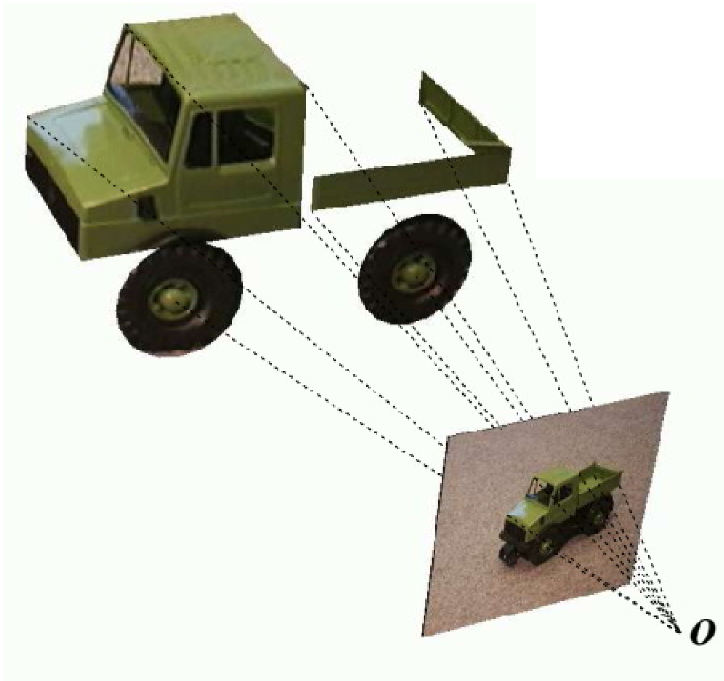
Lecture 06/07- Pose estimation & Absolute orientation

EE382-Visual localization & Perception

Danping Zou @Shanghai Jiao Tong
University

Pose estimation

- Pose estimation has been studied in computer vision since its beginning.
- It is crucial for many vision tasks:
  - Grasping
  - Manipulation
  - Self-localization (SLAM)



Pose estimation

- Pose – The transformation that maps the 3D object model to the sensory data.

$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}$$

Given

- The 3D object model is known.
- The camera is calibrated.
- Corresponding points (2D – 3D)

- All we want to know is $\mathbf{R}, \mathbf{t}$

Pose estimation

- **Algebraic approach:**
  - P3P (Fischer & Bolles. 1981)
  - PnP (Quan & Lan. 1999)
  - EPnP (Lepetit, et al. 2008)
- Positive aspects:
  - Fast
  - No initial guess
- Negative aspects:
  - Prone to noises
  - Numeric instabilities

Pose estimation

- **Optimization approach**
  - Nonlinear least squared problem

$$\min_{\mathbf{R},\mathbf{t}} \sum_1^n d(\mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_i, \mathbf{x}_i)^2$$

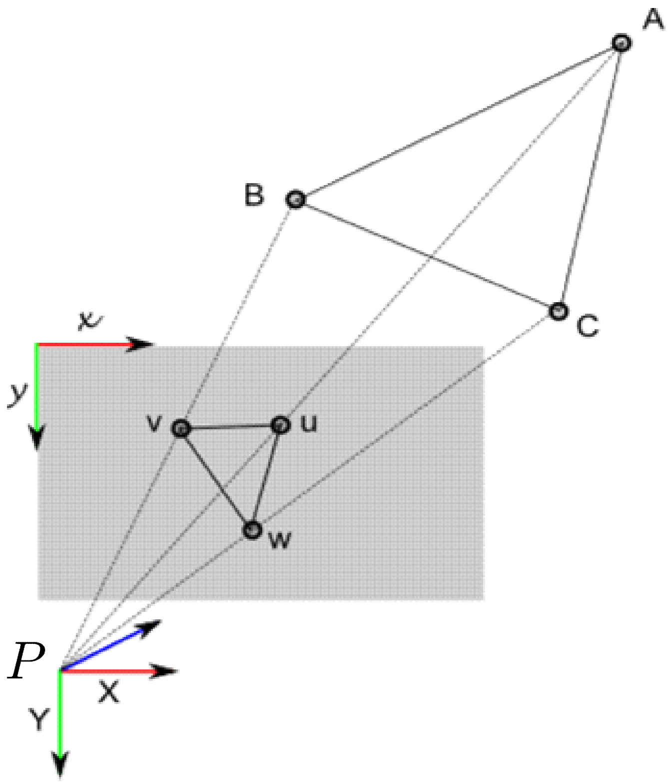
  - where $d()$ is the distance function.
  - Solved by Levenberg-Marquartdt algirthm
- Positive aspects:
  - Numerically stable
- Negative aspects:
  - Need initial guess
  - Sometimes diverge

Pose estimation

- **Hybrid approach**
  - Combine positive aspects of both algebraic algorithms and optimization algorithms:
    - Numerical stability
    - Robust, i.e. noise does not harm much
    - Speed
  - POSIT (DeMenthon & Davis. 1995)

