自律走行システム特論（第4回）
Micro Mechatronics (4th)

- LiDARを用いた環境認識について　Environmental Recognition Using LiDAR
  - 環境認識の応用例を解説

    Explaining application examples of environmental recognition
  - 壁検出のプログラミングによる実習（・壁までの距離と角度の検出）

    Practical training with programming on a wall detection (detecting distance and angle to a wall)
- 自己位置推定（スキャンマッチング）Self-localization method (Scan matching)
  - ~~ICP、~~パーティクルフィルタなどについて解説

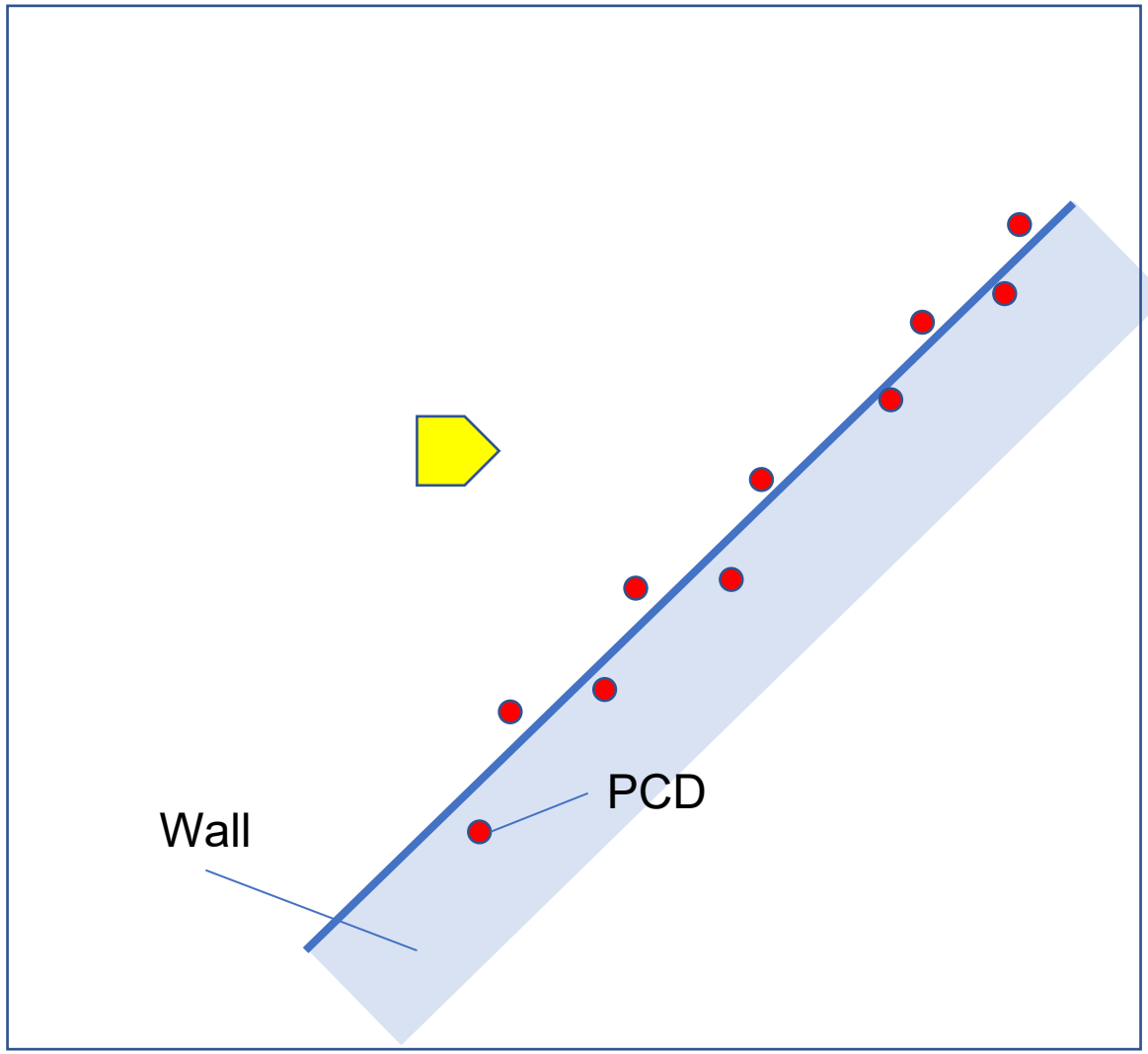
    Explanation about the scan matching method such as ~~ICP,~~ particle filters.
  - プログラミングによる実習　Practical training with programming
- 課題の確認と考察（環境認識、パーティクルフィルター）

