特徴点マッチングの改良と実験の準備

M2 田川幸汰

1 前回までの結果と概要

以下に前回までの特徴点マッチングの結果（図 1）と実行時間（表 1）、自己位置推定結果（図 2）を示す。

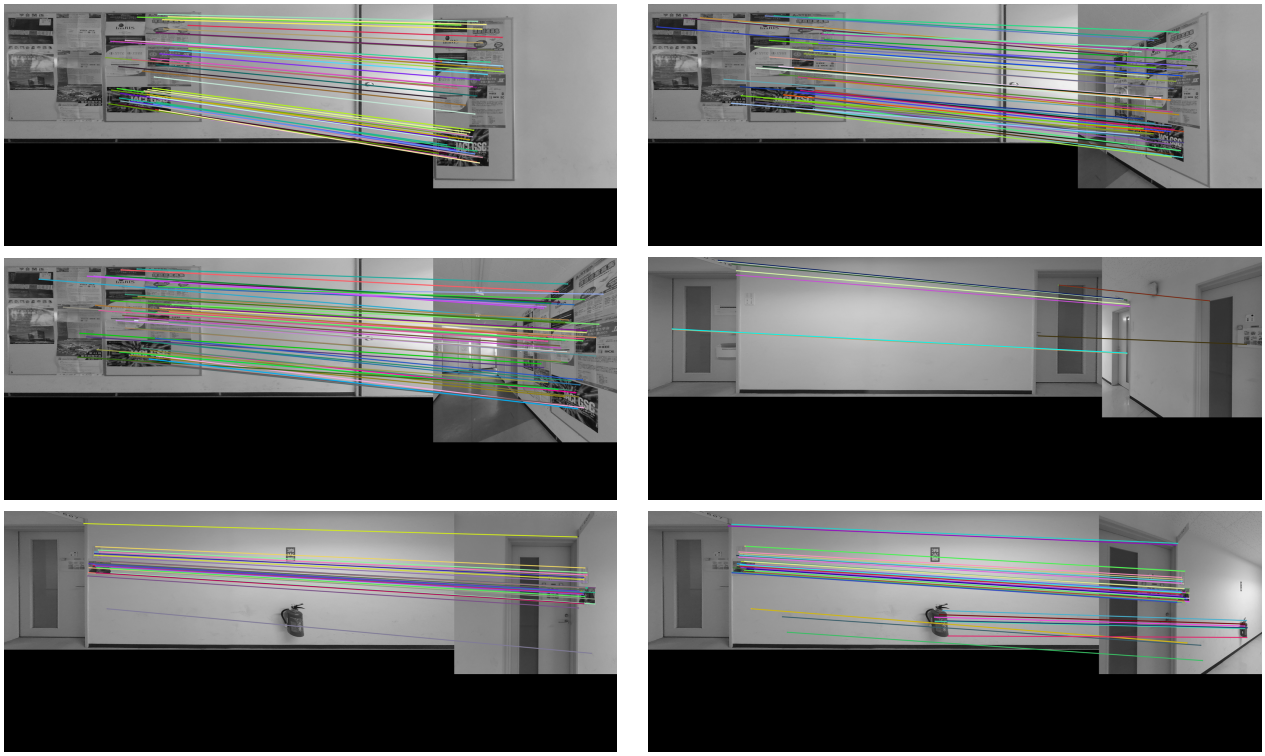

図 1: SuperPoint, SuperGlue による特徴点マッチング結果

表 1: 実行時間

	特徴点の事前検出 (秒)	特徴点検出 (秒)	特徴点マッチング (秒)	自己位置推定 (秒)
SuperPoint,SuperGlue	389.025	0.134	5.516	0.141

機械学習ベースの特徴点マッチング手法である SuperPoint [1] および SuperGlue [2] に対して、現在位置に基づく特徴点マッチングなどの修正を行うことで、マッチング精度および実行時間を改善した。さらに、自己位置推定の結果として一部のカメラ位置が壁の裏側に推定される問題が発生していたが、これについても幾何的な制約を導入することで、カメラ位置が正しく壁の表側に推定されるように改善した。

2 現在位置に基づく特徴点マッチング

表 1 より、処理全体における実行時間のボトルネックは特徴点マッチング処理であることが明らかとなった。従来は、入力された画像に対して毎回すべてのテクスチャ画像と特徴点マッチングを実施していて、処理時間が大幅に増加していた。この問題を解消するために、以下の 2 点の改善を行った。

