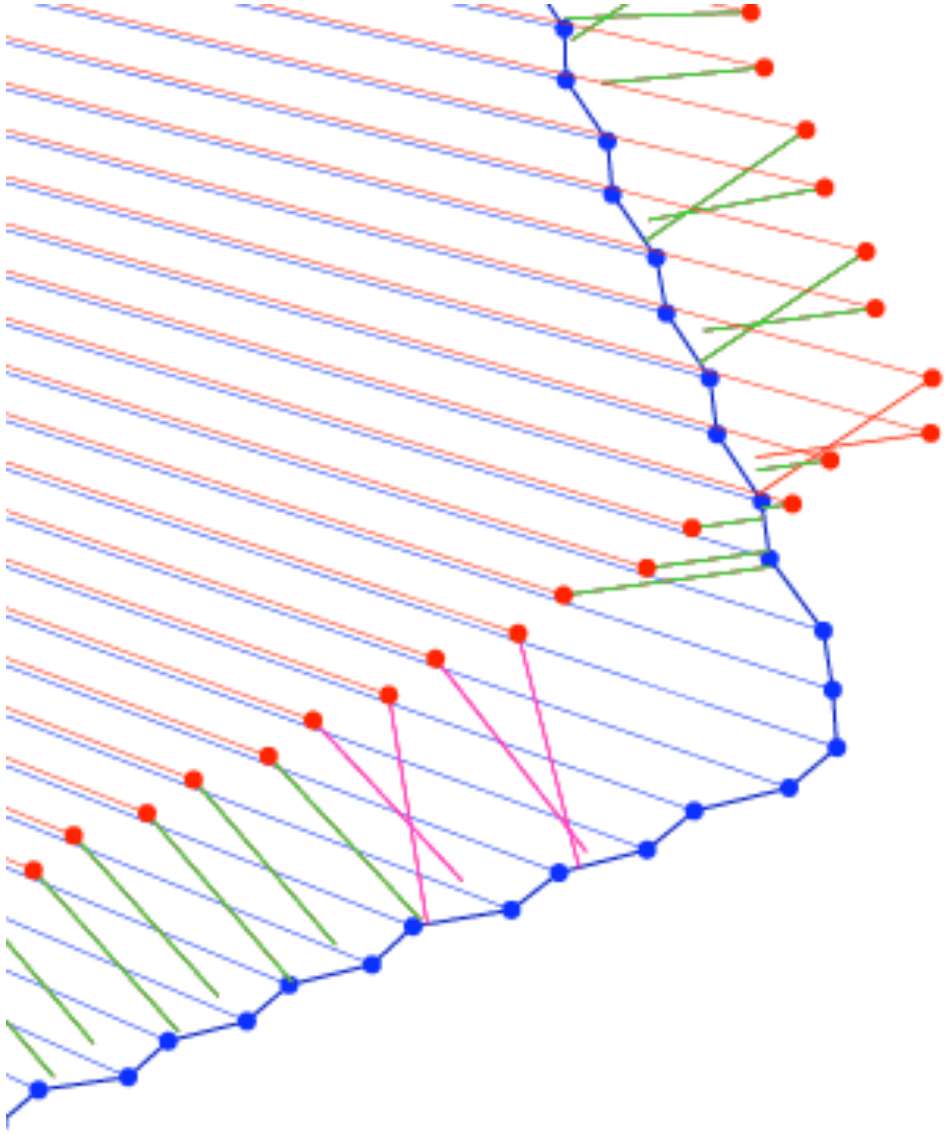


F1/10
Autonomous Racing

Scan Matching
- Part 2

Madhur Behl

CS 4501
Rice Hall 120

F1/10 2020 Autonomous Racing Teams CS 4501

File Edit View Insert Format Data Tools Add-ons Help Last edit was made 37 minutes ago by anonymous

Vehicle Number

	A	B	C	D	E	F	G
1	**Vehicle Number**	**Vehicle N/W SSID**	**Member 1 (First + Last Name)**	**Member 2 (First + Last Name)**	**Member 3 (First + Last Name)**	**Member 4 (First + Last Name)**	**Color Code (Assigned by TA)**
2							
3	1	team_1	Raghava Pamula	Catherine Bradberry	Logan Hylton		
4							
5	2	team_2	Ryan Kann	Wyatt Joyner	Corey Lando	Jack Herd	
6							
7	3	team_3	Rahul Batra	Rashid Lasker	Stephen Shiao	Adeet Patel	
8							
9	4	team_4	Bradley Knaysi	Peng Zhang	Emory Ducote		
10							
11	5	team_5	Jaspreet Ranjit	Jack Schumann	Hsing Chun Lin	Micah Harris	
12							
13	6	team_6	Sean Shih	Chris Han			
14							
15	7	team_7	Kaiying Shan	Hanzhi Zhou			
16							
17	8	team_8	Michael Tang	Catherine Im	Austin Bunting		
18							
19	9	team_9					

Scan Matching

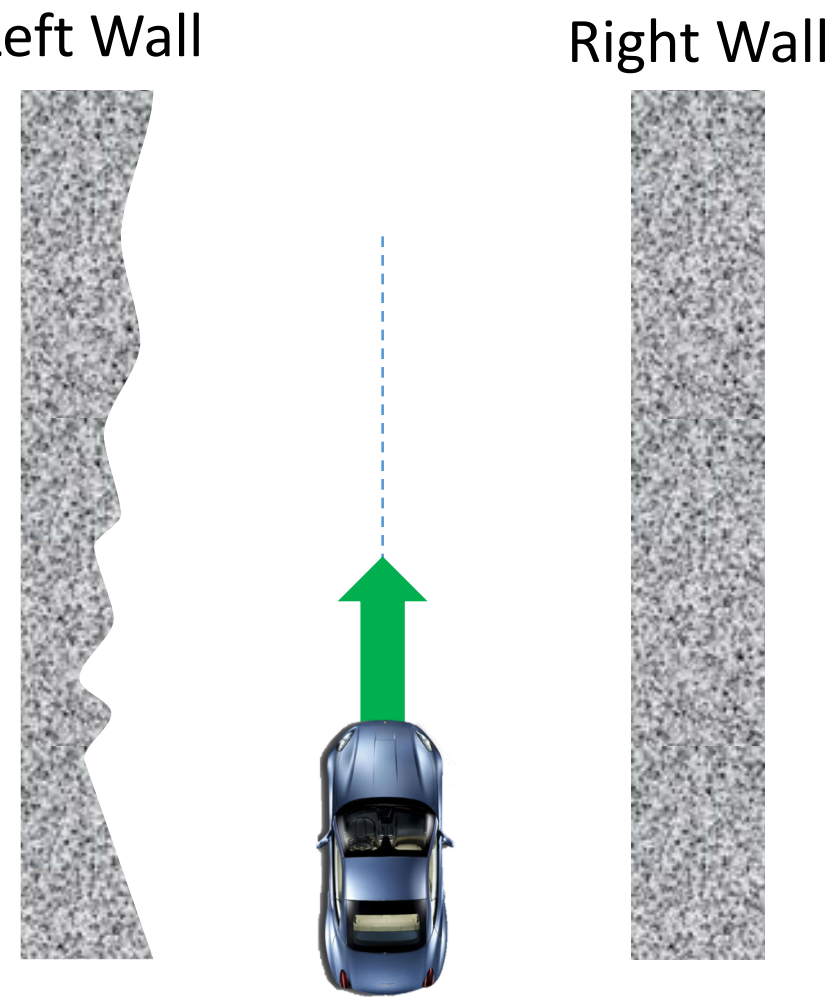
Raw Odometry

Odometry with LIDAR

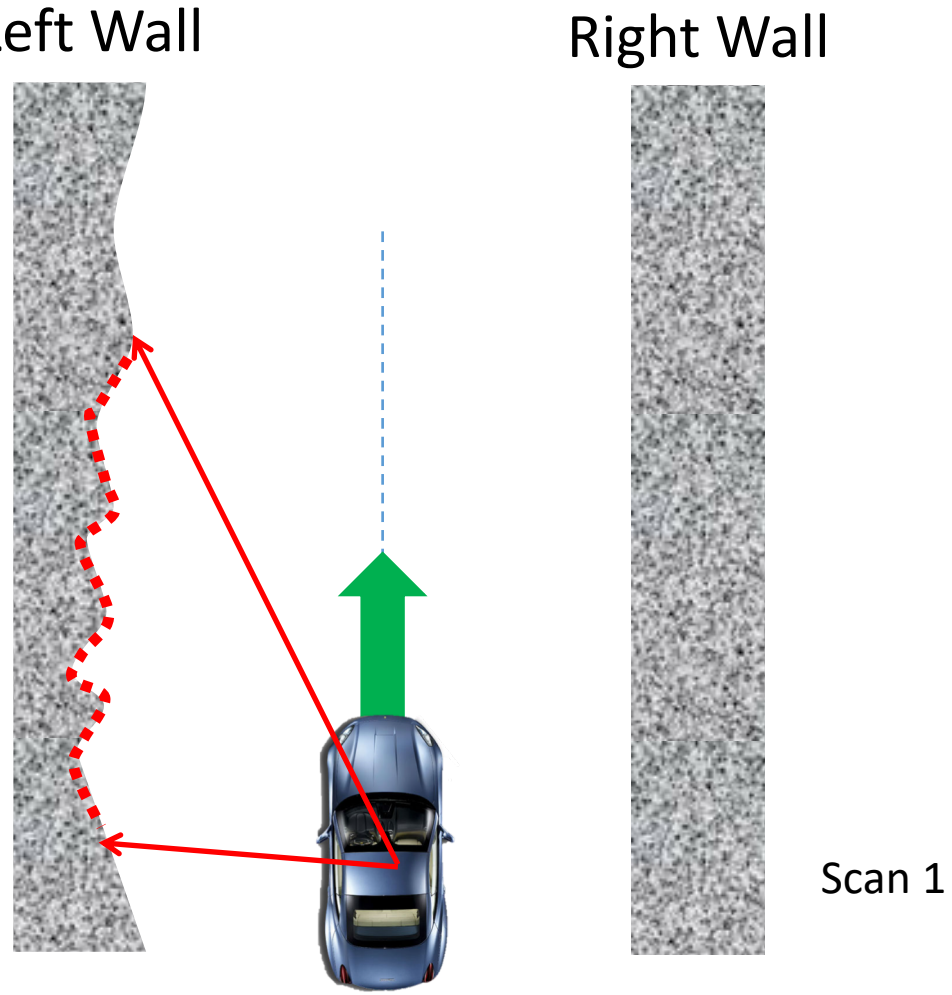
- Odometry = Estimation of position (x, y, z) and attitude (θ, φ, ψ)

- In what follows, will use “position” as short for “position and attitude”

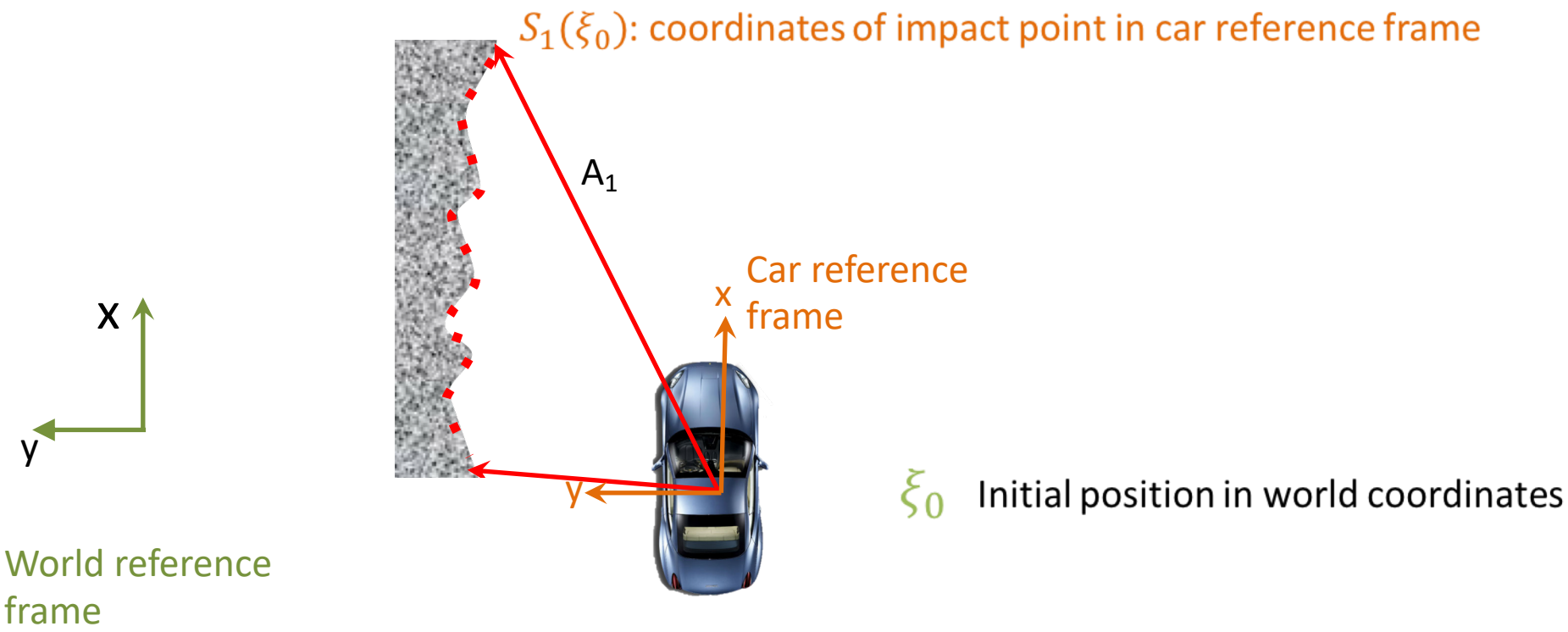
LIDAR for odometry: Scan matching



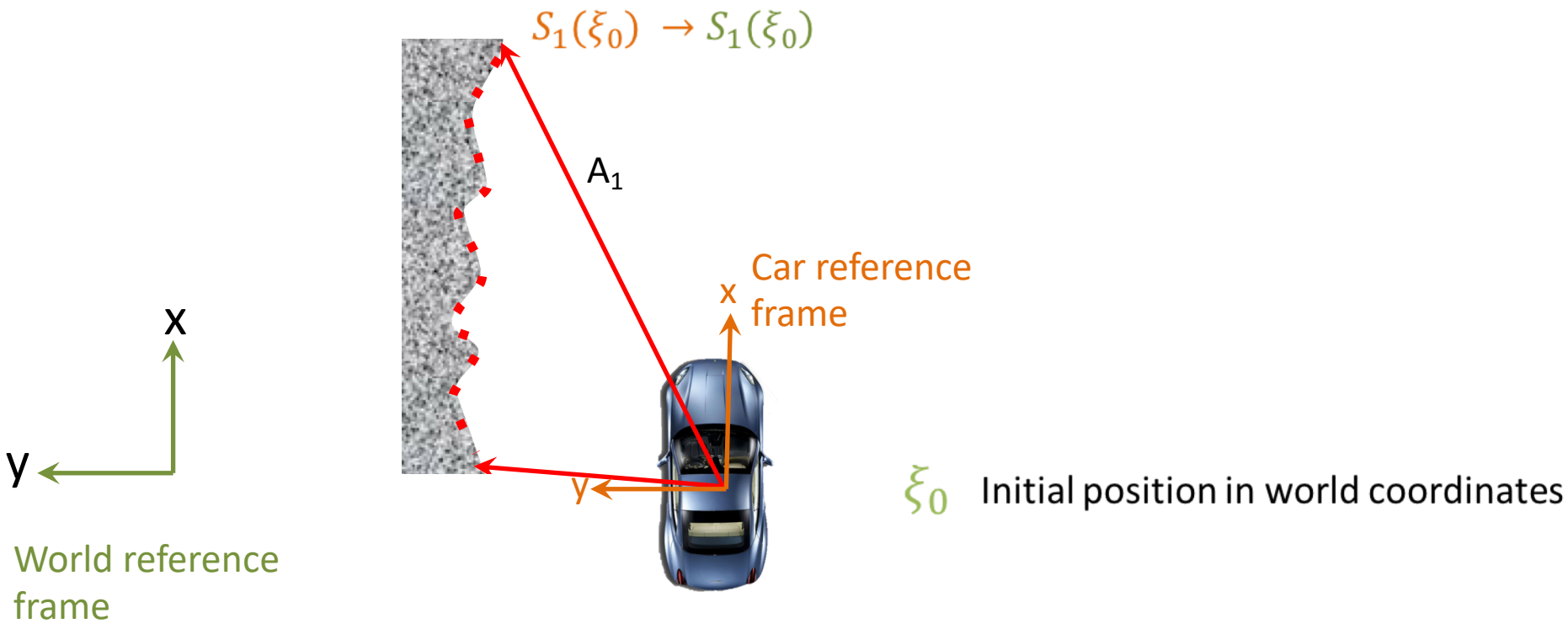
LIDAR for odometry: Scan matching

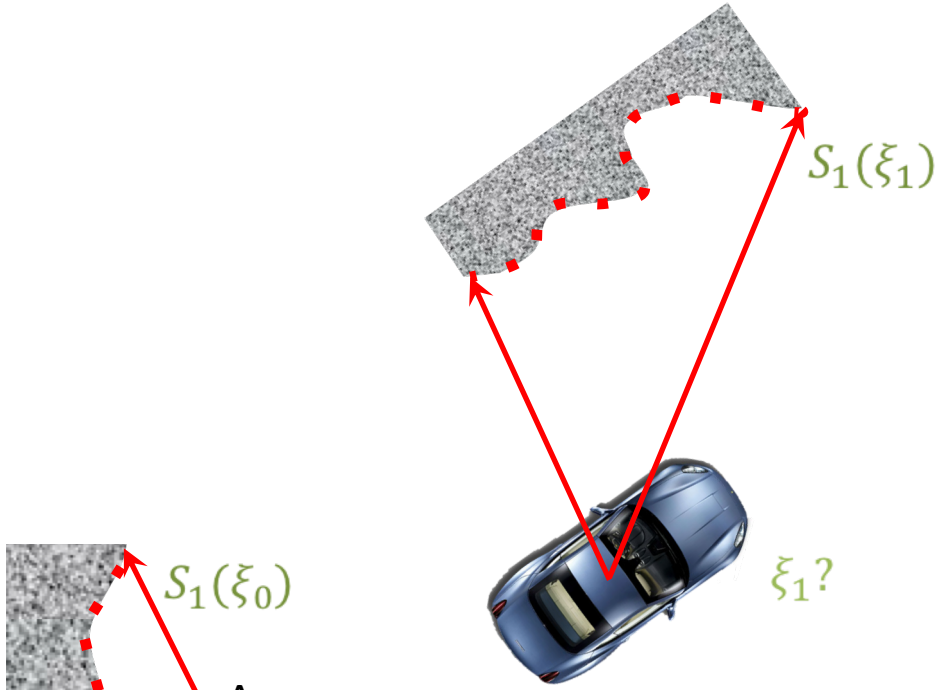


LIDAR for odometry: Scan 1



LIDAR for odometry: Scan 1



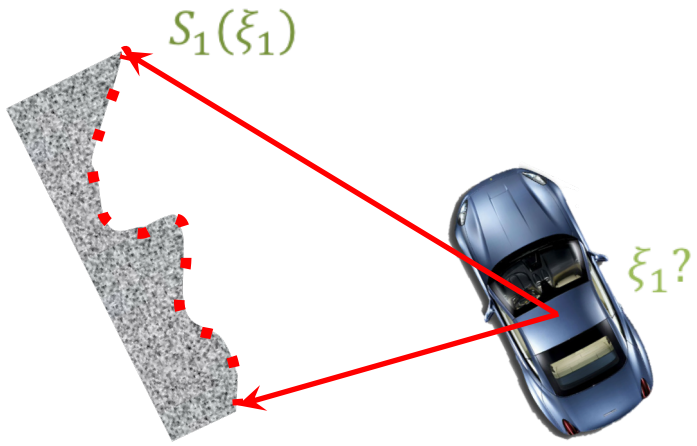


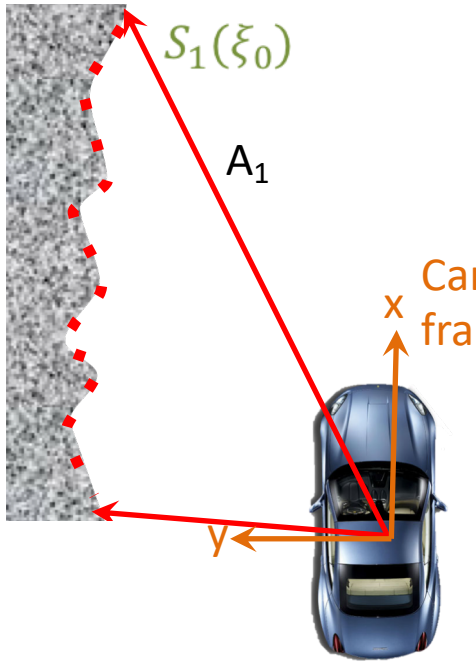
x

y

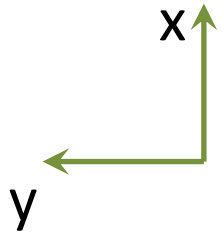
World reference frame

ξ_0 Initial position in world coordinates



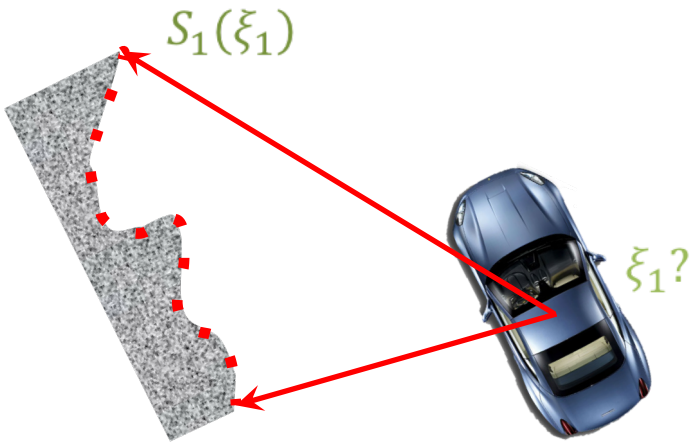


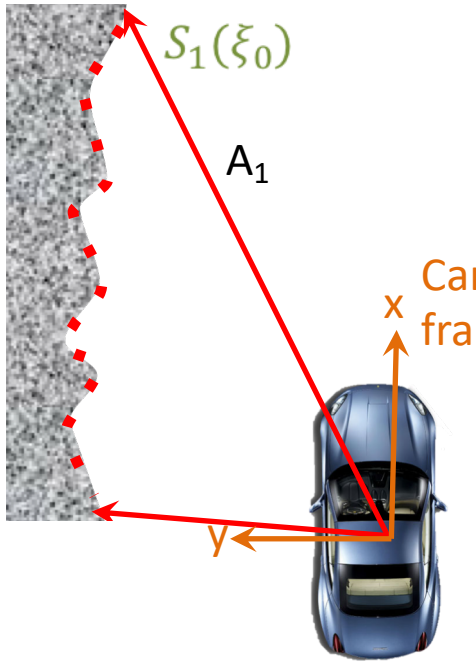
Car reference frame

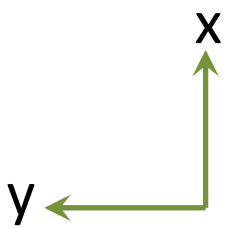


World reference frame

ξ_0 Initial position in world coordinates



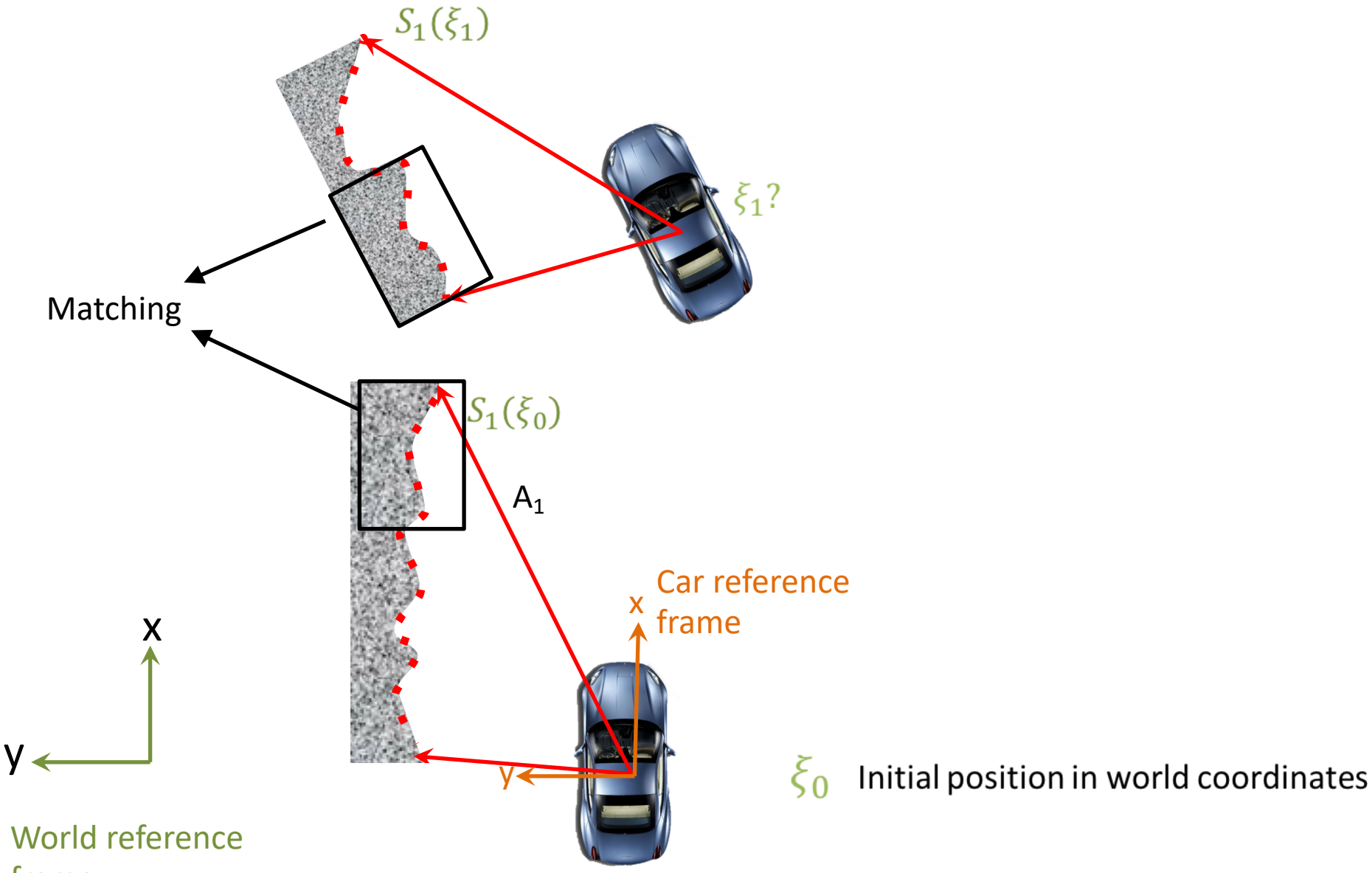


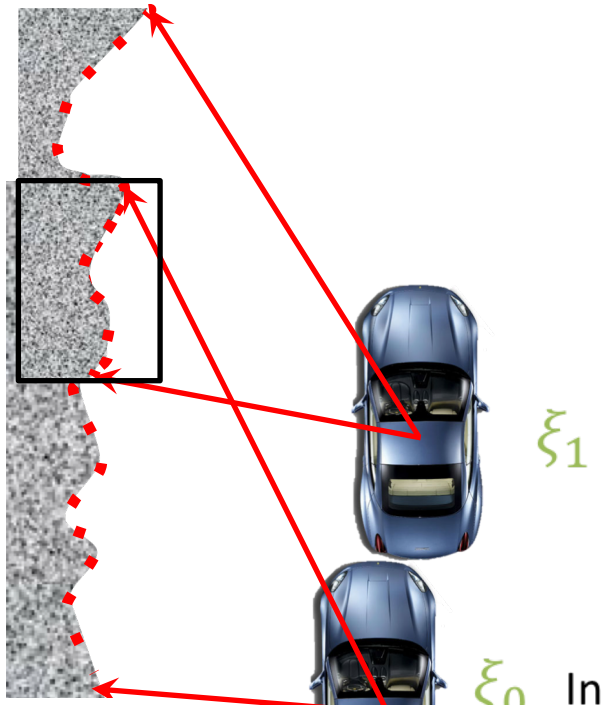


World reference frame

Premise:
most likely car position at Scan 2 is the position that gives best overlap between the two scenes