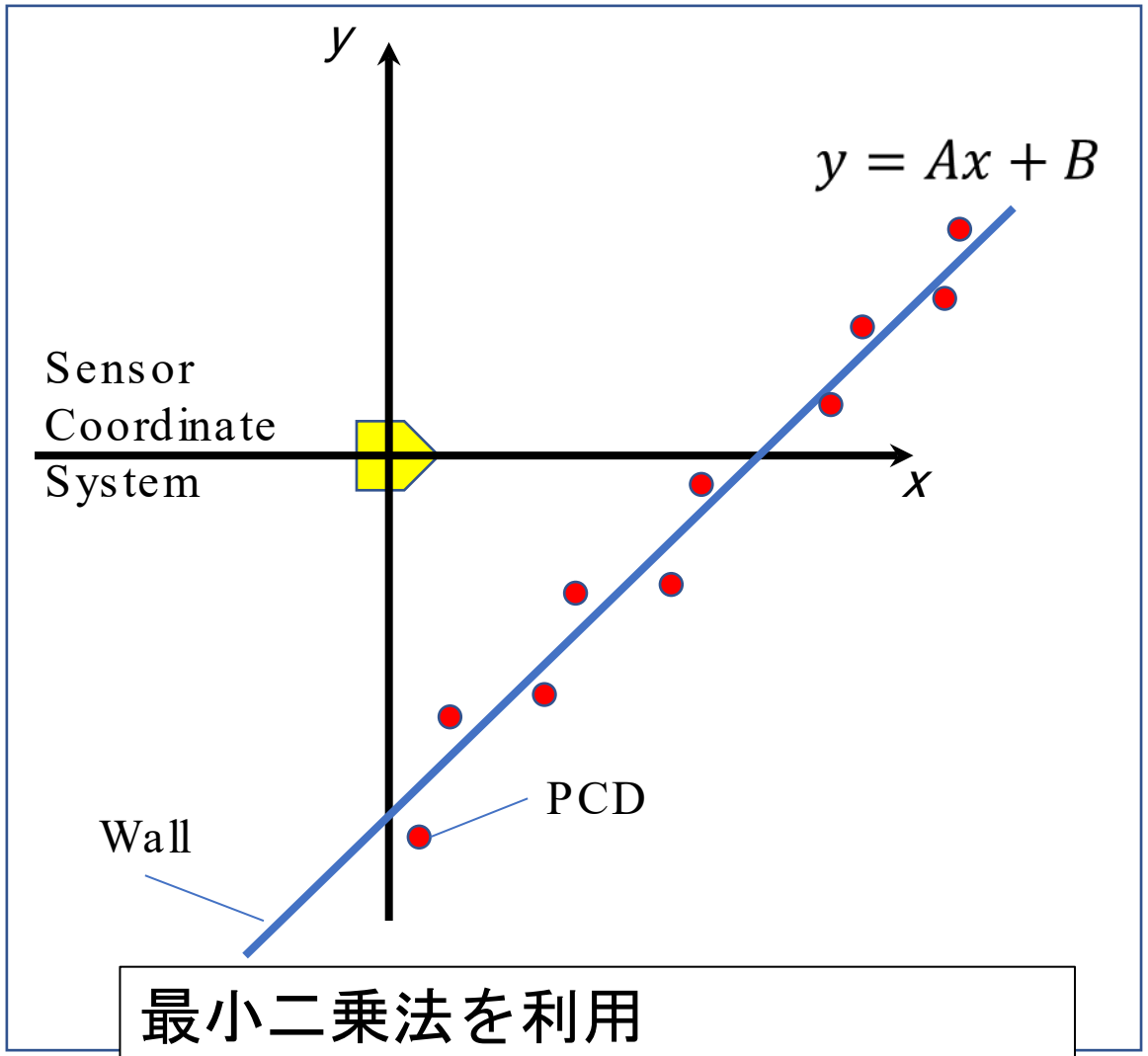
Confirmation and discussion for assignments (environmental recognition,particle filter)

環境認識の応用例（SLAM）
Application examples of environmental recognition (SLAM)

映像あり　Movie available

LiDARを用いた壁検出（壁までの距離と角度の検出）
Wall detection (detecting the distance and angle to a wall)



