点群合わせ処理

ICPからGICP・NDTへ ― 点群剛体変換アルゴリズムの進化と実践

#LiDAR #SLAM #点群処理 #自己位置推定

Nyanziba

本日の流れ

1. **導入** — LiDAR・点群・SLAMとは何か
2. **ICP** — 素朴な定式化と限界
3. **GICP** — 共分散を使った統計的改善
4. **NDT** — 点対分布という新しい表現モデル
5. **VGICP** — GICPの精度 × NDTの速度
6. **比較・まとめ** — 4アルゴリズムの使い分け

Part 1：導入

LiDAR・点群・SLAM

点群とは？

3次元空間における点の集合

- 各点が **(x, y, z)** の座標を持つ
- 反射強度、RGB等のデータを持つ場合もある

2次元の画像との違い：

- **順序に意味がない**（画像のようなピクセル格子がない）
- **密度が不均一**（センサーに近い部分は密、遠い部分は疎）
- **ノイズと外れ値が多い**
- **データ量が大きい**（1スキャンで数万〜数十万点）

取得するための機器

LiDAR

Light Detection and Ranging

- レーザーを照射して反射までの時間から距離を計算（ToF形式）
- 精度は **±1〜3cm**
- ロボティクス、自動運転で使用

深度カメラ

RGB-D Camera

- ToF形式 or **構造化光形式**（赤外線パターンの歪みから距離を計算）
- 近距離(〜2m)で **±1〜2mm**
- 10m以上はそもそも測定できない

点群の使用用途は？ (1/2)

🔍 物体検出

- 自動運転での歩行者・車両・自転車のリアルタイム検出
- Amazonの倉庫ロボットによる商品のピッキング
- 港湾クレーンのコンテナ自動積み下ろし

✂️ セグメンテーション

- 自動運転の走行可能領域の推定（地面・壁・障害物を分離）
- 森林調査での樹木1本1本の自動識別・材積量計測
- 建物・道路・植生などの地物分類（航空LiDAR測量）

点群の使用用途は？ (2/2)

