

# ワイヤーフレームと全方位画像による 簡易モデルを用いた屋内環境での自己位置推定

2025

修士（工学）

情報・知能工学専攻

田川 幸汰

223337

豊橋技術科学大学

# 目 次

<b>第1章 はじめに</b>	<b>2</b>
1.1 研究背景	2
1.2 研究目的	2
1.3 関連研究	3
1.3.1 Visual SLAM による自己位置推定	3
1.3.2 画像ベースの自己位置推定手法	4
1.3.3 簡易 3 次元モデルを用いた位置推定	4
1.4 本論文の構成	5
<b>第2章 簡易 3 次元モデルの生成</b>	<b>6</b>
2.1 簡易 3 次元モデル生成の方針	6
2.2 座標系の定義	6
2.2.1 カメラ座標系	6
2.2.2 画像座標系	7
2.2.3 カメラ内部パラメータの設定	7
2.3 ワイヤーフレーム生成の方針	8
2.4 2 次元マップと 3 次元空間の対応付け	8
2.4.1 アフィン変換による写像	9
2.4.2 非線形な歪みへの対応	9
2.4.3 頂点列の回転方向の判定と統一	9
2.5 カメラ位置に基づく床境界点の最適化	10
2.6 床面および壁面の幾何構造	11
2.6.1 床面のメッシュ化	11
2.6.2 壁面のメッシュ化	11
2.6.3 UV 座標の定義	12
2.7 全方位画像を用いたテクスチャ取得の方針	13
2.8 透視投影画像変換	13
2.9 透視投影画像を用いた全方位カメラの位置姿勢推定	14
2.10 メッシュの座標系変換と投影	15
2.10.1 テクスチャ候補の評価	16
2.11 テクスチャ画像の形状変換	16
2.12 テクスチャの視覚的品質の改善	17
2.12.1 カメラとメッシュの相対位置関係の導出	17
2.12.2 テクスチャのブレンド処理	18

<b>第3章 特徴点マッチングに基づく自己位置推定</b>	<b>19</b>
3.1 特徴点検出およびマッチングの概要 . . . . .	19
3.1.1 共通の前処理 . . . . .	19
3.2 従来手法による特徴点マッチング . . . . .	20
3.2.1 特徴点検出 . . . . .	20
3.2.2 特徴点マッチング . . . . .	20
3.2.3 マッチングの精度向上 . . . . .	20
3.3 学習ベース手法による特徴点マッチング . . . . .	20
3.3.1 特徴点検出 . . . . .	20
3.3.2 特徴点マッチング . . . . .	21
3.3.3 マッチングの精度向上 . . . . .	21
3.4 テクスチャ画像座標から世界座標への変換 . . . . .	22
3.4.1 2次元座標から3次元座標への変換 . . . . .	22
3.5 自己位置推定 . . . . .	23
<b>第4章 自己位置推定結果を用いた屋内ナビゲーション</b>	<b>25</b>
4.1 屋内ナビゲーションの方針 . . . . .	25
4.1.1 簡易モデルによる自己位置推定の課題 . . . . .	25
4.1.2 提案手法に基づく屋内ナビゲーションの基本方針 . . . . .	25
4.1.3 Visual-Inertial Odometry (VIO) による自己位置推定 . . . . .	25
4.1.4 ARKit の座標系の定義 . . . . .	26
4.2 自己位置推定手法の比較検討方針 . . . . .	26
4.3 屋内ナビゲーションシステムの構成 . . . . .	27
4.3.1 目的地設定 . . . . .	28
4.3.2 カメラ画像取得および端末とPC間の通信 . . . . .	28
4.3.3 自己位置推定結果の受信および座標系の更新 . . . . .	29
4.3.4 世界座標系とAR座標系の相互変換 . . . . .	29
4.3.5 目的地および進行方向オブジェクトの描画 . . . . .	30
<b>第5章 実験</b>	<b>31</b>
5.1 実験準備 . . . . .	31
5.2 簡易モデル生成 . . . . .	32
5.2.1 データセットおよびパラメータ設定 . . . . .	32
5.2.2 ワイヤーフレーム生成結果 . . . . .	33
5.2.3 テクスチャ割り当て結果 . . . . .	34
5.3 特徴点マッチング手法の比較実験 . . . . .	34
5.3.1 実験条件 . . . . .	34
5.3.2 実験結果 . . . . .	36
5.4 特徴点マッチングに基づく自己位置推定結果の評価 . . . . .	38
5.5 屋内ナビゲーションにおける自己位置推定結果の比較 . . . . .	39
5.5.1 実験条件 . . . . .	39
5.5.2 実験結果 . . . . .	39

<b>第6章 まとめ</b>	<b>41</b>
6.1 本研究の成果	41
6.2 今後の展望	41

# 第1章 はじめに

## 1.1 研究背景

近年、現実空間とデジタル空間を連携させるデジタルツインの概念が注目されている。デジタルツインの社会実装に関する既存の調査によると、国内企業の約7割がデジタルツインを導入済み、または導入を検討していると報告されている[1]。特に、屋内空間をデジタル上に再現して利用者の位置を推定することで、それに応じた案内や情報提示を行う技術への期待が高まっている。

しかしながら、同時に市場レポートでは約41%の企業が導入にあたって予算制約を主な阻害要因として挙げており[2]、デジタルツイン導入への関心は高い一方で、3次元計測機器や専用センサーへの投資、システム構築や維持にかかるコストが導入における大きな障壁となっていることが示されている。このような社会的背景から、新たな設備投資を必要とせず、カメラで取得した画像のみから屋内環境の3次元モデルを生成する手法が望まれてきた。

しかし、屋内環境は壁や床、天井といった単調な構造が多く、特徴に乏しい場合が多い。そのため、画像情報に基づいて3次元構造を推定する際には、十分な特徴点が得られず、空間構造を正確に復元することが困難である。特に、Visual SLAMに代表されるような、自己位置推定を行いながら高精度なマップ生成を同時に行う手法では、特徴点の不足が自己位置推定の不安定化や、環境マップ品質の低下につながることが指摘されており[3, 7]、画像のみを用いて屋内環境の高精度な3次元モデルを安定して構築することは、依然として大きな課題となっている。

一方で、屋内環境における道案内などのナビゲーション用途では、必ずしも幾何学的に高精度な3次元モデルや、精密な自己位置推定が常に必要であるとは限らない。経路案内や現在位置の把握といった目的においては、空間の大まかな構造を表現できる3次元モデルが得られれば十分である場合も多いと考えられる。このような背景を踏まえ、本研究では屋内ナビゲーション用途に必要な情報に着目し、高精度な3次元形状復元に依存しない、特徴の乏しい環境にも適用可能な空間表現と、画像に基づく自己位置推定手法について検討する。

## 1.2 研究目的

本研究の目的は、特徴に乏しくVisual SLAMの適用が困難な屋内環境において、ナビゲーション用途に求められる実用的な精度と安定性を有する自己位置推定手法を確立することである。特に、高精度な自己位置推定を必ずしも前提としない屋内ナビゲーションを対象とし、既存の建物情報を活用することで、設備投資を抑えつつ安定した動作の実現を目指す。

具体的には、2次元マップから生成したワイヤーフレームモデルに全方位画像から取得したテクスチャを付与した簡易的な3次元モデルを作成し、入力画像とモデル上のテクスチャとの特徴点マッチングに基づいて自己位置を推定する手法を提案する。

本手法のアプローチには、大きく二つの特徴がある。

第一に、導入障壁が低い点である。提案手法は、建物に既存の2次元マップと全方位画像という、比較的容易に取得可能な情報のみを用いて環境モデルを構築する。そのため、高価な3次元

計測機器による事前の精密スキャンや、ビーコン・マーカーなどの設備を環境側に設置する必要がなく、デジタルツイン導入の課題となっているコストと手間を大幅に削減できる。

第二に、特徴の乏しい環境における安定性と、ナビゲーションの継続性を重視した設計思想である。公共施設やオフィスビルでの経路案内においては、ミリメートル単位の厳密な自己位置推定精度よりも、自己位置を見失わずに追跡し続ける「安定性」が極めて重要となる。一般的なVisual SLAMは、特徴が豊かな環境では精密な自己位置推定が可能だが、単調な壁面など特徴が乏しい環境では、特徴点不足によりトラッキングが破綻しやすい。本研究では、空間の大まかな構造を表現した簡易モデルを参照することでこの問題を回避する。これにより、厳密な推定精度よりも、特徴の乏しい環境下であっても安定して自己位置推定を行うことで、ナビゲーション動作を継続させることを目指す。

以上より、本研究は高精度な三次元形状の復元や新たな設備投資に依存することなく、画像情報のみに依存する従来のVisual SLAMでは自己位置の維持が困難な特徴の乏しい環境下においても、安定して動作可能な屋内ナビゲーション向け自己位置推定の枠組みを提示することを目的とする。

## 1.3 関連研究

### 1.3.1 Visual SLAM による自己位置推定

Visual SLAMは、カメラ画像から特徴点を抽出および追跡することで、自己位置推定と環境マップ生成を同時に行う代表的な手法である。ORB-SLAM[3] や ORB-SLAM2[4] に代表される手法では、ORB特徴量を用いた高精度なトラッキングおよびループ検出により、高精度な自己位置推定と三次元マップ生成が可能である。また、近年では深層学習を導入した手法[5]も提案されており、特徴点抽出やマッチングの頑健性向上が試みられている。

一方で、Visual SLAMは十分な特徴点が安定して得られることを前提としており、壁面や床面が単調な屋内環境では、特徴点不足によりトラッキングが不安定になることが指摘されている[3, 7]。例えば、弱いテクスチャ領域を含む環境を対象とした実験では、追跡可能な特徴点数が大きく減少し、その結果として自己位置推定軌道が真値から逸脱する様子が報告されている（図1.1）[6]。

したがって、特徴点の乏しい屋内環境において、画像情報のみに依存するVisual SLAM 単独で既知のマップ座標系上における絶対的な自己位置を高精度かつ安定して推定することは、依然として困難な課題である。

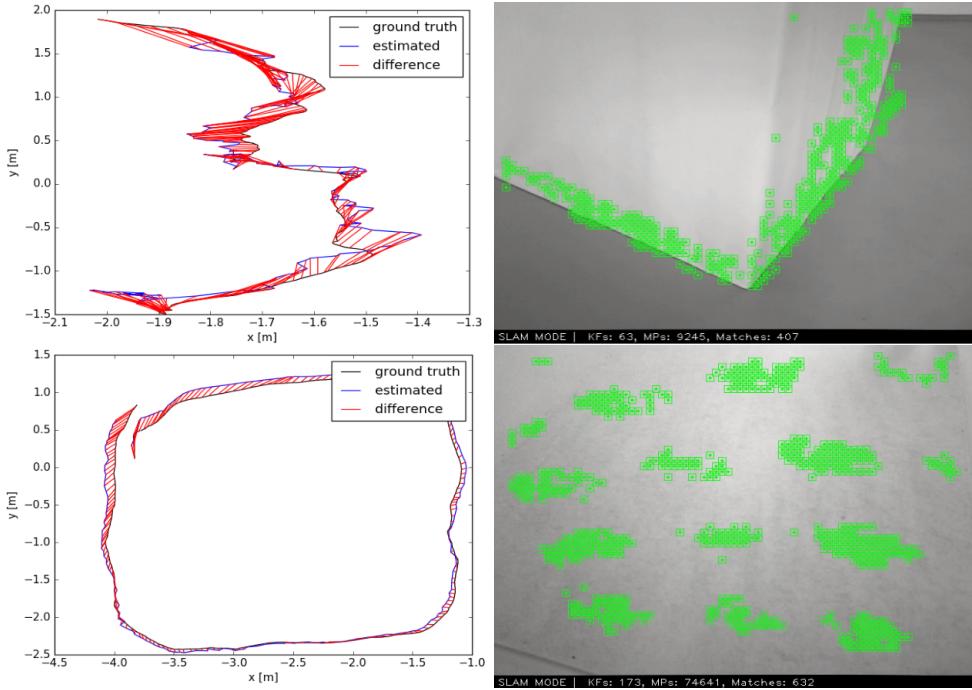


図 1.1: 弱いテクスチャ環境における自己位置推定結果および追跡特徴点の可視化. 出典: Y. Liu et al.(2022)[6], Fig. 4

### 1.3.2 画像ベースの自己位置推定手法

環境マップ生成を行わず、既存の環境モデルとカメラ画像との対応付けにより自己位置推定を行う手法も多く提案されている。代表的なアプローチとしては、SfM (Structure from Motion) により事前に構築した 3 次元点群と画像特徴量とのマッチングに基づく手法 [8] や、大規模画像データベースを用いた画像検索に基づく位置推定手法 [9] が挙げられる。