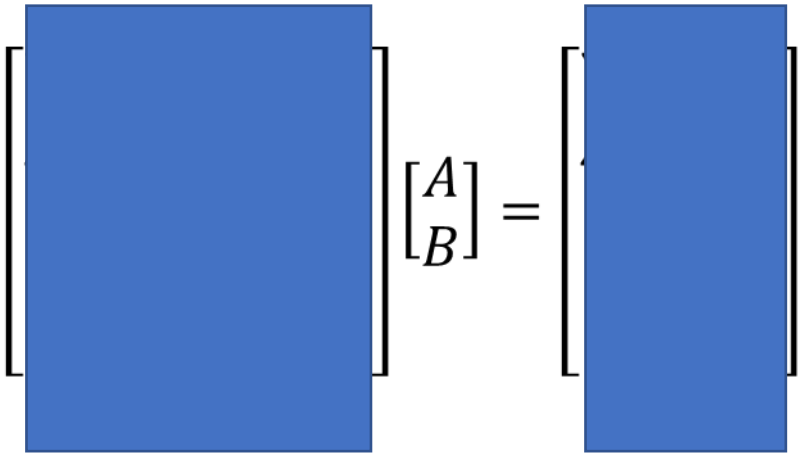


最小二乗法を利用
Using the least-squares method

最小二乗法　Least Squares Method

データの組(x_i, y_i)が多数与えられたときに，もっともらしい直線の式を$y=Ax+B$とおく。最小二乗法を用いて、AとBの関係を求めよ。

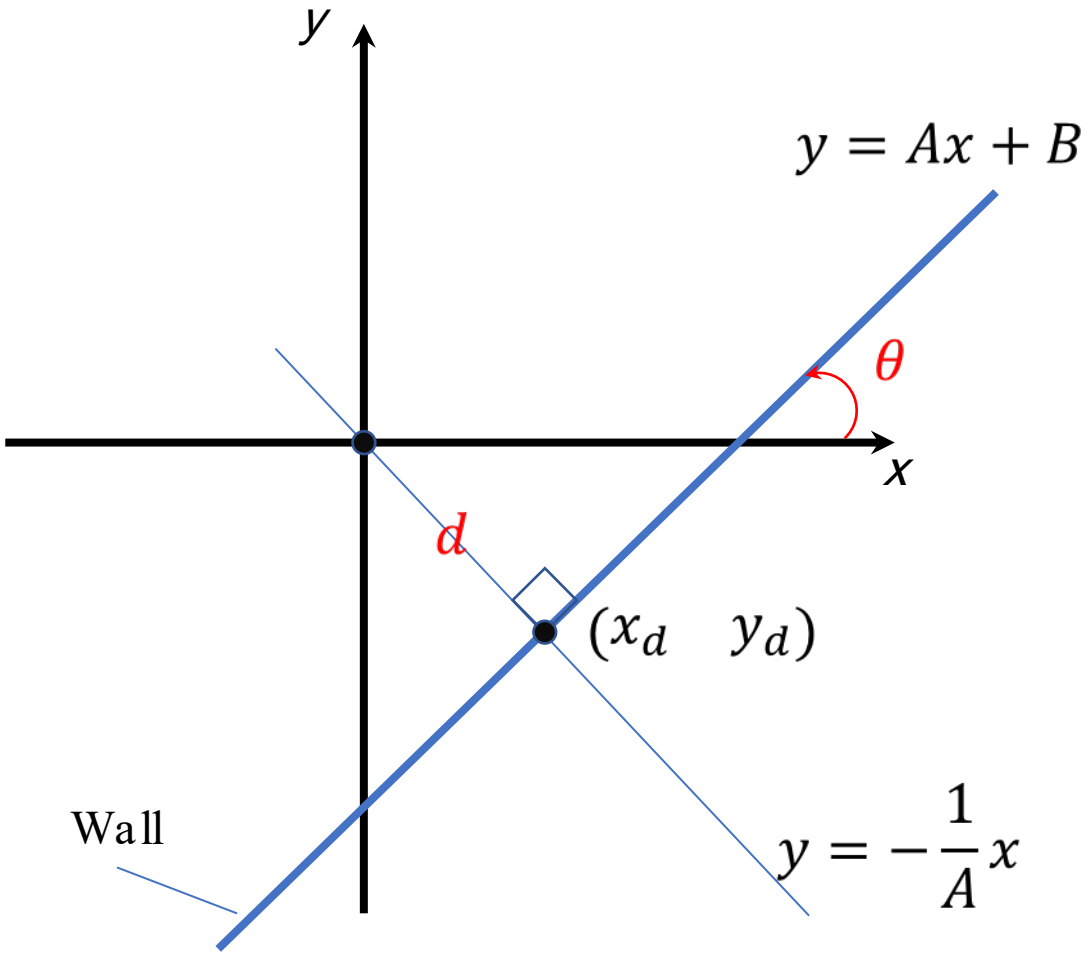
Given a large number of pairs of data (x_i, y_i), let $y=Ax+B$ be the equation for a plausible line. Using the least-squares method, find the relationship between A and B.



$$A =$$

$$B =$$

壁までの距離と角度の検出
Detecting the distance and angle to a wall



$$\begin{cases} x_d = \\ y_d = \end{cases}$$

$$\begin{cases} \theta = \\ d = \end{cases}$$

プログラムによる実習（LiDAR）#2
Practical training with programming (LiDAR) #2

- LiDARを用いた壁を検出するプログラムを作成して提出する

  Create and submit a program to detect the wall using LiDAR.

- 推定した壁までの距離(t-d)と角度(t-θ)をグラフにして提出する

  Submit a graph of your estimated distance (t-d) and angle (t-θ) to the wall.

