Lecture 08 – Pose Estimation I

Nov 11nd, 2018

Danping Zou,

Associate Professor

Institute for Sensing and Navigation

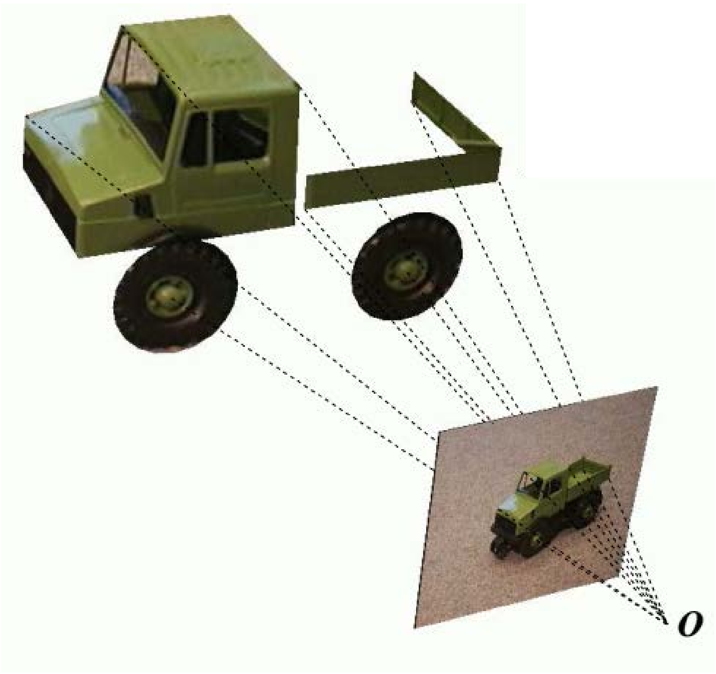
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Outline

- **About pose estimation**
- **3D-to-3D registration**
  - Rotation only
  - Rotation plus Translation
  - Unknown correspondences – Iterative Closest Point (ICP)
- **3D-to-2D registration (Camera pose estimation)**
  - 3D objects
    - Close-form algorithm - P3P
    - Iterative algorithm - POSIT
    - Iterative algorithm - Nonlinear least squares
  - Planar objects
    - Known patterns – Checkboard box, QR pattern
    - Planar Pictures

Pose estimation

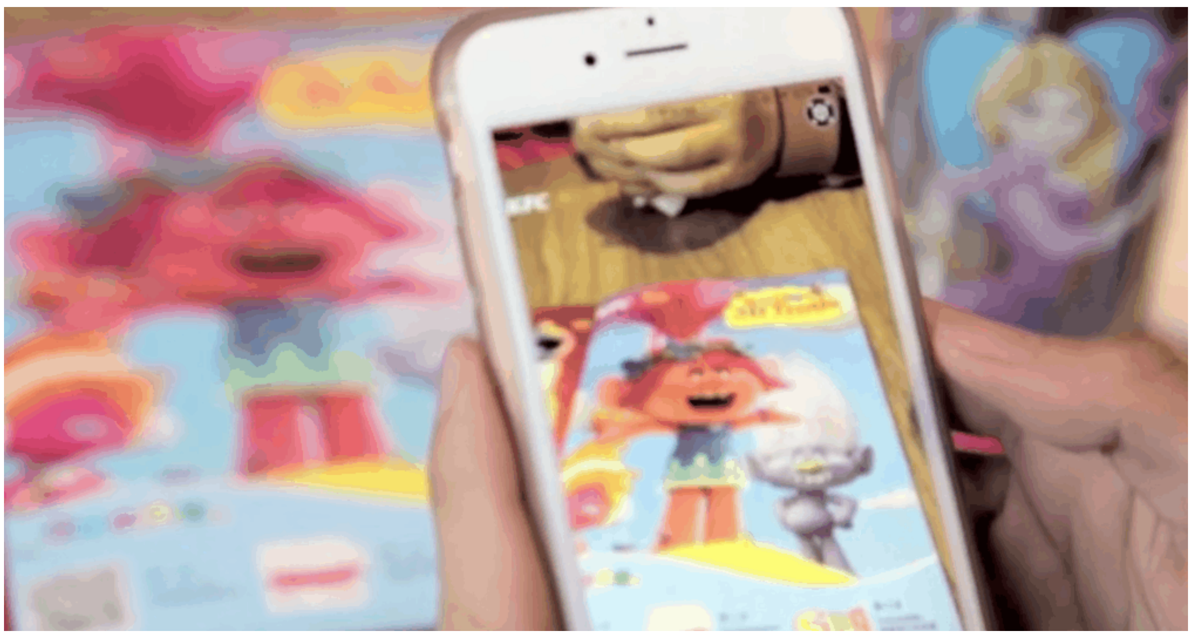
- Given a 3D model and its projection on the image, we want to get the camera pose with respect to the 3D model.
- This process can also treated as 3D-2D registration problem



$$\boldsymbol{X} \leftrightarrow \boldsymbol{x} \quad \Rightarrow \quad \mathbf{R}, \boldsymbol{t}$$