これらの手法は、事前に構築された環境モデルを利用するため、未知環境を探索する SLAM に比べて累積誤差の影響を受けにくく、安定した自己位置推定が可能である。しかし、高密度かつ高精度な 3 次元モデルやデータベースの事前構築が前提となる。そのため、テクスチャが単調な環境では特徴点マッチングに基づくモデル生成自体が困難である点や、点群構築のために大規模な撮影や計測が必要となり、導入・運用面での負担が大きいという課題がある。

### 1.3.3 簡易 3 次元モデルを用いた位置推定

高精度な 3 次元形状復元に依存せず、簡易的な三次元環境モデルや幾何学的制約を用いて自己位置推定を行う試みも報告されている。

例えば、Sattler ら [15] は、事前に撮影された画像群に対して SfM (Structure from Motion) を適用し、3 次元空間内の特徴点群のみを抽出・保存することで、軽量な環境マップを構築している。彼らは、このスペースな 3 次元点と入力画像の特徴量との直接的な対応付けを行うことで、高密度なモデルを持たずとも高速かつ高精度な位置推定が可能であることを示した。

また、都市環境などを対象とした Schindler ら [14] の研究では、建物のファサードが持つ幾何学的な繰り返しパターンや、空間内の主要な平面構造に着目している。この手法では、環境全体を厳密に復元するのではなく、位置特定に有効な特徴を選択的にデータベース化したり、建物を

鉛直平面として近似するなどの幾何学的制約を利用することで、効率的な自己位置推定を実現している。

これらの手法は、必ずしも高密度な3次元形状復元を行わずとも、特徴点の配置や簡易的な幾何情報のみで実用的な自己位置推定が可能であることを示している点で重要である。

本研究は、こうした「簡易モデルによる推定」というアプローチを踏襲しつつ、モデル生成のソースとして、既に存在する「2次元マップ」を利用する点に独自性がある。全方位画像を用いて効率的にテクスチャを付与した簡易3次元モデルを構築することで、現地での大規模な事前撮影やSfMによる点群生成プロセスを省略し、特徴点が乏しい屋内環境においても低コストで導入可能なナビゲーション手法の実現を目指すものである。

## 1.4 本論文の構成

本論文の構成を以下に示す。第2章では、本研究の基盤となる簡易3次元モデルの生成手法について説明する。第3章では、入力画像と簡易モデルのテクスチャ間の特徴点マッチング、およびその結果に基づく自己位置推定手法について説明する。第4章では、自己位置推定結果を用いた屋内ナビゲーション手法について説明する。第5章では、提案手法により実際に生成された簡易3次元モデルおよび自己位置推定結果を示し、推定精度の屋内ナビゲーションにおける有効性を検証するために行った実験について説明する。

# 第2章 簡易3次元モデルの生成

## 2.1 簡易3次元モデル生成の方針

本研究では、屋内ナビゲーションに必要な自己位置推定を効率的に行うため、現実空間を厳密に模倣するのではなく、計算コストとデータ量を最小限に抑えた簡易的な3次元モデルを生成する方針をとる。

一般に、屋内環境の3次元モデル化には、レーザースキャナによる高密度な点群計測（図2.1左）や、多数の画像を用いたフォトグラメトリによるメッシュ生成（図2.1中央）が用いられることが多い。これらの手法は、環境の形状を詳細に再現できる反面、データ容量が肥大化しやすく、モバイル端末など計算資源の制約が大きい環境において実行する場合、リアルタイムな描画や照合処理の実現は困難である。また、モデル構築のために専門的な機材や多大な計算時間を要する点も課題となる。

これに対し、本研究で提案する簡易モデル（図2.1右）は、2次元マップから抽出した少数の頂点と線分を幾何的な骨格とし、そこに全方位画像から取得したテクスチャ情報を付与することで構成される。

このように、必要最低限の幾何構造をワイヤーフレームで、視覚情報をテクスチャとして表現するアプローチをとることで、高密度な点群や精密なメッシュの生成に依存することなく、ナビゲーション用途として実用上十分な情報の確保と、システム運用における軽量性の両立を目指す。



図2.1: 異なる3次元モデル表現の比較。従来の点群(a)やメッシュ(b)と比較し、本研究(c)では構造を大幅に簡略化している。

## 2.2 座標系の定義

### 2.2.1 カメラ座標系

簡易モデル生成に用いる各座標軸の関係を図2.2に示す。

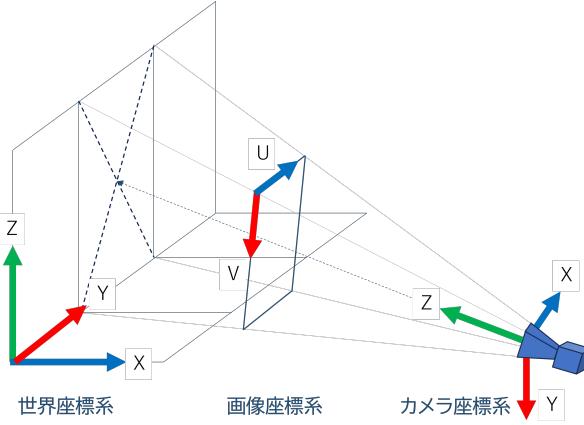


図 2.2: 世界座標系とカメラ座標系、画像座標系の関係

カメラ座標系は OpenCV における座標系に準拠し、カメラの焦点位置を原点とする。光軸方向を  $Z$  軸、水平右方向を  $X$  軸、鉛直下方向を  $Y$  軸と定める。一方、世界座標系はモデル床面を  $X-Y$  平面、鉛直上向きの法線方向を  $Z$  軸として定義する。世界座標系上の 3 次元点  $\mathbf{p}_w$  は、カメラの回転行列  $\mathbf{R}$  と並進ベクトル  $\mathbf{t}$  を用いて、次式によりカメラ座標系上の点  $\mathbf{p}_c$  に変換される。

$$\mathbf{p}_c = \mathbf{R}\mathbf{p}_w + \mathbf{t} \quad (2.1)$$

### 2.2.2 画像座標系

画像座標系は画像左上を原点とし、水平方向を  $U$  軸、垂直方向を  $V$  軸と定める。カメラ座標系上の 3 次元点  $\mathbf{p}_c = (x_c, y_c, z_c)^\top$  は、カメラ内部パラメータ行列  $\mathbf{K}$  を用いて、次式により画像座標系上の同次座標  $\mathbf{p}_s$  へ射影される。

$$\mathbf{p}_s = \mathbf{K}\mathbf{p}_c \quad (2.2)$$

ここで、 $\mathbf{p}_s = (u_s, v_s, w_s)^\top$  とすると、実際の透視投影画像上の画素座標  $(u, v)$  は、 $w_s$  による正規化を行うことで、次式で与えられる。

$$u = \frac{u_s}{w_s}, \quad v = \frac{v_s}{w_s} \quad (2.3)$$

### 2.2.3 カメラ内部パラメータの設定

本研究では理想的な透視投影モデルを仮定し、カメラ内部パラメータを幾何学的に設定する。内部パラメータ行列  $\mathbf{K}$  は次式で表される。

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (2.4)$$

ここで、 $f_x, f_y$  はピクセル単位の焦点距離を表す。出力する透視投影画像の幅を  $W_p$ 、高さを  $H_p$  とし、水平視野角を  $\Theta$ 、垂直視野角を  $\Phi$  と設定した場合、各焦点距離は次式で表される。

$$f_x = \frac{W_p}{2 \tan(\Theta/2)}, \quad f_y = \frac{H_p}{2 \tan(\Phi/2)} \quad (2.5)$$

また  $c_x, c_y$  は画像中心を表し、透視投影画像の中心座標  $(W_p/2, H_p/2)$  に設定する。

## 2.3 ワイヤーフレーム生成の方針

本節では、2次元マップ画像を基盤として、屋内環境の幾何構造を記述する3次元ワイヤーフレームモデルを構築する手法について述べる。本手法における生成プロセスは、以下の3つの主要なフェーズから構成される。

### 1. 2次元マップと3次元空間の対応付け:

2次元マップ座標と世界座標の幾何学的対応関係を定義し、モデル生成の基盤となる床領域を確定する。

### 2. カメラ位置に基づく床境界点の最適化:

定義された床領域に対し、カメラ位置を考慮して床境界点を追加し、形状再現に必要な頂点の密度を確保する。

### 3. 3次元幾何構造の構築:

最適化された境界点列に基づき、床面および壁面のメッシュ化処理を行い、最終的な3次元モデルを生成する。

なお、「床境界点」とは、屋内空間における床面と壁面の境界線を構成する一連の頂点列を指す。本研究では、ユーザがマップ上で大まかな対応点や領域の指示を与え、それ以降の詳細な頂点配置や構造化をシステムが自動で行う「半自動」なアプローチを採用する。これにより、複雑な屋内形状への柔軟な対応と、モデリング作業の効率化の両立を図る。

また、ワイヤーフレーム生成にあたっては、以下の情報が利用可能であることを前提とする。

- 屋内環境を表す2次元マップ:

建物のフロアマップや設計図などの画像データ。

- 2次元マップと3次元空間の対応点:

マップ上の画素座標と、実空間の3次元座標との対応関係を示す3点以上の点対。これは、マップと実空間の位置合わせのために用いられる。

- 屋内環境内でテクスチャを取得したカメラ位置:

全方位画像の撮影位置。これは後述する床境界点のサンプリングにおいて、形状の詳細度を決定するために用いられる。

次節より、これらの入力情報に基づく具体的なモデル生成アルゴリズムについて述べる。

## 2.4 2次元マップと3次元空間の対応付け

本研究では、2次元マップ上で定義された床境界の頂点座標を、実際の3次元空間内の床面上へ写像することで位置合わせを行う。床面は水平であると仮定し、実空間上の3次元座標は常に  $Z = 0$  に固定されるものとする。

この写像を行う手法として、本研究では「アフィン変換による自動推定」と、マップの歪みを考慮した「手動による直接対応付け」の2種類のアプローチを状況に応じて使い分ける。

#### 2.4.1 アフィン変換による写像

建築図面や正確なフロアマップが得られる場合、2次元マップと実環境との関係は、平行移動・回転・スケーリング・せん断を含むアフィン変換によって十分に表現可能である。

まず、2次元マップ上の点  $\mathbf{p}_i = (x_i, y_i)^\top$  と、それに対応する実空間（床面）上の点  $\mathbf{q}_i = (X_i, Y_i)^\top$  の対応ペアを  $N$  点 ( $N \geq 3$ ) 取得する。両者の関係は、アフィン変換行列  $\mathbf{A}$  を用いて以下のように表される。

$$\begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} = \mathbf{A} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \quad (2.6)$$

ここで、すべての対応点に対して変換誤差が最小となるような行列  $\mathbf{A}$  を推定する。具体的には、以下の再投影誤差の二乗和  $E$  を最小化する最小二乗法により  $\mathbf{A}$  を決定する。

$$\hat{\mathbf{A}} = \underset{\mathbf{A}}{\operatorname{argmin}} \sum_{i=1}^N \|\mathbf{q}_i - \mathbf{A}\tilde{\mathbf{p}}_i\|^2 \quad (2.7)$$

ただし、 $\tilde{\mathbf{p}}_i = (x_i, y_i, 1)^\top$  は同次座標表現である。このようにして求めた変換行列  $\hat{\mathbf{A}}$  を用いることで、2次元マップ上で定義されたすべての頂点を、一括して3次元座標系へ変換する。

