

MODULE 4 LESSON 2

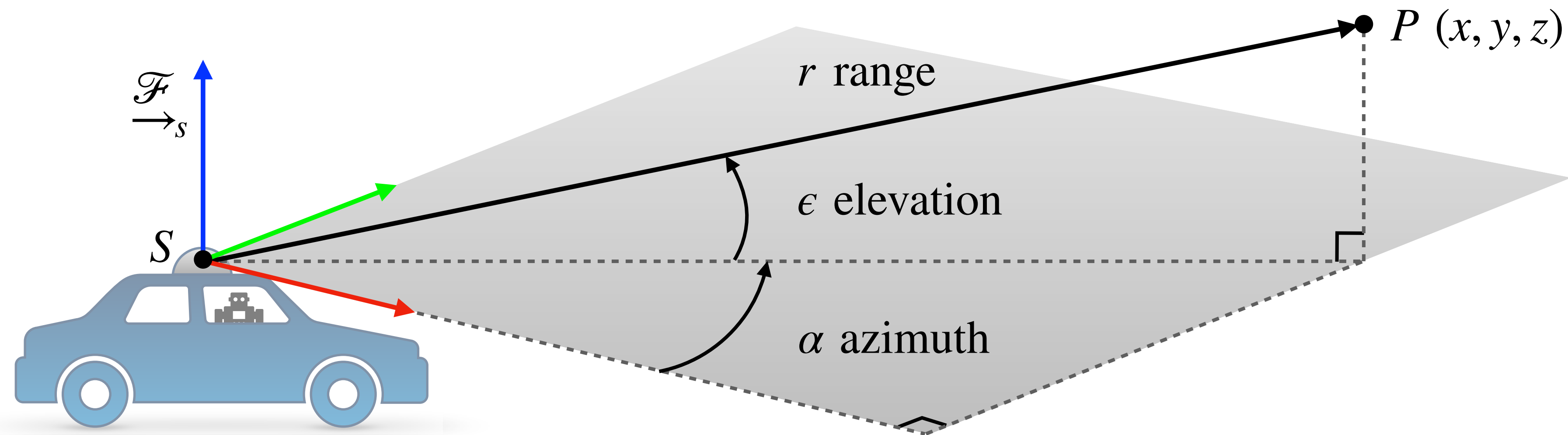
LIDAR SENSOR MODELS AND POINT CLOUDS

LIDAR Sensor Models and Point Clouds

By the end of this lesson, you will be able to...

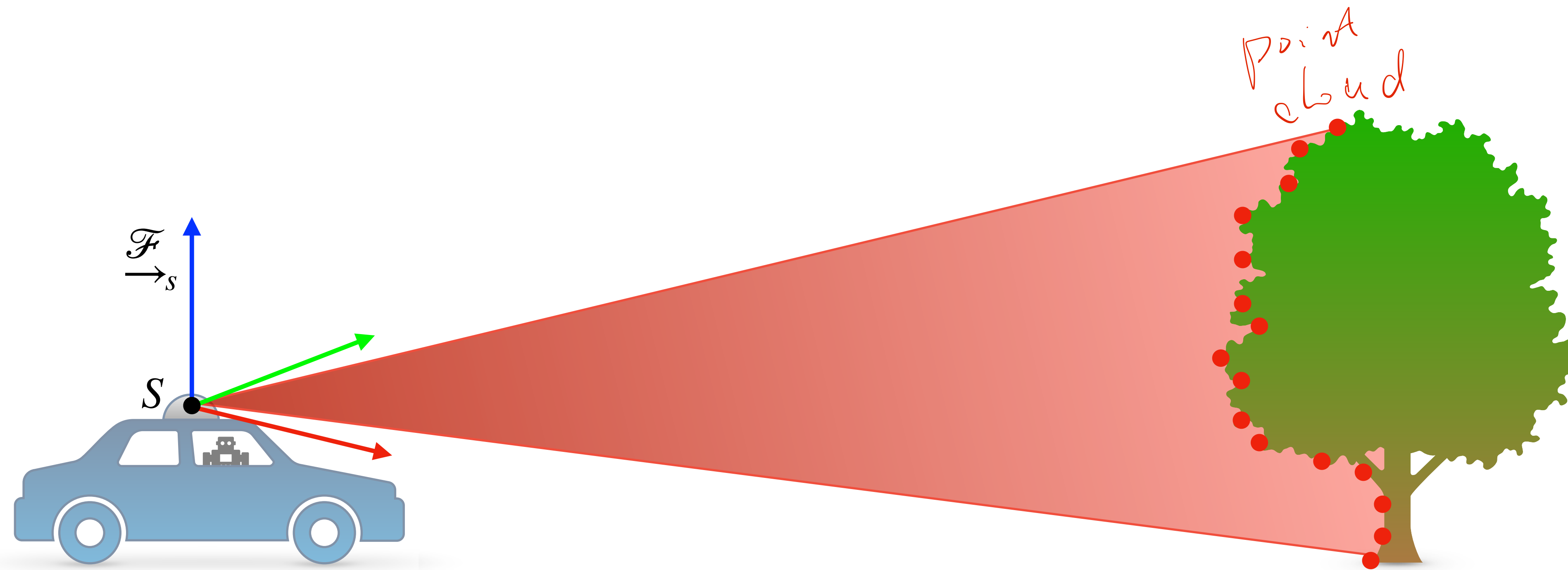
- Describe the basic point cloud data structure
- Describe common spatial operations on point clouds
- Use the method of least squares to fit a plane to a point cloud

LIDAR Point Clouds



$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{h}^{-1}(r, \alpha, \epsilon) = \begin{bmatrix} r \cos \alpha \cos \epsilon \\ r \sin \alpha \cos \epsilon \\ r \sin \epsilon \end{bmatrix}$$