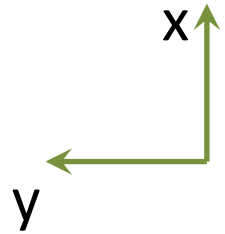
ξ_0 Initial position in world coordinates


$S_1(\xi_1)$
$\xi_1?$
Matching
$S_1(\xi_0)$
A_1
Car reference frame
x
y
x
y
World reference frame
ξ_0 Initial position in world coordinates



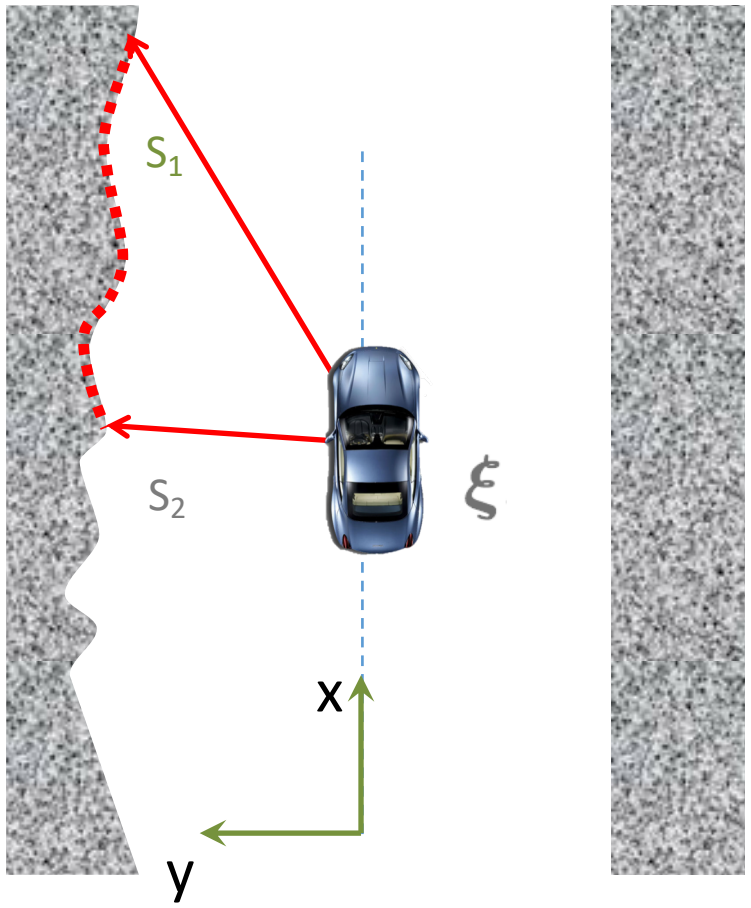
ξ_1

ξ_0 Initial position in world coordinates



World reference frame

Scan matching: optimization

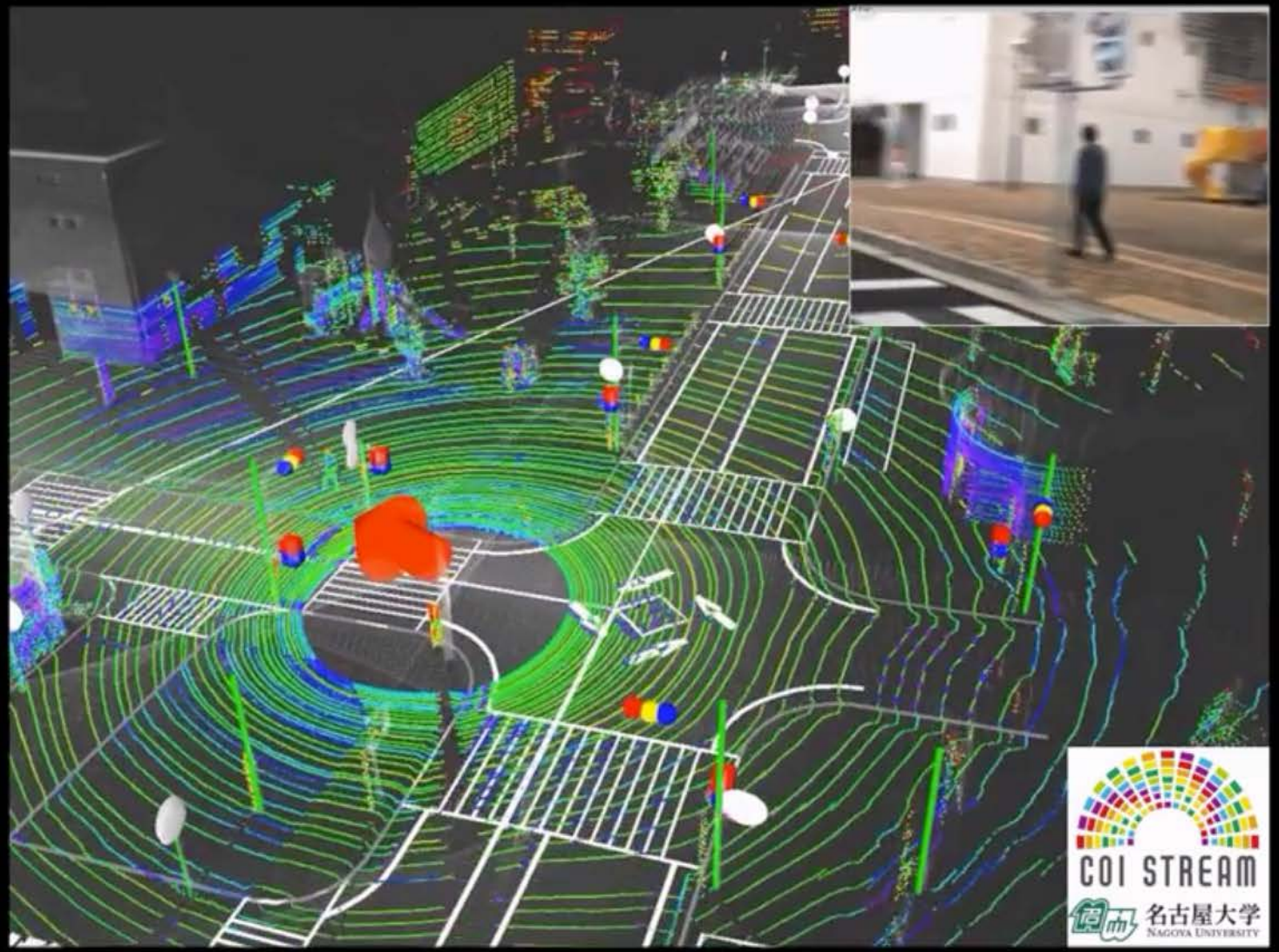


Impact coordinates of i^{th} step in world frame

Total of n steps (n = 1084)

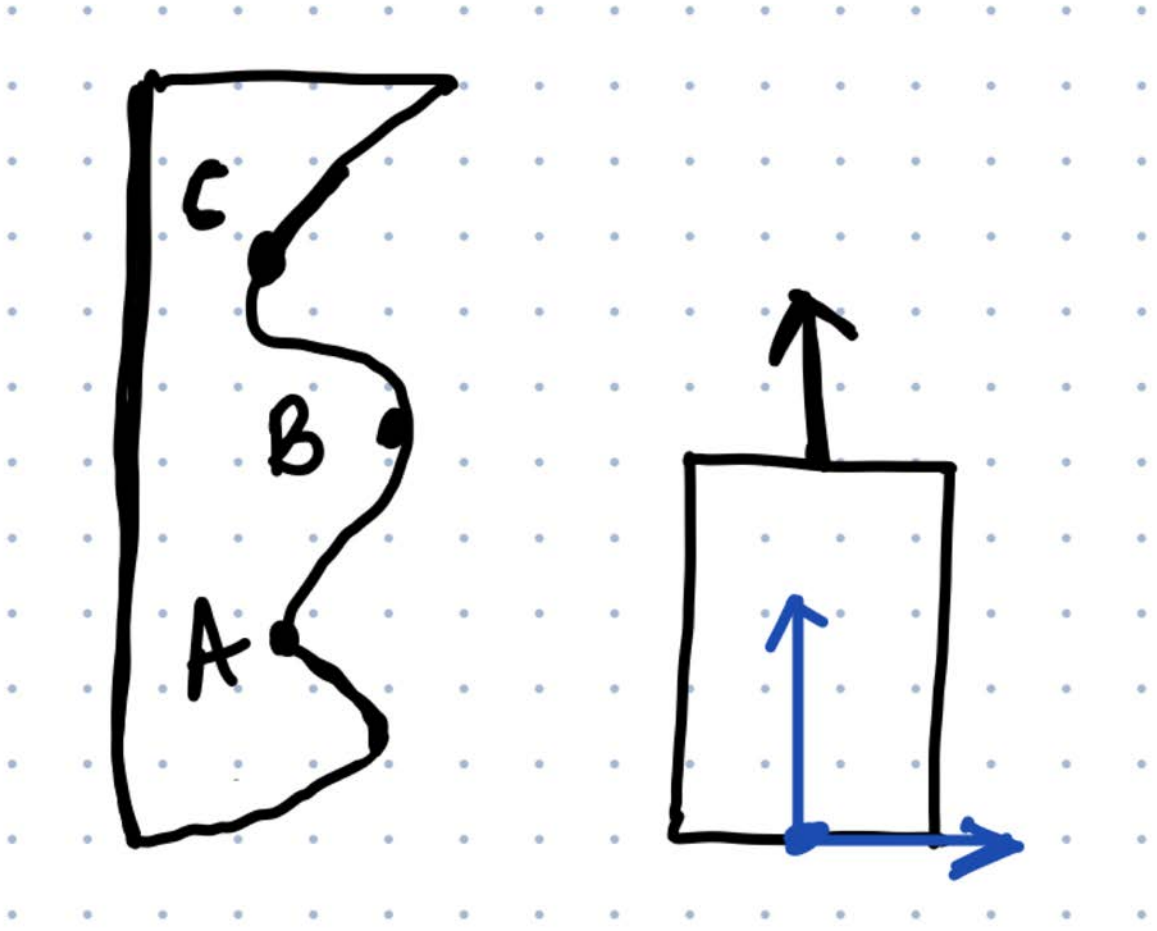
$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - M(\mathbf{S}_i(\boldsymbol{\xi}))\right]^2$$

Measure of match


COI STREAM
名古屋大学
NAGOYA UNIVERSITY

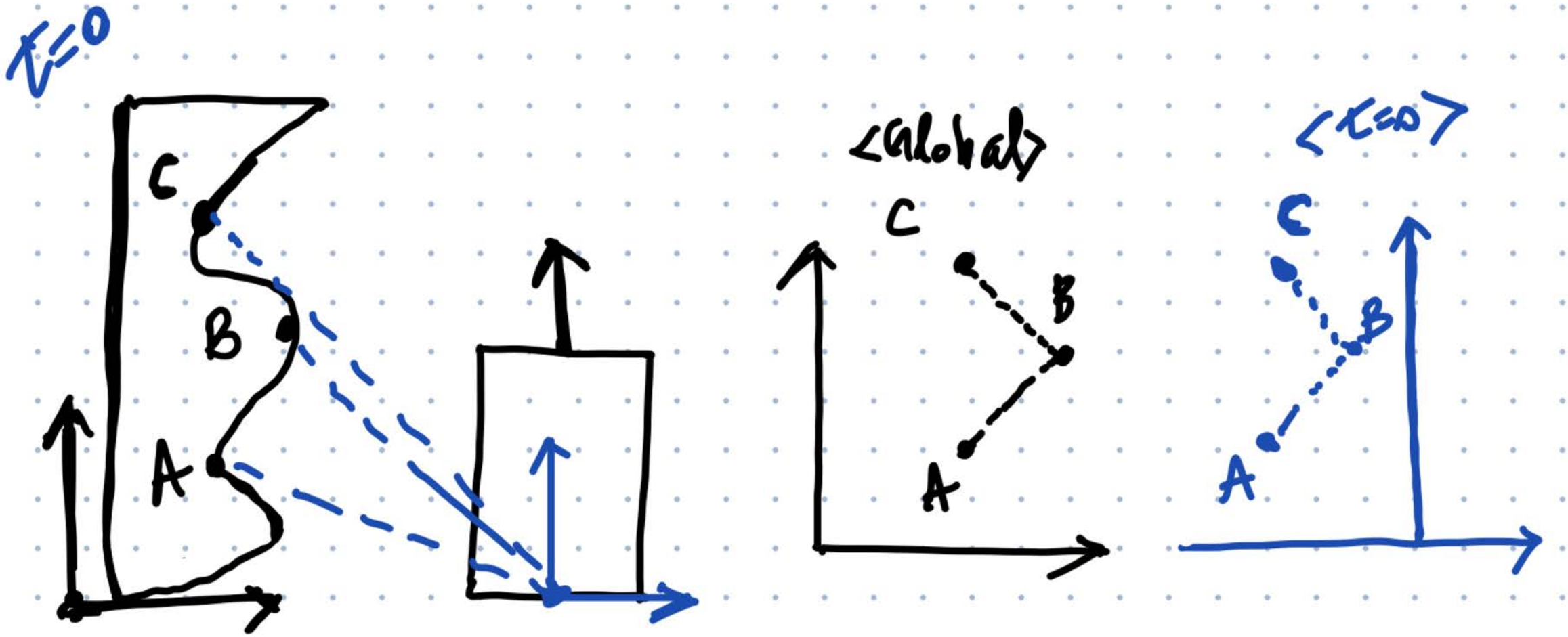
Problem Setup

Let’s say my robot is in some environment with landmarks A, B, C



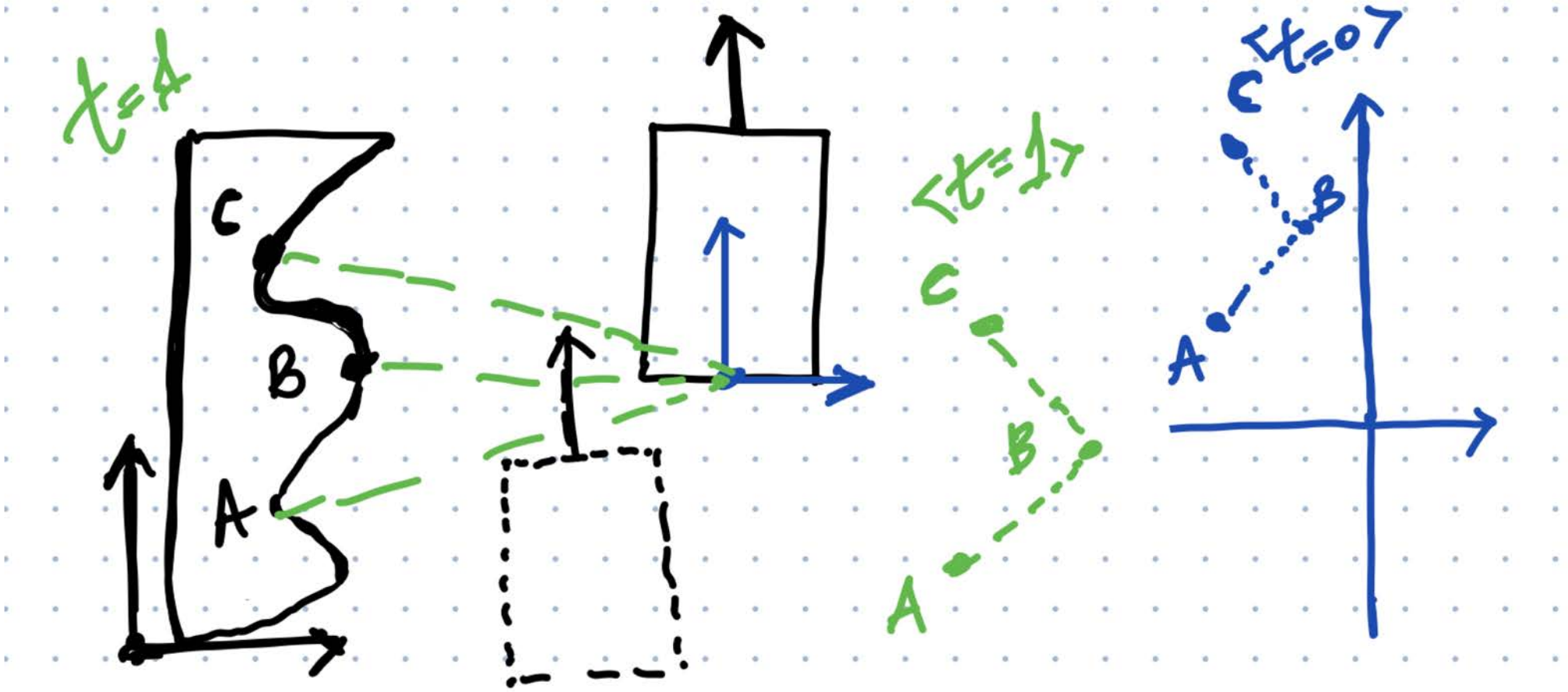
Problem Setup

At t=0, measurements of distances to A,B,C are taken:



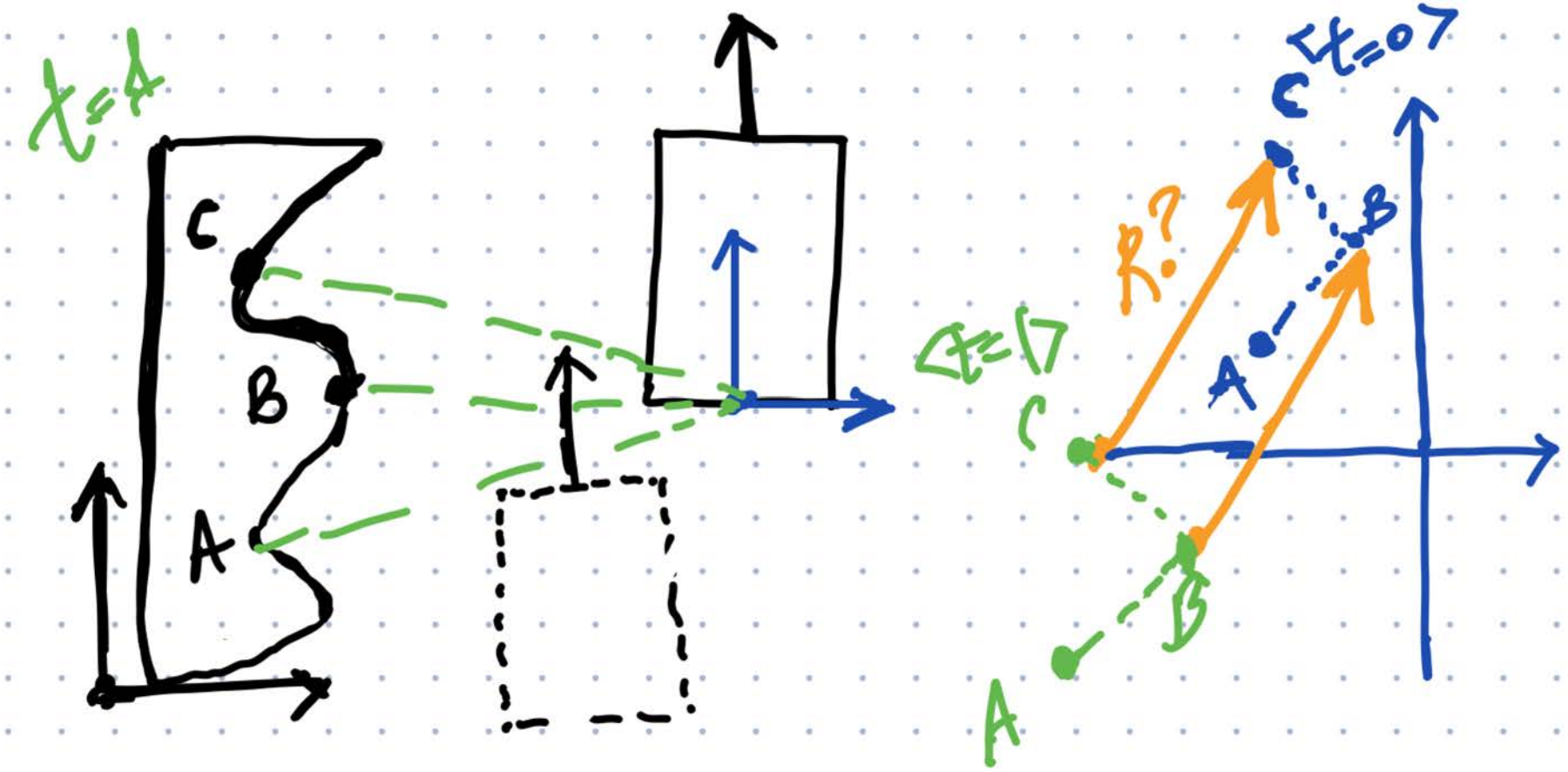
Problem Setup

At t=1, moving in some unknown direction, the distances to the landmarks will have changed:



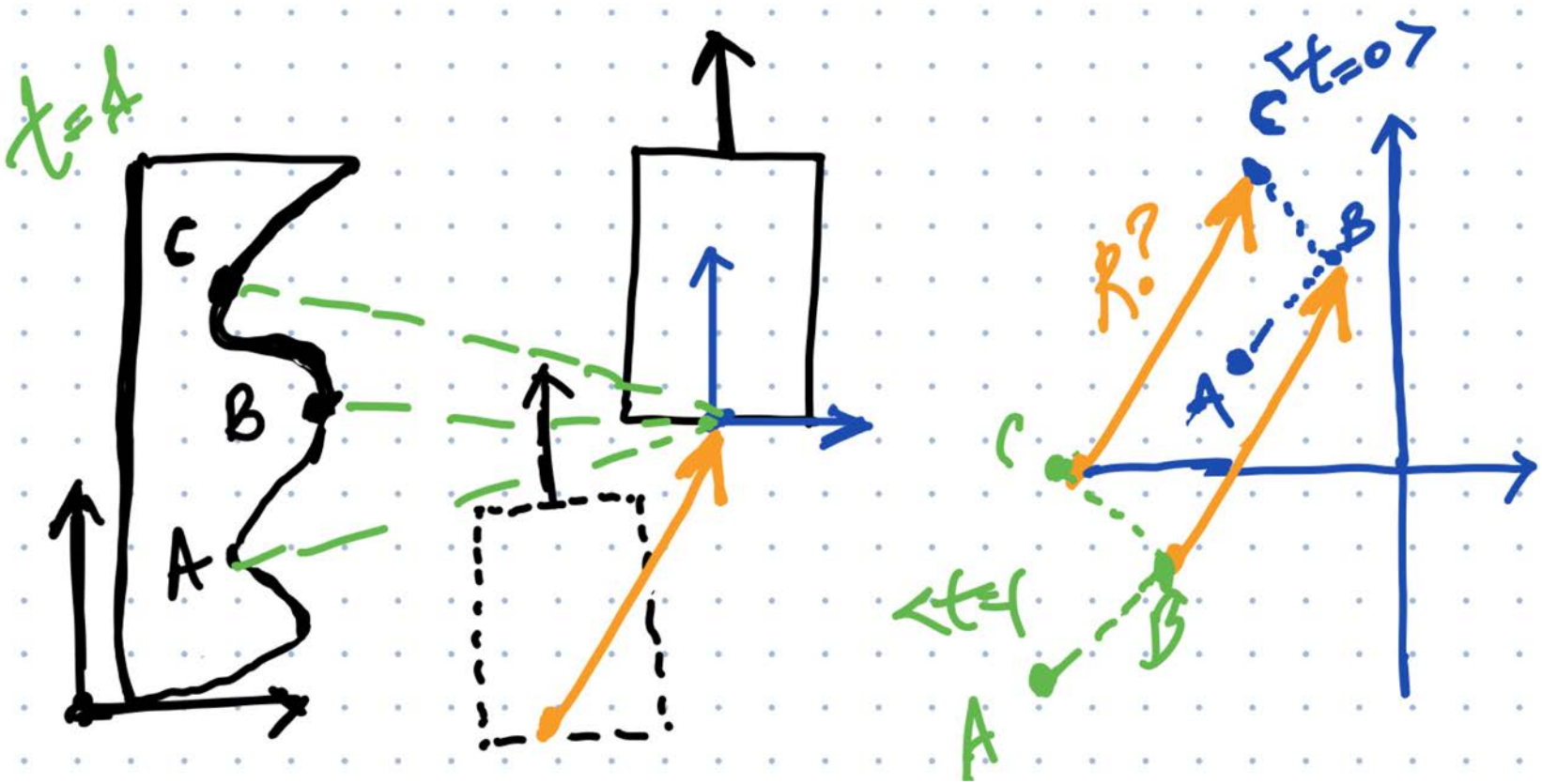
Problem Setup

We want to find transform R, the transforms the two set of points to be the closest:



Problem Setup

Why? The transform R represents how much the robot has moved in time:



How do we find R?