#### 2.4.2 非線形な歪みへの対応

一方、簡易的な2次元マップを利用する場合、マップ自体が非線形な歪みを含んでいることがあり、単一のアフィン変換行列では十分な精度で写像できない場合がある。このような場合には、上述の自動変換を用いず、床境界の頂点に対して実測した3次元座標を手動で直接割り当てる手法を採用する。これにより、局所的な歪みが全体の位置合わせに悪影響を与えることを防ぎ、整合性の取れたモデル生成を可能とする。

#### 2.4.3 頂点列の回転方向の判定と統一

床境界の多角形の幾何学的整合性を保つため、頂点順序を時計回りに統一する処理を行う。

一般に、平面上の  $N$  角形の頂点列  $\mathbf{p}_i = (x_i, y_i)^\top$  ( $i = 1, \dots, N$ ) が与えられたとき、その経路が囲む符号付き面積  $S$  は次式で定義される。

$$S = \frac{1}{2} \sum_{i=1}^N (x_i y_{i+1} - x_{i+1} y_i) \quad (2.8)$$

ここで、 $\mathbf{p}_{N+1} = \mathbf{p}_1$  とする。本研究の設定するマップ画像の座標系は  $Y$  軸下向きであるため、頂点列が時計回りの場合、この符号付き面積  $S$  は正の値をとる。すべての頂点を時計回りに統一するため、算出された  $S$  が負の値となった場合、頂点列の順序を逆順  $(\mathbf{p}_N, \mathbf{p}_{N-1}, \dots, \mathbf{p}_1)$  に並べ替える補正を行う。

## 2.5 カメラ位置に基づく床境界点の最適化

高品質なテクスチャを生成するためには、テクスチャの歪みを抑えるための詳細な頂点が必要となる。しかし、これらすべてを手作業で指定することは多大な労力を要するため、現実的ではない。そこで本研究では、床境界上においてカメラ配置を考慮しながら適切な間隔で境界点を自動生成する手法を提案する。図 2.3 に、カメラ位置に基づいて床境界点を追加する概念図を示す。

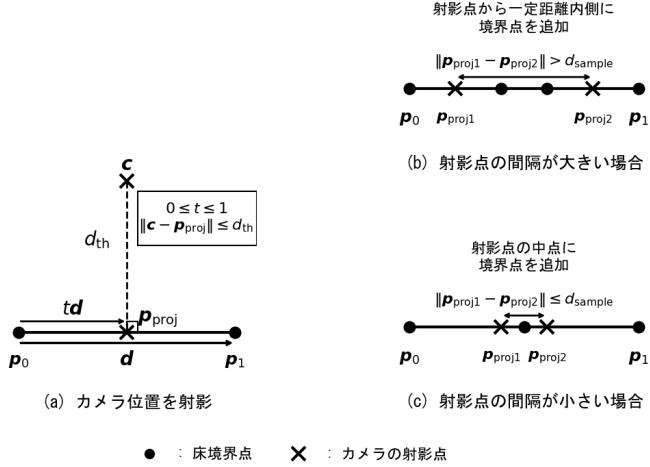


図 2.3: 床境界点の追加方法。カメラからの射影点周辺に重点的に頂点を配置する。

床境界は閉じた多角形として定義され、隣接する 2 点  $p_0$  と  $p_1$  が 1 つの境界エッジを構成する。本手法では、まず各エッジの始点  $p_0$  を固定の境界点として採用した上で、カメラ位置に応じた点を動的に追加していく。

具体的には、各カメラ位置  $c$  から境界エッジへの正射影を考える。エッジの方向ベクトル  $d$  および、始点からの距離比を表す射影係数  $t$  は次式で与えられる。

$$d = p_1 - p_0 \quad (2.9)$$

$$t = \frac{(c - p_0)^\top d}{d^\top d} \quad (2.10)$$

これより、境界直線上への射影点  $p_{\text{proj}}$  は次式で表される。

$$p_{\text{proj}} = p_0 + t d \quad (2.11)$$

ここで、射影係数  $t$  が  $0 \leq t \leq 1$  (エッジの範囲内) を満たし、かつカメラとの距離が閾値以下である場合、すなわち

$$\|c - p_{\text{proj}}\| \leq d_{\text{th}} \quad (2.12)$$

を満たす場合に、その射影点を有効な候補とする。

次に、得られた有効な射影点に基づき、最終的なサンプリング位置（追加の境界点）を決定する。テクスチャマッピング時の歪みを最小限に抑えるには、カメラ正面に近い領域、すなわち射影点付近の幾何形状を重点的に保持することが望ましい。そこで、同一エッジ上に複数の有効な

射影点  $\mathbf{p}_{\text{proj}1}, \mathbf{p}_{\text{proj}2}$  が存在する場合には、その点間距離とユーザが定義するサンプリング間隔  $d_{\text{sample}}$  との大小関係に応じて処理を分岐する。

具体的には、射影点間の距離が十分に大きい場合 ( $\|\mathbf{p}_{\text{proj}1} - \mathbf{p}_{\text{proj}2}\| > d_{\text{sample}}$ ) には、2つの射影点に挟まれた区間ににおいて、それぞれの射影点から相手側へ一定距離だけ進んだ位置に境界点を追加する。一方、射影点間の距離が小さい場合 ( $\|\mathbf{p}_{\text{proj}1} - \mathbf{p}_{\text{proj}2}\| \leq d_{\text{sample}}$ ) には、過度な細分化を防ぐため、両者の中点を1つの境界点として追加する。これにより、カメラ視点に対して最適な密度で床境界点を配置することが可能となる。

最後に、生成された床境界点列全体について隣接点間の間隔を確認し、間隔が大きすぎる部分には中点を追加し、間隔が小さすぎる部分では点を統合する。この後処理により、床境界全体が極端に偏ることなく、おおむね一定のサンプリング間隔で分布するように調整し、メッシュ生成の安定性を確保する。

## 2.6 床面および壁面の幾何構造

床面は、前節までに得られた床境界頂点列  $\{\mathbf{P}_i\}$  によって囲まれた領域として定義される。ここで、床境界は辺同士が互いに交差しない単純多角形を形成しているものとする。また、各床境界頂点は床面上に存在し、その  $z$  座標は  $z = 0$  に固定されている。

床面に対応する天井境界点列  $\{\mathbf{P}_i^{\text{ceil}}\}$  は、床境界頂点列の平面形状を保ったまま、高さ方向に一定量  $H$  だけ平行移動することで生成する。すなわち、床境界頂点  $\mathbf{P}_i = (x_i, y_i, 0)$  に対し、対応する天井境界頂点は  $\mathbf{P}_i^{\text{ceil}} = (x_i, y_i, H)$  として与えられる。

### 2.6.1 床面のメッシュ化

実際の屋内環境における床形状は、L字路や障害物による凹凸などが存在するため、必ずしも凸多角形とはならず、凹部を含む一般多角形となる。このような形状に対して、均質で安定したメッシュを生成するため、本研究では「制約付き Delaunay 三角形分割（Constrained Delaunay Triangulation）」を用いて床面を分割する。この手法は、Shewchuk によって提案された高品質な2次元メッシュ生成手法として知られている [16]。

具体的には、床境界の頂点集合を入力とし、隣接する床境界頂点同士を結ぶラインを拘束辺 (Segment) として設定する。これにより、多角形の外部や、本来存在しない穴の部分にメッシュが生成されるのを防ぎ、床形状のトポロジーを正確に反映した分割が可能となる。また、三角形の最大面積を制約条件として与えることで、過度に細長い三角形 (Sliver) の生成を抑制し、テクスチャマッピングに適した形状品質を確保する。

最後に、全ての三角形メッシュの各面に対して、2.4.3 節で述べた回転方向の判定方法に基づき、頂点順序が時計回りとなるよう統一する。これにより、全メッシュの法線ベクトルが垂直上向き (+z 方向) に揃い、面の向きを統一する。

### 2.6.2 壁面のメッシュ化

壁面の幾何構造は、床境界と天井境界の対応関係に基づいて生成される。床境界および天井境界は同一の頂点数と順序を持つ閉ループであるため、対応する上下の隣接頂点対を用いることで壁面を構成できる。

具体的には、床境界の隣接する頂点  $\mathbf{P}_i, \mathbf{P}_{i+1}$  と、それらに対応する天井境界頂点  $\mathbf{P}_i^{\text{ceil}}, \mathbf{P}_{i+1}^{\text{ceil}}$  を結ぶことで、四角形メッシュを定義する。このようにして生成された床面および壁面の幾何構造により、屋内環境の簡易 3 次元ワイヤーフレームモデルが構築される。

### 2.6.3 UV 座標の定義

UV 座標とは、3 次元モデルの表面にテクスチャ画像を対応付けるために用いられる 2 次元座標である。これは、それぞれのテクスチャ画像ごとに独立して割り当てられた局所座標系であり、画像の横軸を  $u$ 、縦軸を  $v$  とし、全領域が  $[0, 1]$  の範囲に正規化されて表現される。テクスチャマッピングにおいて、3 次元メッシュの幾何学的形状を保存したままテクスチャを割り当てるためには、メッシュごとにこの UV 空間上で適切な座標を定義する必要がある。

あるメッシュを構成する  $N$  個の頂点の世界座標を  $\mathbf{v}^{(i)} \in \mathbb{R}^3$  ( $i = 0, \dots, N - 1$ ) とする。まず、この面上に 2 つの直交する基底ベクトル ( $U$  軸ベクトル  $\mathbf{e}_u$  および  $V$  軸ベクトル  $\mathbf{e}_v$ ) を定義する。 $\mathbf{e}_u$  は、最初の辺の方向ベクトルとして次式で定義する。

$$\mathbf{e}_u = \frac{\mathbf{v}^{(1)} - \mathbf{v}^{(0)}}{\|\mathbf{v}^{(1)} - \mathbf{v}^{(0)}\|} \quad (2.13)$$

次に、面の法線ベクトル  $\mathbf{n}$  を、隣接する 2 辺の外積により求める。

$$\mathbf{n} = \frac{(\mathbf{v}^{(1)} - \mathbf{v}^{(0)}) \times (\mathbf{v}^{(2)} - \mathbf{v}^{(0)})}{\|(\mathbf{v}^{(1)} - \mathbf{v}^{(0)}) \times (\mathbf{v}^{(2)} - \mathbf{v}^{(0)})\|} \quad (2.14)$$

これらを用いて、 $U$  軸および法線  $\mathbf{n}$  の双方に直交するベクトルとして、 $V$  軸ベクトル  $\mathbf{e}_v$  を算出する。

$$\mathbf{e}_v = \mathbf{n} \times \mathbf{e}_u \quad (2.15)$$

得られた基底ベクトル  $\mathbf{e}_u, \mathbf{e}_v$  を用いて、各頂点  $\mathbf{v}^{(i)}$  をこの平面上に投影し、一時的な 2 次元座標  $(u'_i, v'_i)$  を計算する。これは、頂点  $\mathbf{v}^{(0)}$  を原点とした相対ベクトルと基底ベクトルの内積により求められる。

$$u'_i = (\mathbf{v}^{(i)} - \mathbf{v}^{(0)})^\top \mathbf{e}_u, \quad v'_i = (\mathbf{v}^{(i)} - \mathbf{v}^{(0)})^\top \mathbf{e}_v \quad (2.16)$$

最後に、得られた局所座標群が幾何学的形状を維持したまま単位正方形領域  $[0, 1]^2$  に収まるよう正規化を行う。全頂点の座標範囲に基づき、以下の変換式により最終的な UV 座標  $(u_i, v_i)$  を決定する。

$$u_i = \frac{u'_i - u'_{\min}}{S}, \quad v_i = \frac{v'_i - v'_{\min}}{S} \quad (2.17)$$

ここで、 $u'_{\min}, v'_{\min}$  は各軸における座標の最小値であり、 $S$  は座標群の最大辺長 ( $S = \max_k \{\max(u'_k - u'_{\min}, v'_k - v'_{\min})\}$ ) を表す。これにより、メッシュの実空間での形状比率を保った最大化された UV 座標が付与される。

