LIDAR

Robotic Perception

LIDAR: Light Detection and Ranging

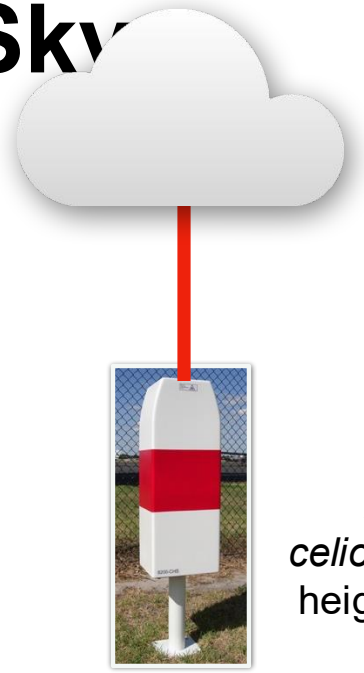
Velodyne

Hokuyo

SICK

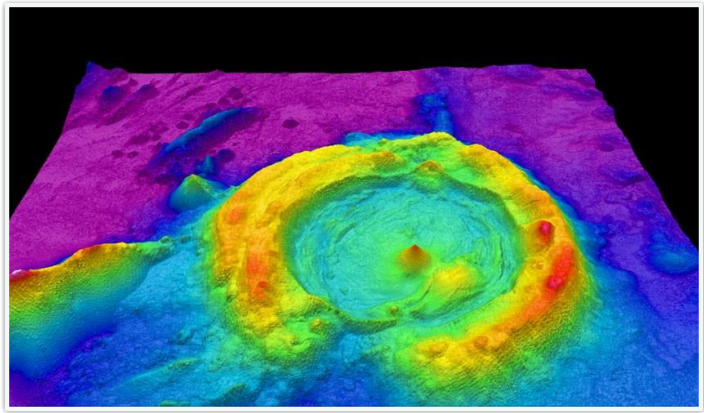
Measuring Earth, Sea and Sky

- LIDAR originated in the 1960s shortly after the invention of the laser
- First used by meteorologists to measure clouds



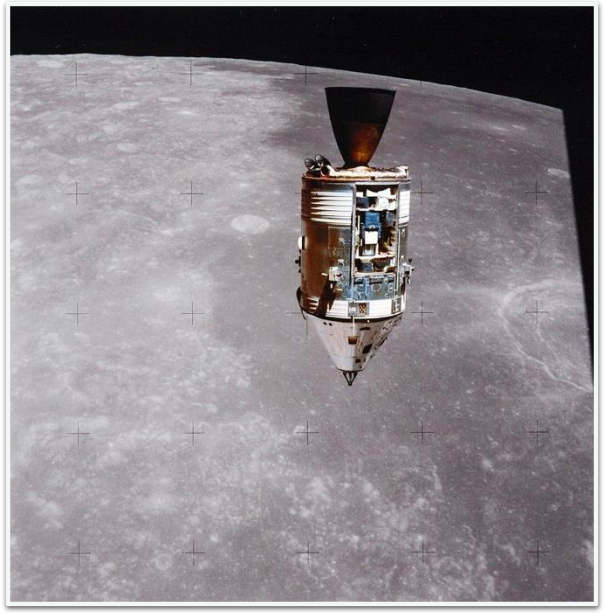
A ground-based *celiometer* measures the height of a cloud ceiling