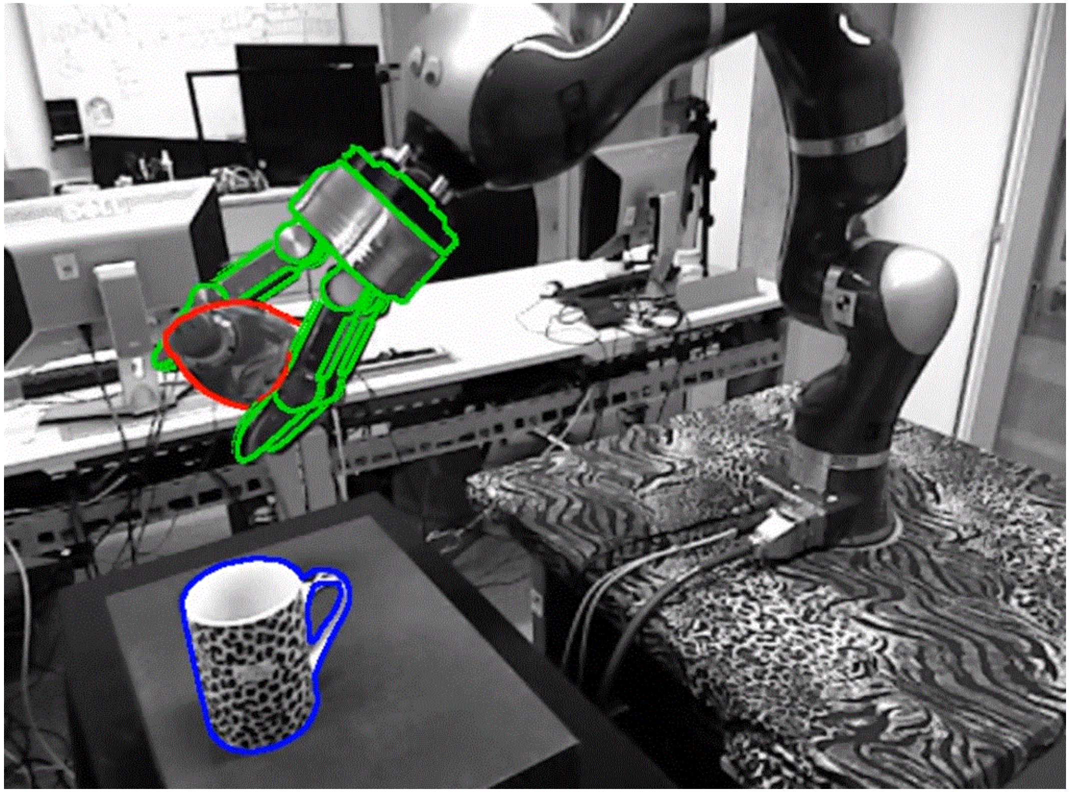
Pose estimation

- Pose estimation is a basic problem in augmented reality.
- About Augmented reality



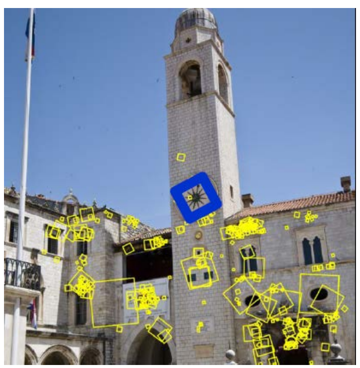
Pose estimation

- Pose estimation is also critical for grasping and manipulation in robotics.



Pose estimation

- Pose estimation is also applied to image-based localization
  - use 3D reconstruction method to generate 3D point clouds of the scene
  - extract the feature points from the query image and match them to the 3D points (Pose estimation)



Query image

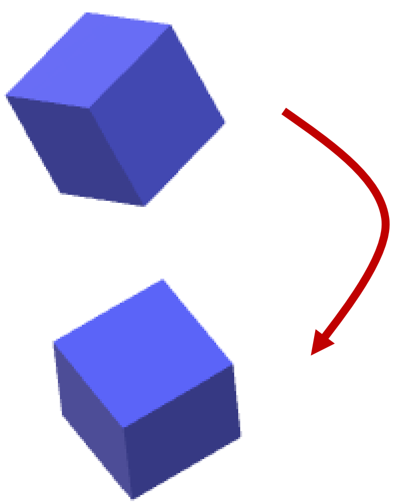
Feature matching

Pose estimation

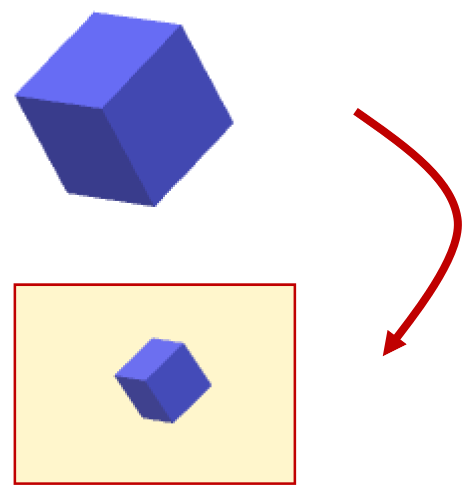
3D point cloud from internet photos

Pose estimation

- We first discuss about 3D-3D registration and then discuss about 2D-2D registration



