version 1.2

TIER IV ACADEMY
自動運転システム構築塾

Day1　自動運転システム実践解説

自動運転システムの自己位置推定技術

Tier IV
Intelligent Vehicle

目次

第１章：自己位置推定とは

第２章：位置推定手法の種類

第３章：Autoware の自己位置推定システム

1. Autoware の自己位置推定
2. Autoware での位置推定の実装

第4章：まとめ

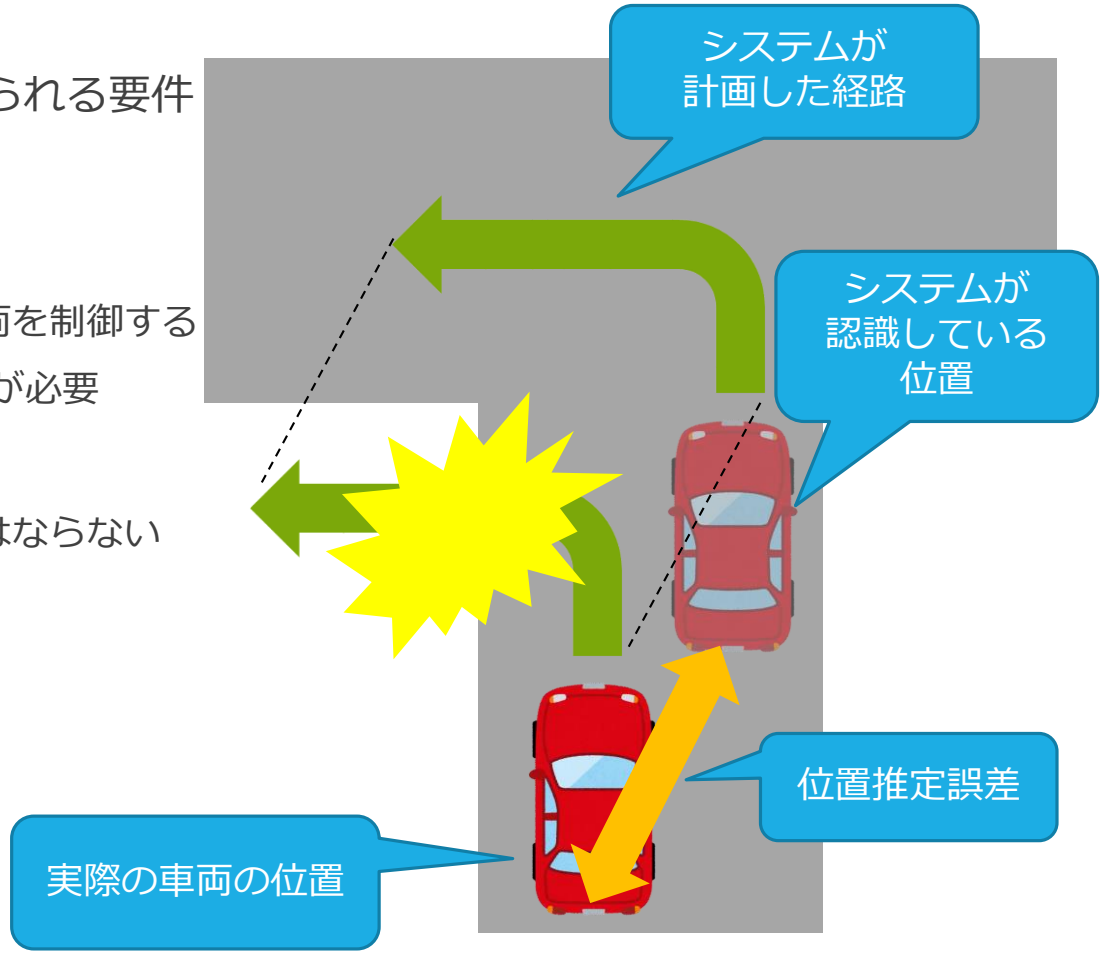
自動運転システムの自己位置推定技術

第１章：自己位置推定とは

自己位置推定とは

- 走行中の車両の **位置**・**向き** を推定すること

- 自動運転システムの位置推定に求められる要件
  1. 精度(数10cm以内)
  2. リアルタイム性
     - 現在の車両の位置・向きから、車両を制御するため、高い精度・リアルタイム性が必要
  3. ロバスト性(安定性)
     - 場所・環境の変化に影響を受けてはならない



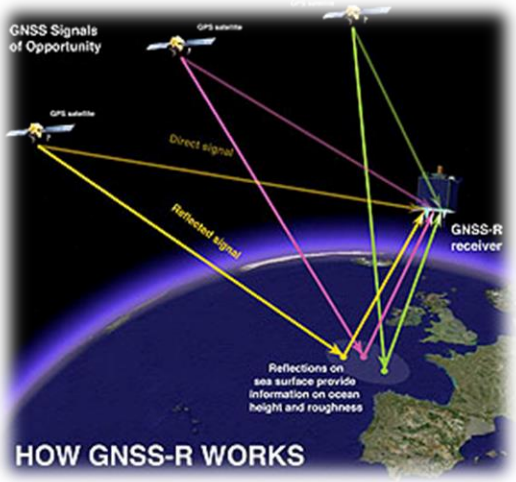
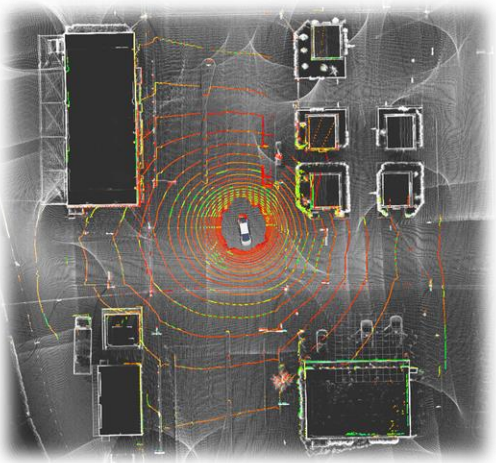
位置推定が高精度にできないと、
重大な事故につながる可能性がある

自動運転システムの自己位置推定技術

第２章：位置推定手法の種類

位置推定手法

位置推定には様々な手法があり、用いられるセンサーも異なる

	デッドレコニング (自律航法)	GNSS (Global Navigation Satellite System)	スキャンマッチング
用いられる センサー	IMU ホイールエンコーダ等	GNSS受信機	LIDAR
利点	どこでも使える	地図がなくても 使える	高精度な 位置推定が可能
欠点	誤差の蓄積大	• 信号の受信状況に影響 • トンネル内では使えず	• 地図データが必須 • 特徴のない エリアでは 使えず
			

http://noc.ac.uk/conference/ocean-observation-workshop

デッドレコニング（自律航法）

車両に取り付けられた内部センサを用いて、車両の位置を逐次的に推定

- 内部センサの例
  - IMU (Inertial Measurement Unit)
    - 慣性計測装置、3軸ジャイロ+3方向加速度計
  - オドメトリ
    - ホイールエンコーダによるタイヤの回転角・回転数

- 利点
  - 場所によって精度が変わらない
    (衛星信号が届かない場所でも使える)
  - 短期的には精度が良い

- 欠点
  - ホイールエンコーダはタイヤの滑りを検知できない
  - 誤差の蓄積 → デッドレコニング単体では位置推定困難

http://gigazine.net/news/20140909-google-self-driving-car-sensor/

GNSS（測位衛星システム）

複数の衛星からの信号を受信機が受信することにより、グローバルな(地球上の)位置を取得

- 様々な測位衛星システムが存在し、衛星群によって目的が異なる
- GPSといっても、GPS以外の衛星に対応したGNSS機能が搭載されているものが多数

1.グローバル軌道衛星群

	GPS (米)	GLONASS(露)	Galileo(EU)	BeiDou(中)
（計画）衛星数	32	24	30 (8機試験中)	35 (15機運用中)

2.補強衛星群 - 測位精度を向上するための補正信号を送信 (現在運用中)

	WAAS (米)	EGNOS(EU)	MSAS(日)	GAGAN(印)
衛星数	4	3	2	2

3.特定地域衛星群 - 特定地域上空に衛星を配置し、衛星信号を受信しやすくする

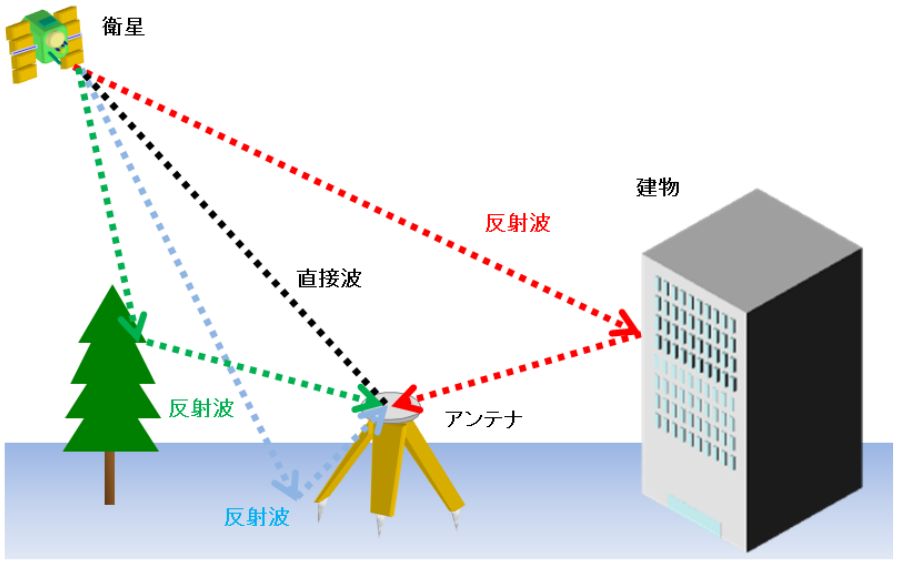
	QZSS(日)	IRNSS(印)
衛星数	4+3	7

※よく言われるGPSは、アメリカによって開発・運用されているシステムであり、あくまでGNSSの1種類です

GNSS（測位衛星システム）

GNSSの大きな誤差要因 - マルチパス

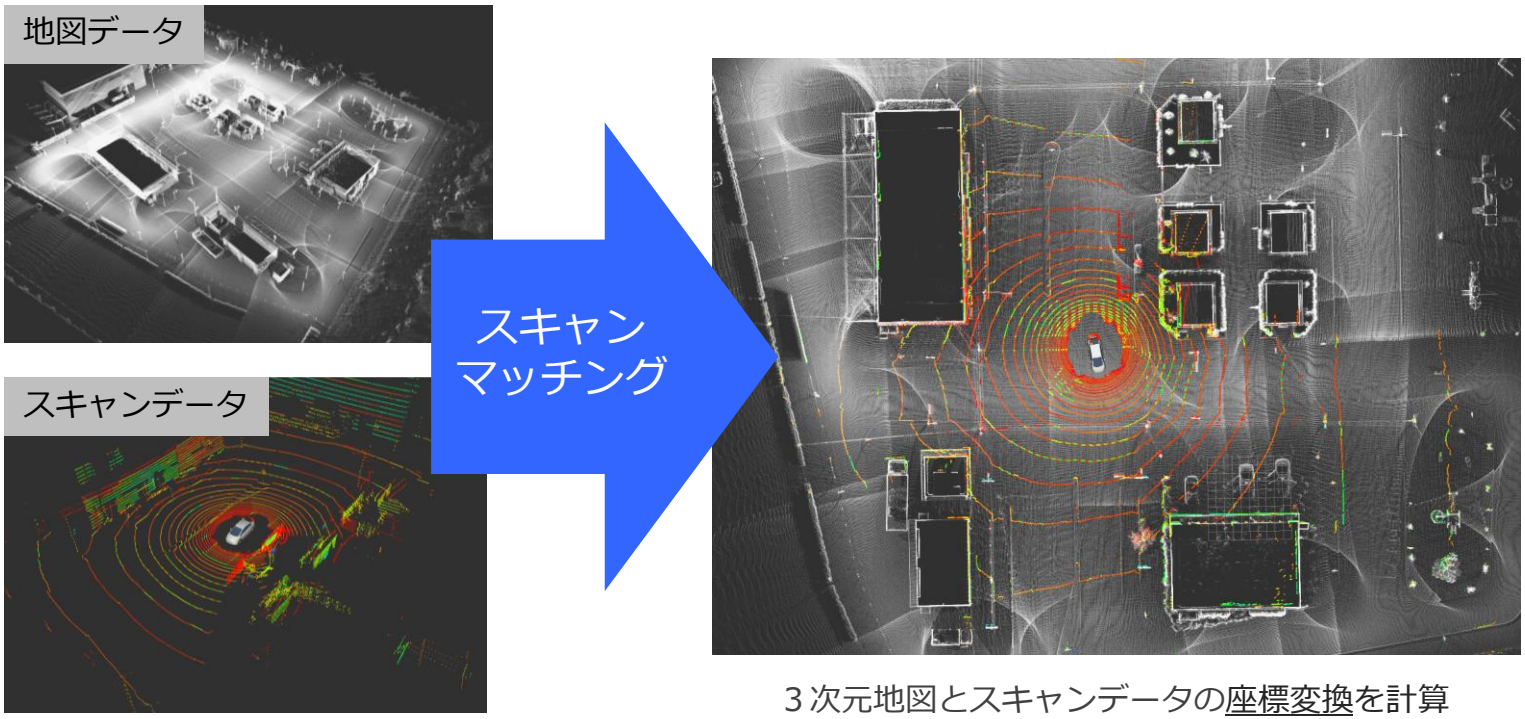
- 信号が受信機に直接届かず、反射物に跳ね返った信号を受信することによって、受信遅延が生じるために発生
- 約 1～10mの誤差が発生