- Now commonly used for surveying and mapping

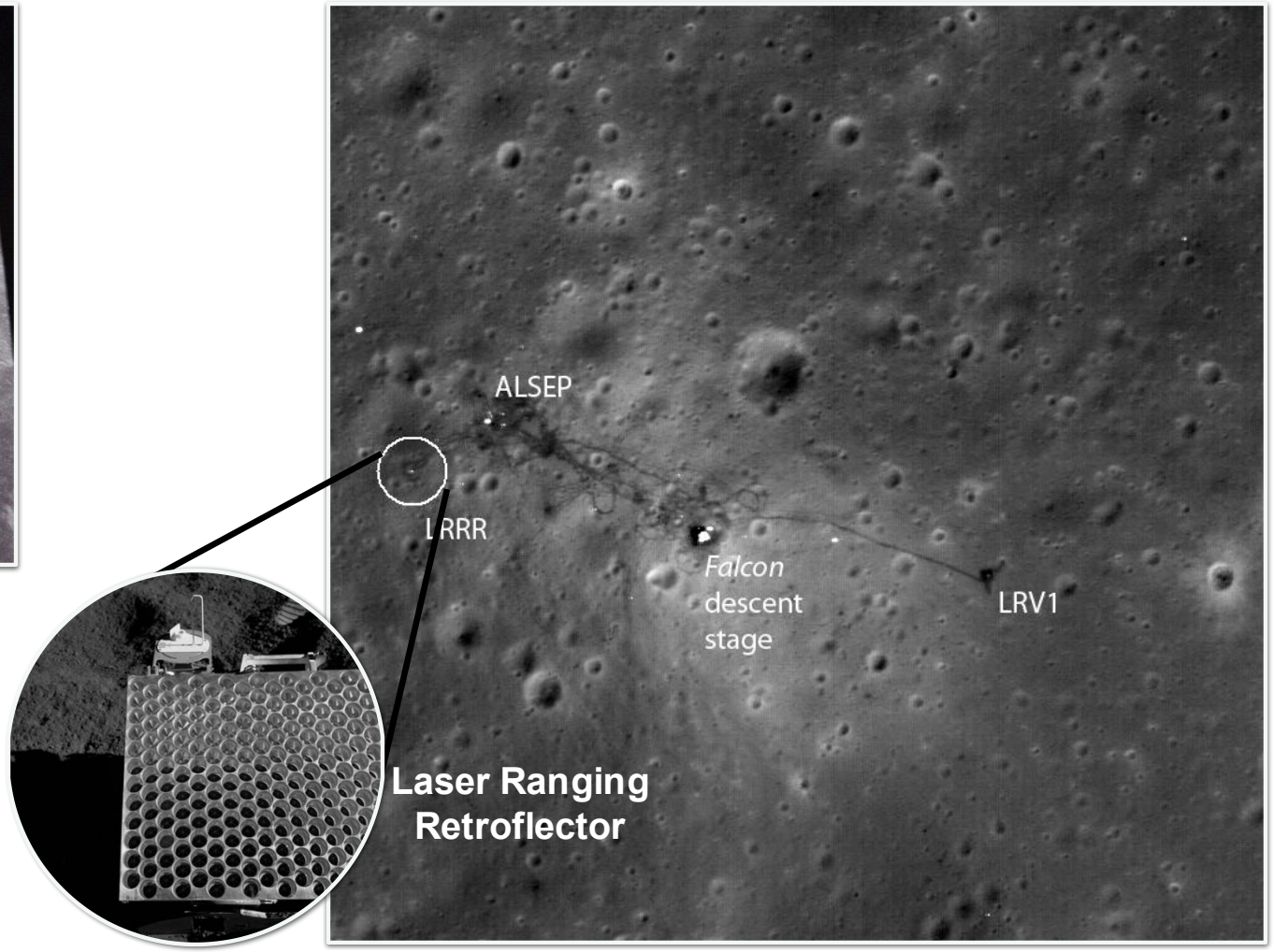


An airborne LIDAR measures the geometry of the seafloor

Apollo 15

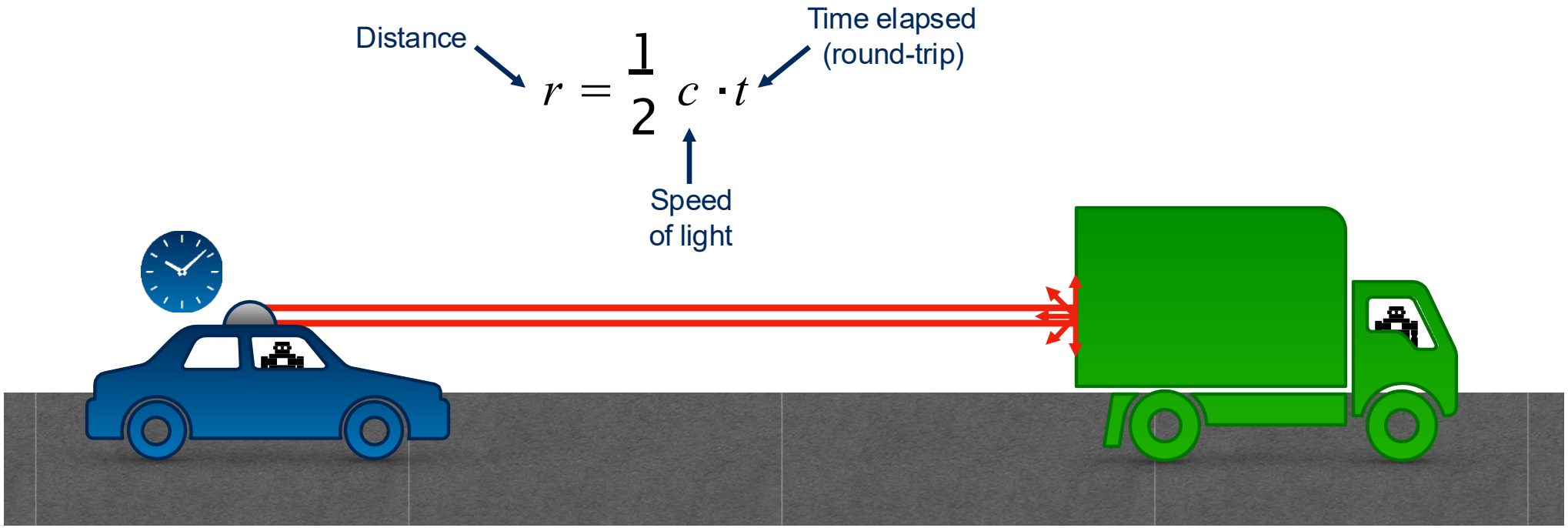


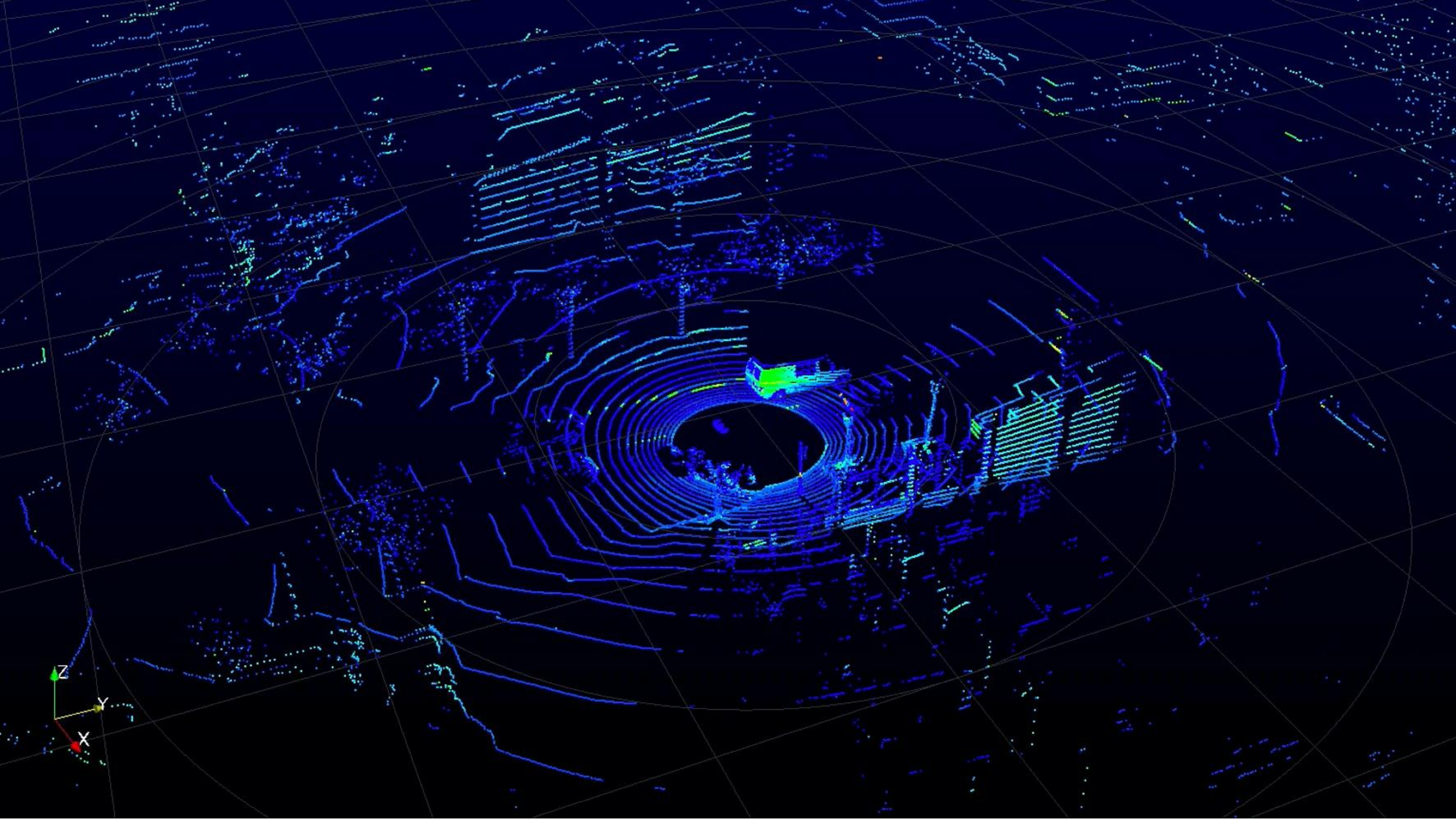
Apollo 15 Command Module



Apollo 15 Landing Site

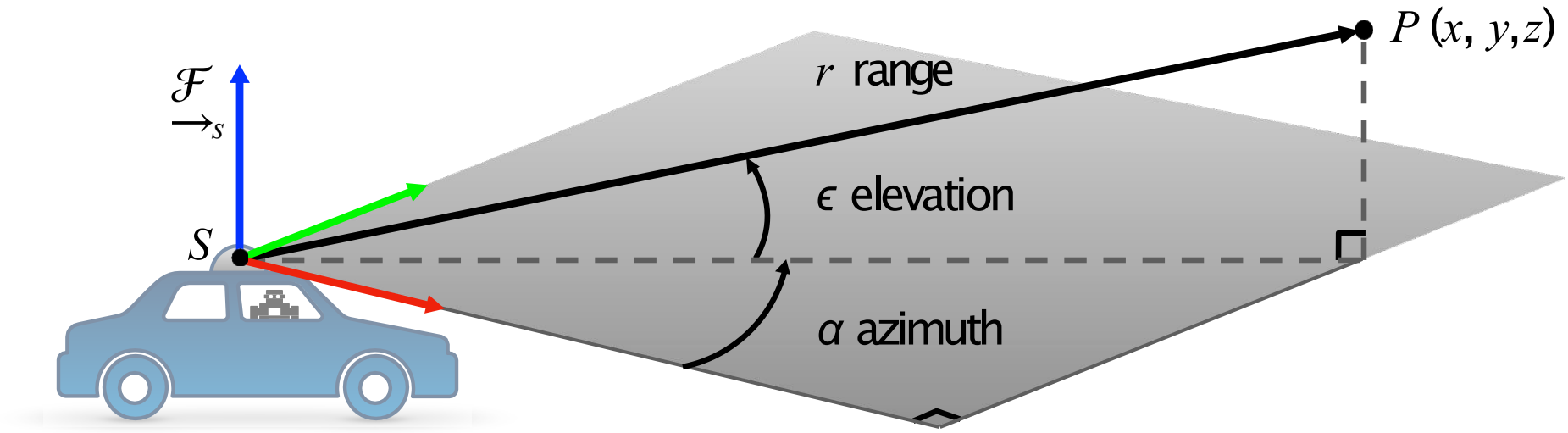