📍 位置合わせ

- SLAMによる自己位置推定（ロボット・自動運転）
- 工場での製品とCADモデルの寸法検査
- 建設現場の施工前後の地形比較

🗺️ 3D地図構築

- 自動運転用HDマップ（高精度地図）の作成
- 文化財のデジタルアーカイブ
- ゲーム・映画VFXの都市モデル生成

> 今回は **位置合わせ** と **3D地図構築** で使用されるアルゴリズムについてLTします

SLAMとは

Simultaneous Localization and Mapping

地図生成と自己位置推定の同時解決

Localization（自己位置推定）

「自分が地図のどこにいるか」を推定する

→ **点群マッチングが核心！**

Mapping（地図生成）

「周囲の環境地図」を同時に構築する

→ 累積した点群を統合

地図がないと位置がわからない、位置がわからないと地図が作れない → SLAMはこれを同時に解く

点群の「位置合わせ」問題

時刻 t の点群 P（ソース）と、時刻 t+1 の点群 Q（ターゲット）が与えられたとき

最適な剛体変換 T = (R, t) を求めよ

- **R**：3×3 回転行列 — どの方向にどれだけ回転したか
- **t**：3次元並進ベクトル — どの方向にどれだけ移動したか

> これを解くのが **ICP / GICP / NDT** の役割

Part 2 : ICP

Iterative Closest Point

ICP — 処理の流れ

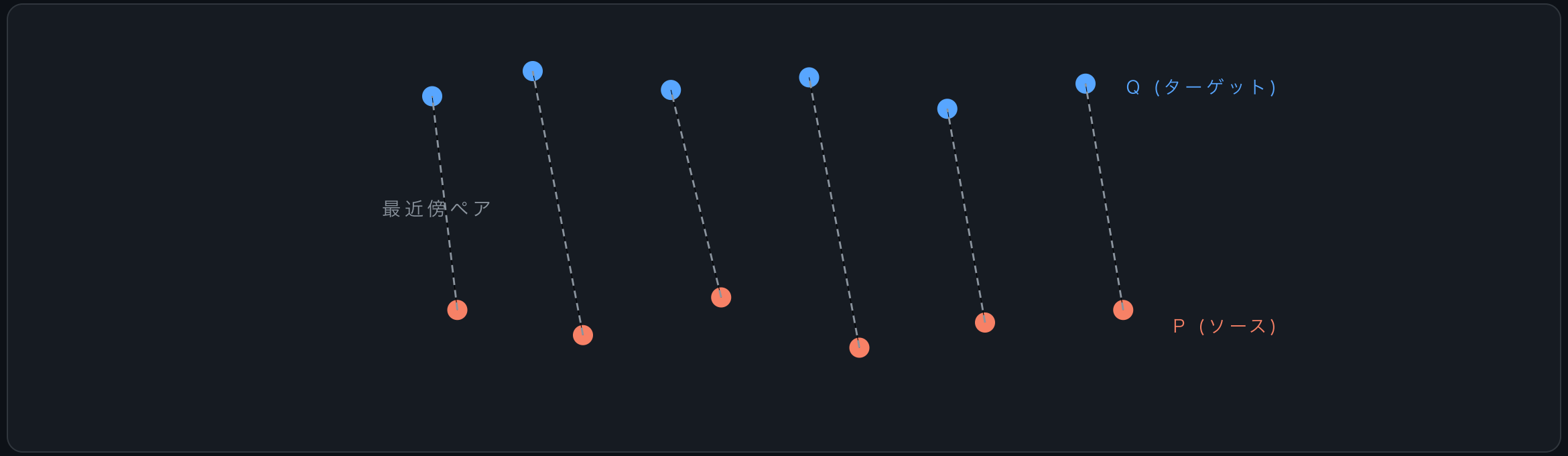
繰り返しによって2つの点群を段階的に重ね合わせていく

収束していなければ ① に戻る

- **① 最近傍対応付け** ： 各 p_i に対して最も近い q_i をペアにする
- **② 変換の最適化** ： ペア間の距離を最小化する (R, t) を計算
- **③ 点群の更新** ： ソース点群を (R, t) で変換
- **④ 収束判定** ： 変換量が閾値以下なら終了

ICP — 最近傍対応付けの図示

各ソース点 p_i に対して、ターゲット内で最も近い点 q_i をペアにする



この「ペアの距離の合計」を最小化する変換を求めるのがICPの目的関数

ICP — 目的関数

点間ユークリッド距離の二乗和を最小化する

$$T^* = \arg\min_{R,t} \sum_i \| q_i - (R p_i + t) \|^2$$

- p_i：ソース点群の各点
- q_i：ターゲット点群の最近傍点
- R, t：最適化したい剛体変換
- $\| \cdot \|^2$：ユークリッド距離の二乗