Pose estimation

- P3P algorithm
  - Fischer and Bolles : "Perspective three-point problem"



1. 3D-2D correspondences:

$$A \leftrightarrow u, B \leftrightarrow v, C \leftrightarrow w$$

2. Depths of A,B,C are solved (Law of cosines):

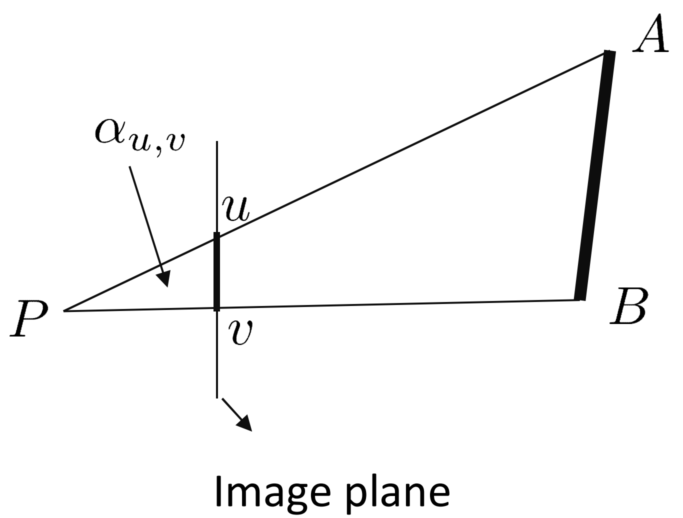
$$PA, PB, PC$$

3. Distances converted into pose configurations

$$\mathbf{R}, \mathbf{t}$$

Pose estimation

- Solving the depth - Law of cosines



$$PA^2 + PB^2 - 2PA.PB.\cos(\alpha_{u,v}) = AB^2$$

$$PA, PB, PC \leftrightarrow x_1, x_2, x_3$$
$$AB, AC, BC \leftrightarrow d_{12}, d_{13}, d_{23}$$

$$f_{i,j}(x_i, x_j) = x_i^2 + x_j^2 - 2x_i x_j \cos\theta_{ij} - d_{ij}^2 = 0$$

$$\begin{cases} f_{1,2}(x_1, x_2) = x_1^2 + x_2^2 - 2x_1x_2\cos\theta_{12} - d_{12}^2 = 0 \\ f_{1,3}(x_1, x_3) = x_1^2 + x_3^2 - 2x_1x_3\cos\theta_{13} - d_{13}^2 = 0 \\ f_{2,3}(x_2, x_3) = x_2^2 + x_3^2 - 2x_2x_3\cos\theta_{23} - d_{23}^2 = 0 \end{cases}$$

Pose estimation

- Elimination

$$\begin{cases} f_{1,2}(x_1,x_2) = x_1^2 + x_2^2 - 2x_1x_2\cos\theta_{12} - d_{12}^2 = 0 \\ f_{1,3}(x_1,x_3) = x_1^2 + x_3^2 - 2x_1x_3\cos\theta_{13} - d_{13}^2 = 0 \\ f_{2,3}(x_2,x_3) = x_2^2 + x_3^2 - 2x_2x_3\cos\theta_{23} - d_{23}^2 = 0 \end{cases}$$

$$\left.\begin{array}{l} \begin{cases} f_{1,2}(x_1,x_2) = 0 \\ \left.\begin{array}{l} f_{1,3}(x_1,x_3) = 0 \\ f_{2,3}(x_2,x_3) = 0 \end{array}\right\} \xrightarrow{\cancel{x_3}} h(x_1,x_2) = 0 \end{cases} \end{array}\right\} \xrightarrow{\cancel{x_2}} g(x_1) = 0$$

$$\Downarrow \; x = x_1^2$$

$$\boxed{g(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0}$$

Four possible solutions

Elimination of two polynomials

- Sylvester Resultant (结式）：Given two polynomials

$$f = a_0x^l + \cdots + a_l, \quad a_0 \neq 0$$
$$g = b_0x^m + \ldots + b_m, \quad b_0 \neq 0$$

The Sylvester matrix is defined as :

$$\mathrm{Syl}(f,g,x) = \begin{pmatrix} a_0 & 0 & \cdots & 0 & b_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \ddots & \vdots & b_1 & b_0 & \ddots & \vdots \\ a_2 & a_1 & \ddots & 0 & b_2 & b_1 & \ddots & 0 \\ \vdots & & \ddots & a_0 & \vdots & & \ddots & b_0 \\ & \vdots & & a_1 & & \vdots & & b_1 \\ a_{l-1} & & & & b_{m-1} & & & \\ a_l & a_{l-1} & & \vdots & b_m & b_{m-1} & & \vdots \\ 0 & a_l & \ddots & & 0 & b_m & \ddots & \\ \vdots & \ddots & \ddots & a_{l-1} & \vdots & \ddots & \ddots & b_{m-1} \\ 0 & \cdots & 0 & a_l & 0 & \cdots & 0 & b_m \end{pmatrix} \in \mathbb{R}^{(l+m)\times(l+m)}$$

The Sylvester resultant is computed : $\mathrm{Res}(f,g,x) = \det(Syl(f,g,x))$

Elimination of two polynomials

- The resultant $\mathrm{Res}(f, g, x)$ is also an integer polynomial in the coefficients of f and g.
- f and g have a common factor if and only if $\mathrm{Res}(f, g, x) = 0$

$$\mathrm{Res}(f, g, x) = \det(Syl(f, g, x)) = 0$$

- If two polynomials have a common factor, it means that they have common roots.