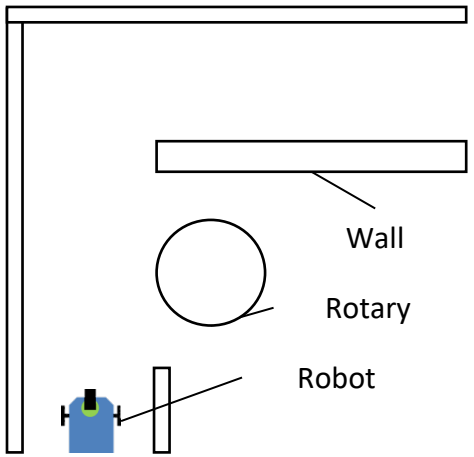
- ファイルデータ形式　Data Format in Input file

  # Time [s], Size, x(1)[mm], y(1), Intensity(1), x(2)[mm], y(2)[mm], Intensity(2), …. x(size)[mm], y(size)[mm], Intensity(size)

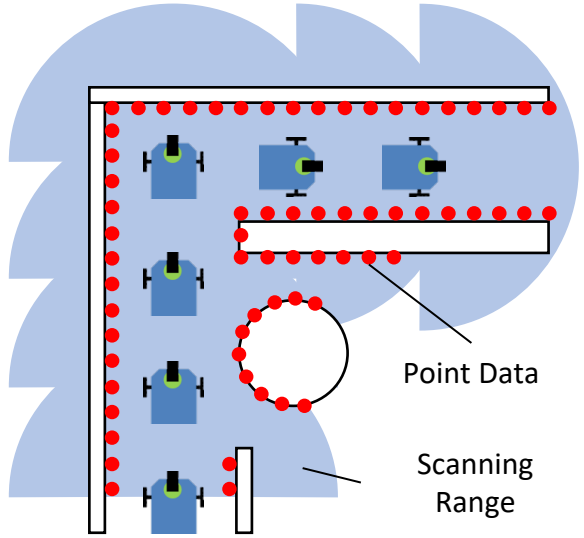
- パラメータ　Parameters

  ….

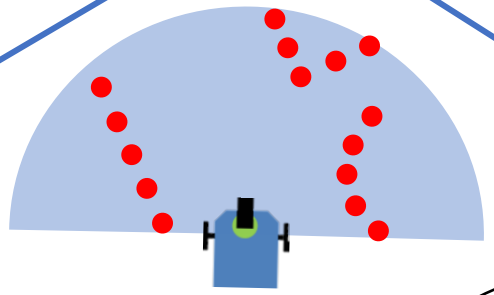
自己位置推定（スキャンマッチング）
Self-localization method (Scan matching)