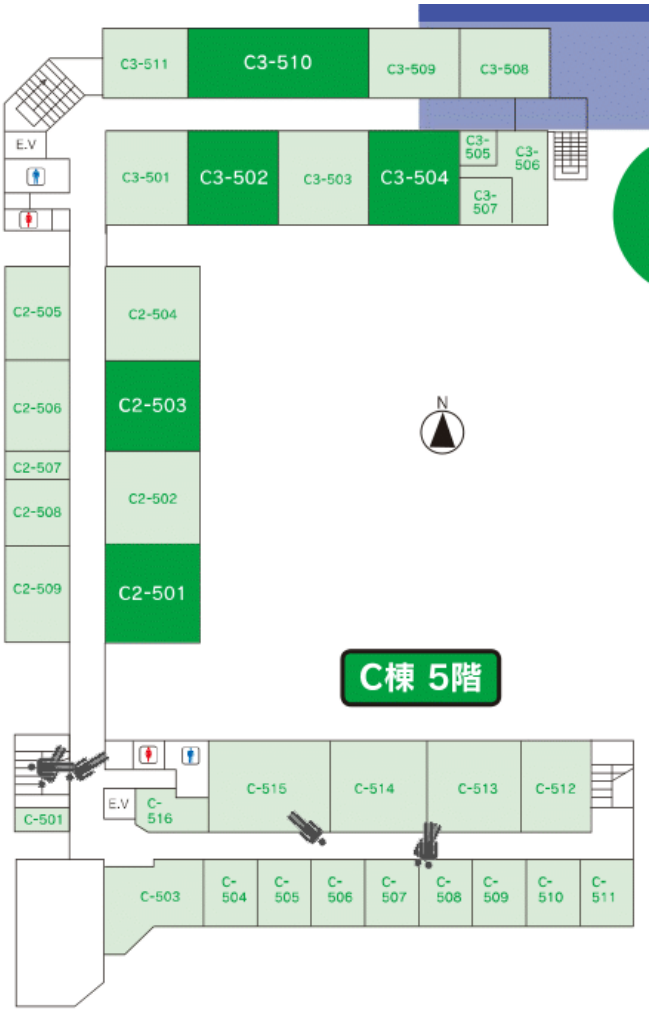

図 2: 自己位置推定結果

1. **初回のみ全テクスチャとのマッチングを実施**
   最初の 1 枚の入力画像に対しては、従来通り全テクスチャ画像と特徴点マッチングを実施

2. **2 回目以降は近傍テクスチャに限定**
   2 回目以降の入力画像については、1 つ前の画像から得られた自己位置推定結果（現在位置）を用いて、位置的に近いテクスチャ画像を 5 件選定し、それらとのみ特徴点マッチングを実施

また、この処理変更に伴い、各テクスチャ画像に対応する重心位置をあらかじめファイルに記述・保存する仕組みを追加した。

2.1 特徴点マッチングの修正

精度と実行時間の改良のための主な修正項目を以下に示す。

1. **特徴点検出およびマッチングのパラメータ調整**
   SuperPoint における最大検出数（max_keypoints）を 1500 から 500 に削減し、処理時間を短縮加えて、SuperGlue における Sinkhorn 反復回数（sinkhorn_iterations）を 10 から 15 に増やし、マッチングの精度を確保また、マッチングのしきい値（match_threshold）を 0.4 から 0.3 に変更し、より信頼性の高いマッチのみを採用

2. **距離スコアに基づくマッチ候補の絞り込み**
   SuperGlue によって得られたマッチのうち、スコアが高い上位 100 件のみを選択し、RANSAC による外れ値除去処理に入力するよう変更 RANSAC 処理における計算量を削減

3. **不要処理の削減と前処理の導入**
   マッチング結果の画像描画は、最良マッチが更新された場合に限って実施するよう変更また、特徴点の正規化を事前に行うことで、マッチングモデルへの入力サイズを削減し、GPU メモリの使用効率と処理速度を向上

3 幾何的な制約による自己位置推定結果の修正

一部の入力画像において、カメラ位置が壁の裏側に推定される問題が発生していた。この原因を調査するため、12 個の初期値から求めた自己位置推定結果のうち、目的関数のスコアが高い上位 3 件の解を分析した（図 3）。その結果、上位 2 件はいずれもスコアが同一であり、カメラ位置は壁の裏側に投影され、さらにカメラ姿勢が天地反転していることが確認された。

```
========== self-localization for input\half\input_-90.png ==========
input\half\input_-90.pngの特徴点検出時間：0.066秒
input\half\input_-90.pngの特徴点マッチング時間：3.981秒
最良マッチ数: 660
=== ナンバーワンの推定結果 ===
誤差 J_best: 4.622894410195802e-06
R_best:
 [[-0.02094977 -0.9997724   0.00403082]
 [ 0.0016813   0.00399646  0.9999906 ]
 [-0.99977912  0.02095636  0.0015972 ]]
C_best: [-1.02943697  6.98679868  1.29191141]

=== ナンバー2の推定結果 ===
誤差 J_2nd: 4.622894410195802e-06
R_2nd:
 [[-0.02094977  0.9997724  -0.00403082]
 [ 0.0016813  -0.00399646 -0.9999906 ]
 [-0.99977912 -0.02095636 -0.0015972 ]]
C_2nd: [1.02943697 6.98679868 1.29191141]
```