Challenge:

- If we knew A, B, C (landmarks) exactly, then this would be easy
- We don’t know which measurement is for which landmark

Solution:

- Assume closest points correspond to each other → “correspondence match”
- Iteratively find the best transform R

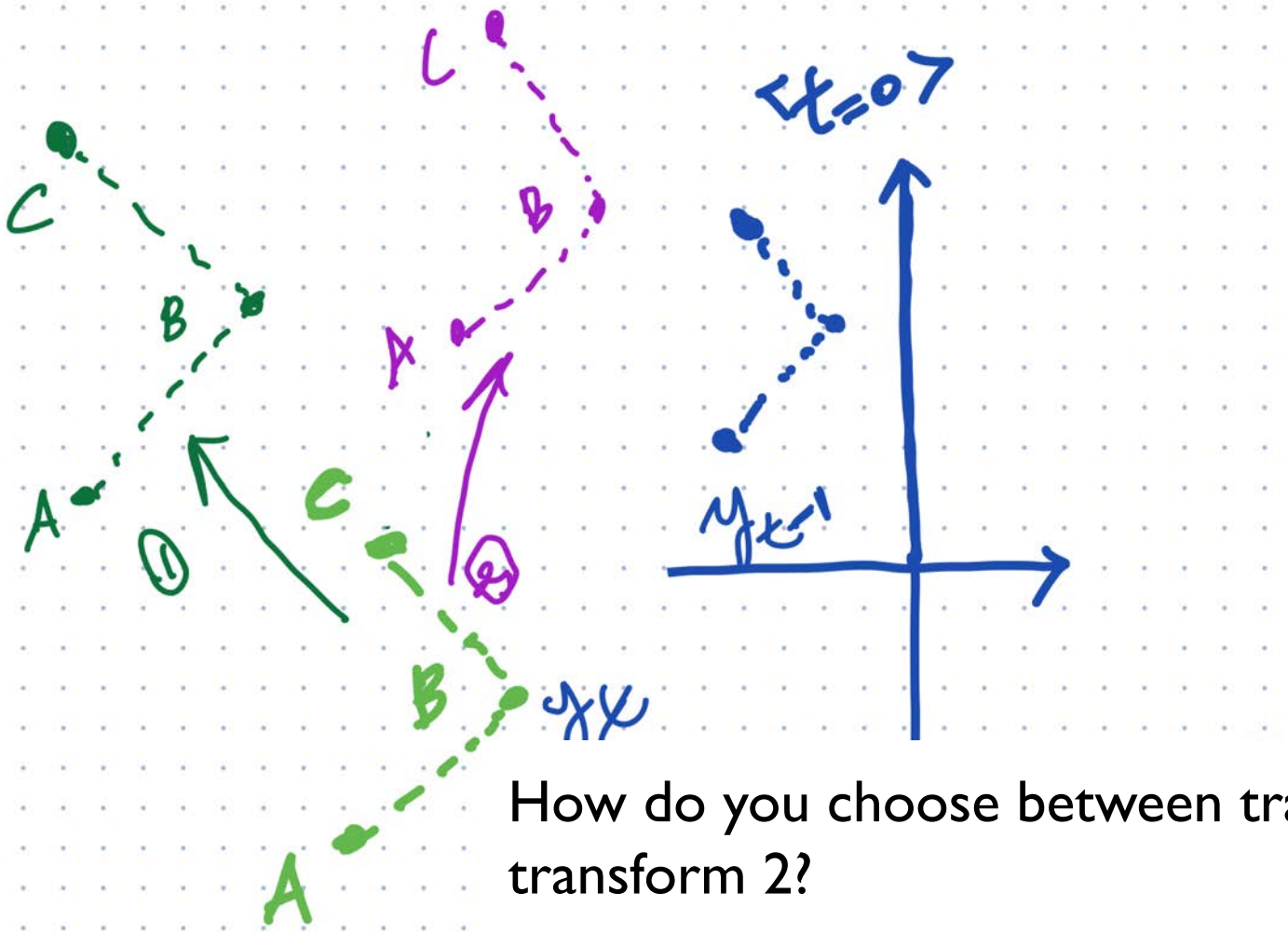
Iterative search for best transform

1. Make some initial “guess” of R (current guess)
2. For each point in new scan (t=k+1), find closest point in previous set (t=k) (**correspondence search**)
3. Make another “better guess” of R (next guess)
4. Set next guess to current guess, repeat steps 2-4 until converges

What makes a “better guess”? → Need a metric

How do you choose between 1 or II?

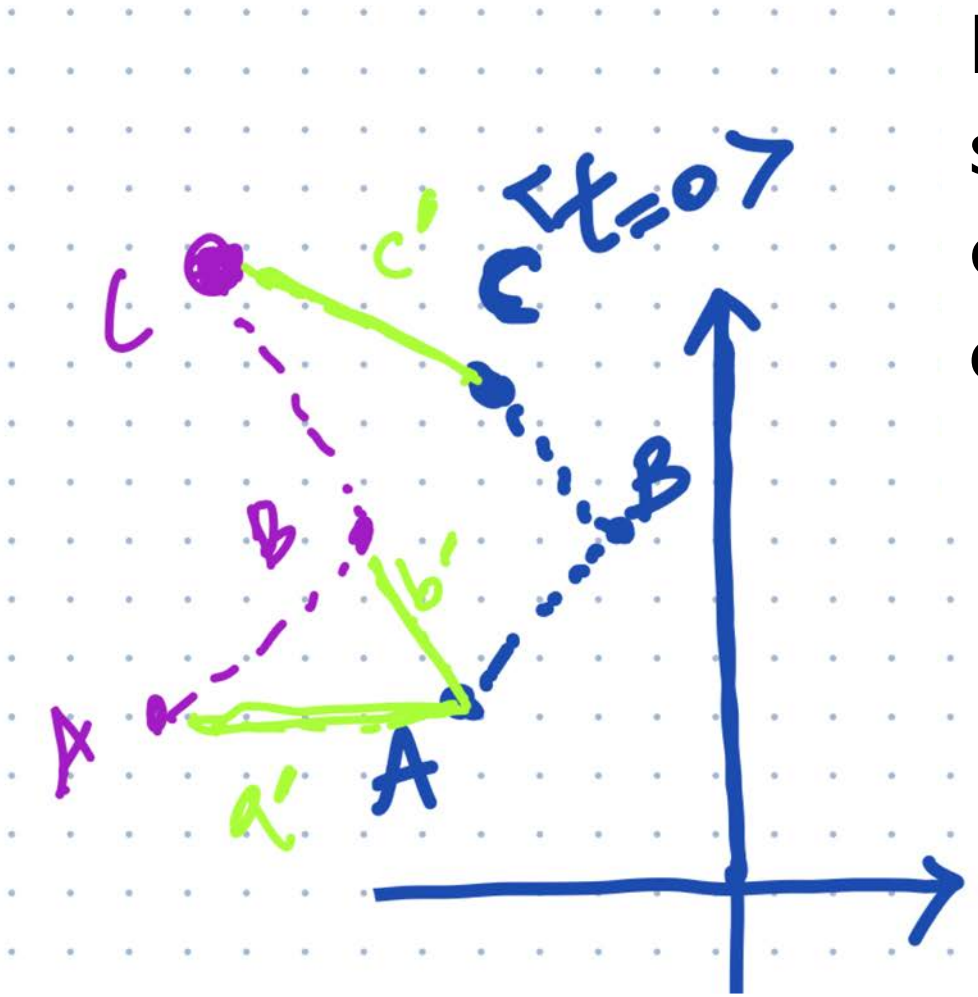
(Assume correspondences have been foud)



How do you choose between transform I or transform 2?

Candidate Match Function

Point-to-scan point

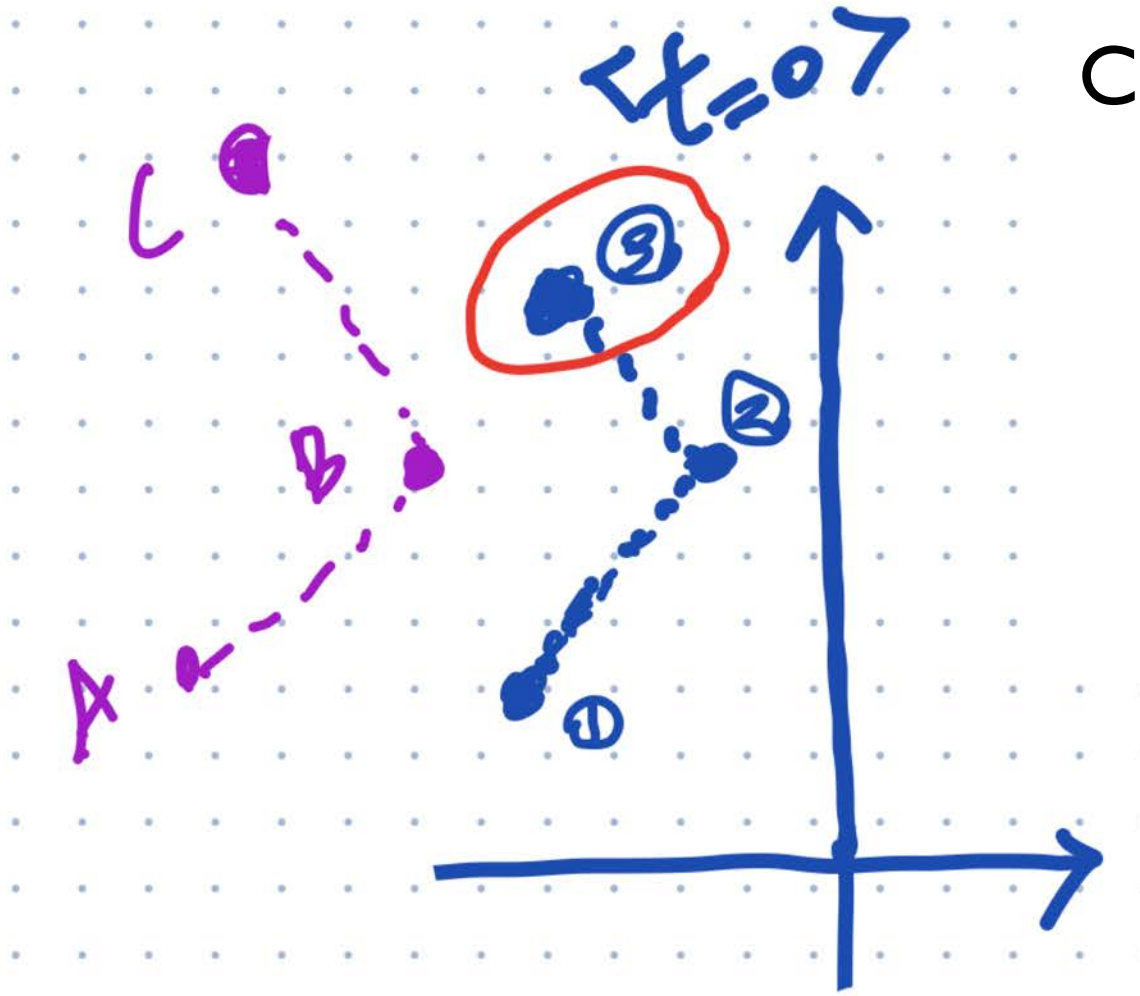


Find closest point and sum the square of distances to the closest point.

$$|a'|^2+|b'|^2+|c'|^2$$

Better Candidate Match Function

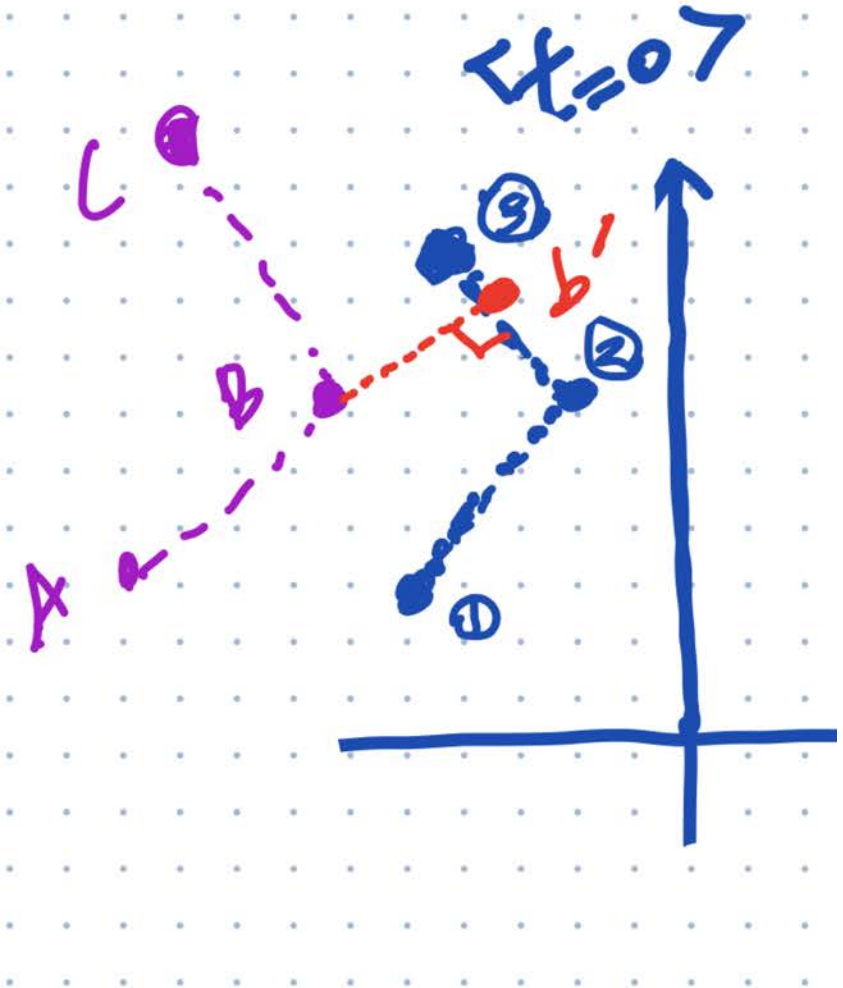
Point to projected point (point-to-point) metric:



Closest point to B is 3

Better Candidate Match Function

Find projection of point onto reference surface:

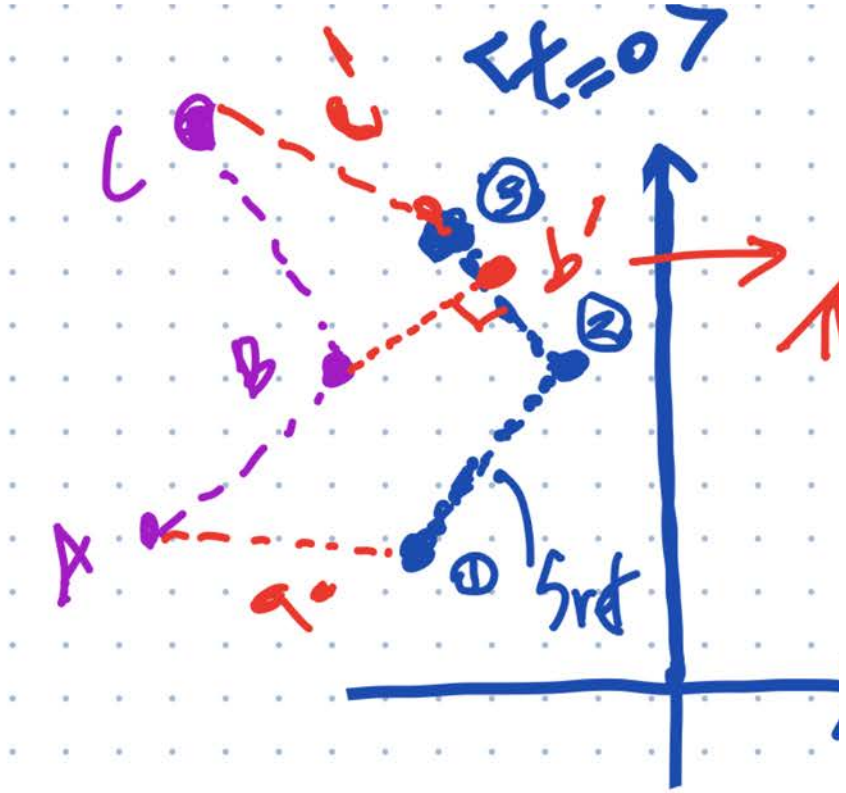


Closest point to B is 3

Find projected **point** on line segment 2-3, b'

Better Candidate Match Function

Squared sum of projected distances (point-to-point)



Closest point to B is 3

Find projected **point** on line segment 2-3, b'

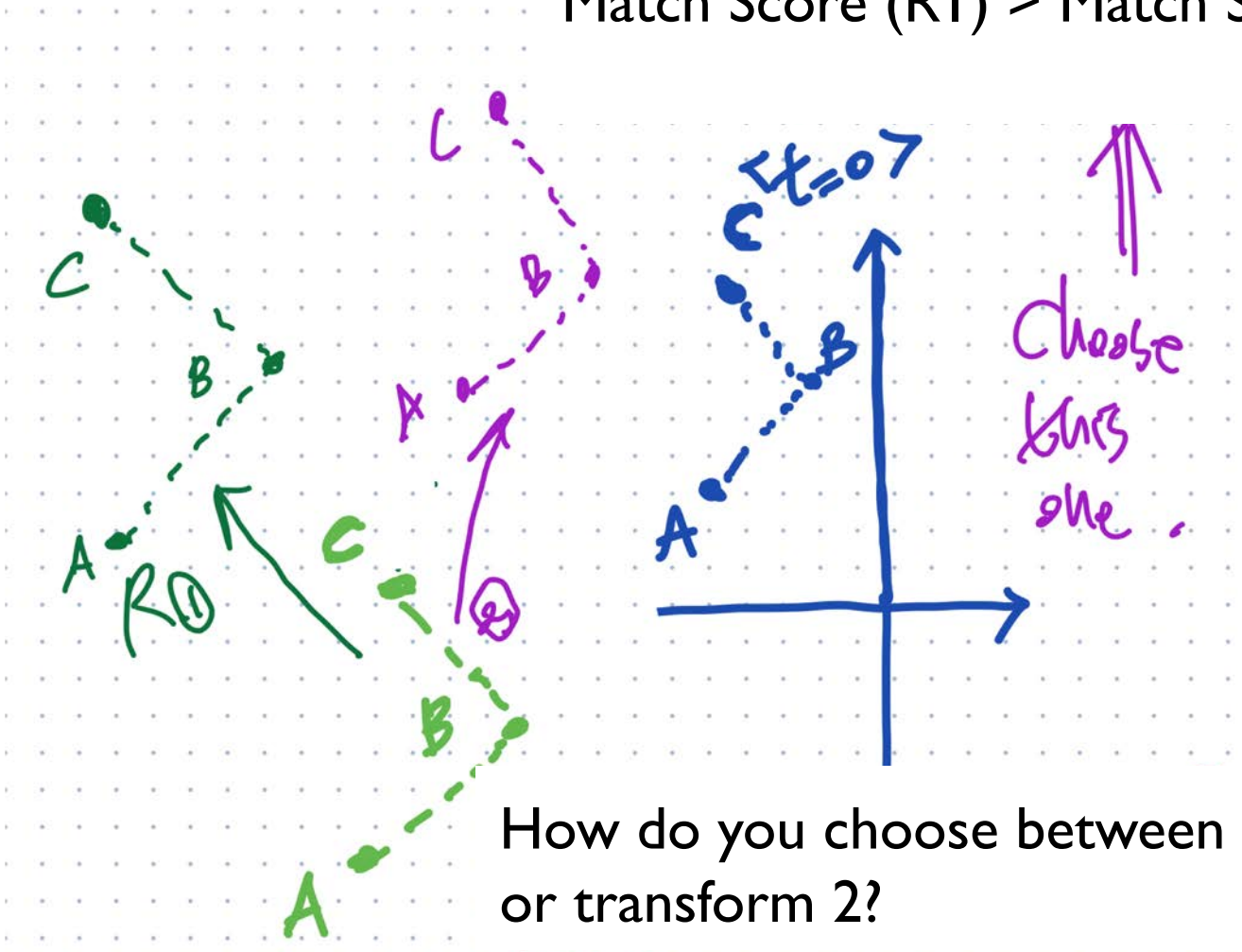
Square the sum of projected distances

$$Score = \|a'\|^2 + \|b'\|^2 + \|c'\|^2$$

How do you choose between I or II

Choose R2

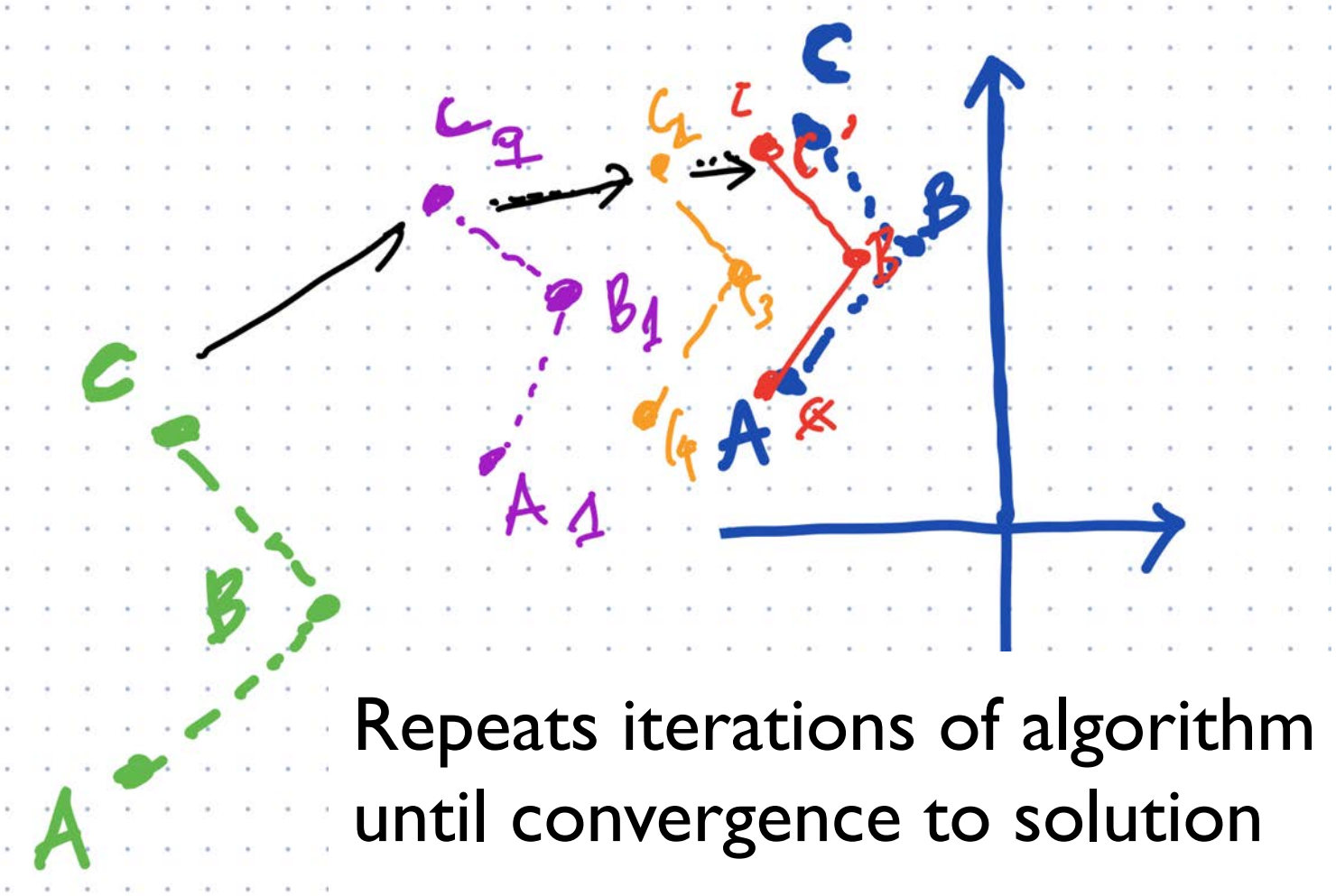
Match Score (R1) > Match Score (R2)



How do you choose between transform 1 or transform 2?

Iterative Matching

Uses point-to-point metric



Repeats iterations of algorithm until convergence to solution

Iterative Closest Point (ICP) Algorithm

- Given: two corresponding point sets:

$$X = \{x_1, \ldots, x_n\}$$

$$P = \{p_1, \ldots, p_n\}$$

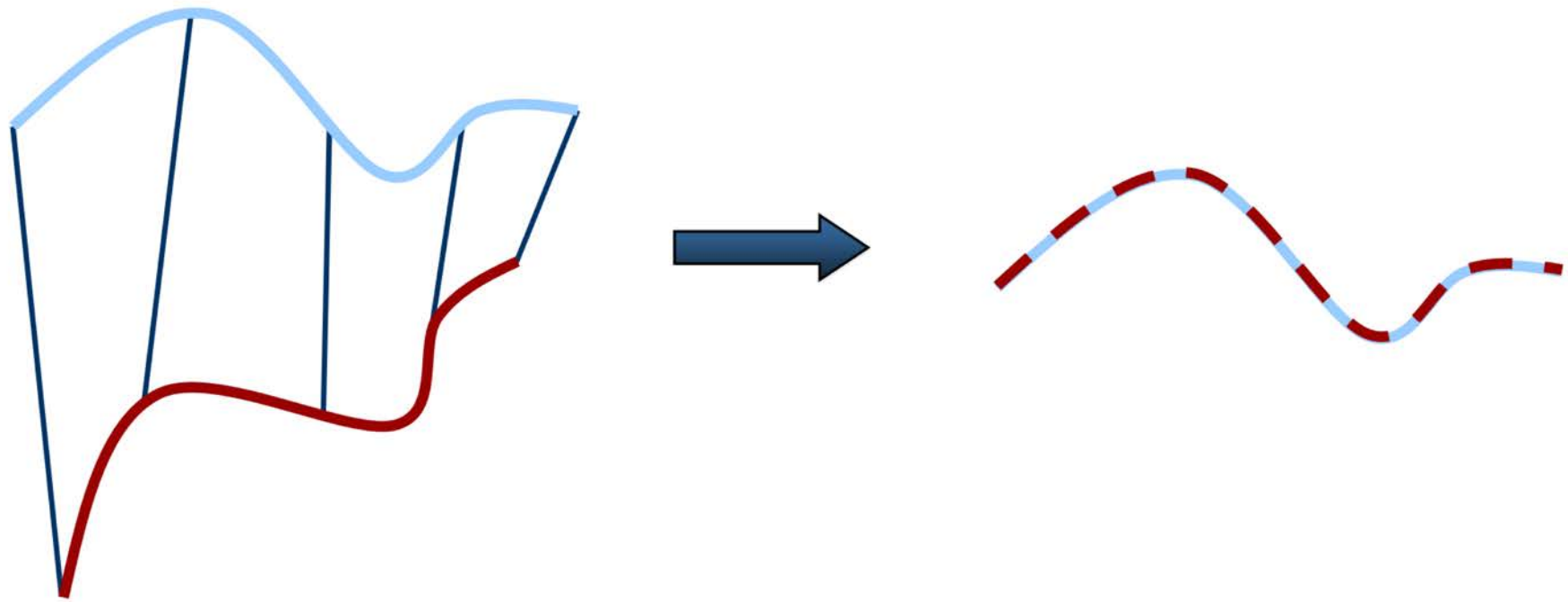
- Wanted: translation t and rotation R that minimizes the sum of the squared error:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2$$

Where x_i and p_i are corresponding points.

Key Idea

- If the correct correspondences are known, the correct relative rotation/translation can be calculated in closed form.



Center of Mass

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \quad \text{and} \quad \mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

are the centers of mass of the two point sets.

Idea:

- Subtract the corresponding center of mass from every point in the two point sets before calculating the transformation.
- The resulting point sets are:

$$X' = \{x_i - \mu_x\} = \{x'_i\}$$

and

$$P' = \{p_i - \mu_p\} = \{p'_i\}$$

SVD

Let $W = \sum_{i=1}^{N_p} x'_i p'^T_i$

denote the singular value decomposition (SVD) of W by:

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

where $U, V \in \mathbb{R}^{3\times3}$ are unitary, and

$\sigma_1 \geq \sigma_2 \geq \sigma_3$ are the singular values of W.

