Use all pairs (p,q) to estimate the transformation from model 1 to model 2: How to estimate the transformation from model 1 to model 2, fiven 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 cp, cq of both models
    - Subtract from each point coordinates the coordinates of its corresponding centroid:
        - p' = p - cp and q' = q - cq
- Compute Covariance Matrix: Cpq = Sum(p' * q^'T)
- Compute the optimal rotation
    - Calculate the singular value decomposition (SVD) of the covariance matrix: Cpq = USV'
    - Calculate rotation matrix: R = UV^T
- Compute the optimal translation: T = cq - Rcp

Why does the Kabsch Algorithm needs to be applied at every iteration?

The reason for the Kabsch Algorithm being applied at every iteration is that our proximity approach is not able to initially find all correspondences for each (p,q) pair. Thus, the algorithm needs to revalue it's estimation based on the new values it's provided with, until it converges or a stop criterion is met.

Global Alignment

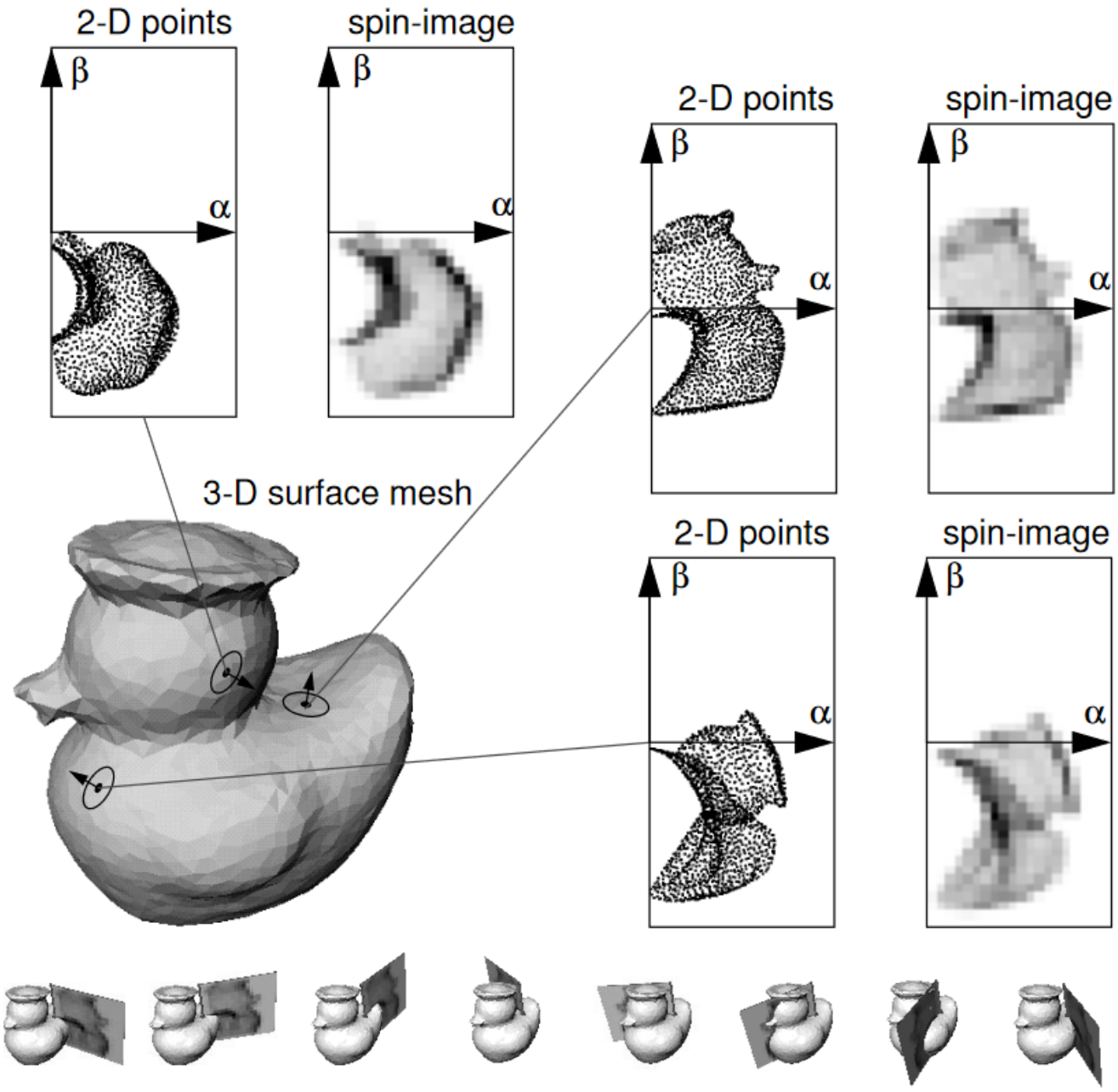
Consider 2 models (point clouds) of the same object (fully or partially)

- 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?

