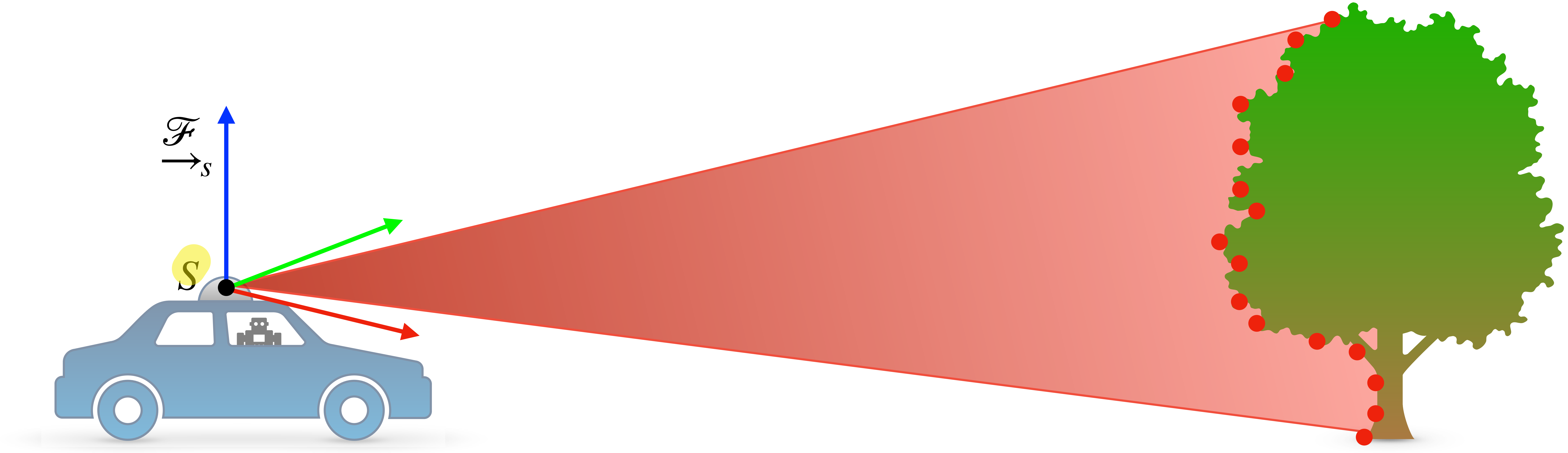
MODULE 4 LESSON 3

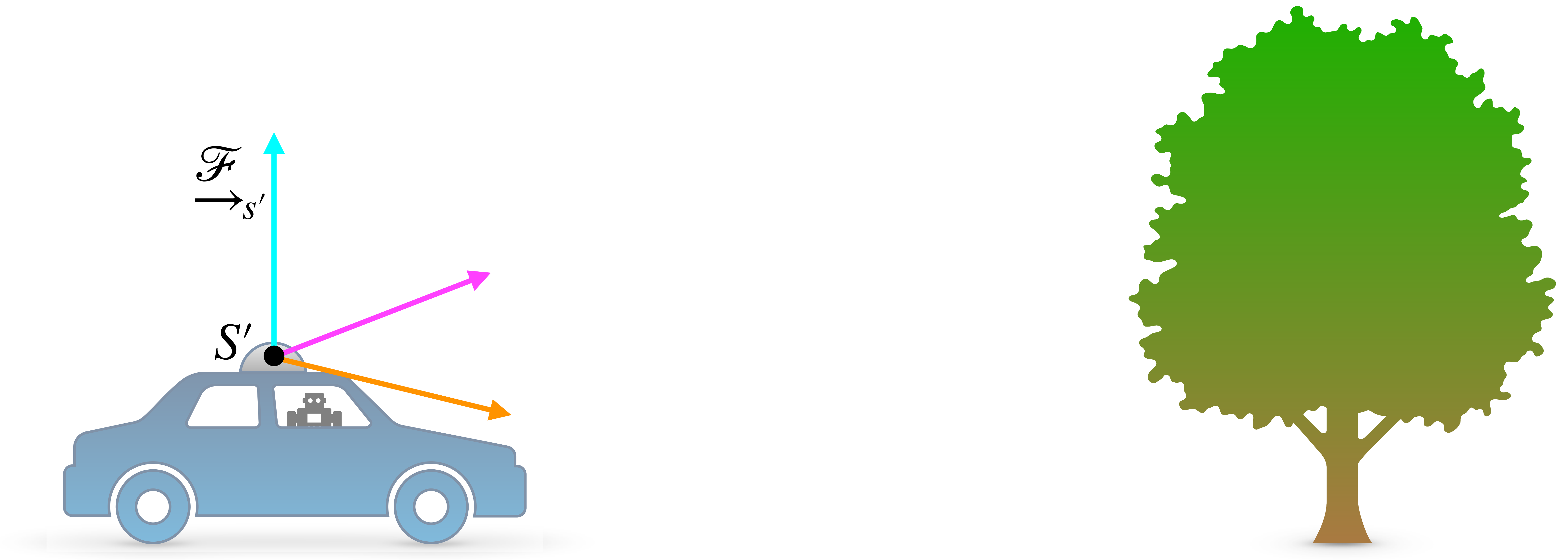
POSE ESTIMATION FROM LIDAR DATA

State Estimation via Point Set Registration

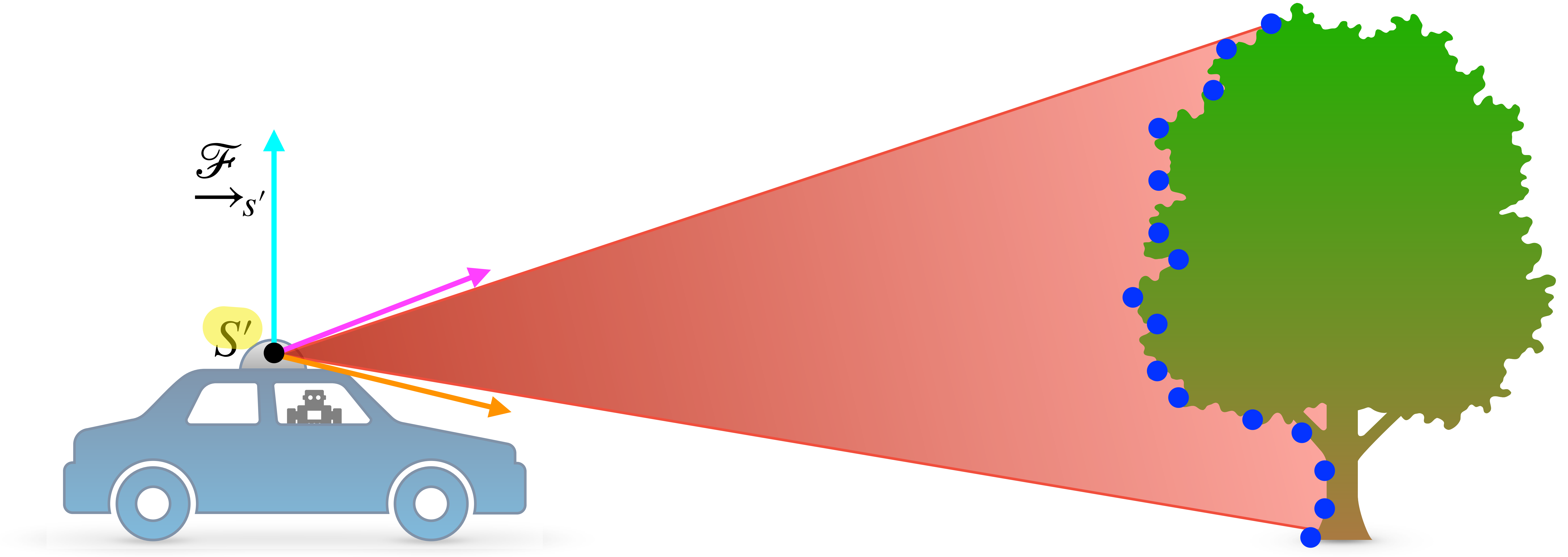
State Estimation via Point Set Registration