図 3: カメラ位置

カメラ位置が壁の裏側に誤って推定される問題に対処するため、カメラ座標系の Y 軸が常に下向きであるという幾何学的制約を導入した。具体的には、カメラ姿勢行列 R の要素 $R[1,2]$ が負（$R[1,2] < 0$）であることを条件とし、これはカメラの Y 軸が世界座標系の Z 軸負方向と一致していることを意味する。この判定処理を加えることで、壁の裏側と判定された解を候補から除外することが可能となった。

3.1 特徴点検出およびマッチング結果

特徴点検出およびマッチングは SuperPoint、SuperGlue の公式実装 ([3]) を参考にした。実行環境と、SuperPoint 及び SuperGlue のパラメータを表 2 に示す。

表 2: 実行環境及びハイパーパラメータ

GPU	NVIDIA GeForce RTX 4060 Ti, NVIDIA driver version=560.94, CUDA version=12.6
SuperPoint	nms_radius=4, keypoint_threshold=0.005, max_keypoints=500
SuperGlue	weights='outdoor', sinkhorn_iterations=15, match_threshold=0.3

実行結果を図 4 に示す。

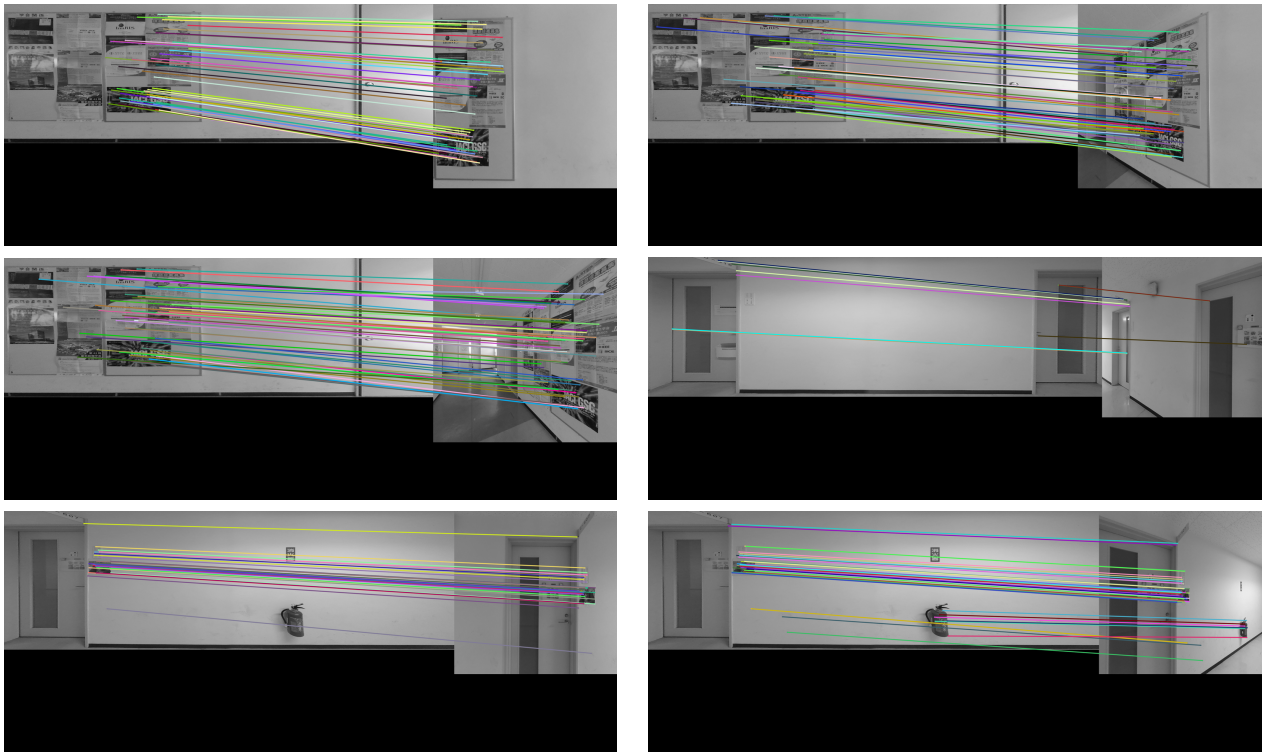

図 4: SuperPoint, SuperGlue によるマッチング結果

SuperPoint および SuperGlue を用いた場合には、従来の手法とは異なり、テクスチャが豊富な画像において高精度なマッチングが行えるだけでなく、テクスチャが乏しい画像においても正しいデータベース画像を選択し、特徴点マッチングを安定して行うことができた。これは、SuperPoint が画像全体からより意味的な特徴点を抽出し、SuperGlue が空間的な関係性を考慮したマッチングを行うためであり、従来の手法（SIFT や AKAZE）に比べて、ロバスト性と適応性の両面で優れていることを示している。ただし、類似したテクスチャを有する別の画像と誤ってマッチングしてしまう例が一部で見られた。これは、画像間の構造的類似性や局所的なパターンが極めて近い場合に、SuperPoint および SuperGlue の判断が困難になるためであると考えられる。

また、特徴点マッチングの結果を用いた自己位置推定の結果を図**??**に示す。回転行列の初期値として 12 通りの視点を与え、それぞれについて目的関数を評価し、最も誤差（目的関数値）が小さい解を大域最適解として採用している。

```
========== self-localization for input\half\input_-90.png ==========
input\half\input_-90.pngの特徴点検出時間：0.066秒
input\half\input_-90.pngの特徴点マッチング時間：3.981秒
最良マッチ数: 660
=== ナンバーワンの推定結果 ===
誤差 J_best: 4.622894410195802e-06
R_best:
 [[-0.02094977 -0.9997724   0.00403082]
 [ 0.0016813   0.00399646  0.9999906 ]
 [-0.99977912  0.02095636  0.0015972 ]]
C_best: [-1.02943697  6.98679868  1.29191141]

=== ナンバー2の推定結果 ===
誤差 J_2nd: 4.622894410195802e-06
R_2nd:
 [[-0.02094977  0.9997724  -0.00403082]
 [ 0.0016813  -0.00399646 -0.9999906 ]