Elimination of two polynomials

- Example I:
  - Consider two polynomials

$$f(x) = x^5 - 3x^4 - 2x^3 + 3x^2 + 7x + 6$$

$$g(x) = x^4 + x^2 + 1$$

$$\mathrm{Res}(f, g, x) = \begin{vmatrix} 1 & -3 & -2 & 3 & 7 & 6 & 0 & 0 & 0 \\ 0 & 1 & -3 & -2 & 3 & 7 & 6 & 0 & 0 \\ 0 & 0 & 1 & -3 & -2 & 3 & 7 & 6 & 0 \\ 0 & 0 & 0 & 1 & -3 & -2 & 3 & 7 & 6 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{vmatrix} = 0$$

*In fact, the common factor of f and g is $x^2 + x + 1$

Elimination of two polynomials

- Example II:
  - Consider the following polynomials

$$f = x^2y - 3xy^2 + x^2 - 3xy$$
$$g = x^3y + x^3 - 4y^2 - 3y + 1$$

  - Compute $\mathrm{Res}(f, g, x)$

Elimination of two polynomials

- Compute $\mathrm{Res}(f, g, x)$

$$f = (y+1)x^2 - 3y(y+1)x$$

$$g = (y+1)x^3 + (y+1)(-4y+1)$$

$$\mathrm{Syl}(f,g,x) = \begin{bmatrix} (y+1) & -(3y^2+3y) & 0 & 0 & 0 \\ 0 & (y+1) & -(3y^2+3y) & 0 & 0 \\ 0 & 0 & (y+1) & -(3y^2+3y) & 0 \\ (y+1) & 0 & 0 & (-4y^2-3y+1) & 0 \\ 0 & (y+1) & 0 & 0 & (-4y^2-3y+1) \end{bmatrix}^T$$

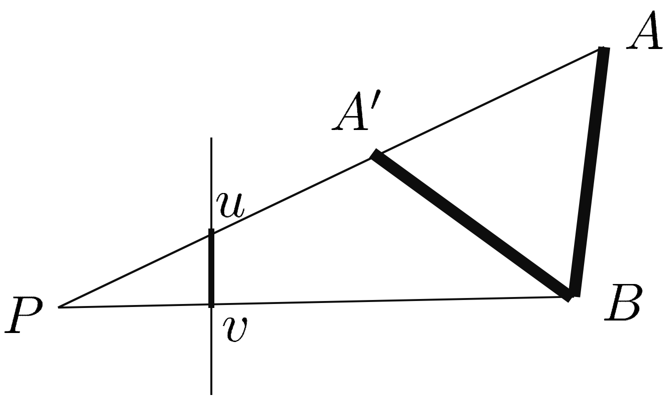
$$\begin{aligned} \mathrm{Res}(f,g,x) &= \det(\mathrm{Syl}(f,g,x)) \\ &= -(y+1)^5(4y-1)(27y^3-4y+1) \end{aligned}$$

$f(x,y) = 0$ and $g(x,y) = 0$ if and only if

$$(y+1)^5(4y-1)(27y^3-4y+1) = 0$$

Pose estimation

- Depth ambiguity (multiple solutions)



There are **four solutions** in maximum.

One additional point can eliminate the ambiguity!

- Extract four triangles out of the four points, this gives you 16 solutions at maximum, then merge these and you have a pose .
- New problem: Merging results (finding the common root) can be very difficult and expensive.

PnP algorithm

- PnP algorithm (Quan & Lan. 1999):
  - For three points, we have

$$\begin{cases} f_{1,2}(x_1, x_2) = 0 \\ f_{1,3}(x_1, x_3) = 0 \\ f_{2,3}(x_2, x_3) = 0 \end{cases} \Rightarrow g(x) = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0 = 0$$

$$(x = x_1^2)$$

  - If we have four points, we get

$$\begin{cases} f_{1,3}(x_1, x_3) = 0 \\ f_{1,4}(x_1, x_4) = 0 \\ f_{3,4}(x_3, x_4) = 0 \end{cases} \Rightarrow g'(x) = a'_4 x^4 + a'_3 x^3 + a'_2 x^2 + a'_1 x + a'_0 = 0$$

$$\begin{cases} f_{1,2}(x_1, x_2) = 0 \\ f_{1,4}(x_1, x_4) = 0 \\ f_{2,4}(x_2, x_4) = 0 \end{cases} \Rightarrow g''(x) = a''_4 x^4 + a''_3 x^3 + a''_2 x^2 + a''_1 x + a''_0 = 0$$