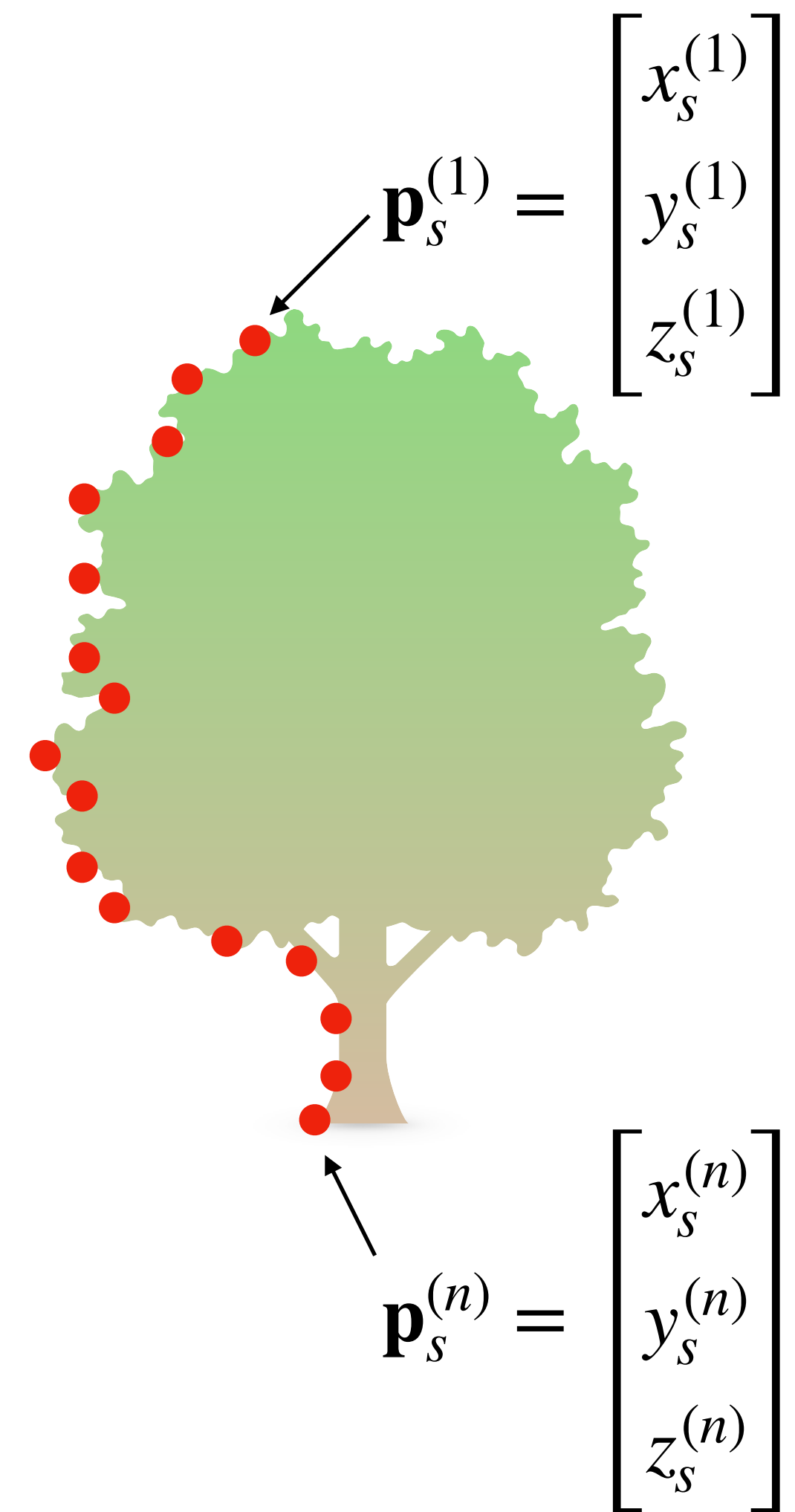
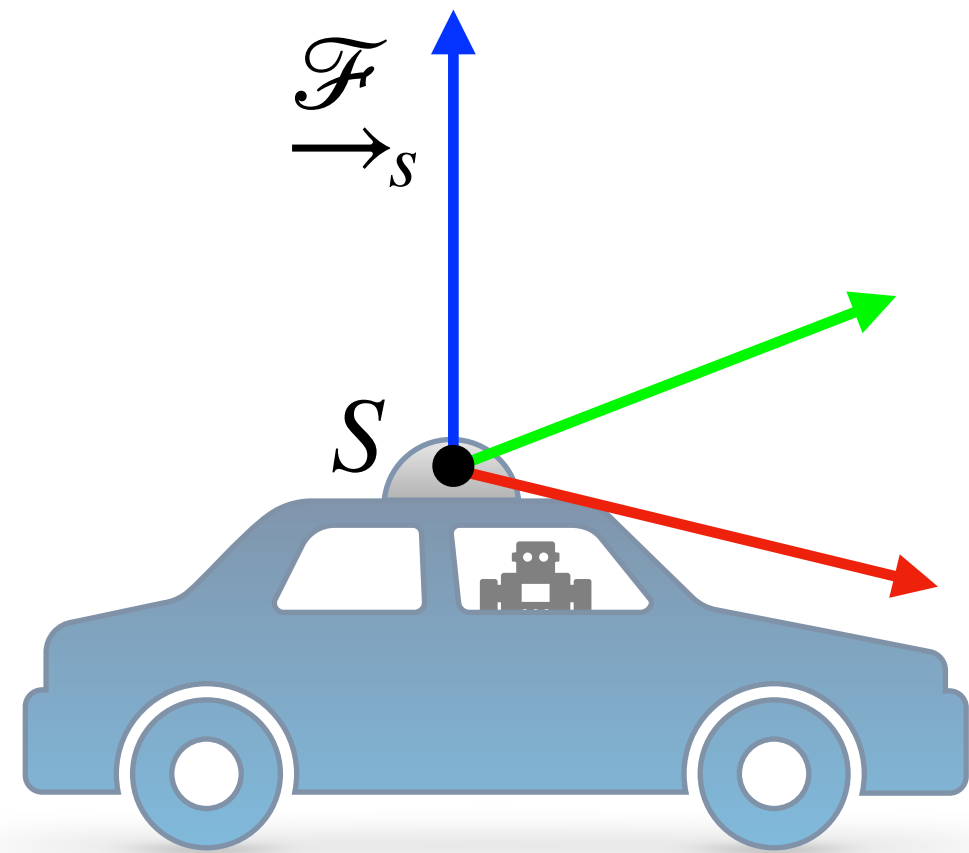
LIDAR Point Clouds