測位衛星技術株式会社

スキャンマッチング

地図データとスキャンデータがきれいに重なる座標変換を計算し、
地図内の位置・向きを算出



3次元地図とスキャンデータの座標変換を計算
車両の位置・向き

代表的なスキャンマッチングのアルゴリズム

- ICP (Iterative Closest Point) - P.J. Besl et al. (1992)
- 2D-NDT (Normal Distributions Transform) - P. Biber et al. (2003)
- 3D-NDT - E. Takeuchi et al. (2006) , M. Magnusson et al. (2007)

参考：SLAM

SLAM - Simultaneous Localization and Mapping

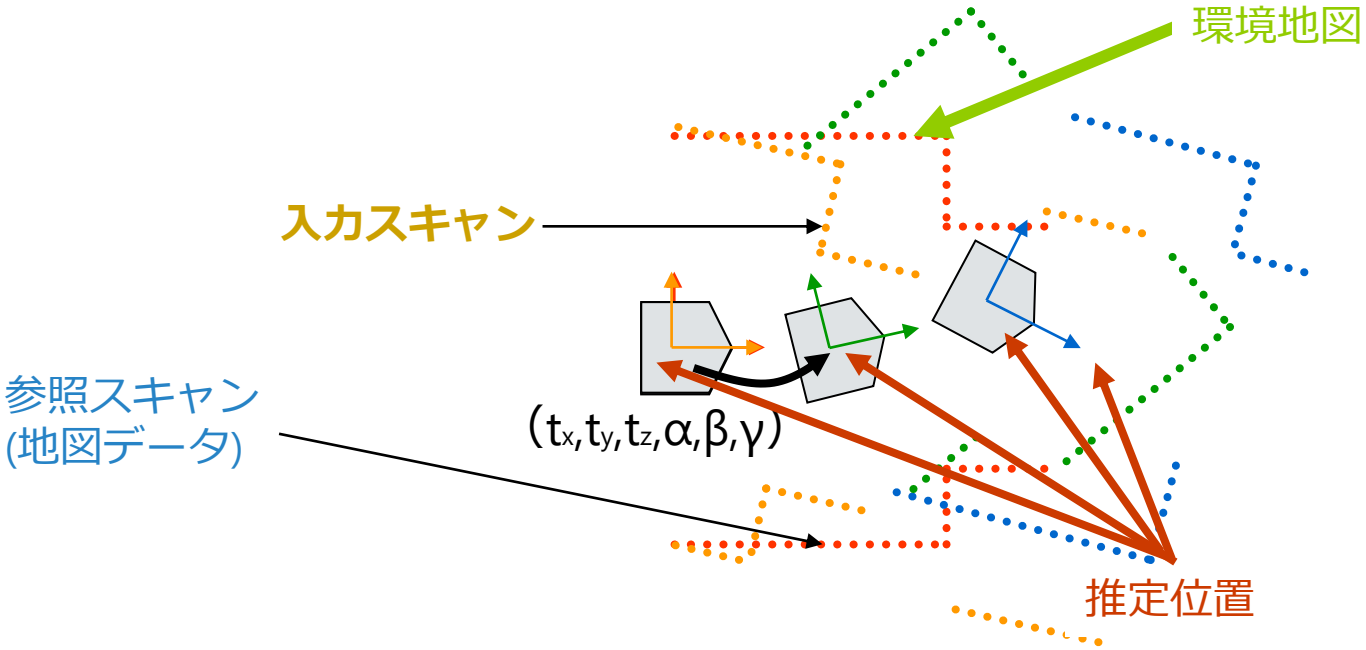
自己位置推定と環境地図作成を同時に行うこと

Localization (位置推定)

- 地図 が与えられている
- 計測値を地図と照らし合わせて 位置を特定

Mapping (地図生成)

- 位置 が与えられている
- 位置情報に計測値を重ね合わせて 地図を作成

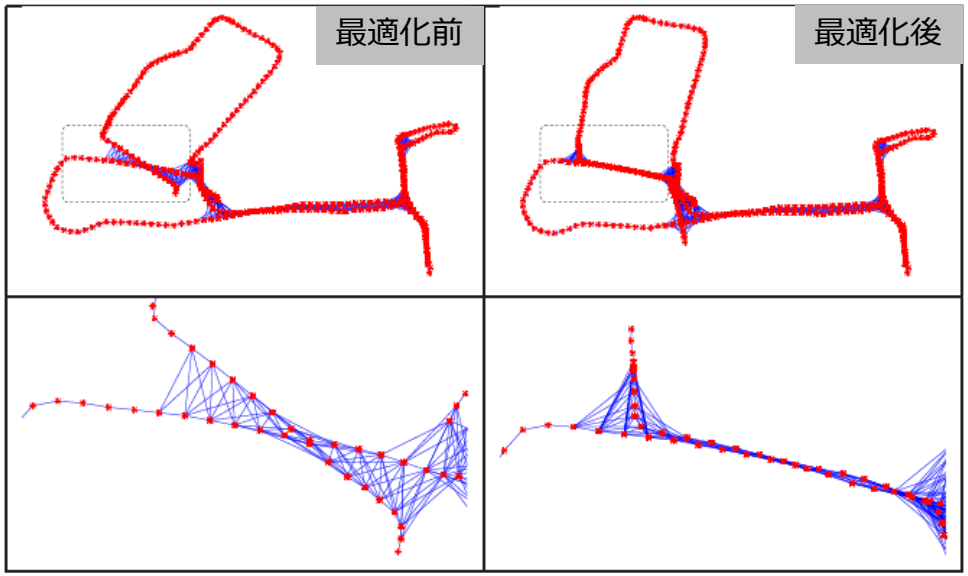


参考：Graph SLAM

SLAMにおいて、地図の誤差修正、ループクロージングに対応

ノードとノード間の拘束からなるグラフの最適化

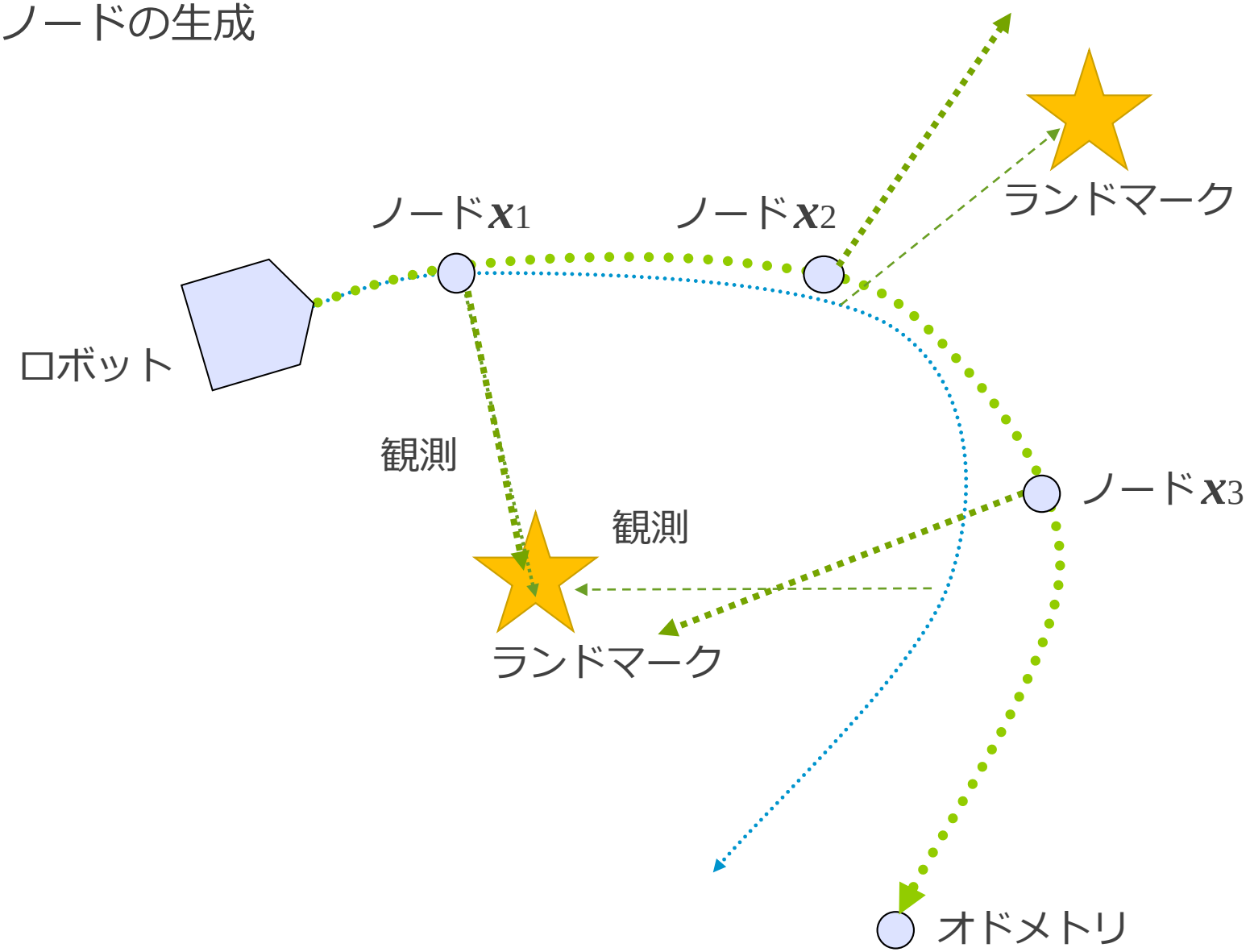
1. ノードの生成
2. ノード間の拘束の生成
3. ノード位置の最適化



Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A., & Hertzberg, J. (2008).
Globally consistent 3D mapping with scan matching. Robotics and Autonomous Systems, 56(2), 130-142.