✦ 閉形式解（SVD分解）で R と t を解析的に求めることができる

ICP — 収束のイメージ

反復ごとにソース点群がターゲットに近づいていく

繰り返しのたびに対応付けが更新され、変換がより正確になっていく

ICPの課題

課題1：外れ値に弱い

$\|q_i - Rp_i - t\|^2$ はノイズ点の誤差を等しく扱う

→ **1点の外れ値が最適化を大きく歪める**

課題2：初期値依存・局所解問題

最近傍対応付けは局所的な操作

→ **初期位置がずれていると間違った局所解に収束**

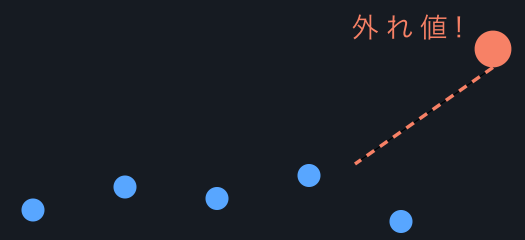
課題3：点対点の対応が「面」を無視する

平面上に並んだ点群を孤立した点として扱う

→ **面・エッジの幾何構造を活かせない**

ICPの課題 — 外れ値と面構造の図示

外れ値の影響



1点の外れ値が全体の
最適化を引っ張ってしまう

面構造の無視

面上の点は隣の点にも
対応できてしまう

→ 面構造を活かせない

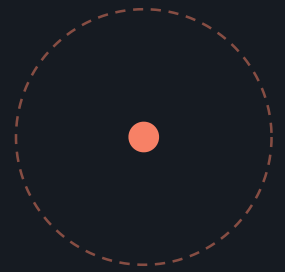
Part 3 : GICP

Generalized ICP

GICP — アイデア

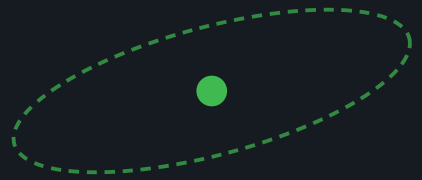
「各点を "点" ではなく "局所的な分布" として扱う」

ICPの誤差モデル



全方向の誤差を均等に扱う

GICPの誤差モデル



面に沿った方向の誤差は軽く扱う

共分散行列 C_i は各点の近傍 k 点から局所PCAで推定する

GICP — 共分散行列の意味

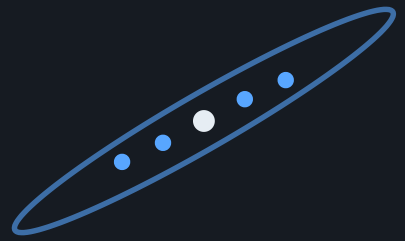
近傍点の分布から「面かエッジか孤立点か」を判別できる

平面上の点

$\lambda_1 \approx \lambda_2 \gg \lambda_3$

法線方向の分散が小さい

エッジ上の点



$\lambda_1 \gg \lambda_2 \approx \lambda_3$

1方向にだけ分散が大きい

孤立した点

$\lambda_1 \approx \lambda_2 \approx \lambda_3$

全方向に均等（ICPと同じ）

$\lambda_1, \lambda_2, \lambda_3$ は共分散行列の固有値（PCAの主成分）

GICP — 目的関数

マハラノビス距離による重み付き最小化

$$T^* = \arg\min_{R,t} \sum_i d_i^T \left(C_i^Q + R C_i^P R^T \right)^{-1} d_i$$

$$d_i = q_i - (R p_i + t)$$

- d_i：変換後ソース点とターゲット点の差分
- C_i^Q：ターゲット点の局所共分散行列
- C_i^P：ソース点の局所共分散行列
- $(\cdot)^{-1}$：マハラノビス距離として重み付け