State Estimation via Point Set Registration

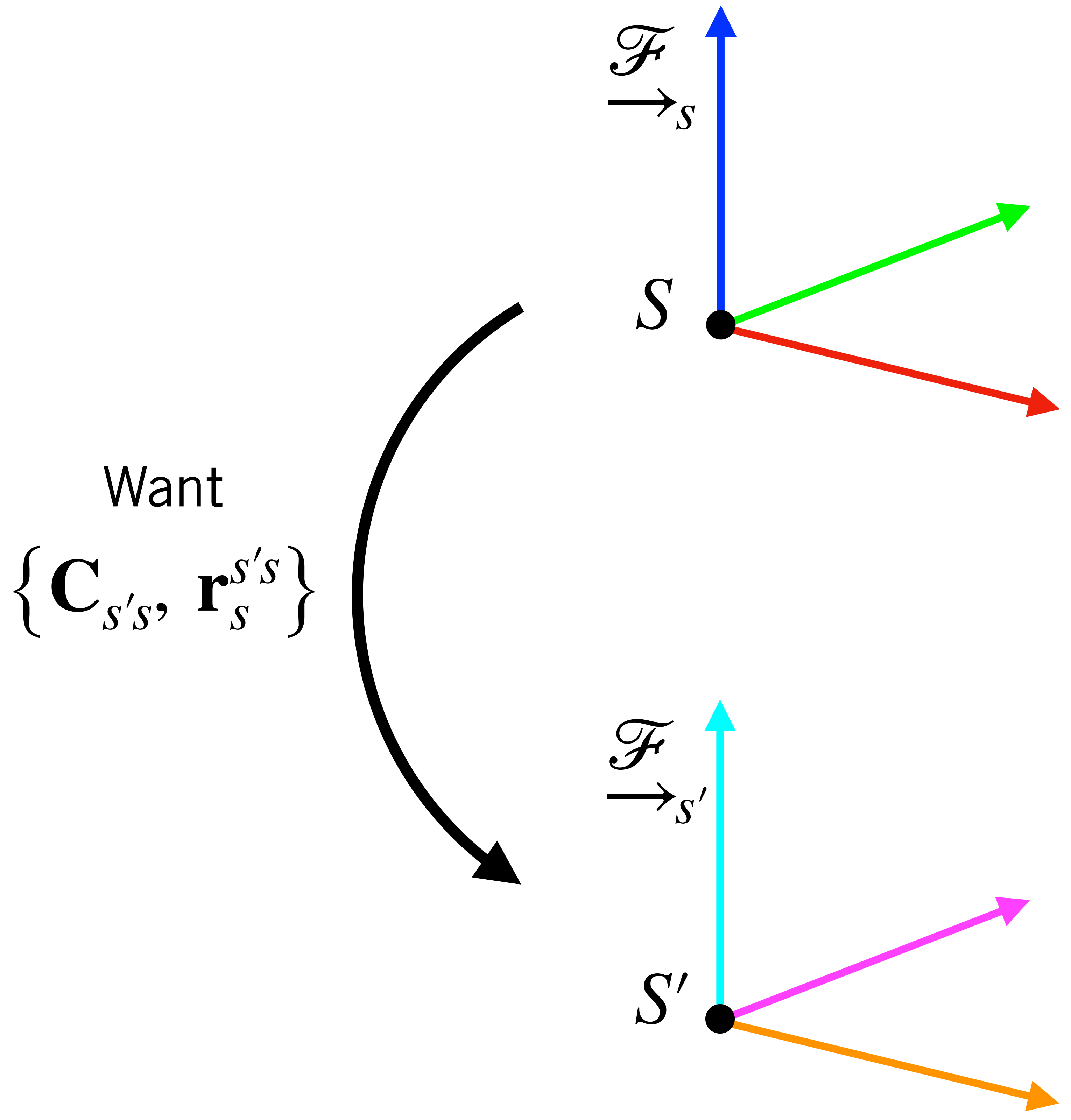


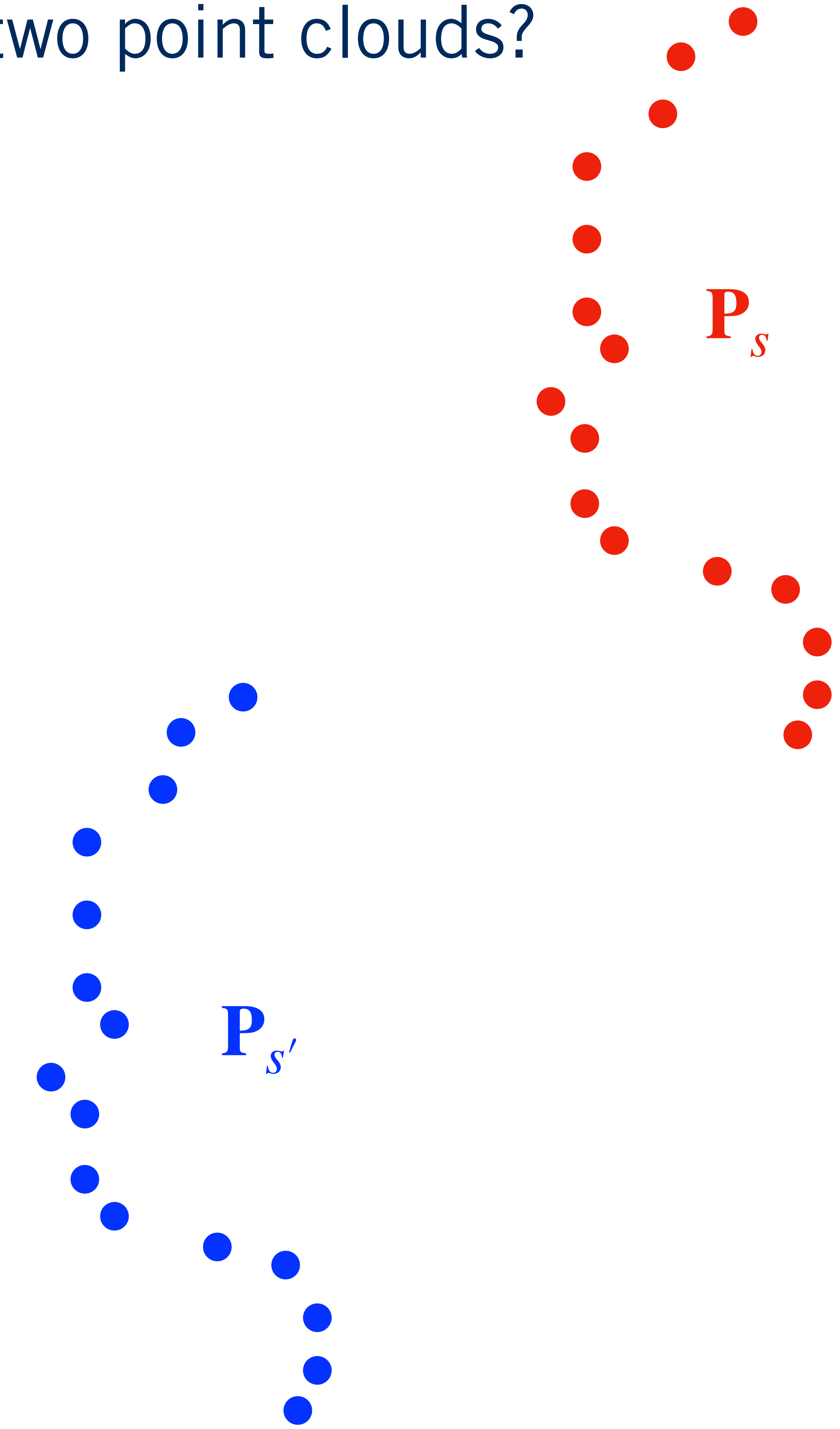
State Estimation via Point Set Registration



State Estimation via Point Set Registration

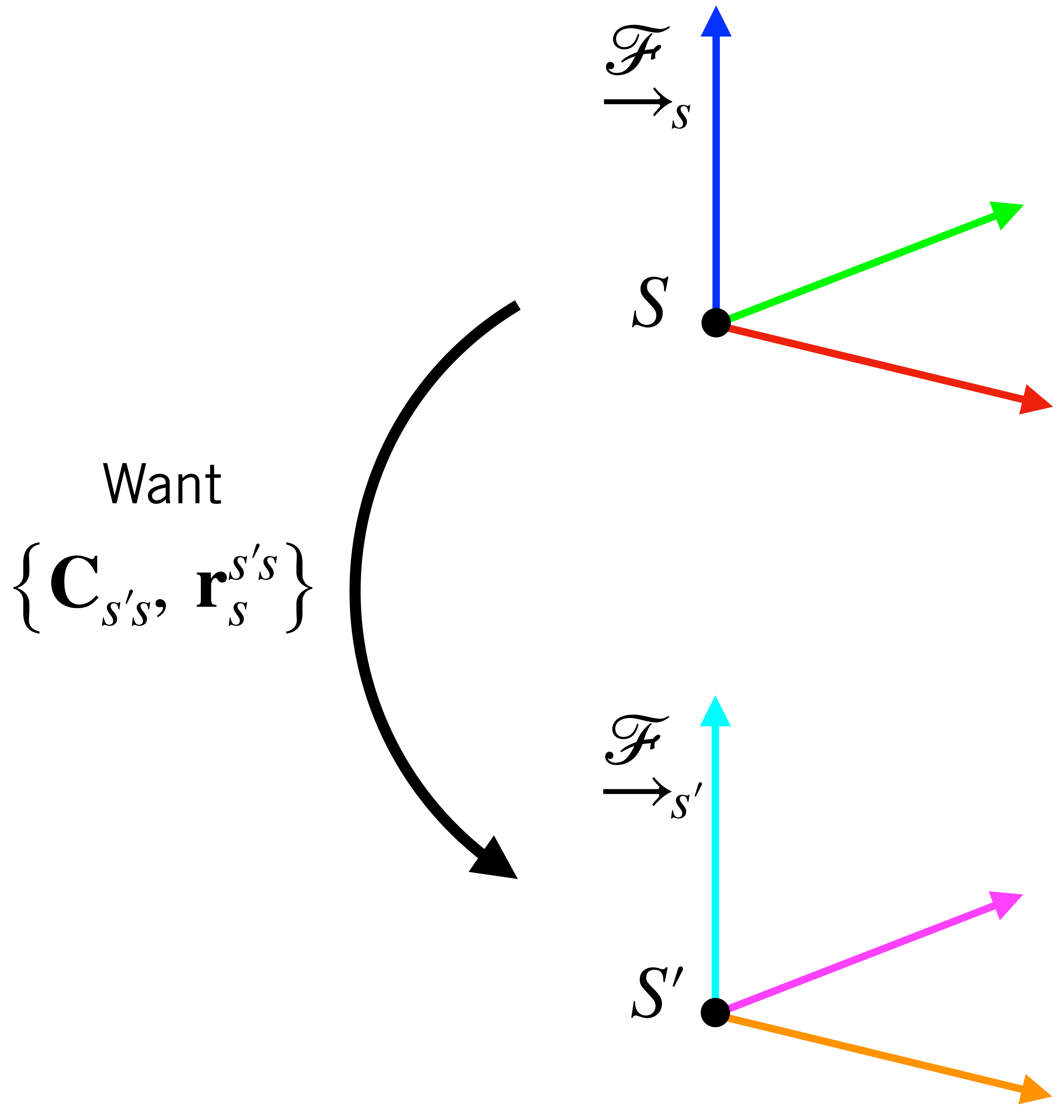
What motion of the car best aligns the two point clouds?