PnP algorithm

- Four points lead to three equations:

$$g(x) = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0 = 0$$

$$g'(x) = a'_4 x^4 + a'_3 x^3 + a'_2 x^2 + a'_1 x + a'_0 = 0$$

$$g''(x) = a''_4 x^4 + a''_3 x^3 + a''_2 x^2 + a''_1 x + a''_0 = 0$$

- Written in matrix form

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ a'_0 & a'_1 & a'_2 & a'_3 & a'_4 \\ a'_0 & a'_1 & a'_2 & a'_3 & a'_4 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = \mathbf{At} = 0$$

Hey, we see the Homogenous system again!

PnP algorithm

- We try to solve the homogenous system first.

$$\mathbf{A}\mathbf{t} = 0$$

$$(\mathbf{A} \in \mathbb{R}^{3\times5}, \mathbf{t} \in \mathbb{R}^{5\times1})$$

- SVD to find the two bases of the null space.

$$\mathbf{A} = \mathbf{U}_{3\times5}\mathrm{diag}(\sigma_1, \sigma_2, \sigma_3, \boxed{0, 0})(\mathbf{v}_1, \ldots, \boxed{\mathbf{v}_4, \mathbf{v}_5})^T$$

$$\mathbf{t} = \lambda\mathbf{v}_4 + \rho\mathbf{v}_5$$

$$t_i t_j = t_k t_l, (i + j = k + l)$$

(i,j,k,l)
(4, 2, 3, 3)
(4, 1, 3, 2)
(4, 0, 3, 1)
(4, 0, 2, 2)
(3, 1, 2, 2)
(3, 0, 2, 2)
(2, 0, 1, 1)

PnP algorithm

- Those constraints form an overdetermined system

$$\begin{bmatrix} b_1 & b_2 & b_3 \\ b_1' & b_2' & b_3' \\ \dots & & \\ b_1^{(7)} & b_2^{(7)} & b_3^{(7)} \end{bmatrix} \begin{bmatrix} \lambda^2 \\ \lambda\rho \\ \rho^2 \end{bmatrix} = \mathbf{B}\mathbf{y} = 0$$

- SVD again to get the solution

$$\mathbf{B} = \mathbf{U}_{7\times 3}\mathrm{diag}(\sigma_1, \sigma_2, \boxed{\sigma_3})(\mathbf{v}_1, \dots, \boxed{\mathbf{v}_3})^T$$

$$\hat{\mathbf{y}} = \mathbf{v}_3$$

$$\lambda/\rho = y_1/y_2$$
$$\lambda/\rho = y_2/y_3$$

- To get $\mathbf{t}$ and x

$$\mathbf{t} = \lambda \mathbf{v}_4 + \rho \mathbf{v}_5$$

$$x = t_2/t_1 \text{ or } t_3/t_2 \text{ ,or } t_4/t_3, \text{ or } t_5/t_4$$

$$x_1 = \sqrt{x}$$

Extract the pose

- From the depth value to the extrinsic parameters:

$$x_1, x_2, x_3, x_4 \rightarrow \mathbf{R}, \mathbf{t}$$

- 3D coordinates from the depth

$$^{C}\mathbf{x}_i \leftarrow x_i \mathbf{K}^{-1} \mathbf{m}_i$$

$$\Big\Updownarrow \; \mathbf{R}, \mathbf{t}$$

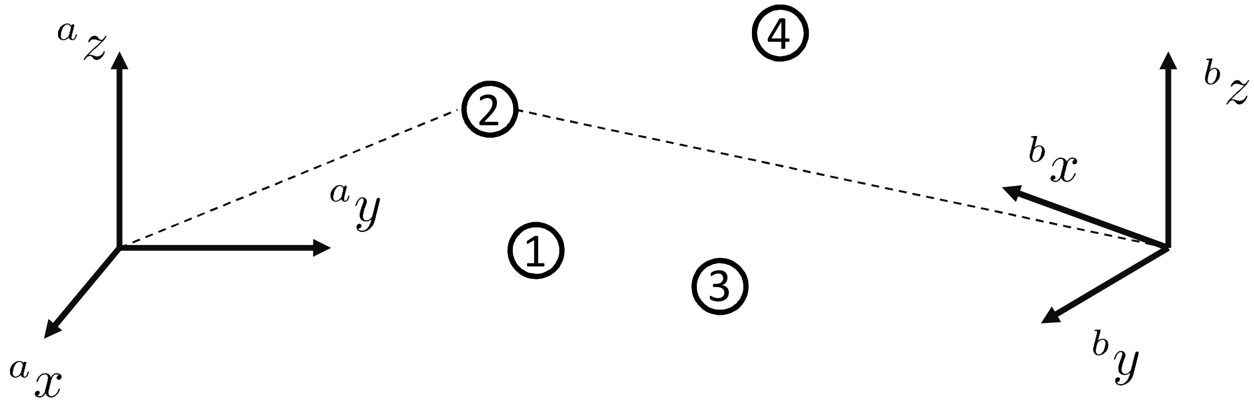
$$^{W}\mathbf{x}_i$$

$$\text{depth} \rightarrow \text{camera coordinates} \leftrightarrow \text{world coordinates}$$