SVD

Theorem (without proof):

If rank(W) = 3, the optimal solution of E(R,t) is unique and is given by:

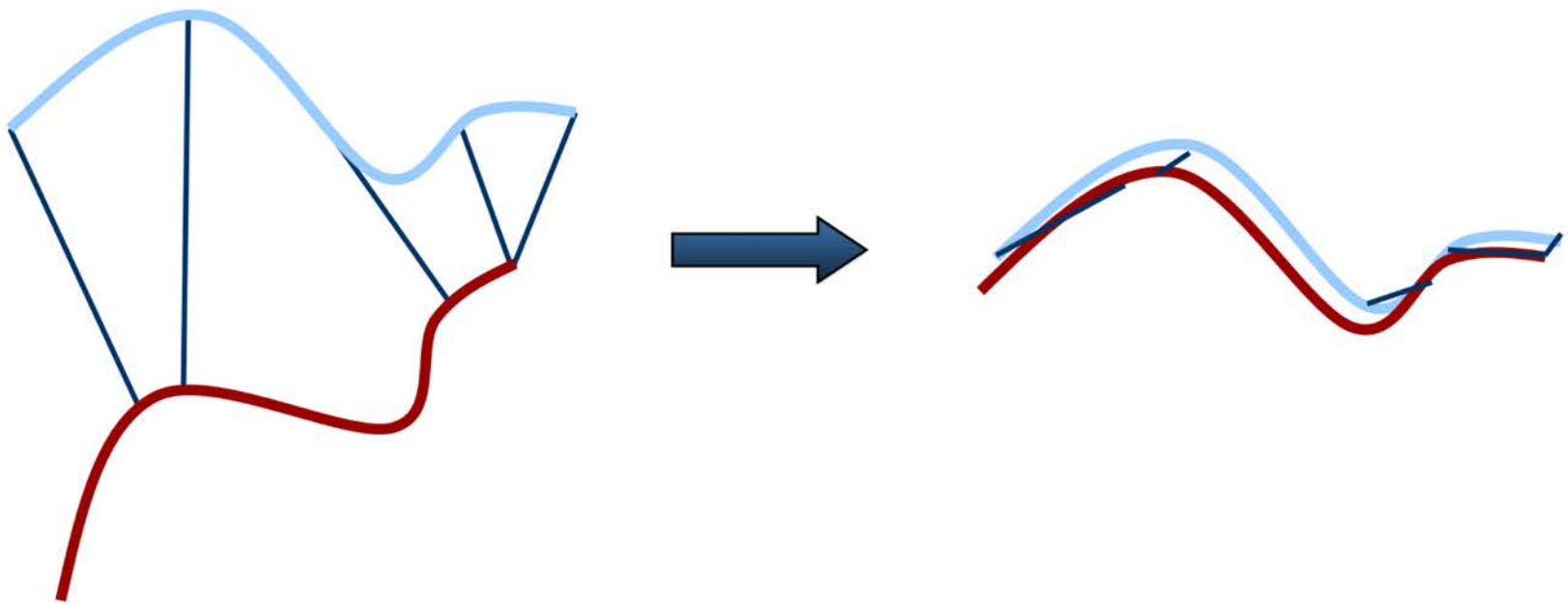
$$R = UV^T$$
$$t = \mu_x - R\mu_p$$

The minimal value of error function at (R,t) is:

$$E(R,t) = \sum_{i=1}^{N_p} (||x_i'||^2 + ||y_i'||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

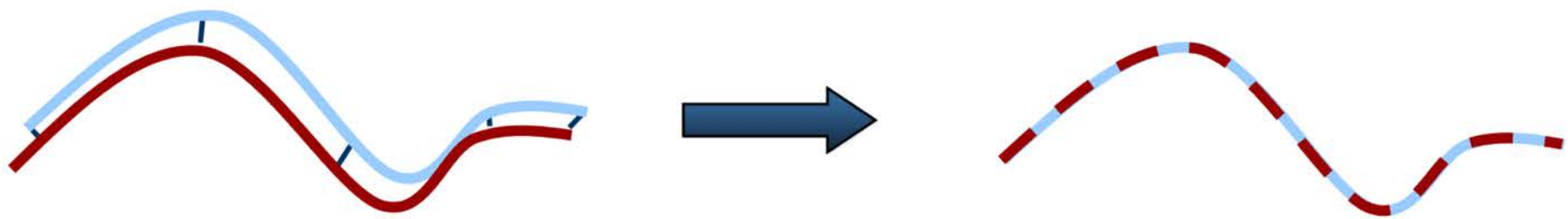
ICP with Unknown Data Association

- If correct correspondences are not known, it is generally impossible to determine the optimal relative rotation/translation in one step



ICP-Algorithm

- Idea: iterate to find alignment
- Iterated Closest Points (ICP) [Besl & McKay 92]
- Converges if starting positions are "close enough"



ICP-Variants

- Variants on the following stages of ICP have been proposed:

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Performance of Variants

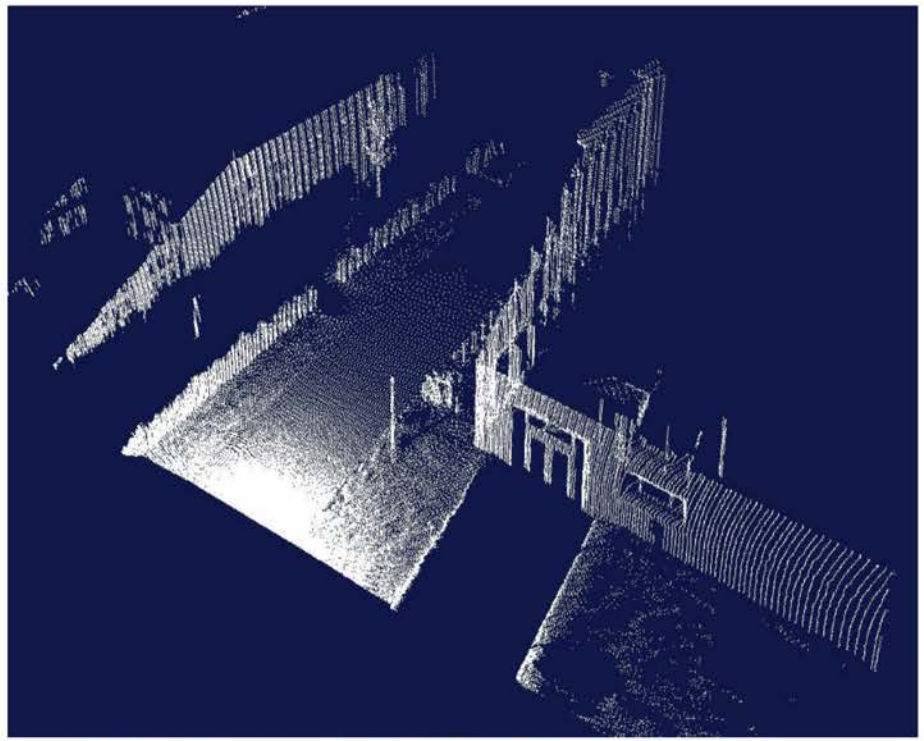
- Various aspects of performance:
  - Speed
  - Stability (local minima)
  - Tolerance wrt. noise and/or outliers
  - Basin of convergence
    (maximum initial misalignment)
- Here: properties of these variants

Selecting Source Points

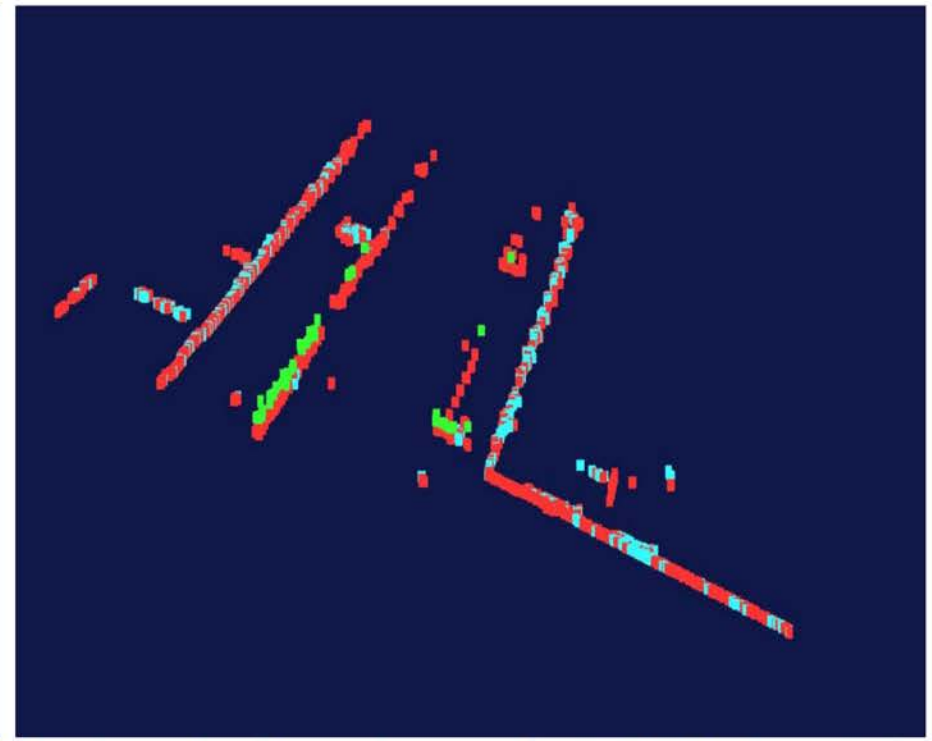
- Use all points
- Uniform sub-sampling
- Random sampling
- Feature based Sampling
- Normal-space sampling
  - Ensure that samples have normals distributed as uniformly as possible

Feature-Based Sampling

- try to find “important” points
- decrease the number of correspondences
- higher efficiency and higher accuracy
- requires preprocessing



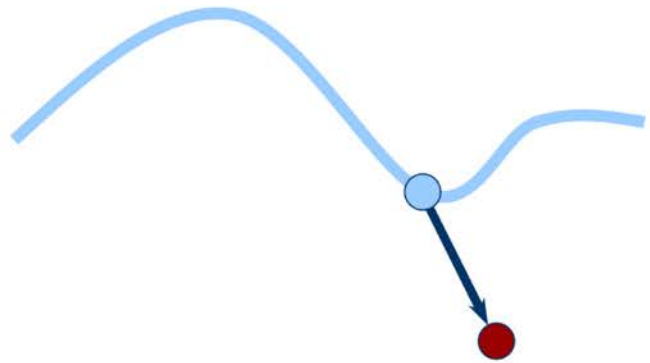
3D Scan (~200.000 Points)



Extracted Features (~5.000 Points)

Closest-Point Matching

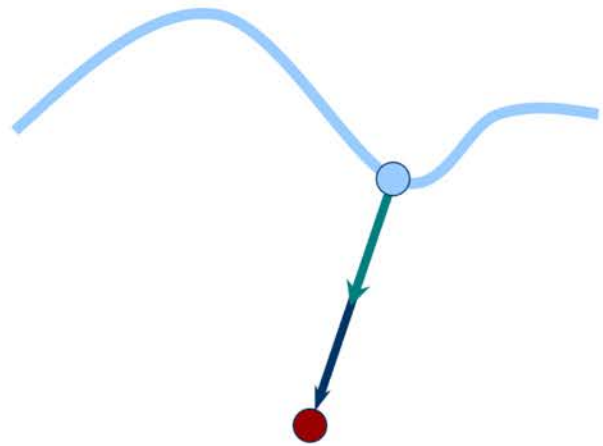
- Find closest point in other the point set



Closest-point matching generally stable, but slow and requires preprocessing

Normal Shooting

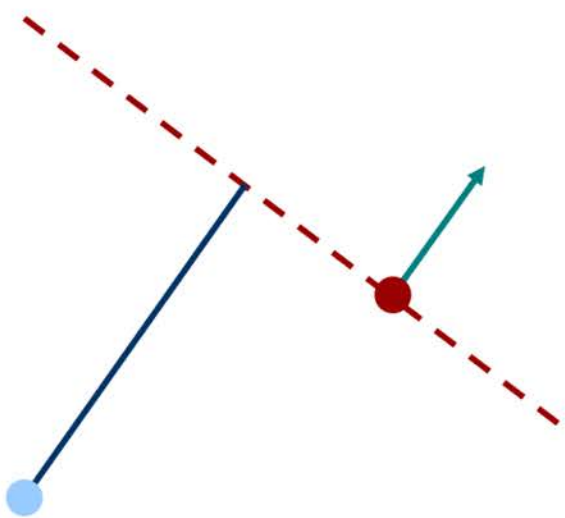
- Project along normal, intersect other point set

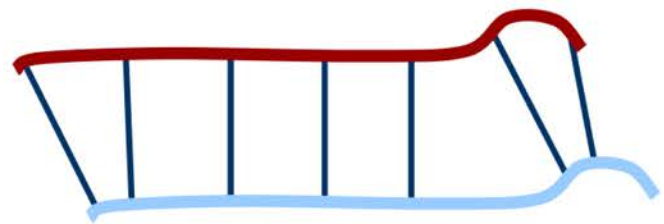


Slightly better than closest point for smooth structures, worse for noisy or complex structures

Point-to-Plane Error Metric

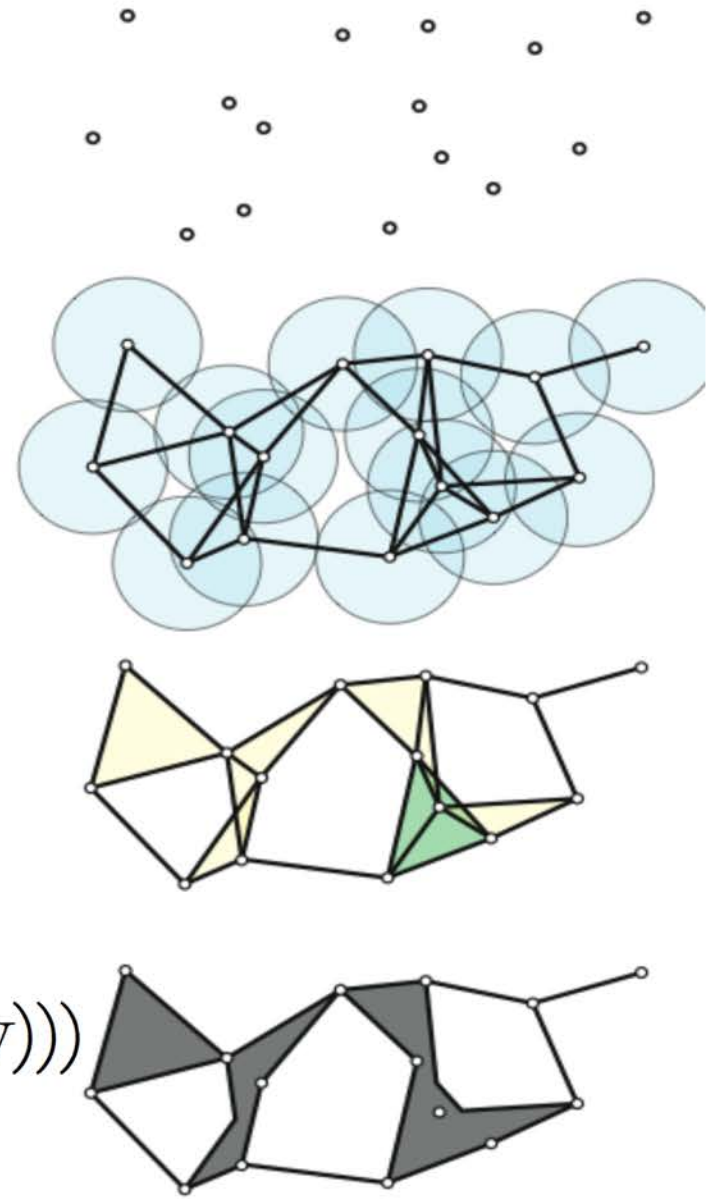
- Using point-to-plane distance instead of point-to-point lets flat regions slide along each other [Chen & Medioni 91]





Data Types

- Point sets
- Line segment sets (polylines)
- Implicit curves : $f(x,y,z) = 0$
- Parametric curves : $(x(u),y(u),z(u))$
- Triangle sets (meshes)
- Implicit surfaces : $s(x,y,z) = 0$
- Parametric surfaces $(x(u,v),y(u,v),z(u,v)))$



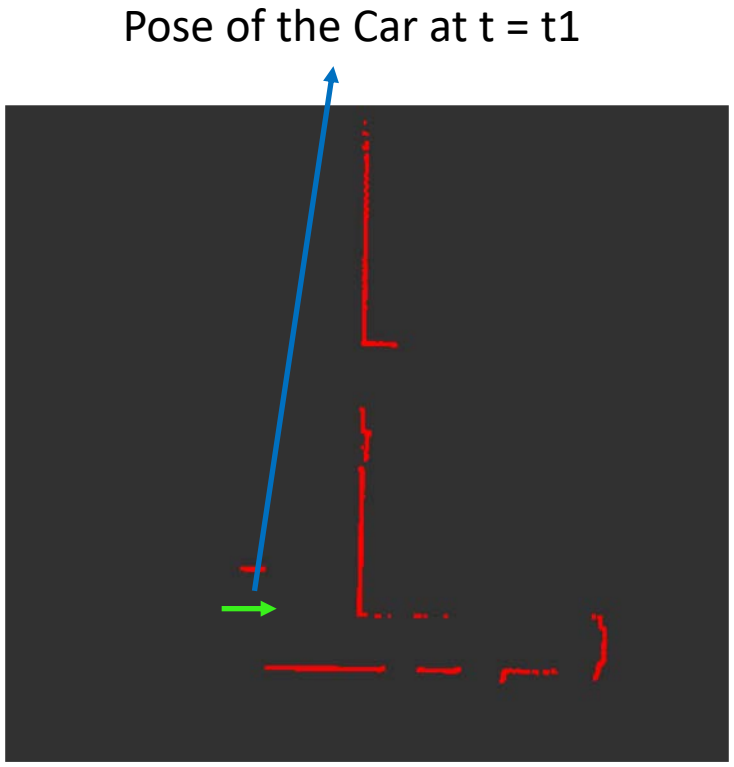
Various Different Ways to Realize Scan-Matching

- Iterative closest point (ICP)
- Scan-to-scan
- Scan-to-map
- Map-to-map
- Feature-based
- RANSAC for outlier rejection

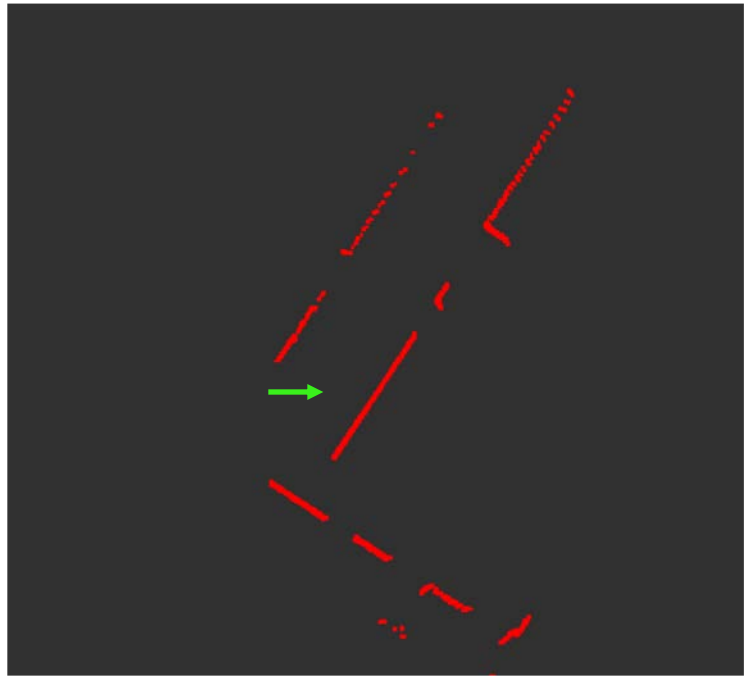
ICP Assumptions and Limitations

- Assumes a sufficiently fast scan rate
  - Significant overlap will exist between sequential scans
  - Points with high correspondence will exist
- Assumes surface will be non-smooth and heterogeneous
  - Smooth surface will look the same to the algorithm

Scan Matching

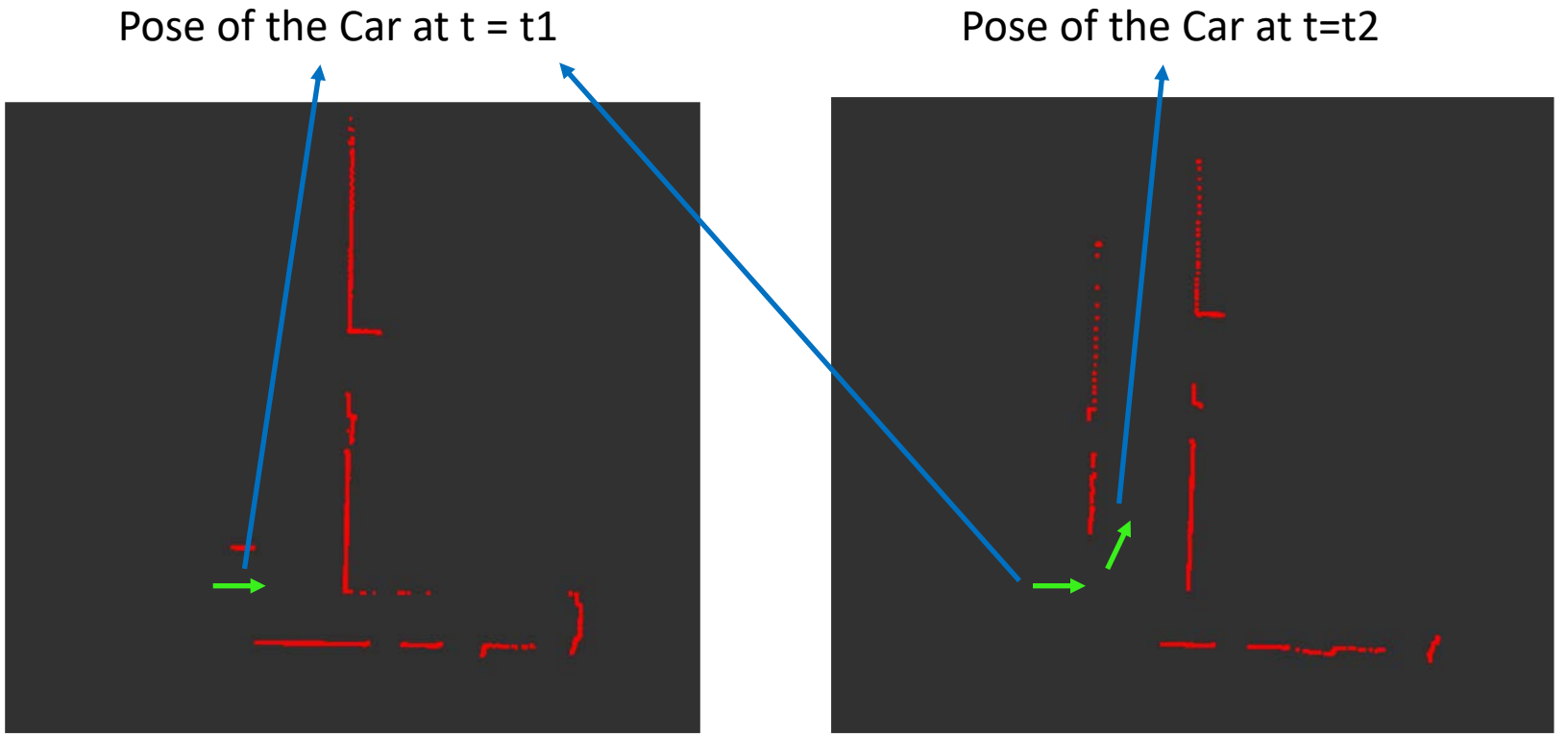


Laser Scans w.r.t car at Time t = t_1



Laser Scans w.r.t car at Time t = t_2

Scan Matching



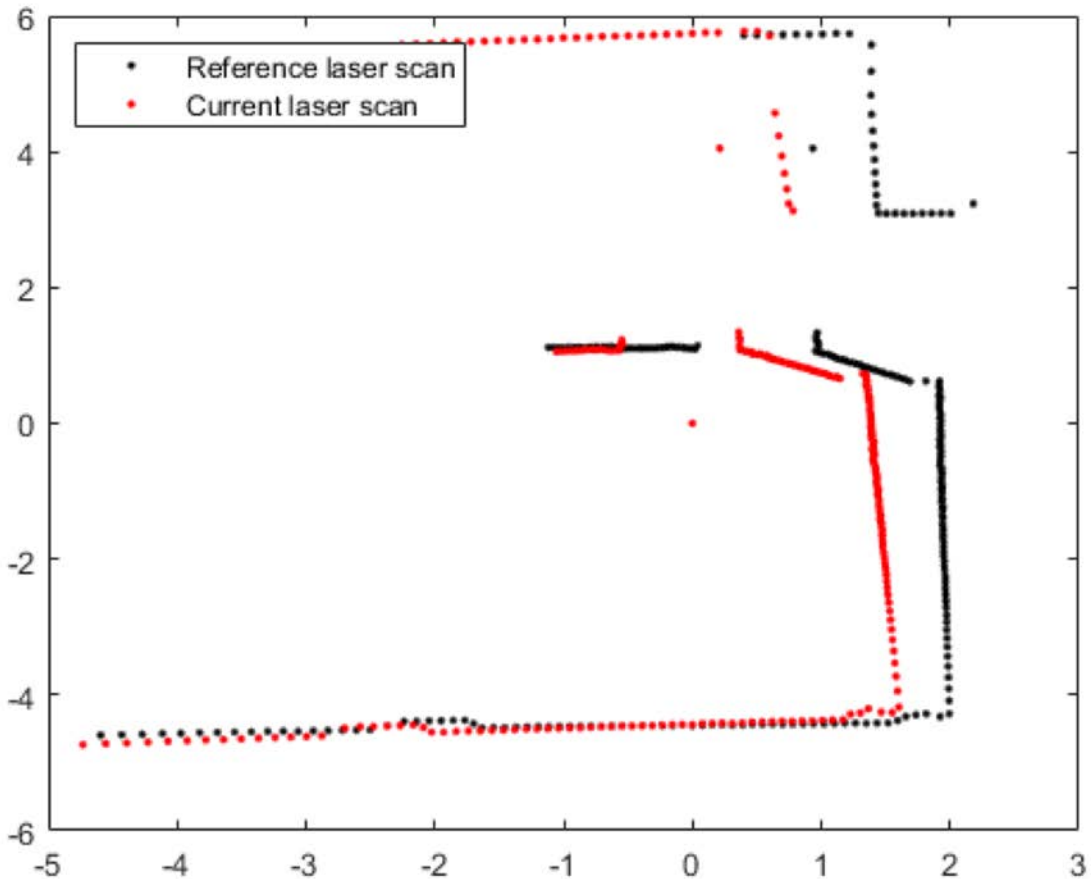
Laser Scans w.r.t car at Time t = t_1

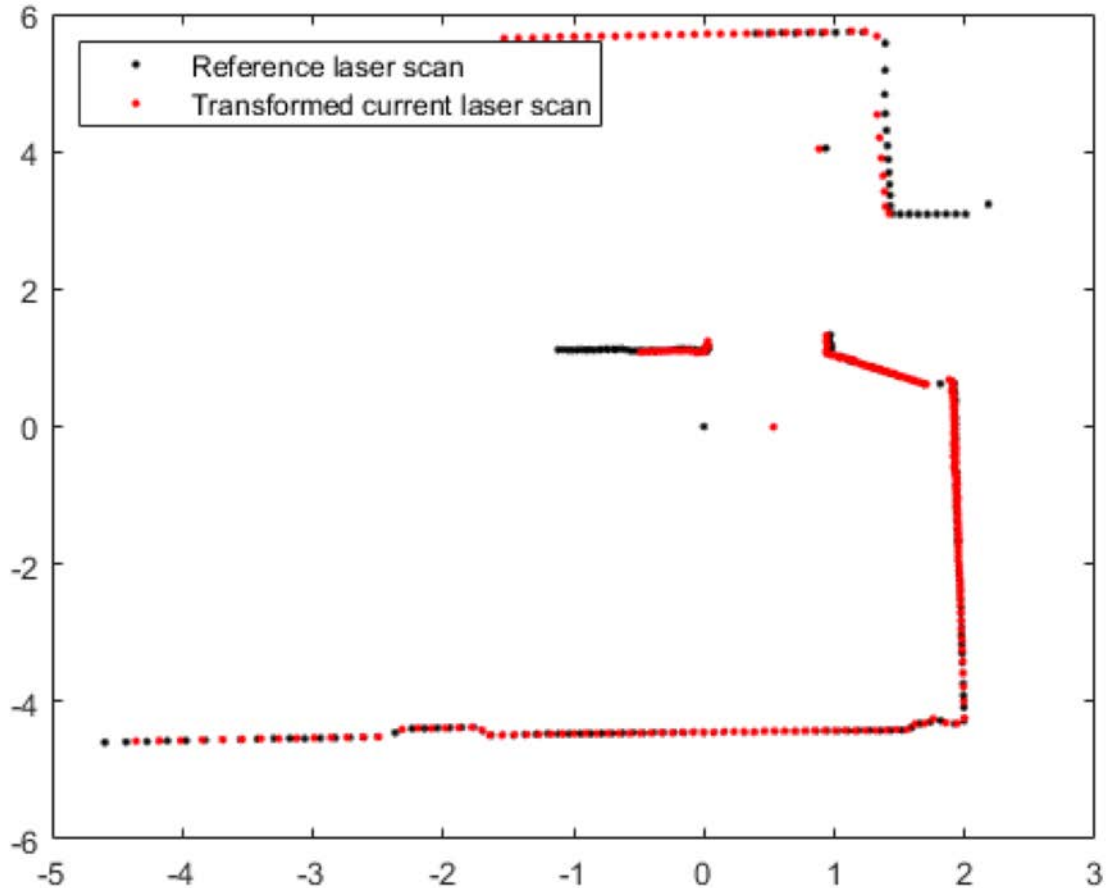
Laser Scans w.r.t car at Time t = t_2

```
transform = matchScans(currentScan,referenceScan)
```