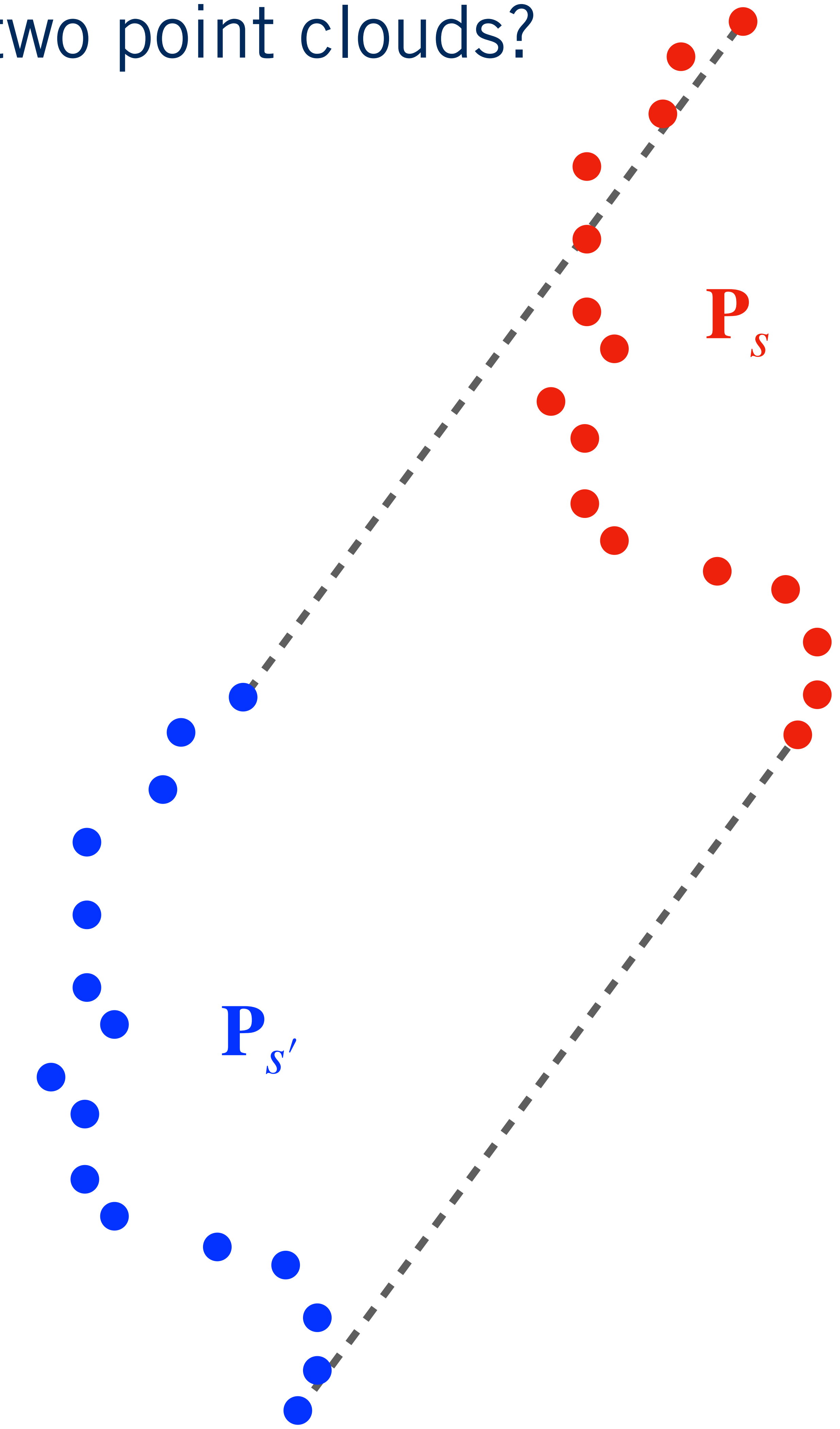




State Estimation via Point Set Registration

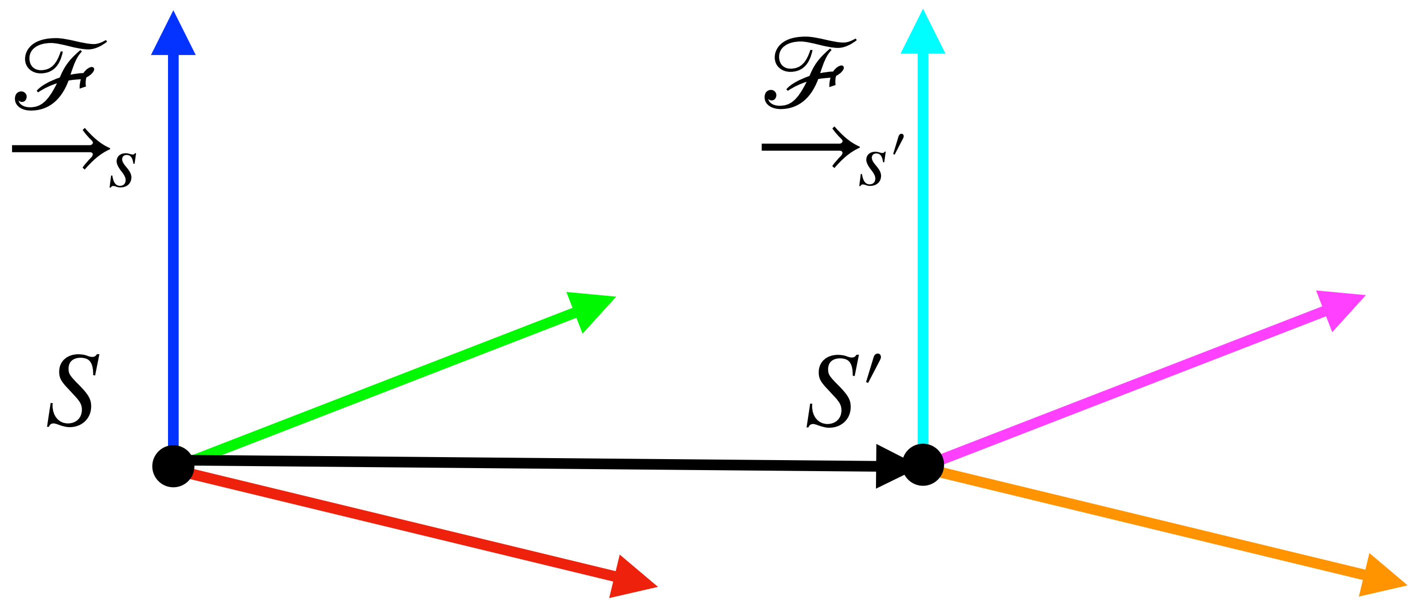
What motion of the car best aligns the two point clouds?



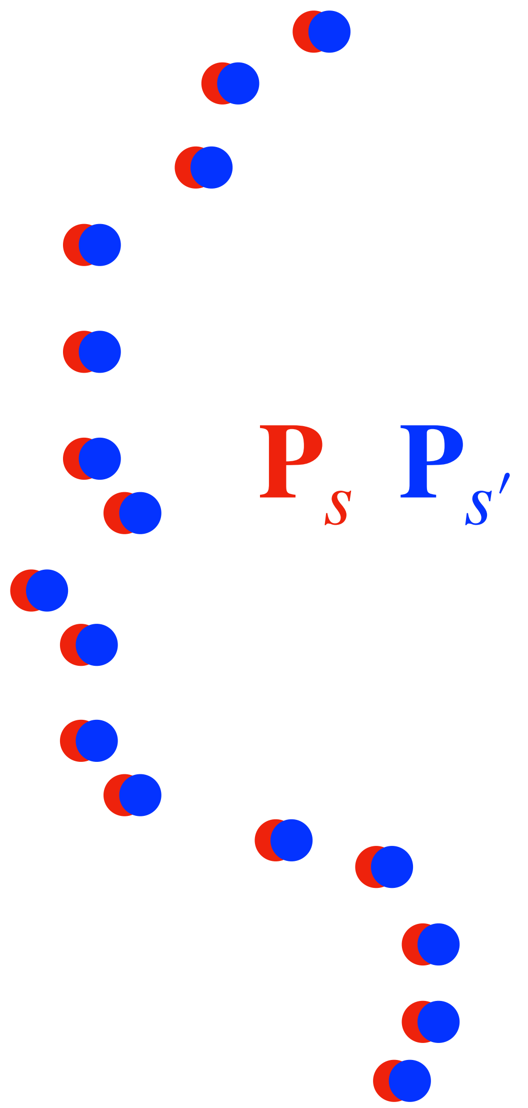


State Estimation via Point Set Registration

What motion of the car best aligns the two point clouds?

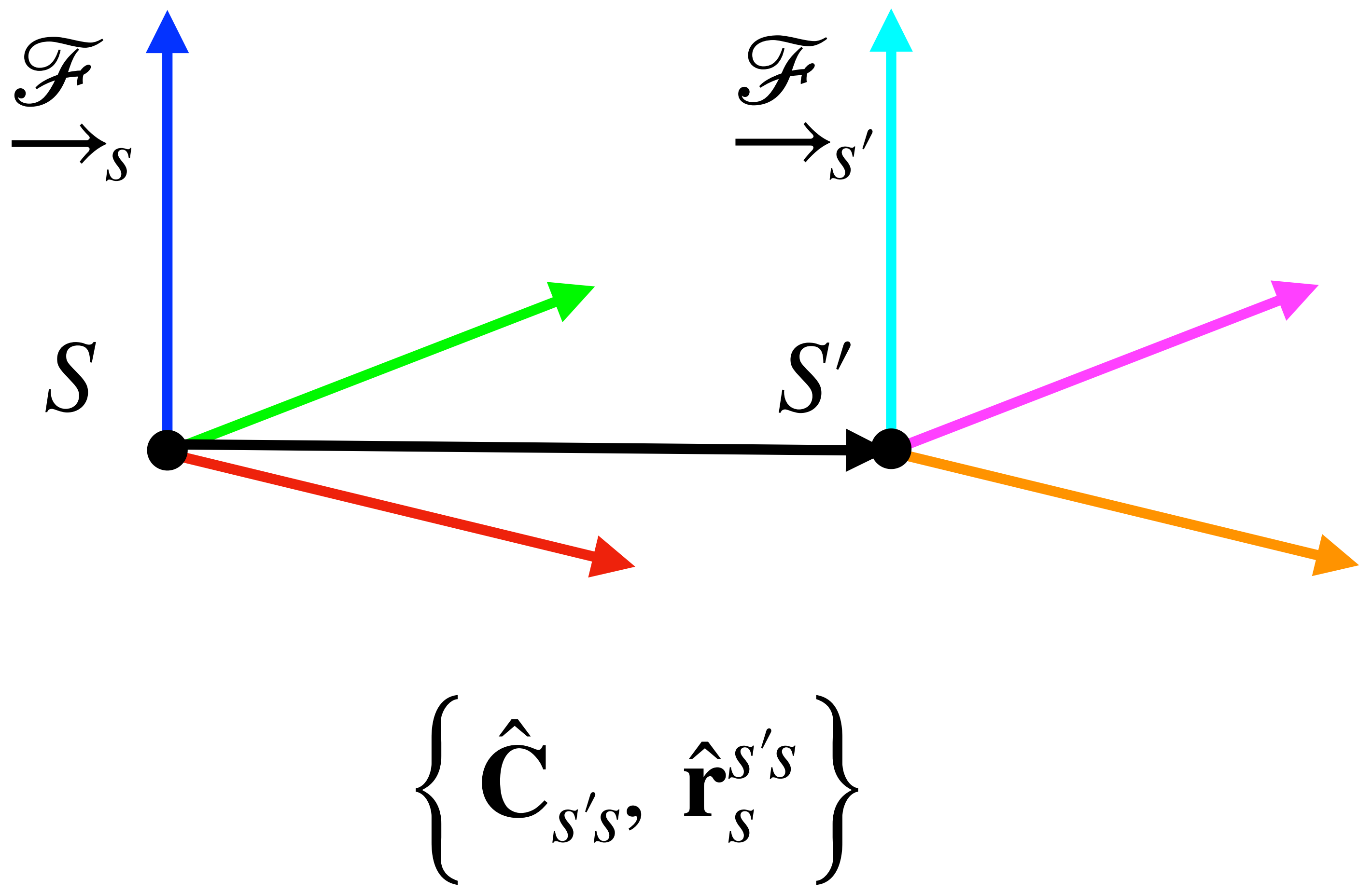


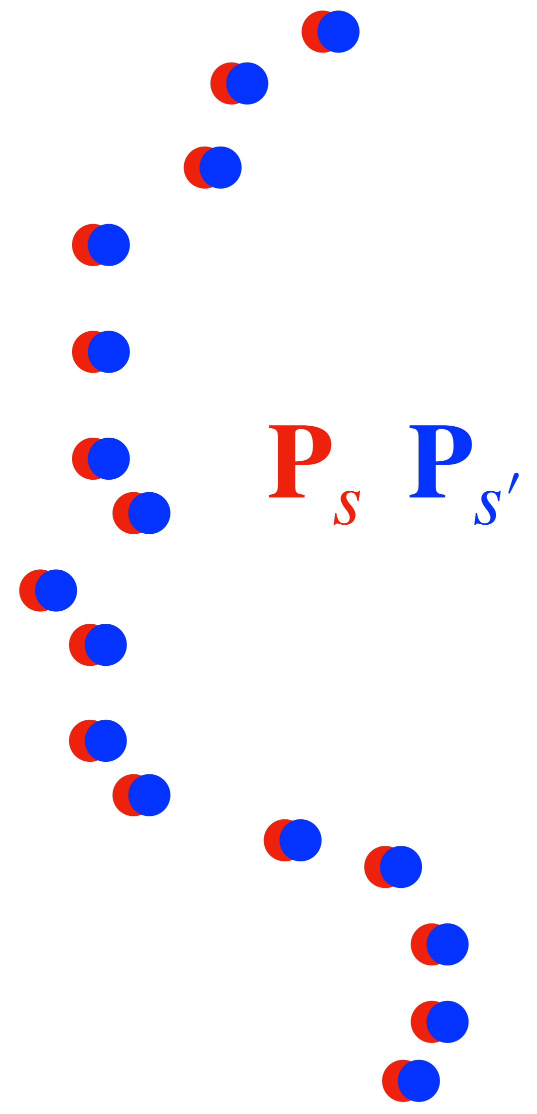
$$\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\}$$



State Estimation via Point Set Registration

What motion of the car best aligns the two point clouds?

