Module 4: LIDAR Sensing

内容
- Operating principles of LIDAR sensors
- Basic LIDAR sensor models
- Working with LIDAR point clouds
- Localization via point cloud registration

Lesson 1: Light Detection and Ranging Sensors

内容
- Describe the operating principles of LIDAR sensors
- Use basic LIDAR sensor models in 2D and 3D
- Describe the major sources of measurement errors for LIDAR sensors

単語
- ceilometer: a device for determining the cloud ceiling, esp. by means of a reflected light beam.（雲高計、うんこうけい）
- altimeter: 高度計
- pulse: a pulse of electrical current, light or sound is a temporary increase in its level.

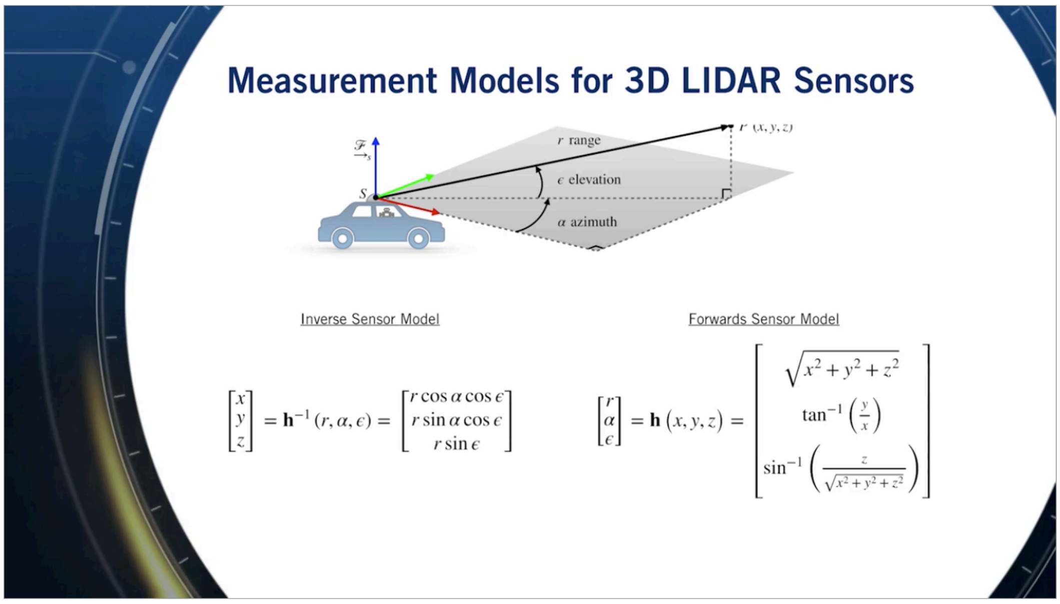
LIDAR: Light Detection and Ranging
- can measure distances to a single point, a 2D slice of the world, or perform a full 3D scan.
- メーカー：Velodyne in California, Hokuyo（北陽電機） in Japan, SICK in Germany

Measuring Earth, Sea and Sky
- LIDAR originated in the 1960s shortly after the invention of the laser.
- First used by meteorologists to measure clouds.
  - not only to measure water clouds, but also to detect volcanic ash and air pollution.
- Airborne LIDAR sensors are commonly used to survey and map the earth's surface for agriculture, geology, military, and other uses.
- Apollo 15: use a laser altimeter to map the surface of the moon.

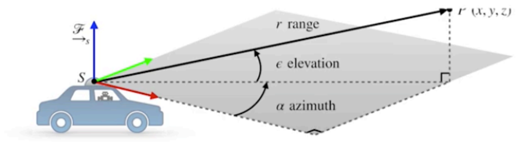
Measuring Distance with Time-of-Flight
1点の場合：
- To build a basic LIDAR in one dimension: a laser, a photodetector, a very precise stopwatch.
- The laser first emits a short pulse of light, usually in the near infrared frequency band along some known ray direction.
  - At the same time, the stopwatch begins counting.
- As long as the surface of the target isn't too polished or shiny, the laser pulse will scatter off the surface in all directions, and some of that reflected light will travel back along the original ray direction.
  - round-trip time.
- Since light travels much faster than cars, it's a good approximation to think of the LIDAR and the target as being effectively stationary during the few nanoseconds that it takes for all of this to happen.
- The photodetector also tells the intensity of the return pulse relative to the intensity of the pulse that was emitted.
  - This intensity information is less commonly used for self-driving, but it provides some extra information about the geometry of the environment and the material the beam is reflecting off of.
  - intensity informationの使い方：create 2D images from LIDAR intensity data and then use computer vision algorithms.
- See in the dark: since LIDAR is its own light source, it provides a way for self-driving cars to see in the dark.