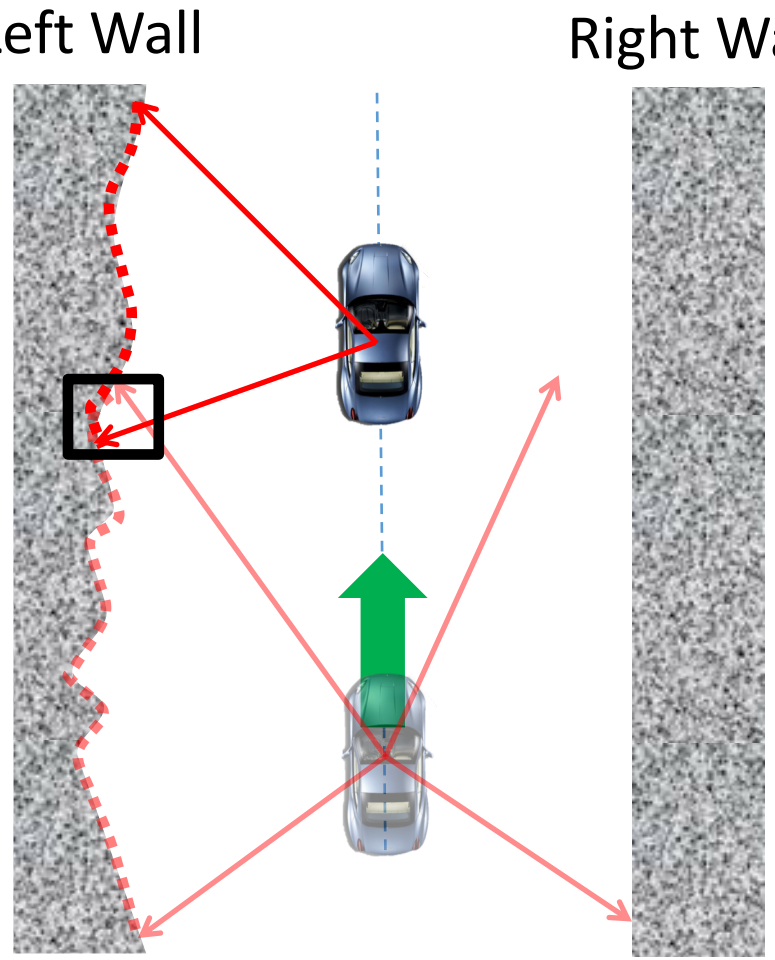
```
transform = 1×3

    0.5348   -0.0065   -0.0336
```



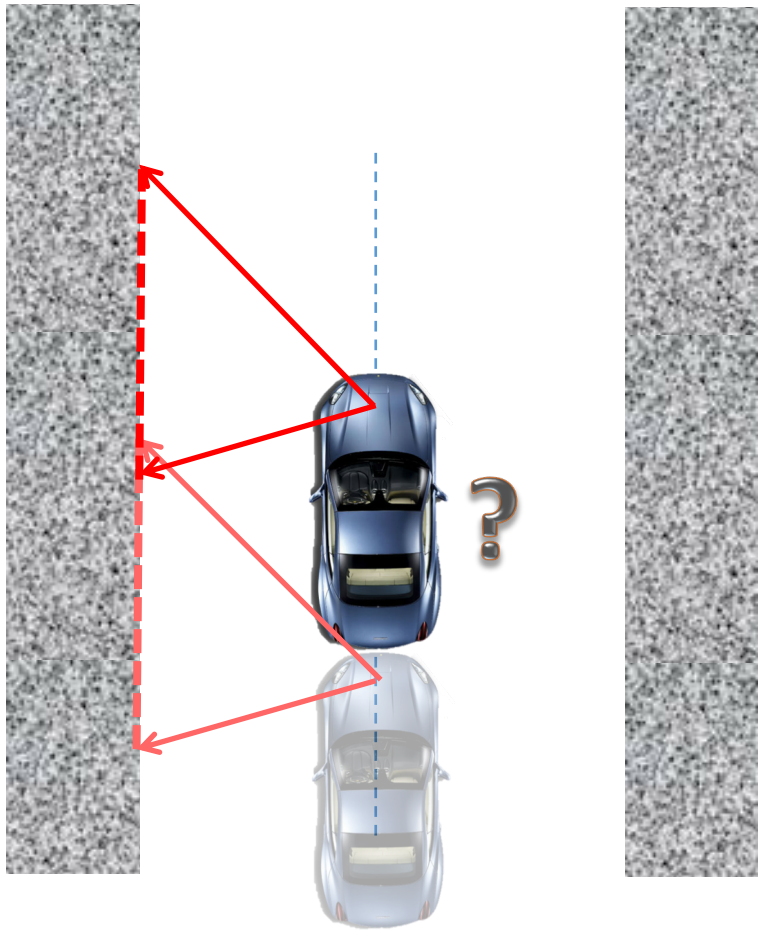


Scan matching: requirements



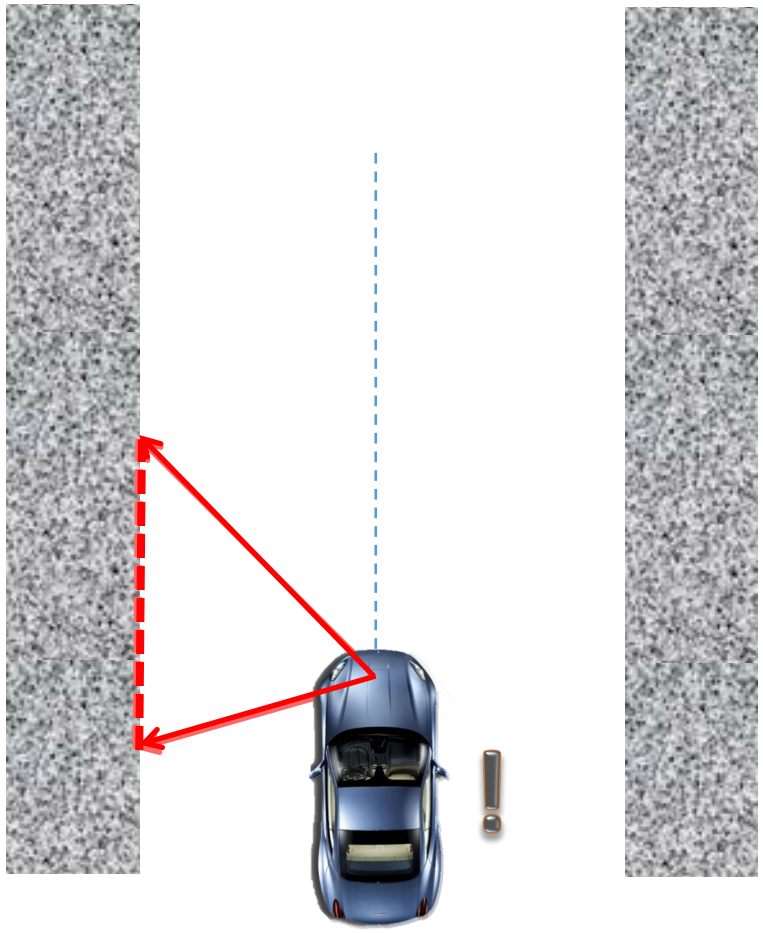
- Sufficiently fast scan rate
- A slow scan rate can lead to few matches between scans.
- Not really a risk for today's LIDARs

Scan matching : requirements



- Non-smooth, or heterogeneous, surfaces.
- Smooth surfaces all look the same to the matching algorithm.

Scan matching : requirements



- Non-smooth, or heterogeneous, surfaces.
- Best match between scans is if car did not move.

Failed scan matching

hokuyo_node

electric | fuerte | groovy | hydro | indigo | jade | Documentation Status

Package Summary

✔ Released ✖ No API documentation

A ROS node to provide access to SCIP 2.0-compliant Hokuyo laser range finders (including 04LX).

- Maintainer status: maintained
- Maintainer: Chad Rockey <chadrockey AT gmail DOT com>
- Author: Brian P. Gerkey, Jeremy Leibs, Blaise Gassend
- License: LGPL

Package Links

Tutorials
Troubleshooting
FAQ
Change List
Reviews

Dependencies (9)
Jenkins jobs (4)

Wiki

Distributions
ROS/Installation
ROS/Tutorials
RecentChanges
hokuyo_node

Page

Immutable Page
Info
Attachments
More Actions:

urg_node

groovy | hydro | indigo | jade | kinetic | Documentation Status

Package Summary

✔ Released ✔ Continuous integration ✔ Documented

urg_node

- Maintainer status: maintained
- Maintainer: Tony Baltovski <tony.baltovski AT gmail DOT com>
- Author: Chad Rockey <chadrockey AT gmail DOT com>, Mike O'Driscoll <modriscoll AT clearpath DOT ai>
- License: BSD
- Bug / feature tracker: https://github.com/ros-drivers/urg_node/issues

Package Links

- **Code API**
- **Msg API**
- FAQ
- Changelog
- Change List
- Reviews

Dependencies (14)

Used by (3)

Jenkins jobs (10)

laser_scan_matcher

electric | fuerte | indigo | jade | kinetic | Documentation Status

***scan_tools**: laser_ortho_projector | laser_scan_matcher | laser_scan_sparsifier | laser_scan_splitter | ncd_parser | polar_scan_matcher | scan_to_cloud_converter*

Package Summary

✔ Released ✔ Continuous integration ✔ Documented

An incremental laser scan matcher, using Andrea Censi's Canonical Scan Matcher (CSM) implementation. See the web site for more about CSM. NOTE the CSM library is licensed under the GNU Lesser General Public License v3, whereas the rest of the code is released under the BSD license.

- Maintainer status: maintained
- Maintainer: Ivan Dryanovski <ccnyroboticslab AT gmail DOT com>, Carlos <ciaramillo AT gc.cuny DOT edu>, Isaac

Package Links

Code API
FAQ
Changelog
Change List
Reviews

Dependencies (10)
Used by (1)
Jenkins jobs (10)

laser_scan_matcher: ROS

4.3.6 Scan matching

`~max_iterations` (`int`, default: 10)

Maximum ICP cycle iterations

`~max_correspondence_dist` (`double`, default: 0.3)

Maximum distance for a correspondence to be valid

`~max_angular_correction_deg` (`double`, default: 45.0)

Maximum angular displacement between scans, in degrees

`~max_linear_correction` (`double`, default: 0.50)

Maximum translation between scans (m)

`~epsilon_xy` (`double`, default: 0.000001)

A threshold for stopping (m)

`~epsilon_theta` (`double`, default: 0.000001)

A threshold for stopping (rad)

`~outliers_maxPerc` (`double`, default: 0.90)

Percentage of correspondences to consider: if 0.90, always discard the top 10% of correspondences with more error

scan_tools

electric | fuerte | indigo | jade | kinetic | Documentation Status

scan_tools: *laser_ortho_projector* | *laser_scan_matcher* | *laser_scan_sparsifier* | *laser_scan_splitter* | *ncd_parser* | *polar_scan_matcher* | *scan_to_cloud_converter*

Package Summary

✔ Released ✔ Continuous integration ✔ Documented

Laser scan processing tools.

- Maintainer status: maintained
- Maintainer: Ivan Dryanovski <ccnyroboticslab AT gmail DOT com>, Carlos <ciaramillo AT gc cuny DOT edu>, Isaac I.Y. Saito <iiysaito AT opensource-robotics tokyo DOT jp>

Package Links

FAQ
Changelog
Change List
Reviews

Dependencies (8)
Jenkins jobs (10)

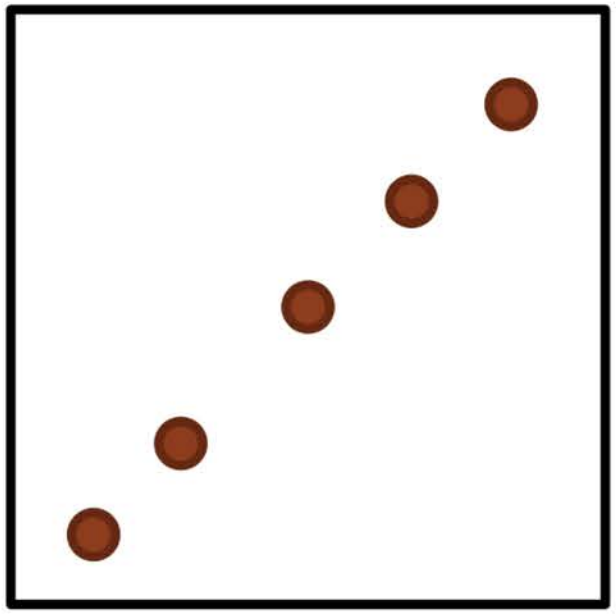
Wiki

Distributions
ROS/Installation
ROS/Tutorials
RecentChanges
scan_tools

Page

Immutable Page
Info
Attachments
More Actions:

Normal Distribution Transform

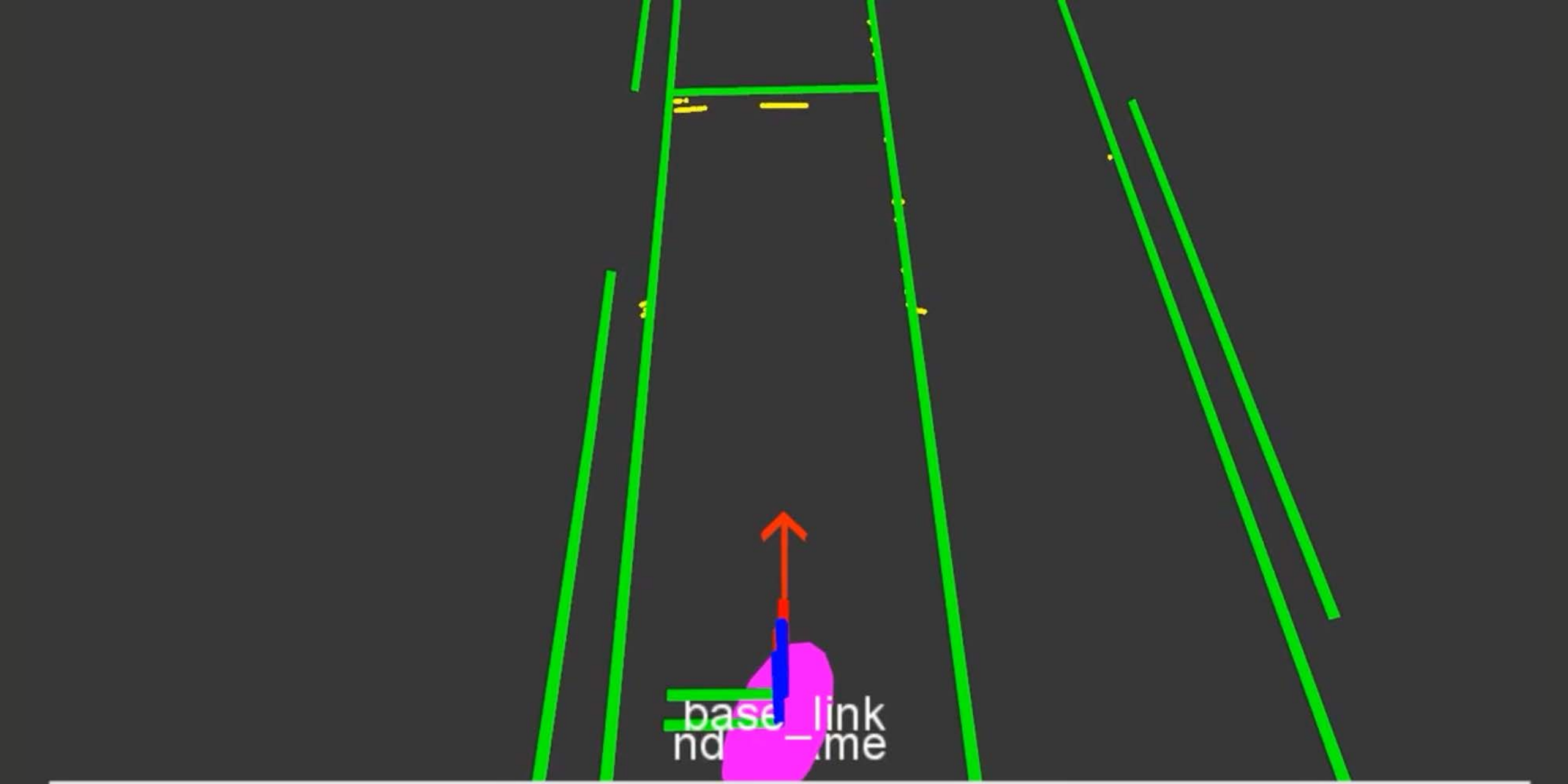


n points measured in a 2D cell.

n points, $2n$ parameters

Gaussian approximation for the 2D cell.

1 distribution, 5 parameters



NDT result is used as likelihood function in particle filtering algorithm.

Road marker matching is also used to improve localization accuracy.

Overall Algorithm

Input data:

- Reference scan, y_{t-1}, current scan y_t, initial guess of **roto-transformation** q_0

Output:

- Tranformation q, the converged solution representing the roto-transformation between the scans

Overall Algorithm

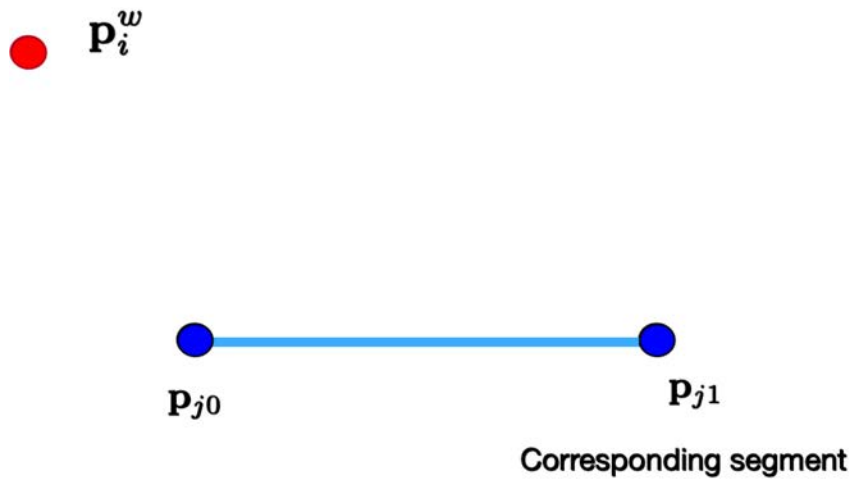
At t=k

1. Use previous guess q_k, transform coordinates of current scan into the frame of the previous scan
2. For each point, find closest line segment (point-to-line **correspondence search**)
3. Update transform
    a. Formulate point-to-line-error objective
    b. Find transform q_{k+1} that minimizes objective

(Repeat until convergence)

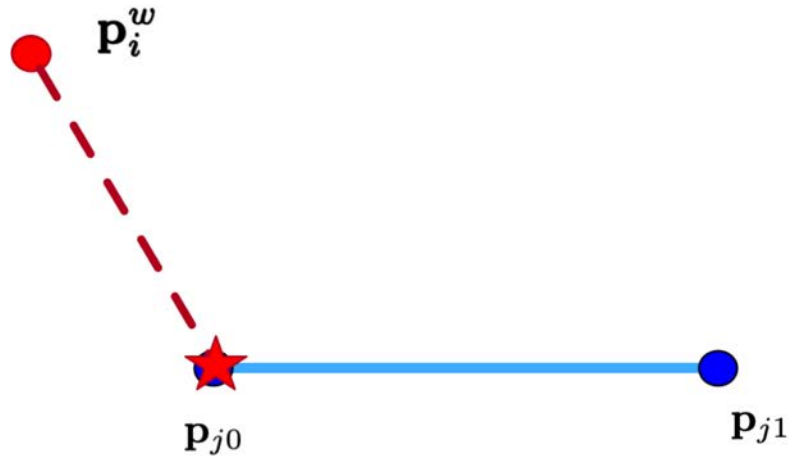
Point-to-point vs Point-to-line

Given a point in current scan and corresponding **line segment**



Point-to-point vs Point-to-line

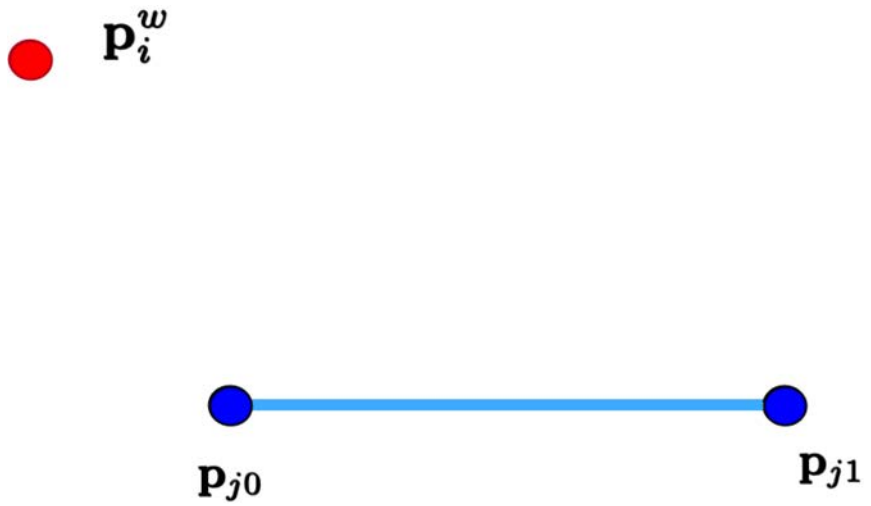
Point-to-point measures the distance as the distance to the nearest **point** **<u>on the segment</u>**



Projection onto a line segment

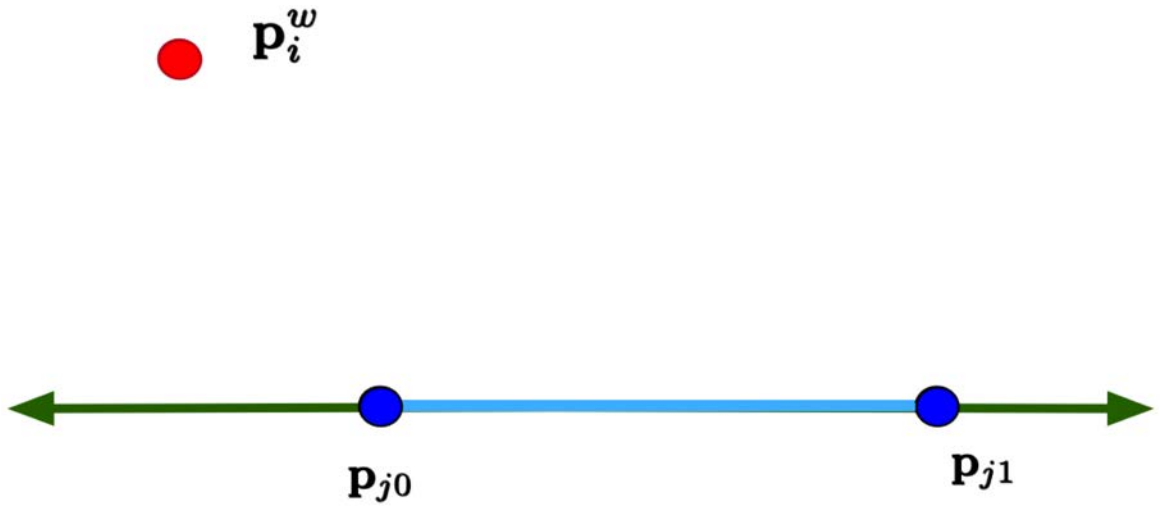
Point-to-point vs Point-to-line

Point-to-line measures the distance as the distance to the nearest **line** **containing the segment**