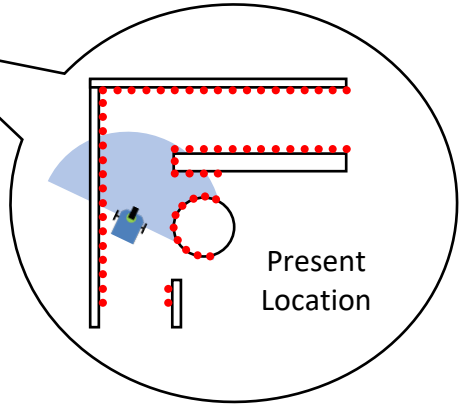
1) Move along a route beforehand



2) Create a map using LiDAR



3) Estimate a position by matching between a map and object data



LiDARを用いた自己位置推定法の種類
Type of Self-Localization method using LiDAR

参照資料：https://www.slideshare.net/hara-y/open-source-slam-3dcvtech

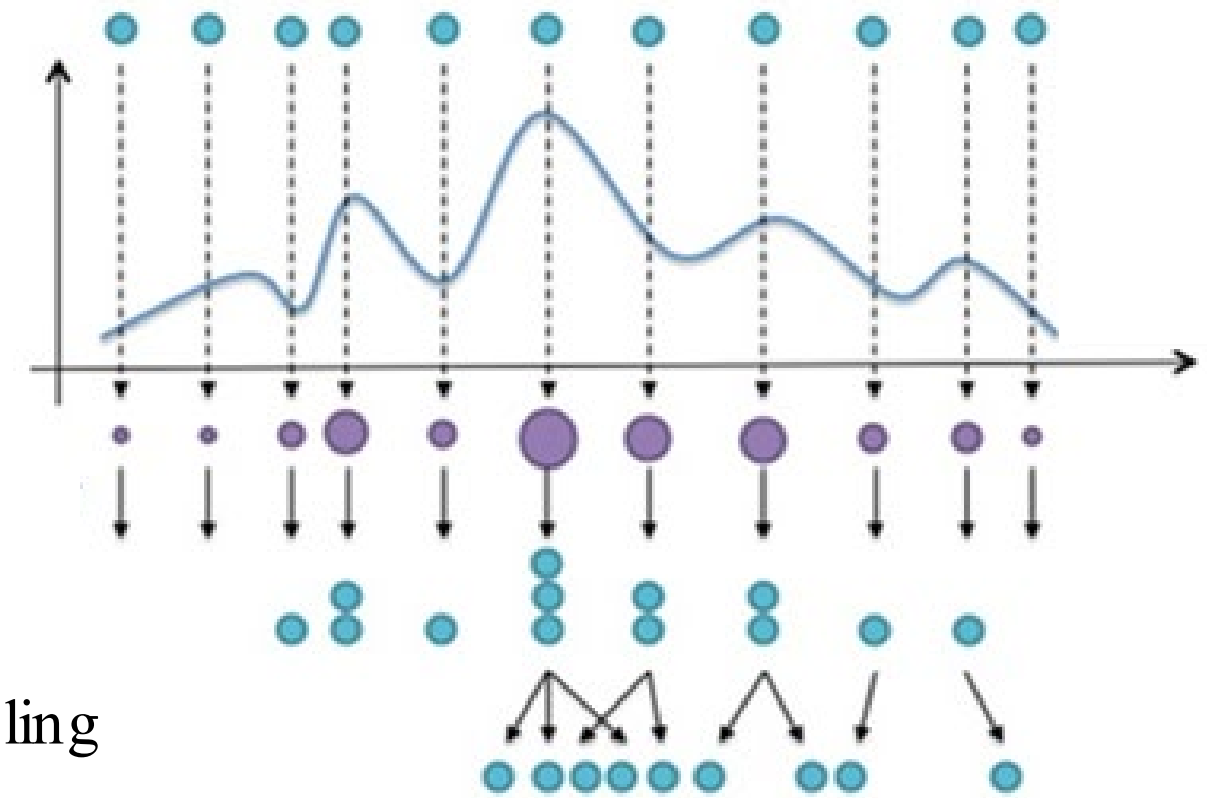
	ベイズフィルタ系 Bayesian Filter Type	レジストレーション系 Registration Type	グラフベース系 Graph-base type
Properties	フィルタリング、オンライン Filtering, On-line	逐次最適化、オンライン Sequential optimization, On-line	全体最適化、オフライン Total optimization, Off-line
Outline of method	事前確率と尤度を確率的に融合 Probabilistic fusion of prior probability and likelihood	点群を最適化計算で位置合わせ Aligning Point Clouds with Optimal Calculations	ロボット位置やランドマーク位置（地図）を表すグラフを最適化 Optimized graphs representing robot and landmark locations
Examples	Extended Kalman Filter, Particle Filter, etc	ICP(Iterative Closest Point), NDT(Normal Distributions Transform)	ポーズ調整、バンドル調整、完全SLAM Pose Graph, Complete SLAM

パーティークルフィルタの概要
Overview of Particle Filter

確率分布を粒子の集まりで表現することで非線形モデルにおいても観測データから状態を推定できる手法

A method that can estimate the state of a target from observed data even in nonlinear models by expressing probability distributions as collections of particles

- 予測　Prediction
- 観測（尤度の計算）
  Observations
  (likelihood calculations)
- 更新（重み付け平均）
  Update (weighted average)
- リサンプリング　Resampling



パーティークルフィルタの概要
Overview of Particle Filter

条件　Conditions

- それらしいと評価ができる尤度関数を設計する
  Design a likelihood function that can be evaluated as plausible
- 粒子の数をあらかじめ決める
  Determine the number of particles in advance.
- システムモデル(*x(t-1)*から*x(t)*を予測する関数)を用意する
  Prepare a system model (a function to predict *x(t)* from *x(t-1)*)

効果　Effectiveness

- 逐次的に内部状態を推定できる
  Can estimate the internal state in a sequential manner.
- センサデータをもとに現在の位置情報を推定できる
  Can estimate current location information based on sensor data
- 非線形なモデルでも適用可能
  Applicable to non-linear models

