Three-Dimensional Sensors
Lecture 5: Point-Cloud Processing

Radu Horaud
INRIA Grenoble Rhone-Alpes, France
Radu.Horaud@inria.fr
http://perception.inrialpes.fr/
3D Data

- Data segmentation: K-means, Gaussian mixtures, spectral clustering.
- Data registration: Iterative closest point (ICP), soft assign methods, robust registration.
Data Representation

- Principal component analysis (PCA): The data are represented in an intrinsic coordinate system and projected onto a lower dimensional space (2D or 1D).
- There are many interesting variants of PCA: probabilistic PCA (PPCA), mixture of PPCA, kernel PCA, etc.
- KD-trees (K-dimensional trees): The 3D point cloud is represented as a binary 3D-tree by recursively splitting the point cloud into two subsets. Provides an efficient way to manipulate the point cloud.
- KNN-graph (K nearest neighbor graph): The 3D point cloud is represented as a sparse undirected weighted graph.
Data Segmentation

- **K-means clustering**: The data are grouped into $K$ spherical clusters. The number of clusters is provided in advance. This algorithm is often used to initialize other clustering methods.

- **The Gaussian mixture model (GMM)**: A more sophisticated clustering method is based on a mixture of Gaussian distributions. This is generally solved using the expectation-maximization algorithm (EM).

- **K-means and GMM** work well on spherical or ellipsoidal groups of points. Spectral clustering operates in the *spectral space* spanned by the eigenvectors of the symmetric matrix associated with a KNN-graph.

- **Clustering methods** need KD-trees for efficiently accessing the data points.
Data Registration

- “Fuse” data gathered with several 3D sensors or with a moving 3D sensor.
- Each sensor provides a point cloud in a sensor-centered coordinate frame.
- There is only a partial overlap between the two point clouds.
- The two point clouds must be registered or represented in the same coordinate frame.
- The registration process requires point-to-point correspondences which is a difficult problem.
Data Registration Methods

- Iterative closest point (ICP) is the most popular rigid registration method that needs proper initialization of the registration parameters (rotation and translation).
- A number of robust variants of ICP were proposed for eliminating bad points (outliers).
- An alternative to ICP is to use a generative probabilistic model such as GMM, or EM-based point registration.
Some Notations and Definitions

- Let’s start with a few more notations:
- The input (observation) space: $\mathbf{X} = [\mathbf{x}_1 \ldots \mathbf{x}_i \ldots \mathbf{x}_n]$, $\mathbf{x}_i \in \mathbb{R}^3$
- The output (latent) space: $\mathbf{Y} = [\mathbf{y}_1 \ldots \mathbf{y}_i \ldots \mathbf{y}_n]$, $\mathbf{y}_i \in \mathbb{R}^d, 1 \leq d \leq 3$
- **Projection**: $\mathbf{Y} = \mathbf{Q}^\top \mathbf{X}$ with $\mathbf{Q}^\top$ a $d \times 3$ matrix.
- $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_d$
Computing the Spread of the Data

- We start with $n$ scalars $x_1 \ldots x_n$; the mean and the variance are given by:

$$\overline{x} = \frac{1}{n} \sum_{i} x_i \quad \sigma_x = \frac{1}{n} \sum_{i} (x_i - \overline{x})^2 = \frac{1}{n} \sum_{i} x_i^2 - \overline{x}^2$$

- More generally, for the data set $X$:
  - The mean: $\overline{x} = \frac{1}{n} \sum_{i} x_i$
  - The covariance matrix is *semi-definite positive symmetric* of dimension $3 \times 3$:

$$C_X = \frac{1}{n} \sum_{i} (x_i - \overline{x})(x_i - \overline{x})^\top = \frac{1}{n} XX^\top - \overline{xx}^\top$$
Maximum-Variance Formulation of PCA

- Let’s center and project the data $X$ onto a line along a unit vector $u$. The variance along this line writes:

$$
\sigma_u = \frac{1}{n} \sum_{i} (u^\top (x_i - \bar{x}))^2
$$

$$
= u^\top \left( \frac{1}{n} \sum_{i} (x_i - \bar{x})(x_i - \bar{x})^\top \right) u
$$

$$
= u^\top C_X u
$$

- Find $u$ maximizing the variance under the constraint that $u$ is a unit vector:

$$
\mathbf{u}^* = \operatorname{argmax}_{u} \left\{ u^\top C_X u + \lambda(1 - u^\top u) \right\}
$$
First note that the $3 \times 3$ covariance matrix is a symmetric semi-definite positive matrix. (The associated quadratic form above is non-negative).