Measuring Distance with Time-of-Flight



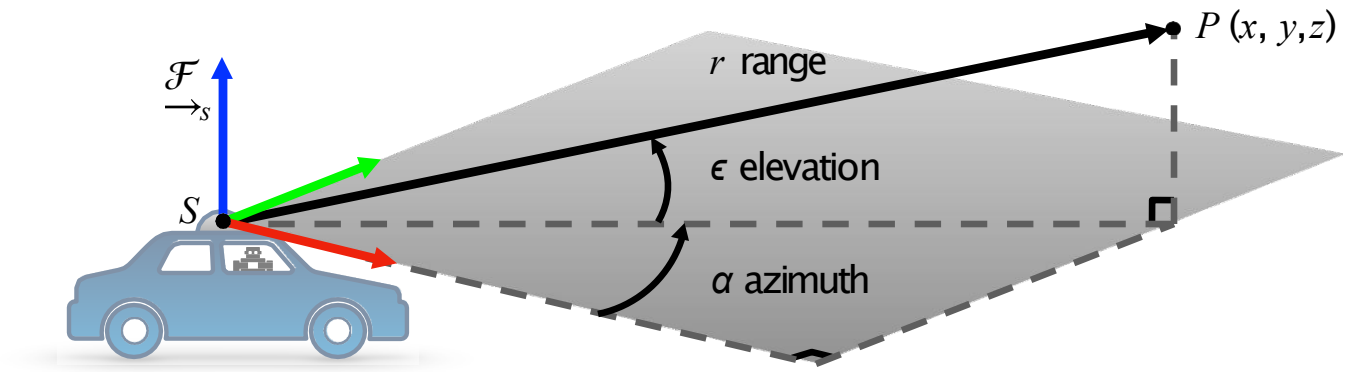

Z
Y
X

Measurement Models for 3D LIDAR Sensors

3D LIDAR sensors report *range*, *azimuth angle* and *elevation angle* (+ return intensity)



Measurement models for 3D LIDAR sensors



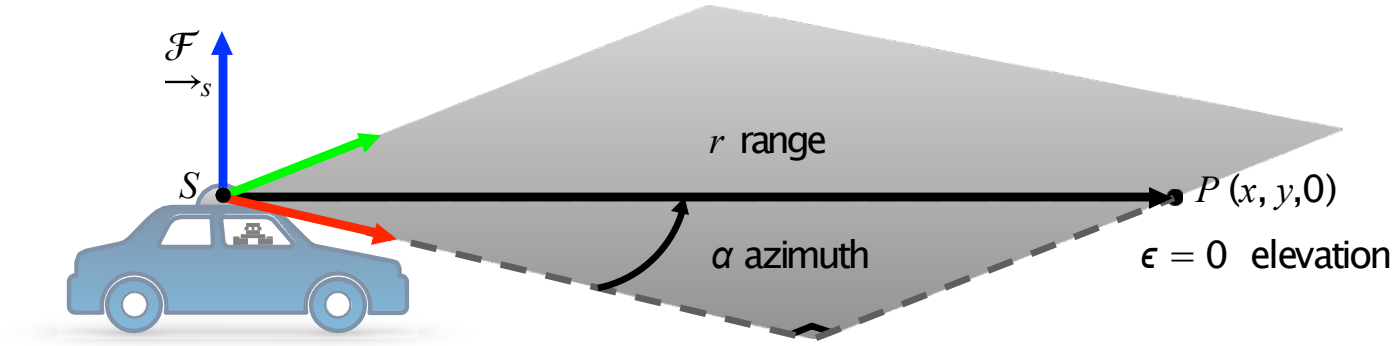
Inverse Sensor Model

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

Forwards Sensor Model

$$\begin{bmatrix} r \\ \alpha \\ \epsilon \end{bmatrix} = \mathbf{h}(x,y,z) = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \tan^{-1}\left(\frac{y}{x}\right) \\ \sin^{-1}\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right) \end{bmatrix}$$

Measurement models for 2D LIDAR sensors



Inverse Sensor Model

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \mathbf{h}^{-1}(r, \alpha, 0) = \begin{bmatrix} r\cos\alpha \\ r\sin\alpha \\ 0 \end{bmatrix}$$

Forwards Sensor Model

$$\begin{bmatrix} r \\ \alpha \\ 0 \end{bmatrix} = \mathbf{h}(x, y, 0) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}\left(\frac{y}{x}\right) \\ 0 \end{bmatrix}$$