参考：Graph SLAM

1. ノードの生成

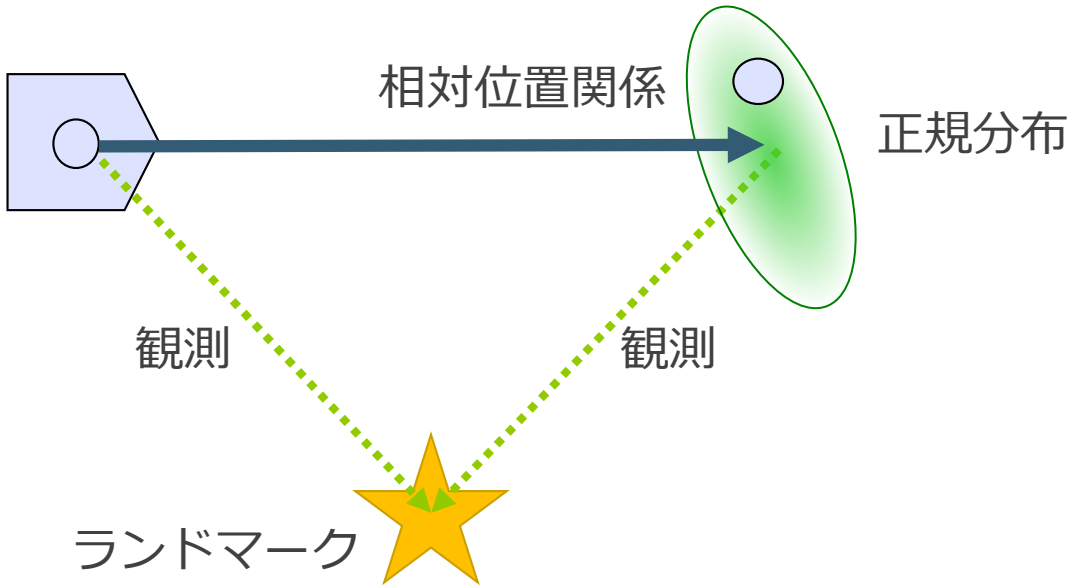


参考：Graph SLAM

拘束 – 2つのノード間の相対位置と分散

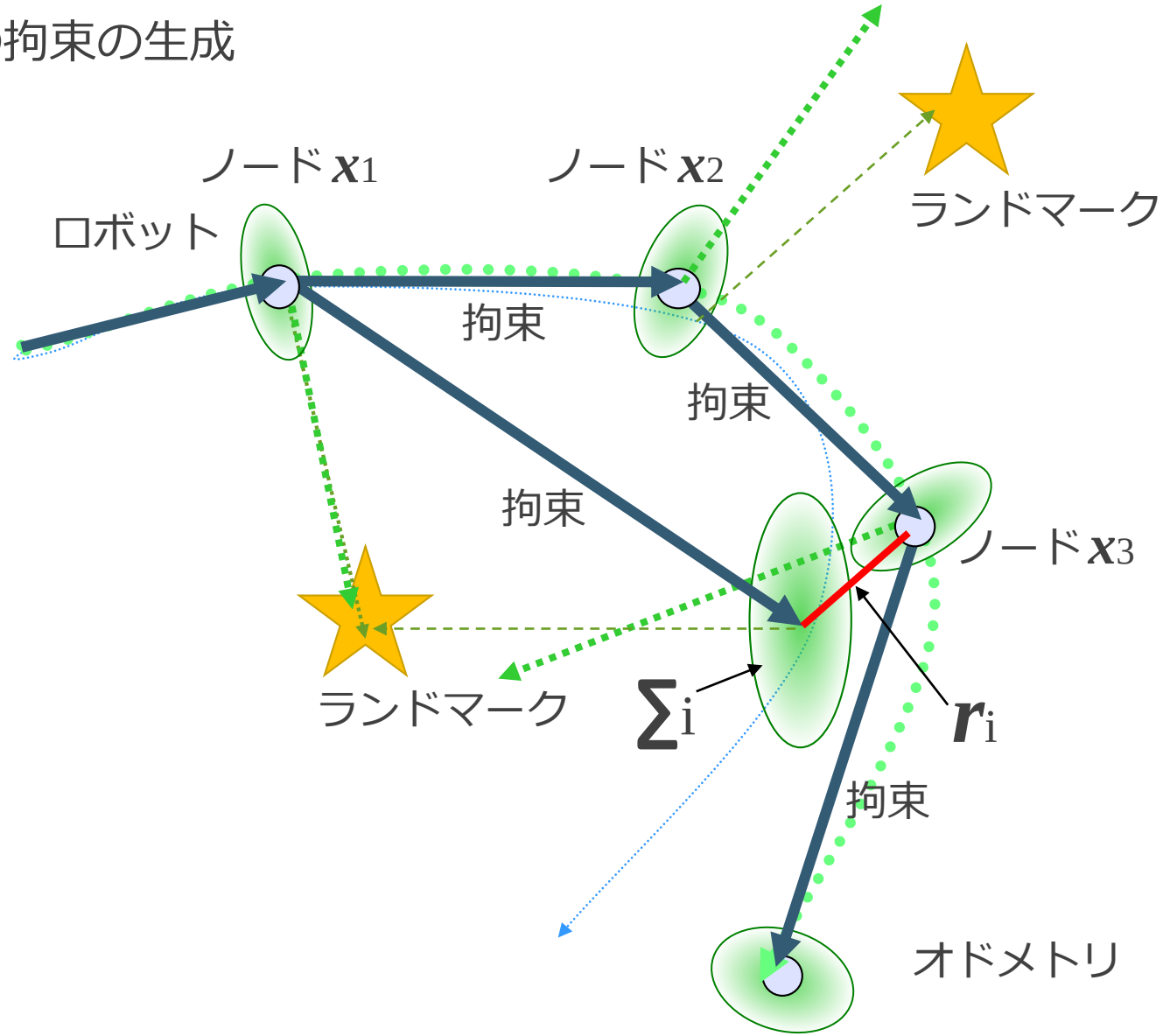
オドメトリでの拘束

ランドマーク観測を
介した拘束



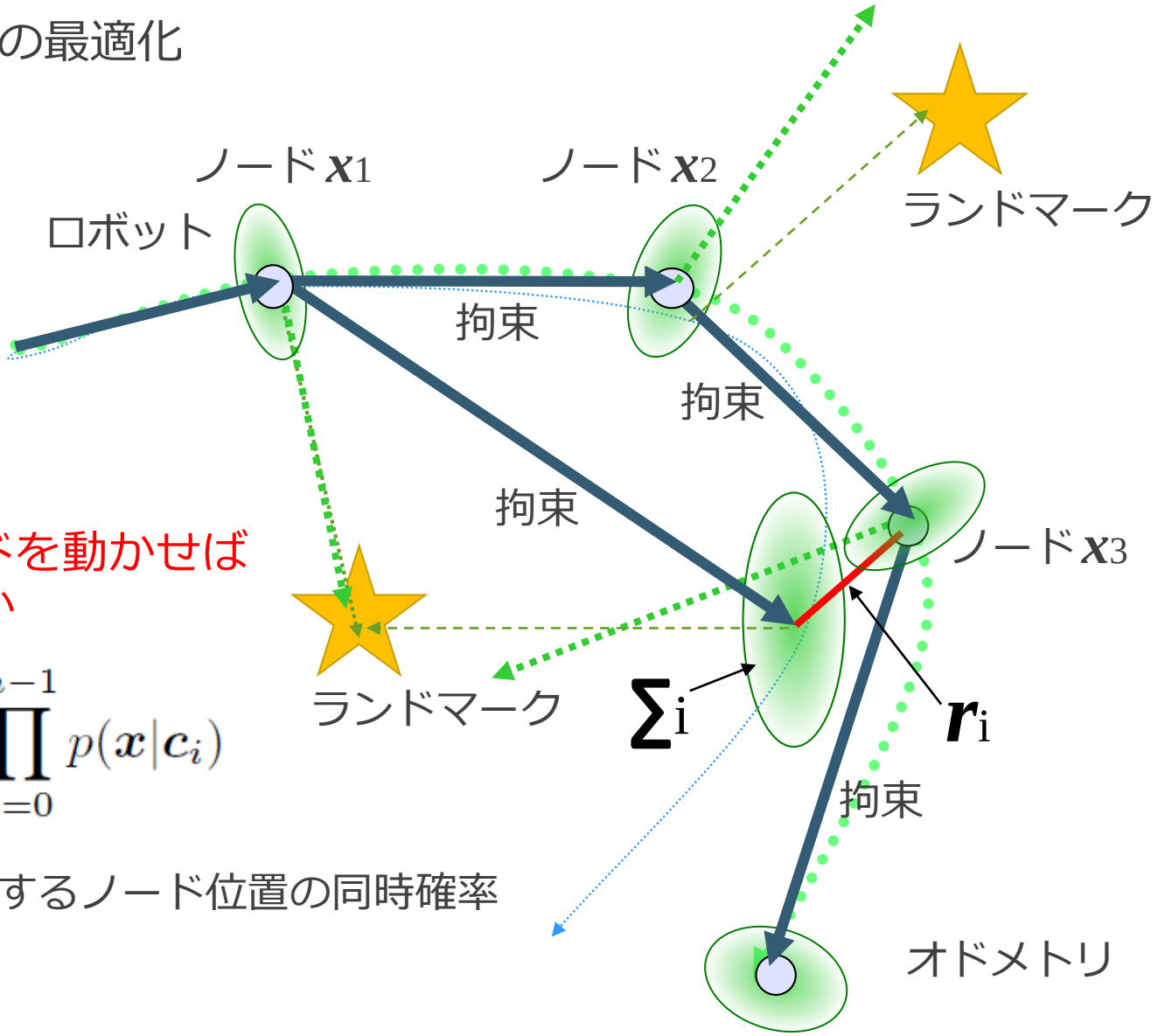
参考：Graph SLAM

2.ノード間の拘束の生成



参考：Graph SLAM

3.ノード位置の最適化



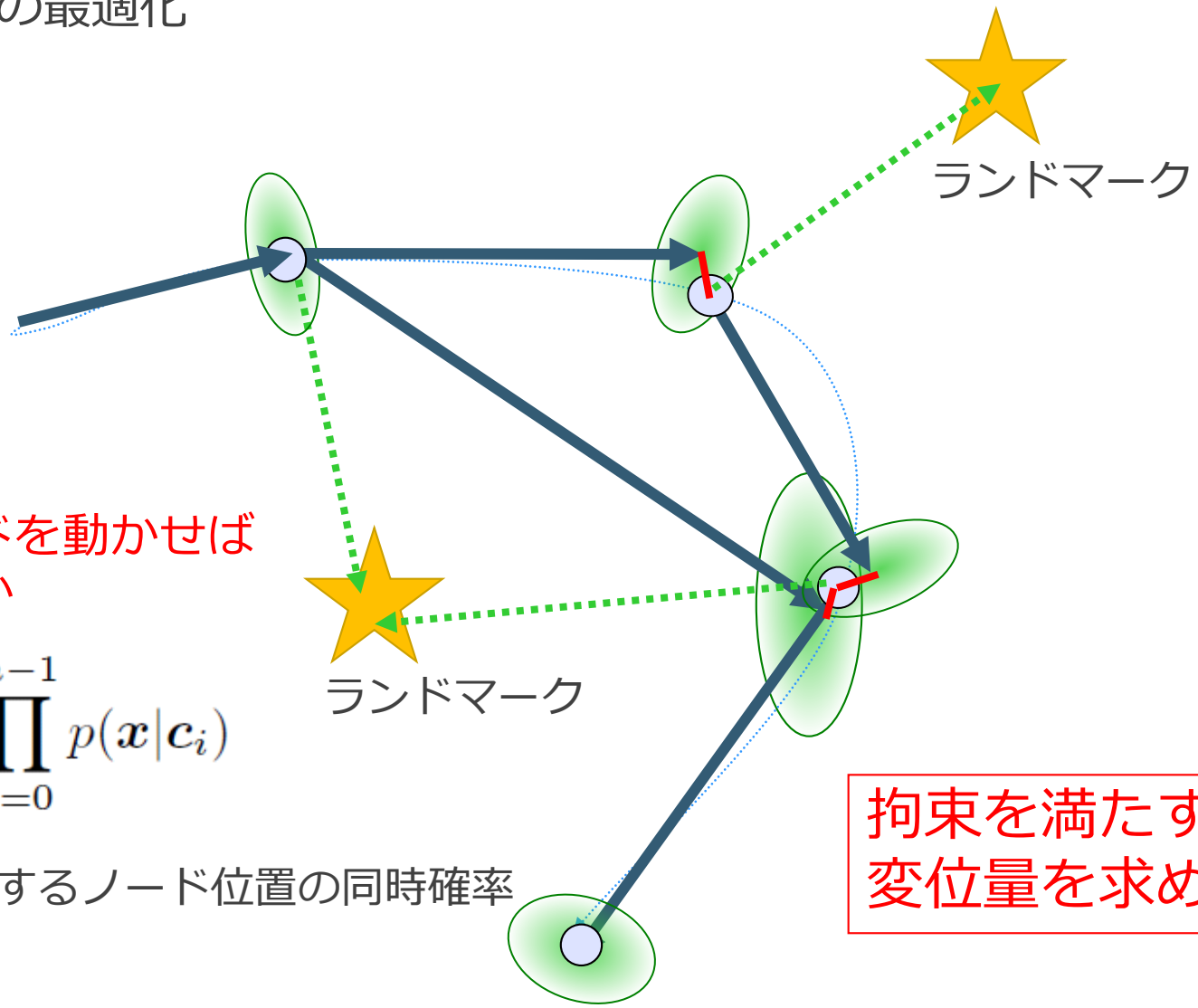
どれだけノードを動かせば
拘束を満たすか

$$p(\boldsymbol{x}) = \prod_{i=0}^{n-1} p(\boldsymbol{x}|\boldsymbol{c}_i)$$

全ての拘束に対するノード位置の同時確率

参考：Graph SLAM

3.ノード位置の最適化



どれだけノードを動かせば
拘束を満たすか

$$p(\boldsymbol{x}) = \prod_{i=0}^{n-1} p(\boldsymbol{x}|\boldsymbol{c}_i)$$

全ての拘束に対するノード位置の同時確率

拘束を満たす
変位量を求める

参考：Graph SLAM

全拘束の同時確率 $p(\boldsymbol{x}) = \prod_{i=0}^{n-1} p(\boldsymbol{x}|\boldsymbol{c}_i)$ を最大化

正規分布を仮定

$$p(\boldsymbol{x}) = \prod_{i=0}^{n-1} \exp(-\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}_i)^t \boldsymbol{\Sigma}_i^{-1} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}_i))$$

$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$ 期待値との差の関数

↓ 対数尤度

$$log(p(\boldsymbol{x})) = -\sum_{i=0}^{n-1} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}_i)^t \boldsymbol{\Sigma}_i^{-1} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}_i)$$

↓ 線形化

$$log(p(\boldsymbol{x})) = -\sum_{i=0}^{n-1} (\boldsymbol{J}_i \boldsymbol{u} - \boldsymbol{r}_i)^t \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{J}_i \boldsymbol{u} - \boldsymbol{r}_i)$$