## 2.7 全方位画像を用いたテクスチャ取得の方針

3次元モデルへのテクスチャ割り当てにおいては、モデル表面を十分に覆う視点から撮影された画像を効率的に取得することが重要である。一般的な透視投影カメラを用いる場合、多方向のテクスチャ情報を網羅するためには、カメラの向きを変えながら多数の画像を撮影する必要があり、撮影およびデータ管理に関わるコストが増大するという課題がある。

そこで本研究では、単一の撮影によって全周囲の視覚情報を取得可能な全方位カメラを用いる。全方位画像は、カメラ位置を中心とする前後・左右・上下の全周囲の風景を一度に記録した画像である。幾何学的には、カメラを中心とした球面上の情報として定義され、データ形式としては、一般に正距円筒図法 (Equirectangular Projection) などを用いて2次元平面画像に展開し保存されている。

この画像に対し、任意の視線方向を持つ仮想カメラを定義し、その画像平面へ画素を再投影することで、特定の方向に対応する透視投影画像を生成できる。この特性を利用して、実際に複数視点から撮影を行うことなく、任意の方向を向いた多数の画像群を計算機上で効率的に生成することが可能となる。

## 2.8 透視投影画像変換

本研究では、全方位画像を球面上の画素情報として扱う。生成する透視投影画像の各画素について、その視線方向に対応する全方位画像の画素値を取得することで、任意方向の画像を生成する。

正距円筒画像の幅および高さをそれぞれ  $W_e, H_e$  とする。全方位カメラの焦点距離  $f$  は全方位画像幅  $W_e$  を用いて次式で表される。

$$f = \frac{W_e}{2\pi} \quad (2.18)$$

生成する透視投影画像の幅および高さをそれぞれ  $W_p, H_p$  とし、透視投影画像上の画素座標を  $(u_p, v_p)$  とすると、この画素に対応する光軸方向を  $z$  軸とする3次元の視線ベクトル  $\mathbf{x}$  は次式で定義される。ただし、 $N[\cdot]$  はベクトルのノルムを1にする正規化する正規化作用素である。

$$\mathbf{x} = N\left[\begin{pmatrix} u_p - W_p/2 \\ v_p - H_p/2 \\ f \end{pmatrix}\right] \quad (2.19)$$

カメラの視線方向を変更するための回転行列  $\mathbf{R}$  を定義する。本研究では光軸周りの回転は考慮せず、水平方向の回転  $\theta_{eye}$  および垂直方向の回転  $\phi_{eye}$  のみにより姿勢を決定する。これに対する回転行列  $\mathbf{R}(\theta_{eye}, \phi_{eye})$  は次式で与えられる。

$$\mathbf{R}(\theta_{eye}, \phi_{eye}) = \begin{pmatrix} \cos \theta_{eye} & 0 & \sin \theta_{eye} \\ 0 & 1 & 0 \\ -\sin \theta_{eye} & 0 & \cos \theta_{eye} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi_{eye} & -\sin \phi_{eye} \\ 0 & \sin \phi_{eye} & \cos \phi_{eye} \end{pmatrix} \quad (2.20)$$

回転後の視線ベクトル  $\mathbf{x}' = (X', Y', Z')^\top$  は、次式で計算される。

$$\mathbf{x}' = \mathbf{R}(\theta_{eye}, \phi_{eye})\mathbf{x} \quad (2.21)$$

得られた視線ベクトル  $\mathbf{x}' = (X, Y, Z)$  を用いて、全方位画像の球面座標系  $(\theta_e, \phi_e)$  は次式で計算される。ただし、 $(X', Y', Z')^\top$  は  $(X, Y, Z)^\top$  のノルムを 1 に正規化したものである。

$$\theta_e = \tan^{-1} \left( \frac{X'}{Z'} \right) \quad (2.22)$$

$$\phi_e = \sin^{-1} \left( \frac{Y'}{\sqrt{X'^2 + Y'^2 + Z'^2}} \right) \quad (2.23)$$

ここで、全方位画像（正距円筒画像）の幅を  $W_e$ 、高さを  $H_e$  とすると、球面座標  $(\theta_e, \phi_e)$  に対応する全方位画像上の画素座標  $(u_e, v_e)$  は次式で与えられる。

$$u_e = (\theta_e + \pi) \frac{W_e}{2\pi} \quad (2.24)$$

$$v_e = \left( \phi_e + \frac{\pi}{2} \right) \frac{H_e}{\pi} \quad (2.25)$$

以上の手順により、全方位画像の画素  $(u_e, v_e)$  の輝度値を参照することで、透視投影画像上の画素  $(u_p, v_p)$  の画素値を決定する。

## 2.9 透視投影画像を用いた全方位カメラの位置姿勢推定

本節では、点特徴および線特徴の双方を用いてカメラ位置姿勢を推定するため、菅谷ら [17] による直交射影モデルに基づく手法を採用する。この手法は、点の共線性誤差と線の共面性誤差を統一的な枠組みで定式化することで、点と線の両特徴を同時に扱い、カメラ姿勢を推定するものである。直交射影の共線性と共面性に基づくカメラ姿勢推定に必要な入力変数は以下の通りである。なお、透視投影画像に投影されない点は入力から除外する。

- $\mathbf{p}_i$  : 世界座標系における空間点  $i$  の座標。
- $\mathbf{v}_i$  :  $\mathbf{p}_i$  に対応する画像上の特徴点ベクトル。
- $\mathbf{d}_i$  : 世界座標系における直線  $L_i$  の方向ベクトル。
- $\mathbf{r}_i$  : 世界座標系における直線  $L_i$  上の点の座標。
- $\mathbf{n}_i$  : 直線  $L_i$  に対応する画像上の直線の法線ベクトル。

画像上の特徴点ベクトル  $\mathbf{v}_i$  は、透視投影画像上で  $\mathbf{p}_i$  に対応する画素を選択し、前節で定義したカメラモデルを用いてカメラ座標系上のベクトルとして表現することで得られる。まず、画像中心を原点とした画素座標  $(u', v')$  は次式で与えられる。

$$u' = u_p - \frac{W_p}{2}, \quad v' = v_p - \frac{H_p}{2} \quad (2.26)$$

これに焦点距離  $f$  を合わせ、 $\mathbf{v}_i = (u', v', f)^\top$  とする。直線の方向ベクトル  $\mathbf{d}_i$  は、直線を構成する 2 つの 3 次元点  $\mathbf{p}_i, \mathbf{p}_{i+1}$  を用いて、次式で計算される。

$$\mathbf{d}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|} \quad (2.27)$$

同様に、画像上の直線の法線ベクトル  $\mathbf{n}_i$  は、直線端点に対応する 2 つの特徴点ベクトル  $\mathbf{v}_i, \mathbf{v}_{i+1}$  の外積により求める。

$$\mathbf{n}_i = \frac{\mathbf{v}_i \times \mathbf{v}_{i+1}}{\|\mathbf{v}_i \times \mathbf{v}_{i+1}\|} \quad (2.28)$$

これらの入力を用いて、複数視線方向の透視投影画像に対する目的関数を反復的に最小化することで、自己位置推定を行う。最終的に得られるカメラの回転行列  $\mathbf{R}$  と並進ベクトル  $\mathbf{t}$  は、世界座標系の3次元座標をカメラ座標系の座標に変換する行列である。

## 2.10 メッシュの座標系変換と投影

テクスチャを割り当てるメッシュの頂点座標を、世界座標系から全方位画像に基づいて生成した透視投影画像の画像座標系へと変換する。まず、メッシュを構成する頂点の世界座標  $\mathbf{v}_{\text{world}}^{(i)} \in \mathbb{R}^3$  ( $i = 1, 2, 3$ ) を、カメラの回転行列  $\mathbf{R}$  および並進ベクトル  $\mathbf{t}$  を用いてカメラ座標系へ変換する。

$$\mathbf{v}_{\text{cam}}^{(i)} = \mathbf{R}\mathbf{v}_{\text{world}}^{(i)} + \mathbf{t} \quad (2.29)$$

変換後の頂点座標を用いて、メッシュの重心位置ベクトル  $\mathbf{c}_{\text{cam}}$  を次式で求める。

$$\mathbf{c}_{\text{cam}} = \frac{1}{3} \sum_{i=1}^3 \mathbf{v}_{\text{cam}}^{(i)} \quad (2.30)$$

この目的のため、メッシュ重心方向  $\mathbf{c}_{\text{cam}}$  を視線方向とする新たな回転行列  $\mathbf{R}_{\text{proj}}$  を構成する。 $\mathbf{R}_{\text{proj}}$  の各列ベクトルとなる直交基底  $\mathbf{x}_{\text{proj}}, \mathbf{y}_{\text{proj}}, \mathbf{z}_{\text{proj}}$  は、Gram-Schmidt の直交化法に基づき、以下の手順で算出する。まず、重心方向ベクトルを正規化し、新たなZ軸  $\mathbf{z}_{\text{proj}}$  とする。

$$\mathbf{z}_{\text{proj}} = \frac{\mathbf{c}_{\text{cam}}}{\|\mathbf{c}_{\text{cam}}\|} \quad (2.31)$$

次に、決定したZ軸  $\mathbf{z}_{\text{proj}}$  を基準とし、世界座標系の上方向ベクトル  $\mathbf{u} = (0, 1, 0)^\top$  に対して直交化を行うことでY軸  $\mathbf{y}_{\text{proj}}$  を求める。具体的には、以下の式により  $\mathbf{u}$  から  $\mathbf{z}_{\text{proj}}$  方向への射影成分を除去し、正規化する。

$$\mathbf{y}' = (\mathbf{I} - \mathbf{z}_{\text{proj}}\mathbf{z}_{\text{proj}}^\top) \mathbf{u}, \quad \mathbf{y}_{\text{proj}} = \frac{\mathbf{y}'}{\|\mathbf{y}'\|} \quad (2.32)$$

最後に、 $\mathbf{y}_{\text{proj}}$  と  $\mathbf{z}_{\text{proj}}$  の外積によりX軸  $\mathbf{x}_{\text{proj}}$  を求め、これらを並べて回転行列  $\mathbf{R}_{\text{proj}}$  を定義する。

$$\mathbf{x}_{\text{proj}} = \mathbf{y}_{\text{proj}} \times \mathbf{z}_{\text{proj}}, \quad \mathbf{R}_{\text{proj}} = [\mathbf{x}_{\text{proj}} \quad \mathbf{y}_{\text{proj}} \quad \mathbf{z}_{\text{proj}}] \quad (2.33)$$

この回転行列を用いて、カメラ座標系上の頂点を回転させる。

$$\mathbf{v}_{\text{proj}}^{(i)} = \mathbf{R}_{\text{proj}}^\top \mathbf{v}_{\text{cam}}^{(i)} \quad (2.34)$$

回転後の3次元点をカメラ内部パラメータ行列  $\mathbf{K}$  を用いて画像座標系へ投影する。ここで、初期設定の画像サイズ  $(W, H)$  および内部パラメータに対し、メッシュ全体が画像内に収まらない場合が発生する可能性がある。そのため、投影された画素座標の画像中心からの最大距離に基づき、透視投影画像の画像サイズおよび内部パラメータを更新する。更新後の画像幅  $W_{\text{tex}}$  および高さ  $H_{\text{tex}}$  は、画像中心  $(W/2, H/2)$  からの最大オフセット量を用いて決定し、それに伴い内部パラメータ行列の光学中心  $(c_x, c_y)$  も画像中央へ移動させる。この処理は、焦点距離を維持した

まま画角のみを拡大することに相当し、解像度を低下させることなくメッシュ全体のテクスチャ取得を可能にする。

### 2.10.1 テクスチャ候補の評価

メッシュ重心までの距離が遠い、あるいはメッシュを斜めから撮影している場合は、テクスチャの解像度不足や歪みにより視覚的品質が劣化する原因となる。そのため、テクスチャ割り当ての優先度を決定するための指標として、以下の幾何学的特徴量を計算する。

