課題要約：点群レジストレーション

課題内容

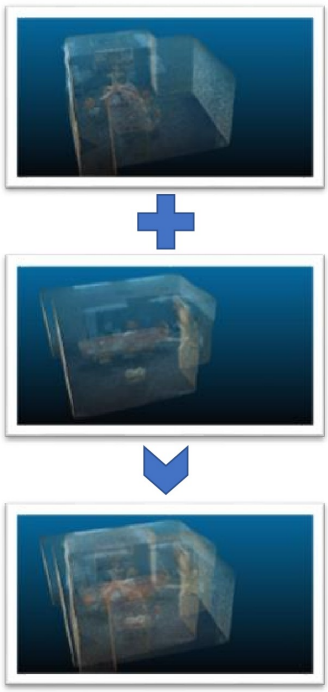
異なる日時に取得した2つの点群データを、座標系が異なる状態から同じ座標系にピッタリ合わせる「レジストレーション」アルゴリズムの実装と評価

点群とは

3D空間上の多数の点（x, y, z座標+色情報）からなるデータで、今回はiPhoneのLiDARで会議室を撮影した約60万点の点群を使用

実装条件

Python（外部ライブラリは numpy, matplotlibのみ）
精度はRMSEで評価（0.1m以下が目標）
poetryやdocstring、型アノテーションも必須

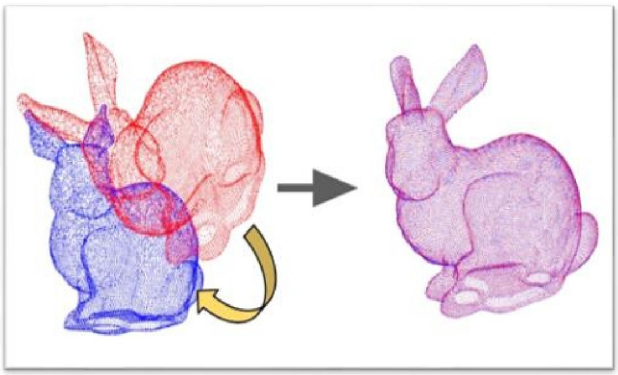


使用技術：レジストレーション方法ICPについて

ICP法の概要	目的	使用例
位置・向きが異なる点群をできる限り一致させる	2つの点群（3D座標群）の位置合わせを行う	3Dスキャンのマージ、地図作成、物体追跡etc

ICPの基本手順

1. 初期位置のまま2つの点群を読み込む
2. 各ソース点に対し、ターゲット点群から最も近い点（最近傍）を見つける
3. 対応点間の誤差を最小化する剛体変換（回転＋並進）を計算
4. ソース点群に変換を適用し更新
5. 誤差（RMSE）の収束を確認し、収束しないなら繰り返す



使用技術：実装と使用ライブラリ

本実装の構成

- 点群読み込み：np.loadtxtを用いてNx6（x, y, z, R, G, B）の形式で読み込む
- 最近傍探索：各点に対してユークリッド距離で最小の点を探索
- 剛体変換：点群の重心を原点に平行移動、対応点から共分散行列Hを構成、SVDで回転行列Rを求め、重心の差から並進tを求める
- RMSEの推移を記録し、収束を検出

使用ライブラリ

Numpy	Matplotlib	osライブラリはファイルパス操作に利用
点群処理・行列計算・SVDなど	RMSEグラフ・点群の3D描画	アルゴリズム処理には関与していません

可視化機能（補助スクリプト）

Visualize_point_cloud.pyにより、transformed_full.txtの点群を3Dプロットし、transformed_full.pngとして保存

結果1 ダウンサンプリングとフルデータのRMSE値

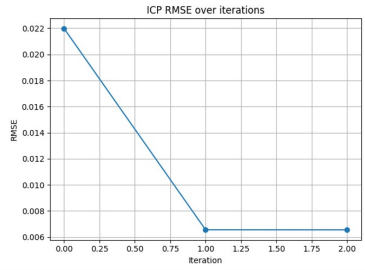
ダウンサンプリング RMSE	Iter 1 0.021983	Iter 2 0.006558	Iter 3 0.006554	Converged.

閾値=10^-4

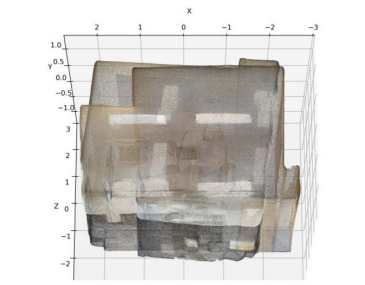
フルデータ RMSE	RMSE 0.268102	ダウンサンプリング のRとtを使用	RMSE 0.268091

- 収束の様子は、処理効率の都合上、ダウンサンプリング点群を用いて可視化した。
- ダウンサンプリング時の変換行列を元の点群に適用し、最終RMSEを計算したところ、精度の改善幅はごくわずかであった。
- 元点群での各イテレーション毎のRMSE評価は、処理に数時間要するため現実的ではなく、省略した。

精度の収束の様子グラフ



実行結果の点群画像



結果２　実際に精度が改善した手法

元の点群データで剛体変換を計算する方法より
ダウンサンプリングで得られた変換行列RとtをRを元の点群データの評価に用いることで、わずかに精度が改善された。

しかし、ダウンサンプリングした点群はほんの一部なため全体に適しているとは限らない。
今回は偶然精度が向上した可能性がある（逆に誤差が広がる可能性もある）

評価の信頼性を保つには、元の点群データ上で剛体変換の計算をすべき。
ダウンサンプリングの結果は初期値には有効だが、そのまま評価に使うのは適切ではない。

こんな状況では実用的に有効な場合もある？

初期位置を大きく外している時は粗い整合として使う	計算コストを抑えたいとき（高速近似として使う）

フルデータ精度の考察と技術的課題

フルデータ精度の考察

原因の推定

- ダウンサンプリングと比べて点数が増加することで局所的な誤差やノイズが増加。
- 形状の差異が顕著になり本来対応すべき点とは異なる点を選んでしまっている可能性がある。
- ICPアルゴリズムは対応点に基づいて剛体変換を行うため、誤対応が全体の整合性に大きな影響を与え精度低下に繋がった可能性がある。

改善の試み

- ダウンサンプリングによるレジストレーション結果を初期位置として用いることで、フルデータでも良好な整合性を確保し、誤対応を抑制できる可能性がある。
- 全ての対応点を等価に扱わず、距離の小さい対応点に高い重みを与えることで、外れ値の影響を相対的に小さくする。

結論

- これらの手法を段階的に導入することで、精度改善の可能性があると考えられる。
- 処理制約下で一部の工夫を実施したが、現時点ではフルデータでのRMSE値を0.1以下に抑えることは出来なかった。

技術的課題

コードを変えずbuildするためにmyproject直下にDockerfile移動させました。

↓

docker buildコマンド実行時poetry exportコマンドが存在しないというエラーが発生。

↓

poetry exportコマンドはPoetry v1.2以降で導入されたという記事から
Poetryのバージョンが古い可能性がある。

↓

解決策として、バージョン1.2以上を入れるようDockerfileを変更。

↓

その後、同様にdocker buildコマンドを実行、poetry exportは正常に動作したが、
pyproject.tomlに“name”フィールドが欠けているというエラーが発生。

↓

pyproject.tomlの変更や解決策を模索したが期限内での解決には至らず、
Dockerコンテナ上での実行は出来なかった。