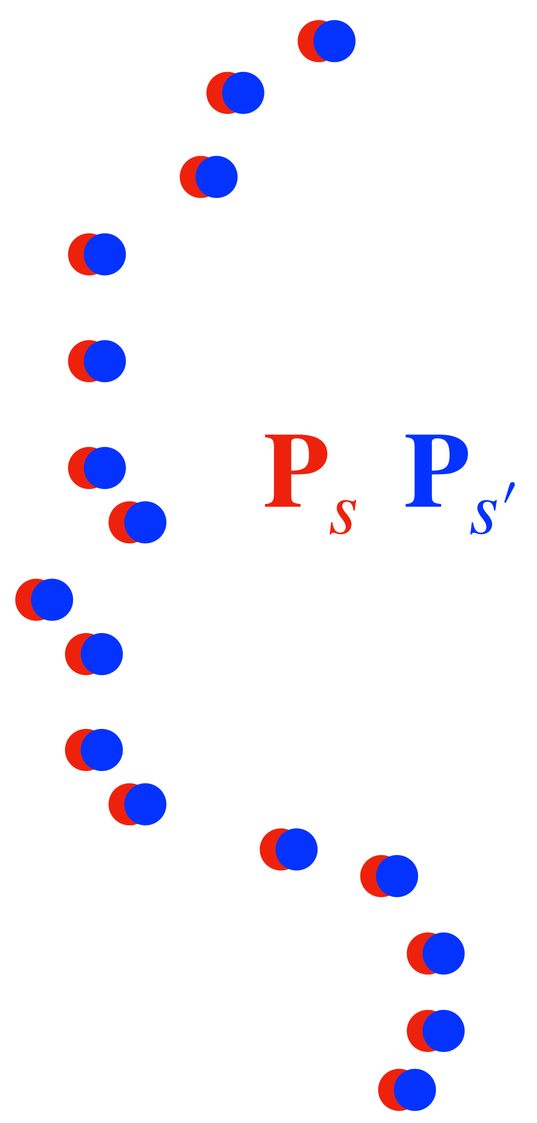




Problem: We don't know which points correspond to each other

The Iterative Closest Point (ICP) Algorithm

Intuition: When the optimal motion is found, corresponding points will be closer to each other than to other points



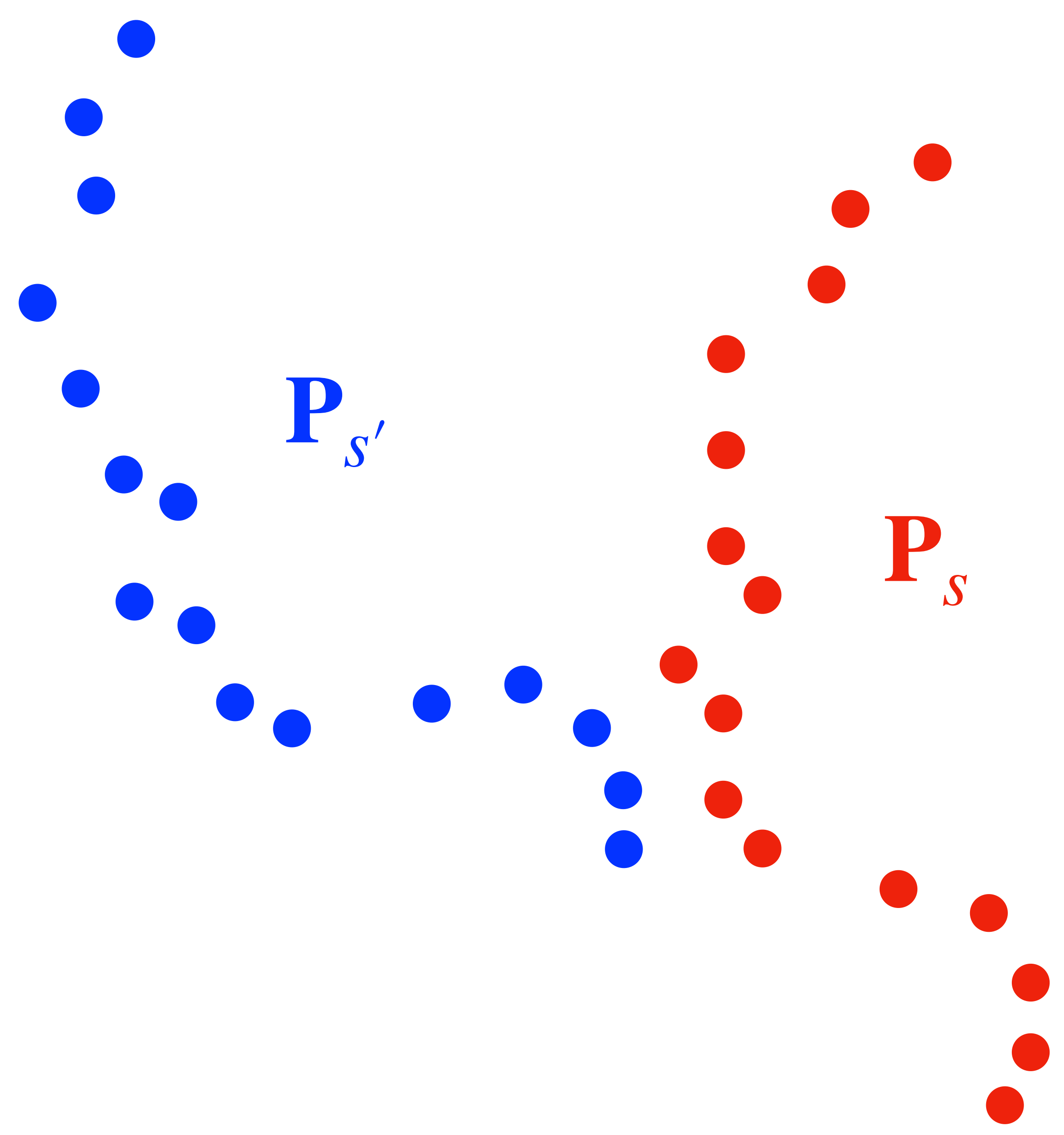
Heuristic: For each point, the best candidate for a corresponding point is the point that is closest to it right now

The Iterative Closest Point (ICP) Algorithm

Procedure:

1. Get an initial guess for the transformation $\left\{ \check{\mathbf{C}}_{s's}, \check{\mathbf{r}}_s^{s's} \right\}$

a motion model from IMU
or wheel odometry.

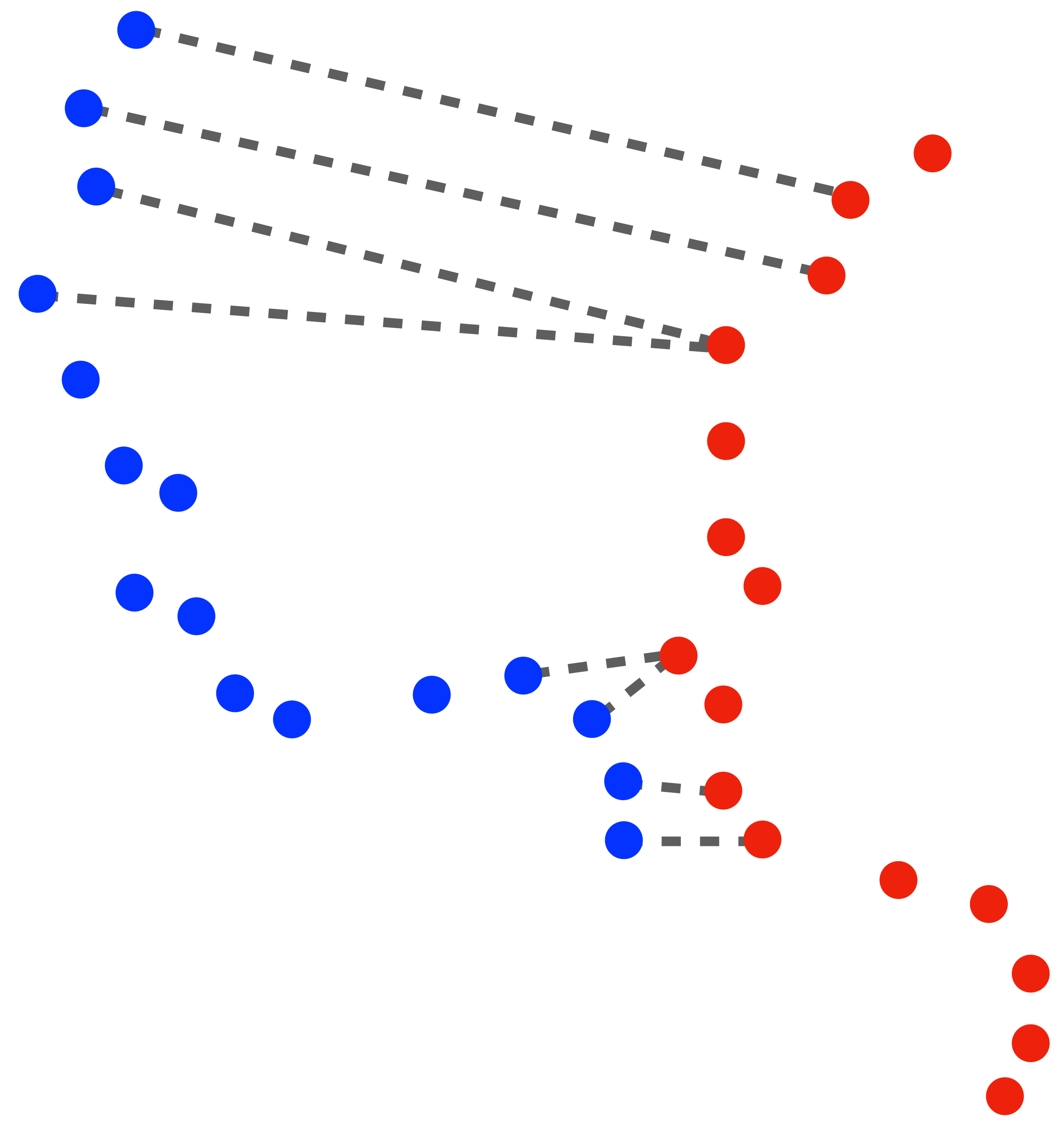


The Iterative Closest Point (ICP) Algorithm

Procedure:

1. Get an initial guess for the transformation $\left\{\check{\mathbf{C}}_{s's}, \check{\mathbf{r}}_s^{s's}\right\}$

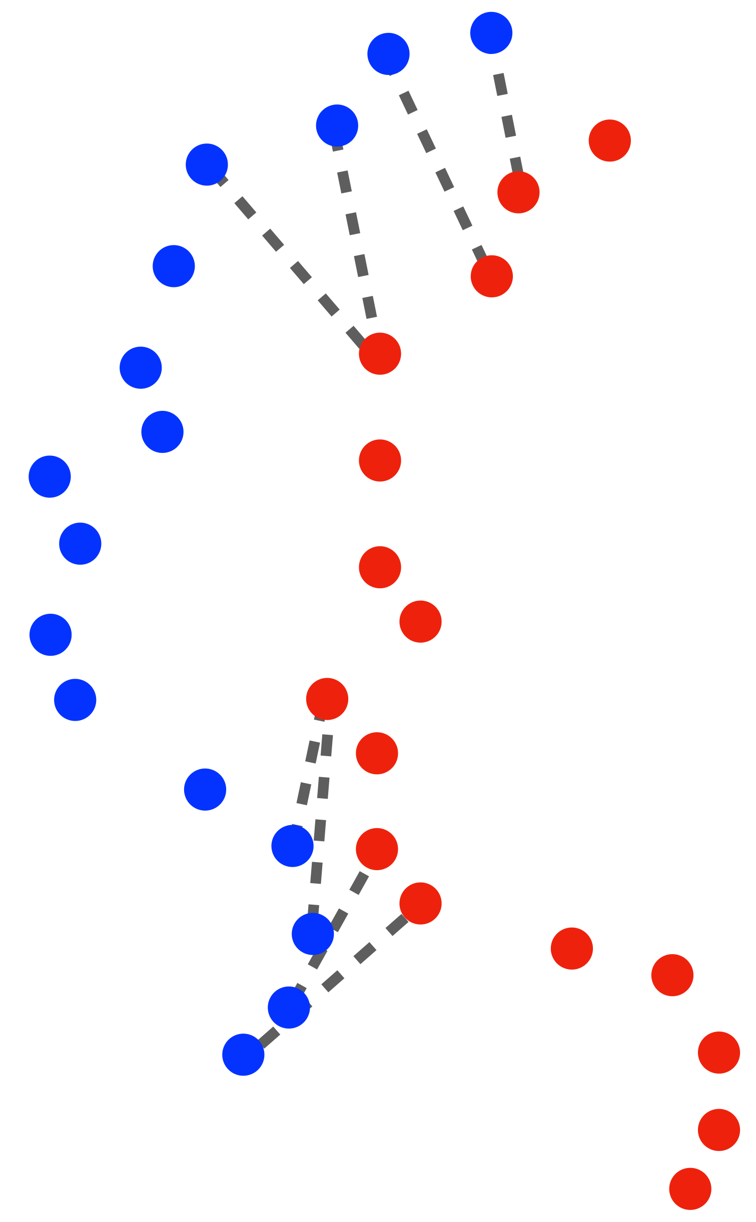
2. Associate each point in $\mathbf{P}_{s'}$ with the nearest point in $\mathbf{P}_s$