GICP — ICPからの変化を図示

目的関数のどこが変わったのか？

ICP

$$\Sigma \ \| \ d_i \ \|^2$$

ユークリッド距離²

全方向均等に誤差を評価

→

GICP

$$\Sigma \ d_i^T \ (C_i^{\varphi} + R \ C_i^{P} \ R^T)^{-1} \ d_i$$

マハラノビス距離

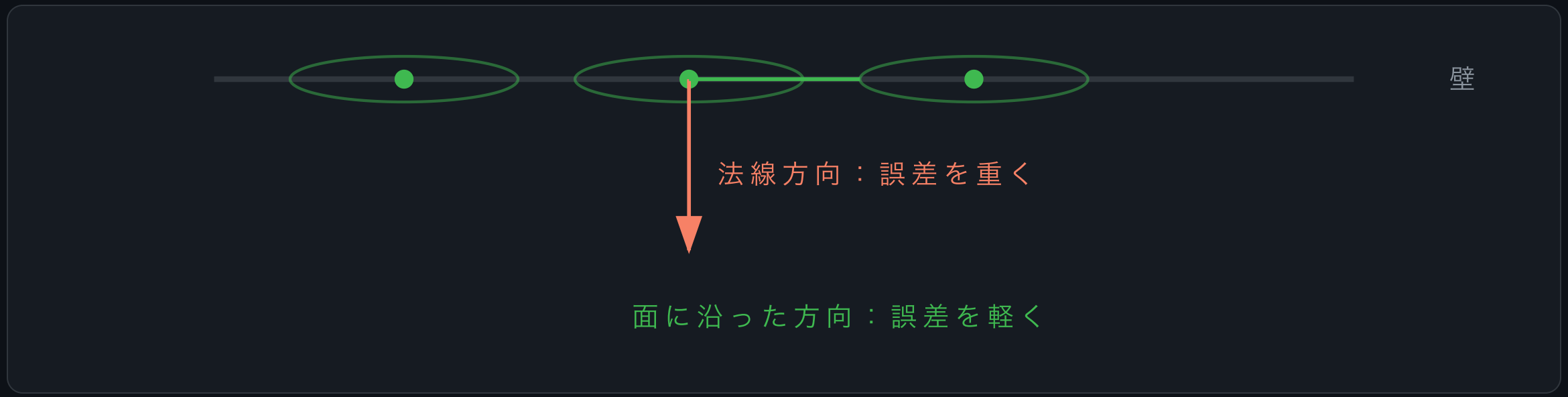
共分散で方向別に重み付け

> 対応付けの枠組みはICPと同じ。**「誤差の測り方」だけを変えた**のがGICP

GICP — 直感的な解釈

壁の面を例に考える

- **面の法線方向**（垂直）→ 共分散が**小さい** → 誤差を**重くペナルティ**
- **面に沿った方向** → 共分散が**大きい** → 誤差を**軽く**扱う



面構造に沿った、幾何学的に意味のある位置合わせが実現する

Part 4 : NDT

Normal Distributions Transform

NDT — アイデアの図示

点の集合 → ボクセルに分割 → 各ボクセルを正規分布でモデル化

NDT — 数式（Step 2：正規分布推定）

各ボクセルの点群から平均と共分散を計算する

$$\mu_k = \frac{1}{n}\sum_j x_j \qquad \Sigma_k = \frac{1}{n}\sum_j (x_j - \mu_k)(x_j - \mu_k)^T$$

- μ_k：ボクセル k 内の点の**平均位置**
- Σ_k：ボクセル k 内の点の**分散共分散行列**
- n：ボクセル k 内の点の数

統計学でおなじみの標本平均・標本共分散そのもの

NDT — 数式（Step 3：スコア最大化）

変換後のソース点が各ボクセルの分布にどれだけ「ハマるか」を最大化する

$$T^* = \arg\max_{R,t} \sum_i \exp\left(-\frac{(T(p_i) - \mu_{k(i)})^T \Sigma_{k(i)}^{-1} (T(p_i) - \mu_{k(i)})}{2} \right)$$