LIDAR Point Clouds | Data Structures

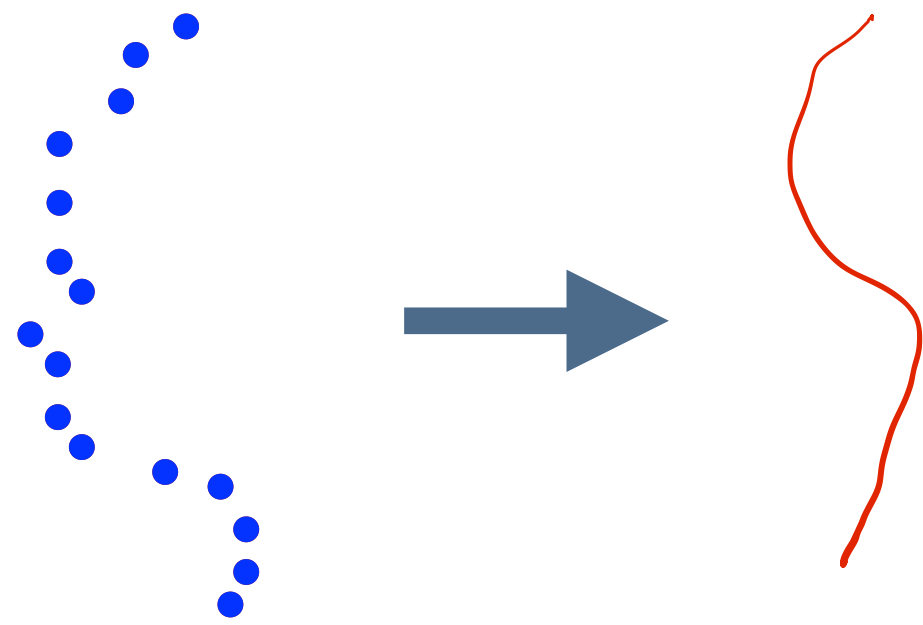
$$\mathbf{P}_s = [\mathbf{p}_s^{(1)} \quad \mathbf{p}_s^{(2)} \quad \dots \quad \mathbf{p}_s^{(n)}] = \begin{bmatrix} x_s^{(1)} & x_s^{(2)} & \dots & x_s^{(n)} \\ y_s^{(1)} & y_s^{(2)} & \dots & y_s^{(n)} \\ z_s^{(1)} & z_s^{(2)} & \dots & z_s^{(n)} \end{bmatrix}$$

