LECTURE 19 – SCAN REGISTRATION

March 21, 2019

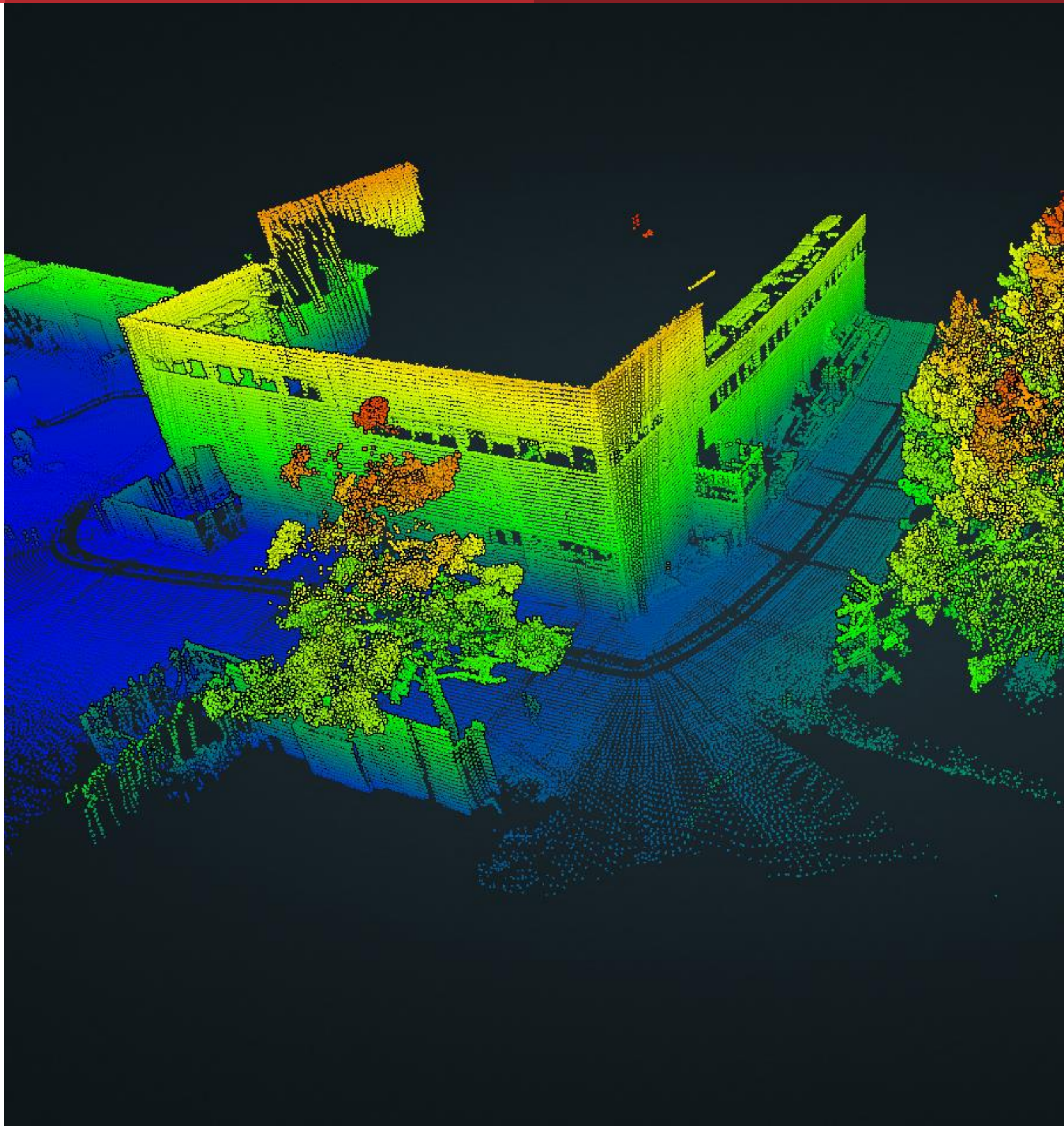
Nicholas Charron

CIVE 497/700 Smart Structure Technology

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

WAVE LABORATORY

SDIC SHAPING SMART INFRASTRUCTURE

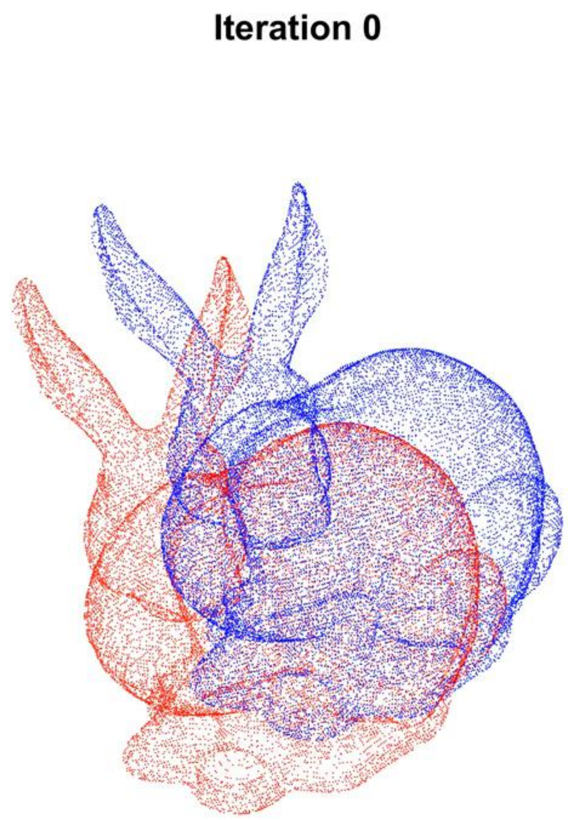


OUTLINE

- Introduction to scan registration
- Iterative closest point (ICP) algorithm
  - Conceptual idea
  - Problem formulation
- Optimization with Lie Algebra
  - Introduction to Lie Algebra
  - Methods for solving optimization problems
  - Solving ICP
- Challenges with ICP
- ICP variations/alternatives
  - Point to Plane
  - G-ICP
  - NDT
- Overview of Task 7

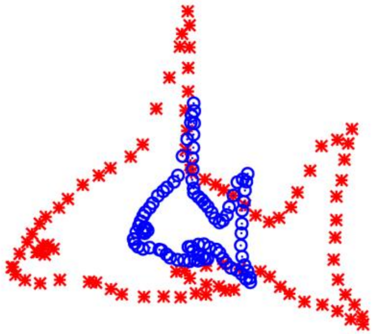
INTRODUCTION TO SCAN REGISTRATION

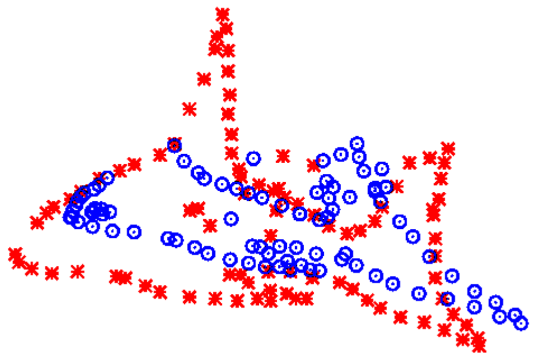
- What is scan registration?
  - Aligning one set of point (scan) to another set of points taken from a different location
  - Attempts to recover the transformation (rotation + translation) that best describes the motion underlying the two scan frames
  - Also known as: point set registration, point alignment, point matching, scan matching

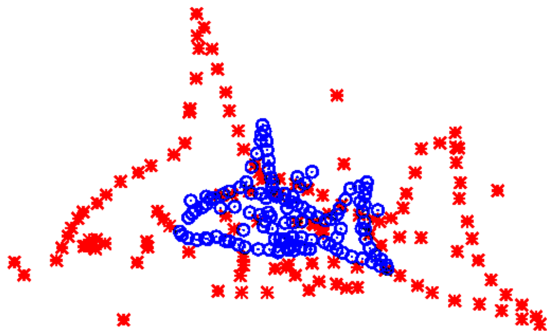


INTRODUCTION TO SCAN REGISTRATION

- What is scan registration?
  - Points can be assumed as:
    - rigid ← $\boxed{\text{Our Focus}}$
    - non-rigid
    - Not to scale

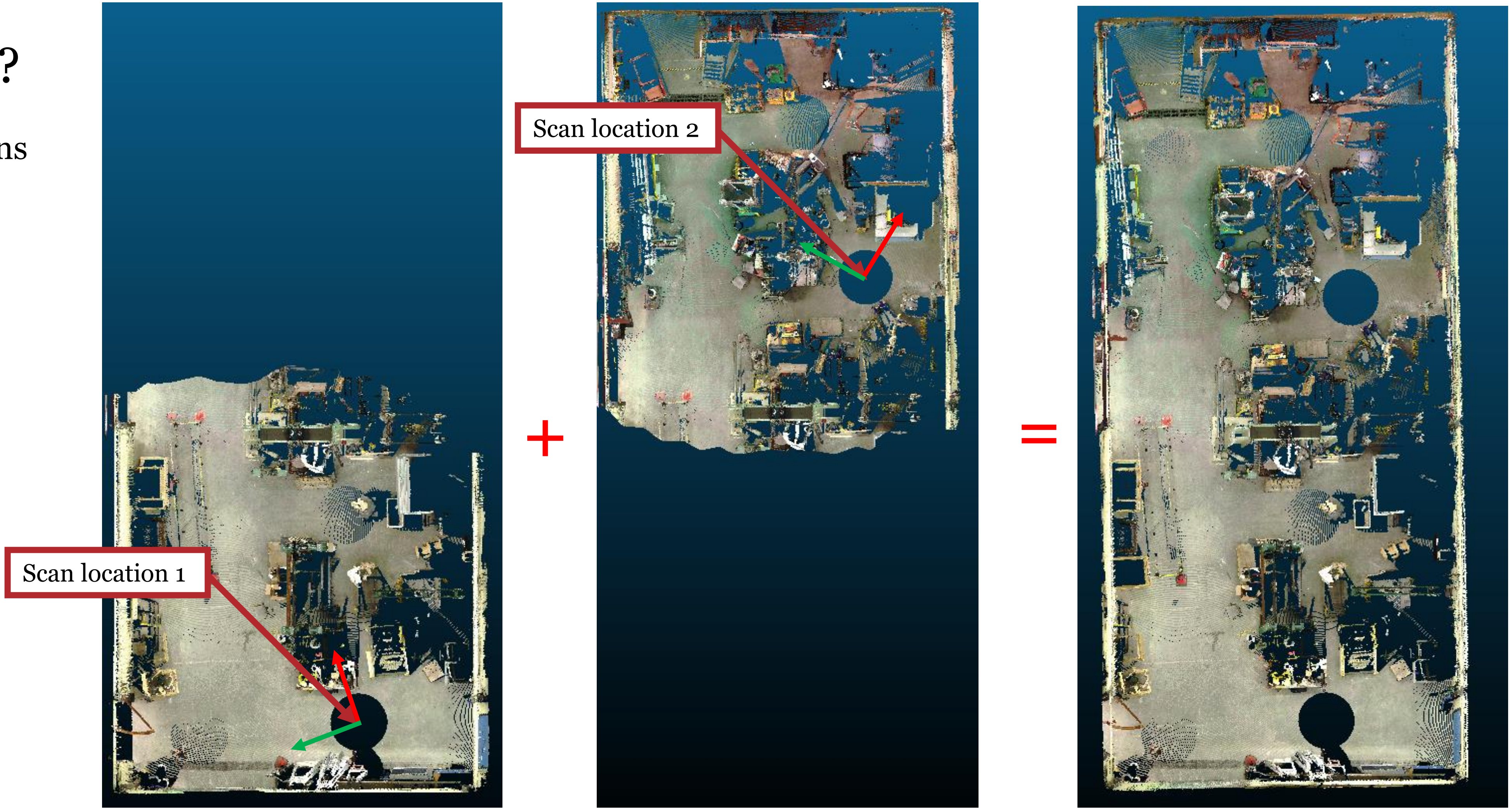






INTRODUCTION TO SCAN REGISTRATION

- Why is it used?
    1. Aggregating scans

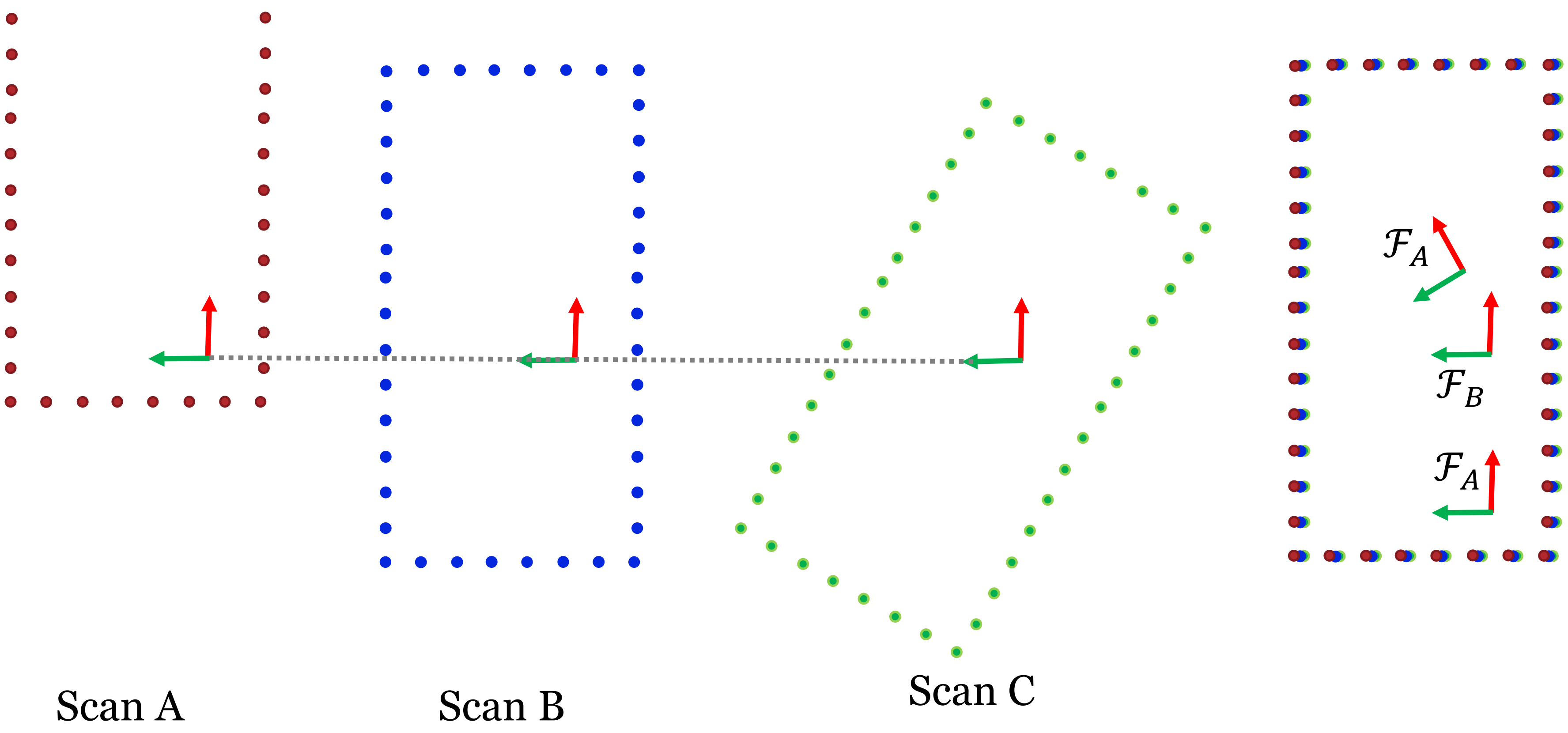


Scan 1 Scan 2 Aggregate Map

INTRODUCTION TO SCAN REGISTRATION

- Why is it used?
    2. Localization

Example: Can we recover the sensor’s trajectory from these scans?