The Iterative Closest Point (ICP) Algorithm

Procedure:

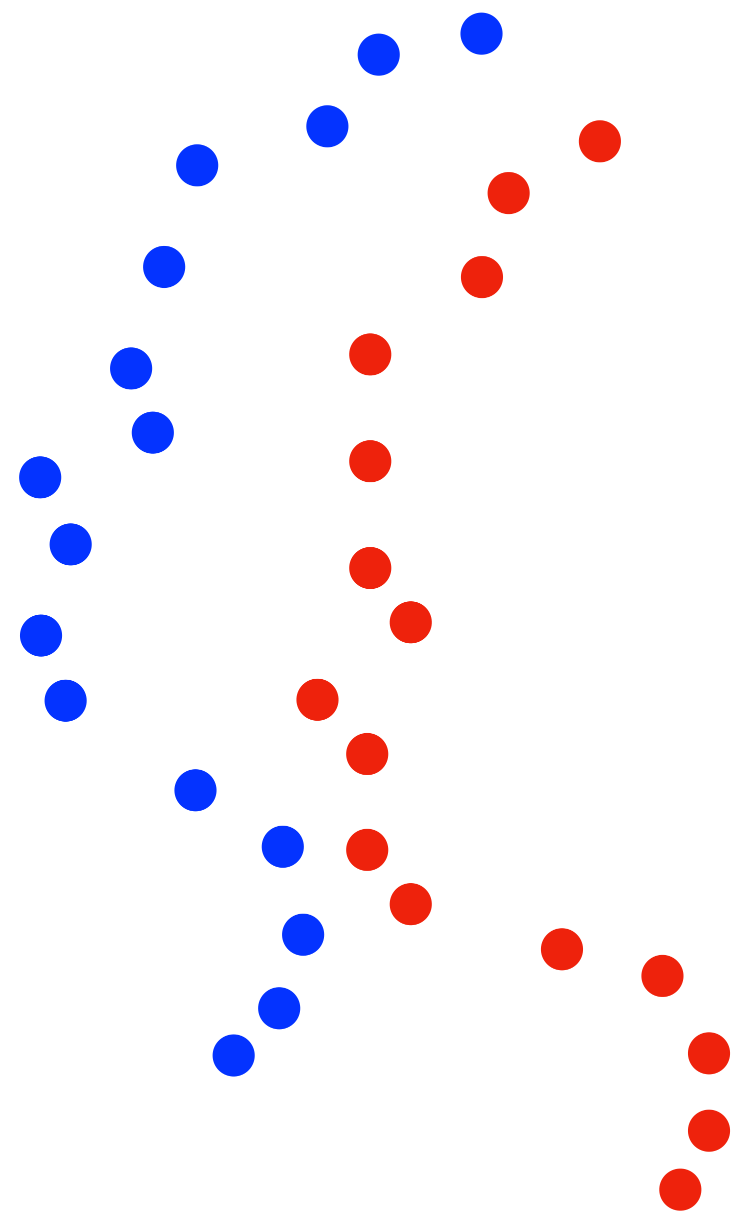
1. Get an initial guess for the transformation $\left\{ \check{\mathbf{C}}_{s's}, \check{\mathbf{r}}_s^{s's} \right\}$
2. Associate each point in $\mathbf{P}_{s'}$ with the nearest point in $\mathbf{P}_s$
3. Solve for the optimal transformation $\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\}$



The Iterative Closest Point (ICP) Algorithm

Procedure:

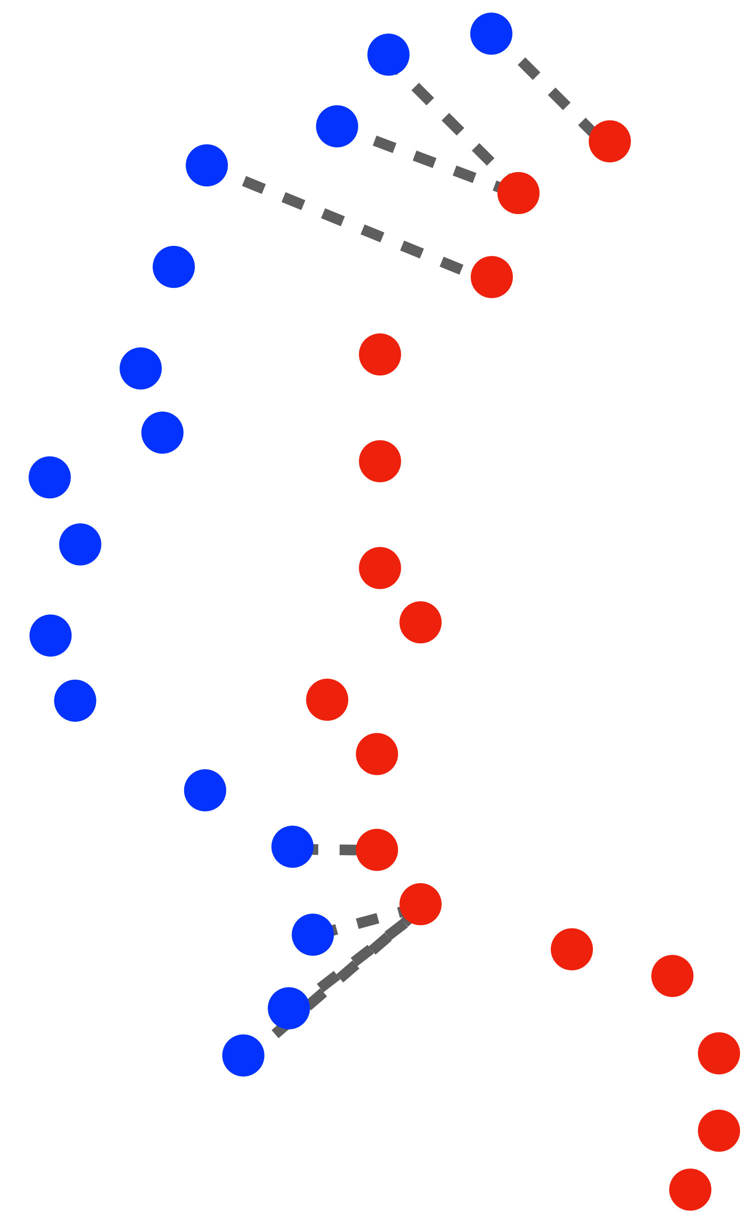
1. Get an initial guess for the transformation $\left\{ \check{\mathbf{C}}_{s's}, \check{\mathbf{r}}_s^{s's} \right\}$
2. Associate each point in $\mathbf{P}_{s'}$ with the nearest point in $\mathbf{P}_s$
3. Solve for the optimal transformation $\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\}$
4. Repeat until convergence



The Iterative Closest Point (ICP) Algorithm

Procedure:

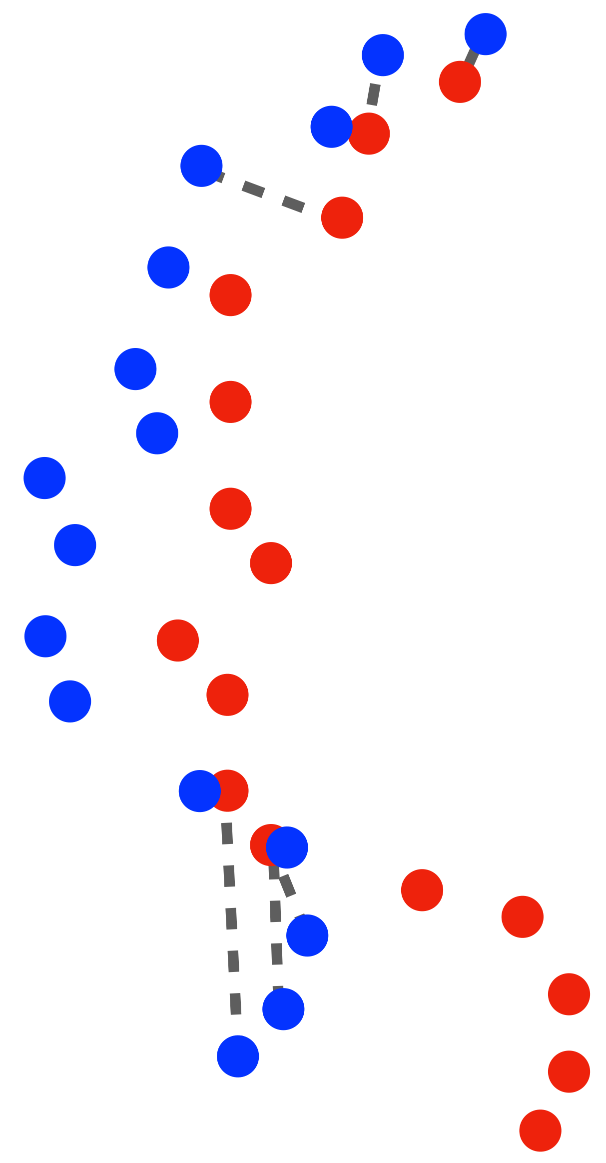
1. Get an initial guess for the transformation $\left\{ \check{\mathbf{C}}_{s's}, \check{\mathbf{r}}_s^{s's} \right\}$
2. Associate each point in $\mathbf{P}_{s'}$ with the nearest point in $\mathbf{P}_s$
3. Solve for the optimal transformation $\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\}$
4. Repeat until convergence



The Iterative Closest Point (ICP) Algorithm

Procedure:

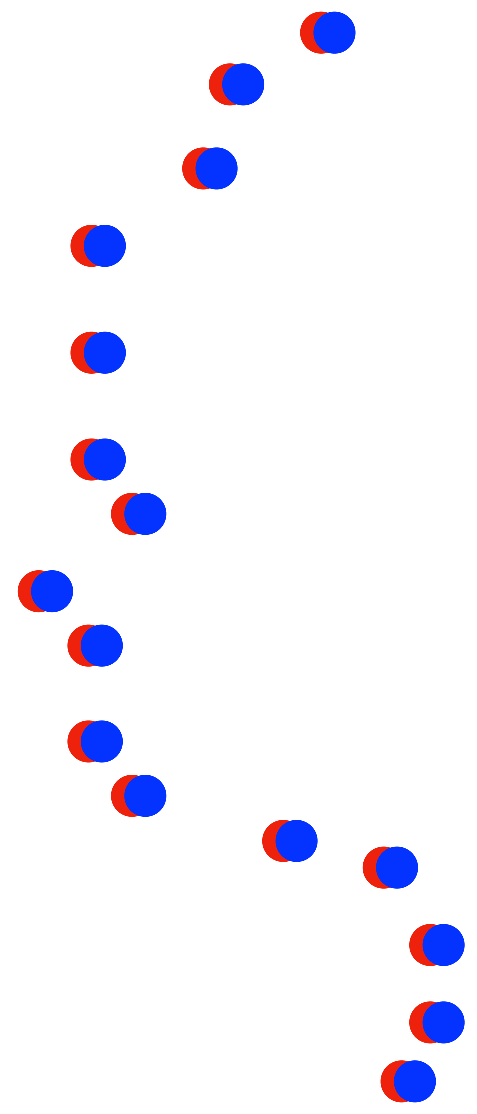
1. Get an initial guess for the transformation $\left\{ \check{\mathbf{C}}_{s's}, \check{\mathbf{r}}_s^{s's} \right\}$
2. Associate each point in $\mathbf{P}_{s'}$ with the nearest point in $\mathbf{P}_s$
3. Solve for the optimal transformation $\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\}$
4. Repeat until convergence



The Iterative Closest Point (ICP) Algorithm

Procedure:

1. Get an initial guess for the transformation $\left\{ \check{\mathbf{C}}_{s's}, \check{\mathbf{r}}_s^{s's} \right\}$
2. Associate each point in $\mathbf{P}_{s'}$ with the nearest point in $\mathbf{P}_s$
3. Solve for the optimal transformation $\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\}$
4. Repeat until convergence



ICP | Solving for the Optimal Transformation

How do we solve for the optimal $\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\}$ at each step?