$\boldsymbol{u}$: 全ノードの変位量

↓ 極値を求める

$$\sum_{i=0}^{n-1} (\boldsymbol{J}_i^t \boldsymbol{\Sigma}_i^{-1} \boldsymbol{J}_i) \boldsymbol{u} = \sum_{i=0}^{n-1} \boldsymbol{J}_i^t \boldsymbol{\Sigma}^{-1} \boldsymbol{r}_i$$

$$\boldsymbol{A} \quad \boldsymbol{u} = \quad \boldsymbol{b}$$

巨大な連立方程式を解く問題に帰着

自動運転システムの自己位置推定技術

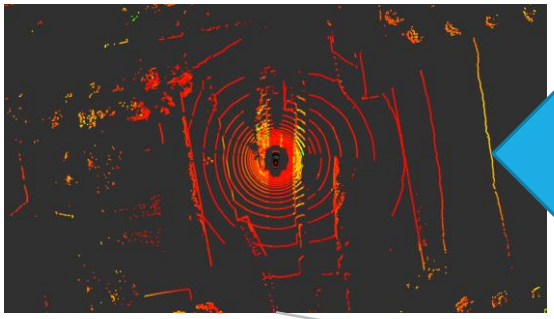
第３章： Autoware の自己位置推定システム

1. Autowareの自己位置推定
2. Autowareでの位置推定の実装

Autowareの自己位置推定システム

Autowareの自己位置推定の特徴

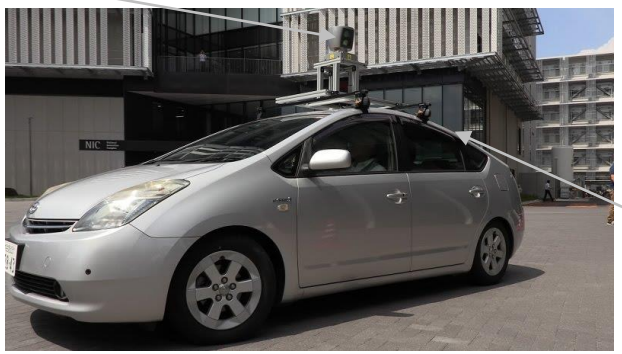
- ✓高精度3次元地図+LIDARのスキャンデータのNDTスキャンマッチング
- ✓高精度(誤差約10cm以内)かつ高速(リアルタイムに動作)
- ✓各種LIDAR対応(Velodyne HDL-64E/32E, VLP-16, Hokuyo 3D-URG)
- ✓GNSSやIMUは補助的に使用(無くても可)



NDT
Scan Matching

LIDARのリアルタイムデータ

高精度3次元地図は、 Autoware PCに保存



GNSS/IMUも補助的に使用可能

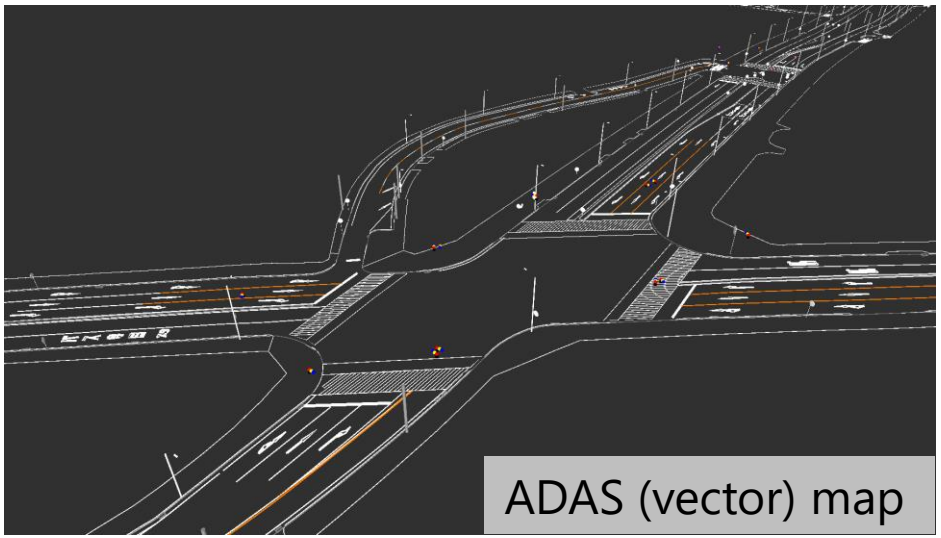
高精度３次元地図

屋外の3次元情報を取得

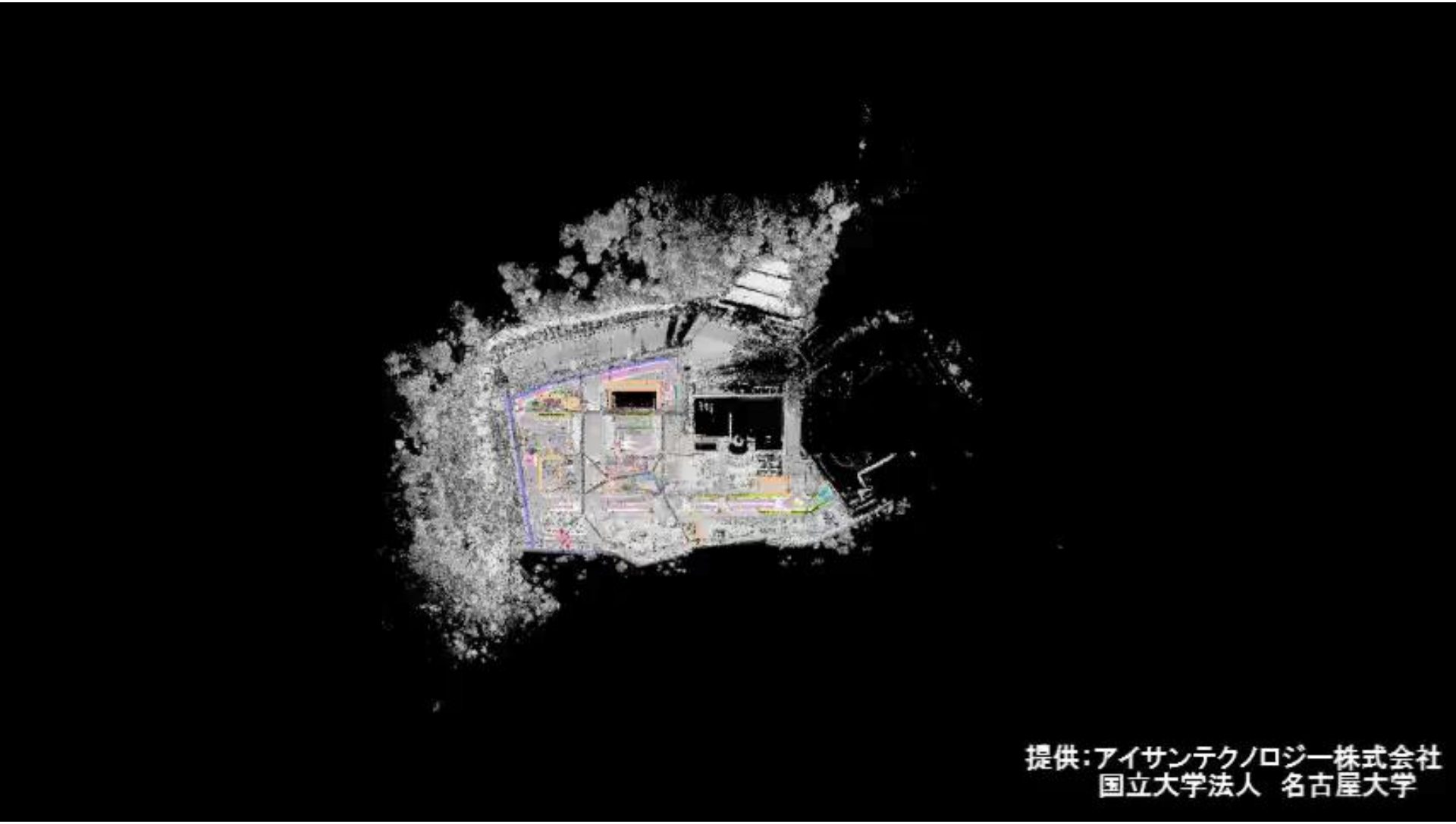
- ポイントクラウド地図
  - ✓ 3次元座標(緯度・経度・標高)
  - ✓ RGB値
- ADAS地図 - 点群地図から地物を抽出
  - ✓ 信号、路面標示 etc.

MMS - Mobile Mapping System

http://www.whatmms.com/whatmms

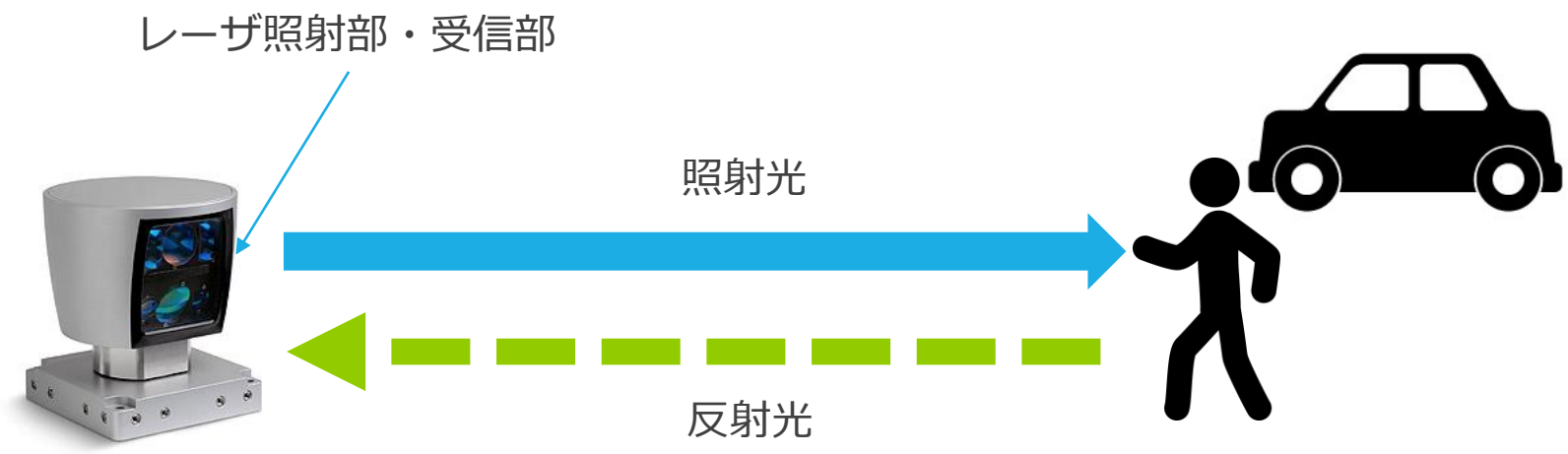


高精度３次元地図



LIDAR

レーザーを対象物に照射し、散乱光を測定することにより、
対象物までの距離(や性質)を取得



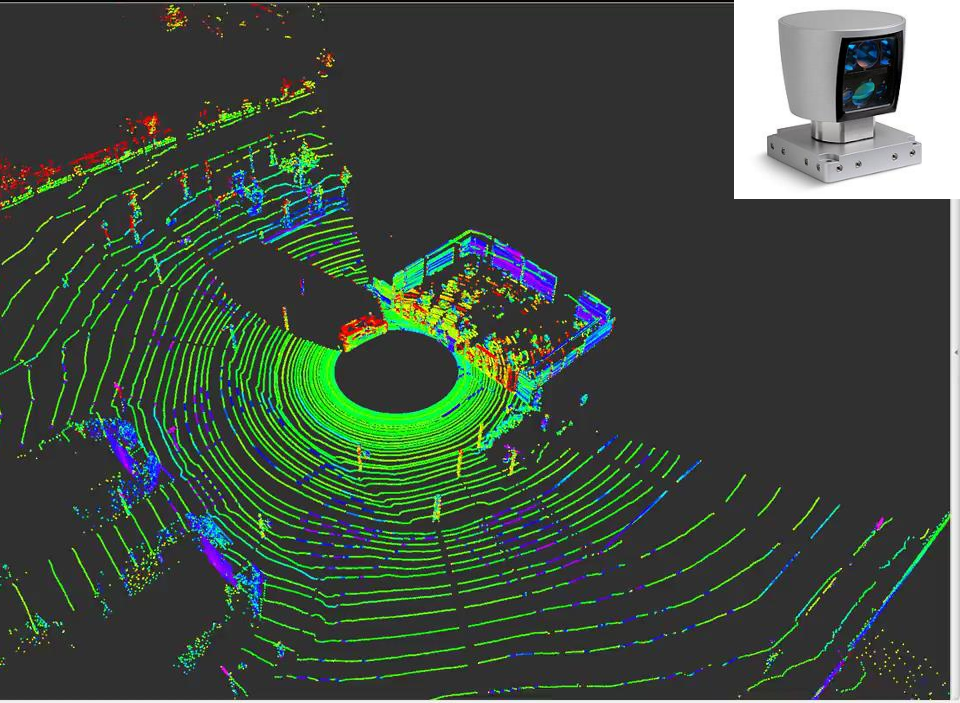
対象物までの距離、位置、反射強度を取得

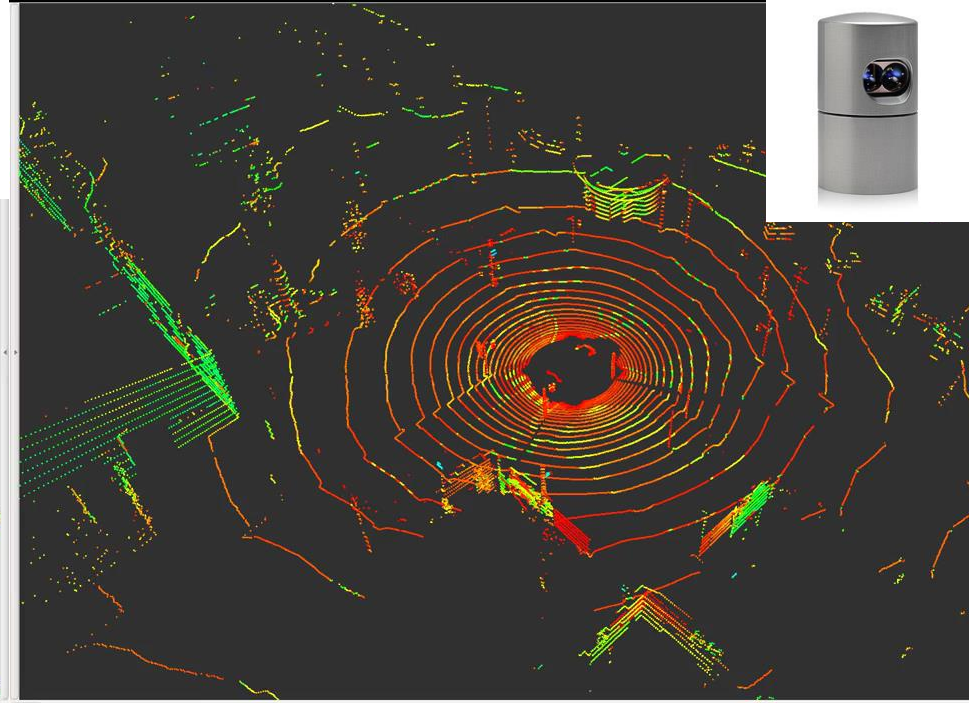
※ Rader – Radio Detection and Ranging レーザーではなく電波（波長が長い）を用いる