Spin Images: Paper

- "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.

Visualization:



Spin-images from points on one surface are compared by computing correlation coefficient with spin-images from points on another surface; when two spin-images are highly correlated, a point correspondence between the surfaces is established. More specifically, before matching, all of the spin-images from one surface (the model) are constructed and stored in a spin-image stack. Next, a vertex is selected at random from the other

surface (the scene) and its spin-image is computed. Point correspondences are then established between the selected point and the points with best matching spin-images on the other surface. This procedure is repeated for many points resulting in a sizeable set of point correspondences(~100). Point correspondences are then grouped and outliers are eliminated using geometric consistency. Groups of geometrically consistent correspondences are then used to calculate rigid transformations that aligns one surface with the other. After alignment, surface matches are verified using a modified iterative closest point algorithm. The best match is selected as the one with the greatest overlap between surfaces.

PHP (Point Feature Histograms) Tutorial

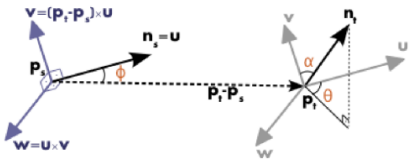
Global Alignment – 3D Feature Descriptors

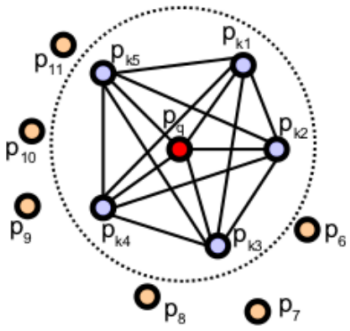
- **PFH (Point Feature Histograms)**

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

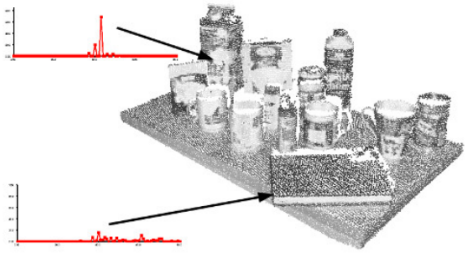
$$\phi = \arccos\left(u \cdot \frac{\left(p_t - p_s\right)}{\left\|p_t - p_s\right\|_2}\right)$$

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





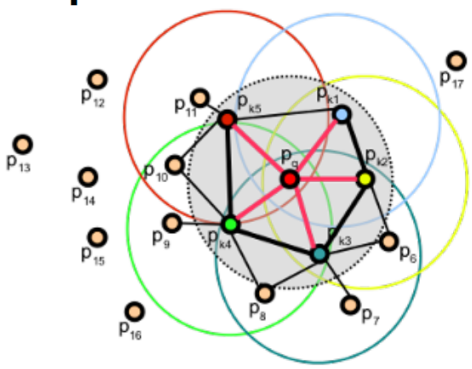
- Calculate these 3 values for all pairs of points within a radius *r* from the considered point (maybe also their distance *d*)
- The set of all triplets/quadruplets are binned in a histogram – This is the multi-dimensional descriptor

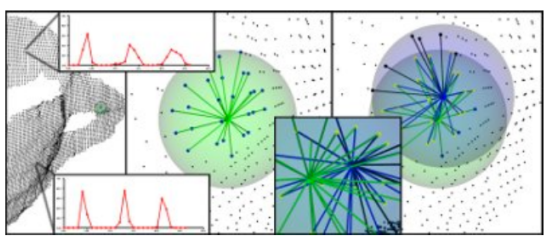


FPFH (Fast Point Feature Histograms) Tutorial

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





RANSAC in 3D Youtube

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 to the rescue

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

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

Pros:

1. RANSAC often produces better results, even in noiseless cases
2. RANSAC does not require a good initial estimate of the transform between the two data sets
3. RANSAC can be used with data sets that do not have as many local features