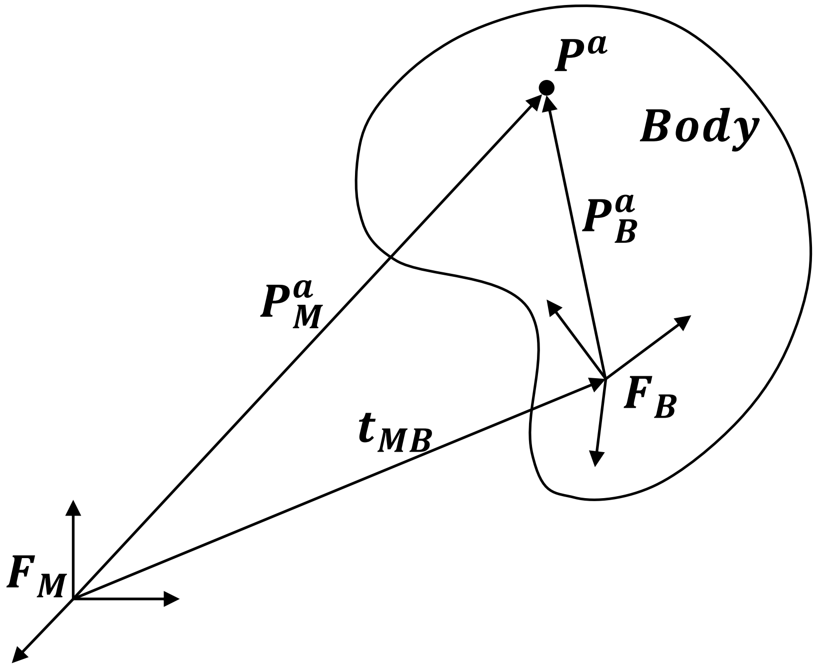
BACKGROUND THEORY | 3D GEOMETRY & NOTATION

$$\boldsymbol{P_M^a} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_M^a = \boldsymbol{R_{MB} P_B^a} + \boldsymbol{t_{MB}}$$

$$\boldsymbol{T_{MB}} = \begin{bmatrix} \boldsymbol{R_{MB}} & \boldsymbol{t_{MB}} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

$$\boldsymbol{P_M^a} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_M^a = \boldsymbol{T_{MB} P_B^a}$$

homogenous form



BACKGROUND THEORY | 3D GEOMETRY & NOTATION

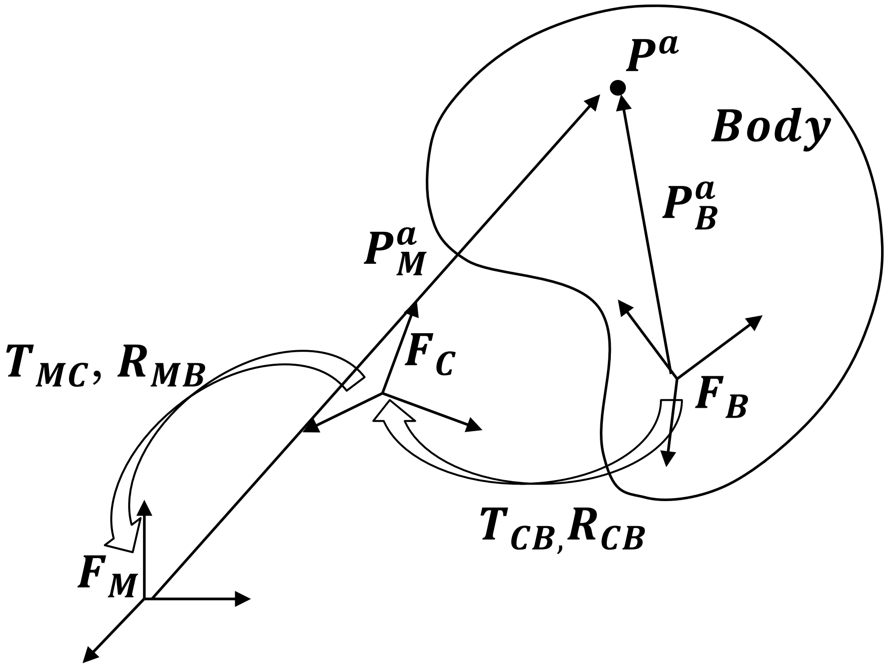
- Note on compounding rotations and transformations:

$$\boldsymbol{P_M^a} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_M^a = \boldsymbol{T_{MC} T_{CB} P_B^a}$$

Ensure subscripts align

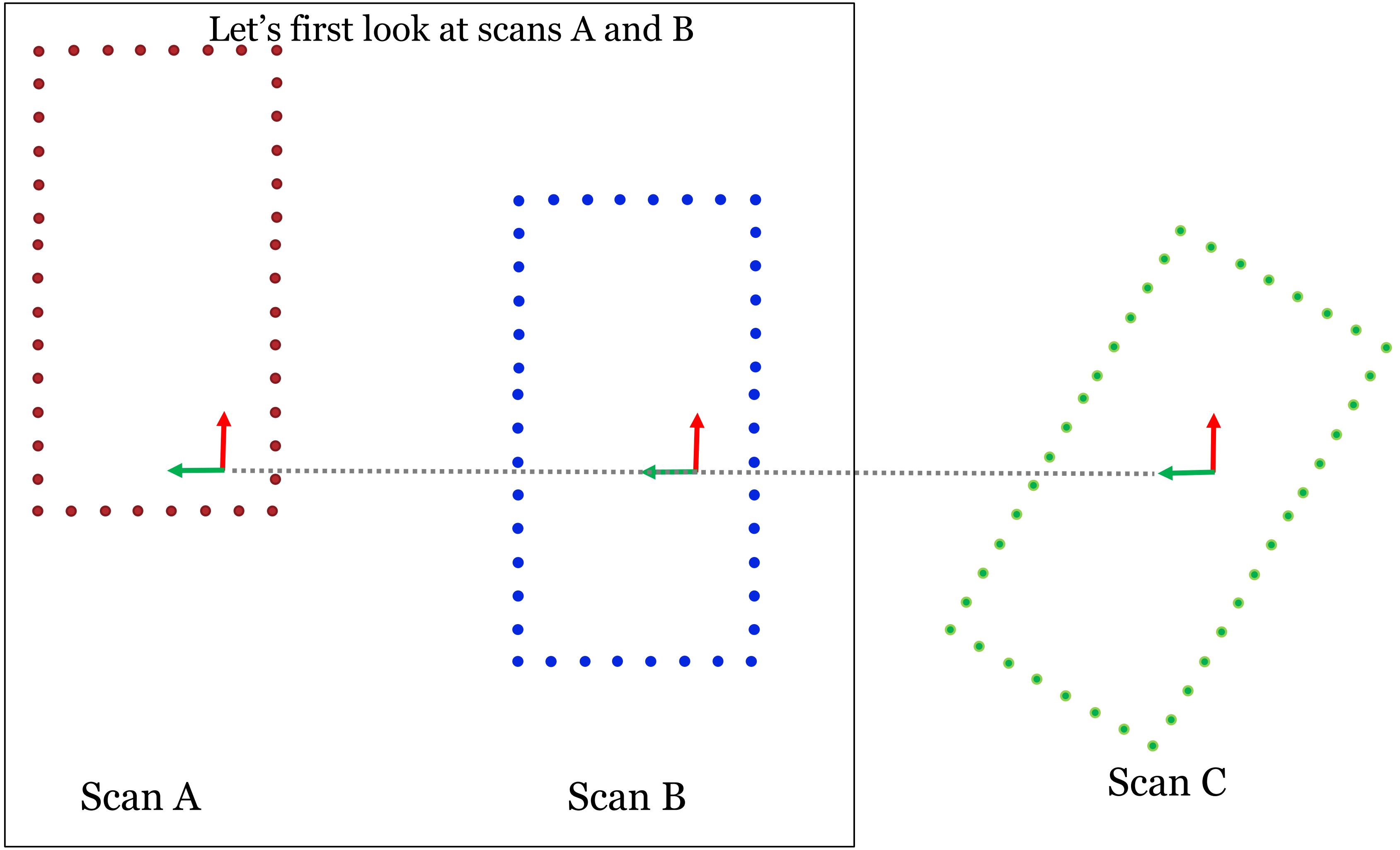
$$\boldsymbol{R_{MB}} = \boldsymbol{R_{MC} R_{CB}}$$

Same applies for rotations



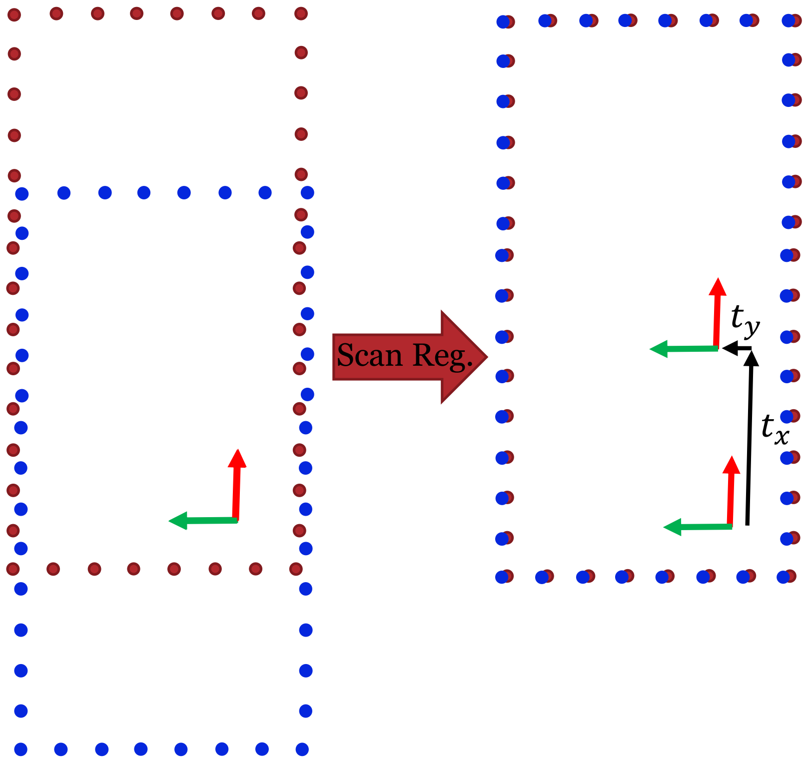
INTRODUCTION TO SCAN REGISTRATION

- Example: 2D scan registration for pose recovery (localization)



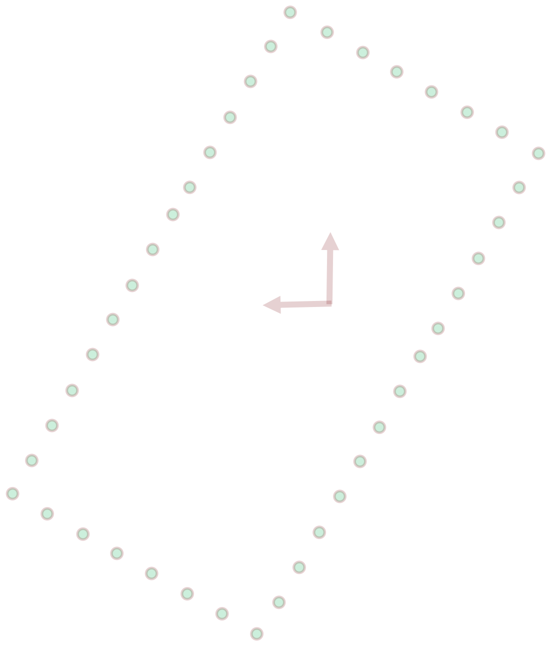
INTRODUCTION TO SCAN REGISTRATION

- Example: 2D scan registration for pose recovery (localization)



Scan A+B

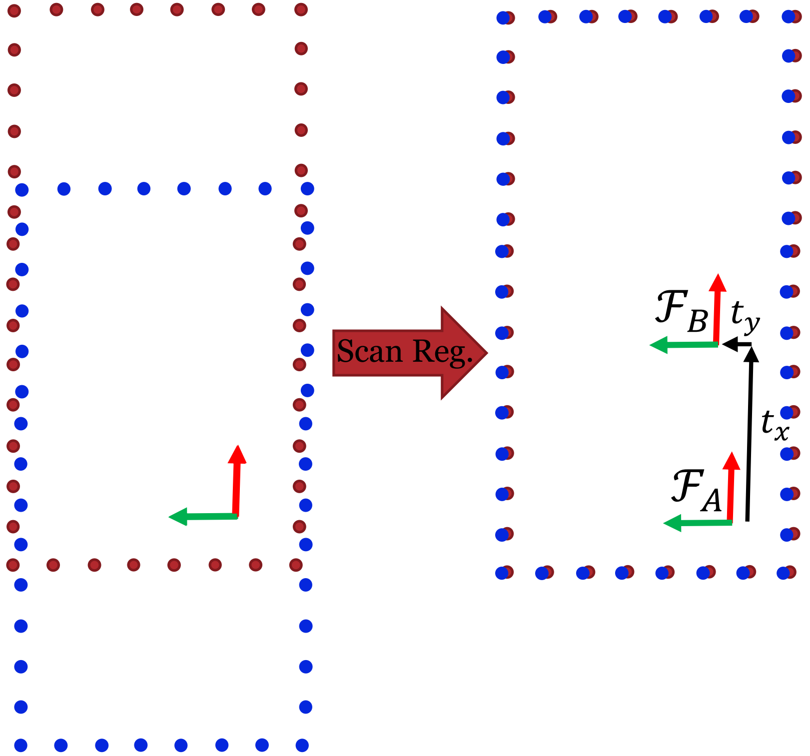
Scan A+B



Scan C

INTRODUCTION TO SCAN REGISTRATION

- Example: 2D scan registration for pose recovery (localization)