$$\begin{bmatrix} \mathbf{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix}$$



$$\mathbf{K}[\mathbf{R}\ \boldsymbol{t}]$$

Rotation-only 3D-3D registration

- Rotation-only 3D-to-3D registration (Recall the last homework)
  - The corresponding points are known:

$$\boldsymbol{x}_i \leftrightarrow \boldsymbol{y}_i$$

  - Only rotation between two point clouds is applied:

$$\boldsymbol{y}_i = \mathbf{R}\boldsymbol{x}_i$$

  - Our problem is to solve the rotation $\mathbf{R}$ from the corresponding points

Rotation-only 3D-3D registration

- In the last homework, we use Gauss-Newton method optimize the object function iteratively,

$$\min_{\mathbf{R}} \sum_{i=1}^{n} (\boldsymbol{y}_i - \mathbf{R}\boldsymbol{x}_i)^2$$

- We start from an initial rotation : $\mathbf{R} \leftarrow \mathbf{R}_0$
- And iteratively solve the incremental parameter $\Delta\theta \in \mathbb{R}^{3\times 1}$

$$\mathbf{R} \leftarrow \mathbf{R} \boxplus \Delta\theta \qquad (\mathbf{R} \leftarrow \mathbf{R}\exp([\Delta\theta]_\times))$$

Rotation-only 3D-3D registration

- $\Delta\theta$ is solved by minimizing

$$\sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R}\exp(\Delta\theta^{\wedge})\boldsymbol{x}_i\|^2$$

$$\approx \sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R}(\mathbf{I} + [\Delta\theta]_{\times})\boldsymbol{x}_i\|^2 \quad \text{(First order approximation)}$$

$$\sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R}\boldsymbol{x}_i - \mathbf{R}[\Delta\theta]_{\times}\boldsymbol{x}_i\|^2$$

$$\sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R}\boldsymbol{x}_i + [\mathbf{R}\boldsymbol{x}_i]_{\times}\Delta\theta\|^2 \ (a \times b = [a]_{\times} b = -b \times a = -[b]_{\times} a)$$

Rotation-only 3D-3D registration

- It is a linear least squares problem :

$$\mathbf{J}\Delta\theta = \boldsymbol{z}$$

$$\begin{bmatrix} -[\mathbf{R}\boldsymbol{x}_1]\times \\ -[\mathbf{R}\boldsymbol{x}_1]\times \\ \cdots \\ -[\mathbf{R}\boldsymbol{x}_n]\times \end{bmatrix} \Delta\theta = \begin{bmatrix} \boldsymbol{y}_1 - \mathbf{R}\boldsymbol{x}_1 \\ \boldsymbol{y}_2 - \mathbf{R}\boldsymbol{x}_2 \\ \cdots \\ \boldsymbol{y}_n - \mathbf{R}\boldsymbol{x}_n \end{bmatrix}$$

$$\rightarrow \Delta\theta = (\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{z}$$

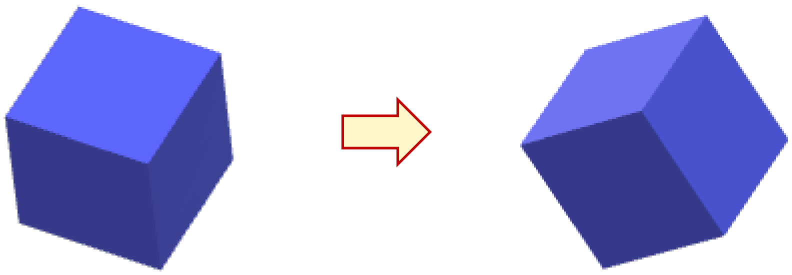
- We can find that the algorithm usually converges very fast even if the initial guess is not good.

Rotation-only 3D-3D registration

- In fact, we can solve the rotation in close from.

$$\boldsymbol{y}_i = \mathbf{R}\boldsymbol{x}_i$$

$$(i = 1, 2, \ldots)$$



Rotation-only 3D-3D registration

- **First approach** – Direct Linear Transformation (DLT)

$$\left.\begin{aligned} \boldsymbol{y}_1 &= \mathbf{R}\boldsymbol{x}_1 \\ \boldsymbol{y}_2 &= \mathbf{R}\boldsymbol{x}_2 \\ &\cdots \\ \boldsymbol{y}_n &= \mathbf{R}\boldsymbol{x}_n \end{aligned}\right\} \qquad \mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$$