パーティークルフィルタの手順
Procedure of Particle Filter

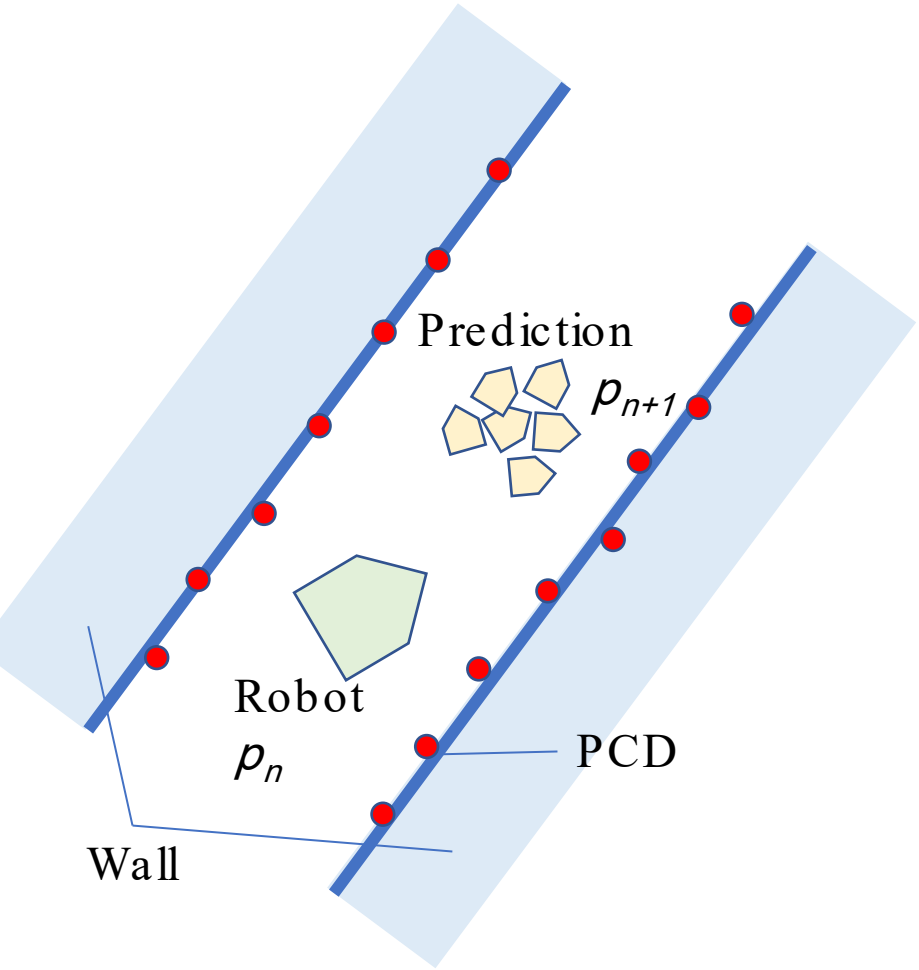
手順　Procedure

- 予測　Prediction
- 観測（尤度の計算）
  Observations (likelihood calculations)
- 更新（重み付け平均）
  Update (weighted average)
- リサンプリング　Resampling
  - 単純ランダム抽出法
    simple random sampling
  - 系統抽出法　systematic sampling
  - 層別抽出法　stratified sampling
  - Kullbuck-Leibler divergence
    (KLD sampling)

特徴　Features

- 時間計算量は粒子数をnとしたときO(n)
  The time calculation quantity is O(n) when the number of particles is n
- 実装が容易
  Easy to implement
- 予測→観測→更新→リサンプリングの繰り返し
  Prediction → Observation → Update → Resampling repeatedly

移動ロボットを用いた例　Example of A Mobile Robot



□ 予測　Prediction

N：全粒子数　Total number of particles

p_n：状態変数（未知）　$p_n = [x_n \quad y_n \quad \theta_n]^T$

State variables (unknown)

o_n：観測データ（既知）Observed data (known)

w_n：重み（全粒子の尤度で正規化）

Weights (normalized by the likelihood of all particles)