Absolute orientation

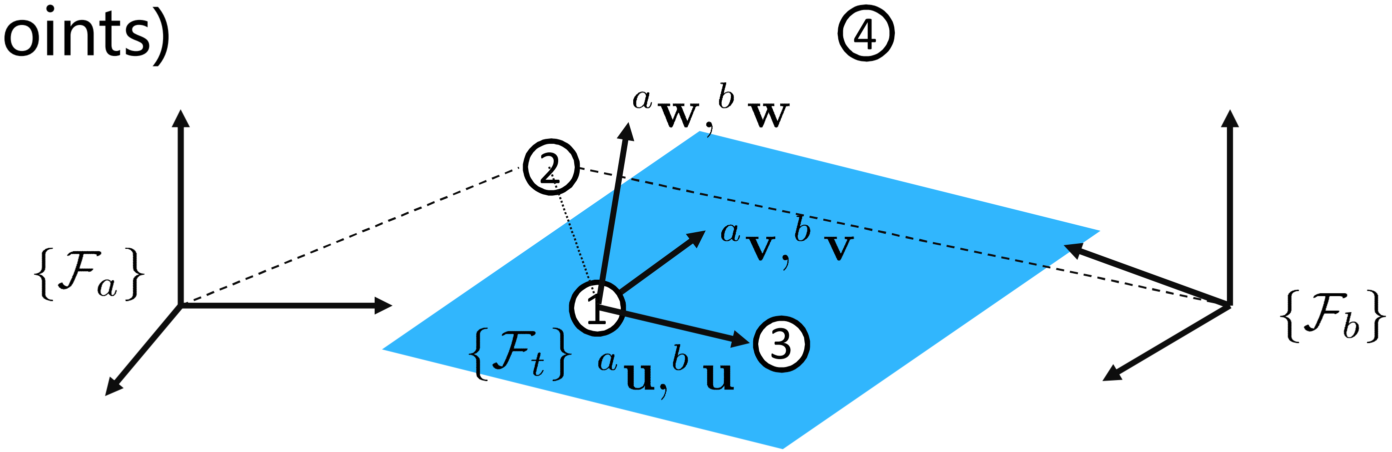
- Given a set of 3D correspondences in different coordinates systems $\{\mathcal{F}_a\}$, $\{\mathcal{F}_b\}$: $\{{}^a\mathbf{x}_i\} \leftrightarrow \{{}^b\mathbf{x}_i\}$
- Absolute orientation is to find a rigid transformation that

$$ {}^a\mathbf{x}_i = {}^a_b\mathbf{R} \cdot {}^b\mathbf{x}_i + {}^a\mathbf{t}_b $$



Absolute orientation

- Step I – Find orientation (by randomly select three points)



$$ {}^a\mathbf{u} = \frac{({}^a\mathbf{x}_3 - {}^a\mathbf{x}_1)}{\|{}^a\mathbf{x}_3 - {}^a\mathbf{x}_1\|} $$

$$ {}^a\mathbf{w} = \frac{({}^a\mathbf{x}_2 - {}^a\mathbf{x}_1)}{\|{}^a\mathbf{x}_2 - {}^a\mathbf{x}_1\|} \times {}^a\mathbf{u} $$

$$ {}^a\mathbf{v} = {}^a\mathbf{w} \times {}^a\mathbf{u} $$

For a direction vector ${}^a\mathbf{r}$ in $\{\mathcal{F}_a\}$, its direction in $\{\mathcal{F}_t\}$ is given by

$$ {}^t\mathbf{r} = \begin{bmatrix} {}^a\mathbf{u}^T \\ {}^a\mathbf{v}^T \\ {}^a\mathbf{w}^T \end{bmatrix} {}^a\mathbf{r} $$

$$ \left.\begin{aligned} {}^t_a\mathbf{R} &= \left[{}^a\mathbf{u}|{}^a\mathbf{v}|{}^a\mathbf{w}\right]^T \\ {}^t_b\mathbf{R} &= \left[{}^b\mathbf{u}|{}^b\mathbf{v}|{}^b\mathbf{w}\right]^T \end{aligned}\right\} {}^a_b\mathbf{R} = ({}^t_a\mathbf{R})^T \, {}^t_b\mathbf{R} $$