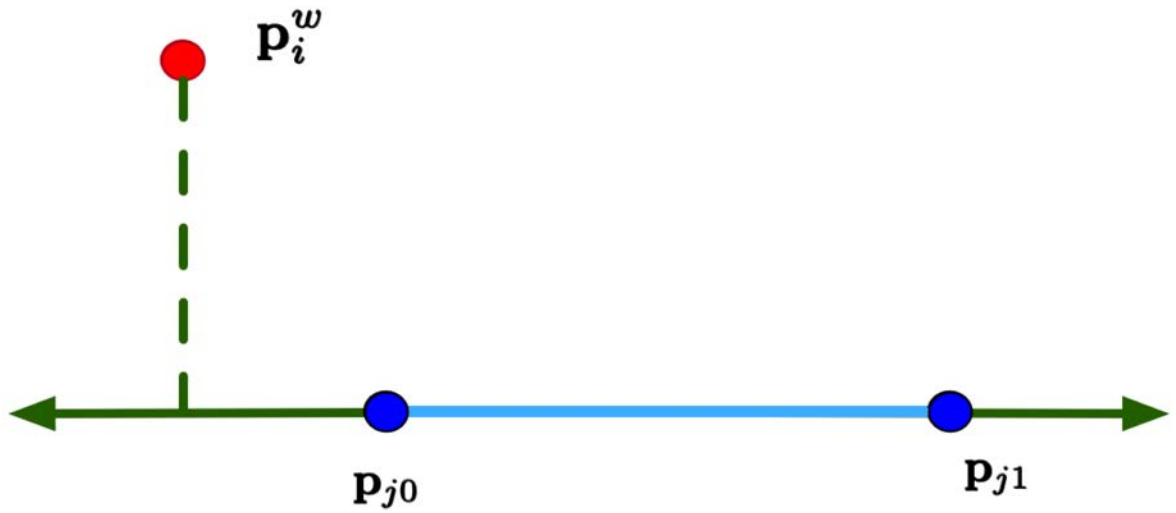
Point-to-point vs Point-to-line

Point-to-line measures the distance as the distance to the nearest **line** **containing the segment**



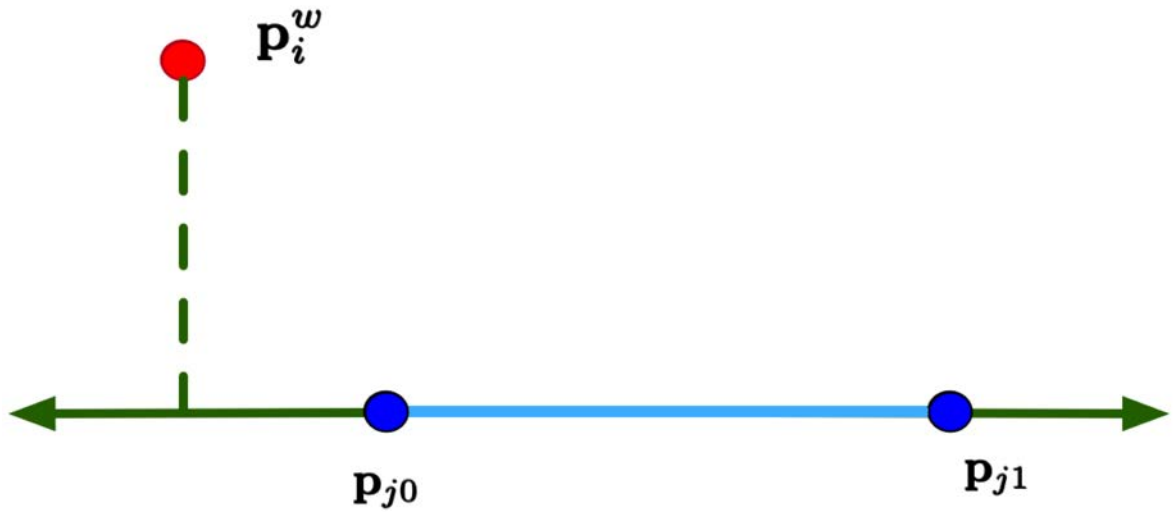
Point-to-point vs Point-to-line

Point-to-line measures the distance as the distance to the nearest **line** **containing the segment**



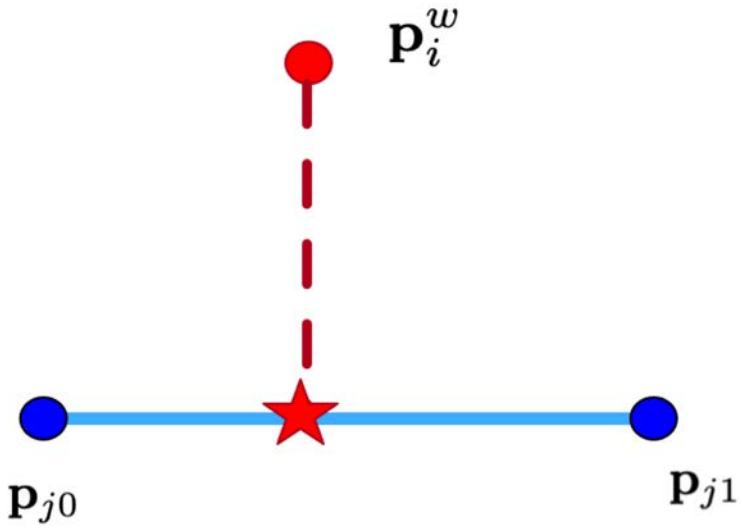
Point-to-line

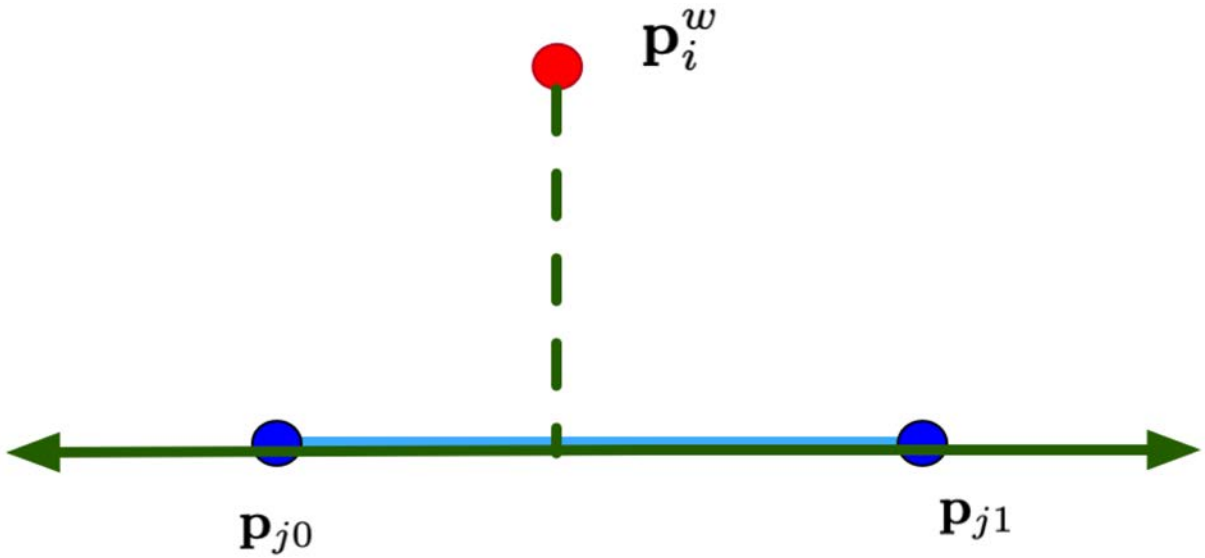
Point-to-line measures the distance as the distance to the nearest **line** **containing the segment**



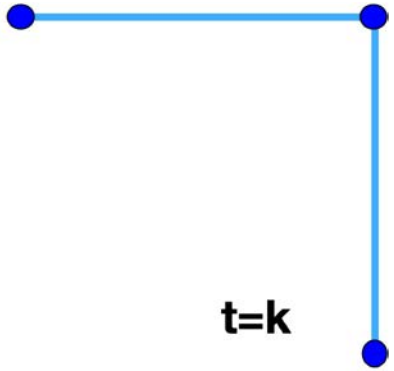
Point-to-point vs Point-to-line

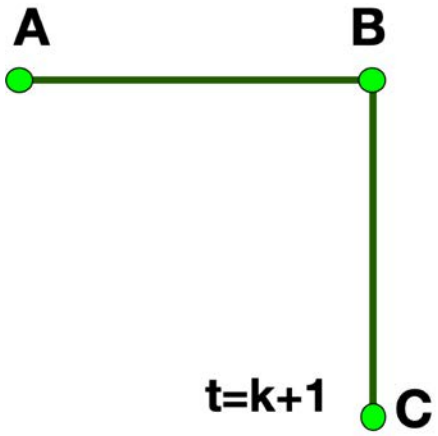
- Point-to-point and point-to-line may coincide in value
- Even in this case,
  - point-to-point **point** on the **nearest <u>line segment</u>**
  - point-to-line is finding distance to nearest **<u>line</u>**



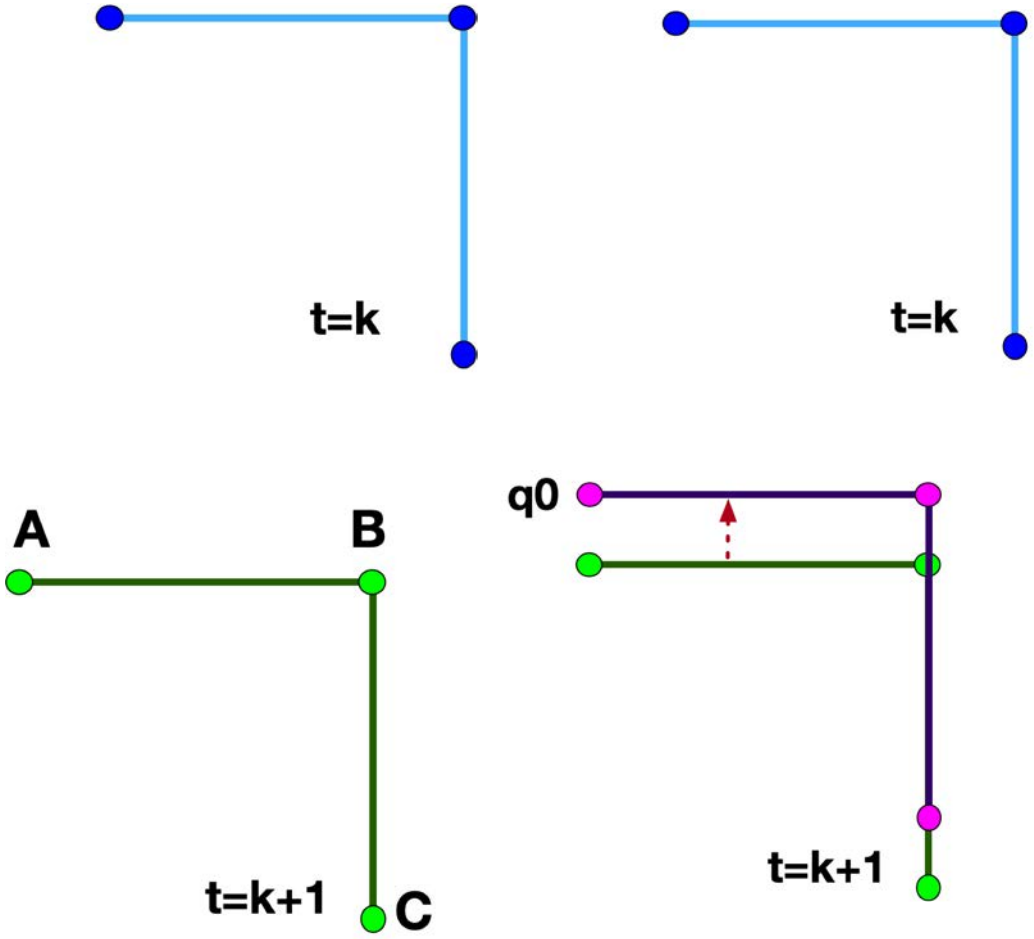


Why is it faster?

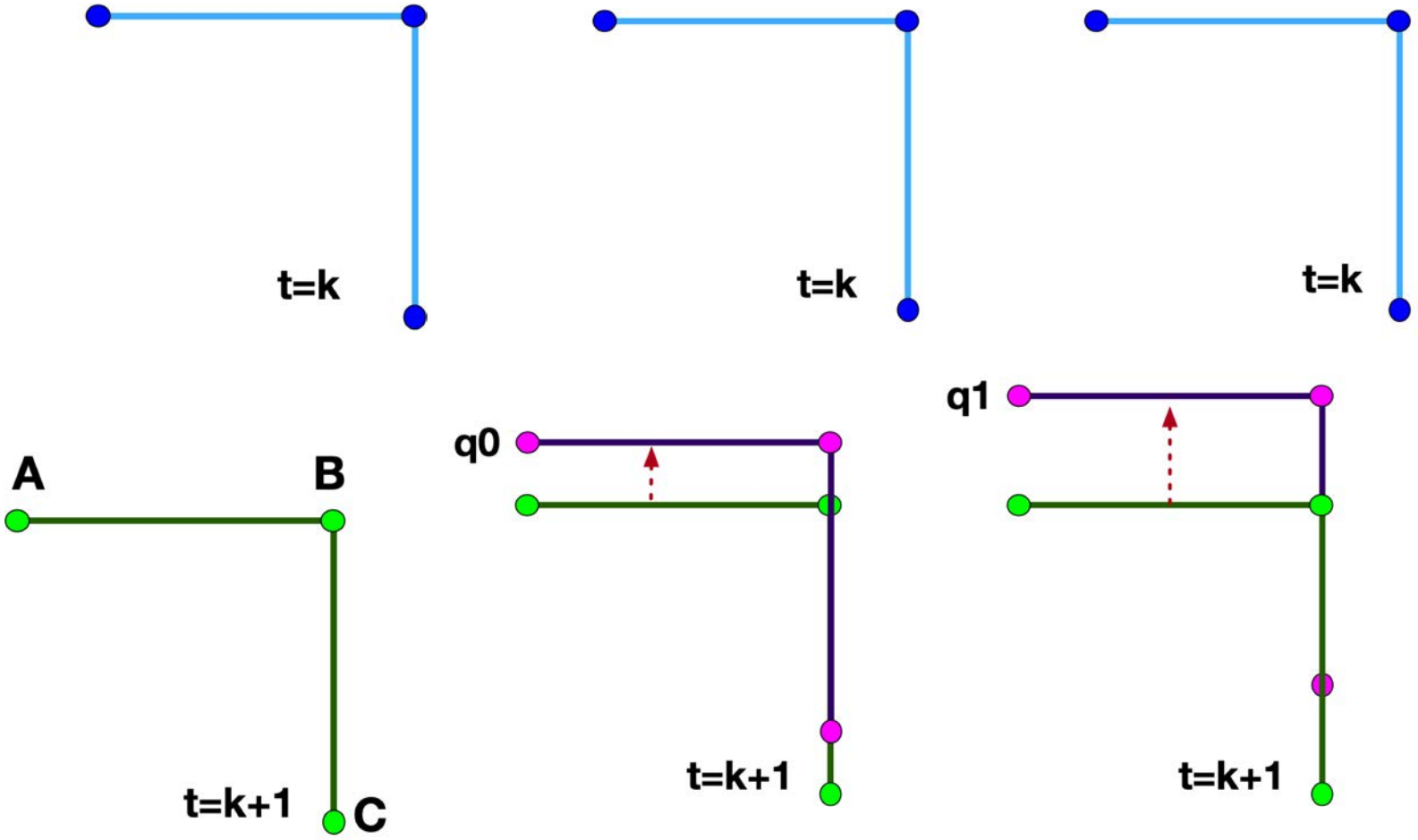




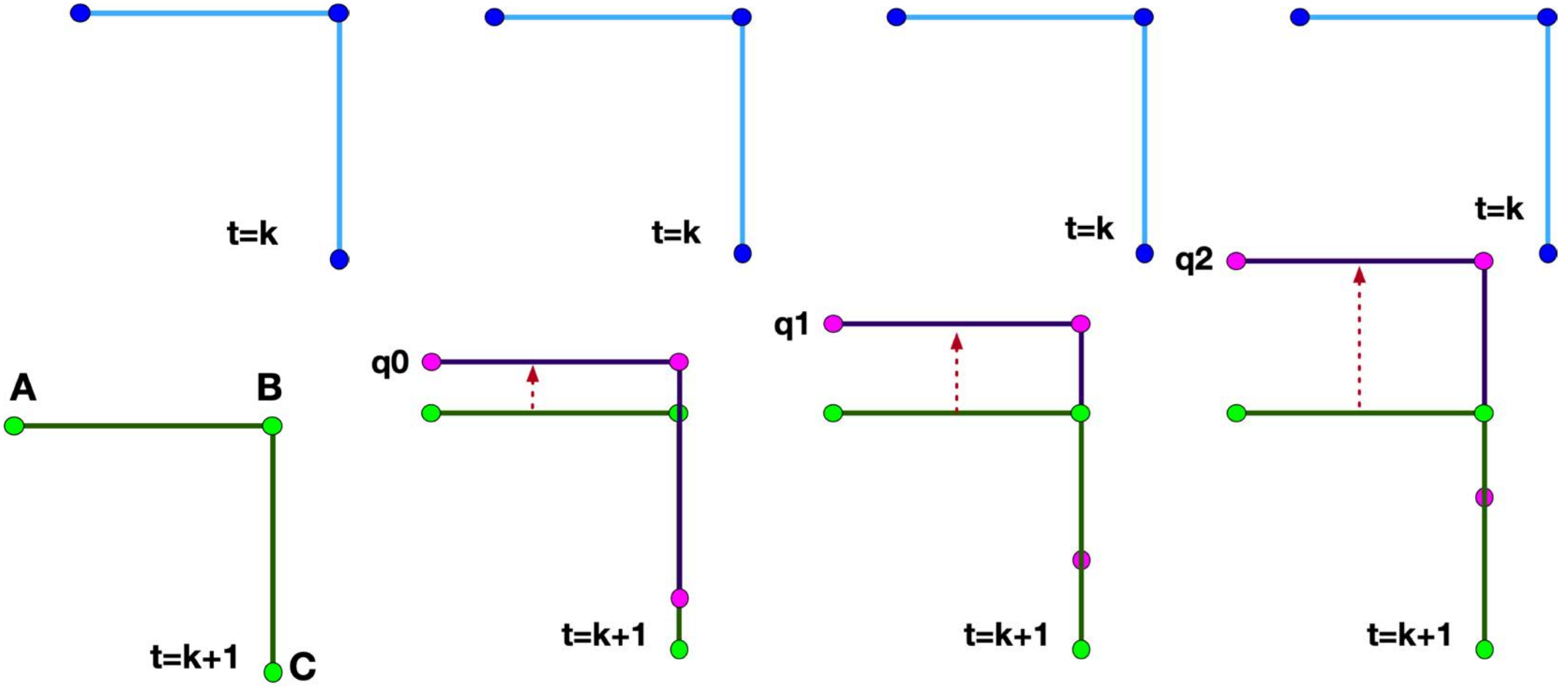
Why is it faster?



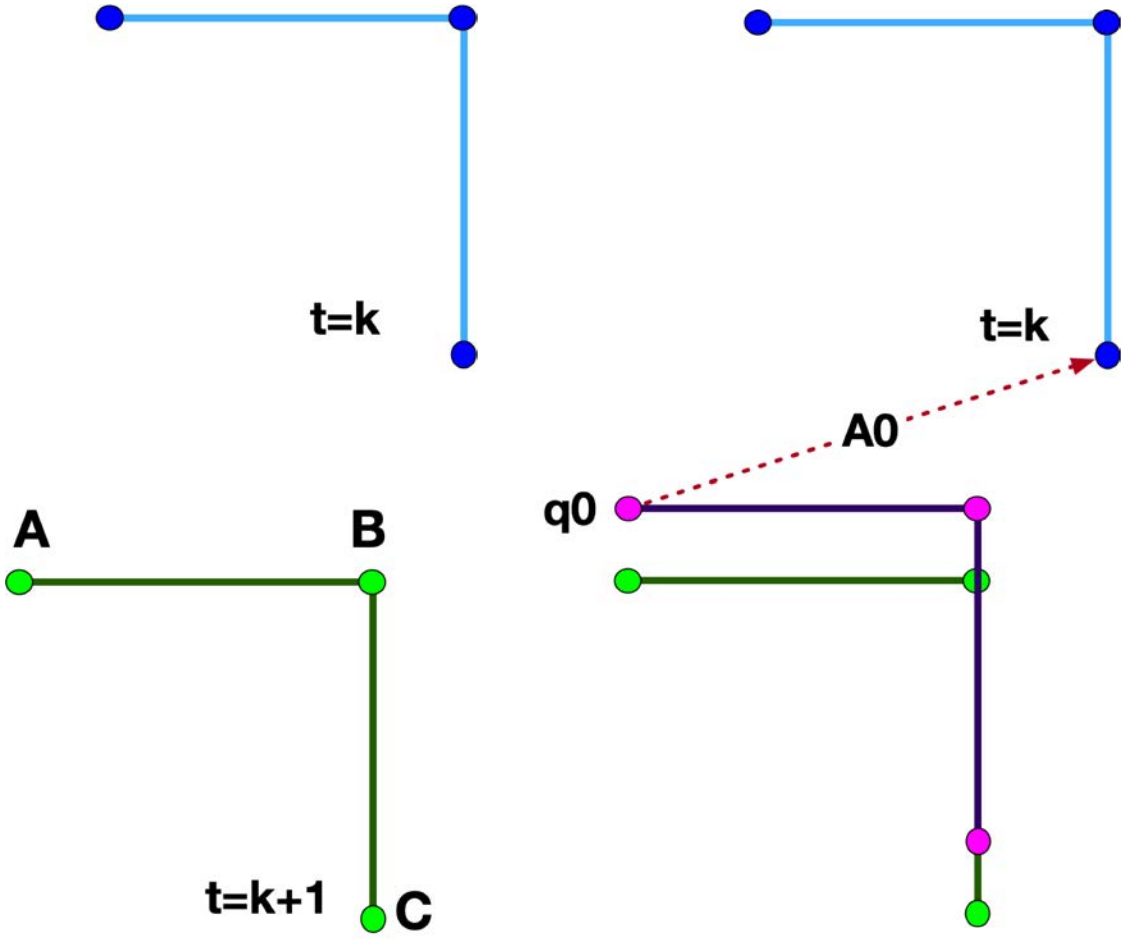
Why is it faster?



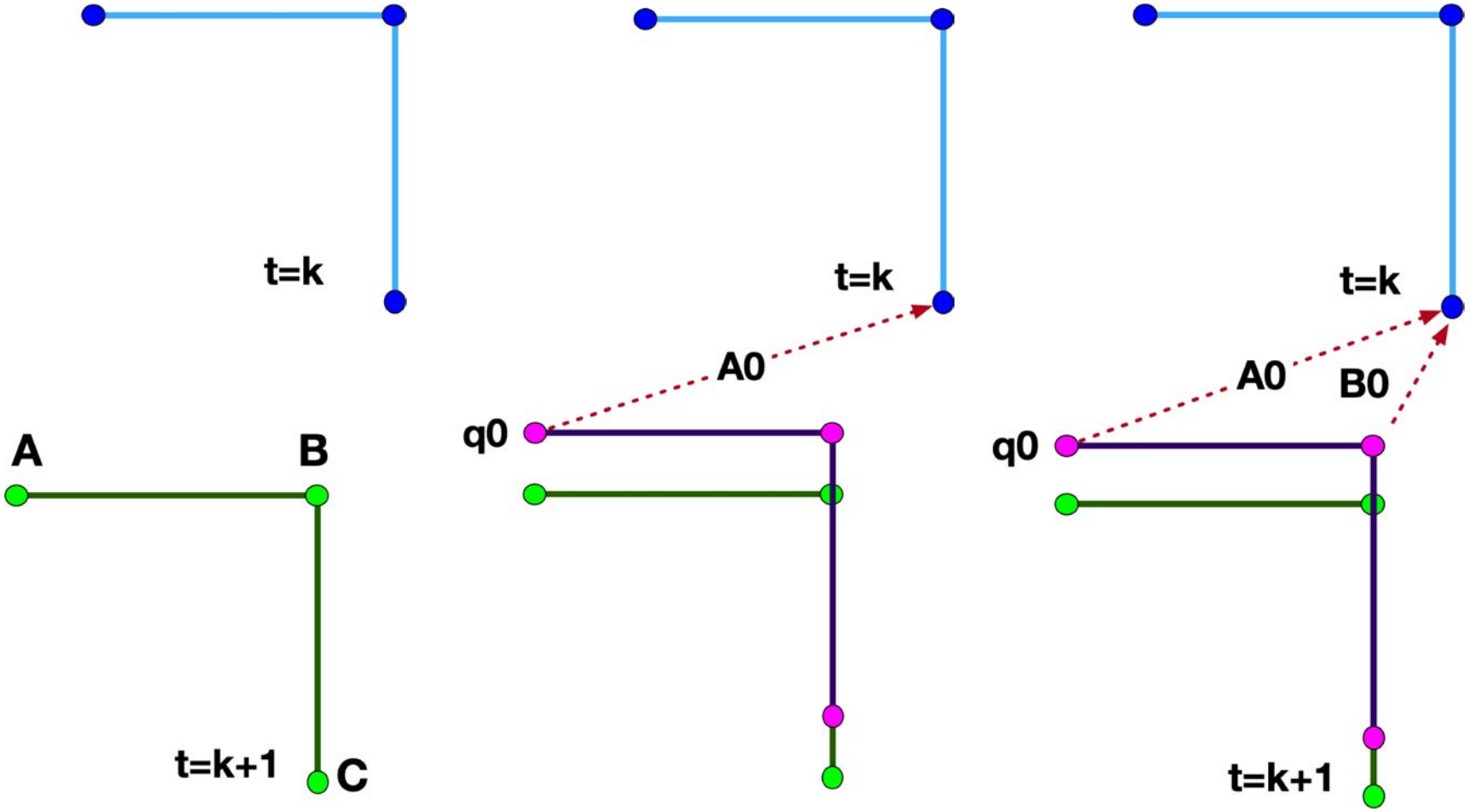
Why is it faster?