$F(p)$：p_n を予測するシステムモデル

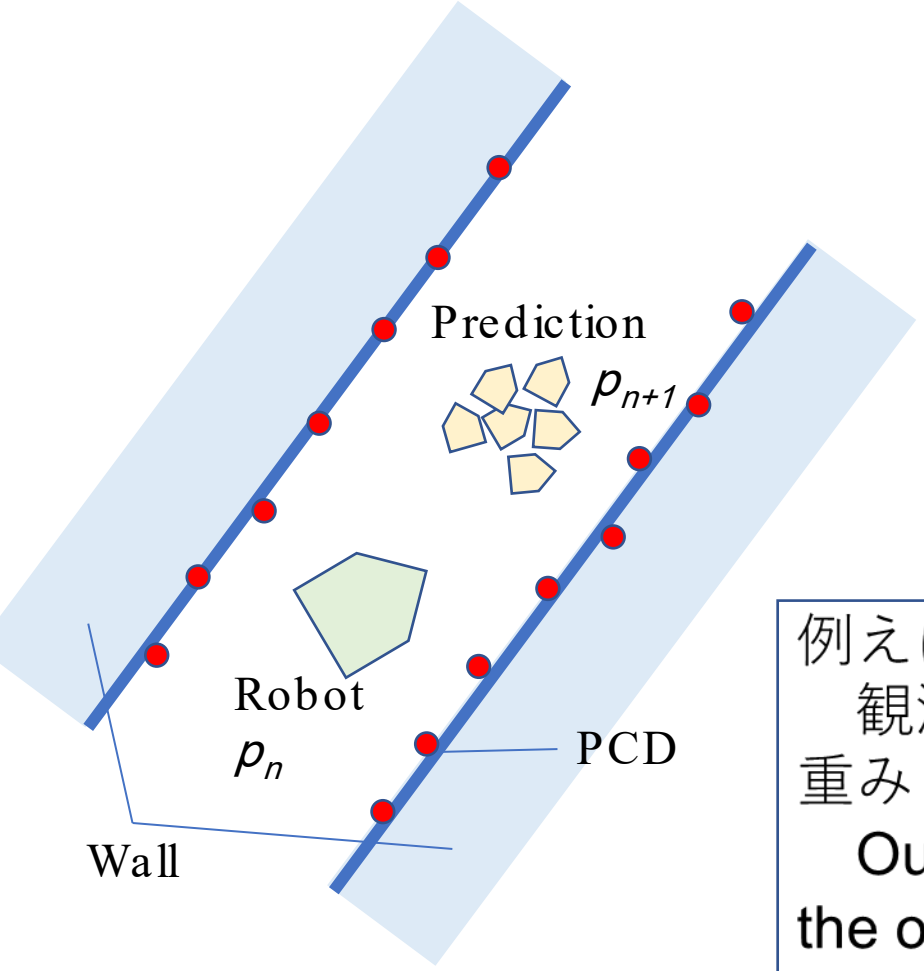
A system model to predict p_n

$$p_{n+1} = F(p_n) = R(\theta_n)[v \cdot dt \quad 0 \quad \omega \cdot dt]^T + p_n$$

＊開始時は均等にばらまくのみ

Only spread out particles evenly, when it starts.

移動ロボットを用いた例　Example of A Mobile Robot



■ 観測（尤度の計算）

Observations (likelihood calculations)

$L(p, o)$：予測したp_nと観測したo_nから尤もらしい確率を与える尤度関数

A likelihood function that gives a plausible probability from the predicted p_n and observed o_n

例えば　Example

観測したPCDをp_{n+1}で座標変換して地図とマッチングした数を重みとして出力

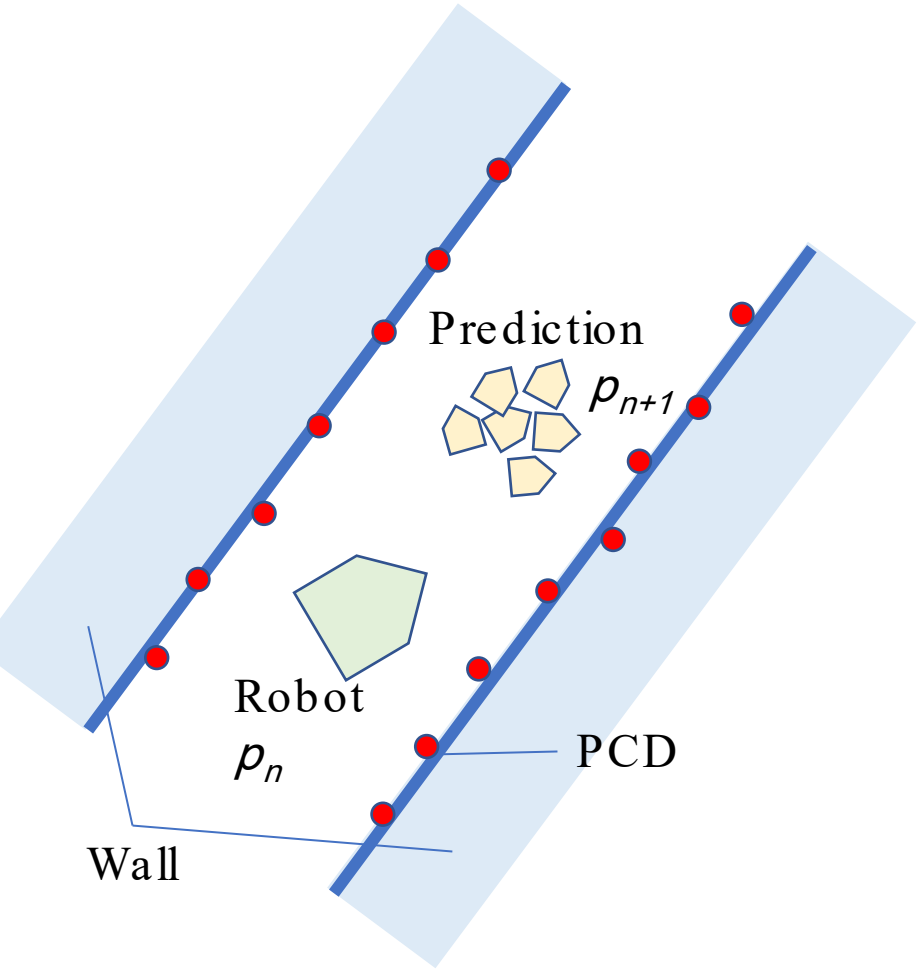
Output　the number of matches to the map as weights, using the observed PCDs that are converted to coordinates by p_{n+1}