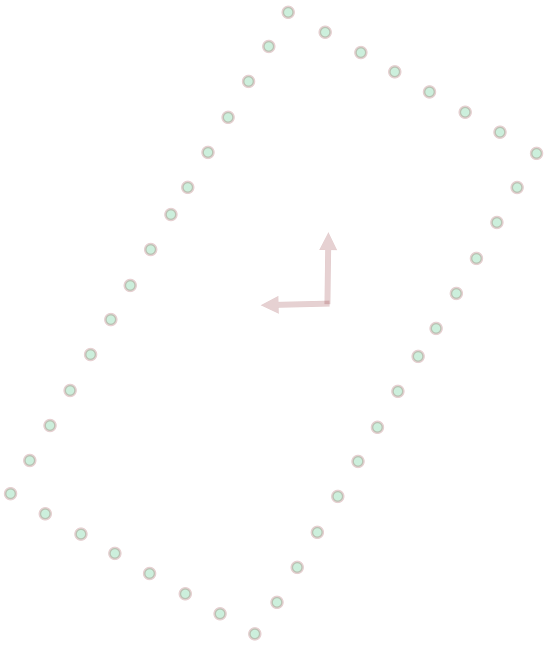


Scan A+B

Scan A+B

$$\boldsymbol{T}_{AB} = \begin{bmatrix} \boldsymbol{R}_{AB} & \boldsymbol{t}_{AB} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} & t_x \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & t_y \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

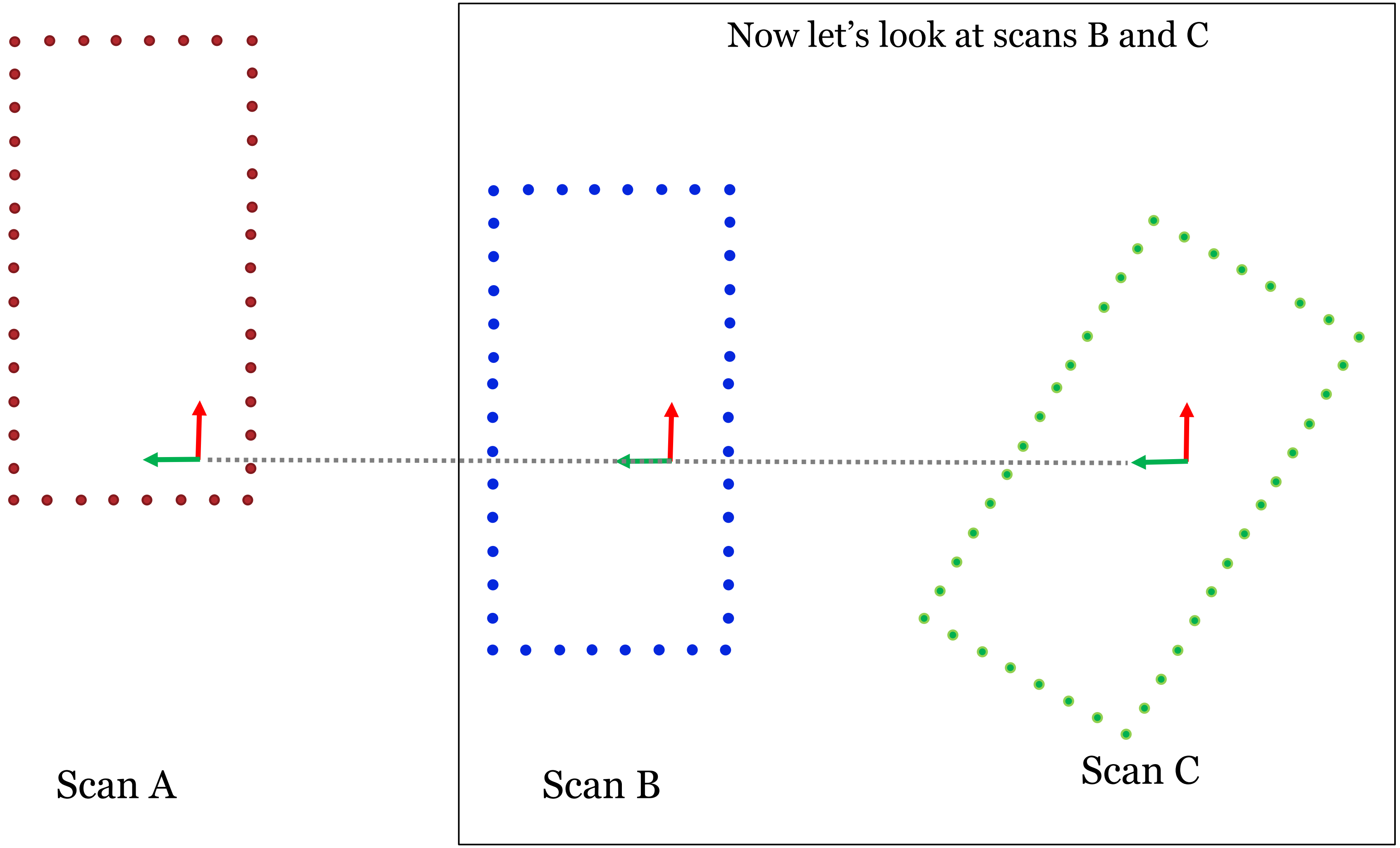
Transforms points from $\mathcal{F}_B$ to $\mathcal{F}_A$



Scan C

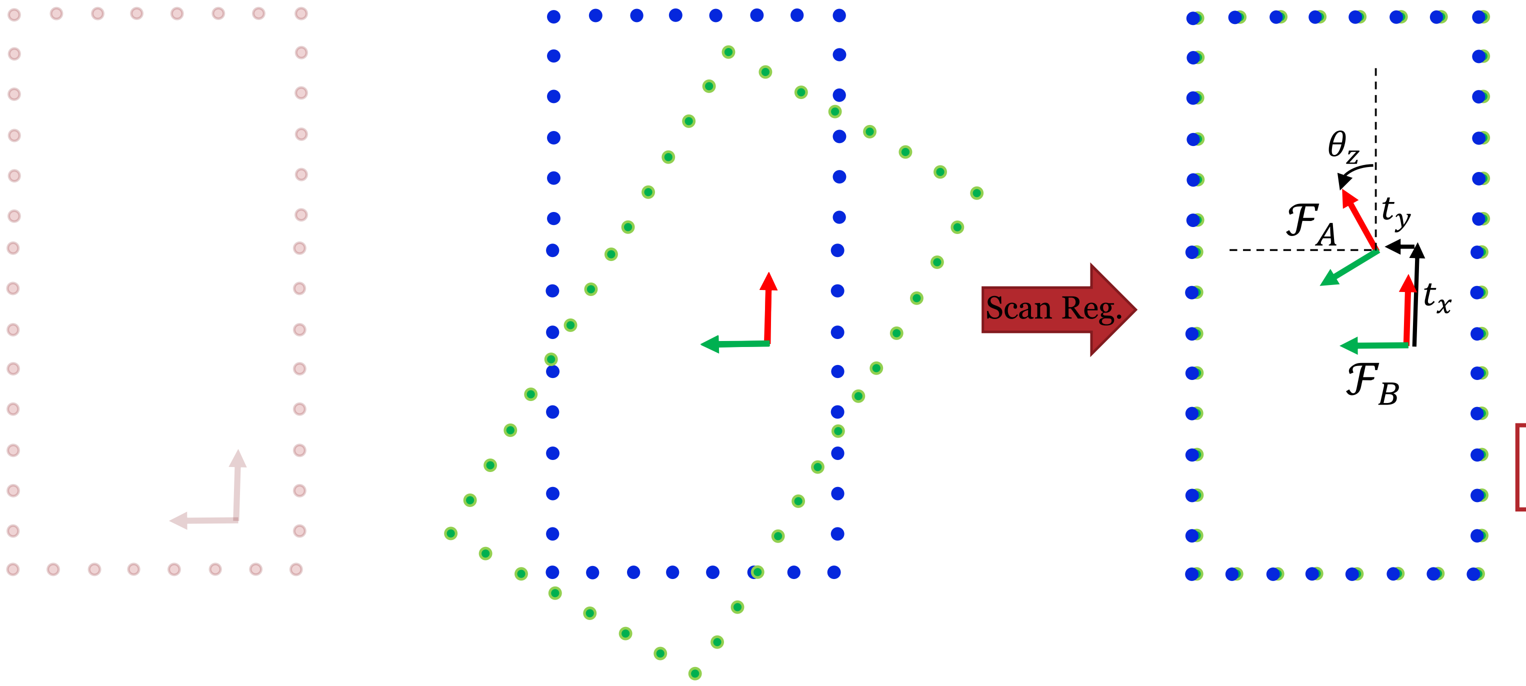
INTRODUCTION TO SCAN REGISTRATION

- Example: 2D scan registration for pose recovery (localization)



INTRODUCTION TO SCAN REGISTRATION

- Example: 2D scan registration for pose recovery (localization)



$$\boldsymbol{T_{BC}} = \begin{bmatrix} \boldsymbol{R_{BC}} & \boldsymbol{t_{BC}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & t_x \\ \sin\theta_z & \cos\theta_z & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transforms points from $\mathcal{F}_C$ to $\mathcal{F}_B$

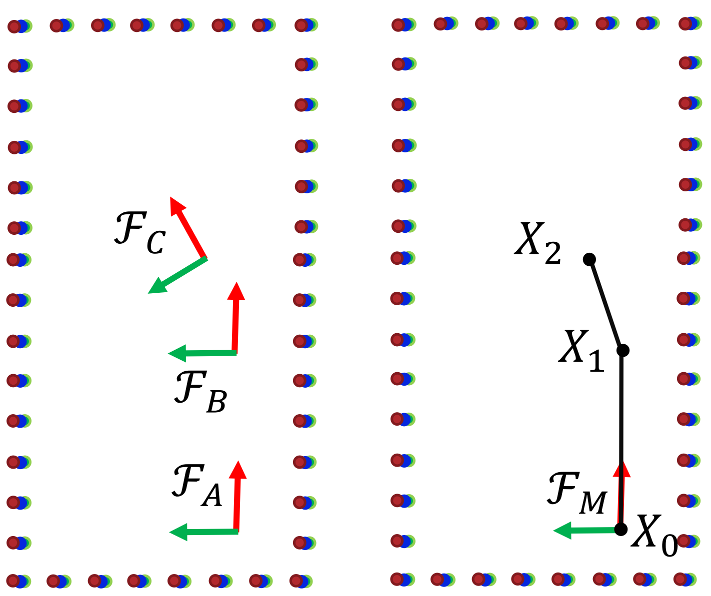
Scan A

Scan B+C

Scan B+C

INTRODUCTION TO SCAN REGISTRATION

- Example: 2D scan registration for pose recovery (localization)



Result

$$X_0 = \boldsymbol{T}_{MA}$$
$$X_1 = \boldsymbol{T}_{MB} = \boldsymbol{T}_{MA}\boldsymbol{T}_{AB}$$
$$X_2 = \boldsymbol{T}_{MC} = \boldsymbol{T}_{MA}\boldsymbol{T}_{AB}\boldsymbol{T}_{BC}$$
$$X = [X_0 \quad X_1 \quad \cdots \quad X_K]^T$$

Pose vector

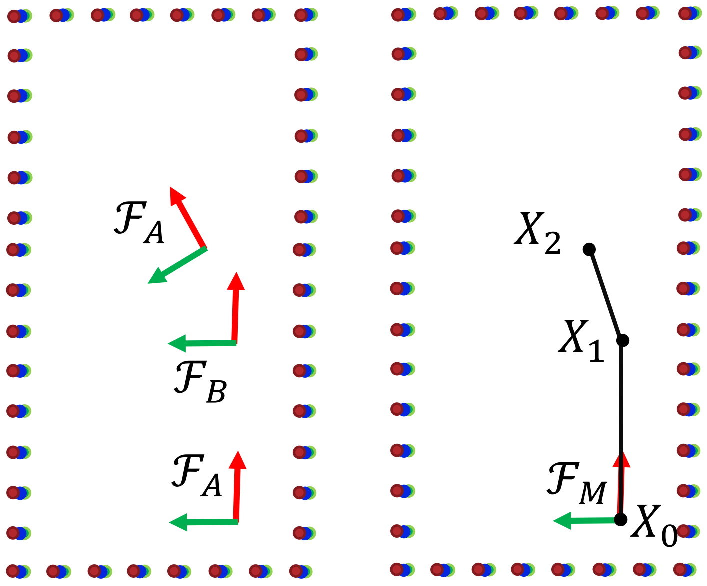
Poses can also be stored as translation + rotation (Euler angles or quaternions)

E.g.,

- Euler angles: $X_k = [\boldsymbol{t}_{x,k} \quad \boldsymbol{t}_{y,k} \quad \boldsymbol{t}_{z,k} \quad \boldsymbol{\theta}_{x,k} \quad \boldsymbol{\theta}_{y,k} \quad \boldsymbol{\theta}_{z,k}]$
- Quaternions: $X_k = [\boldsymbol{t}_{x,k} \quad \boldsymbol{t}_{y,k} \quad \boldsymbol{t}_{z,k} \quad \boldsymbol{q}_{x,k} \quad \boldsymbol{q}_{y,k} \quad \boldsymbol{q}_{z,k} \quad \boldsymbol{q}_{w,k}]$

INTRODUCTION TO SCAN REGISTRATION

- Example: 2D scan registration for pose recovery (localization)



Result

Aggregate map can also be computed:

For: $\boldsymbol{k = (1, K)}$ $\boldsymbol{n = (1, N_k)}$

Where K is the total number of scans (3 for this example), $\boldsymbol{N_k}$ is the total number of points contained in scan k.