- メッシュ重心までの距離  $d$

$$d = \|\mathbf{c}_{\text{cam}}\| \quad (2.35)$$

- 視線方向と法線方向のなす角  $\theta$

まず、メッシュを構成する3つの頂点座標  $\mathbf{v}_{\text{cam}}^{(0)}, \mathbf{v}_{\text{cam}}^{(1)}, \mathbf{v}_{\text{cam}}^{(2)}$  を用いて、メッシュの法線ベクトル  $\mathbf{n}_{\text{cam}}$  を算出する。

$$\mathbf{n}_{\text{cam}} = \frac{(\mathbf{v}_{\text{cam}}^{(1)} - \mathbf{v}_{\text{cam}}^{(0)}) \times (\mathbf{v}_{\text{cam}}^{(2)} - \mathbf{v}_{\text{cam}}^{(0)})}{\|(\mathbf{v}_{\text{cam}}^{(1)} - \mathbf{v}_{\text{cam}}^{(0)}) \times (\mathbf{v}_{\text{cam}}^{(2)} - \mathbf{v}_{\text{cam}}^{(0)})\|} \quad (2.36)$$

これを用い、メッシュ重心方向ベクトル（視線ベクトル） $\mathbf{c}_{\text{cam}}$  とのなす角  $\theta$  を次式で求める。

$$\theta = \arccos \left( \frac{\mathbf{c}_{\text{cam}}^T \mathbf{n}_{\text{cam}}}{\|\mathbf{c}_{\text{cam}}\| \|\mathbf{n}_{\text{cam}}\|} \right) \quad (2.37)$$

距離  $d$  が小さく、かつ角度  $\theta$  が小さいテクスチャを優先的に選択することで、視覚的品質の高いテクスチャ割り当てを実現する。

## 2.11 テクスチャ画像の形状変換

透視投影された入力画像において、各メッシュに対応する領域は、視点方向や撮影条件に起因する透視歪みを含んでおり、画像座標系上では不規則な多角形として観測される。この歪んだ領域をそのままテクスチャとして使用すると、3次元モデルの幾何形状と整合せず、適切なマッピングが行えない。そこで本研究では、画像上のメッシュ領域に対して射影変換を適用し、各メッシュに定義されたUV座標系の幾何学的形状へと正規化する。

まず、入力画像上におけるメッシュ頂点の画素座標を  $\mathbf{p}_{\text{src}}^{(i)} = (x_i, y_i)^T$  ( $i = 1, \dots, 4$ ) とする。これらは、透視投影により歪んだ四角形を形成している。次に、変換先の座標  $\mathbf{p}_{\text{dst}}^{(i)}$  を決定する。本手法では、メッシュデータに付与されている正規化UV座標  $\mathbf{u}^{(i)} = (u_i, v_i)^T$  ( $0 \leq u_i, v_i \leq 1$ ) と、生成するテクスチャ画像の解像度  $(W_{\text{tex}}, H_{\text{tex}})$  を用いて、変換後のターゲット座標  $(x'_i, y'_i)$  を次式で定義する。

$$\mathbf{p}_{\text{dst}}^{(i)} = \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} u_i \cdot W_{\text{tex}} \\ v_i \cdot H_{\text{tex}} \end{pmatrix} \quad (2.38)$$

入力画像上の点  $\mathbf{p}_{\text{src}}^{(i)}$  をテクスチャ画像上の点  $\mathbf{p}_{\text{dst}}^{(i)}$  へ写像する射影行列  $\mathbf{H}$  は、同次座標系において以下の関係を満たす  $3 \times 3$  行列として定義される。

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} \sim \mathbf{H} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}, \quad \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (2.39)$$

この関係式を展開すると、1組の対応点につき以下の2つの線形方程式が得られる。

$$\begin{cases} h_{11}x_i + h_{12}y_i + h_{13} - h_{31}x_ix'_i - h_{32}y_ix'_i - h_{33}x'_i = 0 \\ h_{21}x_i + h_{22}y_i + h_{23} - h_{31}x_iy'_i - h_{32}y_iy'_i - h_{33}y'_i = 0 \end{cases} \quad (2.40)$$

定数倍の不定性を除くと、行列  $\mathbf{H}$  の自由度は8である。したがって、四角形メッシュの4頂点の対応関係を用いることで計8本の連立方程式を立てることができ、DLT法(Direct Linear Transformation)を適用することで  $\mathbf{H}$  の各成分を一意に決定できる。

実際にテクスチャ画像を生成する際は、算出された行列  $\mathbf{H}$  を用いて逆写像を行う。出力画像の各画素座標  $(u, v)$  に対し、逆行列  $\mathbf{H}^{-1}$  を乗算することで、入力画像上の対応座標  $(\hat{x}, \hat{y})$  を求める。

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} \sim \mathbf{H}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad (2.41)$$

算出された座標  $(\hat{x}, \hat{y})$  は一般に実数値となるため、入力画像における近傍4画素の輝度値を用いた双線形補間(Bilinear Interpolation)により、当該画素の輝度値を決定する。この逆写像方式を採用することで、変換後の画像に画素抜けが生じるのを防ぎ、滑らかな正規化テクスチャ画像  $I_{dst}$  を生成することが可能となる。

## 2.12 テクスチャの視覚的品質の改善

メッシュ重心に対してカメラ視線方向が大きな角度をもつ場合、そのカメラから取得した透視投影画像によるテクスチャを割り当てると、射影歪みの影響が大きくなり、テクスチャ境界部における画素の不連続が顕著となる。本研究では、このような射影歪みに起因する視覚的品質の低下を緩和するため、隣接するテクスチャ間の境界領域に対してブレンド処理を適用する。一方、メッシュ重心に対してカメラ視線方向が正面に近い場合には、そのテクスチャを優先的に割り当てることで射影歪みの影響を抑えることができる。そのため、本手法では、メッシュに対して正面方向から撮影されたテクスチャが割り当てられていない場合に限り、左右方向から撮影されたテクスチャを一枚ずつ用いて、これらの境界に対してブレンド処理を適用する。

### 2.12.1 カメラとメッシュの相対位置関係の導出

テクスチャのブレンド処理を行う際、カメラに対してメッシュが左右どちらの方向にあるかを判定する必要がある。そこで、カメラから見たメッシュの相対的な左右位置を表す指標  $s \in \{-1, 0, 1\}$  を定義する。

まず、カメラ座標系における物理的な上方向ベクトル  $\mathbf{u}_{up} = (0, -1, 0)^\top$  を定義する。この  $\mathbf{u}_{up}$  とメッシュの法線ベクトル  $\mathbf{n}_{cam}$  の外積により、メッシュ平面に沿った右方向ベクトル  $\mathbf{r}$  を算出する。

$$\mathbf{r} = \mathbf{u}_{\text{up}} \times \mathbf{n}_{\text{cam}} \quad (2.42)$$

次に、正規化されたカメラの視線ベクトル  $\mathbf{v}_{\text{view}} = \mathbf{c}_{\text{cam}} / \|\mathbf{c}_{\text{cam}}\|$  を用いて、次式により判定を行う。ここで、 $-\mathbf{v}_{\text{view}}$  はメッシュ重心からカメラ中心へ向かうベクトルを表す。

$$s = \text{sgn} \left( (-\mathbf{v}_{\text{view}})^{\top} \frac{\mathbf{r}}{\|\mathbf{r}\|} \right) \quad (2.43)$$

この  $s$  の符号により、カメラがメッシュの正面に向かって左側 ( $s = -1$ )、あるいは右側 ( $s = 1$ ) のいずれに位置しているかを記述する。

### 2.12.2 テクスチャのブレンド処理

視線方向に基づく左右情報  $s \in \{-1, +1\}$  を用いてブレンド領域における左右のテクスチャ割り当てを決定する。ここで、ブレンドを行う画像領域の左側を担当するテクスチャを  $I_L$ 、右側を担当するテクスチャを  $I_R$  と定義する。

一般に、対象物を斜め方向から撮影した際、カメラ正面から離れるほど射影歪みが大きくなる。視線方向左側にテクスチャがある場合 ( $s = -1$ )、取得されたテクスチャ画像の右側面は対象の正面側に近いため、比較的歪みが小さい。したがって、この画像はブレンド領域の右側  $I_R$  を担当する。逆に、視線方向右側にテクスチャがある場合 ( $s = +1$ )、テクスチャ画像の左側面において歪みが小さくなることから、ブレンド領域の左側  $I_L$  を担当する。

ブレンド処理は画像全体ではなく、画像幅  $W$  の中央付近に設定した水平方向の一定範囲に限定して適用する。画素位置  $(x, y)$  におけるブレンド結果  $I(x, y)$  は、次式で与えられる。

$$I(x, y) = w_L(x) I_L(x, y) + w_R(x) I_R(x, y) \quad (2.44)$$

ただし、重み係数は常に正規化条件  $w_L(x) + w_R(x) = 1$  を満たすものとする。ブレンド領域内において、テクスチャの切り替わりを滑らかにするため、シグモイド関数に基づき右側領域用の重み  $w_R(x)$  を次式で定義する。

$$w_R(x) = \frac{1}{1 + \exp(-k \tilde{x})} \quad (2.45)$$

ここで、 $\tilde{x} \in [-1, 1]$  はブレンド領域内で正規化された水平方向位置を表し、 $k$  は重み変化の急峻さを制御するパラメータである。左側領域用の重みは  $w_L(x) = 1 - w_R(x)$  として一意に定まる。

この重み定義により、画像の左端から右端に向かうにつれて、支配的なテクスチャが  $I_L$  から  $I_R$  へと滑らかに遷移する。領域外では一方のテクスチャのみを用いることで、テクスチャ本来の解像度および幾何的整合性を維持しつつ、境界付近の不連続を効果的に低減する。

# 第3章 特徴点マッチングに基づく自己位置推定

## 3.1 特徴点検出およびマッチングの概要

本研究では、自己位置推定を行うために、前節までに生成した簡易モデルのテクスチャ画像と入力画像との対応付けを行う。具体的には、両画像から特徴点を検出し、それらの対応関係を構築することで、入力画像上の2次元特徴点と、3次元モデルに対応付けられたテクスチャ上の点との対応を得る。これに基づき、カメラの位置および姿勢を推定する。なお、本研究では側面のテクスチャのみを特徴点マッチングの対象とし、床面は除外する。これは、床面では同一のテクスチャパターンが連續して用いられることが多く、特徴点の識別による対応付けが困難であるためである。

特徴点検出およびマッチング手法は、大きく従来手法と学習ベース手法の二つに分類される。従来手法は、画像の輝度勾配や局所構造に基づいて特徴点および記述子を設計する手法であり、計算過程が明確であるという利点を持つ。一方で、撮影条件の変化や環境の差異に対するロバスト性には限界がある。これに対して、学習ベース手法は、深層学習を用いて特徴点検出やマッチングの過程をデータから学習する手法である。画像全体の文脈情報を考慮した対応付けが可能であり、テクスチャの少ない環境においても比較的安定した対応が得られる利点を持つ。一方で、従来手法と比較して計算コストが増大する傾向にある。本研究では、これら双方のアプローチについて検討を行う。

### 3.1.1 共通の前処理

本研究では、従来手法と学習ベース手法のいずれにおいても、特徴点検出に先立ち以下の共通した前処理を適用する。

まず、入力画像および簡易モデルのテクスチャ画像をグレースケール化する。

次に、高解像度画像の処理に対応するため、タイリング処理を導入する。画像サイズが大きい場合、一度に処理を行うとメモリ制約を受ける場合や、微細な特徴が見逃される場合がある。そのため、画像を複数のタイル領域に分割し、各タイルに対して個別に特徴点検出を行った後、それらの検出結果を統合することで、画像全体を網羅する特徴点集合を抽出する。

また、処理の高速化を図るため、静的なデータであるテクスチャ画像については事前計算を行う。自己位置推定の実行時に都度計算するのではなく、事前に特徴点検出および記述子の算出を済ませて保存しておくことで、マッチング時の計算負荷を大幅に低減し、システム全体の処理速度を向上させている。