one point

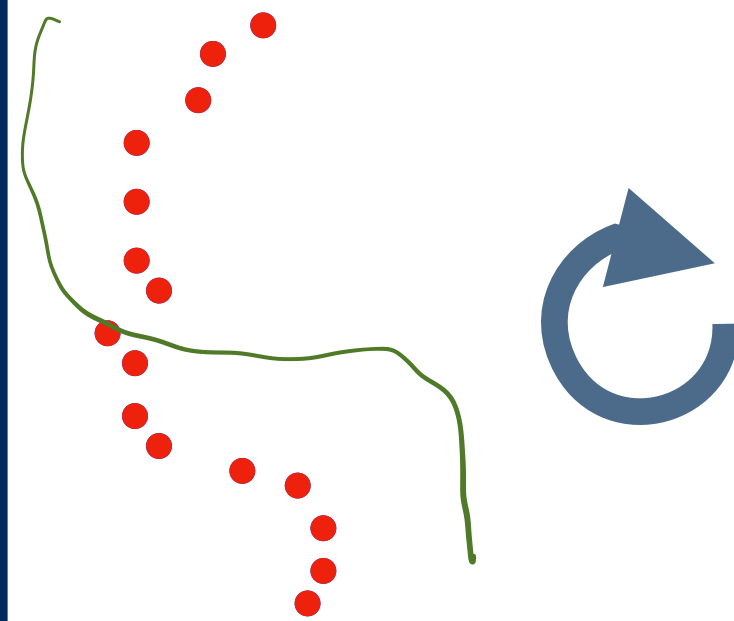


Operations on Point Clouds

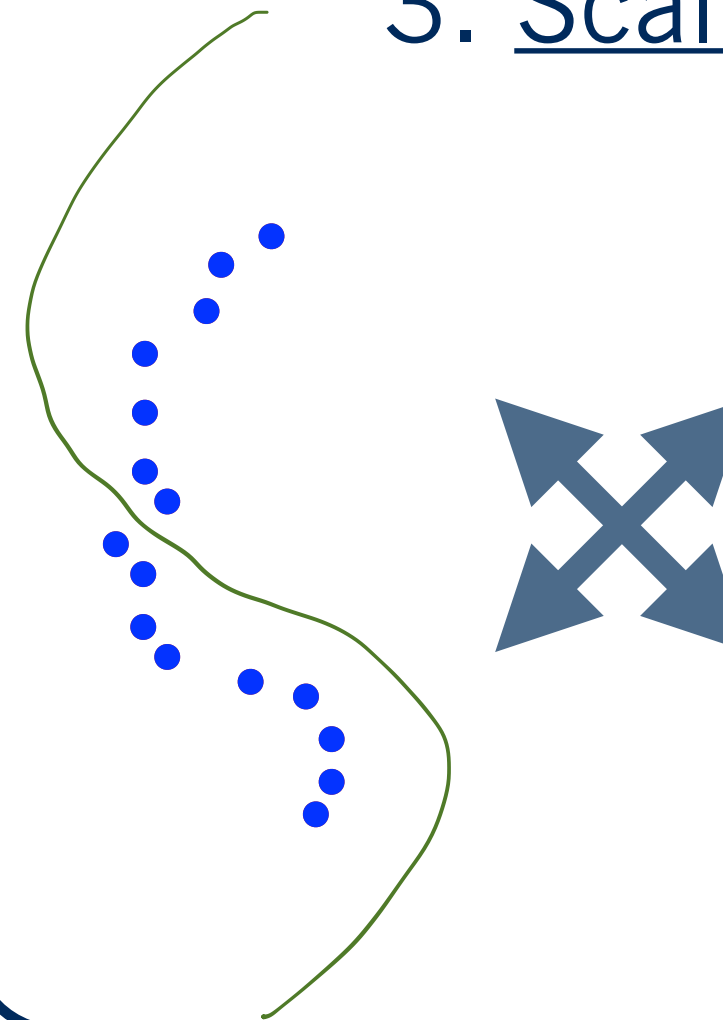
1. Translation



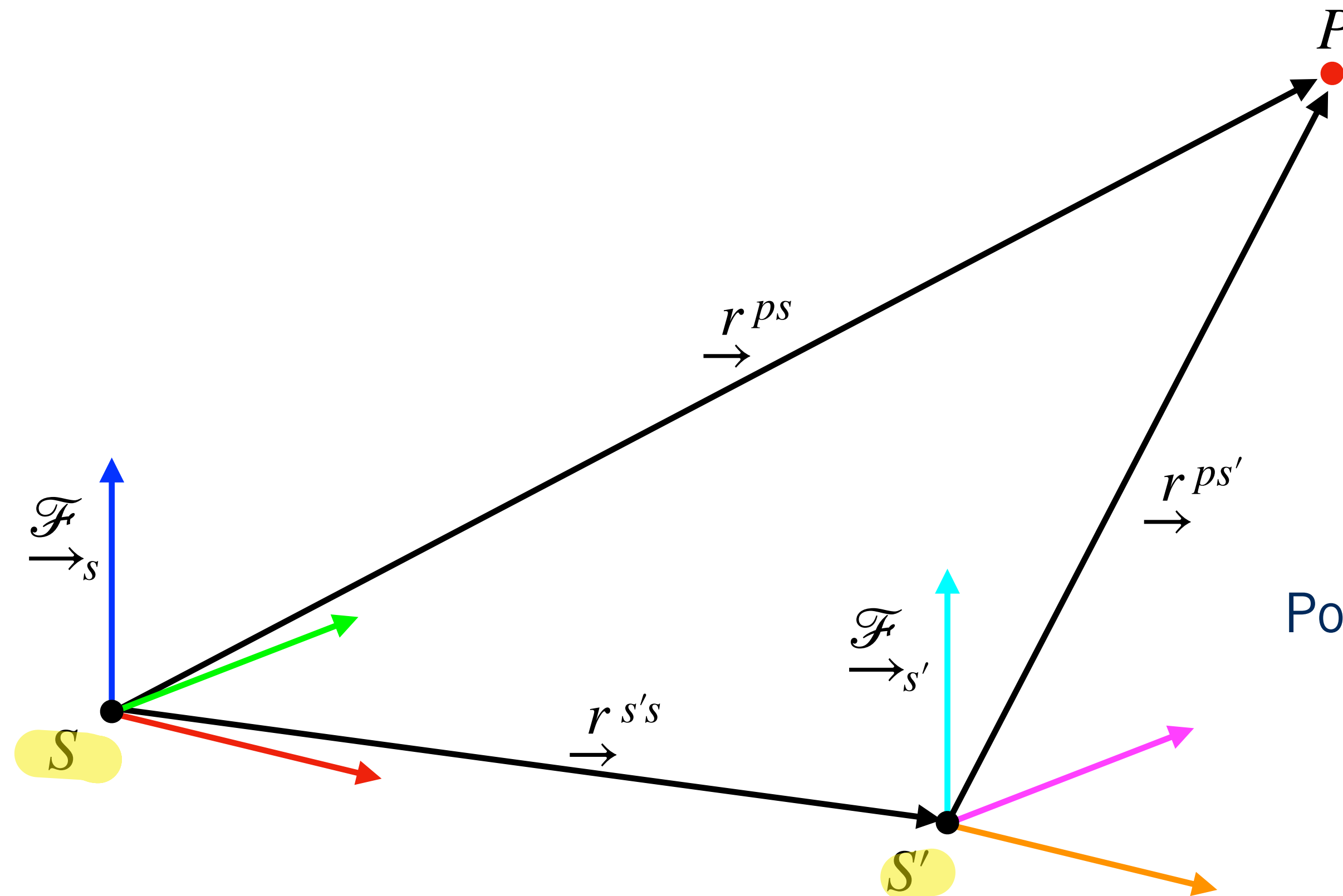
2. Rotation



3. Scaling



Operations on Point Clouds | Translation



From vector addition

$$\vec{r}^{ps'} = \vec{r}^{ps} - \vec{r}^{s's}$$

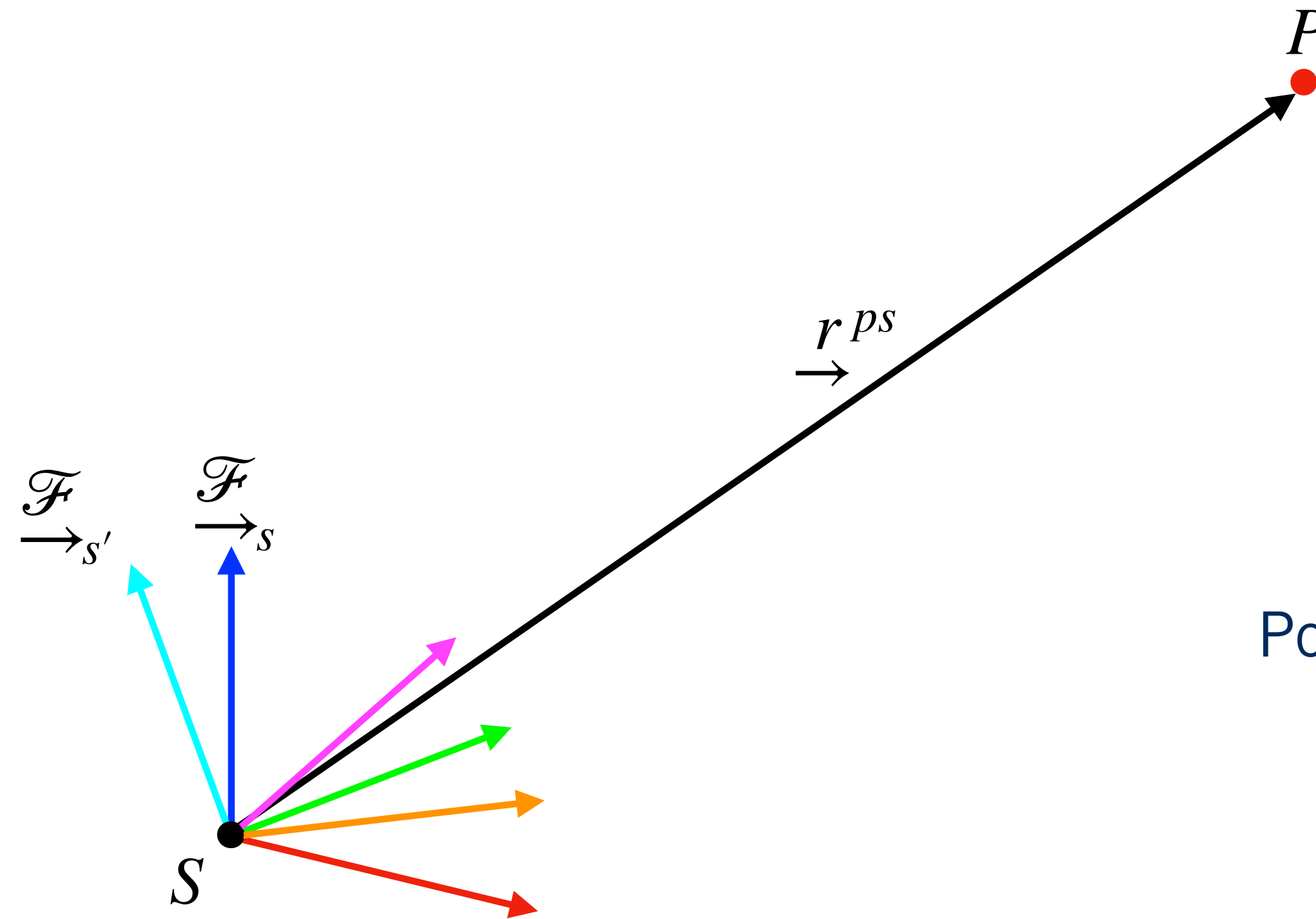
Point coordinates in the translated frame