- Let $\boldsymbol{r} = [r_1, r_2, r_3, \ldots, r_9]^{\mathrm{T}}$, we have

$$\boldsymbol{y}_i = \mathbf{R}\boldsymbol{x}_i \quad \Rightarrow \quad \underbrace{\begin{bmatrix} \boldsymbol{x}_i^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{x}_i^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{x}_i^{\mathrm{T}} \end{bmatrix}}_{3 \times 9} \boldsymbol{r} = \boldsymbol{y}_i$$

Rotation-only 3D-3D registration

- We can solve the vector of rotation elements by using at least three point correspondences.

$$\begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{x}_1^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{x}_1^{\mathrm{T}} \\ \dots & \dots & \dots \\ \boldsymbol{x}_i^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{x}_i^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{x}_i^{\mathrm{T}} \\ \dots & \dots & \dots \end{bmatrix} \boldsymbol{r} = \begin{bmatrix} \boldsymbol{y}_1 \\ \dots \\ \boldsymbol{y}_i \\ \dots \end{bmatrix}$$

$$\mathbf{X}\boldsymbol{r} = \boldsymbol{y}$$

$$\boldsymbol{r} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

Rotation-only 3D-3D registration

- However, DLT method does not impose the SO3 constraints on the rotation matrix.

$$\det(\mathbf{R}) = 1, \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}$$

- We can get a rotation that is the closest to the solved $\mathbf{R}^*$ by SVD:

$$\mathbf{R}^* = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$$

$$\mathbf{R} \leftarrow \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^{\mathrm{T}}$$

Rotation-only 3D-3D registration

- **Second approach – Absolute orientation** : using unit quaternion (naturally handling the nonlinear constraints on rotation)
- A vector $\boldsymbol{x}_i$ after a rotation $\boldsymbol{q}$ is given by $\boldsymbol{y}_i$

$$\tilde{\boldsymbol{y}}_i = \boldsymbol{q} \otimes \tilde{\boldsymbol{x}}_i \otimes \boldsymbol{q}^*$$

- Here, $\tilde{\boldsymbol{x}}_i = \begin{bmatrix} 0 \\ \boldsymbol{x}_i \end{bmatrix} \in \mathbb{R}^{4\times 1}$ and $\tilde{\boldsymbol{y}}_i = \begin{bmatrix} 0 \\ \boldsymbol{y}_i \end{bmatrix} \in \mathbb{R}^{4\times 1}$. We want to seek the unit quaternion :

$$\boldsymbol{q} \in \mathbb{R}^{4\times 1}, \|\boldsymbol{q}\| = 1$$

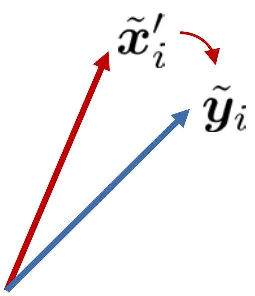
Rotation-only 3D-3D registration

- All we want to do is to seek a unit quaternion that makes

$$\boxed{\tilde{\boldsymbol{y}}_i \leftrightarrow \tilde{\boldsymbol{x}}'_i} = \boldsymbol{q} \otimes \tilde{\boldsymbol{x}}_i \otimes \boldsymbol{q}^*$$

- as close as possible.
- A straightforward way is to minimize the sum of squares :

$$\|\tilde{\boldsymbol{y}}_i - \tilde{\boldsymbol{x}}'_i\|^2$$



$$\rightarrow \underbrace{\boxed{\|\tilde{\boldsymbol{y}}_i\|^2 + \|\tilde{\boldsymbol{x}}'_i\|^2}}_{const.} - 2(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{x}}'_i)$$

Rotation-only 3D-3D registration

- So what we need to do is to find a unit quaternion that maximize the dot products of two quaternions:

$$\begin{aligned} &(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{x}}_i') \\ &= (\tilde{\boldsymbol{y}}_i, \boldsymbol{q} \otimes \tilde{\boldsymbol{x}}_i \otimes \boldsymbol{q}^*) \end{aligned}$$

- Recall that the dot product of two quaternions is invariant to quaternion multiplication.

$$(\mathbf{q} \otimes \mathbf{r})^{\mathrm{T}}(\mathbf{q} \otimes \mathbf{t}) = ([\mathbf{q}]_L \mathbf{r})^{\mathrm{T}}([\mathbf{q}]_L \mathbf{t}) = \mathbf{r}^{\mathrm{T}}[\mathbf{q}]_L^{\mathrm{T}}[\mathbf{q}]_L \mathbf{t} = \mathbf{r}^{\mathrm{T}}\mathbf{t}$$

Rotation-only 3D-3D registration

- We have
$$\begin{aligned}
&(\tilde{\boldsymbol{y}}_i, \boldsymbol{q} \otimes \tilde{\boldsymbol{x}}_i \otimes \boldsymbol{q}^*) \\
&= (\tilde{\boldsymbol{y}}_i \otimes \boldsymbol{q}, \boldsymbol{q} \otimes \tilde{\boldsymbol{x}}_i) \\
&= ([\tilde{\boldsymbol{y}}_i]_L \boldsymbol{q})^{\mathrm{T}}([\tilde{\boldsymbol{x}}_i]_R \boldsymbol{q}) \\
&= \boldsymbol{q}^{\mathrm{T}} [\tilde{\boldsymbol{y}}_i]_L^{\mathrm{T}} [\tilde{\boldsymbol{x}}_i]_R \boldsymbol{q}
\end{aligned}$$