## 3.2 従来手法による特徴点マッチング

### 3.2.1 特徴点検出

本研究では、代表的な局所特徴点検出手法として SIFT および AKAZE を用いる。SIFT は、ガウシアン平滑化により構築されるスケール空間上で極値点を検出し、スケールおよび回転に対して不变な特徴点を得る手法である [18]。各特徴点に対しては、周囲の勾配分布に基づく特徴量が計算され、高い識別性能を持つ。AKAZE は、非線形スケール空間に基づいて特徴点を検出手法であり、Fast Explicit Diffusion を用いることで高速な処理を可能としている [19]。記述子にはバイナリ表現が用いられ、後段のマッチング処理を効率的に行うことができる。

### 3.2.2 特徴点マッチング

従来手法では、最近傍探索に基づくマッチング手法を採用する。記述子間の距離を計算し、最も距離の近い特徴点同士を対応点として選択することで、初期的な対応点集合を得る。SIFT のような実数値記述子に対しては KD-tree を用いた探索を行い、AKAZE のようなバイナリ記述子に対しては LSH (Locality Sensitive Hashing) を用いた探索を行うことで、記述子の特性に応じた最近傍探索を実現している。

### 3.2.3 マッチングの精度向上

最近傍探索によって得られる対応点集合には、特徴量の類似度のみでは除去できない誤対応が含まれる可能性がある。そこで本研究では、複数の手法を組み合わせることで、マッチング精度の向上を図る。まず、誤対応の除去手法として交差検証 (Cross Check) を適用する。画像間の双方向マッチングを行い、互いに第一近傍となるペアのみを採用することで、整合性の高い対応点を抽出する。その後、RANSAC を用いて幾何的整合性に基づく外れ値除去を行う。対応点集合から射影変換モデルを推定し、モデルに適合しない対応点を除外することで、幾何的に整合した対応点集合を得る。これらの処理により、自己位置推定に用いる対応点の精度を向上させ、後段の位置および姿勢推定における安定性の確保を図る。

## 3.3 学習ベース手法による特徴点マッチング

### 3.3.1 特徴点検出

学習ベース手法における特徴点検出手法として SuperPoint を用いる。SuperPoint は、画像中の特徴点 (Interest Point) とそれらに対応する記述子 (Descriptor) を同時に推定する特徴点検出手法である。SuperPoint のアーキテクチャを図 3.1 に示す。画像全体を入力とする畳み込み型ネットワークとして構成されており、共有されたエンコーダによって特徴を抽出した後、特徴点検出用と記述子生成用の 2 つのデコーダに分岐することで、单一のフォワードパスで両者を出力する。この構成により、検出と記述を個別に行う手法と異なり、両タスク間で計算および特徴表現を効率的に共有することが可能となっている [20]。従来手法では、輝度勾配や局所構造といった人手設計された指標に基づいて特徴点検出が行われるのに対し、大量の画像データを用いた学習による安定した特徴点検出が可能である。

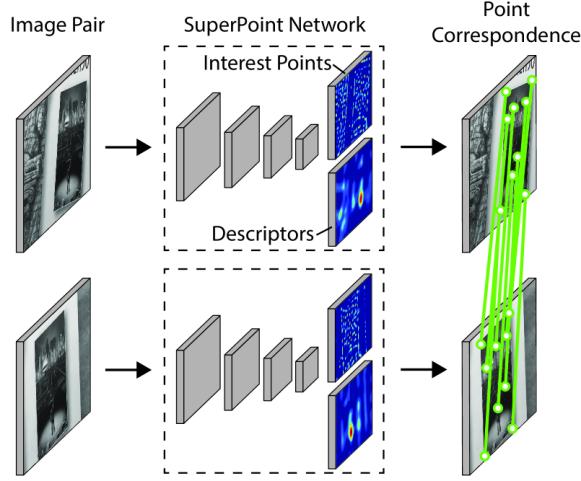


図 3.1: SuperPoint のアーキテクチャ. 出典 : DeTone et al.(2018)[20] Fig. 1

### 3.3.2 特徴点マッチング

学習ベース手法における特徴点マッチング手法として SuperGlue を用いる。SuperGlue は、注意機構付きグラフニューラルネットワークと最適マッチング層の 2 つから構成される特徴点マッチング手法である。SuperGlue のアーキテクチャを図 3.2 に示す。注意機構付きグラフニューラルネットワークでは、キーポイントの位置と記述子を統合した特徴表現を自己注意および相互注意を交互に用いて段階的に更新する。最適マッチング層では、キーポイント間のスコア行列に対し、いずれのキーポイントとも対応しない点を吸収するための仮想ノードである dustbin を追加した上で、Sinkhorn アルゴリズムにより最適な部分対応を推定する [21]。従来手法では、各特徴点を独立に対応付ける最近傍探索が行われるのに対し、特徴点集合全体の文脈情報を考慮した対応付けが可能である。

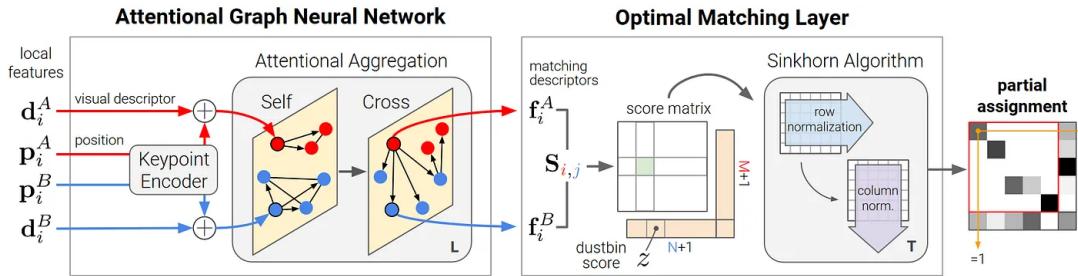


図 3.2: SuperGlue のアーキテクチャ. 出典 : Sarlin et al.(2020)[21] Fig. 3

### 3.3.3 マッチングの精度向上

SuperGlue により得られた対応点集合には、学習に基づく推定結果であるため、局所的に信頼度の低い対応点が含まれる可能性がある。まず、従来手法における交差検証に相当する処理として、SuperGlue の matching score に基づく対応点の選別を行う。matching score は、各特徴点対が正しく対応している可能性の高さを表す指標であり、スコアが所定の閾値以上となる対応点

のみを採用することで、整合性の高い対応点を抽出する。続いて、従来手法と同様に RANSAC を用いて外れ値除去を行う。これにより、学習ベース手法においても従来手法と同等の基準で誤対応を抑制し、自己位置推定に用いる対応点の精度を向上させる。

### 3.4 テクスチャ画像座標から世界座標への変換

簡易モデルのテクスチャ画像と入力画像の対応点は、いずれも画像平面上の 2 次元座標として表現されている。自己位置推定を行うためには、テクスチャ画像の 2 次元特徴点を世界座標系における 3 次元座標へ変換する必要がある。本研究では、テクスチャに割り当てられた画像座標系と、対応する四角形メッシュの四隅の世界座標を用いて、画像座標系から世界座標系への変換を行う。以下に、2 次元座標を 3 次元座標へ変換する手順を示す。

#### 3.4.1 2 次元座標から 3 次元座標への変換

本節では、テクスチャ画像上で検出された特徴点を、世界座標系の 3 次元座標へ逆投影する手法について述べる。まず、特徴点を取得したテクスチャに割り当てられた四角形メッシュを定義する。世界座標系において、当該メッシュの 4 頂点を左上から順に時計回りで  $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$  とする。このとき、メッシュ平面を張る 2 つの基底ベクトル  $\mathbf{B}_1, \mathbf{B}_2$  は、始点  $\mathbf{P}_0$  を基準として次式で表される。

$$\mathbf{B}_1 = \mathbf{P}_1 - \mathbf{P}_0, \quad \mathbf{B}_2 = \mathbf{P}_3 - \mathbf{P}_0 \quad (3.1)$$

次に、第 2.6.3 節で定義した UV 座標系における、上記頂点との対応関係を用いる。四角形メッシュの各頂点  $\mathbf{P}_i$  には、テクスチャ画像上の対応位置を示す UV 座標  $\mathbf{t}_i = (u_i, v_i)$  ( $i = 0, \dots, 3$ ) が定義されている。この対応関係に基づき、当該メッシュ領域が UV 空間上で占める幅  $w_{uv}$  および高さ  $h_{uv}$  を次式で定義する。

$$w_{uv} = u_1 - u_0, \quad h_{uv} = v_3 - v_0 \quad (3.2)$$

ここで、解像度  $(W, H)$  のテクスチャ画像上において検出された特徴点のピクセル座標を  $(x, y)$  とする。この特徴点がメッシュ内のどの位置にあるかを特定するため、正規化パラメータ  $\alpha, \beta \in [0, 1]$  へと変換する。具体的には、ピクセル座標を UV 座標  $(x/W, y/H)$  へと正規化した上で、メッシュ始点  $(u_0, v_0)$  からの相対位置として次式により算出する。

$$\alpha = \frac{x/W - u_0}{w_{uv}}, \quad \beta = \frac{y/H - v_0}{h_{uv}} \quad (3.3)$$

最後に、得られたパラメータ  $\alpha, \beta$  を、世界座標系における基底ベクトルの線形結合に用いることで、特徴点に対応する 3 次元座標  $\mathbf{P}$  を算出する。

$$\mathbf{P} = \mathbf{P}_0 + \alpha \mathbf{B}_1 + \beta \mathbf{B}_2 \quad (3.4)$$

### 3.5 自己位置推定

入力画像上の2次元特徴点と、それに対応するテクスチャ上の3次元特徴点の対応関係を用いて、自己位置推定を行う。

本研究では、松下らによって提案された、直交射影誤差に基づくPnP問題に対する大域最適解の計算手法を用いる[22]。一般的な反復的最適化手法では、初期値依存により局所最適解に陥る可能性があり、複数の解が存在する場合にそれらを網羅的に求めることが困難である。また、対応点の配置によっては収束までの反復回数が増加し、計算時間が長くなるため、リアルタイム処理への適用が難しい。これに対し本手法は、Cayley変換を用いた回転行列の表現により制約条件を除去し、グレブナー基底を用いて解を計算することで、非反復的大域最適解を求めることが可能である。これにより、計算時間の削減と、数学的にあり得る複数の解候補の同時導出を実現する。

本研究では、点特徴の対応のみを入力とする自己位置推定を行う。入力は、画像上の2次元特徴点座標に焦点距離情報を加えたベクトル $(x, y, f)$ と、それに対応する世界座標系における3次元特徴点 $(X, Y, Z)$ である。自己位置推定処理の結果として、虚数解を除いたすべての解候補に對し、目的関数値 $J$ 、回転行列 $\mathbf{R}$ 、および並進ベクトル $\mathbf{t}$ が出力される。

得られた複数の解候補の中から、本研究では幾何学的制約および推定誤差評価に基づき、以下の手順で最終的な自己位置を決定する。

#### 1. 天地反転解の除外

大域最適解法により得られた解候補には、数学的には正しいが物理的にあり得ない「天地が反転した解」が含まれる場合がある。本研究のカメラ座標系では $Y$ 軸が画像下方向を向いており、世界座標系では $Z$ 軸が鉛直上向きに定義されている。カメラが正立している場合、世界座標系の $Z$ 軸（上方向）は、カメラ座標系においては $Y$ 軸の負の方向の成分を持つはずである。