2D&3Dの場合：
- Build a rotating mirror into the LIDAR that directs the emitted pulses along different directions.

- As the mirror rotates, measure distances at points in a 2D slice around the sensor.
- If then add an up and down nodding motion to the mirror along with the rotation, can use the same principle to create a scan in 3D.
- Velodyne LIDARのやり方：sensors create multiple 2D scan lines from a series of individual lasers spaced at fixed angular intervals, which effectively paint the world with horizontal stripes of laser light.
    - The black hole in the middle is a blind spot where the sensor itself is located, and the concentric circles spreading outward from there are the individual scan lines produced by the rotating Velodyne sensor.
    - Each point in the scan is colored by the intensity of the return signal.



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 3D LIDAR Sensors
- 3D LIDAR sensors report range, azimuth angle and elevation angle (+ return intensity)
    - azimuth angle, measured counterclockwise from the sensor's x-axis.
    - The azimuth and elevation angles are measured using encoders that tell the orientation of the mirror.
- For Velodyne LIDARs, the elevation angle is fixed for a given scan line.
- Inverse Sensor Model & Forwards Sensor Model（上記の図を参考）

Measurement Models for 2D LIDAR Sensors
- Inverse Sensor Model: $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = h^{-1}(r, \alpha, 0) = \begin{bmatrix} r cos\alpha \\ r sin\alpha \\ 0 \end{bmatrix}$
- Forward Sensor Model: $\begin{bmatrix} r \\ \alpha \\ 0 \end{bmatrix} = h(x, y, 0) = \begin{bmatrix} \sqrt{x^2+y^2} \\ tan^{-1}\left(\frac{y}{x}\right) \\ 0 \end{bmatrix}$

Sources of Measurement Noise
- Uncertainty in determining the exact time of arrival of the reflected signal.
    - 原因：the stopwatch has a limited resolution.
- Uncertainty in measuring the exact orientation of the mirror. 同じresolutionの原因。

- Interaction with the target (surface absorption（例えばcompletely black）, specular reflection（つまりreturn pulseが全然なし）, etc.)
  - In both of these cases, the LIDAR will typically report a maximum range error, which could mean there is empty space along the beam direction, or that the pulse encountered a highly abortive or highly reflective surface. 危ない。
- Variation of propagation speed (e.g., through materials)
- Forwards Sensor Model (with Noise): $\begin{bmatrix} r \\ \alpha \\ \epsilon \end{bmatrix} = h(x,y,z,v) = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ tan^{-1}\left(\frac{y}{x}\right) \\ sin^{-1}\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right) \end{bmatrix} + v,$

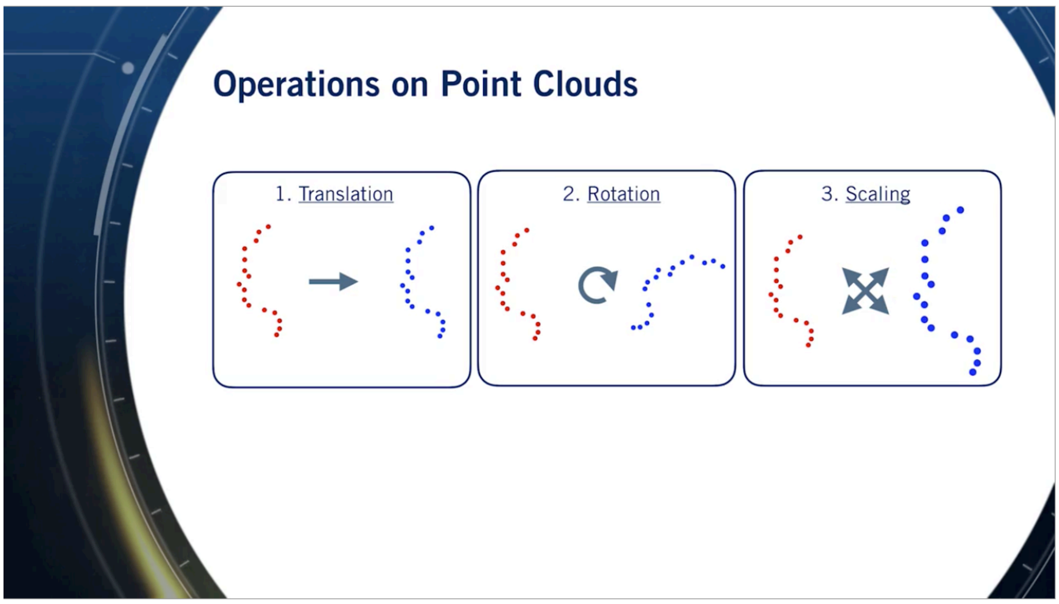
  $v \sim \mathcal{N}(0,R)$
  - an empirically determined or manually tuned covariance.