- We want to solve the maximization problem

$$\arg\max_{\boldsymbol{q}} \boldsymbol{q}^{\mathrm{T}} \mathbf{A} \boldsymbol{q}$$

with respect to $\boldsymbol{q}, \|\boldsymbol{q}\| = 1$, here $\mathbf{A} = [\tilde{\boldsymbol{y}}_i]_L^{\mathrm{T}} [\tilde{\boldsymbol{x}}_i]_R$

Rotation-only 3D-3D registration

- Let $\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, \boldsymbol{q}_4$, $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ be the eigenvectors and corresponding eigenvalues of $\mathbf{A}$.

$$\lambda_i \boldsymbol{q}_i = \mathbf{A} \boldsymbol{q}_i \quad (\lambda_1 \le \lambda_2 \le \lambda_3 \le \lambda_4)$$

- The eigenvectors of a symmetric matrix are orthogonal:

$$\boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{q}_j = 0, (i \neq j), \|\boldsymbol{q}_i\| = 1$$

- $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, \boldsymbol{q}_4\}$ spans the eigen space .

Rotation-only 3D-3D registration

- For a given unit quaternion, we can represent it within the eigen space

$$\boldsymbol{q} = a_1\boldsymbol{q}_1 + a_2\boldsymbol{q}_2 + a_3\boldsymbol{q}_3 + a_4\boldsymbol{q}_4$$

- where $a_1^2 + a_2^2 + a_3^2 + a_4^2 = 1$.

$$\Downarrow \quad (\lambda_i\boldsymbol{q}_i = \mathbf{A}\boldsymbol{q}_i)$$

$$\mathbf{A}\boldsymbol{q} = a_1\lambda_1\boldsymbol{q}_1 + a_2\lambda_2\boldsymbol{q}_2 + a_3\lambda_3\boldsymbol{q}_3 + a_4\lambda_4\boldsymbol{q}_4$$

$$\Downarrow$$

$$\boldsymbol{q}^{\mathrm{T}}\mathbf{A}\boldsymbol{q} = a_1^2\lambda_1 + a_2^2\lambda_2 + a_3^2\lambda_3 + a_4^2\lambda_4$$

$$\Downarrow \quad (a_1^2 = 1 - a_2^2 - a_3^2 - a_4^2)$$

$$\boldsymbol{q}^{\mathrm{T}}\mathbf{A}\boldsymbol{q} = \lambda_1 + a_2^2(\lambda_2 - \lambda_1) + a_3^2(\lambda_3 - \lambda_1) + a_4^2(\lambda_4 - \lambda_1)$$

$$\Downarrow$$

$$\max(\boldsymbol{q}^{\mathrm{T}}\mathbf{A}\boldsymbol{q}) = \lambda_4, \boldsymbol{q} = \boldsymbol{q}_4$$

Summary

- Rotation-only 3D-3D registration
  - Iterative approach (Gauss-Newton)

$$\Delta\theta = (\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{z}, \quad \mathbf{R} \leftarrow \mathbf{R}\exp(\Delta\theta^{\wedge})$$

  - Close form approach
    - Rotation matrix – Direct Linear Transformation

$$\mathbf{X}\boldsymbol{r} = \boldsymbol{y} \rightarrow \mathbf{R} \quad \text{(SVD approximation)}$$

    - Unit quaternion

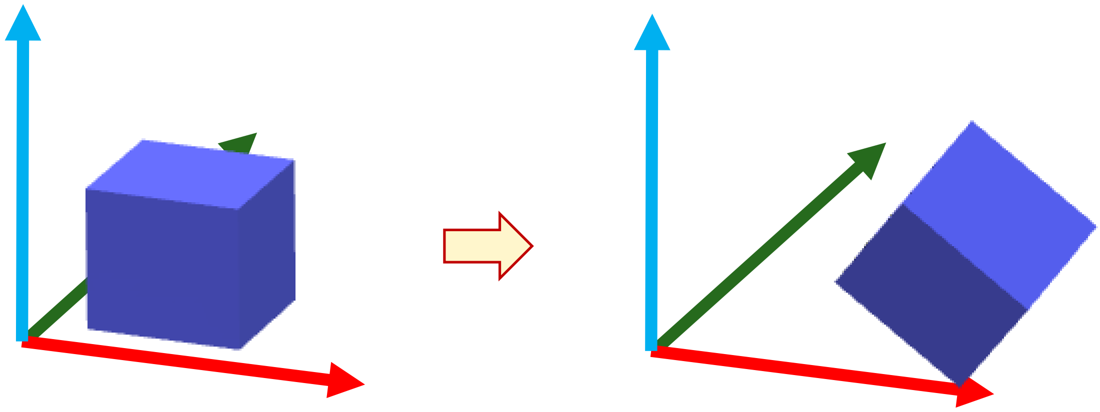
$$\arg\max_{\boldsymbol{q}} \boldsymbol{q}^{\mathrm{T}}\mathbf{A}\boldsymbol{q}$$