Let: $\boldsymbol{S_k = \{P_K^1, P_K^2, \cdots P_K^N\}}$

Be the set of points contained in Scan k, expressed in frame k

$$\boldsymbol{M = \{(P_M^{1,1}, P_M^{1,2}, \cdots P_M^{1,N_1}), (P_M^{2,1}, P_M^{2,2}, \cdots P_M^{2,N_2}), \cdots (P_M^{K,1}, P_M^{K,2}, \cdots P_M^{K,N_K})\}}$$

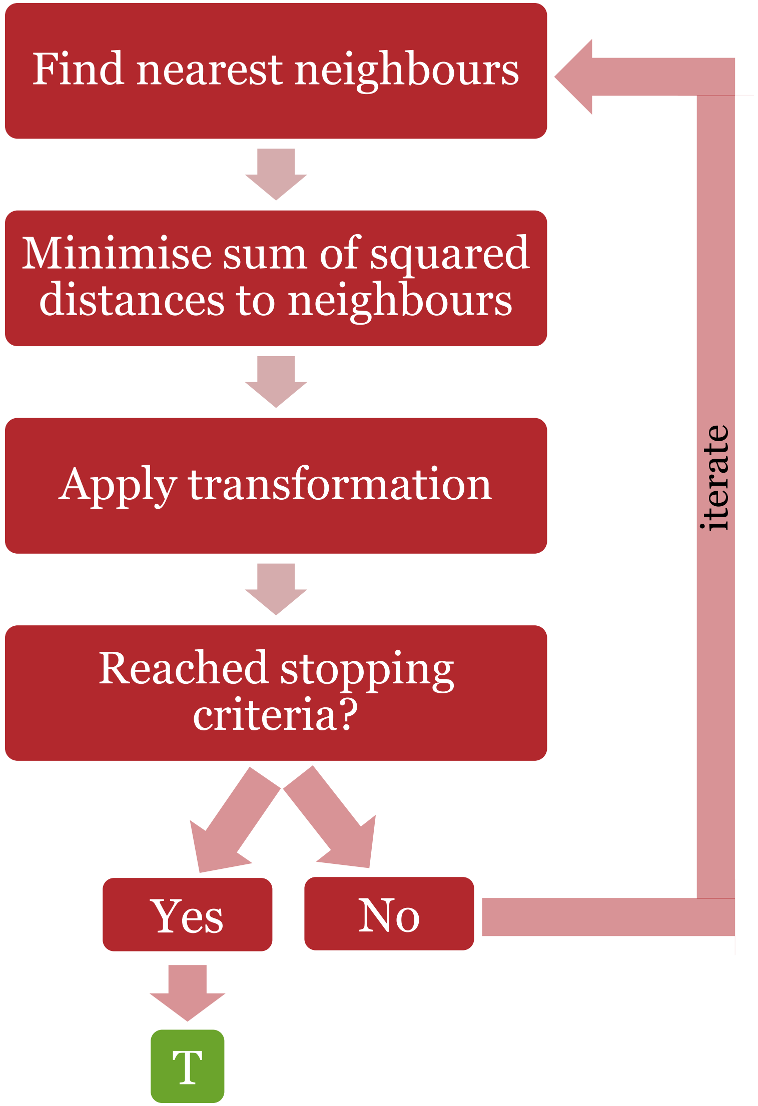
Be the aggregate map which contains all $\boldsymbol{N_k}$ points from all K scans, where each point is expressed in frame M.

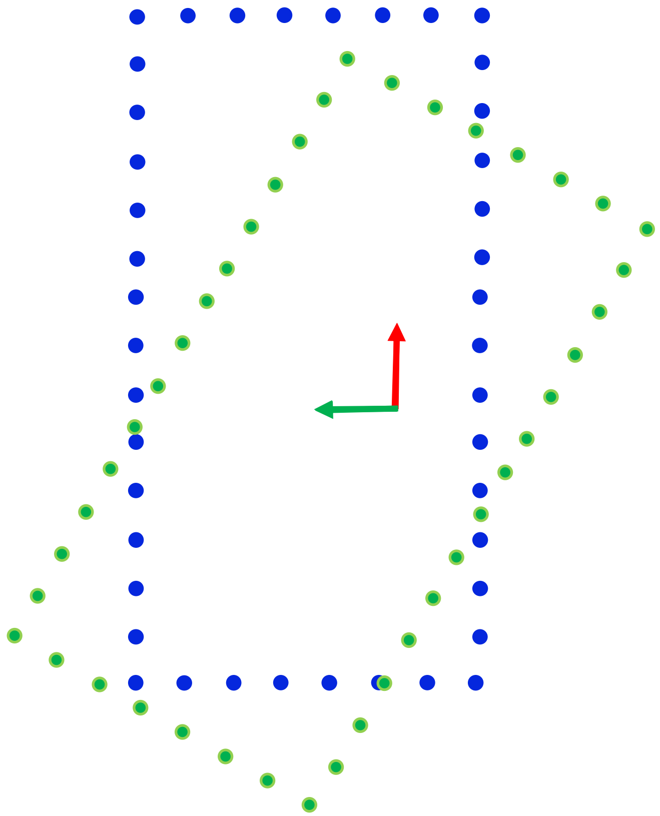
$$\boldsymbol{P_M^{k,n_k} = T_{Mk}\, P_k^{k,n_k}}$$

OUTLINE

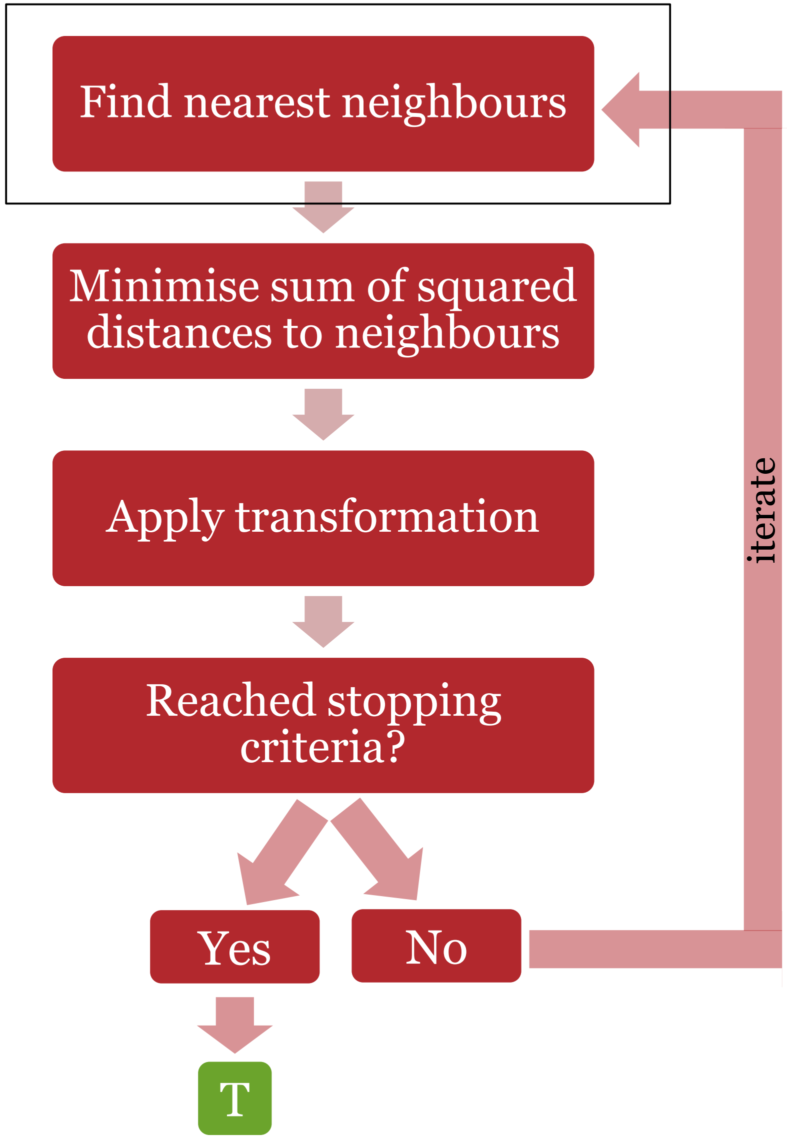
- Introduction to scan registration
- Iterative closest point (ICP) algorithm
  - Conceptual idea
  - Problem formulation
- Optimization with Lie Algebra
  - Introduction to Lie Algebra
  - Methods for solving optimization problems
  - Solving ICP
- Challenges with ICP
- ICP variations/alternatives
  - Point to Plane
  - G-ICP
  - NDT
- Overview of Task 7

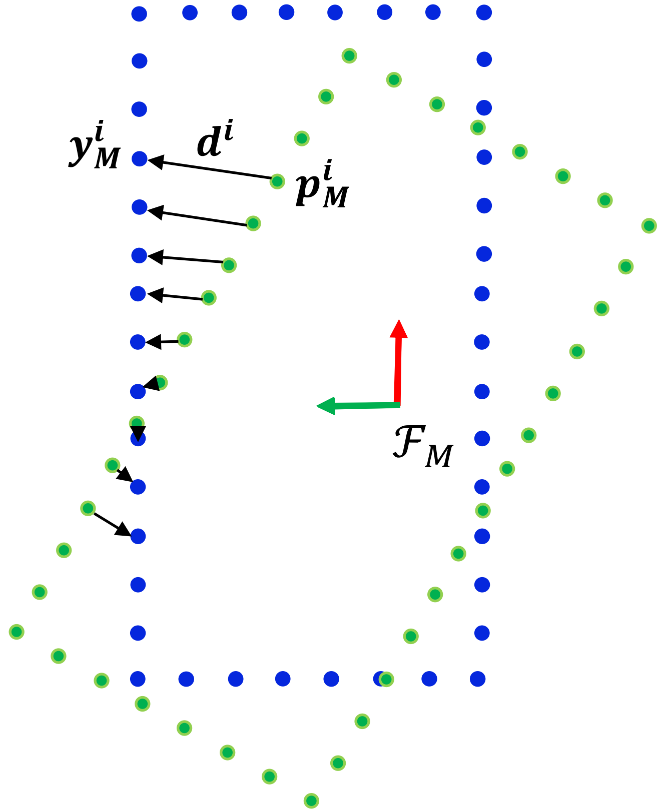
ITERATIVE CLOSEST POINT (ICP)



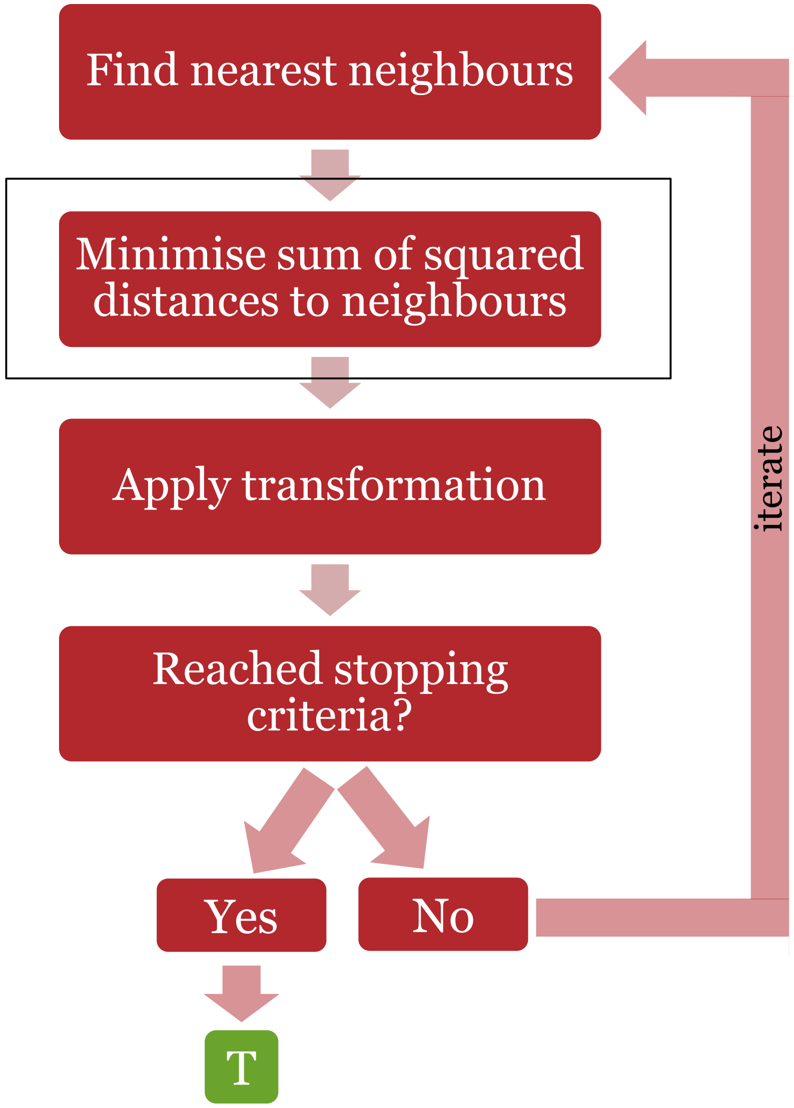


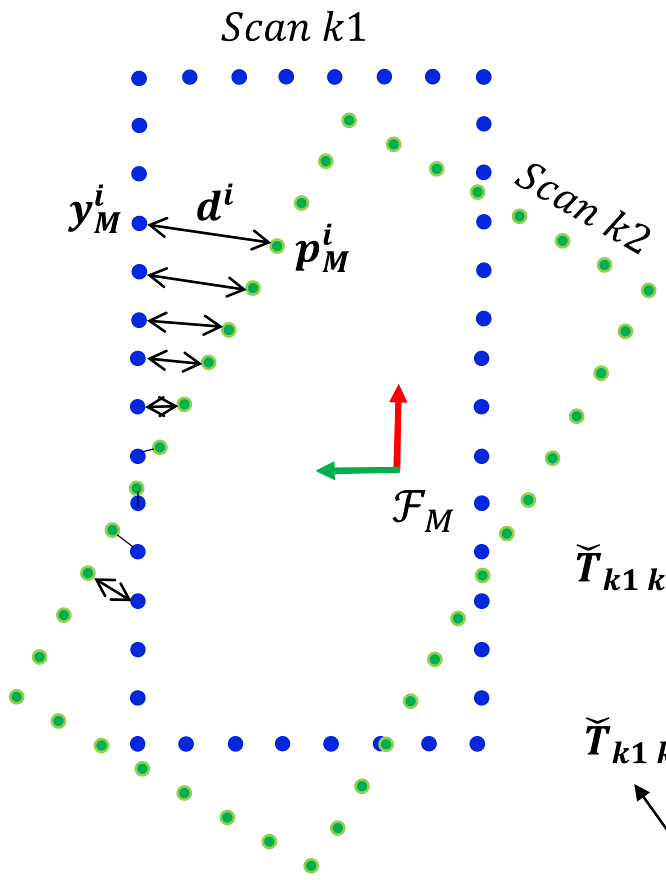
ITERATIVE CLOSEST POINT (ICP)