- 正規分布の確率密度関数の形そのもの
- 変換後の点が分布の中心に近いほどスコアが高い
- 最適化には **Newton法 / 勾配降下法** を使用

✦ ボクセルへの割り当てが「対応付け」の役割を果たす

NDT — スコア計算の図示

変換後の点が分布の中心に近いほど高スコア

点が分布の中心に近い
→ exp(…) が大きい → 高スコア

点が分布から離れている
→ exp(…) が小さい → 低スコア

全点のスコア合計を最大化する T を求める

Part 5：VGICP

GICPの精度 × NDTの速度

VGICP — 3つの対応付けモデル

GICPとNDTの弱点を同時に解決するハイブリッド手法

GICP

分布 ↔ 分布（D2D）

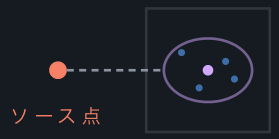
✓ 高精度

✗ NN探索が遅い
✗ GPU不向き

KD木で最近傍探索

NDT

点 ↔ ボクセル分布（P2D）



✓ 高速・GPU向き

✗ 解像度にシビア
✗ 少数点で分布が壊れる

点の位置から分布を推定

VGICP ★

分布 ↔ 集約分布（D2MD）

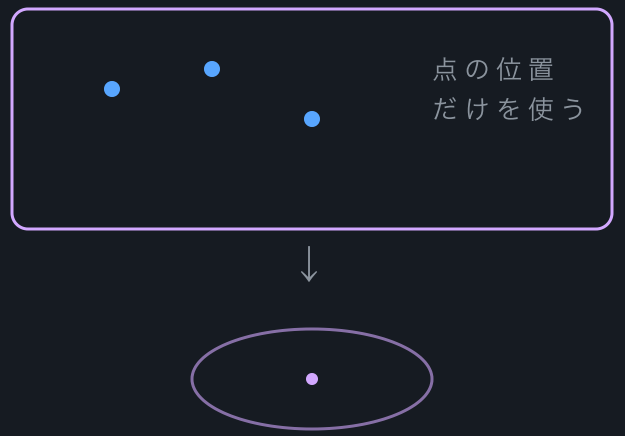
✓ GICPと同等の精度
✓ 高速・GPU向き
✓ 少数点でも安定

各点の共分散を集約して分布を推定

VGICP — ボクセル分布の作り方の違い

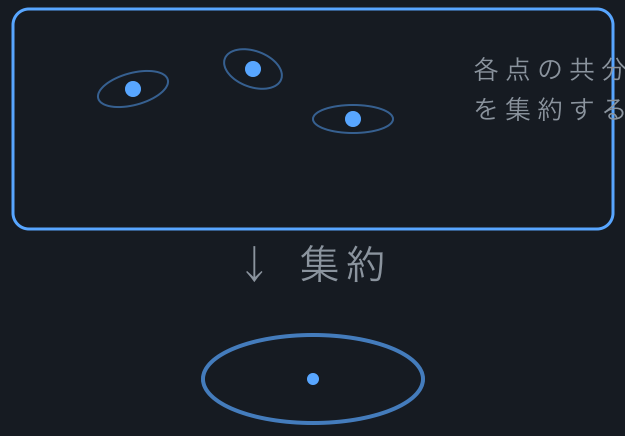
NDTとVGICPでは「ボクセルの分布をどう作るか」が根本的に異なる

NDT のボクセル分布推定



⚠ 点が少ないと分布が壊れる
（3Dでは実用上10点以上必要）

VGICP のボクセル分布推定



↓ 集約

✓ 1点でも有効な分布が得られる
解像度変化にもロバスト

VGICP — 目的関数と性能

GICPの式で b_i, C_i^B をボクセル内の集約値に置き換えるだけ

$$T^* = \arg\min_{R,t} \sum_i N_i \cdot \tilde{d}_i^T \tilde{C}_i^{-1} \tilde{d}_i$$

$$\tilde{d}_i = \frac{\sum_j b_j}{N_i} - T a_i \qquad \tilde{C}_i = \frac{\sum_j C_j^B}{N_i} + T C_i^A T^T$$