$$\mathbf{p}_{s'}^{(j)} = \mathbf{p}_s^{(j)} - \mathbf{r}_s^{s's} \quad \text{each point}$$

$$\mathbf{P}_{s'} = \mathbf{P}_s - \mathbf{R}_s^{s's} \quad \text{whole cloud}$$

$$\mathbf{R}_s^{s's} = [\mathbf{r}_s^{s's} \quad \mathbf{r}_s^{s's} \quad \dots \quad \mathbf{r}_s^{s's}]$$

Operations on Point Clouds | Rotation



The *rotation matrix* takes coordinates from frame s to frame s'

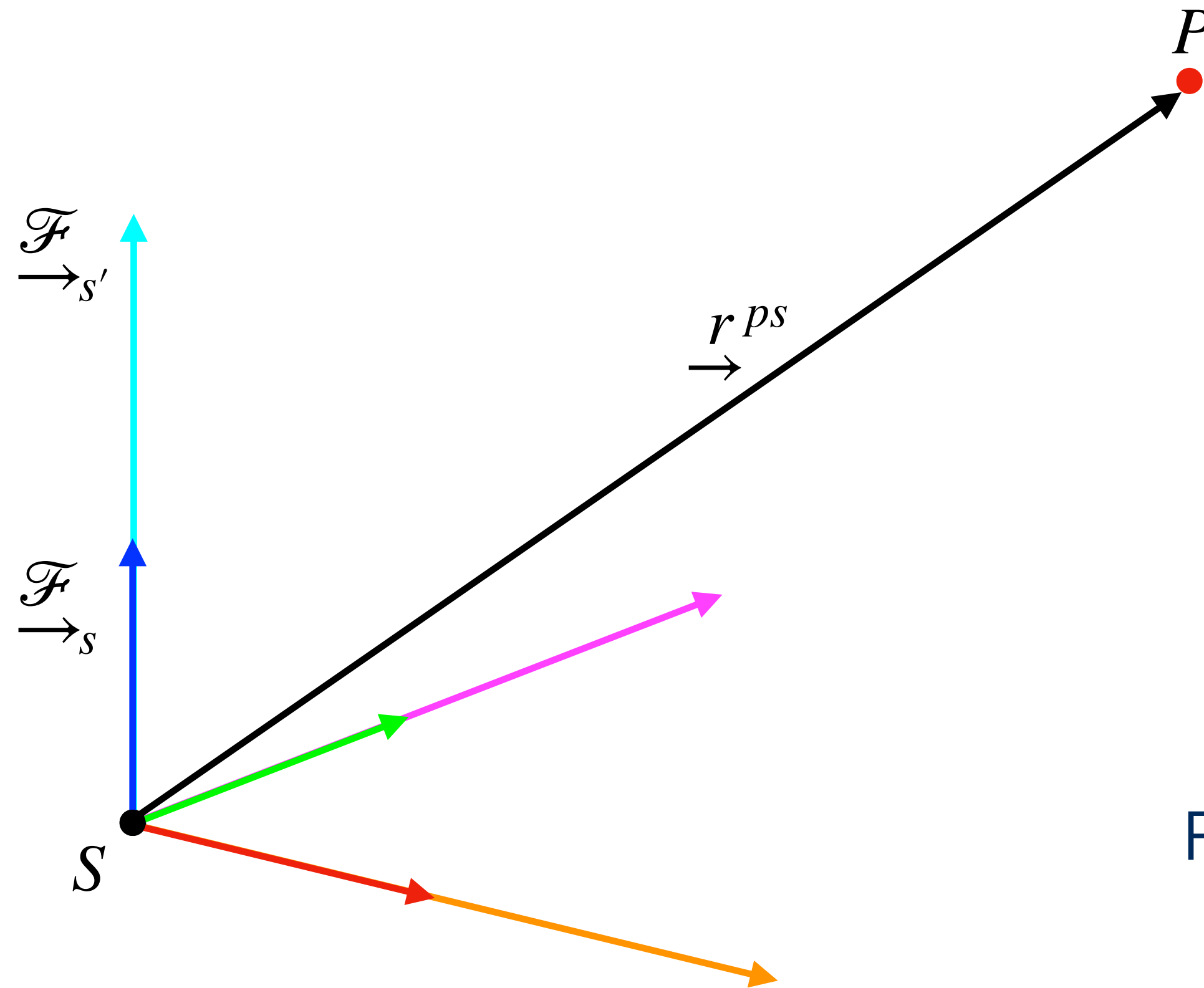
$$\mathbf{r}_{s'} = \mathbf{C}_{s's} \mathbf{r}_s$$