ITERATIVE CLOSEST POINT (ICP)



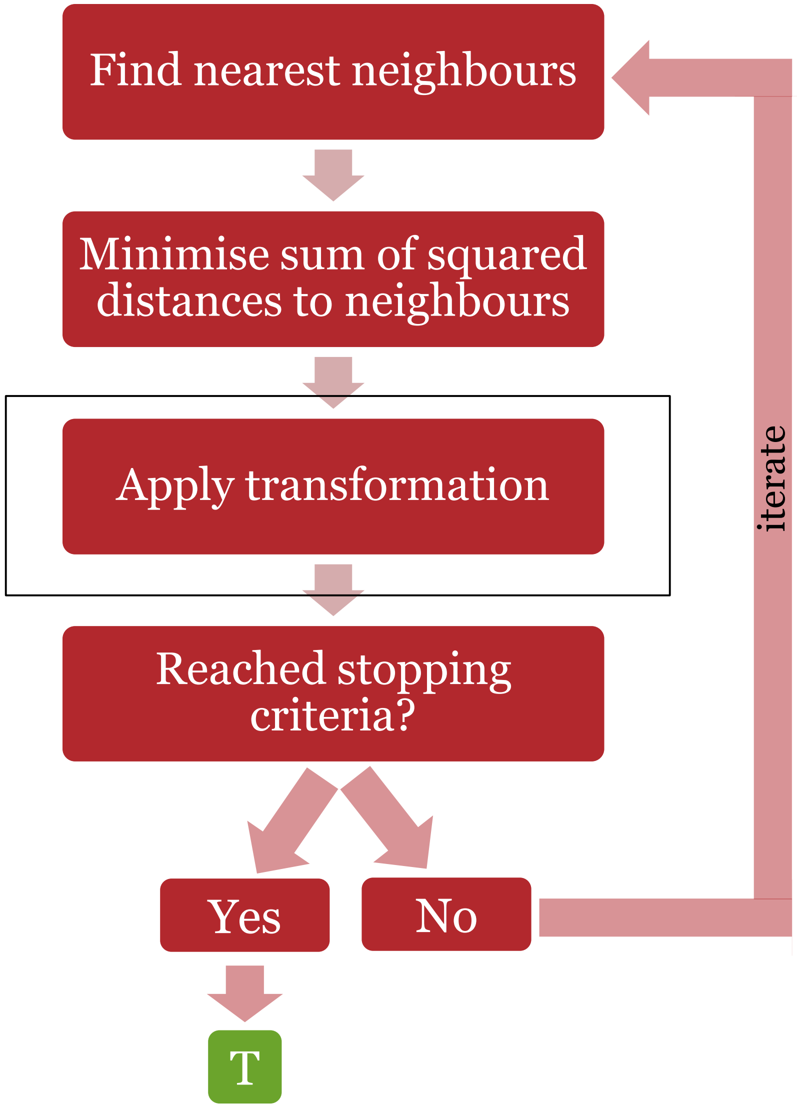


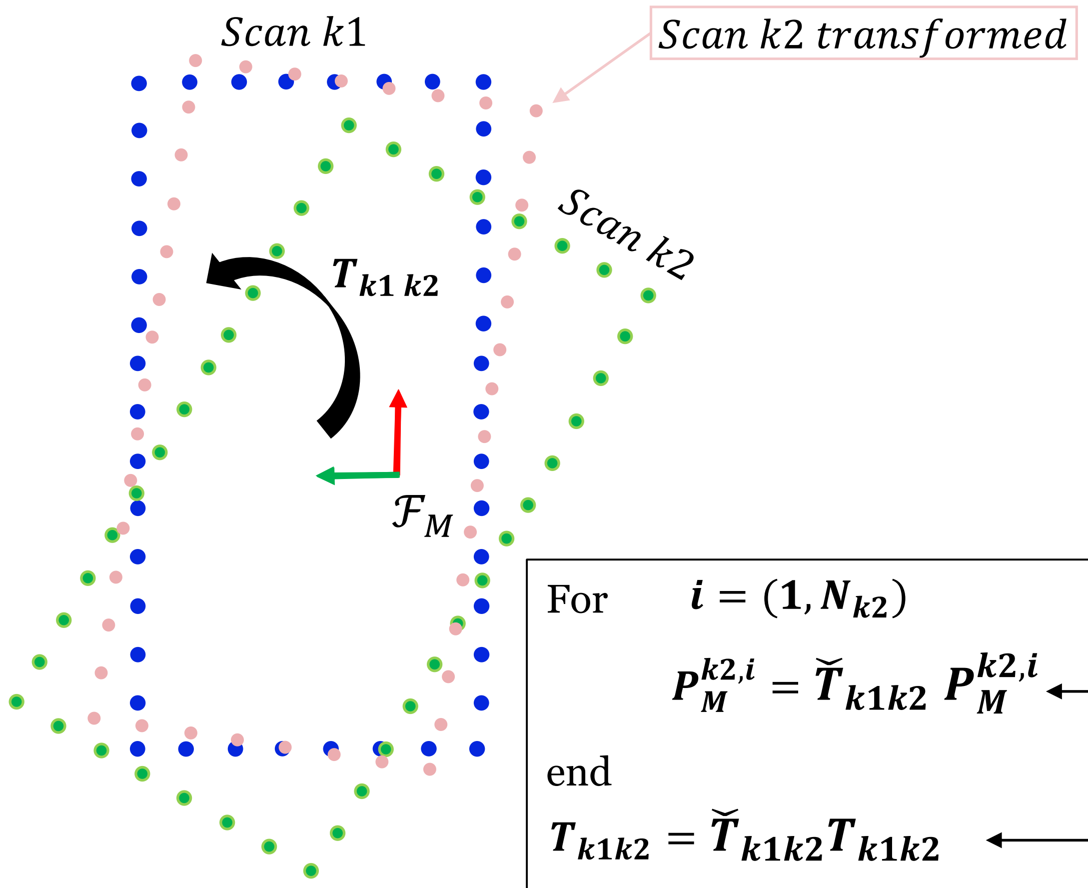
$$\breve{T}_{k1\,k2} = \underset{\breve{T}_{k1k2} \in SE(3)}{\arg\min} \frac{1}{N_{k2}} \sum_{i=1}^{N_{k2}} [d^i]^2$$

$$\breve{T}_{k1\,k2} = \underset{\breve{T}_{k1k2} \in SE(3)}{\arg\min} \frac{1}{N_{k2}} \sum_{i=1}^{N_{k2}} [y_M^i - \breve{T}_{k1k2}\, p_M^i]^2$$

Estimate of the transformation for the current iteration

ITERATIVE CLOSEST POINT (ICP)





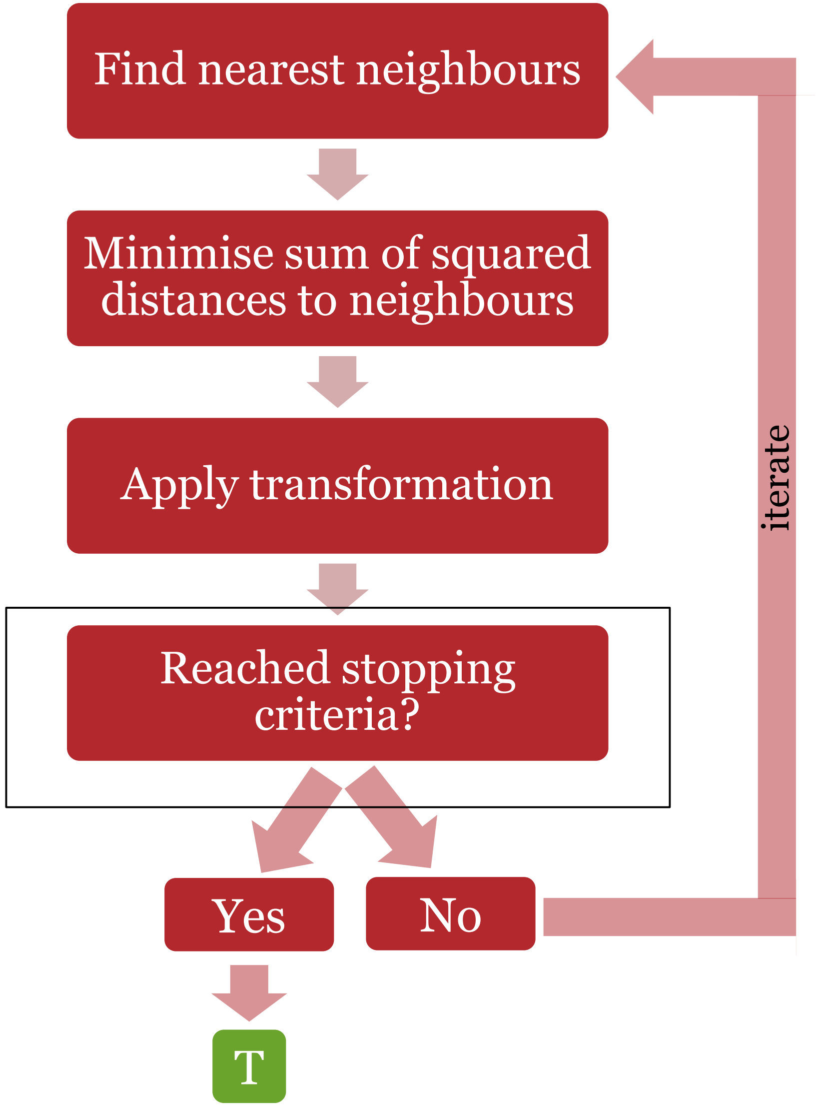
For $\boldsymbol{i} = (\boldsymbol{1}, \boldsymbol{N}_{\boldsymbol{k2}})$

$$\boldsymbol{P}_{\boldsymbol{M}}^{\boldsymbol{k2,i}} = \breve{\boldsymbol{T}}_{\boldsymbol{k1k2}} \boldsymbol{P}_{\boldsymbol{M}}^{\boldsymbol{k2,i}}$$ ← Update all points

end

$$\boldsymbol{T}_{\boldsymbol{k1k2}} = \breve{\boldsymbol{T}}_{\boldsymbol{k1k2}} \boldsymbol{T}_{\boldsymbol{k1k2}}$$ ← Store overall transform for all iterations

ITERATIVE CLOSEST POINT (ICP)



Common stopping criteria:

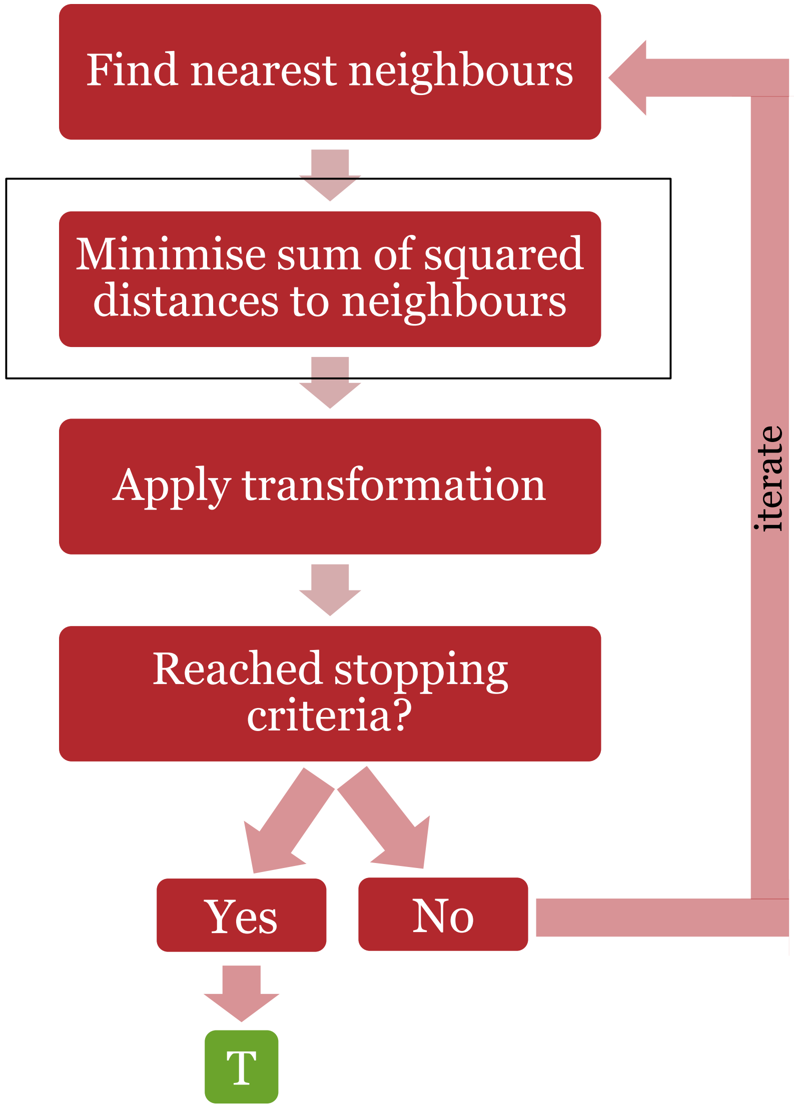
1. Maximum number of iterations
2. Convergence
   - Are the points not changing much with recent iterations?
   - Example stopping criteria:

$$[t_x]^2 + [t_y]^2 + [t_z]^2 < \textit{translation threshold}$$

and/or

$$[\theta_x]^2 + [\theta_y]^2 + [\theta_z]^2 < \textit{rotation threshold}$$

ITERATIVE CLOSEST POINT (ICP)



How do we solve this optimization problem?

$$\breve{T} = \underset{\breve{T} \in SE(3)}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} [y^i - \breve{T} p^i]^2 \longleftarrow$$

Does this look familiar?

This is a least squares problem:

$$\begin{aligned} J(\breve{T}) &= [b - \breve{T}\, m]^T [b - \breve{T}\, m] \\ &= [b^T - m^T \breve{T}^T][b - \breve{T}\, m] \\ &= b^T b - b^T \breve{T}\, m - m^T \breve{T}^T\, b + [\breve{T} m]^T\, \breve{T}\, m \end{aligned}$$

Can we simply take the derivative and set to zero as usual?

$$\boxed{\xcancel{\frac{d}{d\breve{T}} J(\breve{T}) = -b^T m - m^T b + 2\breve{T}\, m = 0}}$$

No: (1) cannot take derivatives of T as we normally do with vectors (2) non-linear, need to iterate

ITERATIVE CLOSEST POINT (ICP)

- Why can't we take a normal derivative of a transformation matrix?
  - Traditional calculus has been defined for vector spaces
  - Why do Transforms not span a Vector space?