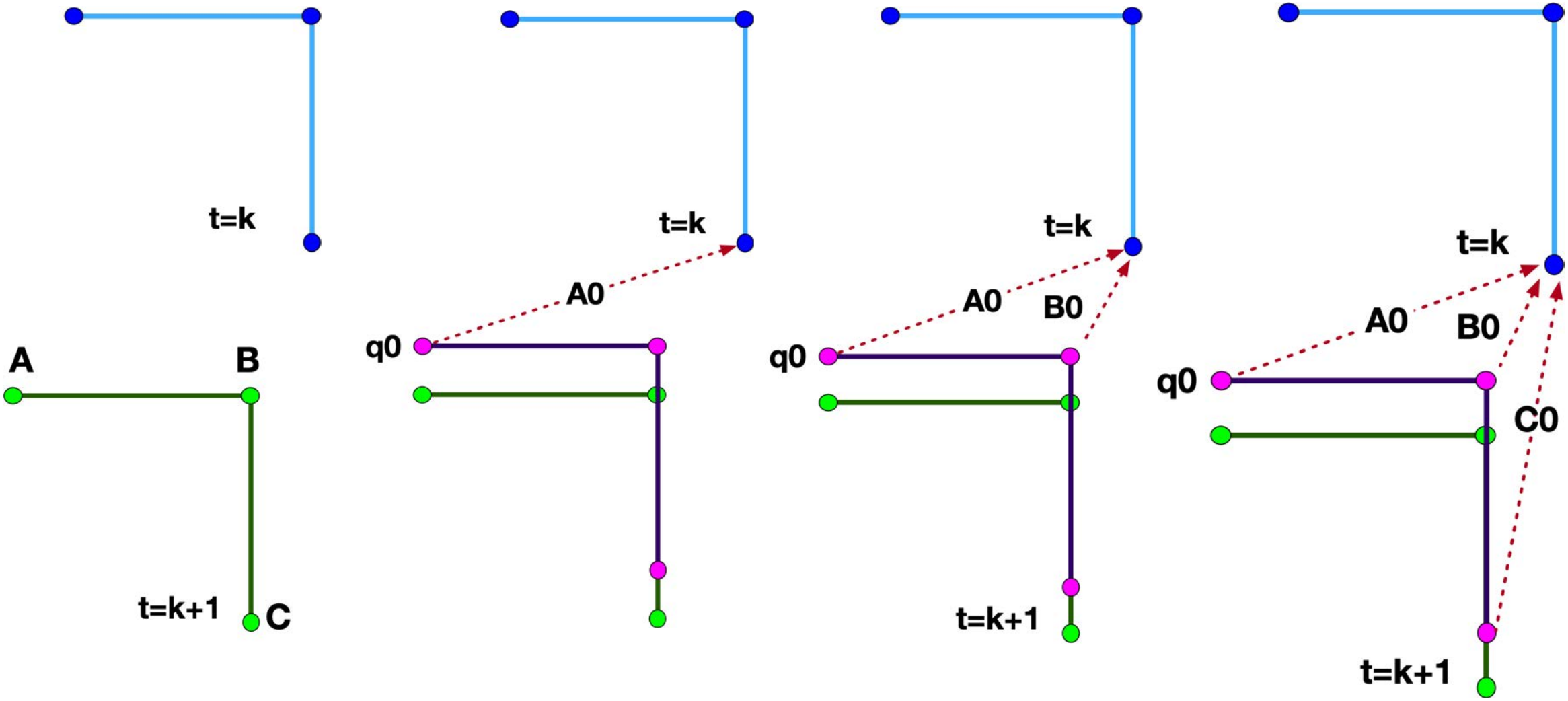
Point-to-point Metric



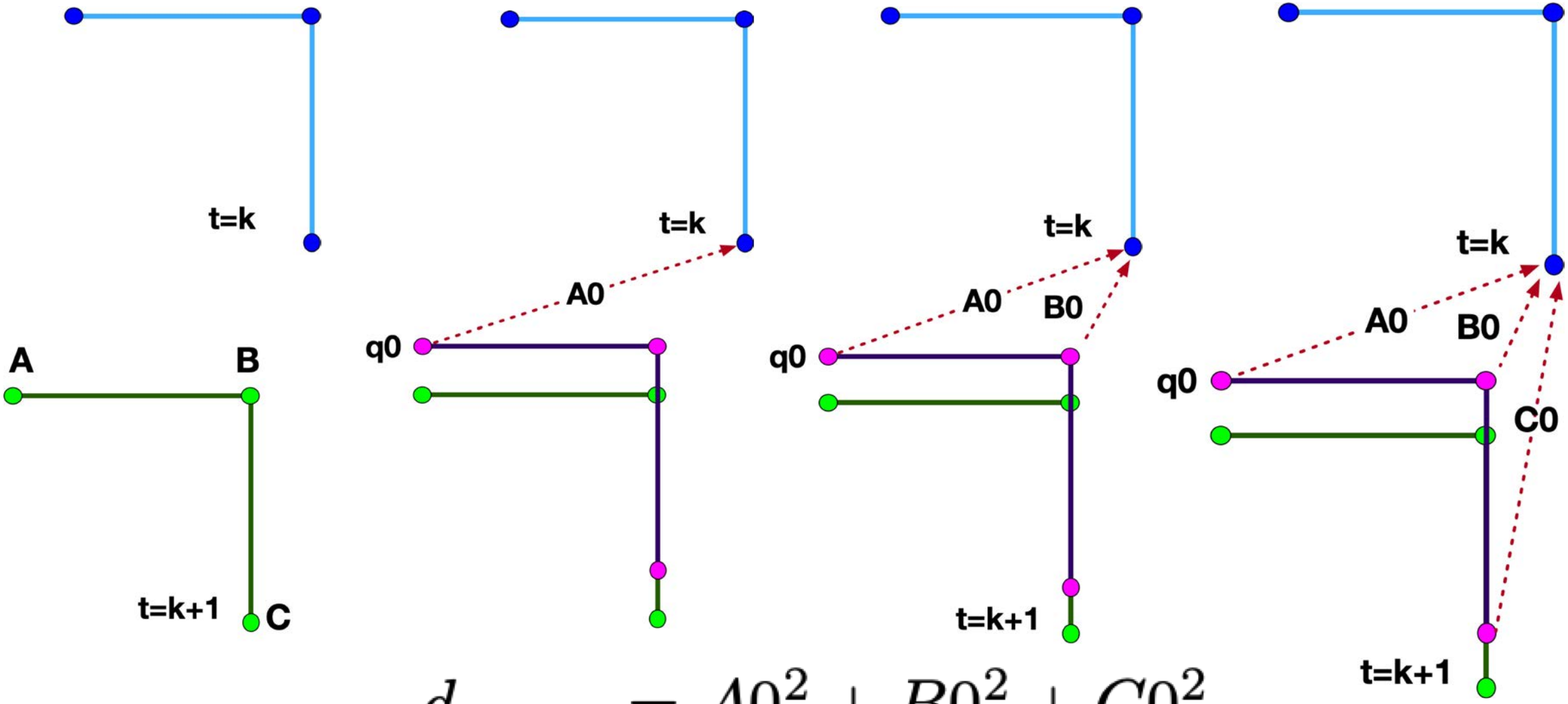
Point-to-point Metric



Point-to-point Metric

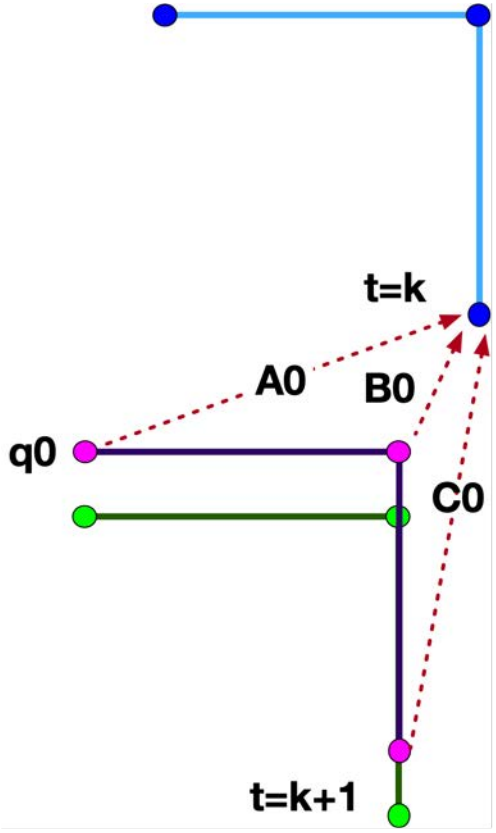


Point-to-point Metric



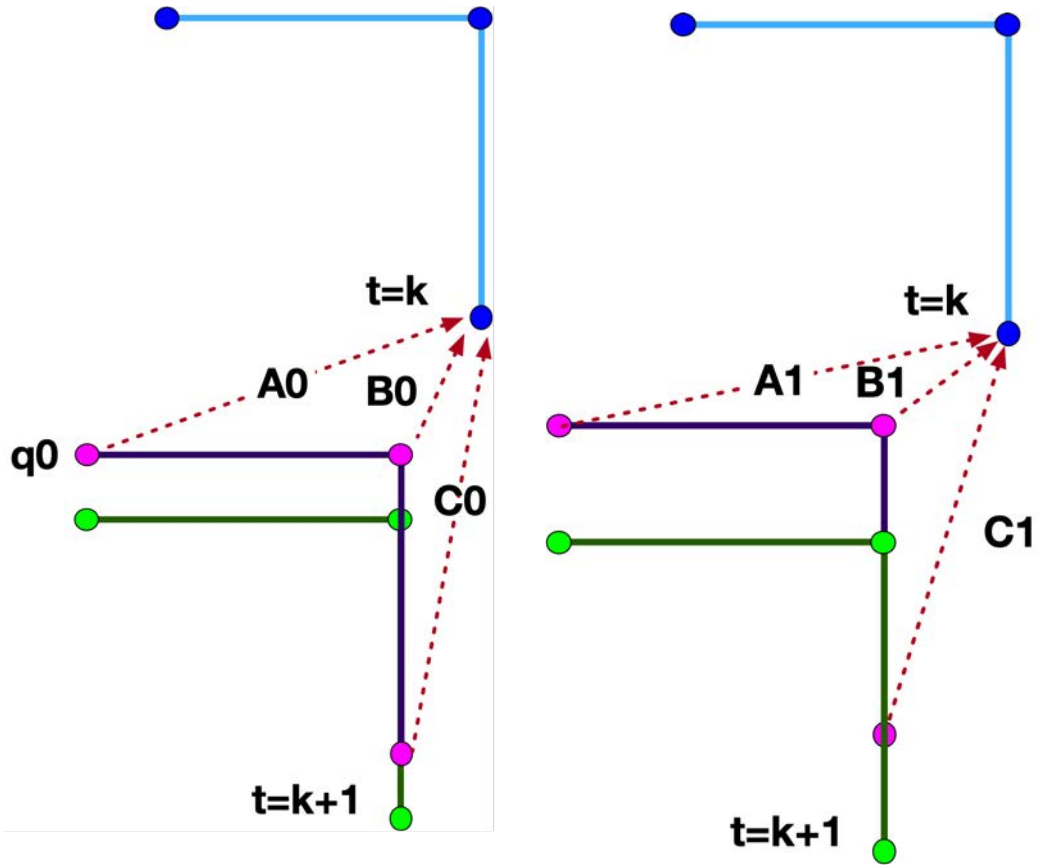
$$d_{p2p,q0} = A0^2 + B0^2 + C0^2$$

Point-to-point Metric



$$d_{p2p,q0} = A0^2 + B0^2 + C0^2$$

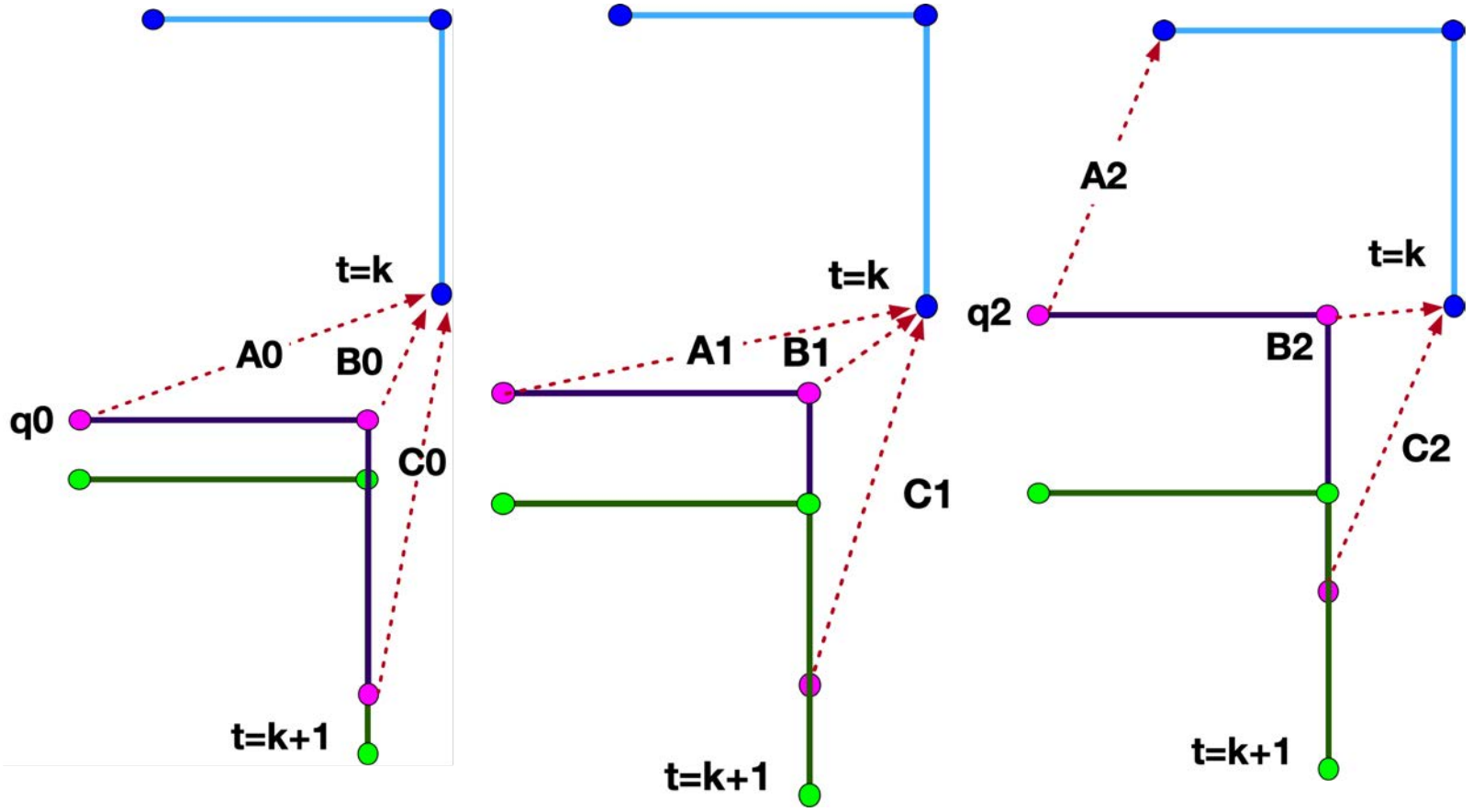
Point-to-point Metric



$$d_{p2p,q0} = A0^2 + B0^2 + C0^2$$

$$d_{p2p,q1} = A1^2 + B1^2 + C1^2$$

Point-to-point Metric

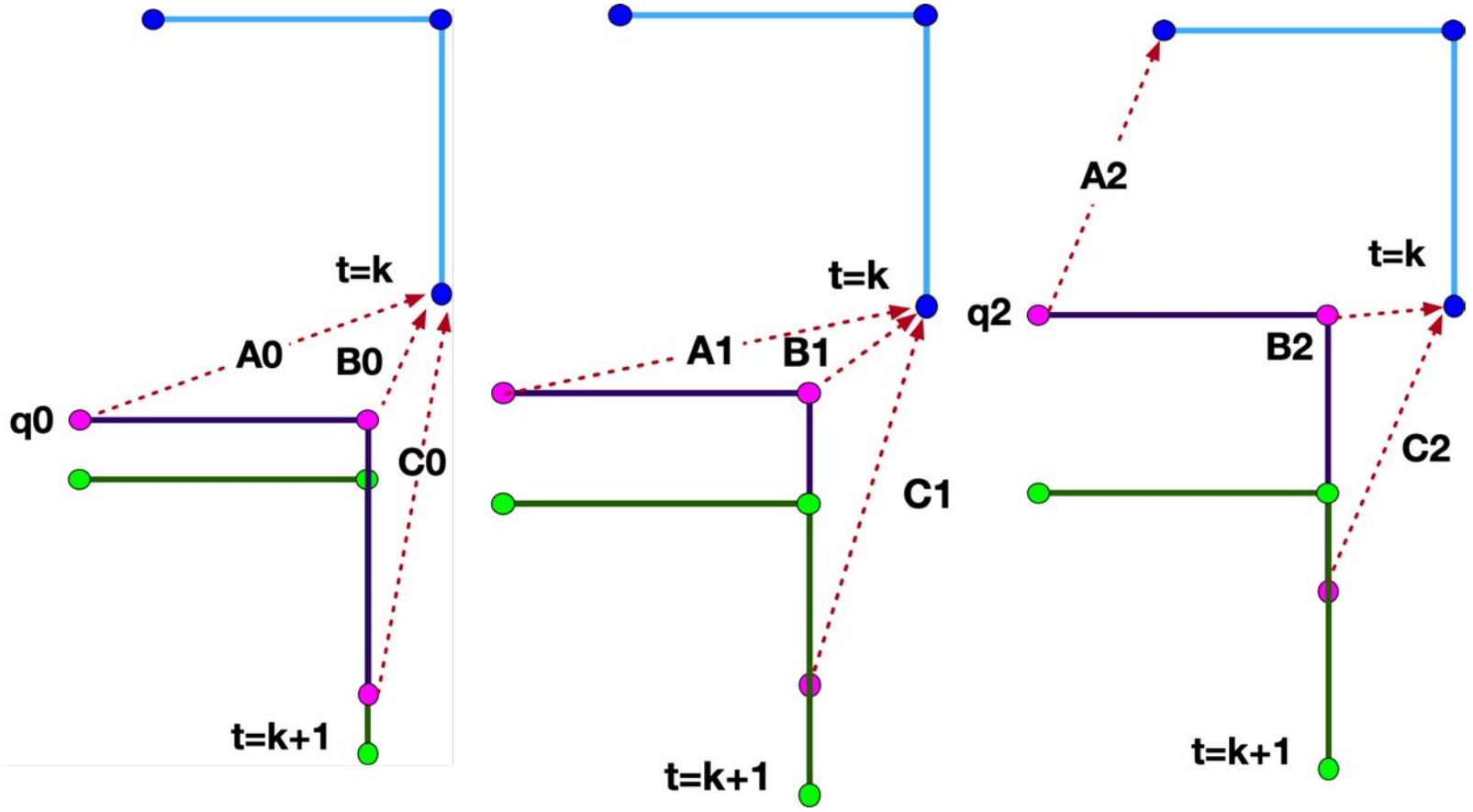


$$d_{p2p,q0} = A0^2 + B0^2 + C0^2$$

$$d_{p2p,q1} = A1^2 + B1^2 + C1^2$$

$$d_{p2p,q2} = A2^2 + B2^2 + C2^2$$

Point-to-point Metric



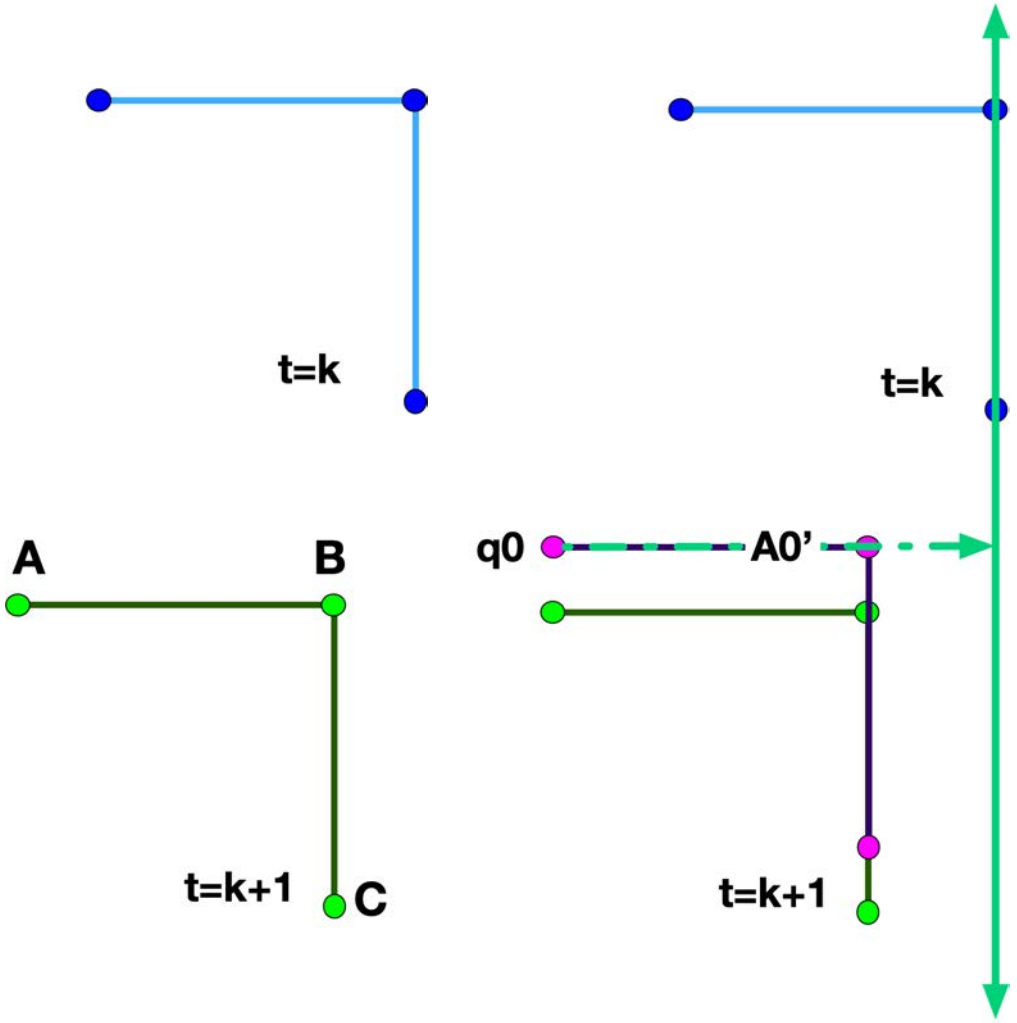
$$d_{p2p,q0} = A0^2 + B0^2 + C0^2$$

$$d_{p2p,q1} = A1^2 + B1^2 + C1^2$$

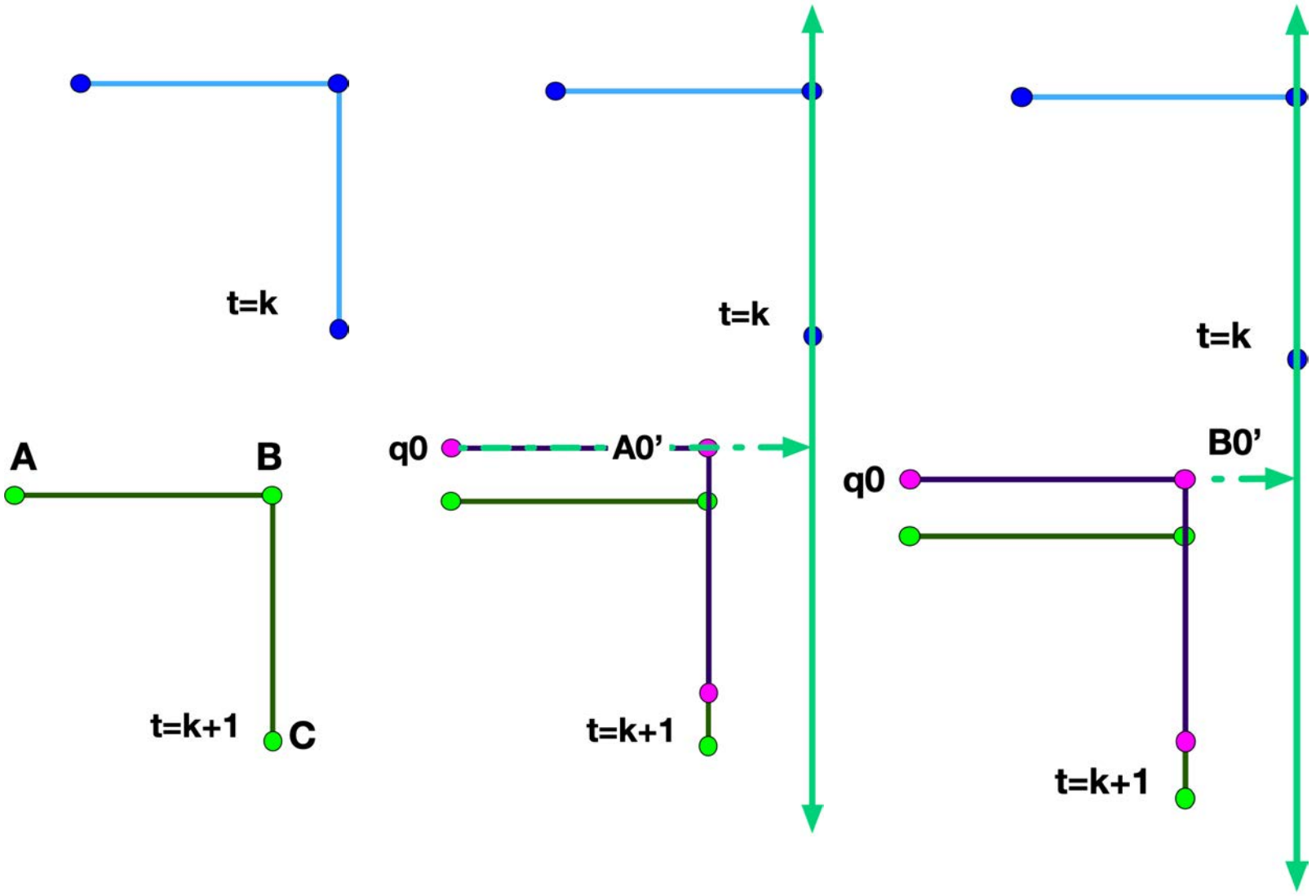
$$d_{p2p,q2} = A2^2 + B2^2 + C2^2$$

$$d_{p2p,q0} \neq d_{p2p,q1} \neq d_{p2p,q2}$$

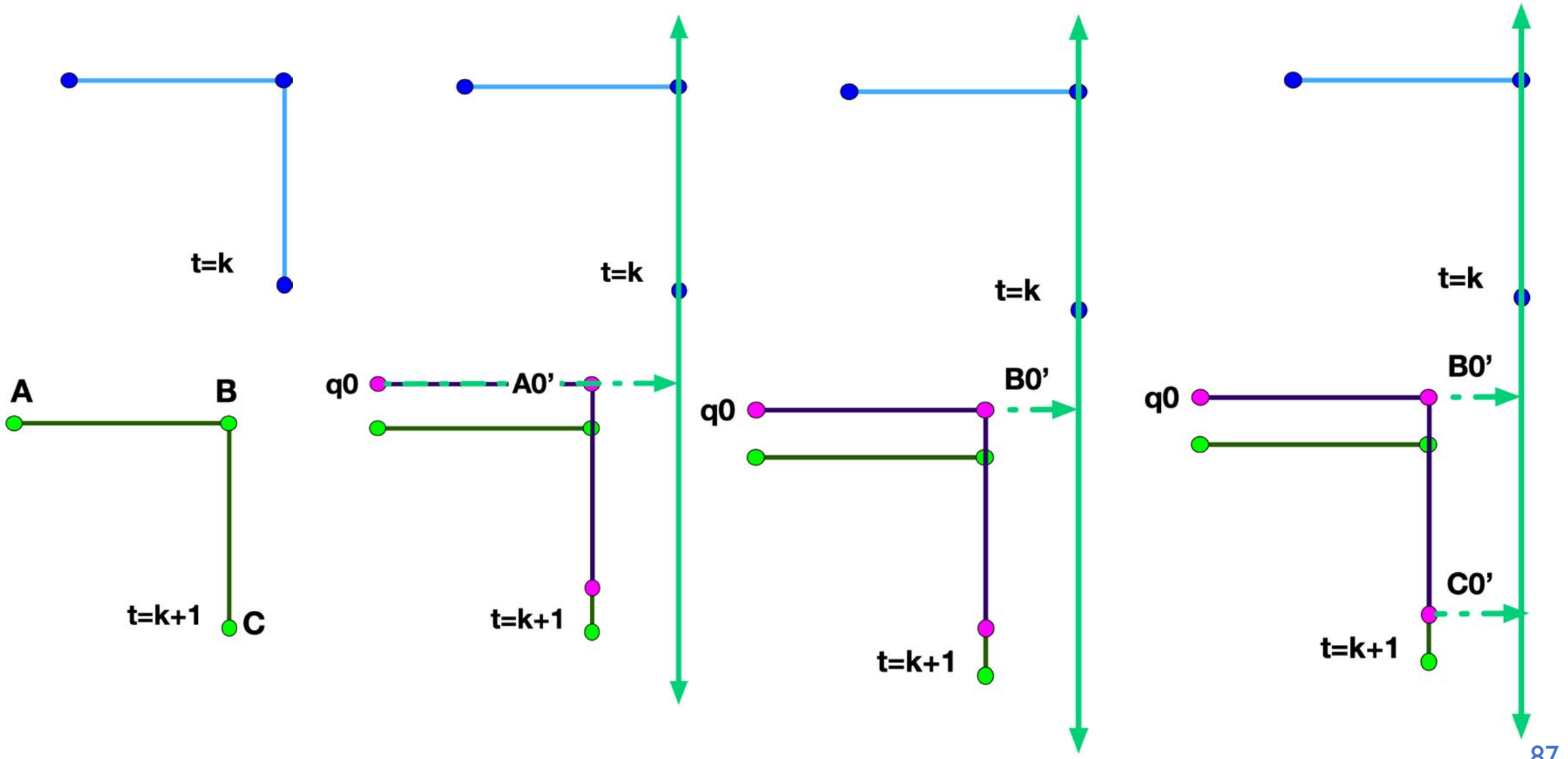
Point-to-line Metric



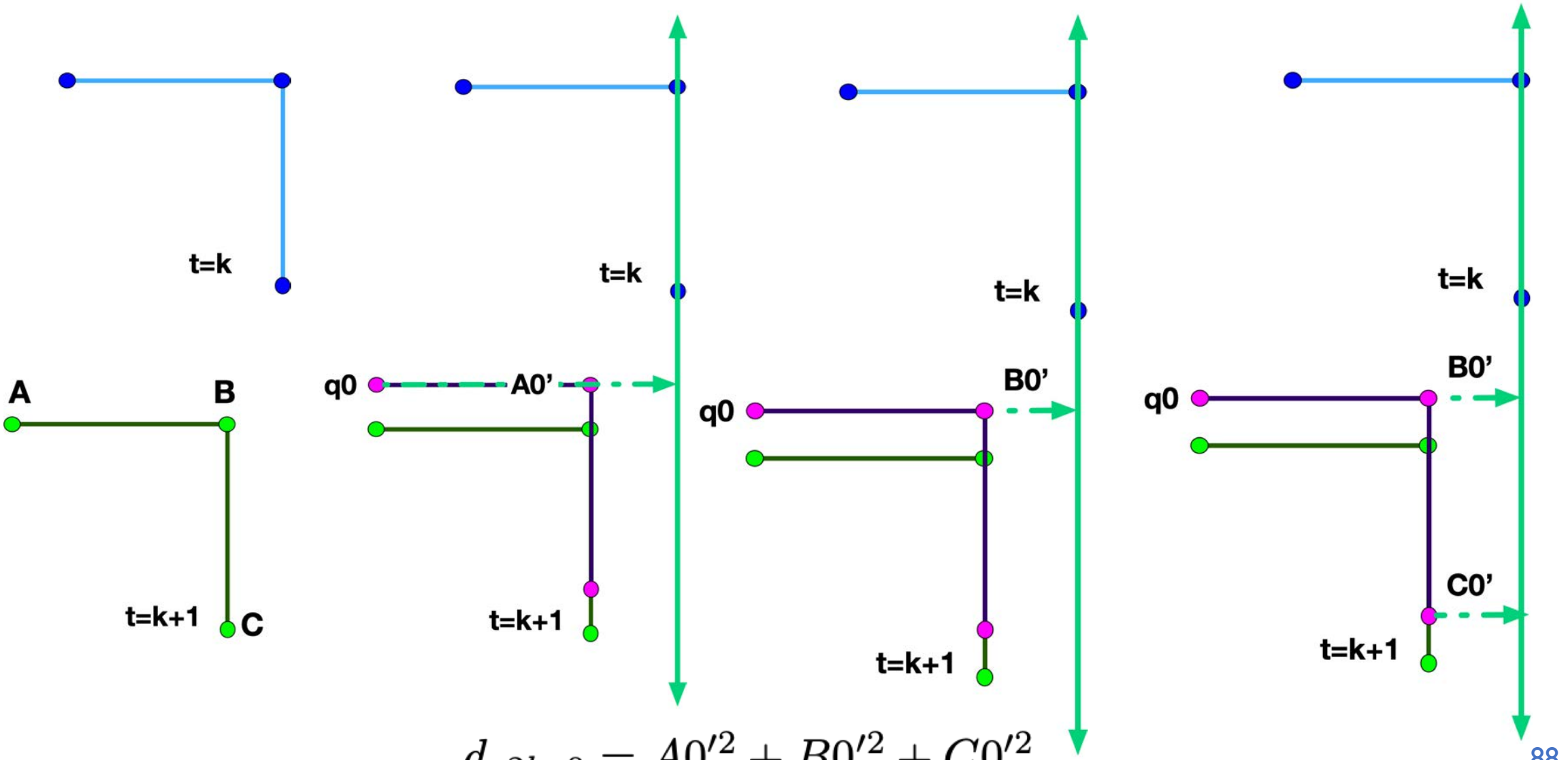
Point-to-line Metric



Point-to-line Metric

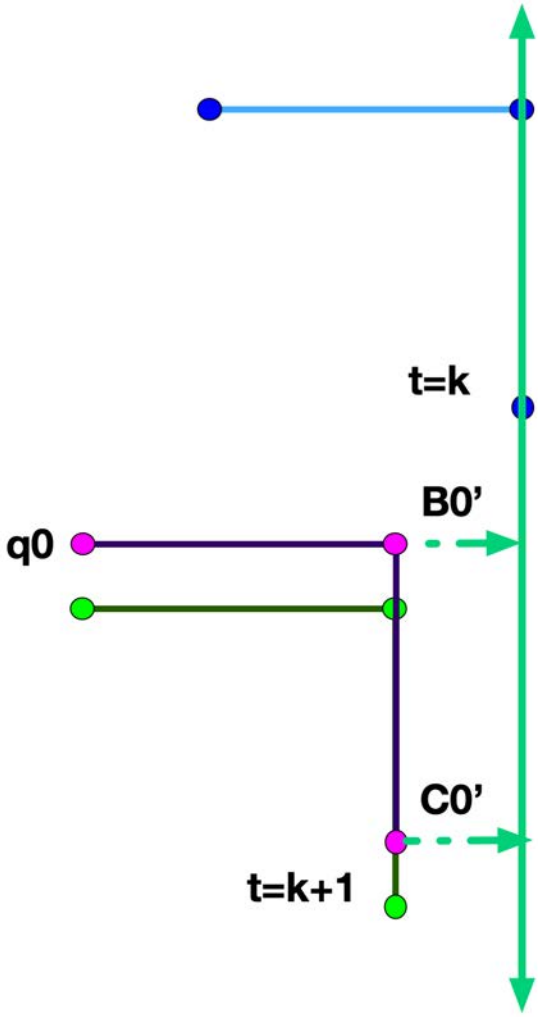


Point-to-line Metric