- Usually we first run the close form approach and then refine the estimation by iterative approach

3D-3D registration (Rigid)

- Now we consider the 3D-3D registration problem using rigid transformation.

$$\boldsymbol{y}_i = \mathbf{R}\boldsymbol{x}_i + \boldsymbol{t}$$
$$(i = 1, 2, \ldots)$$



3D-3D registration (Rigid)

- We can manage to obtain $\mathbf{R}$ by rotation-only registration

$$
\begin{aligned}
\boldsymbol{y}_1 &= \mathbf{R}\boldsymbol{x}_1 + \boldsymbol{t} \\
\boldsymbol{y}_2 &= \mathbf{R}\boldsymbol{x}_2 + \boldsymbol{t} \\
&\cdots \quad \cdots \\
\boldsymbol{y}_n &= \mathbf{R}\boldsymbol{x}_n + \boldsymbol{t}
\end{aligned}
$$

$$\bar{\boldsymbol{y}} = \mathbf{R}\bar{\boldsymbol{x}} + \boldsymbol{t}$$

$$
\begin{aligned}
(\boldsymbol{y}_1 - \bar{\boldsymbol{y}}) &= \mathbf{R}(\boldsymbol{x}_1 - \bar{\boldsymbol{x}}) \\
(\boldsymbol{y}_2 - \bar{\boldsymbol{y}}) &= \mathbf{R}(\boldsymbol{x}_2 - \bar{\boldsymbol{x}}) \\
&\cdots \quad \cdots \\
(\boldsymbol{y}_n - \bar{\boldsymbol{y}}) &= \mathbf{R}(\boldsymbol{x}_n - \bar{\boldsymbol{x}})
\end{aligned}
$$

$$\boldsymbol{r}_i = \mathbf{R}\boldsymbol{s}_i$$

3D-3D registration (Rigid)

- **3D-3D registration algorithm**
- Inputs : corresponding 3D points $\boldsymbol{x}_i \leftrightarrow \boldsymbol{y}_i, (i = 1, \ldots, n)$
- Outputs : rigid transformation $\mathbf{R}, \boldsymbol{t}$
  - Step 1. Compute the normalized vectors

$$\boldsymbol{s}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}},\ \boldsymbol{r}_i = \boldsymbol{y}_i - \bar{\boldsymbol{y}}$$

  - Step 2. Get the orientation by DLT or absolute orientation algorithm

$$\arg\max_{\boldsymbol{q}} \boldsymbol{q}^{\mathrm{T}} \mathbf{A} \boldsymbol{q} \rightarrow \boldsymbol{q} \quad or \quad \mathbf{X}\boldsymbol{r} = \boldsymbol{y} \rightarrow \mathbf{R}$$

  - Step 3. Compute the translation vector

$$\boldsymbol{t} = \bar{\boldsymbol{y}} - \mathbf{R}\bar{\boldsymbol{x}}$$

3D-3D registration (Rigid)

- We can minimize the nonlinear least squares to refine the estimation.

$$\arg\max_{\mathbf{R},\boldsymbol{t}} \sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R}\boldsymbol{x}_i - \boldsymbol{t}\|^2$$

- Gauss-Newton or Levenberg-Marquardt algorithm can be applied.

$$\boldsymbol{X} \leftarrow \boldsymbol{X} \boxplus \Delta \boldsymbol{x}$$

$$\begin{pmatrix} \mathbf{R} \\ \boldsymbol{t} \end{pmatrix} \leftarrow \begin{pmatrix} \mathbf{R}\exp(\Delta\theta^{\wedge}) \\ \boldsymbol{t} + \Delta\boldsymbol{t} \end{pmatrix}$$

- All we need is to solve the incremental step $\Delta \boldsymbol{x} = \begin{pmatrix} \Delta\theta \\ \Delta\boldsymbol{t} \end{pmatrix}$

3D-3D registration (Rigid)

- Using the first-order approximation :

$$\sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R}\exp(\Delta\theta^\wedge)\boldsymbol{x}_i - \boldsymbol{t} - \Delta\boldsymbol{t}\|^2$$

$$\approx \sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R}(\mathbf{I} + [\Delta\theta]_\times)\boldsymbol{x}_i - \boldsymbol{t} - \Delta\boldsymbol{t}\|^2$$

$$= \sum_{i=1}^{n} \|\boldsymbol{y}_i - \mathbf{R} - \boldsymbol{t} + [\mathbf{R}\boldsymbol{x}_i]_\times \Delta\theta - \Delta\boldsymbol{t}\|^2$$

$$\sum_{i=1}^{n} \|\boldsymbol{z}_i - \mathbf{J}_i \begin{bmatrix} \Delta\theta \\ \Delta\boldsymbol{t} \end{bmatrix} \|^2$$