1) $u+v$ exists in V	closure under addition
2) $u+v=v+u$	communative
3) $(u+v)+w=u+(v+w)$	associative
4) 0 exists in V, ie $u+0=u$	additive identity
5) $\forall\ u\in V\ \in(-u)$ s.t. $u+(-u)=0$	inverse
6) cu exists in V	closure under scalar multiplication
7) $c(u+v)\ =\ cu+cv$	distributive
8) $(c+d)u=cu+du$	distributive
9) $c(du)=(cd)u$	
10) $1u=u$	multiplicative identity

Let's consider 1)

$$\boldsymbol{T_A} = \begin{bmatrix} 1 & 0 & 0 & t_{x1} \\ 0 & 1 & 0 & t_{y1} \\ 0 & 0 & 1 & t_{z1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T_B} = \begin{bmatrix} 1 & 0 & 0 & t_{x2} \\ 0 & 1 & 0 & t_{y2} \\ 0 & 0 & 1 & t_{z2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \leftarrow \boxed{\text{valid}}$$

$$\boldsymbol{T_A} + \boldsymbol{T_B} = \begin{bmatrix} 2 & 0 & 0 & t_{x1}+t_{x2} \\ 0 & 2 & 0 & t_{y1}+t_{y2} \\ 0 & 0 & 2 & t_{z1}+t_{z2} \\ 0 & 0 & 0 & \boxed{2} \end{bmatrix}$$

invalid

$$\boxed{\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}}$$

$$\boxed{\boldsymbol{RR^T} = \boldsymbol{I}, \boldsymbol{det}(\boldsymbol{R}) = \boldsymbol{1}}$$

Therefore, we need to redefine our derivatives

$$\frac{\boldsymbol{d}\ \boldsymbol{y}(\boldsymbol{x})}{\boldsymbol{dx}} = \lim_{\Delta x \to 0} \frac{\boldsymbol{y}(\boldsymbol{x} \boxed{+} \Delta \boldsymbol{x}) \boxed{-} \boldsymbol{y}(\boldsymbol{x})}{\Delta \boldsymbol{x}}$$

OUTLINE

- Introduction to scan registration
- Iterative closest point (ICP) algorithm
  - Conceptual idea
  - Problem formulation
- Optimization with Lie Algebra
  - Introduction to Lie Algebra
  - Methods for solving optimization problems
  - Solving ICP
- Challenges with ICP
- ICP variations/alternatives
  - Point to Plane
  - G-ICP
  - NDT
- Overview of Task 7

INTRODUCTION TO LIE ALGEBRA

- Rotations (R) and transformations (T) are special mathematical objects called **matrix Lie groups**
- The set of rotation matrices is called the **special orthogonal group**(SO(3)):

$$SO(3) = \{R \in \mathbb{R}^{3x3} \mid RR^T = \boldsymbol{I}, \det(R) = 1\}$$

$$\boxed{R^T = R^{-1}}$$

- The set of transformation matrices is called the **special Euclidean group** (SE(3)):

$$SE(3) = \left\{T = \begin{bmatrix} R & t \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4x4} \middle| R \in SO(3), t \in \mathbb{R}^3\right\}$$

INTRODUCTION TO LIE ALGEBRA

- Properties of matrix Lie groups:
  - Element: C (or R), T
  - Operator: matrix multiplication

property	$SO(3)$	$SE(3)$
closure	$\mathbf{C}_1, \mathbf{C}_2 \in SO(3) \Rightarrow \mathbf{C}_1\mathbf{C}_2 \in SO(3)$	$\mathbf{T}_1, \mathbf{T}_2 \in SE(3) \Rightarrow \mathbf{T}_1\mathbf{T}_2 \in SE(3)$
associativity	$\mathbf{C}_1(\mathbf{C}_2\mathbf{C}_3) = (\mathbf{C}_1\mathbf{C}_2)\mathbf{C}_3 = \mathbf{C}_1\mathbf{C}_2\mathbf{C}_3$	$\mathbf{T}_1(\mathbf{T}_2\mathbf{T}_3) = (\mathbf{T}_1\mathbf{T}_2)\mathbf{T}_3 = \mathbf{T}_1\mathbf{T}_2\mathbf{T}_3$
identity	$\mathbf{C}, \mathbf{1} \in SO(3) \Rightarrow \mathbf{C}\mathbf{1} = \mathbf{1}\mathbf{C} = \mathbf{C}$	$\mathbf{T}, \mathbf{1} \in SE(3) \Rightarrow \mathbf{T}\mathbf{1} = \mathbf{1}\mathbf{T} = \mathbf{T}$
invertibility	$\mathbf{C} \in SO(3) \Rightarrow \mathbf{C}^{-1} \in SO(3)$	$\mathbf{T} \in SE(3) \Rightarrow \mathbf{T}^{-1} \in SE(3)$

INTRODUCTION TO LIE ALGEBRA

- **Lie algebra**: to every Lie group, there is associated Lie algebra
- The Lie algebra associated with rotations – SO(3) – is given by:

$$so(3) = \left\{ \Phi = \phi^{\wedge} \in \mathbb{R}^{3x3} \middle| \phi \in \mathbb{R}^{3} \right\}$$

where

Skew symmetric operator. This was covered in the multiview geometry lecture where it was used to perform the cross product

$$\phi^{\wedge} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}$$

so(3) is a vector space!

INTRODUCTION TO LIE ALGEBRA

- The Lie algebra associated with transformations – SE(3) – is given by:

$$se(3) = \{\Xi = \xi^{\wedge} \in \mathbb{R}^{4x4} | \xi \in \mathbb{R}^6\}$$

where

$$\xi^{\wedge} = \begin{bmatrix} \rho \\ \phi \end{bmatrix}^{\wedge} = \begin{bmatrix} \phi^{\wedge} & \rho \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathbb{R}^{4x4}$$

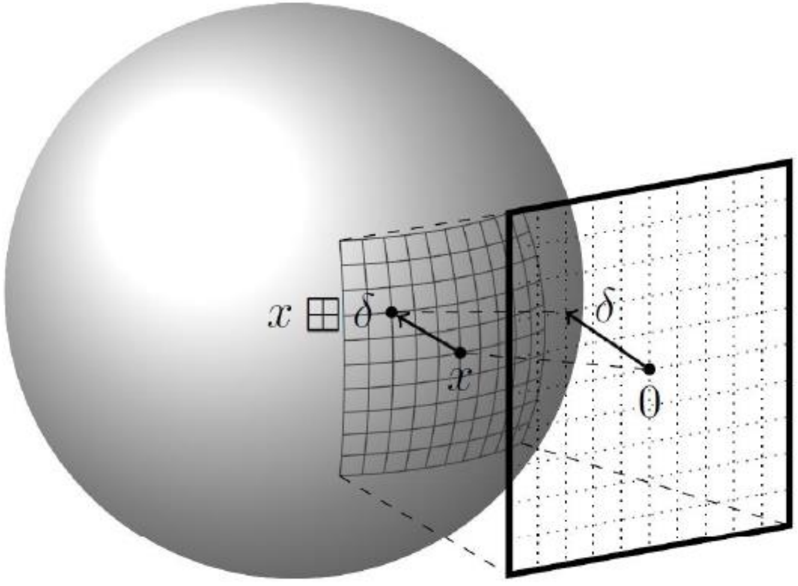
se(3) is a vector space!

Note: this is an overloading of the skew-symmetric operator : $(\cdot)^{\wedge}$

INTRODUCTION TO LIE ALGEBRA

- Converting between Lie group and Lie algebra:

	Lie Group		Lie Algebra
rotations:	$R \in SO(3)$ $\begin{bmatrix} R_1 & R_2 & R_3 \\ R_4 & R_5 & R_6 \\ R_7 & R_8 & R_9 \end{bmatrix}$	$\underset{\text{exp}}{\overset{\text{log}}{\rightleftarrows}}$	$\phi^\wedge \in so(3)$ $\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_1 \end{bmatrix}^\wedge$
transformations:	$T \in SE(3)$ $\begin{bmatrix} R_1 & R_2 & R_3 & \phi_1 \\ R_4 & R_5 & R_6 & \phi_2 \\ R_7 & R_8 & R_9 & \phi_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\underset{\text{exp}}{\overset{\text{log}}{\rightleftarrows}}$	$\xi^\wedge \in se(3)$ $\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge$



where

log: is the matrix logarithmic (logm in matlab)

exp: is the matrix exponential (expm in matlab)

Proofs for these relationships as well as the implementations of the logarithmic and exponential maps will not be covered. For more details, see State Estimation for Robotics by Timothy D. Barfoot (2018)

SOLVING OPTIMIZATION PROBLEMS WITH LIE GROUPS

- Reminder what we are trying to solve:

$$\breve{\boldsymbol{T}} = \underset{\breve{T} \epsilon\, SE(3)}{\arg\min} \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} [\boldsymbol{y^i} - \breve{\boldsymbol{T}}\boldsymbol{p^i}]^{\boldsymbol{2}}$$

$$\boldsymbol{J}(\breve{\boldsymbol{T}}) = \boldsymbol{b^T b} - \boldsymbol{b^T}\breve{\boldsymbol{T}}\,\boldsymbol{m} - \boldsymbol{m^T}\breve{\boldsymbol{T}}^{\boldsymbol{T}}\,\boldsymbol{b} + [\breve{\boldsymbol{T}}\boldsymbol{m}]^{\boldsymbol{T}}\,\breve{\boldsymbol{T}}\,\boldsymbol{m}$$

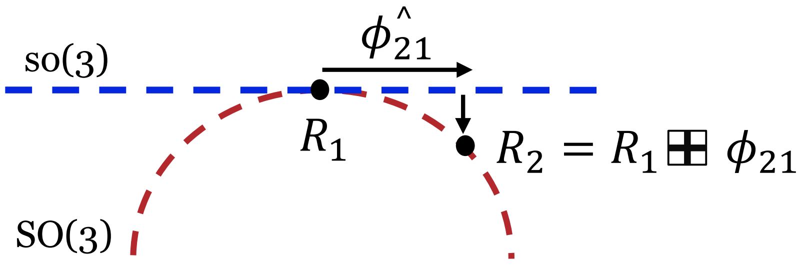
- There are two general approaches for solving optimization problems that contain Lie groups:

1. Redefine the + and – operations and use these to perform the derivatives

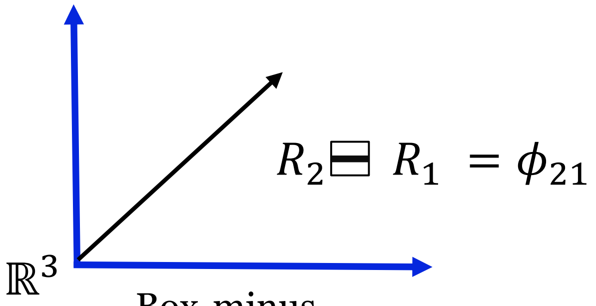
2. Formulate your problem in the Lie algebra space and then solve normally

SOLVING ICP WITH LIE GROUPS

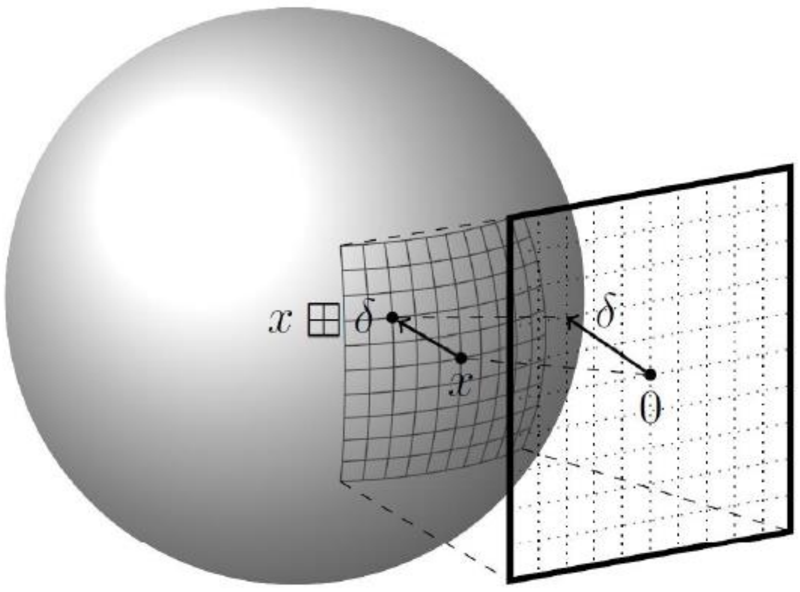
- Method 1: redefining + and – operators
  - These are referred to as "box-plus" $\boxplus$ and "box-minus" $\boxminus$



Box-plus



Box-minus



- Our derivatives (Jacobians) can now be expressed in terms of these operators to solve our optimization problem

$$f_3 : \mathbb{SO}(3) \mapsto \mathbb{R}$$

$$\frac{\partial f_3}{\partial \Phi} = \lim_{\epsilon \to 0} \begin{bmatrix} \frac{f_3(\Phi \boxplus (\mathbf{e}_1 \epsilon)) - f_3(\Phi)}{\epsilon} \\ \frac{f_3(\Phi \boxplus (\mathbf{e}_2 \epsilon)) - f_3(\Phi)}{\epsilon} \\ \frac{f_3(\Phi \boxplus (\mathbf{e}_3 \epsilon)) - f_3(\Phi)}{\epsilon} \end{bmatrix}$$

$$\boldsymbol{J(\breve{T}) = b^T b - b^T \breve{T}\, m - m^T \breve{T}^T\, b + [\breve{T} m]^T\, \breve{T}\, m}$$

- We will not go over this in detail for the sake of time. We have a simpler solution for our specific problem!