Point coordinates in the rotated frame

$$\mathbf{p}_{s'}^{(j)} = \mathbf{C}_{s's} \mathbf{p}_s^{(j)} \quad \text{each point}$$

$$\mathbf{P}_{s'} = \mathbf{C}_{s's} \mathbf{P}_s \quad \text{whole cloud}$$

Operations on Point Clouds | Scaling



The *scaling matrix* is composed of scaling factors for each basis vector of frame s

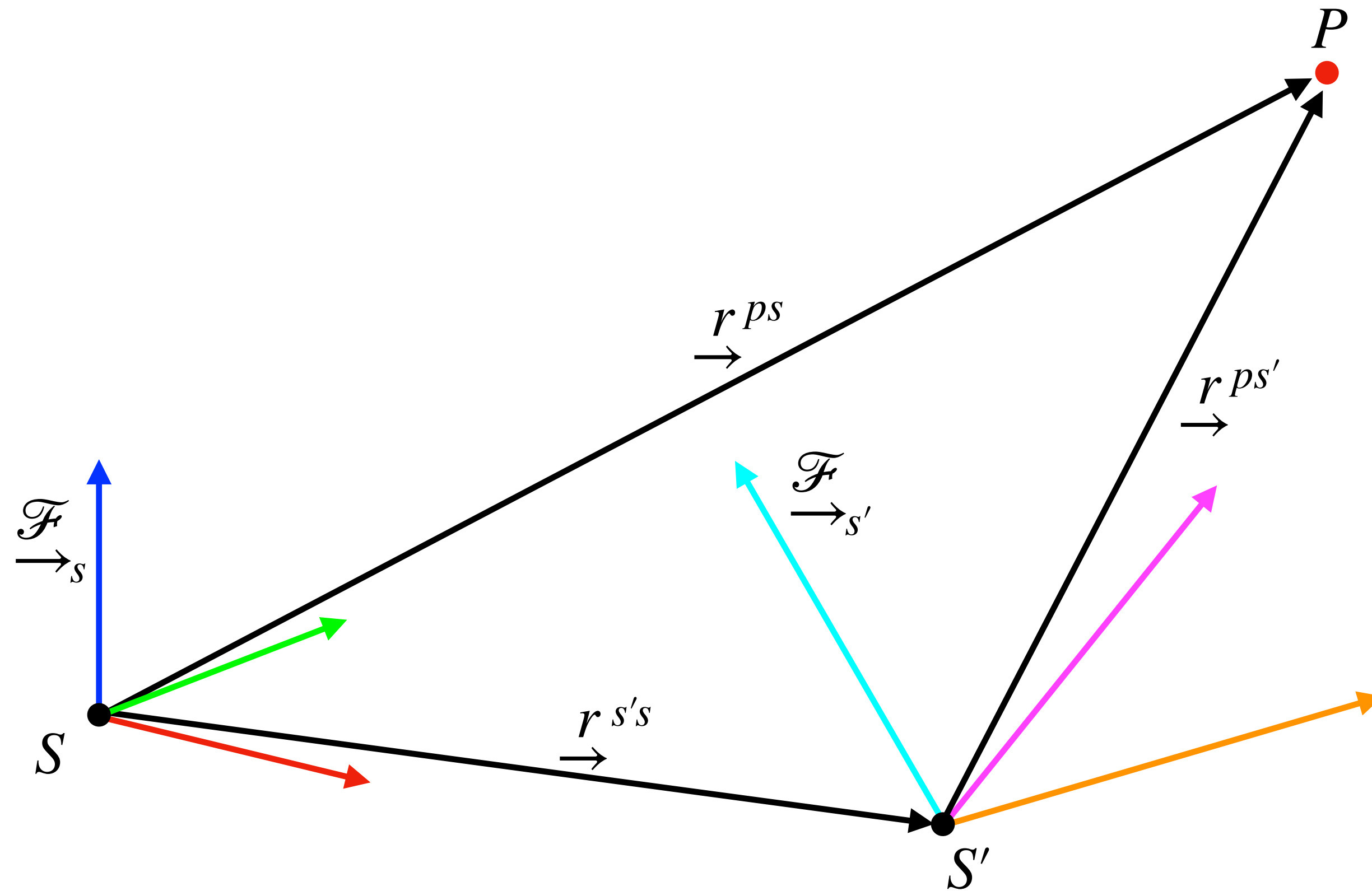
$$\mathbf{r}_{s'} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}}_{\mathbf{S}_{s'/s}} \mathbf{r}_s$$