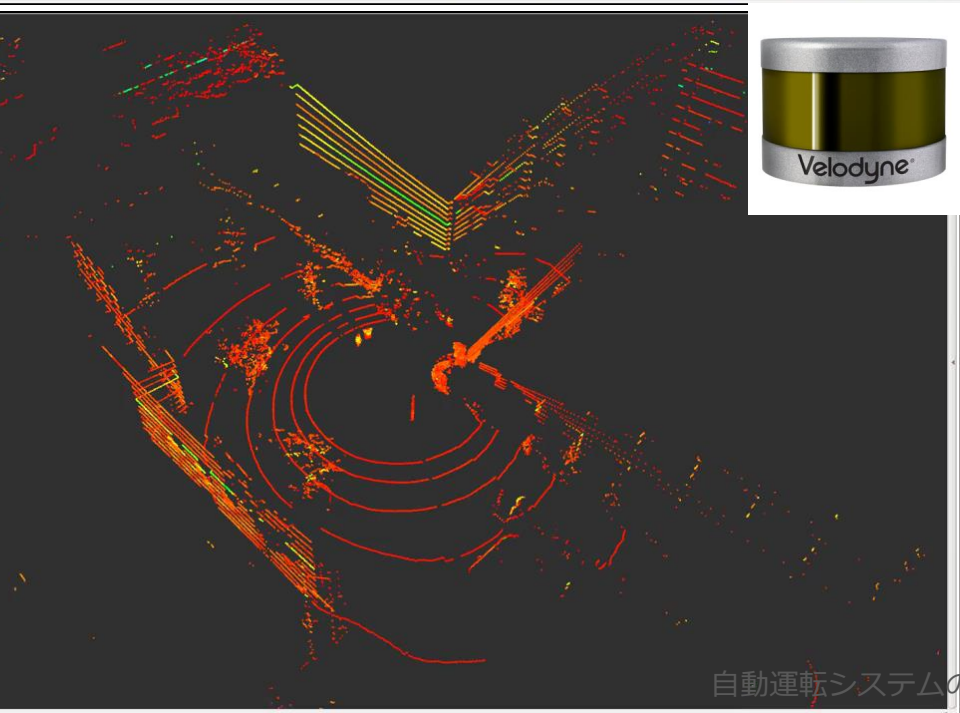
Autowareで動作可能なLIDAR

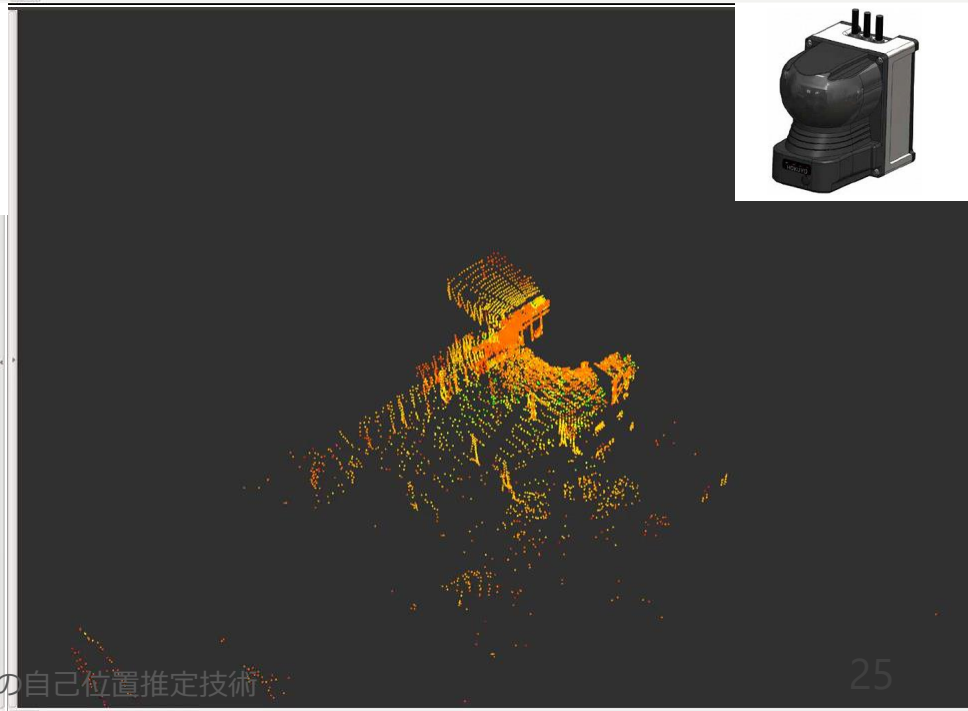
（2016年10月現在）

	 Velodyne HDL-64e	 Velodyne HDL-32e	 Velodyne VLP-16	 Hokuyo 3D-URG
測定距離	~120m	~70m	~100m	~50m
水平視野角	360°			210°
垂直視野角	26.8° (+2°~-24.33°)	41.3° (+10.67°~-30.67°)	30° (+15°~-15°)	40° (+35°~-5°)
測定ポイント数	1,333,000 ポイント/秒	700,000 ポイント/秒	300,000 ポイント/秒	10,360 ポイント/秒
価格	$80,000	$30,000	$8000	$5,000





29 fps
Reset


Velodyne


30 fps
Reset

NDT スキャンマッチング

NDT – Normal Distributions transform

NDT スキャンマッチングのアルゴリズム

1. モデルを一定の大きさのセルに分割
2. 各セルの平均・分散を計算

平均

$$\boldsymbol{q} = \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{x_k}$$

分散

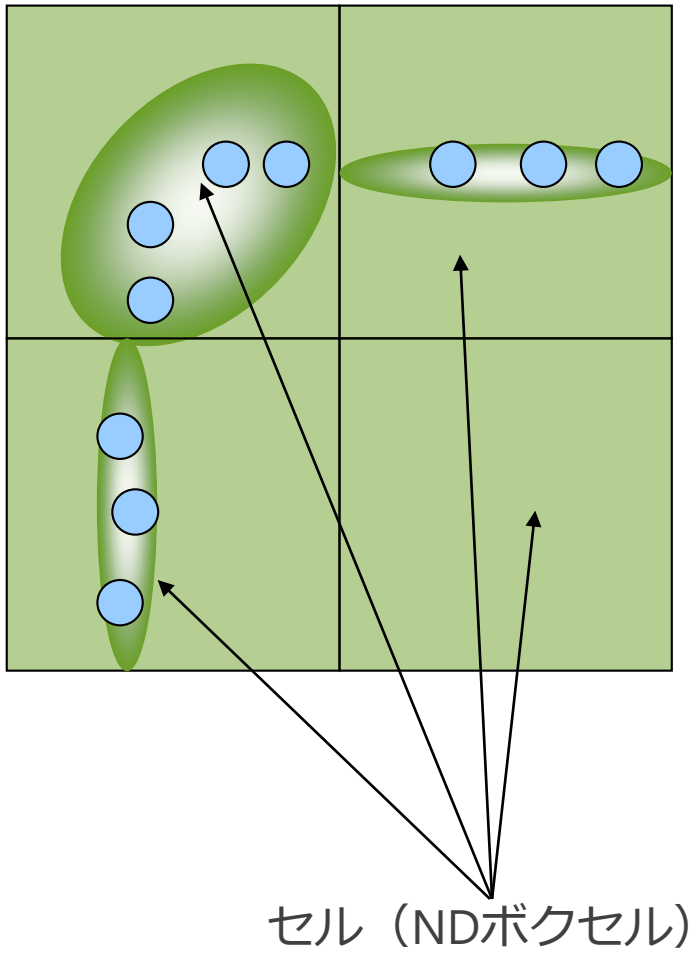
$$\boldsymbol{C} = \frac{1}{n-1}\sum_{k=1}^{n} (\boldsymbol{x_k} - \boldsymbol{q})(\boldsymbol{x_k} - \boldsymbol{q})^T$$

確率密度関数 (PDF*)

$$p(\boldsymbol{x}) = \frac{1}{c}\exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{q})^T \boldsymbol{C}^{-1} (\boldsymbol{x} - \boldsymbol{q})}{2}\right)$$

n : セル内に含まれるポイント数

$\boldsymbol{x}_{k=1,\ldots n}$: セル内に含まれるポイント



* Probability Density Function

NDT スキャンマッチング

3. 入力スキャンの各点に対応する要素を求める
4. 評価値を計算
5. ニュートン法により、入力スキャンの座標変換値を更新

評価関数 $$s(\boldsymbol{p}) = -\sum_{k=1}^{n} p(T(\mathbf{p}, \boldsymbol{x}_k))$$

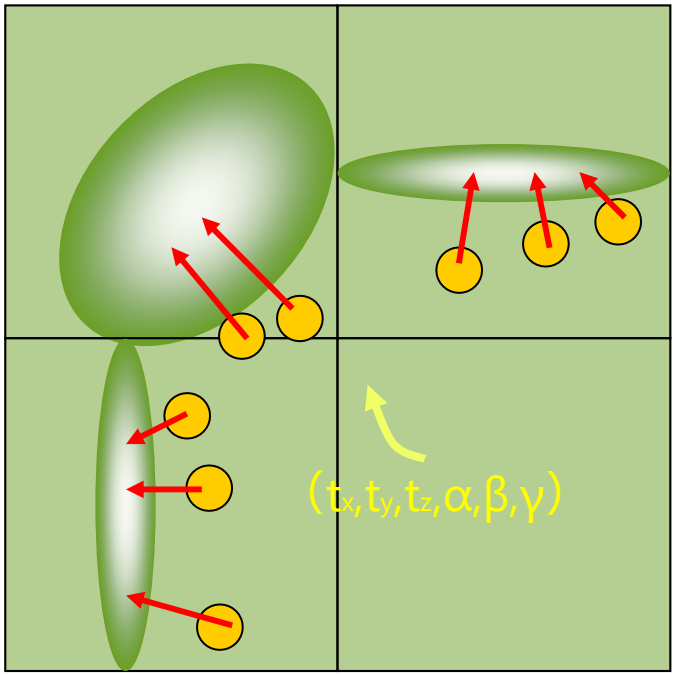
n : セル内に含まれるポイント数

$\mathbf{p}$: 位置・姿勢

$\boldsymbol{x}_{k=1,\ldots n}$: セル内に含まれるポイント

$T(\mathbf{p}, \boldsymbol{x}_k)$: 座標変換後のポイント

6. 3-5 を収束するまで繰り返し

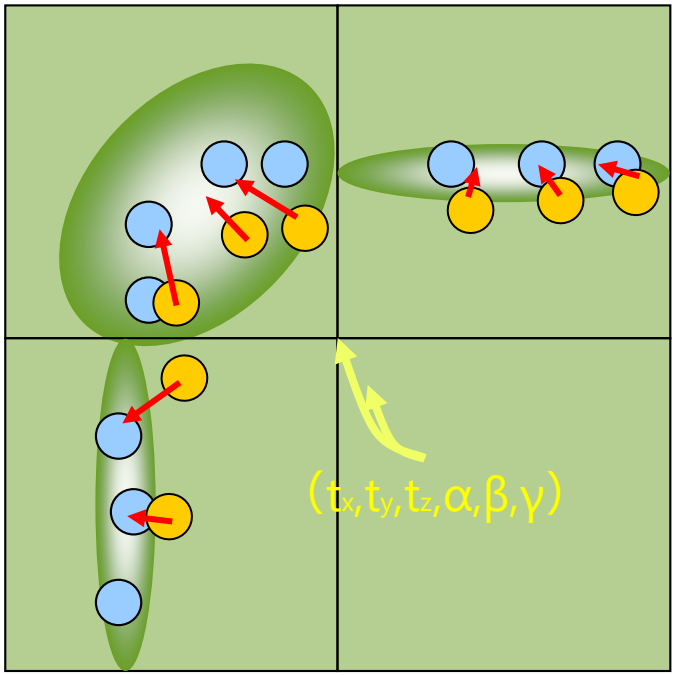


NDT スキャンマッチング

3. 入力 スキャンの各点に対応する要素を求める
4. 評価値を計算
5. ニュートン法により、入力スキャンの座標変換値を更新

評価関数 $$s(\boldsymbol{p}) = -\sum_{k=1}^{n} p(T(\boldsymbol{p}, \boldsymbol{x}_k))$$

6. 3-5を収束するまで繰り返し



計算量：スキャンデータに依存（地図データに依存しない）

Takeuchi Eijiro, and Takashi Tsubouchi.
"A 3-D scan matching using improved 3-D normal distributions transform for mobile robotic mapping." Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on. IEEE, 2006.