SOLVING ICP WITH LIE GROUPS

- Method 2: reformulating problem to work in Lie algebra space
  - Problem:

$$\boldsymbol{T} = \underset{\boldsymbol{T}\epsilon\, SE(3)}{\arg\min}\, \boldsymbol{J}(\boldsymbol{T}) \quad , \quad \boldsymbol{J}(\boldsymbol{T}) = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}} \left[\boldsymbol{y}^{\boldsymbol{i}} - \boldsymbol{T}\boldsymbol{p}^{\boldsymbol{i}}\right]^{\boldsymbol{T}} \left[\boldsymbol{y}^{\boldsymbol{i}} - \boldsymbol{T}\boldsymbol{p}^{\boldsymbol{i}}\right] \qquad -(\mathbf{1})$$

  - Traditional approaches will solve this with iterations
  - There is a way to solve this specific problem without iterating, however the math is out of the scope of this course
  - Solution:

SE(3)

se(3)

Define a **perturbation scheme**: $\boldsymbol{T} = \boldsymbol{exp}(\boldsymbol{\epsilon}^{\wedge})\boldsymbol{T}_{\boldsymbol{op}} \approx (\boldsymbol{I} + \boldsymbol{\epsilon}^{\wedge})\boldsymbol{T}_{\boldsymbol{op}}$

Sub. Into (1) and simplify:

$$\boldsymbol{J}(\boldsymbol{T}) = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}} \left[\boldsymbol{y}^{\boldsymbol{i}} - (\boldsymbol{I} + \boldsymbol{\epsilon}^{\wedge})\boldsymbol{T}_{\boldsymbol{op}}\boldsymbol{p}^{\boldsymbol{i}}\right]^{\boldsymbol{T}} \left[\boldsymbol{y}^{\boldsymbol{i}} - (\boldsymbol{I} + \boldsymbol{\epsilon}^{\wedge})\boldsymbol{T}_{\boldsymbol{op}}\boldsymbol{p}^{\boldsymbol{i}}\right]$$

$$\boldsymbol{J}(\boldsymbol{T}) = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}} \left[(\boldsymbol{y}^{\boldsymbol{i}} - \boldsymbol{z}_{\boldsymbol{i}}) - \boldsymbol{z}_{\boldsymbol{i}}^{\odot}\boldsymbol{\epsilon}\right]^{\boldsymbol{T}} \left[(\boldsymbol{y}^{\boldsymbol{i}} - \boldsymbol{z}_{\boldsymbol{i}}) - \boldsymbol{z}_{\boldsymbol{i}}^{\odot}\boldsymbol{\epsilon}\right]$$

Least squares problem with all vectors!

Where: $\boldsymbol{z}_{\boldsymbol{i}} = \boldsymbol{T}_{\boldsymbol{op}}\boldsymbol{p}^{\boldsymbol{i}}$, $\boldsymbol{\epsilon}^{\wedge}\boldsymbol{z}_{\boldsymbol{i}} = \boldsymbol{z}_{\boldsymbol{i}}^{\odot}\boldsymbol{\epsilon}$, $\boldsymbol{z}_{\boldsymbol{i}}^{\odot} = \begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\eta}\end{bmatrix}^{\odot} = \begin{bmatrix}\boldsymbol{\eta I} & -\boldsymbol{\rho}^{\wedge}\\ \mathbf{0}^{\boldsymbol{T}} & \mathbf{0}^{\boldsymbol{T}}\end{bmatrix}$

4x6

SOLVING ICP WITH LIE GROUPS

- Method 2: reformulating problem to work in Lie algebra space
  - Solution continued:

$$\boldsymbol{J}(\boldsymbol{T}) = \sum_{i=1}^{N} \left[(\boldsymbol{y}^{i} - \boldsymbol{z}_i) - \boldsymbol{z}_i^{\odot}\boldsymbol{\epsilon}\right]^{T} \left[(\boldsymbol{y}^{i} - \boldsymbol{z}_i) - \boldsymbol{z}_i^{\odot}\boldsymbol{\epsilon}\right]$$

Take derivative w.r.t Lie algebra (vector) and set to zero:

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\epsilon}^{T}} = -\sum_{i=1}^{N} \boldsymbol{z}_i^{\odot T} \left[(\boldsymbol{y}^{i} - \boldsymbol{z}_i) - \boldsymbol{z}_i^{\odot} \boldsymbol{\epsilon}^{*}\right] = \boldsymbol{0}$$

$$\left[\sum_{i=1}^{N} \boldsymbol{z}_i^{\odot T} \boldsymbol{z}_i^{\odot}\right] \boldsymbol{\epsilon}^{*} = -\sum_{i=1}^{N} \boldsymbol{z}_i^{\odot T} \left[(\boldsymbol{y}^{i} - \boldsymbol{z}_i) - \boldsymbol{z}_i^{\odot}\boldsymbol{\epsilon}\right]$$

Update the operating point and iterate:

$$\boldsymbol{T}_{op} = \boldsymbol{exp}\left(\boldsymbol{\epsilon}^{*\wedge}\right)\boldsymbol{T}_{op}$$

SOLVING ICP WITH LIE GROUPS

- Summary
  - Problem: to solve ICP we want to minimise J

$$J(\boldsymbol{T}) = \sum_{i=1}^{N} [\boldsymbol{y}^i - \boldsymbol{T}\boldsymbol{p}^i]^T [\boldsymbol{y}^i - \boldsymbol{T}\boldsymbol{p}^i]$$

  - This cannot be treated as a normal optimization problem since we are optimizing for a transformation matrix which is not a vector (cannot take the derivative w.r.t T normally)
  - It can be solved in two ways:

(1) redefine derivatives with Lie algebra using ⊞ and ⊟

(2) Rewrite the problem and solve in the Lie algebra space:

$$J(\boldsymbol{T}) = \sum_{i=1}^{N} [(\boldsymbol{y}^i - \boldsymbol{z}_i) - \boldsymbol{z}_i^{\odot}\boldsymbol{\epsilon}]^T [(\boldsymbol{y}^i - \boldsymbol{z}_i) - \boldsymbol{z}_i^{\odot}\boldsymbol{\epsilon}]$$

Substitute our perturbation scheme & simplify

$$\frac{\partial J}{\partial \boldsymbol{\epsilon}^T} = -\sum_{i=1}^{N} \boldsymbol{z}_i^{\odot T} [(\boldsymbol{y}^i - \boldsymbol{z}_i) - \boldsymbol{z}_i^{\odot}\boldsymbol{\epsilon}] = \boldsymbol{0}$$

Take derivative w.r.t perturbations, set to zero to find perturbations. Then iterate

OUTLINE

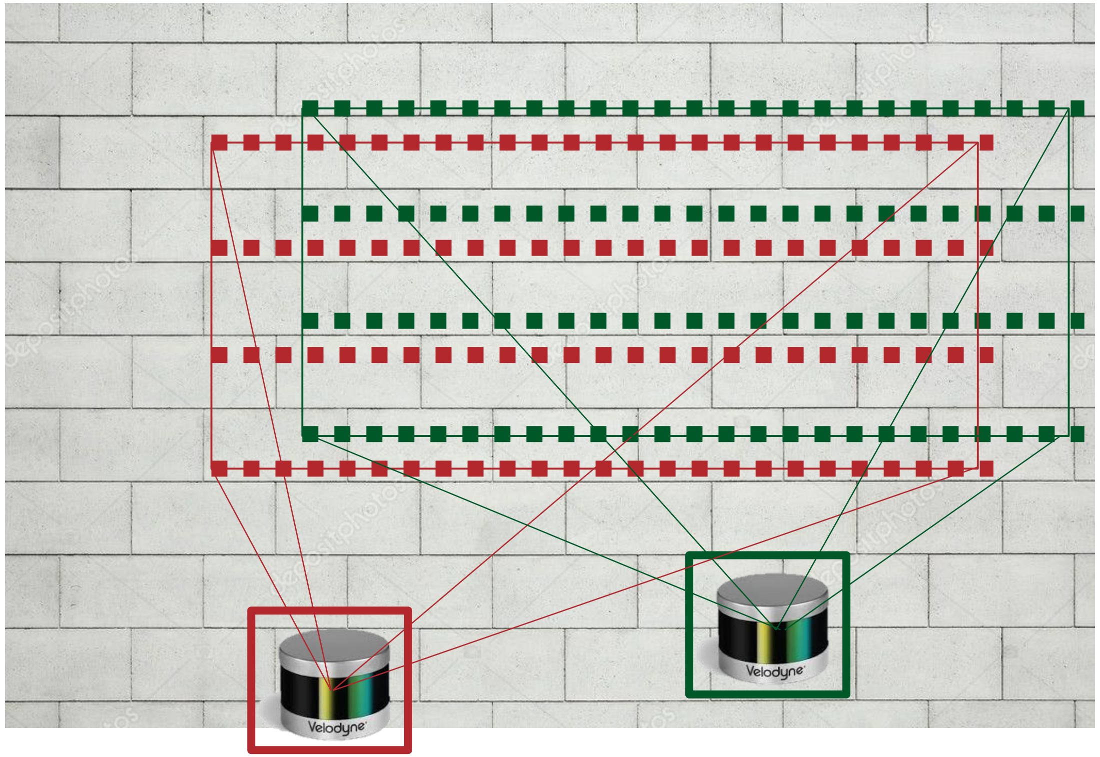
- Introduction to scan registration
- Iterative closest point (ICP) algorithm
  - Conceptual idea
  - Problem formulation
- Optimization with Lie Algebra
  - Introduction to Lie Algebra
  - Methods for solving optimization problems
  - Solving ICP
- Challenges with ICP
- ICP variations/alternatives
  - Point to Plane
  - G-ICP
  - NDT
- Overview of Task 7

CHALLENGES WITH ICP

- Main Challenges:
  1. Point to point association not always ideal
  2. Non-overlapping points
  3. Local minima
  4. Computationally expensive

CHALLENGES WITH ICP

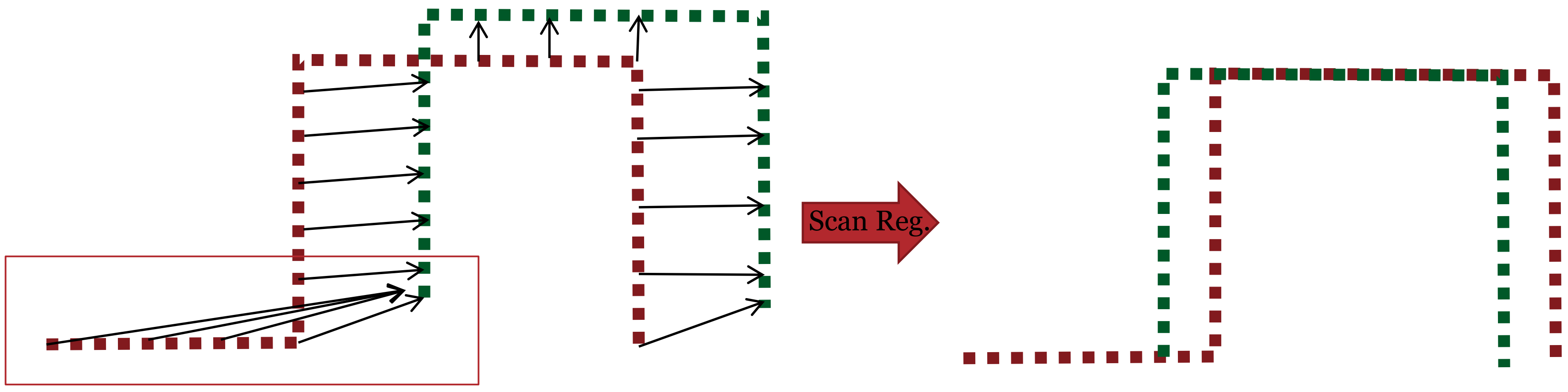
1. Point to point association not always ideal



*This problem is more severe with sparse point clouds, which is the case with lidars