Absolute orientation

- Step II – its trivial to find translation after rotation is found

$$ {}^{a}\mathbf{X}_i = {}^{a}_{b}\mathbf{R} \cdot {}^{b}\mathbf{X}_i + {}^{a}\mathbf{t}_b $$

$$ {}^{a}\hat{\mathbf{t}}_b = \frac{1}{N} \sum_i ({}^{a}\mathbf{X}_i - {}^{a}_{b}\mathbf{R} \cdot {}^{b}\mathbf{X}_i) $$

Absolute orientation

- Is there a better solution for step I:
  - The rotation transform depends on the selection of the triad $\{\mathcal{F}_t\}$.
  - Different selection leads to different transformation.
- A better solution is put all the corresponding points into consideration and maximize the correlation :

$$({}^b\mathbf{r})^T({}^b\mathbf{r}')$$

- where ${}^b\mathbf{r}' = {}^b_a\mathbf{R}\,{}^a\mathbf{r}$.

About rotation

- Let’s recall the matrix representation of rotation

$$\mathbf{R} \in \mathbb{R}^{3\times 3}$$

- Nine parameters with the following constraints:
  - Orthonormal : $\mathbf{R}^T\mathbf{R} = \mathbf{I}$
  - Right hand : $\det \mathbf{R} = 1$
- Number of parameters are much larger than the degree of freedom (3)
- Those constraints are nonlinear, making the problem difficult to solve.

Unit quaternion

- Another elegant approach to represent a rotation transformation is using an **unit quaternion**.
  - **Four** parameters instead of **nine** parameters
  - The orthonormal issue can be easily addressed
  - Is still convenient for coordinate transformation

Quaternion

- What is a quaternion?
  - A quaternion is a vector with four components:
  - A composite of a scalar and three different imaginary parts
- Mathematically,

$$\mathbf{q} = q_w + \underbrace{q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}}_{\text{Imaginary part}}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

$$\mathbf{ij} = \mathbf{k}, \mathbf{jk} = \mathbf{i}, \mathbf{ki} = \mathbf{j}$$

Story of quaternion

- A letter Hamilton wrote later to his son Archibald:
  - Every morning in the early part of October 1843, on my coming down to breakfast, your brother William Edward and yourself used to ask me: "**Well, Papa, can you multiply triples**?" Whereto I was always obliged to reply, with a sad shake of the head, "**No, I can only add and subtract them**."

Here as he walked by on the 16th of October 1843. Sir William Rowan Hamilton in a flash of genius discovered the fundamental formula for quaternion multiplication

$$i^2 = j^2 = k^2 = ijk = -1$$

& cut it on a stone of this bridge.

Quaternion

- General quaternions :

$$\mathbf{q} = q_w + \mathbf{q}_v = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix}$$

- Real quaternions :

$$q_w = \begin{bmatrix} q_w \\ \mathbf{0}_v \end{bmatrix}$$

- Pure quaternions (3D vectors):

$$\mathbf{q}_v = \begin{bmatrix} 0 \\ \mathbf{q}_v \end{bmatrix}$$

Quaternion

- The sum of two quaternion

$$\mathbf{p}+\mathbf{q}=\begin{bmatrix}p_w\\p_x\\p_y\\p_z\end{bmatrix}+\begin{bmatrix}q_w\\q_x\\q_y\\q_z\end{bmatrix}=\begin{bmatrix}p_w+q_w\\p_x+q_x\\p_y+q_y\\p_z+q_z\end{bmatrix}$$

- It is commutative and associative

$$\mathbf{p}+\mathbf{q}=\mathbf{q}+\mathbf{p}$$
$$\mathbf{p}+(\mathbf{q}+\mathbf{r})=(\mathbf{p}+\mathbf{q})+\mathbf{r}$$

Quaternion

- Product:

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix}$$

  - Not commutative: $\mathbf{p} \otimes \mathbf{q} \neq \mathbf{q} \otimes \mathbf{p}$
  - Associative: $(\mathbf{p} \otimes \mathbf{q}) \otimes \mathbf{r} = \mathbf{p} \otimes (\mathbf{q} \otimes \mathbf{r})$
  - Distributive over sum:

$$\mathbf{p} \otimes (\mathbf{q} + \mathbf{r}) = \mathbf{p} \otimes \mathbf{q} + \mathbf{p} \otimes \mathbf{r}$$

$$(\mathbf{p} + \mathbf{q}) \otimes \mathbf{r} = \mathbf{p} \otimes \mathbf{r} + \mathbf{q} \otimes \mathbf{r}$$

  - Bi-linear

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = [\mathbf{q}_1]_L \, \mathbf{q}_2 \qquad \text{and} \qquad \mathbf{q}_1 \otimes \mathbf{q}_2 = [\mathbf{q}_2]_R \, \mathbf{q}_1$$