Use least-squares!

$$\left\{ \hat{\mathbf{C}}_{s's}, \hat{\mathbf{r}}_s^{s's} \right\} = \operatorname{argmin}_{\{\mathbf{C}_{s's}, \mathbf{r}_s^{s's}\}} \mathscr{L}_{\mathrm{LS}} \left(\mathbf{C}_{s's}, \mathbf{r}_s^{s's} \right)$$

$$\mathscr{L}_{\mathrm{LS}} \left(\mathbf{C}_{s's}, \mathbf{r}_s^{s's} \right) = \sum_{j=1}^{n} \left\| \mathbf{C}_{s's} \left(\mathbf{p}_s^{(j)} - \mathbf{r}_s^{s's} \right) - \mathbf{p}_{s'}^{(j)} \right\|_2^2$$

Careful: Rotations need special treatment because they don't behave like vectors!

ICP | Solving for the Optimal Transformation

1. Compute the *centroids* of each point cloud

$$\boldsymbol{\mu}_s = \frac{1}{n}\sum_{j=1}^{n} \mathbf{p}_s^{(j)} \qquad\qquad \boldsymbol{\mu}_{s'} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{p}_{s'}^{(j)}$$

2. Compute a matrix capturing the *spread* of the two point clouds

$$\mathbf{W}_{s's} = \frac{1}{n}\sum_{j=1}^{n} \left(\mathbf{p}_s^{(j)} - \boldsymbol{\mu}_s\right)\left(\mathbf{p}_{s'}^{(j)} - \boldsymbol{\mu}_{s'}\right)^T$$

ICP | Solving for the Optimal Transformation

3. Use the *singular value decomposition* of the matrix to get the optimal rotation

singular values (scaling)

diagonal matrix

$$\mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{W}_{s's}$$

$\mathbf{U}^T\mathbf{U} = \mathbf{1}$ (rotation)

$\mathbf{V}^T\mathbf{V} = \mathbf{1}$ (rotation)

Do not want scaling.

$$\hat{\mathbf{C}}_{s's} = \mathbf{U}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det \mathbf{U} \det \mathbf{V} \end{bmatrix}\mathbf{V}^T$$

This term ensures that we get a proper rotation without any reflection

ICP | Solving for the Optimal Transformation

3. Use the *singular value decomposition* of the matrix to get the optimal rotation

singular values
(scaling)

$$\mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{W}_{s's}$$

$\mathbf{U}^T\mathbf{U} = \mathbf{1}$
(rotation)

$\mathbf{V}^T\mathbf{V} = \mathbf{1}$
(rotation)

$$\hat{\mathbf{C}}_{s's} = \mathbf{U}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det\mathbf{U}\det\mathbf{V} \end{bmatrix}\mathbf{V}^T$$

This term ensures that we get a
proper rotation without any reflection

4. Use the optimal rotation to get the optimal translation by *aligning the centroids*

$$\hat{\mathbf{r}}_s^{s's} = \boldsymbol{\mu}_s - \hat{\mathbf{C}}_{s's}^T \boldsymbol{\mu}_{s'}$$

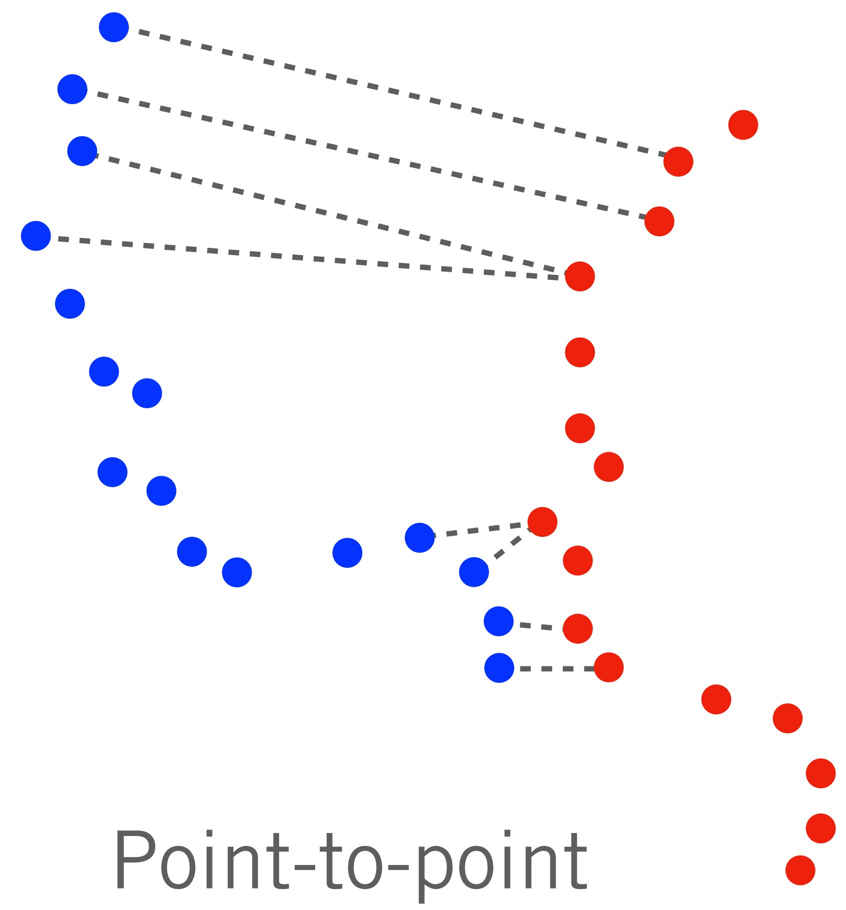
ICP | Estimating Uncertainty

We can obtain an estimate of the covariance matrix of the ICP solution using this formula:

$$\underbrace{\mathrm{cov}(\hat{\mathbf{x}})}_{\text{covariance of estimated motion parameters}} \simeq \left[\left(\frac{\partial^2 \mathcal{L}}{\partial \mathbf{x}^2} \right)^{-1} \frac{\partial^2 \mathcal{L}}{\partial \mathbf{z} \partial \mathbf{x}} \underbrace{\mathrm{cov}(\mathbf{z})}_{\text{covariance of all measurements (both point clouds)}} \frac{\partial^2 \mathcal{L}}{\partial \mathbf{z} \partial \mathbf{x}}^T \left(\frac{\partial^2 \mathcal{L}}{\partial \mathbf{x}^2} \right)^{-1} \right]_{\mathbf{x}=\hat{\mathbf{x}}}$$

ICP | Variants

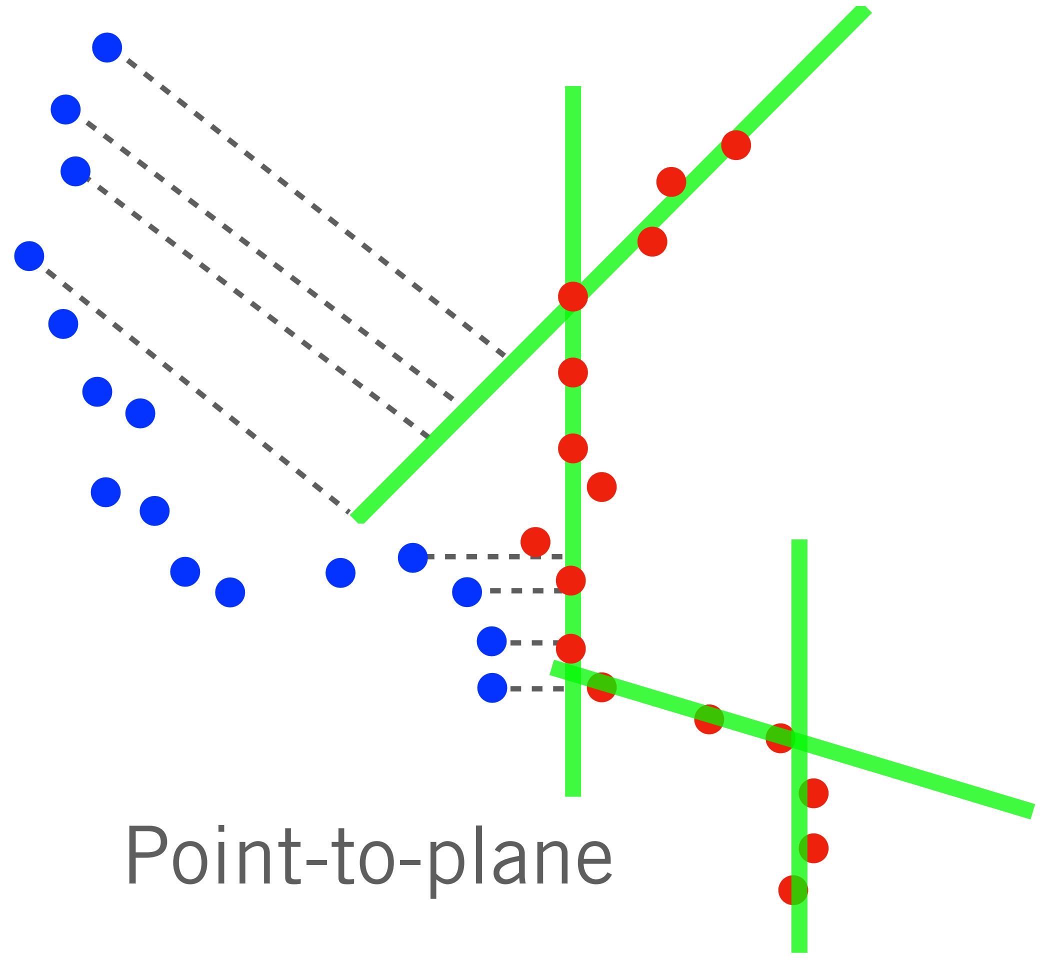
Point-to-point ICP minimizes the *Euclidean distance* between each point in $\mathbf{P}_{s'}$ and the *nearest point* in $\mathbf{P}_s$



Point-to-point

Point-to-plane ICP minimizes the *perpendicular distance* between each point in $\mathbf{P}_{s'}$ and the *nearest plane* in $\mathbf{P}_s$