Sources of Measurement Noise

- Uncertainty in determining the exact time of arrival of the reflected signal
- Uncertainty in measuring the exact orientation of the mirror
- Interaction with the target (surface absorption, specular reflection, etc.)
- Variation of propagation speed (e.g., through materials)

Forwards Sensor Model (with Noise)

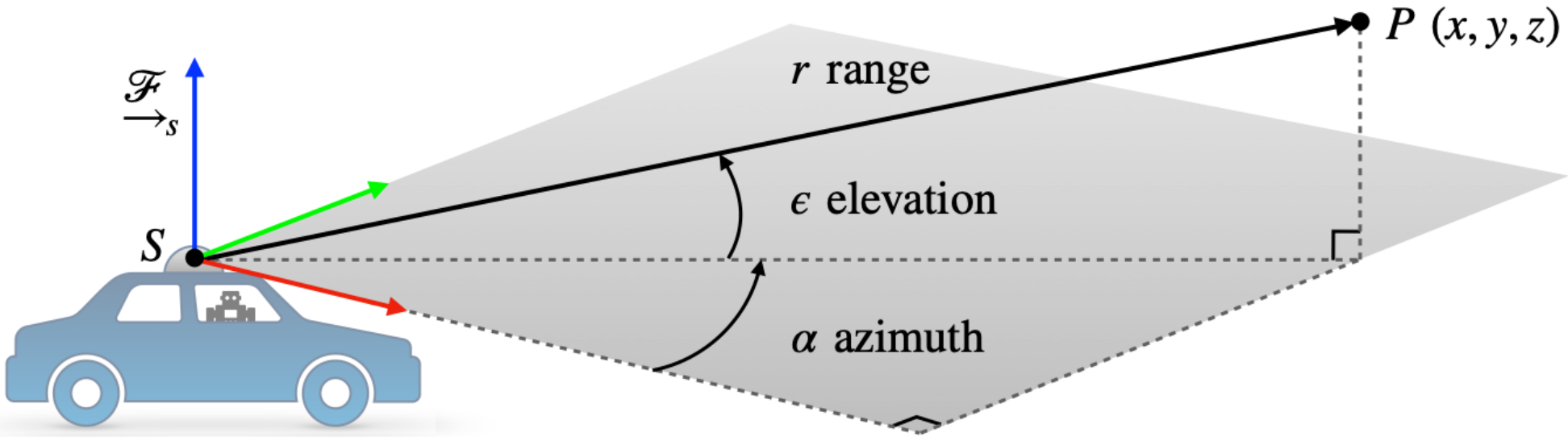
$$\begin{bmatrix} r \\ \alpha \\ \epsilon \end{bmatrix} = \mathbf{h}(x, y, z, \mathbf{v}) = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \tan^{-1}\left(\frac{y}{x}\right) \\ \sin^{-1}\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \end{bmatrix} + \mathbf{v}$$