 [-0.99977912 -0.02095636 -0.0015972 ]]
C_2nd: [1.02943697 6.98679868 1.29191141]
```

図 5: カメラ位置

そのため、カメラ座標系の Y 軸が常に下向きであるという前提に基づき、世界座標系における Z 軸の負方向に一致するかどうか（$R[1,2] < 0$）を用いて、「壁の手前か裏側か」を判定する処理を追加した。この判定により、壁の裏側に位置する解を大域最適解の候補から除外することができた（図 6）。

図 6: 自己位置推定結果の改良版

4 実行時間の比較

自己位置推定機能を応用したリアルタイムアプリケーションを開発する際には、特徴点の抽出およびマッチング処理を高速に行う必要がある。そのため、各手法の実行時間を比較し、リアルタイム性の観点からの評価を行った。比較結果を表 3 に示す。

表 3: 実行時間 (入力画像 1 の結果)

	特徴点の事前検出 (秒)	特徴点検出 (秒)	特徴点マッチング (秒)	自己位置推定 (秒)
SIFT	18.312	0.248	3.151	0.166
AKAZE	18.213	0.182	1.939	0.123
SuperPoint,SuperGlue	389.025	0.134	5.516	0.141

結果より、特徴点の事前検出（テクスチャ画像に対する特徴点検出）に関しては、SuperPoint および SuperGlue を用いた手法において実行時間が他の手法と比べて極端に長くなった。ただし、この処理は各テクスチャ画像に対して一度だけ実行すれば十分であり、あらかじめ検出した特徴点をファイルとして保存しておくことで、再利用が可能であるため、実行時間の増加は大きな問題とはならない。

また、3 手法すべてにおいて特徴点マッチングに比較的長い時間を要しているが、これは入力画像とすべてのデータベース画像（テクスチャ画像）との間で総当たりのマッチングを行っているためである。現実の利用においては、自己位置推定結果をもとに、周辺に存在する一部のテクスチャ画像とのみマッチングを行うことが可能であり、このような工夫により実行時間は大幅に短縮できると考えられる。

5 今後の計画

今後の研究計画を以下に示す。

1. 7 月：C 棟 5 階全範囲の 3 次元モデル作成（テクスチャ付加）
2. 8 月：線特徴を用いた自己位置推定結果の補助
3. 8 月以降：自己位置推定機能を応用した実用的なアプリケーションのデモ開発
    - 生成された 3 次元モデルと自己位置推定を組み合わせ、目的地までのルートを提示するナビゲーションシステムを構築

参考文献

[1] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” *arXiv preprint arXiv:1712.07629*, 2018.

[2] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Geiger, “Superglue: Learning feature matching with graph neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4938–4947, 2020.

[3] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superglue pretrained network: Superglue: Learning feature matching with graph neural networks.” https://github.com/magicleap/SuperGluePretrainedNetwork, 2020. Accessed: 2025-07-09.