$$w_{n+1} = L(p_{n+1}, o_{n+1})$$

重みの正規化
Weight Normalization

$$w_{n+1} = \frac{w_{n+1}^{(i)}}{\sum_i^n w_{n+1}^{(i)}}$$

移動ロボットを用いた例
Example of A Mobile Robot



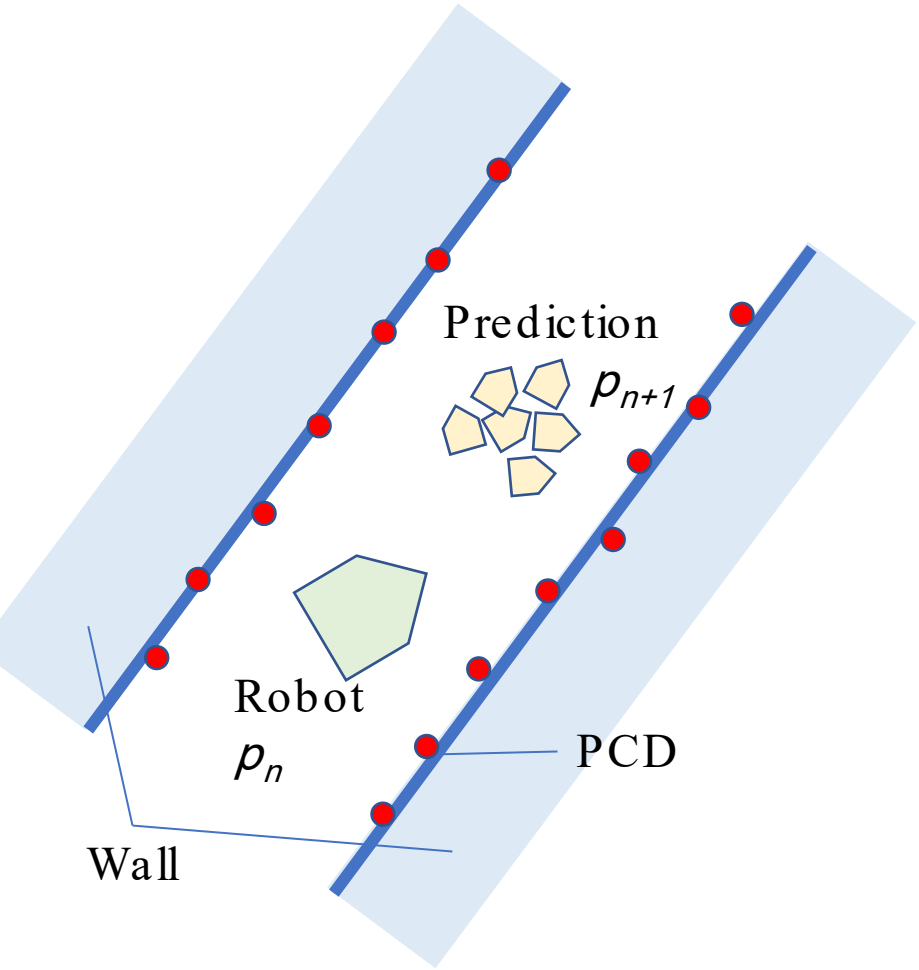
- 更新（重み付け平均）
  Update (weighted average)

$$\hat{p} = \sum_{i}^{n} w_{n+1}^{(i)} p_{n+1}^{(i)}$$

$\hat{p}$が推定した自己位置

$\hat{p}$ is estimated self-localization

移動ロボットを用いた例
Example of A Mobile Robot



■ リサンプリング　Resampling

重みの大きさに応じてパーティークルを再配分する。その際、適当なノイズを付与する。

Redistribute the particles according to the magnitude of the weight. At that time, appropriate noise is given.

$$p_{n+1}^{(j)} = p_{n+1}^{(i)} + \delta^{(j)}$$

jは重みに応じて数を決める

The number of j depends on the weight

プログラムによる実習（パーティークルフィルタ）#1
Practical training with programming (Particle Filter) #1

- パーティークルフィルタ用いた自己位置推定するプログラムを作成して提出する

  Create and submit a program to estimate self-positions using the particle filter.

- 推定した位置(x,y)、方位(t-yaw)をグラフにして提出する

  Submit a graph of your estimated position(x,y), orientation(t-yaw).

- ファイルデータ形式　Data Format in Input file

  # Time [s], Size, x(1)[mm], y(1), Intensity(1), x(2)[mm], y(2)[mm], Intensity(2), …. x(size)[mm], y(size)[mm], Intensity(size)

  # Time [s], Encoder Counter1(incremental difference), Encoder Counter2 (incremental difference), PWM1, PWM2

- パラメータ　Parameters

  車輪半径 Wheel radius : 0.0406[m], トレッド Tread : 0.2765[m], ギア比 Ratio of gear : 54.0699,

  1回転分のカウント Num. of counter for one revolution : 2048