Motion Distortion
- Typical scan rate for a 3D LIDAR is 5-20Hz when scanning objects at distances of 10 to 100m.
- 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.
- Although the car is unlikely to be moving at an appreciable fraction of the speed of light, it is often going to be moving at an appreciable fraction of the rotation speed of the sensor itself.
- Correcting this motion distortion usually requires an accurate motion model for the vehicle provided by GPS and INS for example.

Lesson 2: LIDAR Sensor Models and Point Clouds

内容
- 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

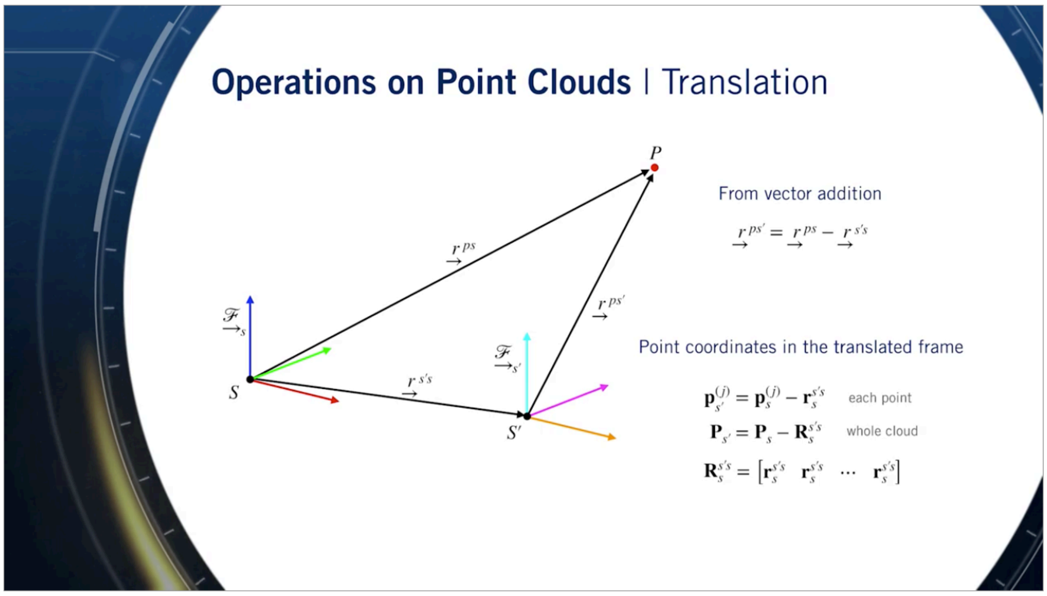


- For some LIDAR setups, it's not uncommon for these point clouds to contain upwards of a million points or more.

LIDAR Point Clouds | Data Structures

- assign an index to each of the points: $P_s = \left[p_s^{(1)} \quad p_s^{(2)} \ldots p_s^{(n)}\right] = \begin{bmatrix} x_s^{(1)} & x_s^{(2)} & \ldots & x_s^{(n)} \\ y_s^{(1)} & y_s^{(2)} & \ldots & y_s^{(n)} \\ z_s^{(1)} & z_s^{(2)} & \ldots & z_s^{(n)} \end{bmatrix}$

Operations on Point Clouds
- 考え方：Objects in the world mostly stay put while the reference frame attached to the vehicle moves and observes the world from different perspectives.