$$\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$$

Motion Distortion

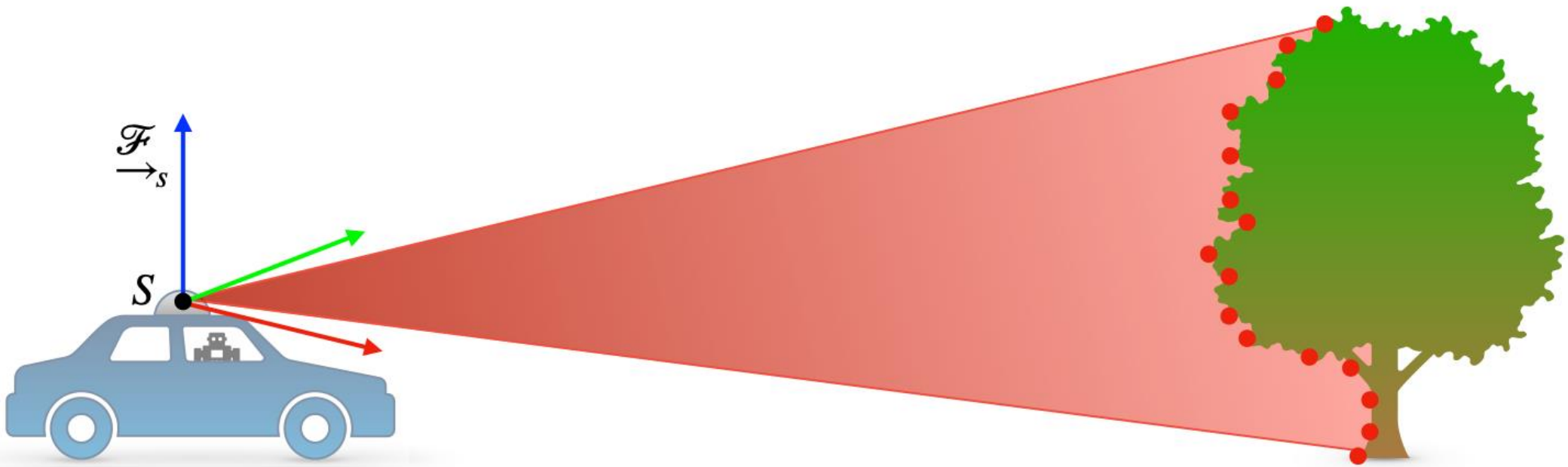
- Typical scan rate for a 3D LIDAR is 5-20 Hz
- For a moving vehicle, each point in a scan is taken from a slightly different place
- Need to account for this if the vehicle is moving quickly, otherwise motion distortion becomes a problem

LIDAR Point Clouds



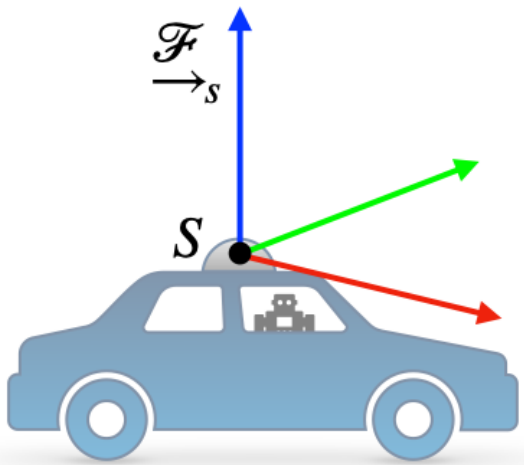
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{h}^{-1}(r, \alpha, \epsilon) = \begin{bmatrix} r\cos\alpha\cos\epsilon \\ r\sin\alpha\cos\epsilon \\ r\sin\epsilon \end{bmatrix}$$