Quaternion

- Left/right quaternion-product

$$[\mathbf{q}]_L = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & -q_z & q_y \\ q_y & q_z & q_w & -q_x \\ q_z & -q_y & q_x & q_w \end{bmatrix}, \qquad [\mathbf{q}]_R = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & q_z & -q_y \\ q_y & -q_z & q_w & q_x \\ q_z & q_y & -q_x & q_w \end{bmatrix}$$

- Or briefly

$$[\mathbf{q}]_L = q_w \mathbf{I} + \begin{bmatrix} 0 & -\mathbf{q}_v^\top \\ \mathbf{q}_v & [\mathbf{q}_v]_\times \end{bmatrix}, \qquad [\mathbf{q}]_R = q_w \mathbf{I} + \begin{bmatrix} 0 & -\mathbf{q}_v^\top \\ \mathbf{q}_v & -[\mathbf{q}_v]_\times \end{bmatrix}$$

where $\mathbf{q}_v = [q_x, q_y, q_z]^T$

$$[\mathbf{a}]_\times \triangleq \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \qquad [\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b} , \quad \forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^3$$

Quaternion

- Identity : $\mathbf{q}_{\mathbf{1}} = 1 = \begin{bmatrix} 1 \\ \mathbf{0}_v \end{bmatrix}$
- Conjugate : $\mathbf{q}* = \begin{bmatrix} q_w \\ -\mathbf{q}_v \end{bmatrix}$ $\quad (\mathbf{p} \otimes \mathbf{q})^* = \mathbf{q}^* \otimes \mathbf{p}^*$
- Norm : $\|\mathbf{q}\| \triangleq \sqrt{\mathbf{q} \otimes \mathbf{q}^*} = \sqrt{\mathbf{q}^* \otimes \mathbf{q}} = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \in \mathbb{R}$
- Inverse : $\mathbf{q} \otimes \mathbf{q}^{-1} = \mathbf{q}^{-1} \otimes \mathbf{q} = \mathbf{q}_{\mathbf{1}}$

$$\mathbf{q}^{-1} = \mathbf{q}^* / \|\mathbf{q}\|^2$$

- Dot product : $\mathbf{p}^T \mathbf{q} = p_w q_w + p_x q_x + p_y q_y + p_z q_z$

$$= \mathbf{p} \otimes \mathbf{q}^* \text{ or}$$

$$= \mathbf{p}^* \otimes \mathbf{q}$$

Quaternion

- Quaternion multiplication matrix is orthogonal :

$$[\mathbf{q}]_L[\mathbf{q}]_L^T = \mathbf{I}_{4\times 4} \qquad [\mathbf{p}]_R[\mathbf{p}]_R^T = \mathbf{I}_{4\times 4}$$

- Dot product is preserved :

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

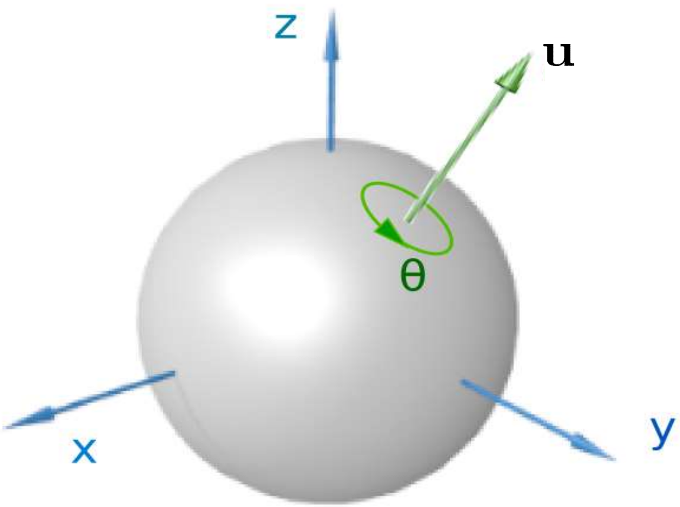
$$\mathbf{r}^T\mathbf{t} = ([\mathbf{q}]_R\mathbf{r})^T([\mathbf{q}]_R\mathbf{r}) = (\mathbf{r} \otimes \mathbf{q})^T(\mathbf{t} \otimes \mathbf{q})$$

Unit quaternion

- For unit quaternion, we have

$$\|\mathbf{q}\| = 1 \text{ and } \mathbf{q}^{-1} = \mathbf{q}^*$$

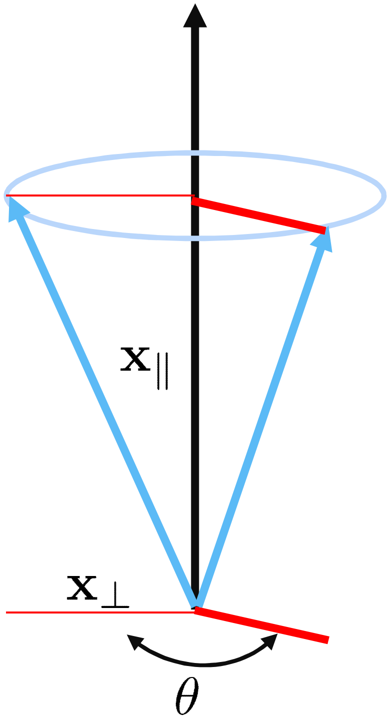
- It can always be written in the form:



$$\mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ \mathbf{u}\sin(\theta/2) \end{bmatrix}$$