回転行列 $\mathbf{R}$ の要素 $R_{23}$ は、世界座標系の $Z$ 軸ベクトル $(0, 0, 1)^\top$ をカメラ座標系へ変換した際の $Y$ 成分に相当する。したがって、以下の条件を満たす解のみを採用する。

$$R_{23} < 0 \quad (3.5)$$

#### 2. カメラ中心位置の算出

残った各解候補について、回転行列 $\mathbf{R}$ および並進ベクトル $\mathbf{t}$ から、世界座標系におけるカメラ中心位置 $\mathbf{C}$ を次式により算出する。

$$\mathbf{C} = -\mathbf{R}^\top \mathbf{t} \quad (3.6)$$

#### 3. 位置誤差による評価

現在位置の基準値 $\mathbf{C}_{gt}$ が既知である場合、推定されたカメラ位置 $\mathbf{C}$ とのユークリッド距離を、位置誤差 $e_{pos}$ として次式で定義する。

$$e_{pos} = \|\mathbf{C} - \mathbf{C}_{gt}\| \quad (3.7)$$

#### 4. 視線方向ベクトルの算出

各解候補について世界座標系における視線ベクトル $\mathbf{f}_{world}$ を次式で定義する。これはカメラ座標系における光軸ベクトル $(0, 0, 1)^\top$ を世界座標系へ逆変換したものである。

$$\mathbf{f}_{world} = \mathbf{R}^\top (0, 0, 1)^\top \quad (3.8)$$

## 5. 姿勢誤差による評価と最適解の選択

現在の視線方向の基準値  $\mathbf{f}_{\text{gt}}$  が既知である場合、推定されたカメラ視線方向  $\mathbf{f}_{\text{world}}$  とのなす角を、姿勢誤差  $e_{\text{rot}}$  として算出する。

$$e_{\text{rot}} = \arccos \left( \frac{\mathbf{f}_{\text{world}}^\top \mathbf{f}_{\text{gt}}}{\|\mathbf{f}_{\text{world}}\| \|\mathbf{f}_{\text{gt}}\|} \right) \quad (3.9)$$

最終的に、位置誤差  $e_{\text{pos}}$  および姿勢誤差  $e_{\text{rot}}$  が所定の閾値以下となる解候補のうち、姿勢誤差  $e_{\text{rot}}$  が最小となるものを最終的な自己位置推定結果として採用する。これは、姿勢誤差が大きい解は特徴点マッチングの局所的な誤りを含んでいる可能性が高いためである。現在位置のもしくは現在の視線方向の基準値が与えられていない場合、目的関数  $J$  の値が小さいものを自己位置推定結果として採用する。

# 第4章 自己位置推定結果を用いた屋内ナビゲーション

## 4.1 屋内ナビゲーションの方針

### 4.1.1 簡易モデルによる自己位置推定の課題

実際のカメラ入力を用いてリアルタイムに動作させる場合、特徴点マッチング処理に伴う計算コストや通信遅延により、推定位置の更新が滞り、端末の実際の位置と乖離する可能性がある。さらに、屋内環境では特徴の乏しい壁面や照明変化の影響により、常に十分な特徴量が得られるとは限らず、マッチングに失敗して自己位置が更新されない状況も想定される。また、屋内ナビゲーションでは利用者がモバイル端末を持って移動するため、自己位置推定には高いリアルタイム性と連続性が求められるが、特徴点マッチング単独ではこれらを常に保証することが困難である。

### 4.1.2 提案手法に基づく屋内ナビゲーションの基本方針

本研究では、特徴点マッチングに基づく自己位置推定結果を用いて、屋内空間におけるナビゲーションを実現することを目的とする。推定した自己位置情報は、ユーザに対して進行方向や目的地までの誘導情報を提示するために用いられる。

しかしながら、前述の通り特徴点マッチングに基づく手法は計算コストが高く、リアルタイム性や安定性の観点から単独での利用には課題がある。そのため、本研究では提案する自己位置推定手法をナビゲーションの「絶対的な位置補正」の基準情報として用いつつ、自己位置推定間の移動量推定には別の補間手段を導入する方針とする。具体的には、モバイル端末に標準的に搭載されている VIO (Visual-Inertial Odometry) による自己位置推定結果を併用する。VIO により高頻度かつ連続的な位置追跡を行い、特徴点マッチングに失敗した場合や推定結果が得られない期間においても、ナビゲーションの連続性を維持する。

### 4.1.3 Visual-Inertial Odometry (VIO) による自己位置推定

Visual-Inertial Odometry (VIO) は、端末に搭載された IMU (加速度計・ジャイロスコープ等の慣性センサ) とカメラから取得した画像情報を統合することで、端末の位置姿勢を高頻度に推定する手法である。本研究では、Apple 社が提供する ARKit を VIO の実行環境として用いる [23]。ARKit は、この VIO 技術を中心とした自己位置推定フレームワークであり、内部処理において局所的な特徴点マップを構築することで、モバイル端末単体での SLAM (Simultaneous Localization and Mapping) に相当する機能を実現している。ARKit による自己位置推定は、端末内のみで処理されるため外部インフラを必要とせず、高いリアルタイム性が確保される。一方で、長時間動作させた際の累積誤差（ドリフト）や、白い壁のみが映る場合などカメラ視野が環境特徴に乏しい状況では、推定精度が低下する可能性がある点に注意が必要である [24]。

ARKit では、ワールド座標系が内部的に定義されており、セッション初期化時の端末位置が原点として設定される。座標軸は右手系で定義されており、一般に重力と反対方向を Y 軸とし、初期化時のカメラの視線方向に基づいて Z 軸および X 軸が決定される。以降のフレームにおける端末の位置姿勢は、このワールド座標系に対する相対的な変換行列として逐次更新される。

#### 4.1.4 ARKit の座標系の定義

本研究では、ARKit によって定義されるワールド座標系を AR 座標系と呼ぶ。ここで、ユーザが定義する世界座標系を含めた各座標系の関係を図 4.1 に示す。

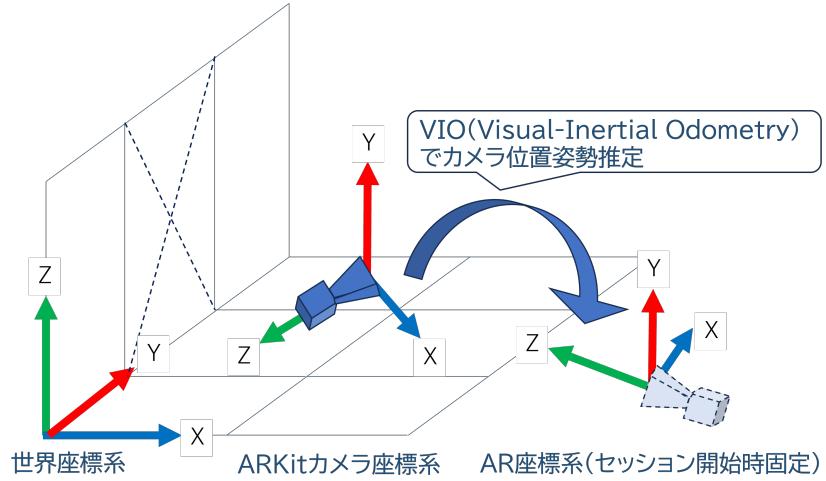


図 4.1: 世界座標系と ARKit カメラ座標系、AR 座標系の関係

ここで注意すべき点は、カメラ座標系の軸の符号定義が異なることである。OpenCV におけるカメラ座標系では、X 軸は右方向、Y 軸は下方向、Z 軸は奥行き方向が正となる。一方、ARKit におけるカメラ座標系では、X 軸は右方向、Y 軸は上方向、Z 軸は手前方向が正となる。ARKit のカメラ座標  $X_{camAR}$  と、本研究で用いる OpenCV のカメラ座標  $X_{camPC}$  の関係は、変換行列  $S$  を用いて次式で表される。

$$X_{camPC} = S X_{camAR}, \quad S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (4.1)$$

## 4.2 自己位置推定手法の比較検討方針

本研究では、屋内ナビゲーションにおける実用的な自己位置推定手法を明らかにするため、性質の異なる 3 つの手法を比較対象として設定した。これらは、提案手法単独の場合、既存のモバイル端末における標準的手法の場合、および両者を組み合わせた場合をそれぞれ代表するものである。

### 1. 提案手法（特徴点マッチングによる自己位置推定）

本研究の提案内容そのものの性能を評価するための基準として位置付ける。環境モデルと

の対応付けに基づいて絶対的な自己位置推定が可能である一方、計算時間や誤推定が課題となる可能性がある。

## 2. ARKit による自己位置推定

モバイル端末上で動作する一般的な自己位置推定手法として位置付ける。リアルタイム性に優れている一方、視覚情報に乏しい環境では精度が低下する可能性がある。

## 3. 複合的手法（提案手法 + ARKit）

提案手法を基準情報として用いつつ、推定が困難な状況では ARKit の推定結果を利用する方式である。これにより、両者の利点を活かしたより安定的かつ実用的な自己位置推定の可能性を検討する。

以上の3つの手法を比較することで、屋内ナビゲーションにおける実用的な自己位置推定手法の在り方について検証する。

## 4.3 屋内ナビゲーションシステムの構成

屋内ナビゲーションシステムの構成について説明する。システムの流れを図 4.2 に示す。

本システムは、ユーザインターフェースおよびセンシングを担うモバイル端末と、高負荷な自己位置推定処理を担う PC サーバーによって構成される。

処理の全体的な流れは以下の通りである。まず、ユーザがモバイル端末上で目的地を設定すると、ナビゲーションが開始される。モバイル端末は、取得したカメラ画像および ARKit のトラッキング情報を PC へ送信する。PC 側では、受信した画像と簡易 3 次元モデルを用いて特徴点マッチングを行い、世界座標系における厳密なカメラ位置姿勢を算出する。その結果がモバイル端末へ送信されることで、AR 座標系と世界座標系の間で整合性が取られ、目的地の AR 描画や、端末の自己位置のフィードバックが可能となる。以下の節で、各処理の詳細について述べる。

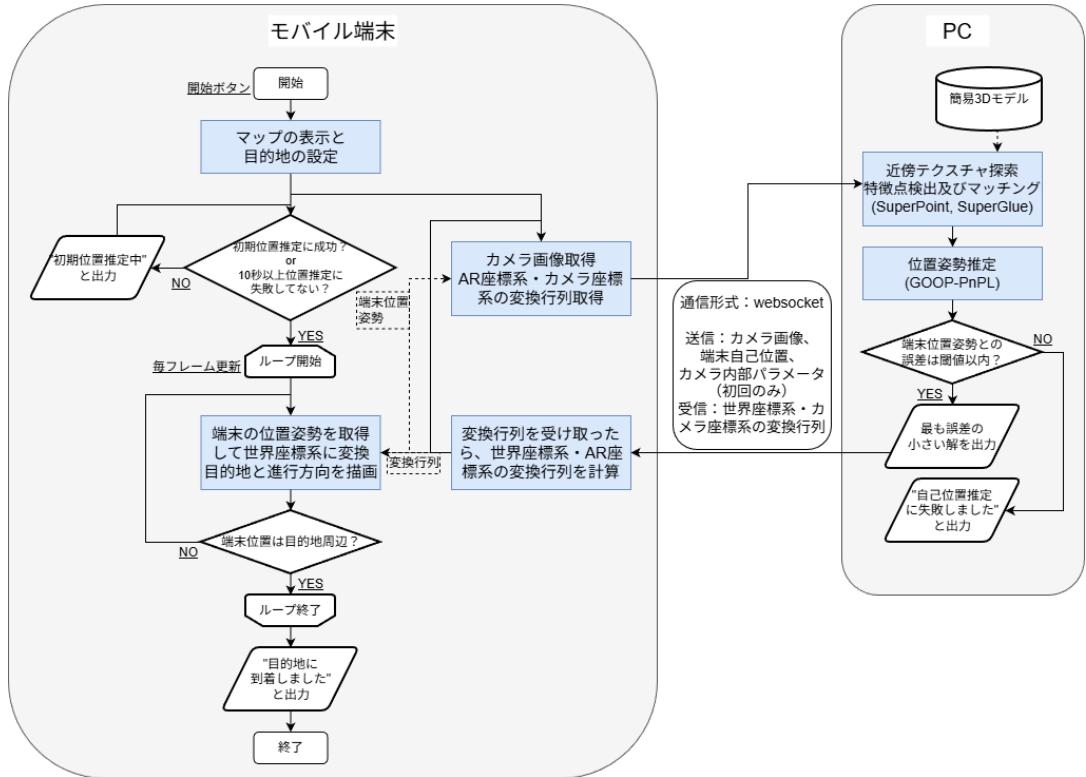


図 4.2: 屋内ナビゲーションシステムの構成

#### 4.3.1 目的地設定

マップ画像上のタップ操作により目的地を指定する方法を用いる。ユーザは表示されたマップ画像上を順にタップすることで、目的地までの経路点を指定する。各タップ位置は、マップ上の画像座標として取得される。入力が確定すると、あらかじめ算出したマップ上の画像座標と世界座標を対応付けるホモグラフィ行列を用いて、世界座標系へ変換する。変換後の世界座標列は、後続の自己位置推定およびナビゲーション処理における目的地情報として利用する。