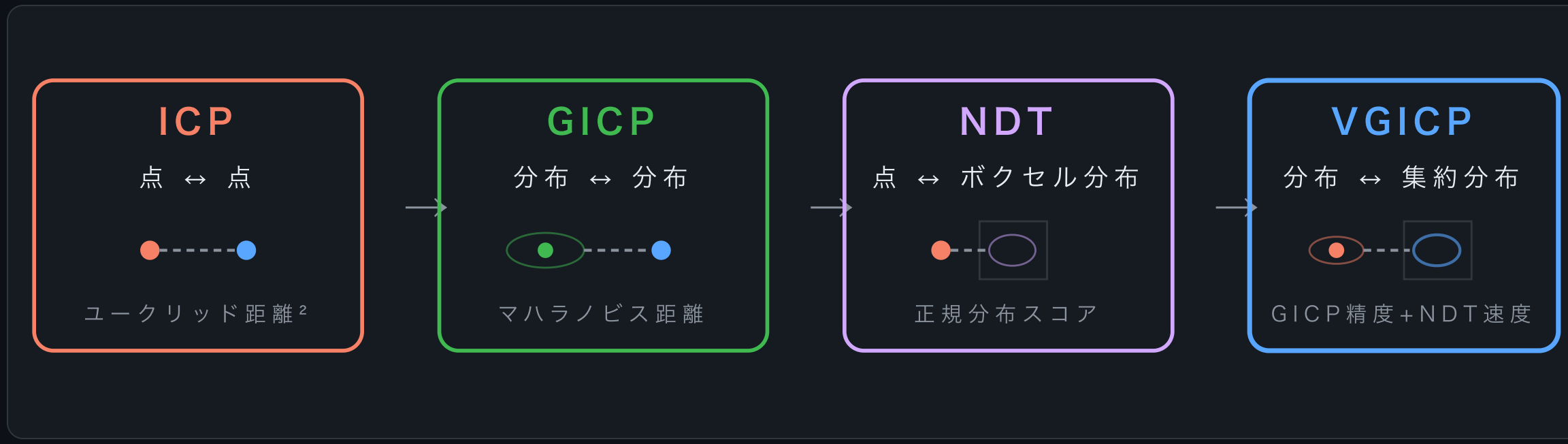
N_i：ボクセル内の点数で重み付け。NN探索不要 → 各点が独立 → GPU並列化が可能

手法	速度（15,000点）	精度	GPU対応
GICP	8 Hz（1スレッド）	◎	✗
NDT	19 Hz	△（解像度依存）	○
VGICP（CPU）	**30 Hz**	◎	—
VGICP（GPU）	**120 Hz**	◎	◎

Part 6：比較とまとめ

4アルゴリズムの使い分け

アルゴリズムの進化の流れ



ICP → GICP → NDT → VGICP は「誤差の測り方」と「対応付けの仕組み」の進化として整理できる

4アルゴリズムの比較表

比較軸	ICP	GICP	NDT	VGICP
対応付け	点↔点	分布↔分布	点↔ボクセル分布	分布↔集約分布
目的関数	ユークリッド距離²	マハラノビス距離	正規分布スコア	マハラノビス距離
外れ値耐性	△	○	◎	○
疎な点群	△	○	◎	◎
速度	低	低〜中	中〜高	**高（GPU 120Hz）**
実装の複雑さ	低	中	中	中〜高

まとめ

ICP は基礎

点間距離^{2}の反復最小化。SVDで解ける閉形式解がある

GICP は統計的拡張

共分散行列でマハラノビス距離を導入し、面・エッジ構造を誤差に織り込む

NDT は表現モデルの変革

対応付けの概念自体を変え、点対分布のマッチングに切り替える

VGICP はGICPとNDTの融合

GICPの共分散集約 + NDTのボクセル対応付けで精度と速度を両立

Thank You

点群処理は「最適化問題」— 他の分野の知識が直結する面白い領域です

ICPからGICP・NDTへ — 点群剛体変換アルゴリズムの進化と実践

Q&A

質疑応答

補足：ボクセルサイズはどう決める？

NDT / VGICPの精度を左右する重要なパラメータ

- **小さすぎる** → ボクセル内の点が少なくなり分布が不安定になる
- **大きすぎる** → 空間の構造が平均化されて位置合わせ精度が落ちる

目安

環境	推奨ボクセルサイズ	理由
屋内（狭い空間）	0.5〜1.0 m	壁や家具の構造を残す
屋外（広い空間）	1.0〜3.0 m	建物や地形の大きな構造に合わせる
高密度点群	小さめでOK	各ボクセルに十分な点が入る
疎な点群	大きめにする	分布推定に必要な点数を確保