Operations on Point Clouds | Translation

P
$\underset{\rightarrow}{r}^{ps}$
$\underset{\rightarrow}{r}^{ps'}$
$\underset{\rightarrow}{\mathcal{F}}_s$
$\underset{\rightarrow}{\mathcal{F}}_{s'}$
$\underset{\rightarrow}{r}^{s's}$
S
S'

From vector addition

$$\underset{\rightarrow}{r}^{ps'} = \underset{\rightarrow}{r}^{ps} - \underset{\rightarrow}{r}^{s's}$$

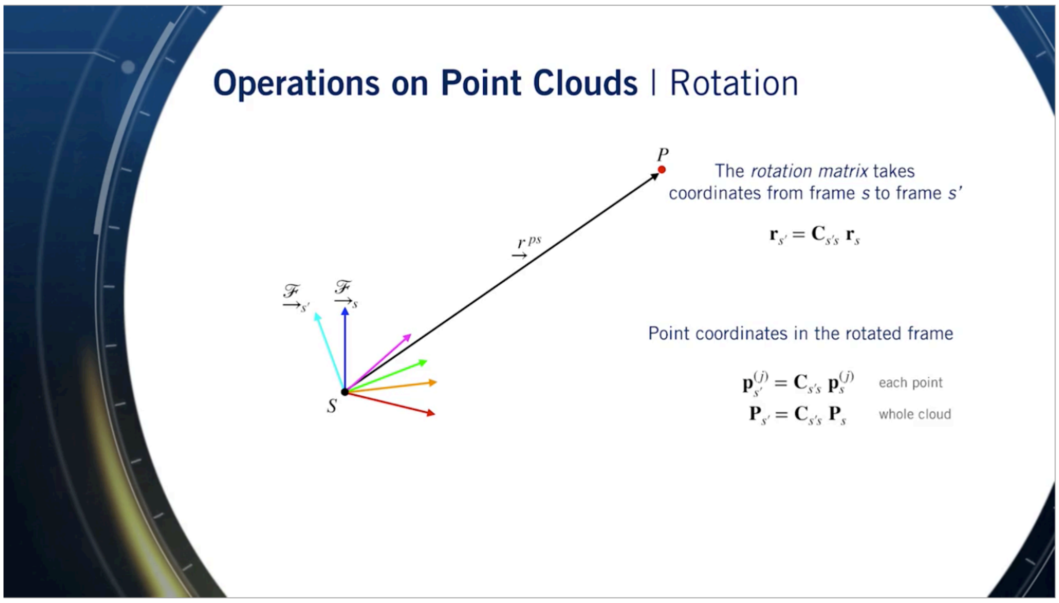
Point coordinates in the translated frame

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

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

$$\mathbf{R}_s^{s's} = \left[\mathbf{r}_s^{s's} \quad \mathbf{r}_s^{s's} \quad \cdots \quad \mathbf{r}_s^{s's}\right]$$

Operations on Point Clouds | Translation（上記のスライドを参考）



Operations on Point Clouds | Rotation

P
$\underset{\rightarrow}{r}^{ps}$
$\underset{\rightarrow}{\mathcal{F}}_{s'}$
$\underset{\rightarrow}{\mathcal{F}}_s$
S

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

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

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

- Depending on the language or linear algebra library, probably not need to build this R matrix explicitly.
    - In Python, the NumPy library is smart enough to repeat the frame-to-frame translation implicitly using broadcasting semantics.

Operations on Point Clouds | Rotation

- What does change in this case is actually the set of basis vectors we use to express the coordinates of the vector S to P.

Operations on Point Clouds | Scaling

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

$$r_{s'} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}}_{S_{s's}} r_s$$

- Often, but not always, these scaling factors are the same.
- Point coordinates in the scaled frame: $p_{s'}^{(j)} = S_{s's} p_s^{(j)}$ (each point), $P_{s'} = S_{s's} P_s$ (whole cloud).