（補足）ICP スキャンマッチング

ICP – Iterative Closest Point

ICPスキャンマッチングのアルゴリズム

1. 2つのスキャン の 最近傍点（Nearest Neighbor）を求める
2. NN間の距離の和を最小化
   - 評価関数を最小化する座標変換 (T (並進) , R (回転)) を反復的に計算

地図・スキャン全ての点について計算

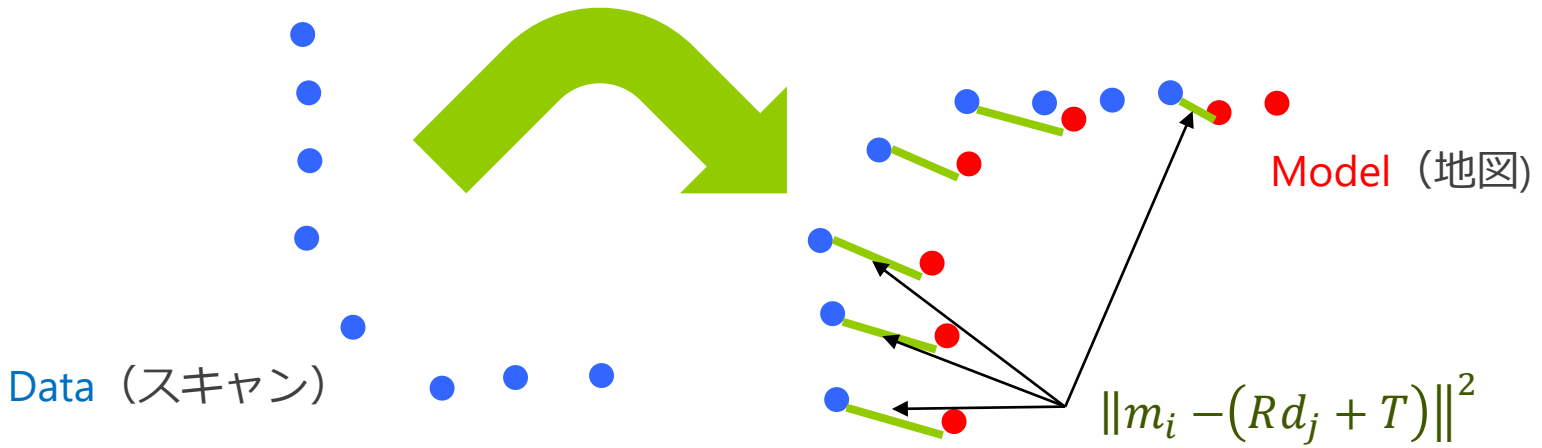
iとjが対応点であればw_{ij}=1、対応点でなければw_{ij}=0

$$E(R,T)=\sum_{i=1}^{N_m}\sum_{j=1}^{N_d} w_{ij}\left\|\boldsymbol{m_i}-\left(R\boldsymbol{d_j}+T\right)\right\|^2$$

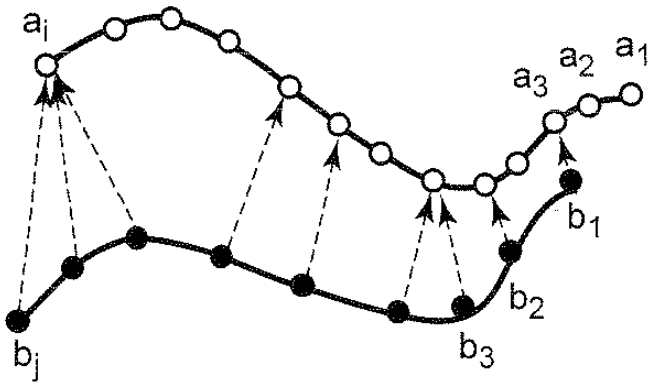
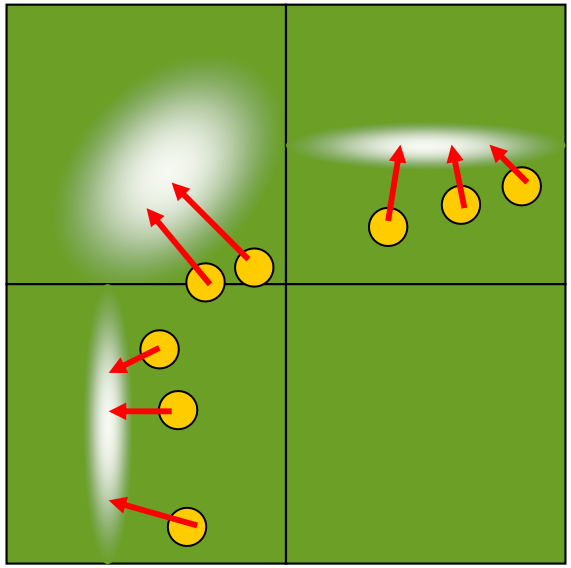
座標変換後のスキャン

NN間の距離(の2乗)

並進:T 回転:R



（補足）ICPとNDTの比較

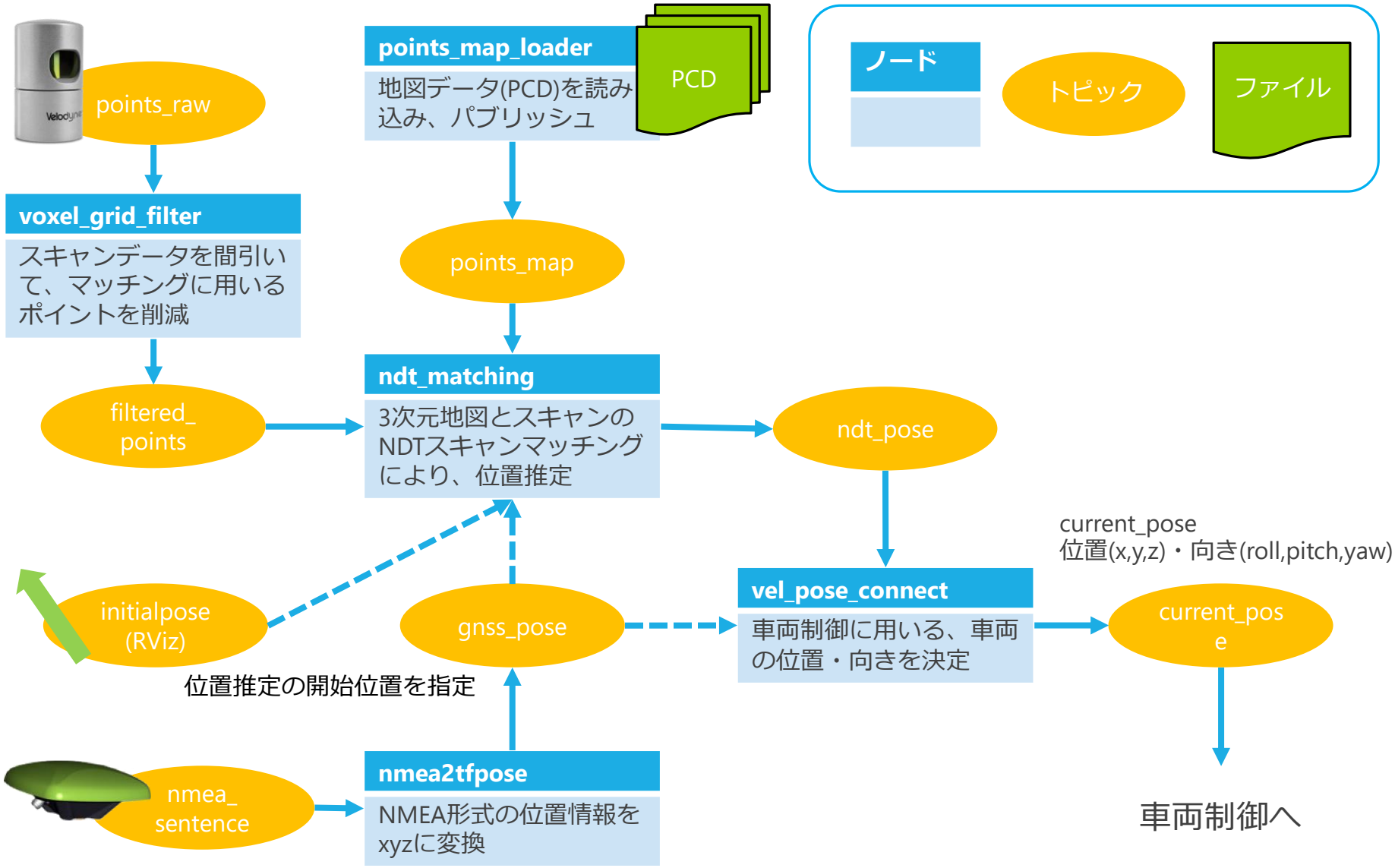
	ICP (Iterative Closest Points)	NDT (Normal Distributions Transform)
計算量 M: 地図 N: スキャン	O(MN) (O(N log M) – KD-treeを用いた場合) 地図とスキャンのポイント数に依存	O(N) 地図のポイント数には依存しない
アルゴリズム	最近傍点間の2乗和を最小化	地図空間を正規分布で近似し、入力スキャンの対応要素を探索
		

自動運転システムの自己位置推定技術

第３章： Autoware の自己位置推定システム

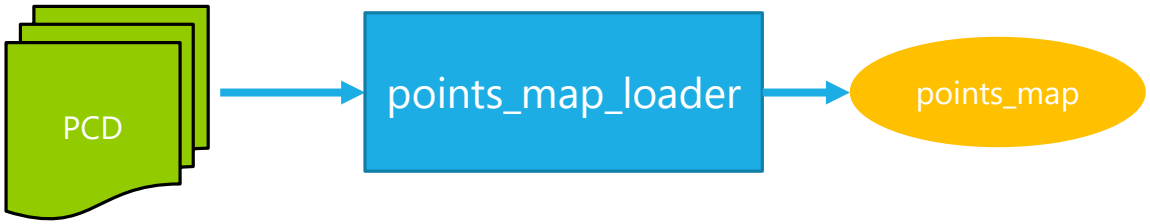
1. Autowareの自己位置推定
2. Autowareでの位置推定の実装

ノード構成 全体図


points_raw
voxel_grid_filter
スキャンデータを間引いて、マッチングに用いるポイントを削減
filtered_points
points_map_loader
地図データ(PCD)を読み込み、パブリッシュ
PCD
points_map
ndt_matching
3次元地図とスキャンのNDTスキャンマッチングにより、位置推定
ndt_pose
initialpose (RViz)
位置推定の開始位置を指定
gnss_pose
vel_pose_connect
車両制御に用いる、車両の位置・向きを決定
current_pose
current_pose
位置(x,y,z)・向き(roll,pitch,yaw)
車両制御へ
nmea_sentence
nmea2tfpose
NMEA形式の位置情報をxyzに変換
ノード
トピック
ファイル