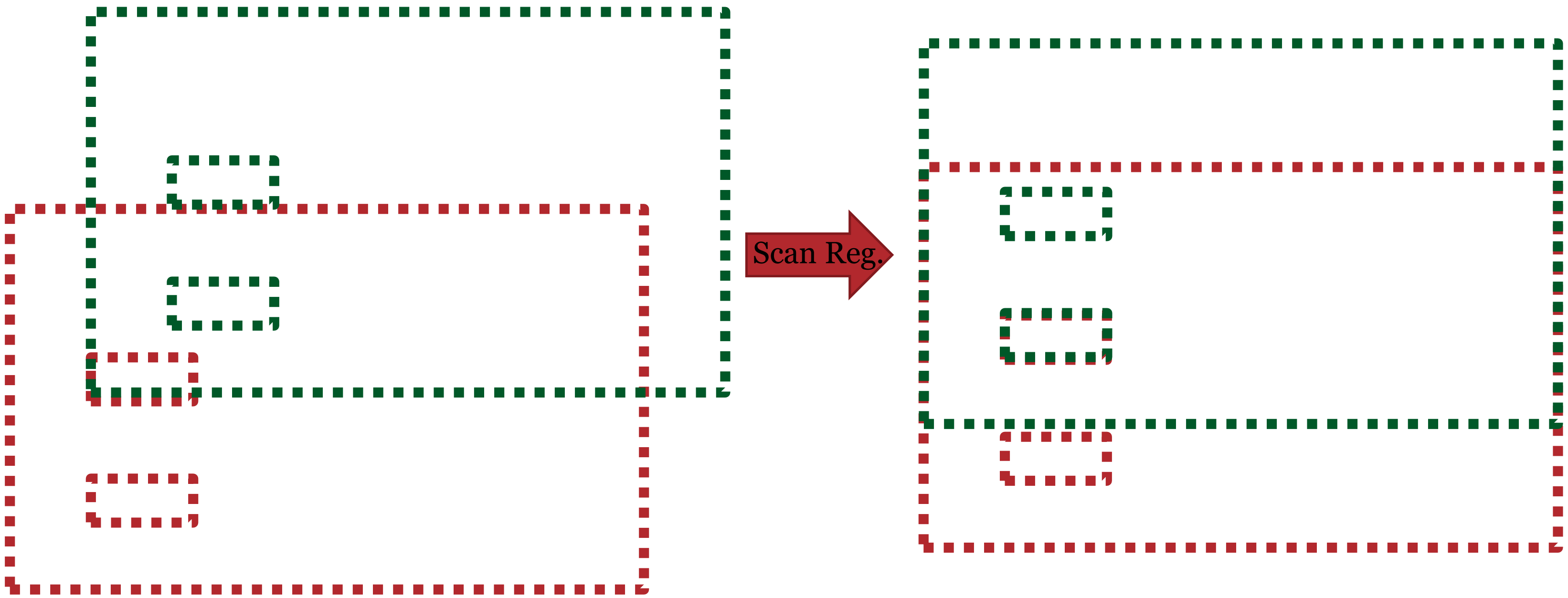
CHALLENGES WITH ICP

2. Non-overlapping points

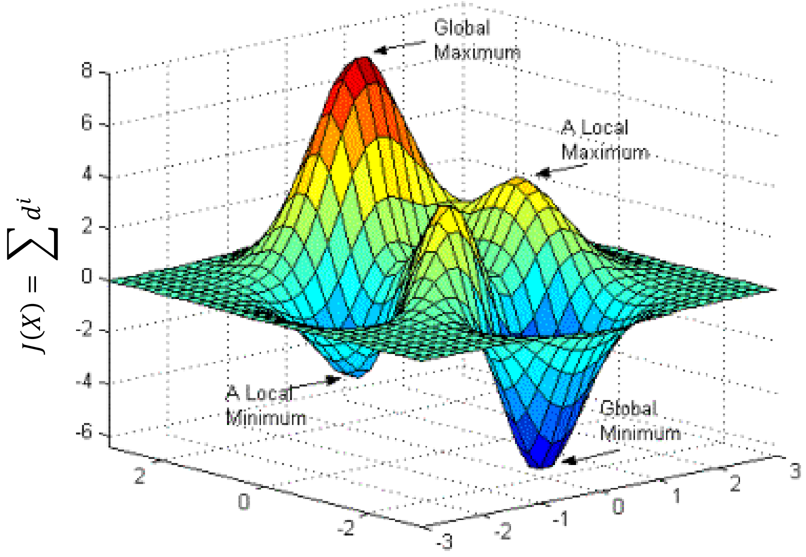


CHALLENGES WITH ICP

3. Local minima



Non convex optimization

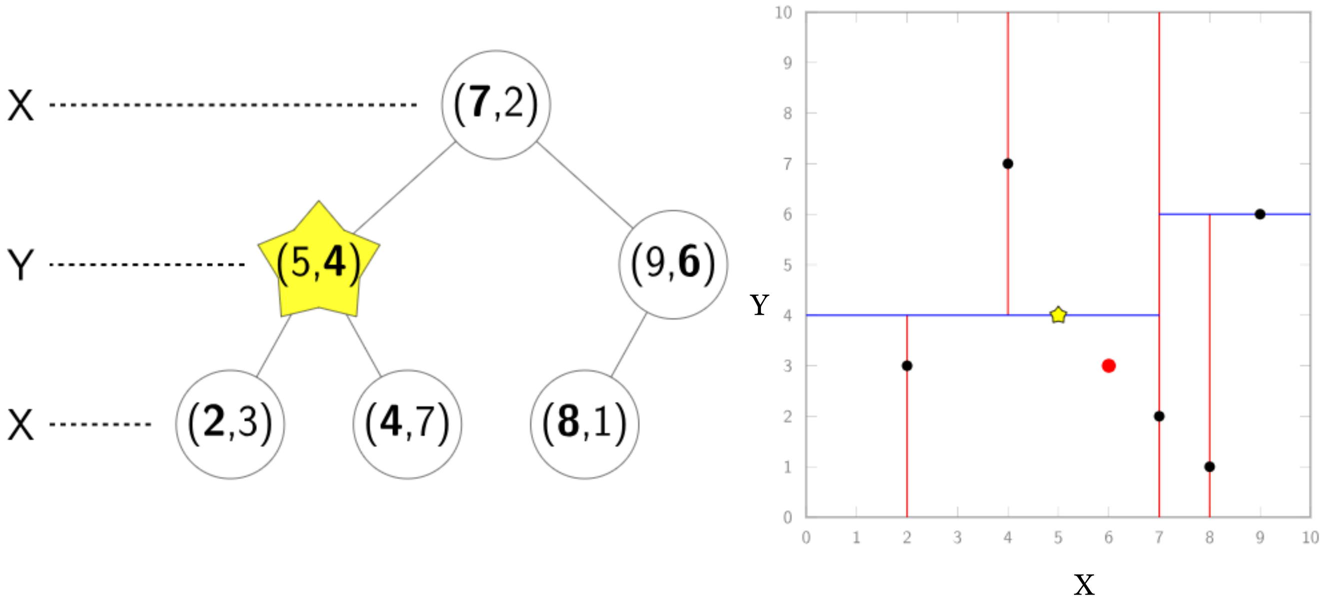


CHALLENGES WITH ICP

4. Computationally expensive

- Each iteration, 3D volume searches are needed to be performed for each point in the cloud
    - As #iterations and #points increases, so does the computation time
    - Can have 100s of iterations, over millions of points
- Volume search is normally accelerated using a search tree (K-D tree) which can be built once and used for neighbour searches for each point, each iteration

2D Search Tree Example



OUTLINE

- Introduction to scan registration
- Iterative closest point (ICP) algorithm
  - Conceptual idea
  - Problem formulation
- Optimization with Lie Algebra
  - Introduction to Lie Algebra
  - Methods for solving optimization problems
  - Solving ICP
- Challenges with ICP
- ICP variations/alternatives
  - Point to Plane
  - G-ICP
  - NDT
- Overview of Task 7

ICP VARIATIONS/ALTERNATIVES

There are hundreds of variations to regular ICP

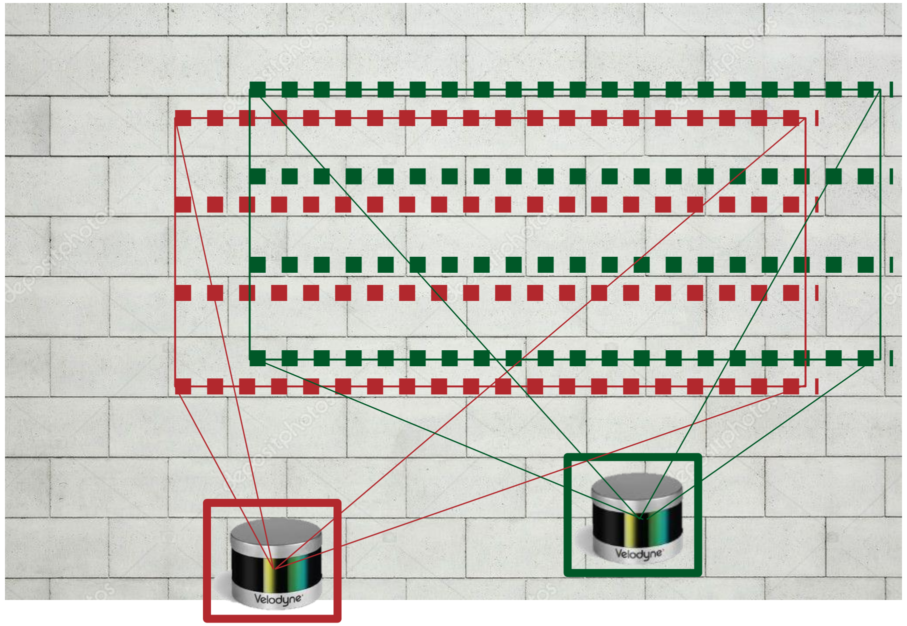
We will cover the following three common methods:

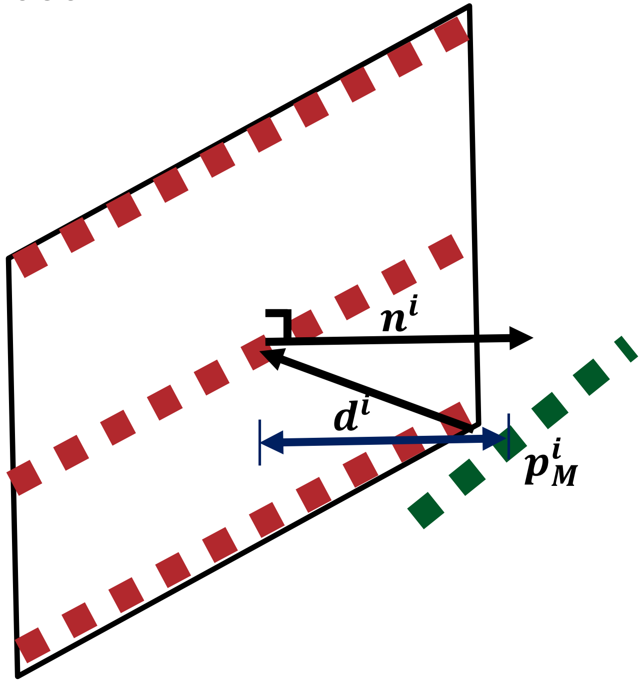
1. Point to Plane ICP
2. Generalized ICP (G-ICP)
3. Normal Distributions Transforms (NDT)

ICP VARIATIONS/ALTERNATIVES

1. Point to Plane ICP

   - Goal: solve the problem of point correspondences discussed before



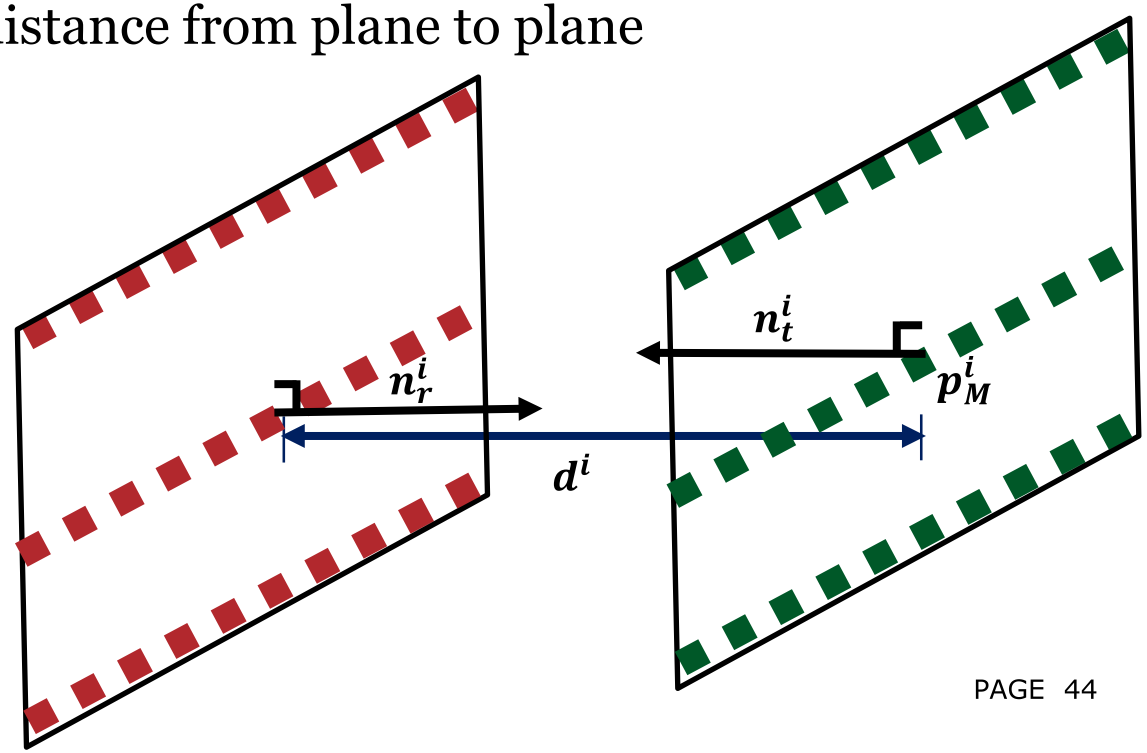


1. Find closest point
2. Fit plane to k nearest neighbours
3. Minimise distance to plane

ICP VARIATIONS/ALTERNATIVES

1. Generalized-ICP (G-ICP)

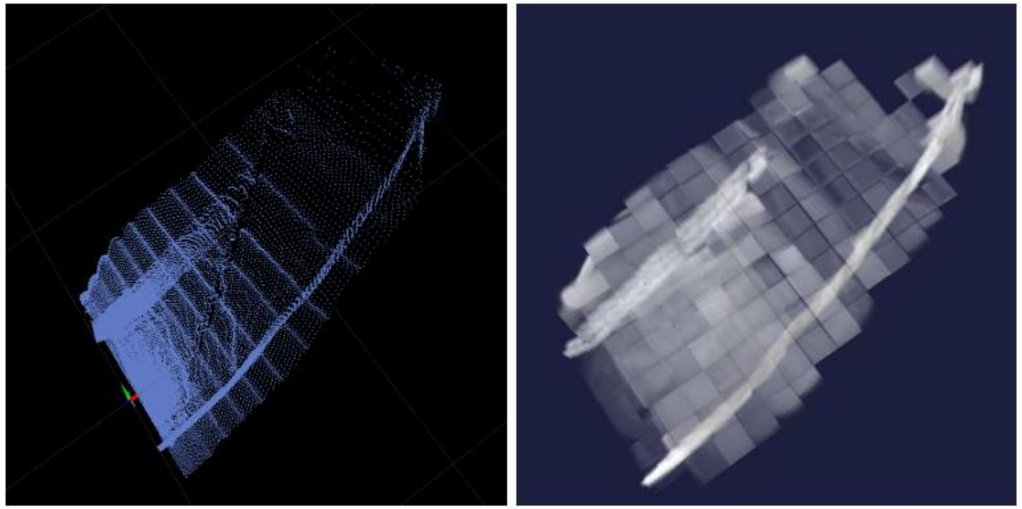
    - Goal: solve the problem of point correspondences discussed before
    - Same procedure as point to plane ICP, but minimised distance from plane to plane



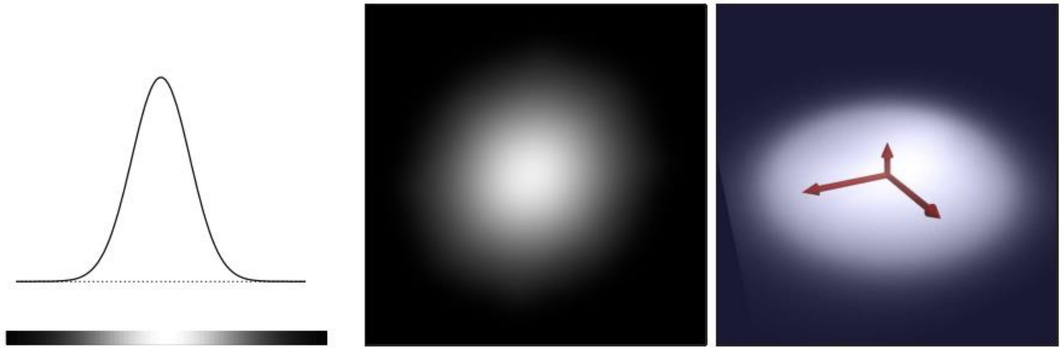
ICP VARIATIONS/ALTERNATIVES

1. Normal Distributions Transform

   - Divide space containing reference scan into a grid of cubes (or squares for the 2D case)
   - Gaussian PDF is computed for each cell using the distributions of points within that cell. This gives a surface representation
   - Find the pose of the current scan that maximises the likelihood that the points of the current scan lie on the reference scan surface



(a) Original point cloud. (b) NDT representation.



(a) 1D (b) 2D (c) 3D

TASK 7 – SCAN REGISTRATION