Operations on Point Clouds | Putting Them All Together

- 使用例：estimating the translation and rotation that best aligns to point clouds so that we can estimate the motion of the self-driving car.
- $p_{s'}^{(j)} = S_{s's} C_{s's} \left(p_s^{(j)} - r_s^{s's} \right)$
- $P_{s'} = S_{s's} C_{s's} \left(P_s - R_s^{s's} \right)$

Finding the Road with 3D Plane Fitting

- 使用例：figuring out where the road surface is and predicting where it's going to be as the car continues driving.
- Equation of a plane in 3D: $z = a + bx + cy$
- Given measurements of (x, y, z), determine the parameters (a, b, c) - use least-squares.
- Measurement error: $\begin{aligned} e_j &= \hat{z}_j - z_j \\ &= \left(\hat{a} + \hat{b}x_j + \hat{c}y_j \right) - z_j \end{aligned}, j = 1 \ldots n$
- Stack all of the measurement errors into matrix form: $\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} - \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$,

  $e = Ax - b$
- Minimize the squared-error criterion to get the least-squares solution for the parameters:

$$\hat{x} = argmin_x \mathscr{L}_{LS}(x)$$

$$\begin{aligned} \mathscr{L}_{LS}(x) &= e^T e \\ &= (Ax - b)^T (Ax - b) \\ &= x^T A^T A x - x^T A^T b - b^T A x + b^T b \end{aligned}$$

- $\frac{\partial \mathscr{L}_{LS}(x)}{\partial x} \Big|_{x=\hat{x}} = 2A^T A \hat{x} - 2A^T b = 0$
    - $A^T A \hat{x} = A^T b$
    - We can solve this linear system using an efficient numerical solver like Python NumPy's solve function.

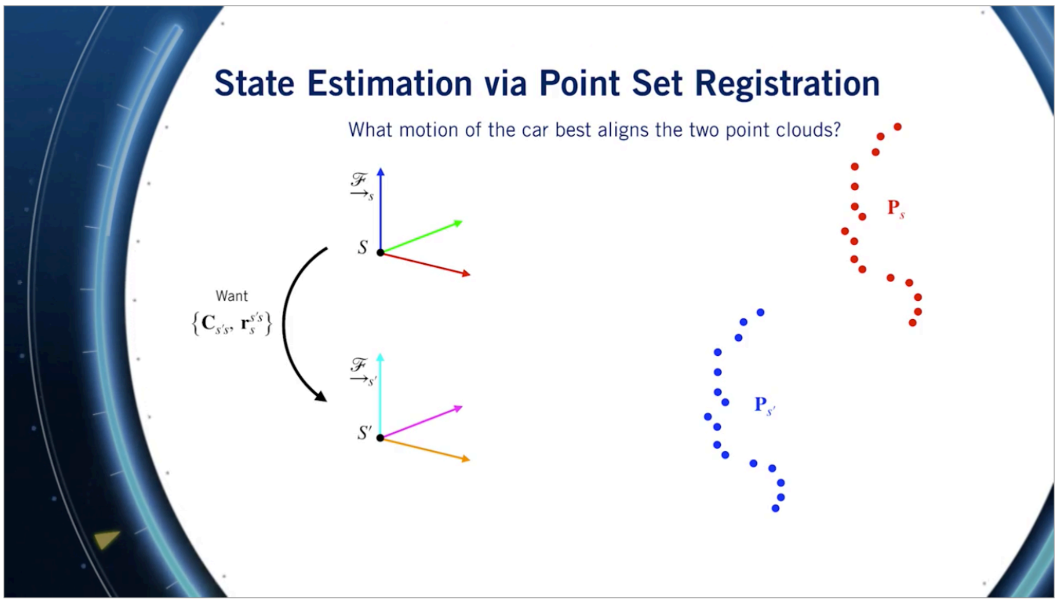
  - Or by using the pseudo-inverse: $\hat{x} = (A^TA)^{-1}A^Tb$
- 注意点：まだnot account for sensor noise in x, y, z measurements.
  - you could even think about including the road parameters in the Kalman filter to estimate them on the fly as the sensor data comes in.
  - the best solution will depend on how much you trust your LIDAR data and how much thought you want to give to uncertainty in the road surface.

The Point Cloud Library (PCL) http://pointclouds.org
- 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.
- One of the most useful algorithms in PCL is called the iterative closest point algorithm, or ICP, which is a common method for estimating the motion of a self-driving car using two LIDAR point clouds.