3D-3D registration (Rigid)

- Gauss-Newton :

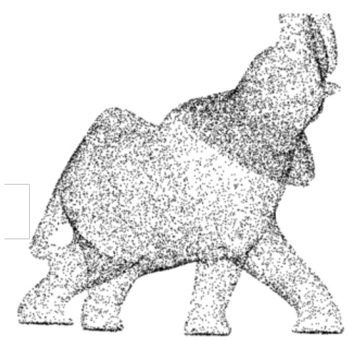
$$\Delta \boldsymbol{x} = (\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{z}$$

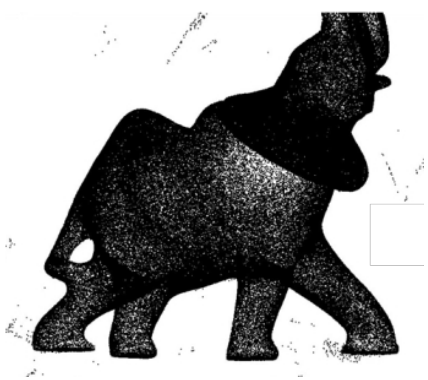
- Levinberg-Marquardt :

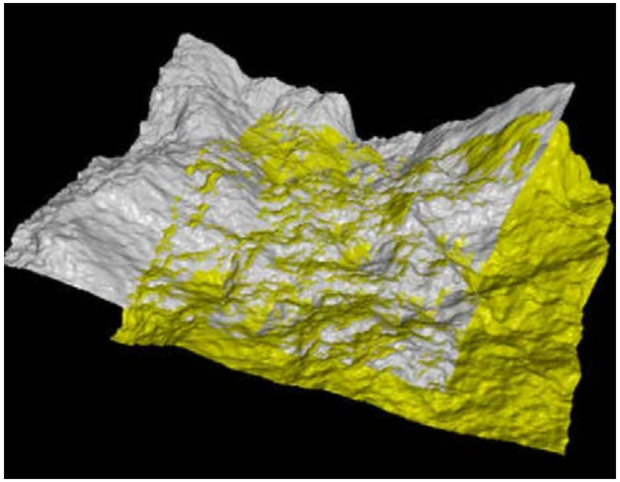
$$\Delta \boldsymbol{x} = (\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{z}$$

Iterative Closest Point algorithm

- If we do not know the point correspondences, how do we get the rigid transformation ?
- For example, given two 3D LiDAR scans
  - The density is different
  - Only the parts of the scans are overlapped

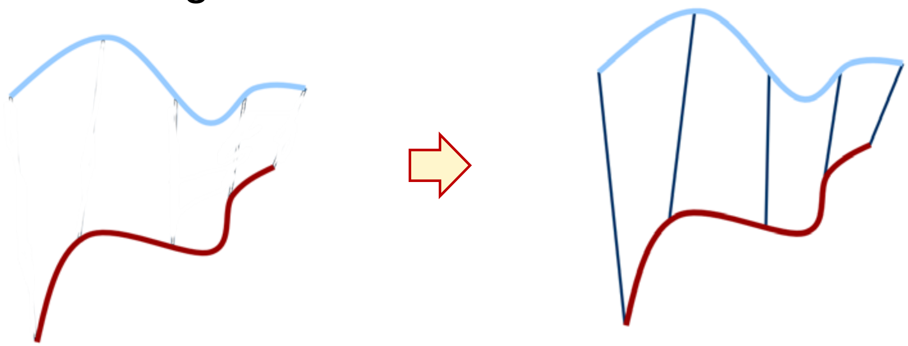




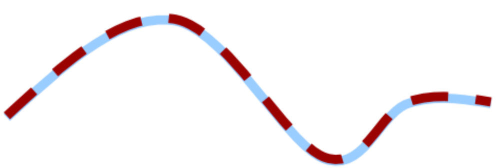


Iterative Closest Point algorithm

- Key idea: Iterate to find the corresponding points and estimate the rigid transformation (alignment)
  - Step 1 – **Matching** : find the closest point as the corresponding point using the current alignment.



  - Step2 – **Updating** : compute the alignment using the close-form solution as introduced previously.



Iterative Closest Point algorithm

- Matching step : for each point $\boldsymbol{y}_i$ in the point cloud, we search its corresponding point $\boldsymbol{x}_i^*$ using the current transformation

$$\arg\min_j \|\boldsymbol{y}_i - \boldsymbol{x}'_j\|^2 \rightarrow \boldsymbol{x}_i^*$$

$$(\boldsymbol{x}'_j = \mathbf{R}\boldsymbol{x}_j + \boldsymbol{t})$$

- After that, we get a set of corresponding 3D points

$$\begin{array}{c} \boldsymbol{y}_1 \leftrightarrow \boldsymbol{x}_1^* \\ \boldsymbol{y}_2 \leftrightarrow \boldsymbol{x}_2^* \\ \cdots \\ \boldsymbol{y}_n \leftrightarrow \boldsymbol{x}_n^* \end{array} \Rightarrow \mathbf{R}, \boldsymbol{t}$$