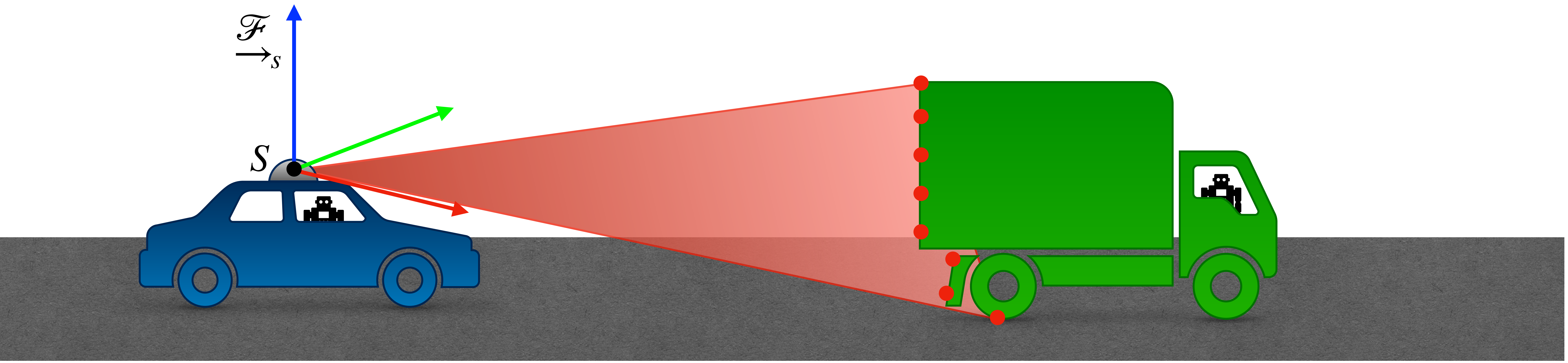
- This tends to work well in structured environments like cities



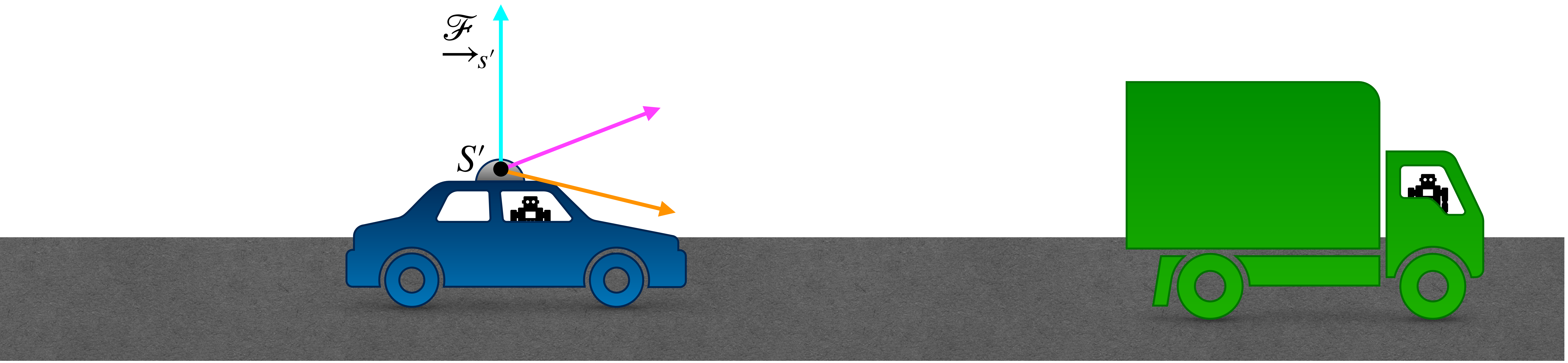
Point-to-plane

Outliers | Objects in Motion

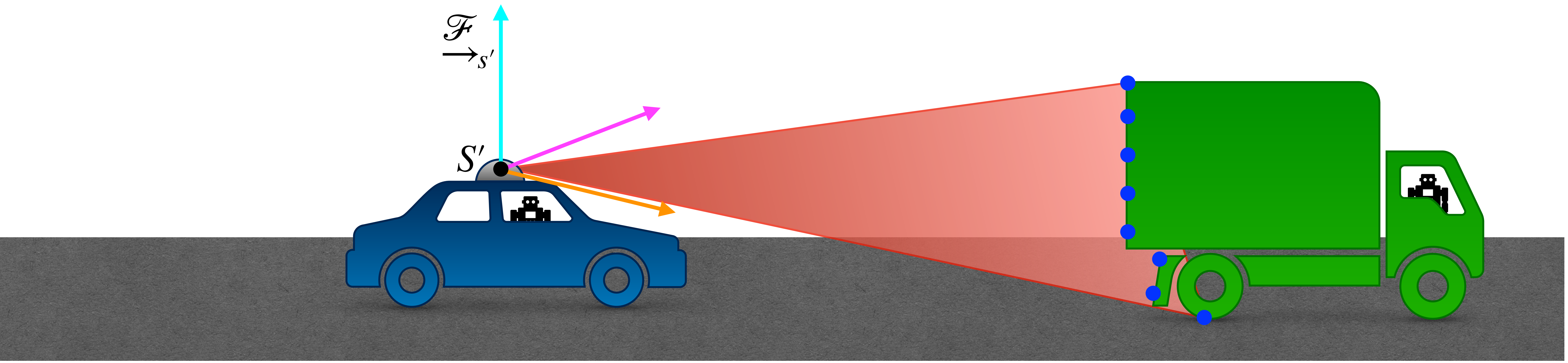
Outliers | Objects in Motion



Outliers | Objects in Motion

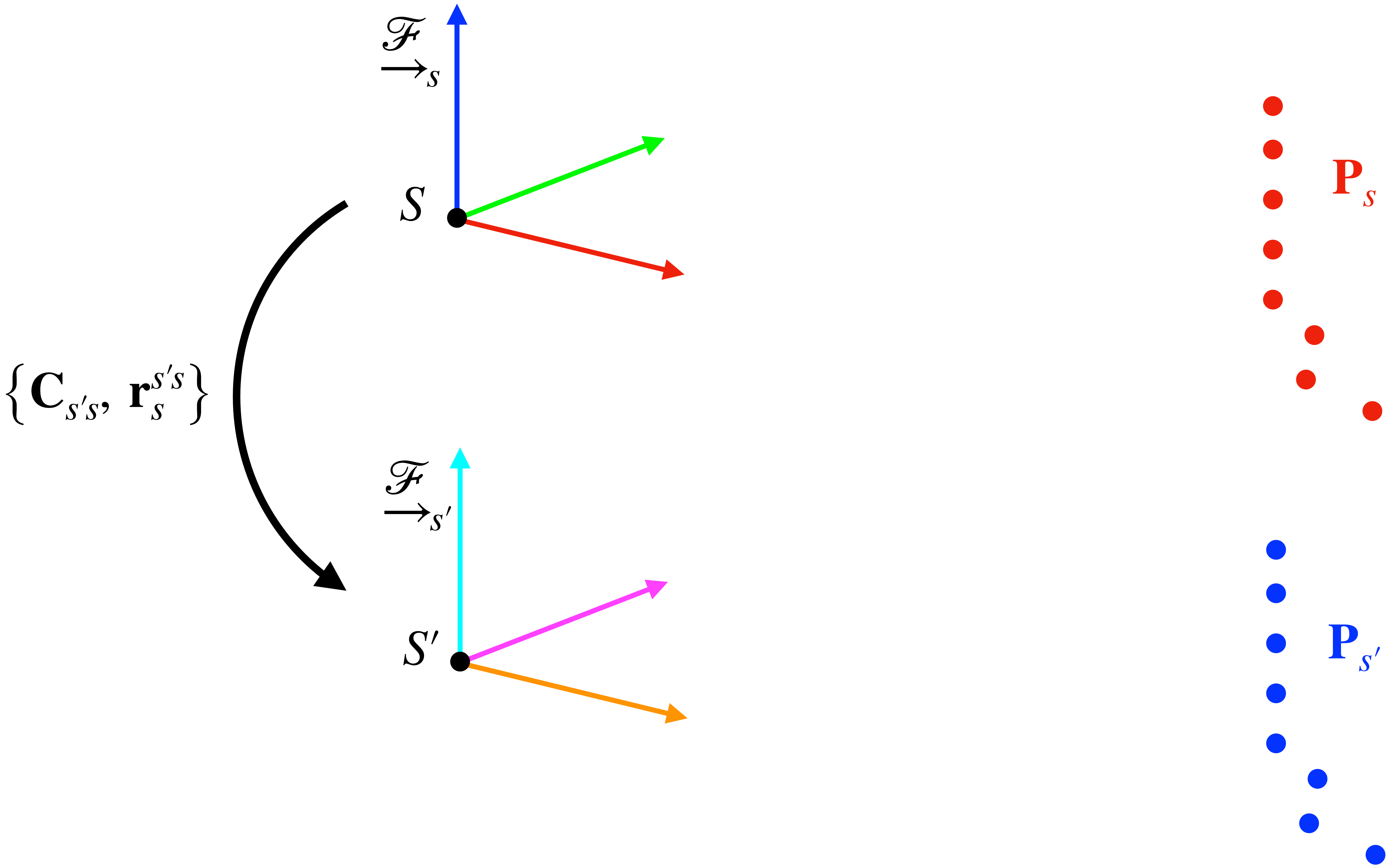


Outliers | Objects in Motion



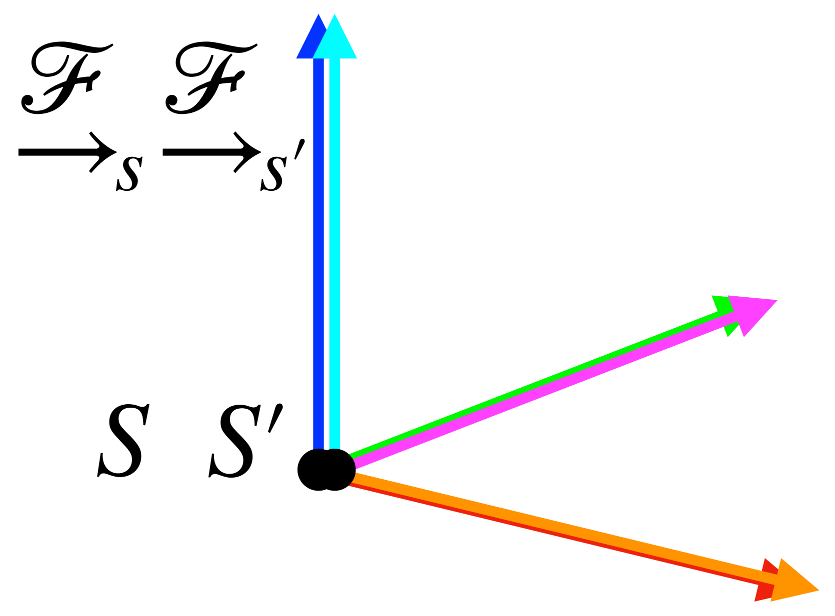
Outliers | Objects in Motion

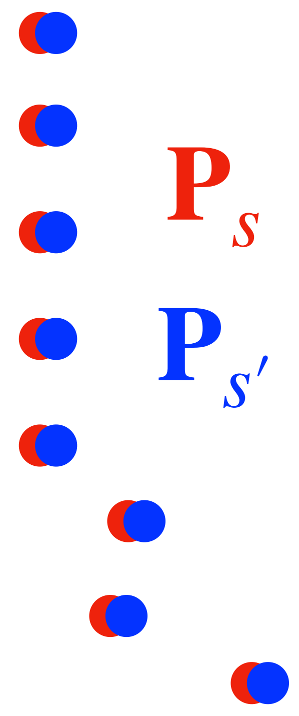
What motion of the car best aligns the two point clouds?



Outliers | Objects in Motion

What motion of the car best aligns the two point clouds?





ICP alone is not always enough!