Lesson 3: Pose Estimation from LIDAR Data

単語
- unitary matrix: ユニタリ行列, a square matrix that is the inverse of its Hermitian conjugate.
- Hermitian conjugate: エルミート共軛（きょうやく）~~随伴（ずいはん）行列~~, a matrix that is the transpose of the matrix of the complex conjugates of the entries of a given matrix.
- complex conjugate: 複素（ふくそ）共軛, the complex number whose imaginary part is the negative of that of a given complex number, the real parts of both numbers being equal.

point set registration problem, one of the most important problems in computer vision and pattern recognition.
内容
- Describe the point set registration problem and how it can be used for state estimation.
- Describe and implement the Iterative Closest Point (ICP) algorithm.
- Understand some common pitfalls of the ICP algorithm.


State Estimation via Point Set Registration

What motion of the car best aligns the two point clouds?
$\underrightarrow{\mathcal{F}}_s$
S
Want
$\{\mathbf{C}_{s's}, \mathbf{r}_s^{s's}\}$
$\underrightarrow{\mathcal{F}}_{s'}$
S'
$\mathbf{P}_s$
$\mathbf{P}_{s'}$

State Estimation via Point Set Registration point set registration problemの定義：given 2 point clouds in two different coordinate frames, and the knowledge that they correspond to or contain the same object in the world, how shall we align them to determine how the sensor must have moved between the two scans?

定義２：figure out the optimal translation and the optimal rotation between the two sensor reference frames that minimizes the distance between the 2 point clouds.

課題：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.
  - the pairs of points that truly correspond to each other will be the ones that are closest to each other in a Euclidean sense.
- Heuristic: For each point, the best candidate for a corresponding point is the point that is closest to it right now.
  - 課題：because the vehicle is moving, it's almost never the case that a laser beam will hit exactly the same surface point twice.
- Procedure
  1. Get an initial guess for the transformation $\{\check{C}_{s's}, \check{r}_s^{s's}\}$
     - 理由：there's no guarantee that the closest point heuristic will actually converge to the best solution. つまりこのinitial guessはclosest point heuristicではない、全く関係ない。
     - It's very easy to get stuck in a local minimum in these kinds of iterative optimization schemes.
     - One of the most common sources for robotics applications like self-driving is a motion model, which could be supplied by an IMU or by wheel odometry, or even a constant velocity / zero velocity model.
     - How complex the motion model needs to be to give a good initial guess really depends on how smoothly the car is moving.
     - If the car is moving slowly relative to the scanning rate of the LIDAR sensor, one may even use the last known pose as the initial guess.
  2. Use the initial guess to transform the coordinates of the points in one cloud into the reference frame of the other, and then match each point the second cloud to the closest point in the first cloud.
     1. Associate each point in $P_{s'}$ with the nearest point in P_s.
     2. There's nothing stopping two points in one cloud from being associated with the same point in the other cloud.
     3. ステップ２の目標はただcorresponding pointsを決めること。
  3. Find the transformation that minimizes the sum of squared distances between the corresponding points.
     1. Solve for the optimal transformation $\{\hat{C}_{s's}, \hat{r}_s^{s's}\}$
  4. Repeat the process using the translation and rotation just solved for as new initial guess.
     1. Repeat until convergence.

ICP | Solving for the Optimal Transformation

$$\{\hat{C}_{s's}, \hat{r}_s^{s's}\} = argmin_{\{C_{s's}, r_s^{s's}\}} \mathscr{L}_{LS}(C_{s's}, r_s^{s's})$$

- Least Squares: $$\mathscr{L}_{LS}(C_{s's}, r_s^{s's}) = \sum_{j=1}^{n} \|C_{s's}\left(p_s^{(j)} - r_s^{s's}\right) - p_{s'}^{(j)}\|_2^2$$
- Careful: Rotations need special treatment because they don't behave like vectors.
  - 理由：If you add two rotation matrices together, the result is not necessarily a valid rotation matrix.
  - 3D rotations belong to something called the special orthogonal group or SO(3).
- Procedures:

1. Compute the centroids of each point cloud: $\mu_s = \frac{1}{n}\sum_{j=1}^{n} p_s^{(j)}, \mu_{s'} = \frac{1}{n}\sum_{j=1}^{n} p_{s'}^{(j)}$