1.DLT (matrix)
2. Absolute orientation (quaternion)
3. Gauss-Newton/Levenberg-Marquardt

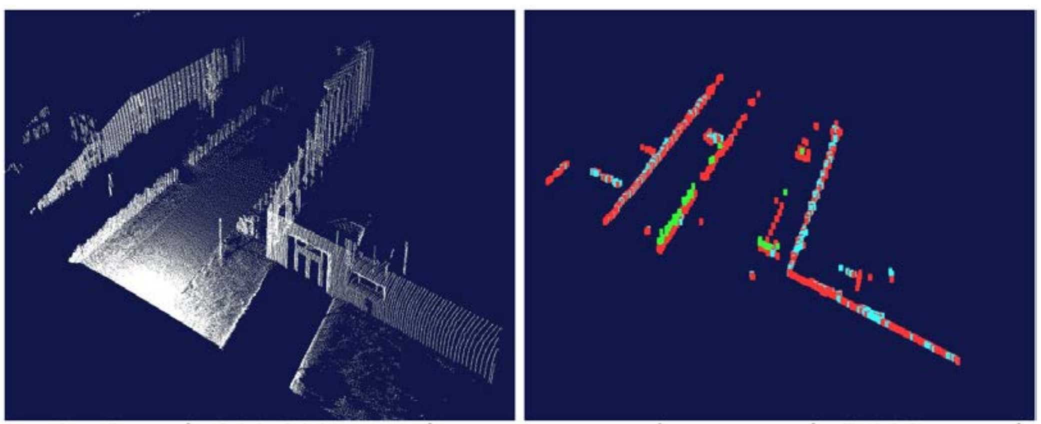
Iterative Closest Point algorithm

- ICP converges only if the starting point is “close enough” to the real solution.

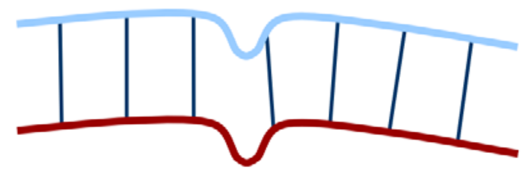
- The performance (accuracy & efficiency) largely depends on the first step – **matching**.
- In practice, several techniques can be used to improve the **matching** performance.

Iterative Closest Point algorithm

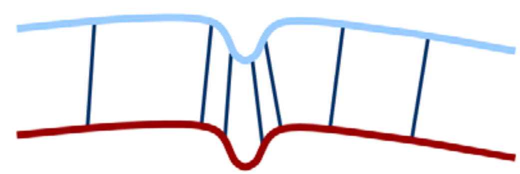
- **Sampling**
  - Uniform sampling
  - Random sampling
  - Normal-space sampling (use when the normal is available)
  - Feature-based sampling



3D Scan (~200.000 Points) Extracted Features (~5.000 Points)



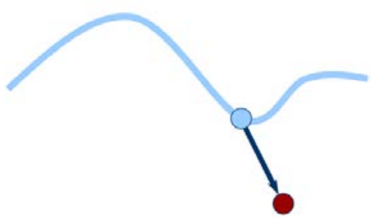
uniform sampling



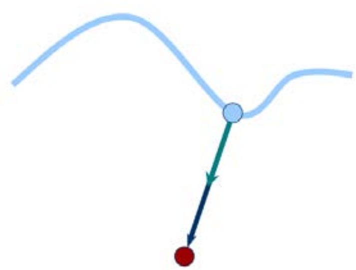
normal-space sampling

Iterative Closest Point algorithm

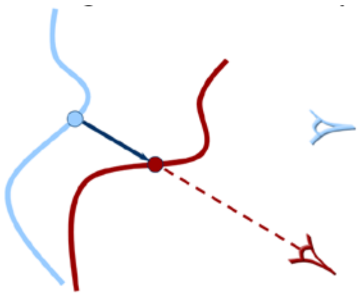
- Finding the corresponding point



Closest point



Normal shooting



Projection

- Using KD-trees or Oct-trees to improve the matching speed.

Summary

- **ICP** algorithm aligns two points clouds without known their correspondences.
- The two point clouds can be of different density, partially overlapped.
- The key idea is to iteratively finding the correspondences and update the rigid transformation.
- A good matching strategy is critical to the performance.
- A initial guess is also critical.
- **ICP** algorithm is widely used in autonomous cars who equipped with LiDAR.

Outline

- ~~**About pose estimation**~~
- ~~**3D-to-3D registration**~~
  - ~~Rotation only~~
  - ~~Rotation plus Translation~~
  - ~~Unknown correspondences – Iterative Closest Point (ICP)~~
- **3D-to-2D registration (Camera pose estimation)**
  - 3D objects
    - Close-form algorithm - P3P
    - Iterative algorithm - POSIT
    - Pose refinement - Nonlinear least squares
  - Planar objects
    - Known patterns – Checkboard box, QR pattern
    - Planar Pictures