Outliers | Robust Loss Functions

Error term

$$\mathbf{e}^{(j)} = \mathbf{C}_{s's}\left(\mathbf{p}_s^{(j)} - \mathbf{r}_s^{s's}\right) - \mathbf{p}_{s'}^{(j)}$$

Least-squares

$$\mathscr{L} = \sum_{j=1}^{n} \mathbf{e}^{(j)\,T}\mathbf{e}^{(j)}$$

Outliers | Robust Loss Functions

Large errors induced by

Less sensitive to outliers

No closed-form solution

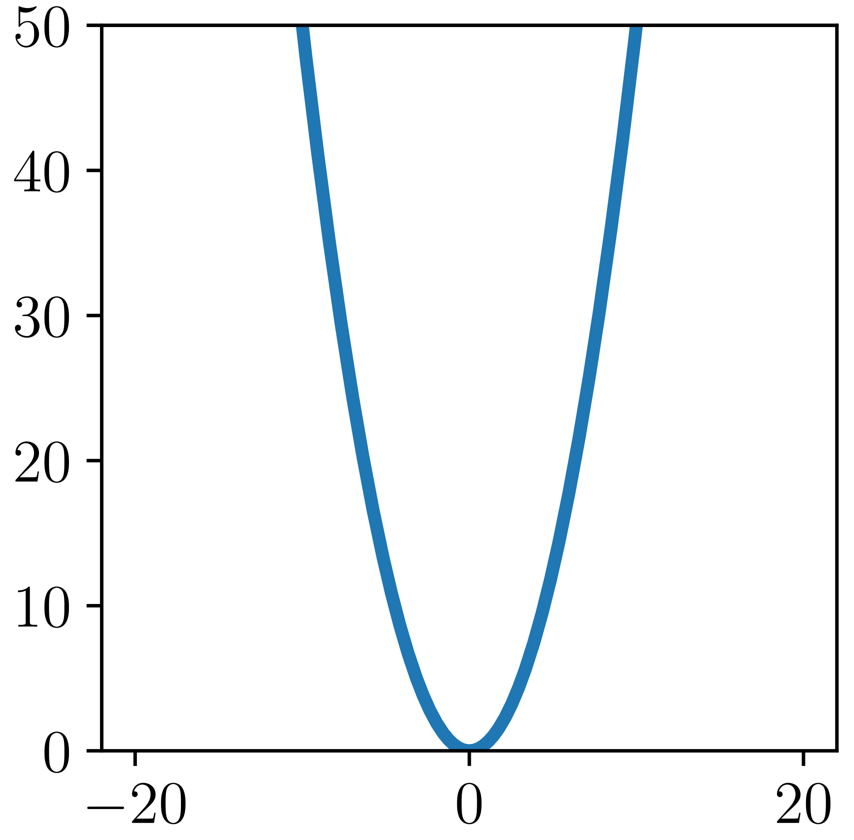
Error term

$$\mathbf{e}^{(j)} = \mathbf{C}_{s's}\left(\mathbf{p}_s^{(j)} - \mathbf{r}_s^{s's}\right) - \mathbf{p}_{s'}^{(j)}$$

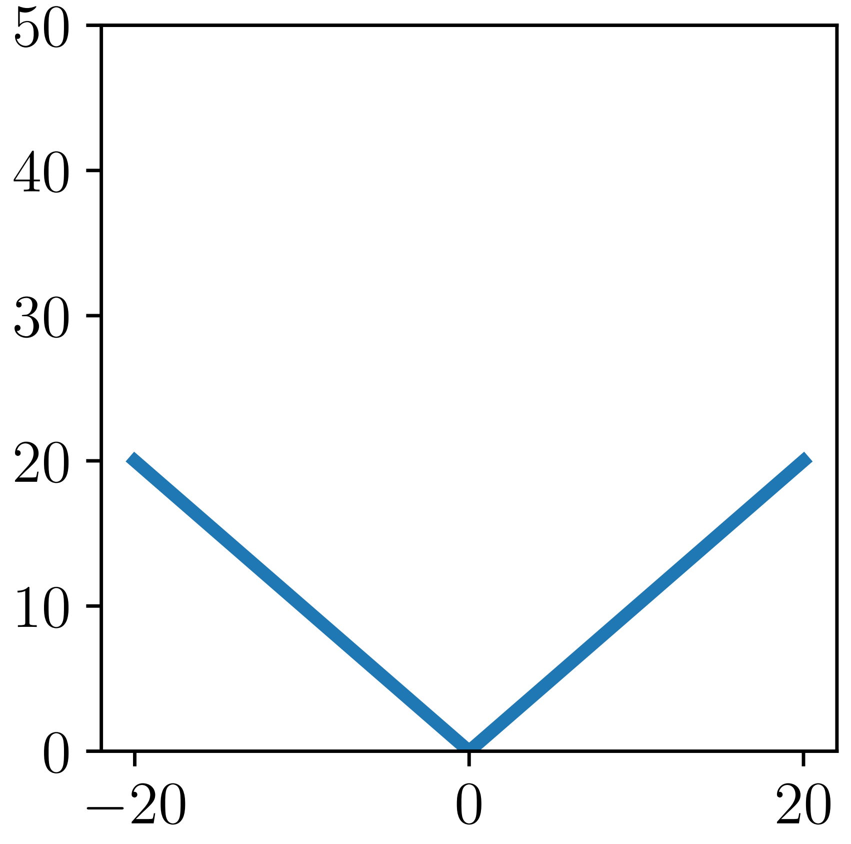
Robust loss

$$\mathscr{L} = \sum_{j=1}^{n} \rho\left(\mathbf{e}^{(j)}\right)$$

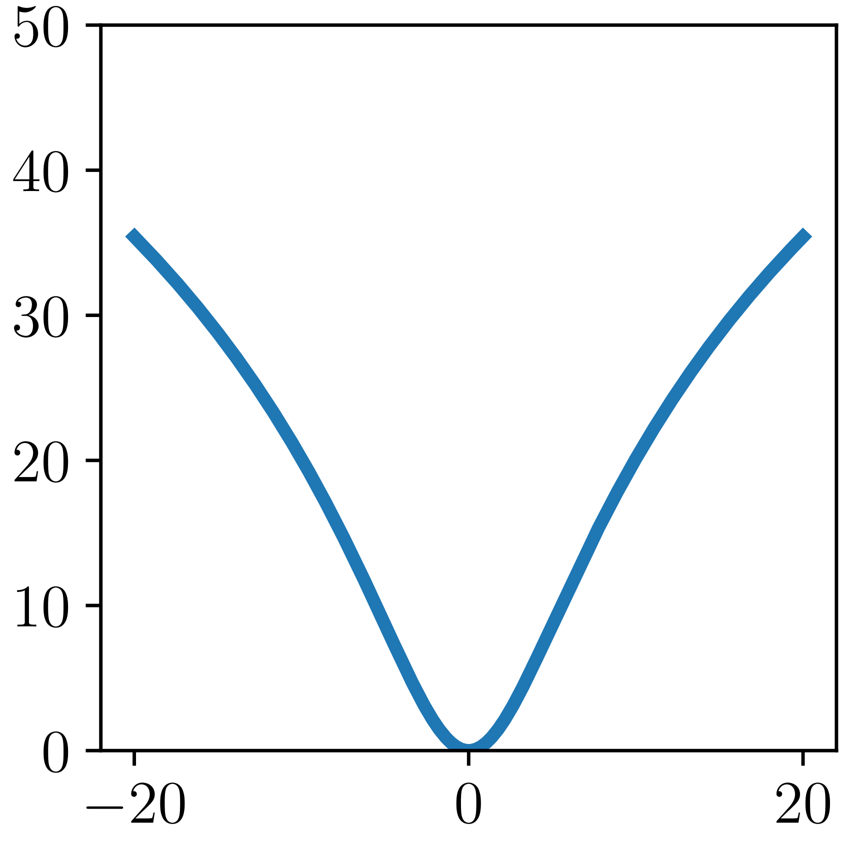
Squared error (L_2)



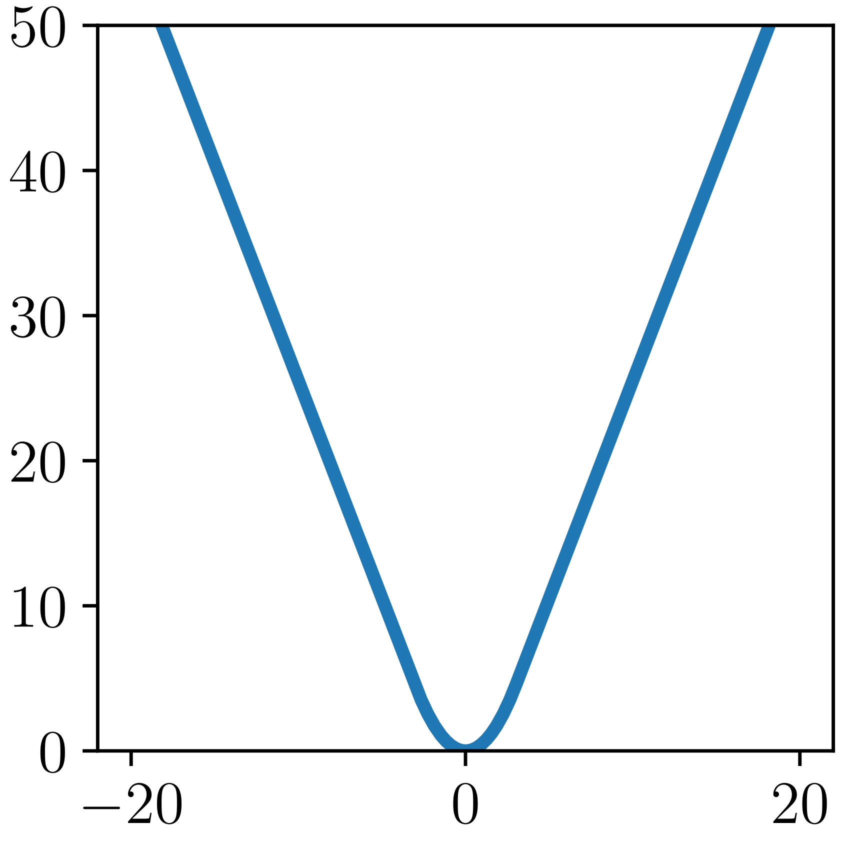
Absolute error (L_1)



Cauchy loss



Huber loss



$$\rho(\mathbf{e}) = \frac{1}{2}\|\mathbf{e}\|_2^2 = \frac{1}{2}\mathbf{e}^T\mathbf{e}$$

$$\rho(\mathbf{e}) = \|\mathbf{e}\|_1 = \sum_i |e_i|$$

$$\rho(\mathbf{e}) = \frac{k^2}{2}\log\left(1 + \frac{1}{k^2}\|\mathbf{e}\|_2^2\right)$$

$$\rho(e) = \begin{cases} \frac{1}{2}\|\mathbf{e}\|_2^2 \\ k\left(\|\mathbf{e}\|_1 - \frac{k}{2}\right) \end{cases}$$

Summary | Pose Estimation from LIDAR Data

- ICP is a way to determine the motion of a self-driving car by aligning point clouds from LIDAR or other sensors

- ICP *iteratively* minimizes the distance between points in each point cloud

- ICP is sensitive to outliers caused by moving objects, which can be partly mitigated using robust loss functions

Summary | LIDAR Sensing

- LIDAR measures distances using laser light and the *time-of-flight* equation

- LIDAR scans are stored as *points clouds* that can be manipulated using common spatial operations (e.g., translation, rotation, scaling)

- The Iterative Closest Point (ICP) algorithm is one way of using LIDAR to localize a self-driving car