Unit quaternion

- Transformation of a vector



$$\mathbf{x}' = \mathbf{q} \otimes \mathbf{x} \otimes \mathbf{q}^*$$

$$\mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ \mathbf{u}\sin(\theta/2) \end{bmatrix}$$

$$\mathbf{x}' = \mathbf{x}_{\perp}\cos(\theta) + (\mathbf{u} \times \mathbf{x})\sin(\theta) + \mathbf{x}_{\|}$$

Unit quaternion

- Convert a unit quaternion to a rotation matrix

$$\mathbf{x}' = \mathbf{q} \otimes \mathbf{x} \otimes \mathbf{q}^*$$

$$\Downarrow$$

$$\mathbf{x}' = \mathbf{R}\mathbf{x}$$

$$\Downarrow$$

$$\boxed{\mathbf{R}\{\mathbf{q}\} = (q_w^2 - \mathbf{q}_v^\top \mathbf{q}_v)\,\mathbf{I} + 2\,\mathbf{q}_v \mathbf{q}_v^\top + 2\,q_w\,[\mathbf{q}_v]_\times}$$

Unit quaternion

	Rotation matrix, $\mathbf{R}$	Quaternion, $\mathbf{q}$
Parameters	$3 \times 3 = 9$	$1 + 3 = 4$
Degrees of freedom	3	3
Constraints	6	1
Constraints	$\mathbf{R}\mathbf{R}^\top = \mathbf{I}\ ,\ \det(\mathbf{R}) = +1$	$\mathbf{q} \otimes \mathbf{q}^* = 1$
Identity	$\mathbf{I}$	1
Inverse	$\mathbf{R}^\top$	$\mathbf{q}^*$
Composition	$\mathbf{R}_1\, \mathbf{R}_2$	$\mathbf{q}_1 \otimes \mathbf{q}_2$
Rotation operator	$\mathbf{R} = \mathbf{I} + \sin\phi\,[\mathbf{u}]_\times + (1-\cos\phi)\,[\mathbf{u}]_\times^2$	$\mathbf{q} = \cos\phi/2 + \mathbf{u}\sin\phi/2$
Rotation action	$\mathbf{R}\,\mathbf{x}$	$\mathbf{q} \otimes \mathbf{x} \otimes \mathbf{q}^*$

Absolute Orientation

- Find best quaternion using all the correspondences
- We want to find the unit quaternion that **maximizes**

$$\sum_i ({}^b\mathbf{r}_i)^T \left({}^b\mathbf{r}_i\right)$$

$$= \sum_i ({}^b\mathbf{r}_i)^T (\mathbf{q} \otimes {}^b\mathbf{r}_i \otimes \mathbf{q}^*)$$

$$= \sum_i ({}^b\mathbf{r}_i \otimes \mathbf{q})^T (\mathbf{q} \otimes {}^b\mathbf{r}_i)$$

$$\mathbf{r}^T\mathbf{t} = (\mathbf{r} \otimes \mathbf{q})^T (\mathbf{t} \otimes \mathbf{q})$$

$$= \sum_i ([{}^b\mathbf{r}_i]_L \mathbf{q})^T ([{}^b\mathbf{r}_i]_R \mathbf{q})$$

$$= \sum_i \mathbf{q}^T ([{}^b\mathbf{r}_i]_L [{}^b\mathbf{r}_i]_R) \mathbf{q} = \boxed{\mathbf{q}^T \mathbf{N} \mathbf{q}}$$

Absolute Orientation

- The solution is the eigen vector with the maximum eigen value of $\mathbf{N}$.
- Algorithm of absolute orientation:
  - Step 1: Compute directions of the points in each coordinate system:

$$\begin{cases} {}^a\mathbf{r}_i = {}^a\mathbf{x}_i - {}^a\bar{\mathbf{x}}_i \\ {}^b\mathbf{r}_i = {}^b\mathbf{x}_i - {}^b\bar{\mathbf{x}}_i \end{cases}$$

  - Step 2: Construct the matrix $\mathbf{N}$ and solve the quaterion by finding the maximum eigen value.
  - Step 3: get the translation vector

Summary

- **P3P & PnP:**
  - Use law of cosine to get the depth of the three points by solving a system of polynomial equations.
  - **Resultant** can be used to solve the polynomial equations
  - PnP solve the depth of each point with unique solution
- Absolute orientation
  - Unit quaternion can be used to represent a rotation
  - AO is used to obtain the rigid transformation between two coordinate systems where the 3D points are defined.