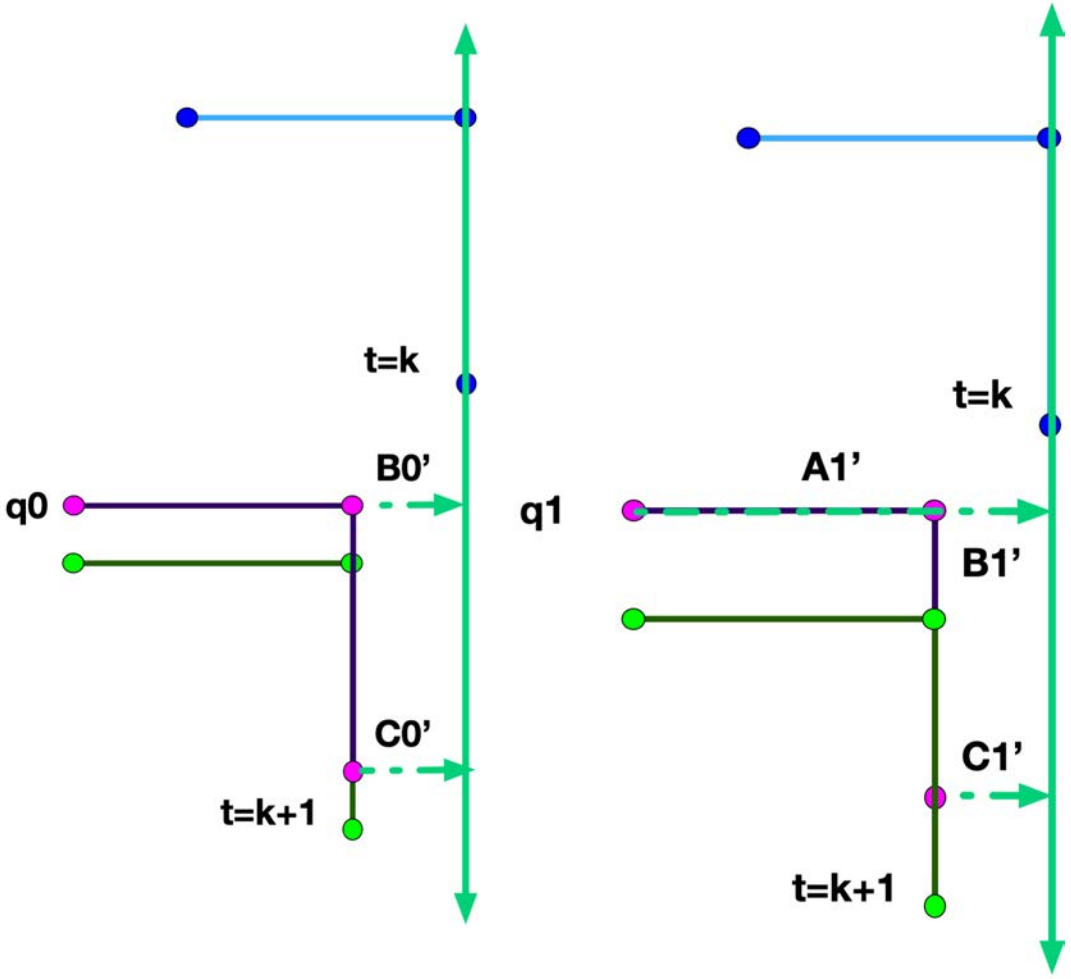
$$d_{p2l,q0} = A0'^2 + B0'^2 + C0'^2$$

Point-to-line Metric



$$d_{p2l,q0} = A0'^2 + B0'^2 + C0'^2$$

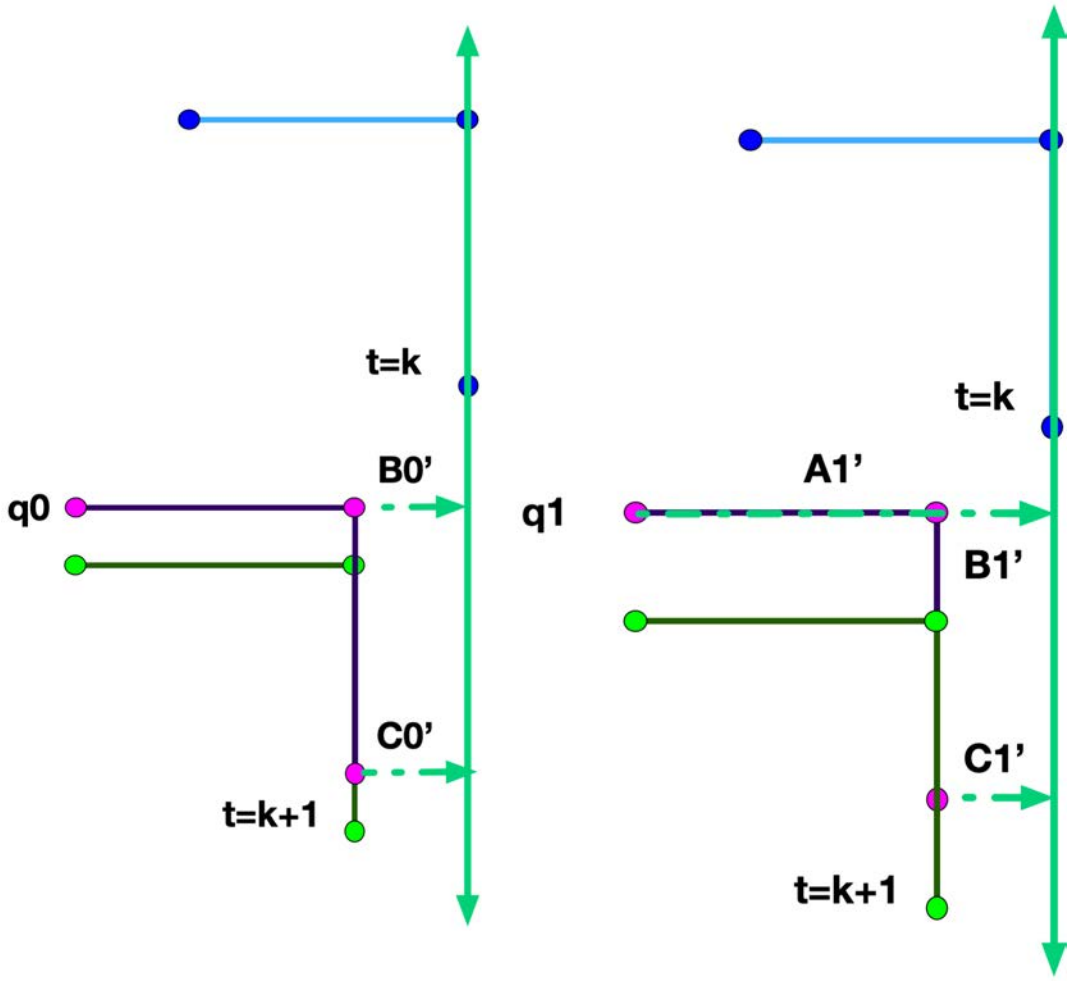
Point-to-line Metric



$$d_{p2l,q0} = A0'^2 + B0'^2 + C0'^2$$

$$d_{p2l,q1} = A1'^2 + B1'^2 + C1'^2$$

Point-to-line Metric

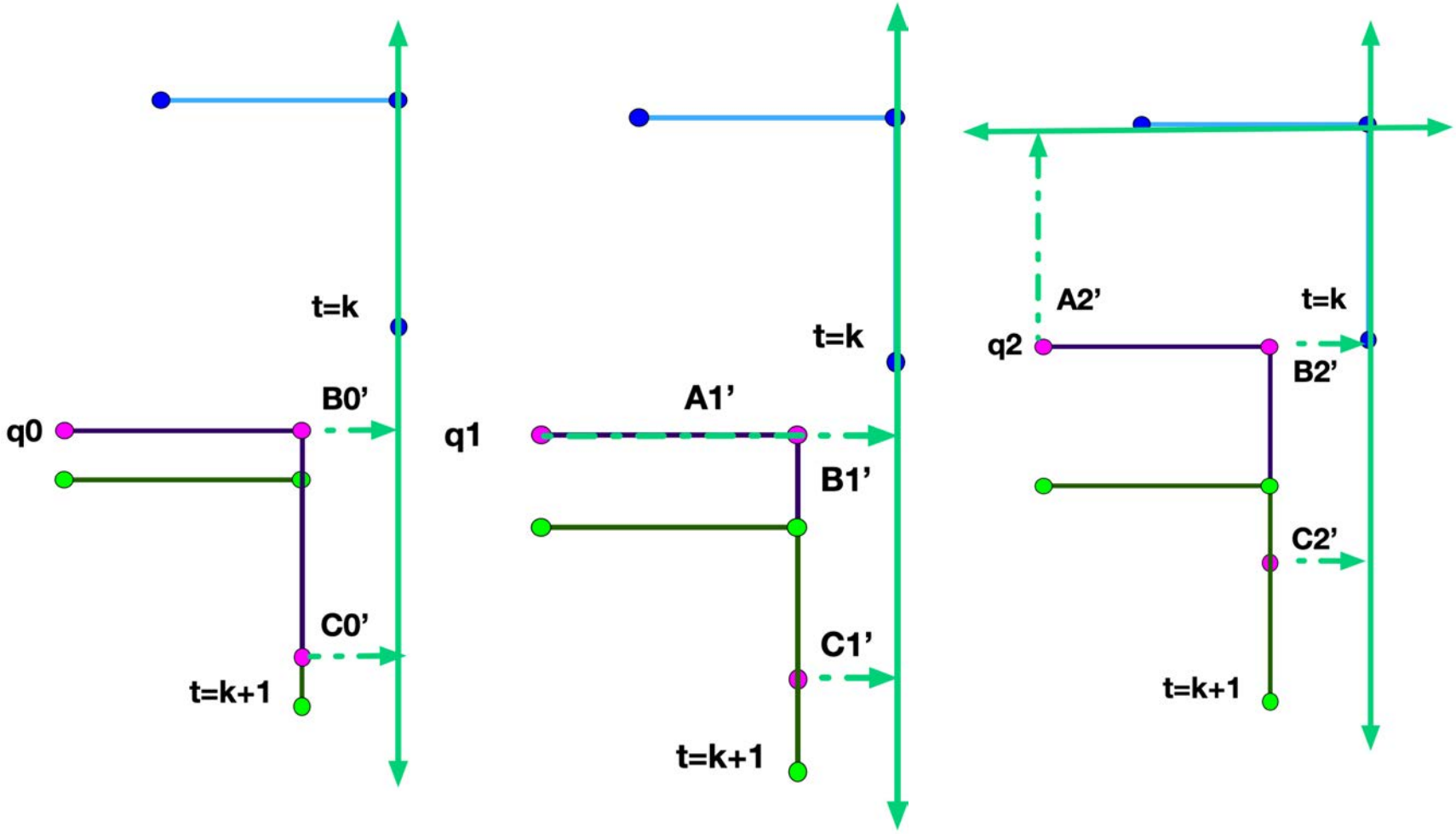


$$d_{p2l,q0} = A0'^2 + B0'^2 + C0'^2$$

$$d_{p2l,q1} = A1'^2 + B1'^2 + C1'^2$$

$$d_{p2p,q0} = d_{p2p,q1}$$

Point-to-line Metric

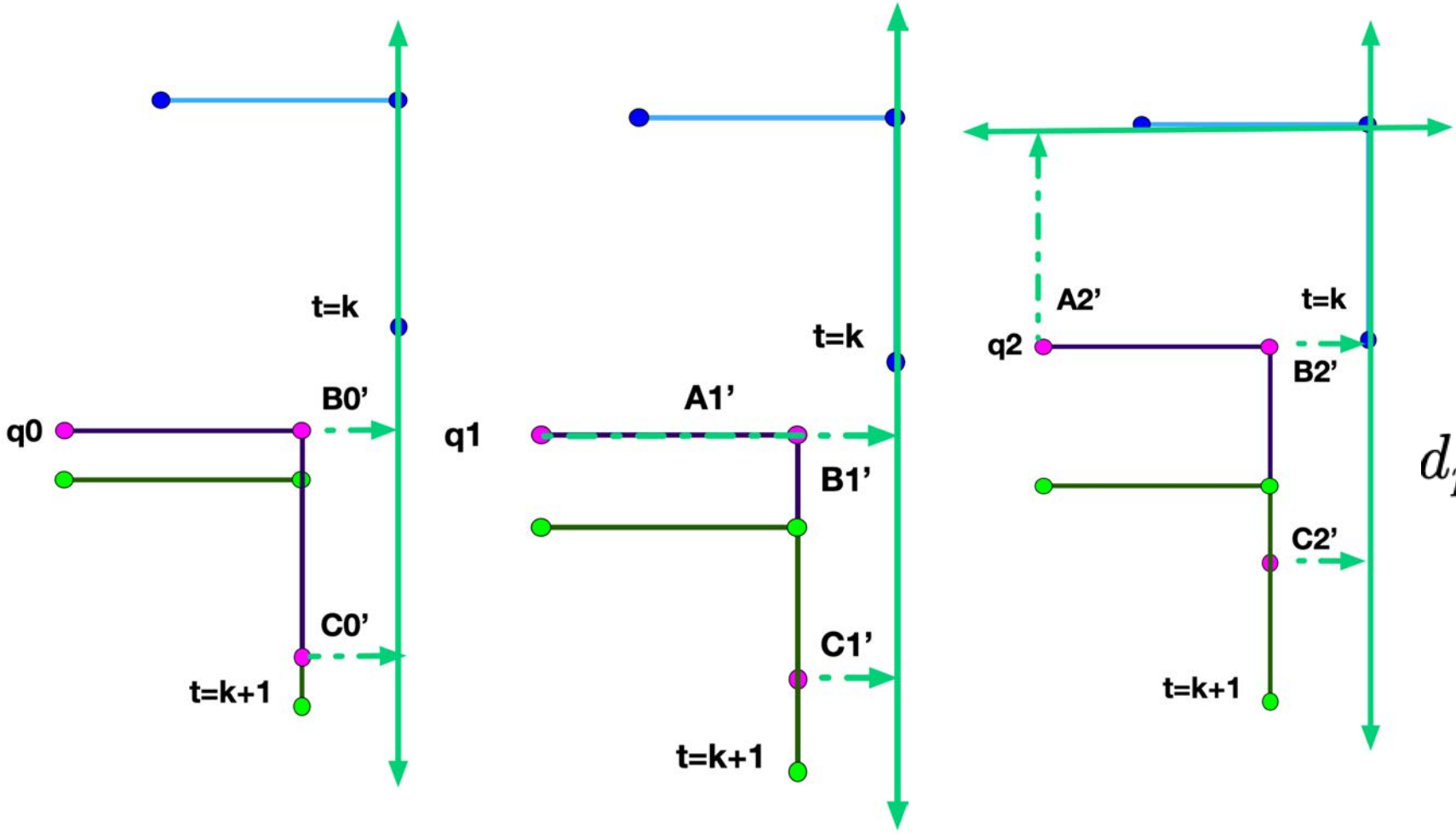


$$d_{p2l,q0} = A0'^2 + B0'^2 + C0'^2$$

$$d_{p2l,q1} = A1'^2 + B1'^2 + C1'^2$$

$$d_{p2l,q2} = A2'^2 + B2'^2 + C2'^2$$

Point-to-line Metric



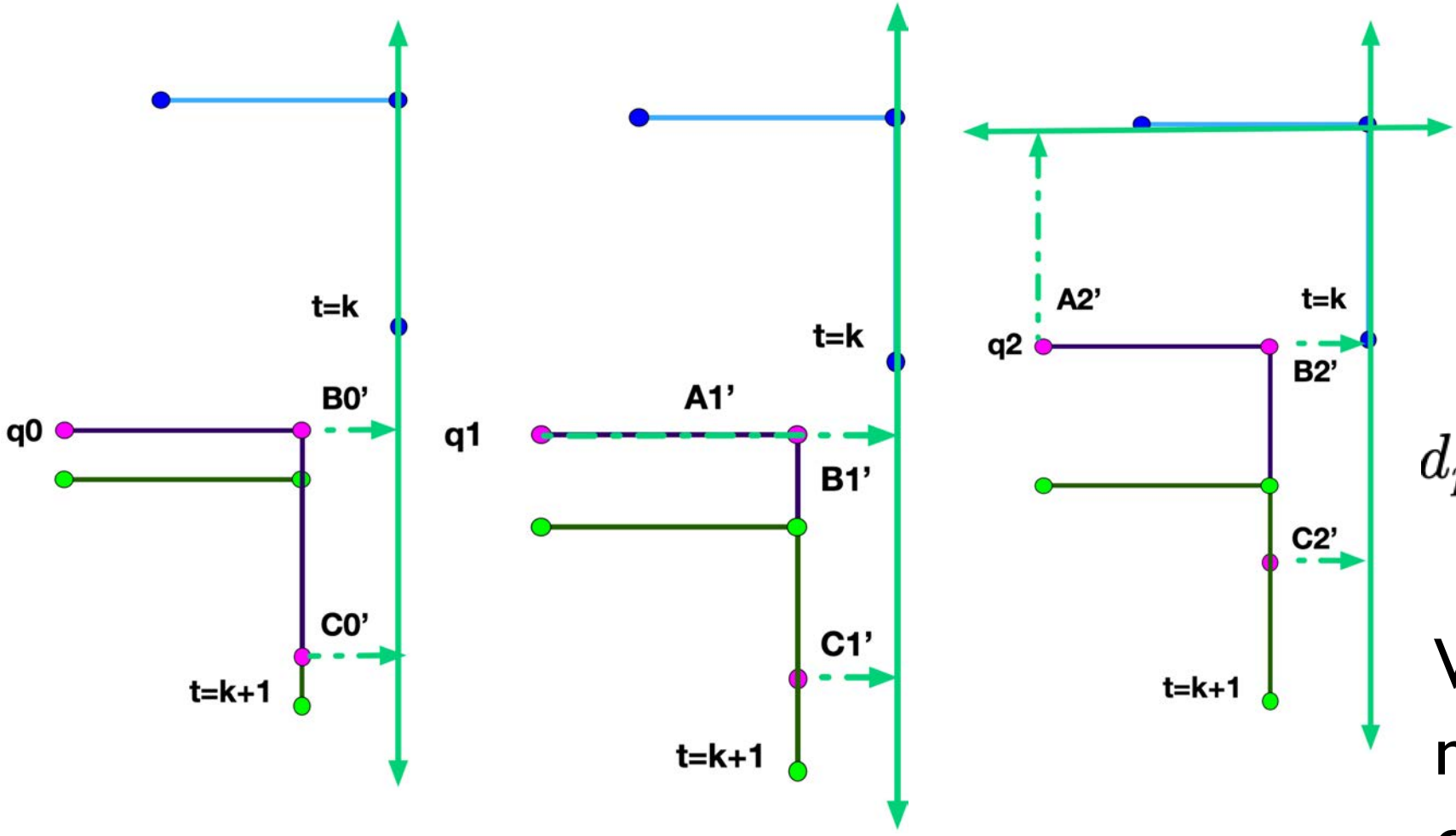
$$d_{p2l,q0} = A0'^2 + B0'^2 + C0'^2$$

$$d_{p2l,q1} = A1'^2 + B1'^2 + C1'^2$$

$$d_{p2l,q2} = A2'^2 + B2'^2 + C2'^2$$

$$d_{p2p,q0} = d_{p2p,q1} \neq d_{p2p,q2}$$

Point-to-line Metric



$$d_{p2l,q0} = A0'^2 + B0'^2 + C0'^2$$

$$d_{p2l,q1} = A1'^2 + B1'^2 + C1'^2$$

$$d_{p2l,q2} = A2'^2 + B2'^2 + C2'^2$$

$$d_{p2p,q0} = d_{p2p,q1} \neq d_{p2p,q2}$$

With point-to-line, next iteration will go directly from q0→ q2

Live Demo: /laserscan & Scan Matching