Points_Map_Loader

（複数の）PCDファイルを読み込み、points_mapトピックにパブリッシュ



PCD（Point Cloud Data）フォーマット

- PCL の標準フォーマット
- 多様な形式をサポート

  XYZ型, XYZRGB型（XYZ+色）,

  XYZI型（XYZ+反射強度）, etc.
- ASCII / Binary の2種類

  Binary は ASCII より保存・読み込みが高速

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 299939
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 299939
DATA ascii
-92770.922 -16333.243 109.088 2.3509886e-38
-92771.492 -16331.994 108.753 1.2471689e-38
-92771.805 -16332.02 108.843 6.0849158e-39
-92772.094 -16332.278 109.014 6.1893938e-39
-92772.375 -16332.604 109.211 1.9345711e-38
-92772.727 -16332.418 109.229 9.120906e-39
…
```

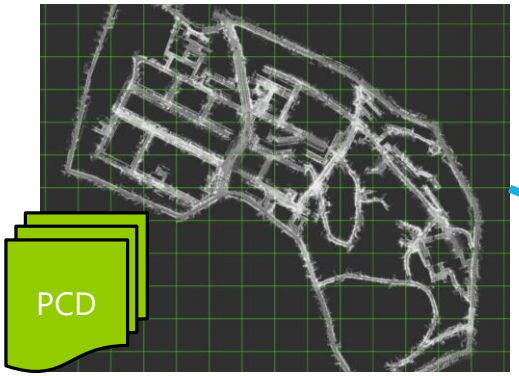
ヘッダ
(形式、データ数など)

データ
(1行1ポイント)

Points_Map_Loader

地図データの部分読み込み

自車位置周辺のPCDファイルのみをパブリッシュし、表示処理の軽量化



PCDファイル：1つのPCDは100m×100m

arealists.txt
filename, min_x,y,z,
max_x,y,z
…

arealistsファイル：PCDファイル名と
xyz座標の最小値・最大値

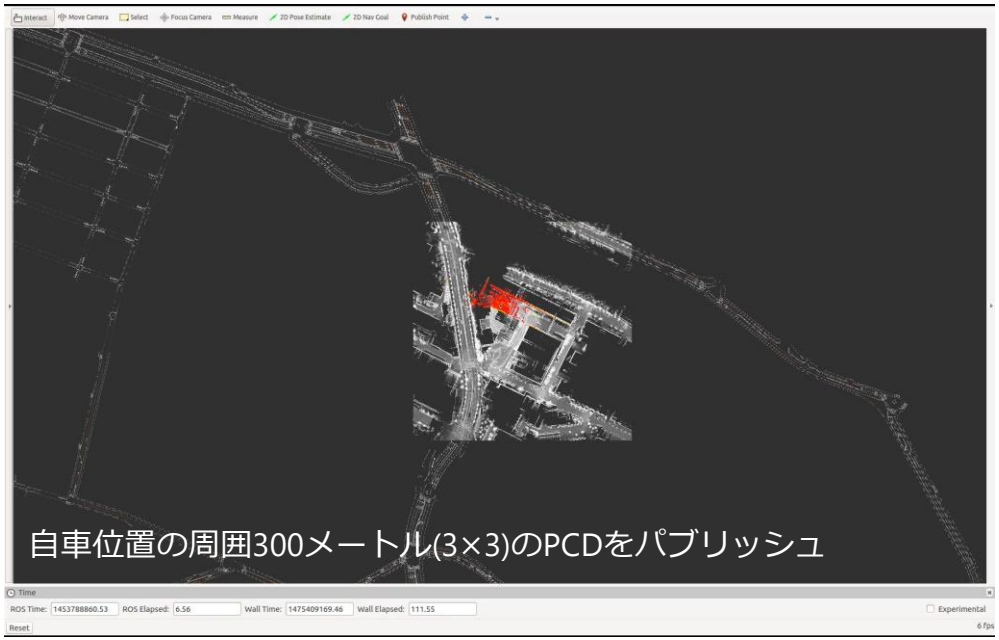
current_pose

current_pose：
ndt_matchingやcurrent_poseから得られる自車位置

current_poseとarealistsファイルから、自車周辺に対応するPCD
ファイルを探索
周辺何メートル読み込むかは指定可能 (1×1, 3×3, 5×5, 7×7)

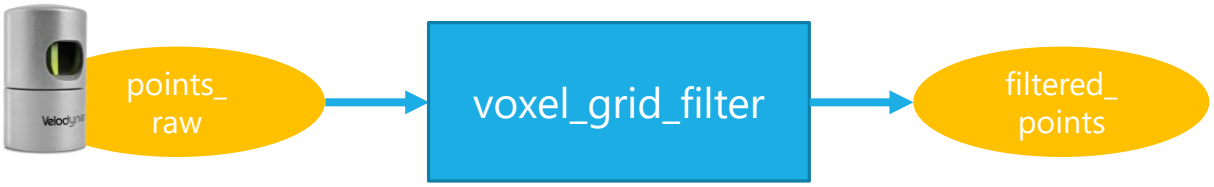
points_map_loade
r

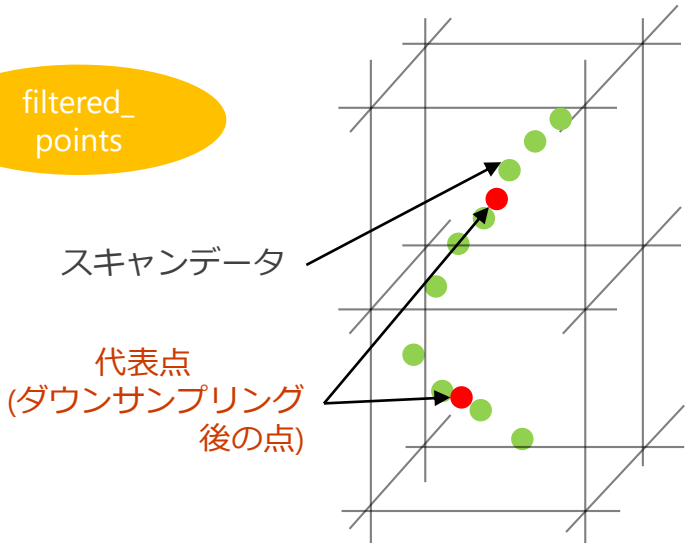
points_map



Voxel_Grid_Filter

LIDAR の スキャンデータ を ダウンサンプリング

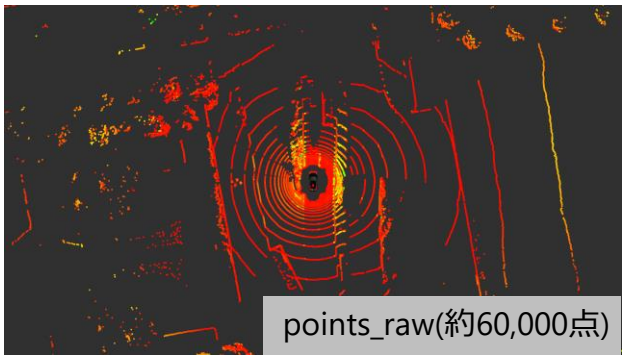


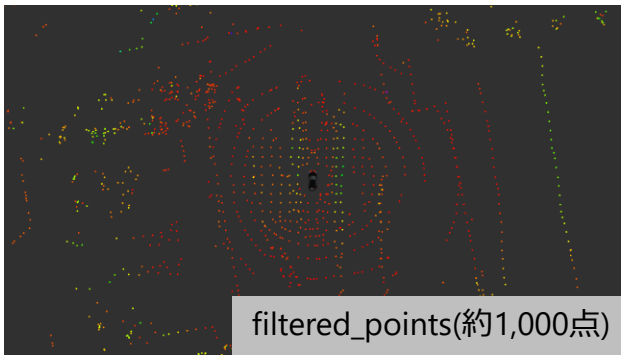


- ✓ スキャンデータを一定の大きさのボクセルに分割
- ✓ 各ボクセルに属するポイントの重心1点に置き換え

ダウンサンプリングを行う理由

- ✓ ポイント数を削減して、マッチング計算の高速化
- ✓ 地図にない未知物体(他車両など)のマッチングへの影響緩和





nmea2tfpose

GNSSで取得されるNMEAセンテンス（緯度・経度・標高）をXYZに変換

*NMEA（National Marine Electronics Association）フォーマット

- ✓ 時刻、緯度・経度、測位品質、衛星数、衛星ID等が分かる

$GPGGA, 052953.000, 3538.9921, N, 13924.1102, E, 1, 8, 1.12, 133.6, M, 39.3, M, , * 51

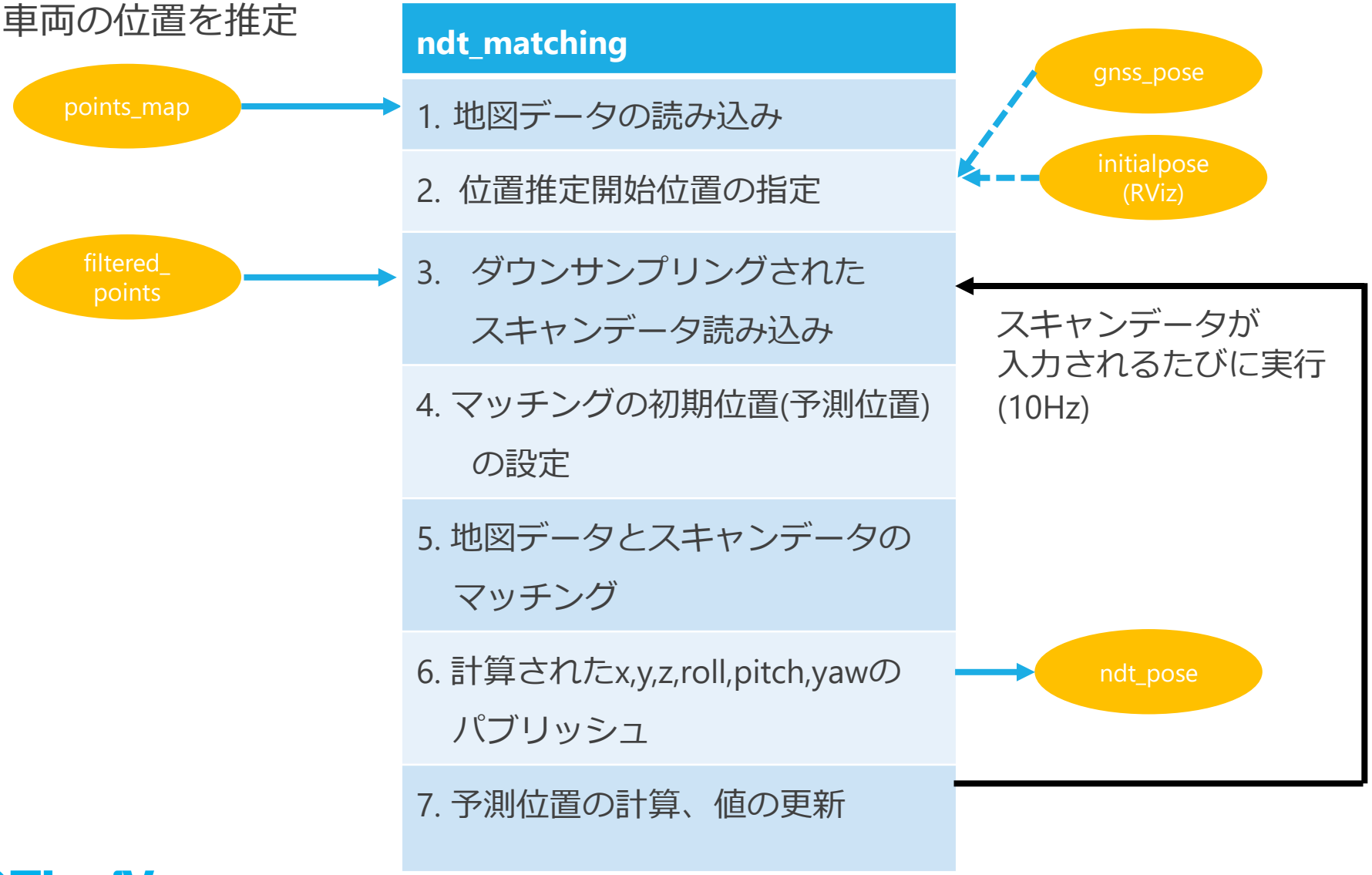
緯度　　経度　　衛星数

$GPGSA, A, 3, 25, 12, 14, 22, 18, 09, 27, 15, , , , , 1.44, 1.12, 0.91 * 09

測位利用衛星ID

NDT_MATCHING

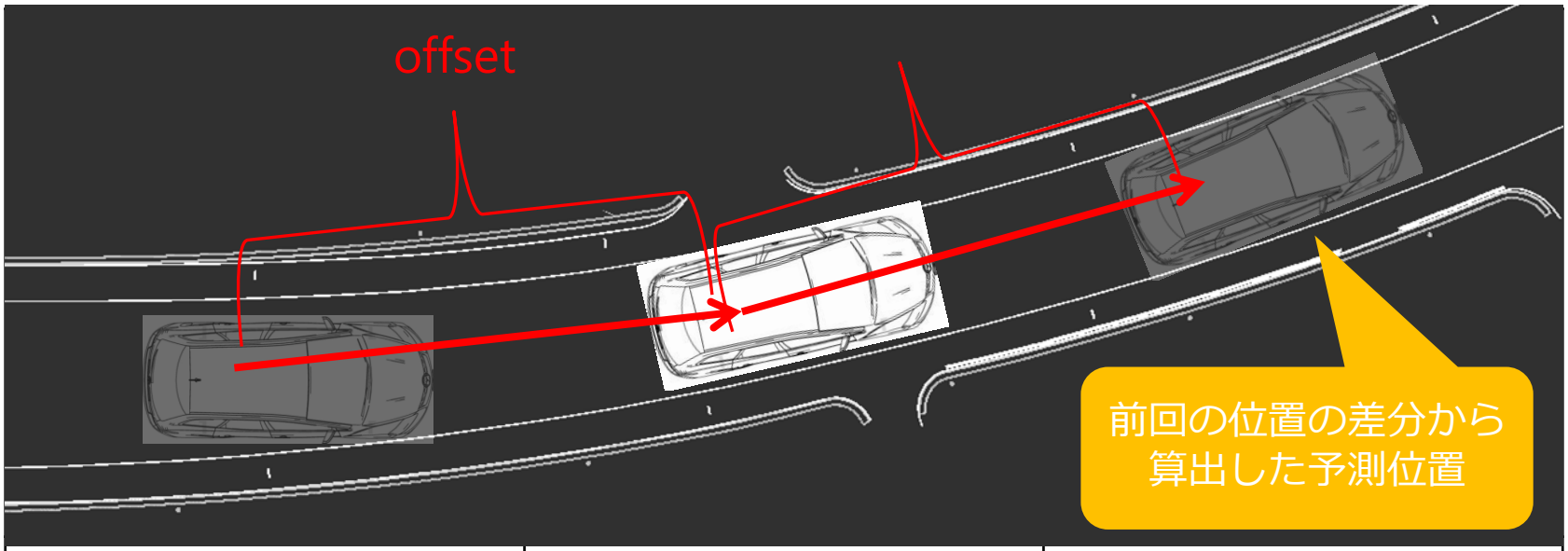
3次元地図とLIDARのスキャンデータのNDTスキャンマッチングにより、
車両の位置を推定



NDT_MATCHING

マッチング探索範囲の限定

- スキャンマッチングは、精度の良いマッチング初期位置を与えることで、収束までの反復計算を減らすことが可能
  →過去2スキャンで得られた位置・向きの差分から、次のスキャンのマッチングの位置・向きを線形補間することで予測



t-1	t	t+1
previous_pose	current_pose offset = current_pose-previous_pose	predict_pose = current_pose+offset

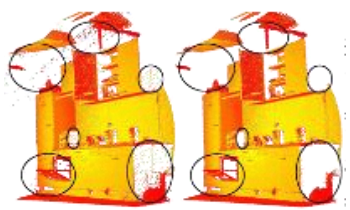
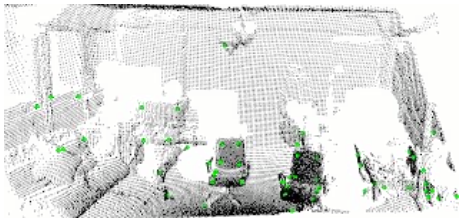
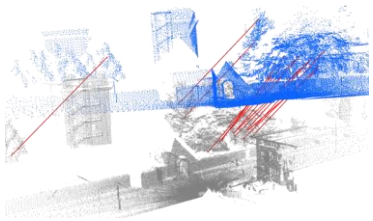
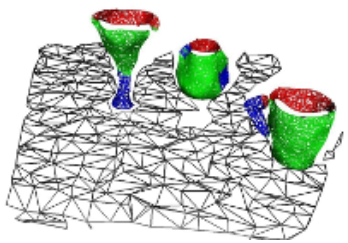
参考： PCL

PCL – pointcloudlibrary

ndt_matching 等の実装には PCL の関数を使用

- 2次元 / 3次元点群処理のためのオープンソースなライブラリ、ツール群
- ROSと強力な連携
- 様々な点群処理の機能をサポート

フィルター　特徴点　位置合わせ　表面処理

・・・

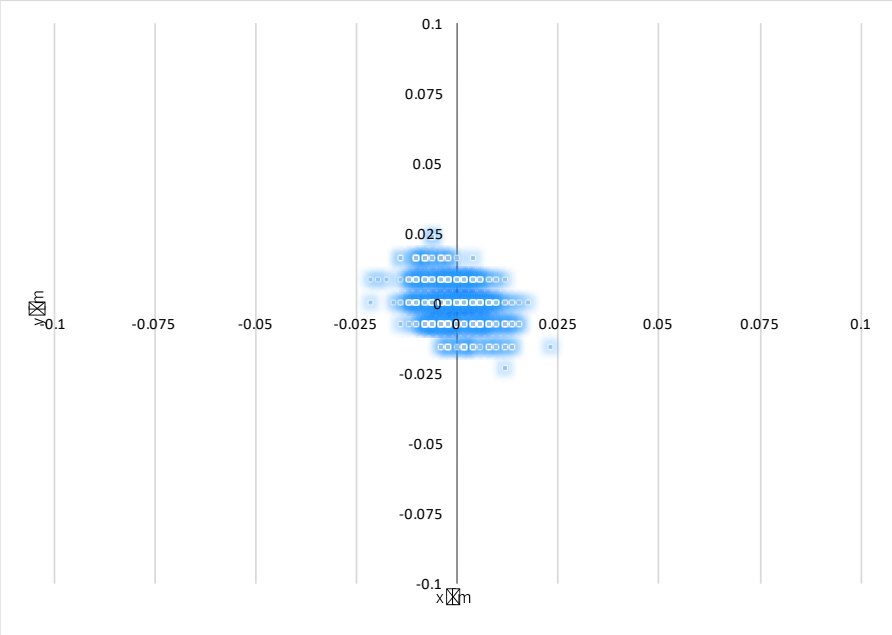
例：ndt_matching.cpp (一部)