Point coordinates in the scaled frame

$$\mathbf{p}_{s'}^{(j)} = \mathbf{S}_{s'/s} \mathbf{p}_s^{(j)} \quad \text{each point}$$

$$\mathbf{P}_{s'} = \mathbf{S}_{s'/s} \mathbf{P}_s \quad \text{whole cloud}$$

Operations on Point Clouds | Putting Them All Together



Point coordinates in the
transformed frame

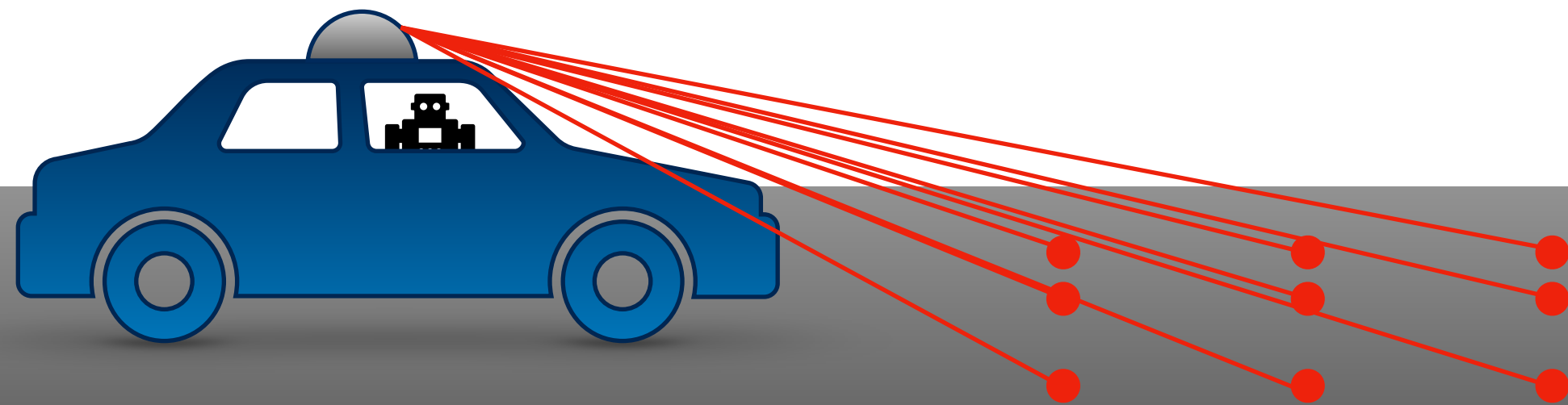
$$\mathbf{p}_{s'}^{(j)} = \mathbf{S}_{s's} \mathbf{C}_{s's} \left(\mathbf{p}_s^{(j)} - \mathbf{r}_s^{s's} \right) \text{ each point}$$

3. Scale 2. Rotate 1. Translate

$$\mathbf{P}_{s'} = \mathbf{S}_{s's} \mathbf{C}_{s's} \left(\mathbf{P}_s - \mathbf{R}_s^{s's} \right) \text{ whole cloud}$$

Finding the Road with 3D Plane Fitting

Where is the road now? Where is it going to be?

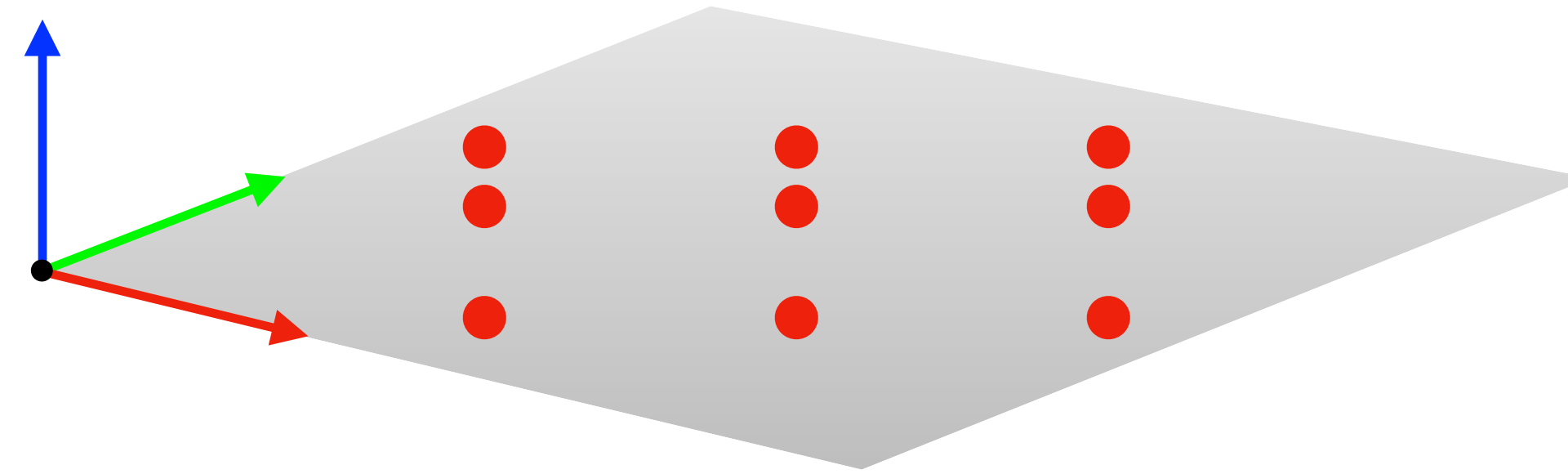


Finding the Road with 3D Plane Fitting

Equation of a plane in 3D:

$$z = a + bx + cy$$

We have measurements of (x, y, z) and we want to determine the parameters (a, b, c) — use **least-squares!**



Measurement error:

$$e_j = \hat{z}_j - z_j \quad \leftarrow \text{observed value}$$
$$= (\hat{a} + \hat{b}x_j + \hat{c}y_j) - z_j \quad j = 1 \dots n$$

predicted

Not account for lidar sensor noise

Finding the Road with 3D Plane Fitting

We can stack all of the measurement errors into matrix form

$$\underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}}_{\mathbf{e}} = \underbrace{\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{\mathbf{x}} - \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}}_{\mathbf{b}}$$