Taking the derivative with respect to $u$ and setting the derivatives equal to 0, yields: $C_X u = \lambda u$

Making use of the fact that $u$ is a unit vector we obtain:

$\sigma_u = \lambda$

Solution: The principal or largest eigenvector–eigenvalue pair $(u_{\text{max}}, \lambda_{\text{max}})$ of the covariance matrix.
Eigendecomposition of the Covariance Matrix

- Assume that the data are centred:
  \[ nC_X = XX^\top = nU\Lambda U^\top \]

Where \( U \) is a \( 3 \times 3 \) orthogonal matrix and \( \Lambda \) is the diagonal matrix of eigenvalues:

\[ \Lambda = [\lambda_1 \lambda_2 \lambda_3] \]

- If the point-cloud lies on a lower \( d \)-dimensional space (collinear or planar points):
  \[ d = \text{rank}(X) < 3 \]

\[ \Lambda_d = [\lambda_1 \lambda_d] \]

\[ C_X = \tilde{U}\Lambda_d\tilde{U}^\top \]

- \( \tilde{U} = UI_{3 \times d} \) is a \( 3 \times d \) column-orthogonal matrix
- \( \tilde{U}^\top = I_{d \times 3}^\top U^\top \) is a \( d \times 3 \) row-orthogonal matrix
Data Representation in the Eigen (Sub)space

- Coordinate change: \( Y = QX \); We have

\[
YY^\top = QXX^\top Q^\top = nQ\tilde{U}\Lambda_d\tilde{U}^\top Q^\top
\]

1. The projected data have a diagonal covariance matrix:

\[
\frac{1}{n} YY^\top = \Lambda_d, \text{ by identification we obtain}
\]

\[
Q = \tilde{U}^\top
\]

2. The projected data have an identity covariance matrix, this is called *whitening the data*:

\[
\frac{1}{n} YY^\top = I_d
\]

\[
Q = \Lambda_d^{-\frac{1}{2}} \tilde{U}^\top
\]

- Projection of the data points onto principal direction \( u_i \):

\[
(y_{1i} \ldots y_{ni}) = \lambda_i^{-1/2} \underbrace{u_i^\top (x_1 \ldots x_n)}_{\text{whitening}}
\]
Summary of PCA

- The eigenvector-eigenvalue pairs of the covariance matrix correspond to a *spectral* representation of the point cloud, or a *within representation*.

- This eigendecomposition allows to reduce the dimensionality of the point cloud to one plane or one line and then to project the cloud onto such a linear subspace.

- The largest eigenvalue-eigenvector pair defines the direction of maximum variance. By projecting the data onto this line one can order the data (useful for data organization, i.e., KD-trees).

- The eigenvalue-eigenvector pairs can be efficiently computed using the power method: get a random unit vector $\mathbf{x}^{(0)}$ and iterate $\mathbf{x}^{(k+1)} = \mathbf{C} \mathbf{x}^{(k)}$, normalize $\mathbf{x}^{(k+1)}$, etc., until $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \varepsilon$. Then $\mathbf{u}_{\text{max}} = \mathbf{x}^{(k+1)}$. 
KD Trees

- KD-tree ($K$-dimensional tree) is a data structure that allows to organize a point cloud under the form of a binary tree.
- The basic idea is to recursively and alternatively project the points onto the $x$, $y$, $z$, $x$, $y$, $z$, etc., axes, to order the points along each axis and to split the set into two halves.
- This point-cloud organization facilitates and accelerates the search of nearest neighbors (at the price of kd-tree construction).
- A more elaborate method (requiring more pre-processing time) is to search for the principal direction and split the data using a plane orthogonal to this direction, and apply this strategy recursively.
An Example of a 2D-tree (1)
An Example of a 2D-tree (2)
An Example of a 2D-tree (3)
An Example of a 2D-tree (5)
K-means Clustering

- What is a cluster: a group of points whose inter-point distance are small compared to distances to points outside the cluster.
- Cluster centers: $\mu_1, \ldots, \mu_m$.
- Goal: find an assignment of points to clusters as well as a set of mean-vectors $\mu_k$.
- Notations: For each point $x_j$ there is a binary indicator variable $r_{jk} \in \{0, 1\}$.
- Objective: minimize the following distortion measure:

$$J = \sum_{j=1}^{n} \sum_{k=1}^{m} r_{jk} \| x_j - \mu_k \|^2$$
The K-means Algorithm

1. Initialization: Choose $m$ and initial values for $\mu_1, \ldots, \mu_m$.
2. First step: Assign the $j$-th point to the closest cluster center:

$$r_{jk} = \begin{cases} 
1 & \text{if } k = \arg \min_l \|x_j - \mu_l\|^2 \\
0 & \text{otherwise}
\end{cases}$$

3. Second Step: Minimize $J$ to estimate the cluster centers:

$$\mu_k = \frac{\sum_{j=1}^{n} r_{jk} x_j}{\sum_{j=1}^{n} r_{jk}}$$

4. Convergence: Repeat until no more change in the assignments.
How to Represent This Point Cloud?
Building a Graph from a Point Cloud

- K-nearest neighbor (KNN) rule
- $\varepsilon$-radius rule
- Other more sophisticated rules can be found in the literature, i.e., Lee and Verleysen. Nonlinear Dimensionality Reduction (Appendix E). Springer. 2007.