```
#include <pcl/registration/ndt.h>

static pcl::NormalDistributionsTransform<pcl::PointXYZ, pcl::PointXYZ> ndt;

ndt.setInputTarget(map_ptr); // 地図データの読み込み

ndt.setInputSouce(filtered_scan_ptr); // スキャンデータの読み込み
ndt.align(output_cloud, init_guess); // マッチング計算
```

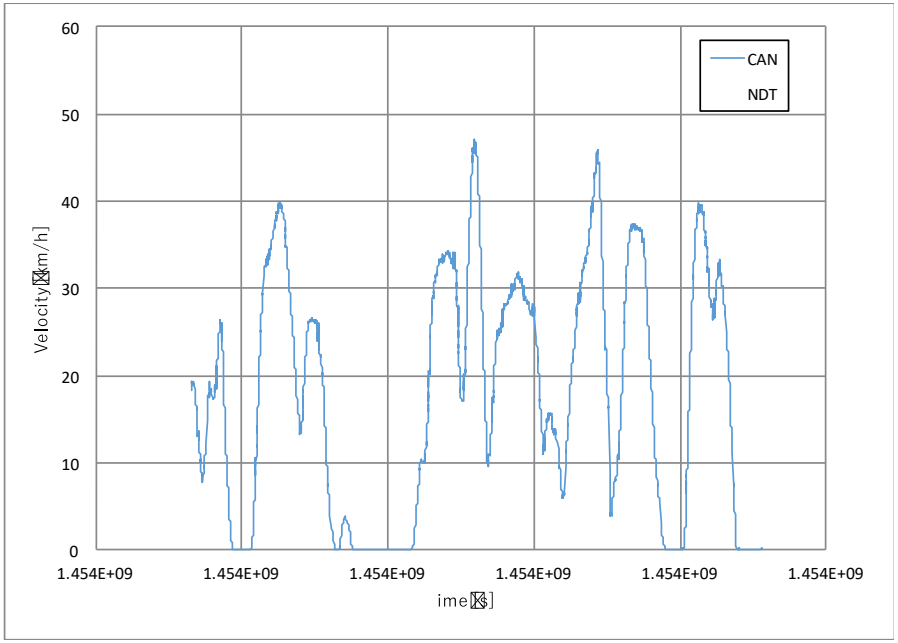
位置推定の様子

Ndt_matching 評価

位置推定精度 - 10cm以内



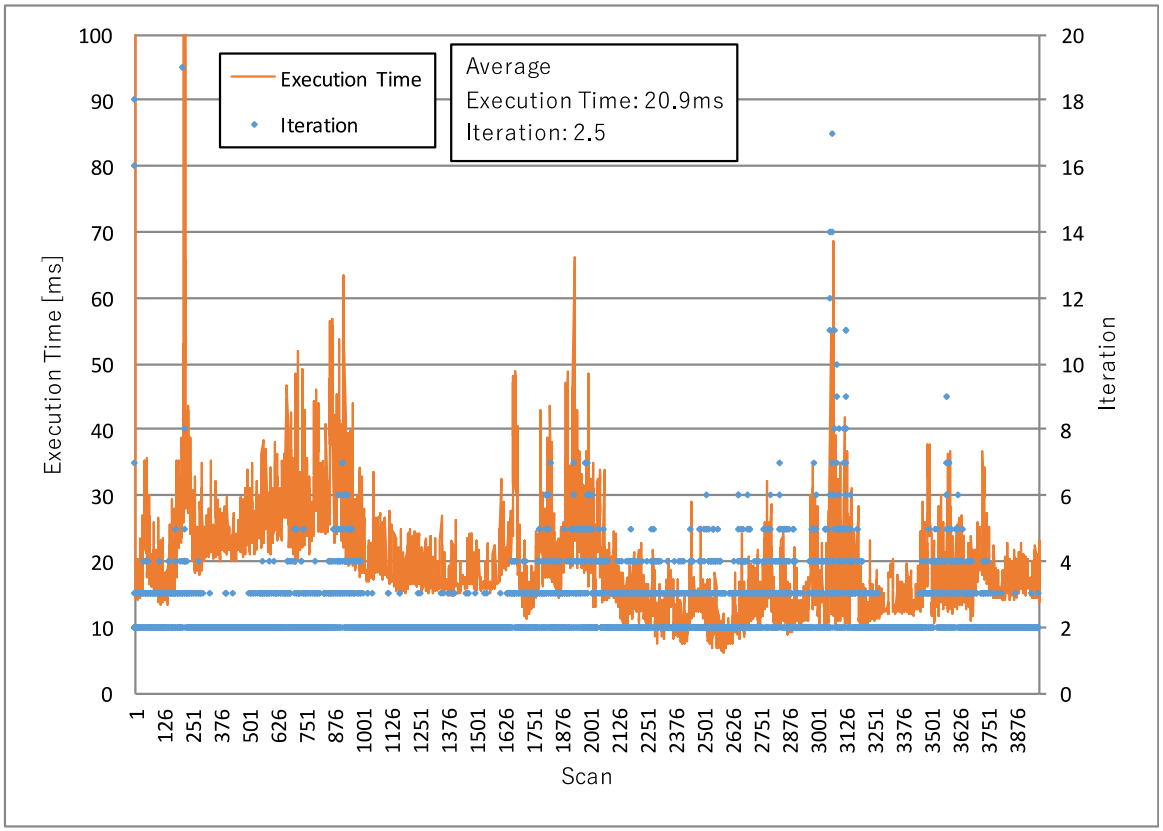
停車時の推定位置の分散
- xyともに5cm以内に収まっている



CANから得られる速度とNDTによる位置推定から計算される速度の比較
- NDTによる位置推定が正確なため、車速も正確に計算可能

Ndt_matching 評価

計算時間 – 30ms以内 （ LIDARの計算間隔内での計算が可能）



位置推定の計算時間と計算収束までのイテレーション数の推移
（横軸：スキャン、縦軸：計算時間）

- 走行の最中、各スキャンに対して　100ms以内の位置推定が可能

自動運転システムの自己位置推定技術

第４章：まとめ

まとめ (1/2)

- 自己位置推定とは
  - 自動運転の位置推定システムには精度・リアルタイム性・ロバスト性が求められる
- 車両の自己位置推定手法
  - デッドレコニング
    - IMUやホイールエンコーダを用いた逐次的位置推定
    - 誤差の蓄積が問題 -> デッドレコニング単体では位置推定が困難
  - GNSS
    - 各国の測位システム、衛星群
    - マルチパスにより1~10m程度の誤差
  - スキャンマッチング
    - 地図データとLIDARのスキャンデータのマッチング
    - 高精度な地図データが不可欠

まとめ (2/2)

- Autowareの位置推定システム
  - 高精度3次元地図
    - MMS（Mobile Mapping System）により計測
    - ポイントクラウド地図/ADAS地図
  - LIDAR
  - NDTスキャンマッチング/ICPスキャンマッチング
    - ICPは地図のデータ量、スキャンのデータ量に依存するが、NDTはスキャンのデータ量のみに依存
  - ノード構成
    - 各機能毎にノード化 points_map_loader/voxel_grid_filter/nmea2tfpose/ndt_matching
    - 精度・計算時間評価 - 自動運転に必要な精度・リアルタイム性を満足

www.tier4.jp

自動運転システムの自己位置推定技術

Appendix

参考文献

- 測位衛星技術株式会社 「GNSSの基礎知識」 Version 1.0 http://gnss.co.jp/gnss_basic
- Borrmann, Dorit, et al. "Globally consistent 3D mapping with scan matching." *Robotics and Autonomous Systems* 56.2 (2008): 130-142.
- P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, Feb 1992.
- Biber, Peter, and Wolfgang Straßer. "The normal distributions transform: A new approach to laser scan matching." *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. Vol. 3. IEEE, 2003.
- Takeuchi, Eijiro, and Takashi Tsubouchi. "A 3-D scan matching using improved 3-D normal distributions transform for mobile robotic mapping." *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006.
- Magnusson, Martin, Achim Lilienthal, and Tom Duckett. "Scan registration for autonomous mining vehicles using 3D-NDT." *Journal of Field Robotics* 24.10 (2007): 803-827.
- Point Cloud Library http://pointclouds.org/