And minimize the squared-error criterion to get the least-squares solution for the parameters

$$\begin{aligned} \hat{\mathbf{x}} &= \operatorname{argmin}_{\mathbf{x}} \mathcal{L}_{\text{LS}}(\mathbf{x}) \\ \mathcal{L}_{\text{LS}}(\mathbf{x}) &= \mathbf{e}^T \mathbf{e} \\ &= (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) \\ &= \mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - \mathbf{x}^T \mathbf{A}^T \mathbf{b} - \mathbf{b}^T \mathbf{Ax} + \mathbf{b}^T \mathbf{b} \end{aligned}$$

Finding the Road with 3D Plane Fitting

$$\begin{aligned}\hat{\mathbf{x}} &= \operatorname{argmin}_{\mathbf{x}} \mathcal{L}_{\text{LS}}(\mathbf{x}) \\ \mathcal{L}_{\text{LS}}(\mathbf{x}) &= \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} - \mathbf{x}^T \mathbf{A}^T \mathbf{b} - \mathbf{b}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{b}\end{aligned}$$

Taking the partial derivative with respect to \mathbf{x} and setting to zero for an optimum gives us the familiar *normal equations*

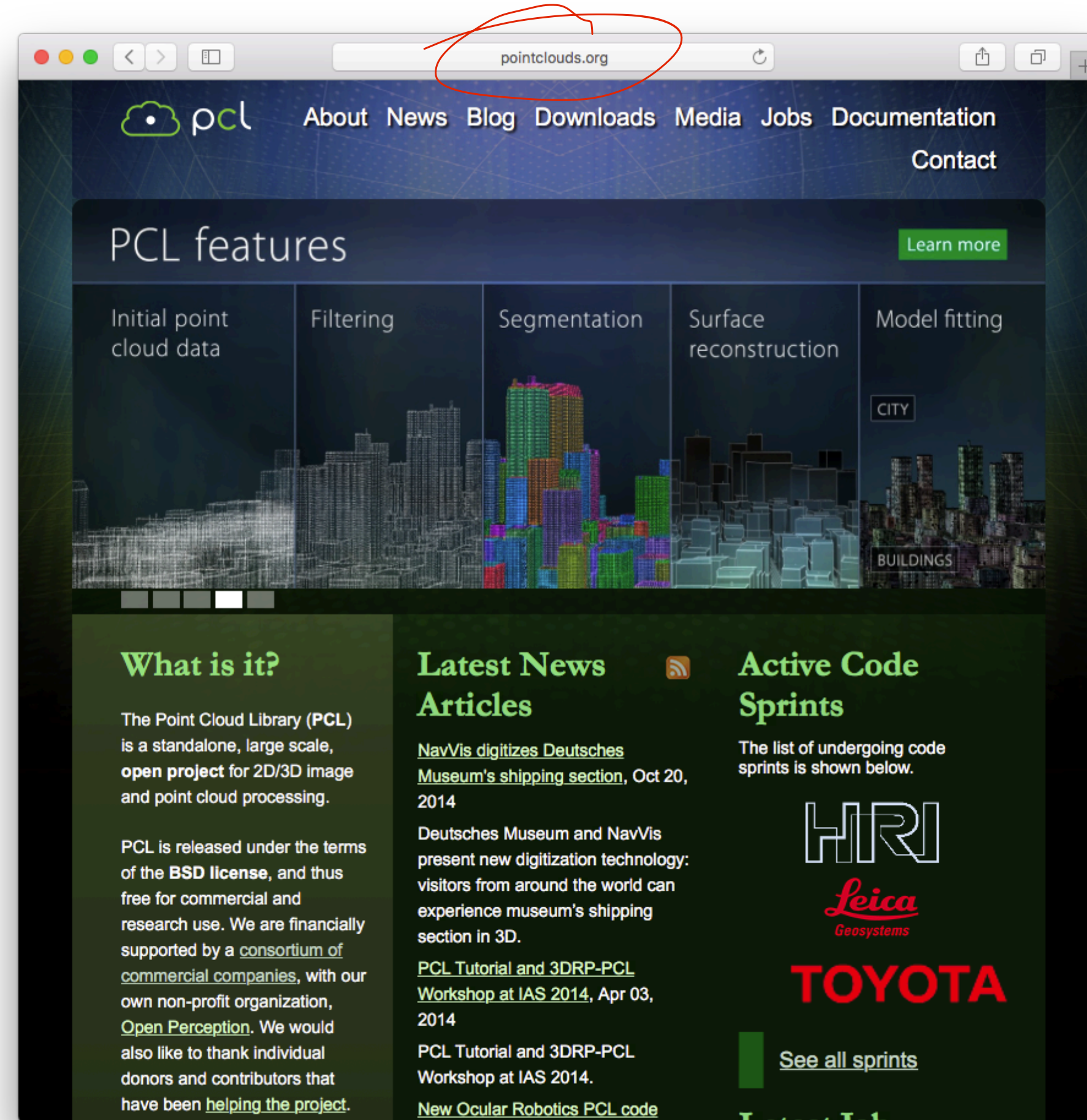
$$\left. \frac{\partial \mathcal{L}_{\text{LS}}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} = 2\mathbf{A}^T \mathbf{A} \hat{\mathbf{x}} - 2\mathbf{A}^T \mathbf{b} = \mathbf{0} \quad \longrightarrow \quad \mathbf{A}^T \mathbf{A} \hat{\mathbf{x}} = \mathbf{A}^T \mathbf{b}$$

We can solve for \mathbf{x} using an efficient numerical solver or by using the *pseudo-inverse*

$$\hat{\mathbf{x}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

The Point Cloud Library (PCL) pcl

- Open-source Point Cloud Library (PCL) has many useful functions for doing basic and advanced operations on point clouds in C++
- Widely used in industry
- Unofficial Python bindings exist



Summary | Point Clouds

- The Cartesian coordinates of all the measurements from a LIDAR scan are stored in a *point cloud*
- Point clouds can be translated, rotated, or scaled
- We can use point clouds for useful self-driving tasks, like fitting a 3D plane to find the road surface
- The Point Cloud Library (PCL) implements many useful tools for working with points clouds in C++