Remark: The KD-tree data structure can be used to facilitate graph construction when the number of points is large.
The Graph Partitioning Problem

- We want to find a partition of the graph such that the edges between different groups have very low weight, while the edges within a group have high weight.

**The mincut problem:**

1. Edges between groups have very low weight, and
2. Edges within a group have high weight.
3. Choose a partition of the graph into $k$ groups that minimizes the following criterion:

$$\text{mincut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A_i})$$

with

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$
RatioCut and NormalizedCut

- Often, the mincut solution isolates a vertex from the rest of the graph.
- Request that the groups are reasonably large.
- **Ratio cut** (Hagen & Kahng 1992) minimizes:

\[
\text{RatioCut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{|A_i|}
\]

- Here $|A|$ refers to the number of vertices in group $A$.
- **Normalized cut**: (Shi & Malik 2000)

\[
\text{NCut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{\text{vol}(A_i)}
\]
What is Spectral Clustering?

- Both ratio-cut and normalized-cut minimizations are NP-hard problems
- Spectral clustering is a way to solve relaxed versions of these problems:
  1. Build the Laplacian matrix of the graph
  2. Compute the smallest (non-null) eigenvalue-eigenvector pairs of this matrix
  3. Map the graph vertices into the space spanned by these eigenvectors
  4. Apply the K-means algorithm to the new point cloud
The Laplacian Matrix of a Graph

- $f : \mathcal{V} \rightarrow \mathbb{R}$, i.e., $f(v_1), \ldots, f(v_n)$.
- $(Lf)(v_i) = \sum_{v_j \sim v_i} (f(v_i) - f(v_j))$
- Connection between the Laplacian and the adjacency matrices:

\[
L = D - A
\]

- The degree matrix: $D := D_{ii} = d(v_i)$.

\[
L = \begin{bmatrix}
2 & -1 & -1 & 0 \\
-1 & 3 & -1 & -1 \\
-1 & -1 & 2 & 0 \\
0 & -1 & 0 & 1
\end{bmatrix}
\]
We consider *undirected weighted graphs*; Each edge $e_{ij}$ is weighted by $w_{ij} > 0$. We obtain:

$$\Omega := \begin{cases} 
\Omega_{ij} = w_{ij} & \text{if there is an edge } e_{ij} \\
\Omega_{ij} = 0 & \text{if there is no edge} \\
\Omega_{ii} = 0 
\end{cases}$$

- The degree matrix: $D = \sum_{i \sim j} w_{ij}$
Often we will consider:

\[ w_{ij} = \exp \left( -\frac{\|x_i - x_j\|^2}{\sigma^2} \right) \]

**L = D − Ω**

- \( L \) is symmetric and positive semi-definite \( \iff w_{ij} \geq 0 \).
- \( L \) has \( n \) non-negative, real-valued eigenvalues:
  \[ 0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n. \]
Laplacian embedding: Mapping a graph on a line

- Map a weighted graph onto a line such that connected nodes stay as close as possible, i.e., minimize
  \[ \sum_{i,j=1}^{n} w_{ij} (f(v_i) - f(v_j))^2, \] or:
  \[ \arg \min_{f} \ f^\top L f \quad \text{with:} \quad f^\top f = 1 \quad \text{and} \quad f^\top 1 = 0 \]

- The solution is the eigenvector associated with the smallest nonzero eigenvalue of the eigenvalue problem: \( Lf = \lambda f \), (the Fiedler vector) \( u_2 \).

- Practical computation of the eigenpair \( \lambda_2, u_2 \): the shifted inverse power method (see lecture 2).
Mapping the Graph’s Vertices on the Eigenvector
Spectral Embedding using the Laplacian

- Compute the eigendecomposition \( L = D - \Omega = U\Lambda U^\top \).
- Select the \( k \) smallest non-null eigenvalues \( \lambda_2 \leq \ldots \leq \lambda_{k+1} \).
- \( \lambda_{k+2} - \lambda_{k+1} = \text{eigengap}. \)
- We obtain the \( n \times k \) column-orthogonal matrix \( \tilde{U} = [u_2 \ldots u_{k+1}] \):

\[
\tilde{U} = \begin{bmatrix}
  u_2(v_1) & \ldots & u_{k+1}(v_1) \\
  \vdots & \ddots & \vdots \\
  u_2(v_n) & \ldots & u_{k+1}(v_n)
\end{bmatrix}
\]

- Embedding: The \( i \)-row of this matrix correspond to the representation of vertex \( v_i \) in the \( \mathbb{R}^k \) basis spanned by the orthonormal vector basis \( u_2, \ldots, u_{k+1} \).
- Therefore: \( Y = [y_1 \ldots y_i \ldots y_n] = \tilde{U}^\top \)
Laplacian Eigenmap
“Fuse” data gathered with several 3D sensors or with a moving 3D sensor.

Each sensor provides a point cloud in a sensor-centered coordinate frame.

There is only a partial overlap between the two point clouds.

The two point clouds must be registered or represented in the same coordinate frame.

The registration process requires point-to-point correspondences which is a difficult problem.