2. Compute a matrix capturing the spread of the two point clouds:
   $$W_{s's} = \frac{1}{n}\sum_{j=1}^{n}\left(p_s^{(j)} - \mu_s\right)\left(p_{s'}^{(j)} - \mu_{s'}\right)^T$$
3. Use the singular value decomposition of the matrix to get the optimal rotation: $USV^T = W_{s's}$,
   $$\hat{C}_{s's} = U\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & detUdetV\end{bmatrix}V^T$$
   1. $U^TU = 1$, (rotation), unitary matrix.
   2. $V^TV = 1$, (rotation), unitary matrix.
   3. S: (scaling matrix) a diagonal matrix, whose non-zero entries are called the singular values (scaling) of the original matrix (W).
   4. We don't want any scaling in a rotation estimate, so replace the S matrix with something like the identity matrix to remove the scaling.
   5. Rotation and Reflection: $Rot(\theta) = \begin{bmatrix}cos\theta & -sin\theta\\ sin\theta & cos\theta\end{bmatrix}$, $Ref(\theta) = \begin{bmatrix}cos2\theta & sin2\theta\\ sin2\theta & -cos2\theta\end{bmatrix}$.
   6. In some cases, the SVD will give rotation matrices that are not proper rotations.
      1. That is, they also have a reflection about the axis of rotation (?). “reflection about the axis of rotation”はまだ分かっていない。
   7. $detUdetV$: This number will be either $+1$ or -1, and will cancel out any reflection.
4. Use the optimal rotation to get the optimal translation by aligning the centroids: $\hat{r}_s^{s's} = \mu_s - \hat{C}_{s's}^T\mu_{s'}$.
5. How confident should we be in the output of the ICP algorithm?

ICP | Estimating Uncertainty

Obtain an estimate of the covariance matrix of the ICP solution using this formula:

$$cov(\hat{x}) \simeq \left[\left(\frac{\partial^2\mathscr{L}}{\partial x^2}\right)^{-1}\frac{\partial^2\mathscr{L}}{\partial z\partial x}cov(z)\frac{\partial^2\mathscr{L}^T}{\partial z\partial x}\left(\frac{\partial^2\mathscr{L}}{\partial x^2}\right)^{-1}\right]_{x=\hat{x}}$$

- This expression tells how the covariance of the estimated motion parameters is related to the covariance of the measurements in the two point clouds using certain second-order derivatives of the least squares cost function.
- While its expression is accurate and fast to compute, it's a little tricky to derive these second-order derivatives when there's a constraint quantity like a rotation matrix in the mix. cost functionにrotation matrixが入っているから、second-order derivativeの計算は不便。
  - So generally hand tuning the ICP covariance matrix to give us acceptable results.