VGICPはNDTよりボクセルサイズの変化に対してロバスト

補足：ICPの初期値問題はどう対処する？

実用上の3つのアプローチ

1. 前フレームの結果を初期値にする

SLAMのように連続フレームで使う場合、前回の変換結果を次回の初期値にする。フレーム間の移動量が小さければ十分に収束する

2. IMU（慣性センサ）で初期値を補助する

LIO（LiDAR-Inertial Odometry）ではIMUの角速度・加速度から大まかな姿勢変化を予測し、それを初期値として与える

3. 特徴量ベースの粗位置合わせを先に行う

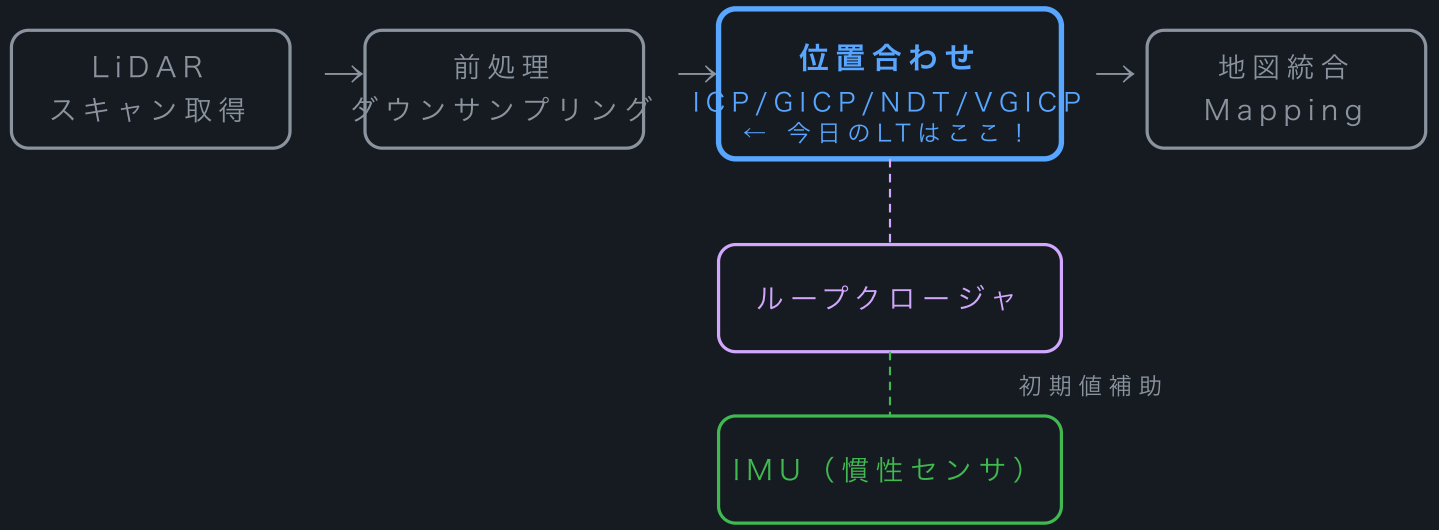
FPFH（Fast Point Feature Histograms）などの3D特徴量でまず大まかに位置を合わせてからICP系の精密位置合わせを行う。2段階アプローチ

補足：どんなライブラリで実装できる？

ライブラリ	言語	特徴
Open3D	Python / C++	GICP実装あり。可視化が簡単
PCL	C++	ICP / NDT 実装あり。ROS連携に強い
small_gicp	C++	koide3作。GICP / VGICP の高速実装
fast_gicp	C++	koide3作。VGICP_CUDA対応

- **Python で試したい** → Open3D がおすすめ。`pip install open3d` で入る
- **ROS で使いたい** → PCL または small_gicp
- **GPU で高速に回したい** → fast_gicp の VGICP_CUDA

補足：SLAM全体のどこに位置するアルゴリズム？



位置合わせはSLAMパイプラインの中核。ここの精度がシステム全体の精度を決める