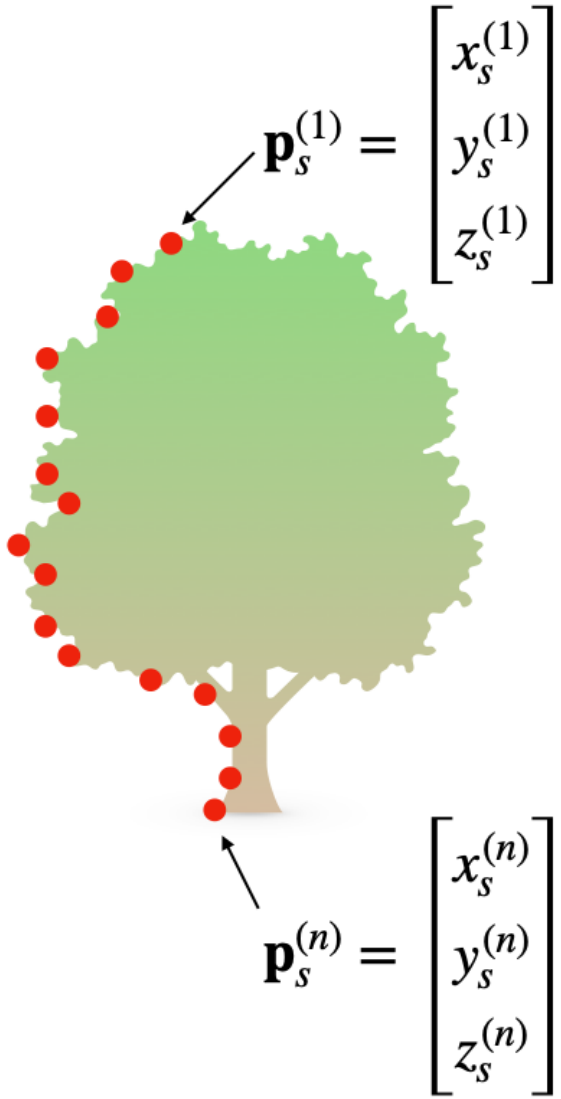
LIDAR Point Clouds



LIDAR Point Clouds | Data Structures

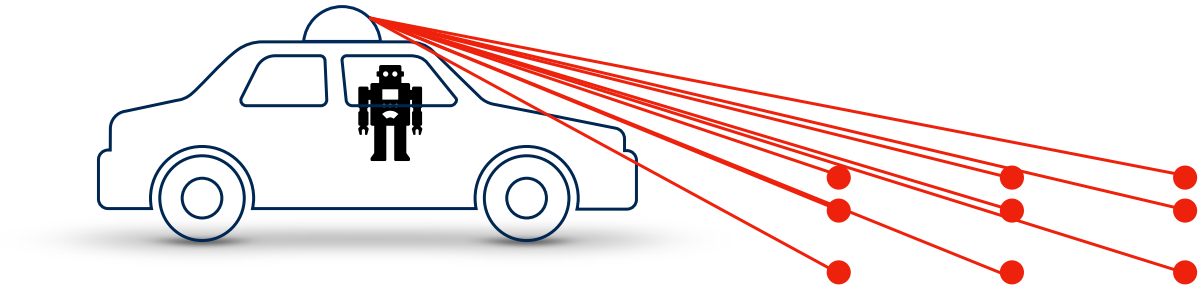
$$\mathbf{P}_s = \begin{bmatrix} \mathbf{p}_s^{(1)} & \mathbf{p}_s^{(2)} & \cdots & \mathbf{p}_s^{(n)} \end{bmatrix} = \begin{bmatrix} x_s^{(1)} & x_s^{(2)} & \cdots & x_s^{(n)} \\ y_s^{(1)} & y_s^{(2)} & \cdots & y_s^{(n)} \\ z_s^{(1)} & z_s^{(2)} & \cdots & z_s^{(n)} \end{bmatrix}$$





Finding the Road with 3D Plane Fitting

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

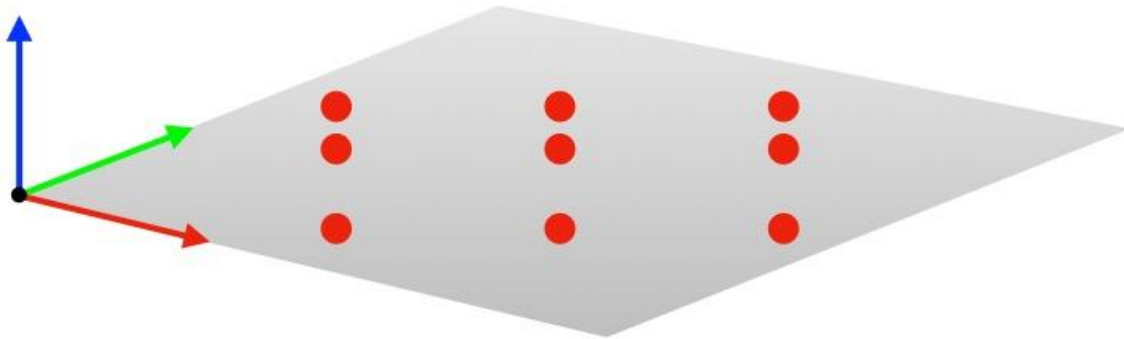


Finding the Road with 3D Plane Fitting

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

Equation of a plane in 3D:

$$z = a + bx + cy$$



Measurement error:

$$e_j = \hat{z}_j - z_j$$

$$= \left(\hat{a} + \hat{b}x_j + \hat{c}y_j\right) - z_j \qquad j = 1...n$$

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

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

Finding the Road with 3D Plane Fitting

$$\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \mathscr{L}_{\mathrm{LS}}(\mathbf{x})$$
$$\mathscr{L}_{\mathrm{LS}}(\mathbf{x}) = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{b}$$

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

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

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

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