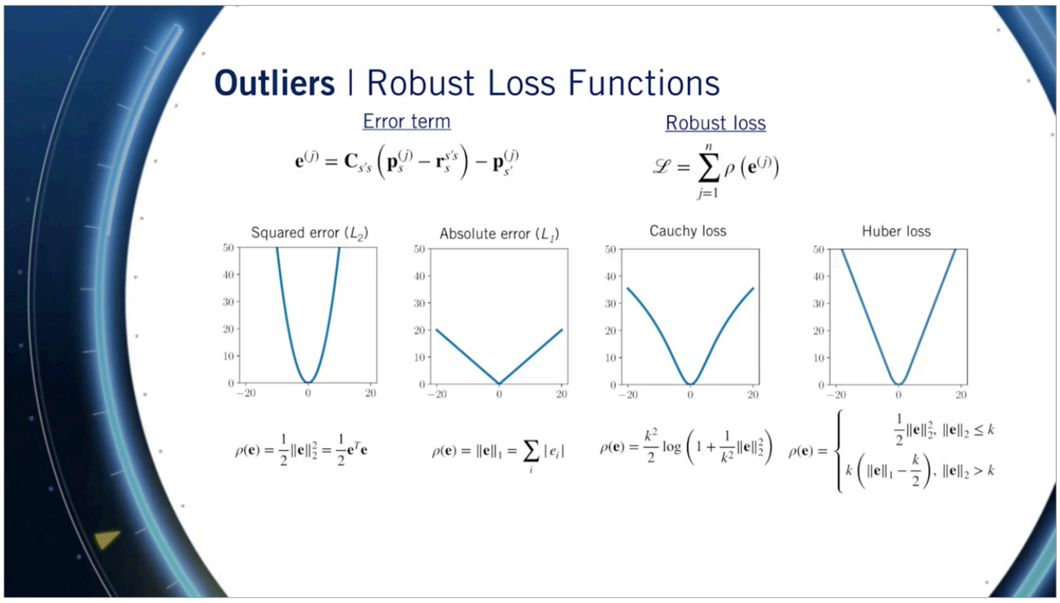
ICP | Variants

- Point-to-point ICP （上記のアルゴ） minimizes the Euclidean distance between each point in $P_{s'}$ and the nearest point in P_s.
- Point-to-plane ICP minimizes the perpendicular distance between each point in $P_{s'}$ and the nearest plane in P_s.
  - This tends to work well in structured environments（つまり物の表面はほぼ平面? ） like cities or indoors.
  - fit a number of planes to the first point cloud.
  - These planes could represent things like walls or road surfaces.
  - The challenging part is actually figuring out where all the planes are.
  - Point-to-plane ICPのアルゴをチェックしよう！

Outliers | Objects in Motion

- シーン：

        - Both vehicles are traveling at the same speed.
        - What motion of the car best aligns the two point clouds?
            - The answer is that we haven’t moved at all.
            - Of course, we did move, just not relative to the vehicle directly ahead of us.
- ICP alone is not always enough!
    - Naively using ICP in the presence of moving objects will tend to pull motion estimates away from the true motion.
- Be careful to exclude or mitigate the effects of outlying points that violate assumptions of a stationary world.
    - one way is fusing ICP motion estimates with GPS and INS estimates.
    - another way is identifying and ignoring moving objects, which could be done with computer vision techniques
    - another way is to choose a different loss function that is less sensitive to large errors induced by outliers than the standard squared error loss.
        - Loss functions which have this property are called robust loss functions, or robust cost functions.
        - Generalizing the least squares loss function so that the contribution of each error is not simply the square of its magnitude, but rather some other function ρ.


Outliers | Robust Loss Functions

Error term

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

Robust loss

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

Squared error (L_2)

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

Absolute error (L_1)

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

Cauchy loss

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

Huber loss

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

Outliers | Robust Loss Functions
- Error term: $e^{(j)} = C_{s's}\left(p_s^{(j)} - r_s^{s's}\right) - p_{s'}^{(j)}$.
- Robust loss: $\mathscr{L} = \sum_{j=1}^{n} \rho(e^{(j)})$.
- 全くディープラーニングのcost functionと同じような考えでしょう。
- Squared error (L_2): $\rho(e) = \frac{1}{2}\|e\|_2^2 = \frac{1}{2}e^T e$.
- Absolute error (L_1): $\rho(e) = \|e\|_1 = \sum_i |e_i|$.
- Cauchy loss: $\rho(e) = \frac{k^2}{2} log\left(1 + \frac{1}{k^2}\|e\|_2^2\right)$.

- Huber loss: $\rho(e) = \begin{cases} \frac{1}{2}\|e\|_2^2 & if \|e\|_2 \leq k \\ k\left(\|e\|_1 - \frac{k}{2}\right) & if \|e\|_2 > k \end{cases}.$
- Key difference: the slope of the loss function does not continue to increase as the errors become larger, but rather, it remains constant or it tapers off.
- Robust loss functions make the ICP problems slightly more difficult because we can no longer derive a nice closed form solution for the point cloud alignment step, and this means we need to add another iterative optimization step inside the main loop. これは当然だ。deep learningの最適化問題と同じく、gradient descentで最適化を解決する。以前のone stepでglobal minimumに飛べる方法はニュートン方法と同じ考えでしょう。
- However, the benefits cannot weigh the added complexity (??). つまりrobust loss functionsを利用するbenefitsがadded complexityと比べると薄い。

- 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.
  - assign less weight to large errors than the usual squared error loss.

論文：
LiDAR and Inertial Fusion for Pose Estimation by Non-linear Optimization: https://arxiv.org/pdf/1710.07104.pdf

- Most compelling ways to do state estimation, really are some guise of an optimization.
- You need to be thinking about what you're unsure about as well, and that's important when you're going to come up for some planning decisions.
- Wouldn't necessarily recommend using raw least squares because the stuff, that's an outlier.
  - Least squares spends all of its time worrying about rather than the stuff that makes sense.