#### 4.3.2 カメラ画像取得および端末と PC 間の通信

端末側では、カメラ画像と撮影時刻を取得し、初期位置推定期時は、カメラの内部パラメータも併せて取得する。また、撮影時における AR 座標系とカメラ座標系の変換行列を保存する。取得したカメラ画像は、通信負荷を低減するために圧縮処理を行う。2 回目以降の推定期では、ARKit により更新されたカメラの位置姿勢情報を用い、これらを JSON 形式にまとめ、WebSocket を用いた双方向通信によって PC へ送信する。

PC で推定されたカメラの位置姿勢情報は、同じく WebSocket を介して端末側へ送信され、世界座標系と AR 座標系の関係の更新に用いられる。なお、一定時間以上姿勢情報が受信されない場合には、自己位置が喪失したものと判断し、初期位置推定を再度実行する。

### 4.3.3 自己位置推定結果の受信および座標系の更新

端末側では、PC で推定された世界座標系とカメラ座標系の変換行列を受信する。受信した世界座標系とカメラ座標系の回転行列  $R_{\text{world} \rightarrow \text{camPC}}$  および並進ベクトル  $t_{\text{world} \rightarrow \text{camPC}}$  と、撮影時における AR 座標系とカメラ座標系の変換行列  $T_{\text{camAR} \rightarrow \text{AR}}$  を用いて、世界座標系と AR 座標系の変換行列を更新する。これにより、世界座標系と AR 座標系の相互変換が可能となり、PC による自己位置推定が困難なタイミングにおいても、ARKit による自己位置推定結果を継続的に利用できる。更新された座標系の関係は、目的の AR オブジェクトの描画や、カメラの位置姿勢を世界座標系へ変換する際に用いられる。変換行列を更新した後、一定時間経過後に再度カメラ画像を取得し、PC による自己位置推定を行う。

### 4.3.4 世界座標系と AR 座標系の相互変換

本システムにおいて、ARKit によって管理される AR 座標系と、簡易モデルが定義されている世界座標系は、原点および座標軸の定義が異なる。そのため、以下の 2 つの目的を実現するには、両座標系間の相互変換が必要となる。

#### 1. 世界座標系から AR 座標系の変換

世界座標系で定義された目的地を、モバイル端末のカメラ映像上の正しい位置に重畳表示するためには、目的地の座標を AR 座標系へ変換する必要がある。

#### 2. AR 座標系から世界座標系の変換

ARKit によって高頻度に追跡される端末の位置姿勢を PC へ送信し、特徴点マッチングによる自己位置推定結果と比較するためには、端末の位置姿勢を世界座標系へ変換する必要がある。

以下に、これらの変換を実現するための行列の導出過程を示す。

ARKit では、毎フレームカメラ変換行列  $T_{\text{camAR} \rightarrow \text{AR}}$  が与えられる。これは ARKit のカメラ座標系から AR 座標系への変換を表す  $4 \times 4$  の同次変換行列であり、次式で定義される。

$$T_{\text{camAR} \rightarrow \text{AR}} = \begin{pmatrix} R_{\text{camAR} \rightarrow \text{AR}} & t_{\text{camAR} \rightarrow \text{AR}} \\ \mathbf{0}^\top & 1 \end{pmatrix} \quad (4.2)$$

ここで、 $R_{\text{camAR} \rightarrow \text{AR}}$  は ARKit のカメラ座標系から AR 座標系への回転行列、 $t_{\text{camAR} \rightarrow \text{AR}}$  は AR 座標系におけるカメラ中心位置を表す並進ベクトルである。

よって、AR 座標系における座標  $X_{\text{AR}}$  は、ARKit のカメラ座標  $X_{\text{camAR}}$  から式 4.3 で求められる。

$$X_{\text{AR}} = R_{\text{camAR} \rightarrow \text{AR}} X_{\text{camAR}} + t_{\text{camAR} \rightarrow \text{AR}} \quad (4.3)$$

一方、世界座標系における座標  $X_{\text{world}}$  は、OpenCV のカメラ座標  $X_{\text{camPC}}$  から式 4.4 で求められる。

$$X_{\text{camPC}} = R_{\text{world} \rightarrow \text{cam}} X_{\text{world}} + t_{\text{world} \rightarrow \text{cam}} \quad (4.4)$$

ここで、式 (4.1) で示した通り、ARKit のカメラ座標系と OpenCV のカメラ座標系には軸定義の違いによる変換行列  $S$  が介在する。式 (4.4) および式 (4.1) を式 (4.3) に代入することで、世界座標系から AR 座標系への変換は次のように整理される。

$$X_{\text{AR}} = R_{\text{cam} \rightarrow \text{AR}} (S(R_{\text{world} \rightarrow \text{cam}} X_{\text{world}} + t_{\text{world} \rightarrow \text{cam}})) + t_{\text{cam} \rightarrow \text{AR}} \quad (4.5)$$

$$= (R_{\text{cam} \rightarrow \text{AR}} S R_{\text{world} \rightarrow \text{cam}}) X_{\text{world}} + (R_{\text{cam} \rightarrow \text{AR}} S t_{\text{world} \rightarrow \text{cam}} + t_{\text{cam} \rightarrow \text{AR}}) \quad (4.6)$$

したがって、世界座標系から AR 座標系への変換行列  $\mathbf{R}_{\text{world} \rightarrow \text{AR}}, \mathbf{t}_{\text{world} \rightarrow \text{AR}}$  は次式で求められる。

$$\mathbf{R}_{\text{world} \rightarrow \text{AR}} = \mathbf{R}_{\text{cam} \rightarrow \text{AR}} \mathbf{S} \mathbf{R}_{\text{world} \rightarrow \text{cam}} \quad (4.7)$$

$$\mathbf{t}_{\text{world} \rightarrow \text{AR}} = \mathbf{R}_{\text{cam} \rightarrow \text{AR}} \mathbf{S} \mathbf{t}_{\text{world} \rightarrow \text{cam}} + \mathbf{t}_{\text{cam} \rightarrow \text{AR}} \quad (4.8)$$

逆に、AR 座標系から世界座標系への変換行列  $\mathbf{R}_{\text{AR} \rightarrow \text{world}}, \mathbf{t}_{\text{AR} \rightarrow \text{world}}$  は、式 (4.8) の逆変換として次式で導出される。

$$\mathbf{X}_{\text{world}} = \mathbf{R}_{\text{world} \rightarrow \text{AR}}^\top \mathbf{X}_{\text{AR}} - \mathbf{R}_{\text{world} \rightarrow \text{AR}}^\top \mathbf{t}_{\text{world} \rightarrow \text{AR}} \quad (4.9)$$

$$= \mathbf{R}_{\text{AR} \rightarrow \text{world}} \mathbf{X}_{\text{AR}} + \mathbf{t}_{\text{AR} \rightarrow \text{world}} \quad (4.10)$$

ここで、各項は以下の通りである。

$$\begin{aligned} \mathbf{R}_{\text{AR} \rightarrow \text{world}} &= \mathbf{R}_{\text{world} \rightarrow \text{AR}}^\top \\ &= (\mathbf{R}_{\text{cam} \rightarrow \text{AR}} \mathbf{S} \mathbf{R}_{\text{world} \rightarrow \text{cam}})^\top \end{aligned} \quad (4.11)$$

$$\begin{aligned} \mathbf{t}_{\text{AR} \rightarrow \text{world}} &= -\mathbf{R}_{\text{world} \rightarrow \text{AR}}^\top \mathbf{t}_{\text{world} \rightarrow \text{AR}} \\ &= -\mathbf{R}_{\text{AR} \rightarrow \text{world}} (\mathbf{R}_{\text{cam} \rightarrow \text{AR}} \mathbf{S} \mathbf{t}_{\text{world} \rightarrow \text{cam}} + \mathbf{t}_{\text{cam} \rightarrow \text{AR}}) \end{aligned} \quad (4.12)$$

#### 4.3.5 目的地および進行方向オブジェクトの描画

更新された世界座標系と AR 座標系の関係に基づき、目的地および進行方向を示す AR オブジェクトの描画を行う。まず、マップから取得した経路上の目的地座標を世界座標系で取得し、前節で求めた変換式を用いて AR 座標系への変換を行うことで、AR 空間上における目的地の描画位置を算出する。

算出された AR 座標系上の位置に対して AR アンカーを生成する。AR アンカーは、AR 空間中の特定位置に仮想オブジェクトを安定して配置するための基準点として用いられ、端末の移動にかかわらず一貫した位置関係を保つ役割を持つ。生成したアンカーに目的地を示すオブジェクトを紐付けることで、目的地をカメラ映像上の正しい位置に重畳表示する。目的地が更新された場合には、既存のアンカーを削除し、新たな座標に基づいてアンカーを再生成する。

また、各フレームにおいてカメラの位置姿勢を取得し、カメラ前方に進行方向を示す矢印オブジェクトを配置する。矢印オブジェクトが目的地の方向を向くように姿勢を更新することで、ユーザに対して目的地までの直感的な誘導を実現する。さらに、カメラの位置および前方方向ベクトルを毎フレーム更新することで、自己位置推定結果を継続的に利用可能とし、PC による自己位置推定結果が適切であるかを判断するための指標としても用いることができる。

カメラ位置と目的地オブジェクトとの距離を算出し、一定の閾値以内に到達した場合には、目的地に到着したと判定する。経路上に複数の目的地が存在する場合には、次の目的地へ切り替え、対応する AR オブジェクトの更新を行う。

# 第5章 実験

## 5.1 実験準備

表 5.1 に実験で使用した機材および実行環境を示す。

表 5.1: 実験で使用した機材および実行環境

項目	内容
全方位カメラ	RECOH THETA 360
全方位カメラ解像度	11K ( $11008 \times 5504$ )
全方位カメラ焦点距離	約 1752
OS	Windows 11
CPU	Intel Core i7-14700
GPU	NVIDIA GeForce RTX 4060 Ti
メモリ	32 GB
使用言語	Python 3.9.13
主な使用ライブラリ	PyTorch 2.5.1+cu121 ,OpenCV 4.10
屋内ナビゲーション端末	iPad Pro 12.9 インチ (第 5 世代)
OS (iOS)	17.5
端末カメラ解像度	( $1920 \times 1440$ )
端末カメラ焦点距離	約 1595
屋内ナビゲーション実装環境	MacBook Air 13 インチ
使用言語	swift 5.10
主な使用ライブラリ	ARKit, SceneKit

簡易モデルの作成および屋内ナビゲーションに関する実験は、所属する大学の C 棟 5 階において実施した。当該フロアは廊下および複数の部屋から構成されており、単色壁面が連続するテクスチャの少ない領域が存在する。C 棟 5 階のフロアマップおよび、全方位画像の撮影位置を図 5.1 に示す。全方位画像は、一部の例外を除き、およそ 4 m 間隔で合計  $N$  箇所において撮影した。撮影時のカメラ高さは床面から 1.4 m である。

ただし、当該環境には類似した外観を有する領域が多く、特徴点マッチングに必要な情報が十分に得られない箇所が存在する。そこで本研究では、過度に環境のテクスチャ量を増加させることなく、最低限の特徴情報を得ることを目的として、壁面の一部に A2 サイズのポスターを掲示した。ポスターの掲示間隔はおよそ 4 m に 1 枚程度とし、環境全体が高テクスチャ化することを避けるよう配慮した。



図 5.1: 世界座標系とカメラ座標系、画像座標系の関係

各全方位カメラについては、2方向以上の透視投影画像を画角  $90 \times 90$  度生成し、図 5.2 に示すように透視投影画像上の2次元座標と世界座標の3次元座標を対応付けることで、全方位カメラの位置と姿勢を推定した。すべての全方位カメラにおいて、推定位置と実測位置のずれが  $10\text{cm}$  未満であることを確認している。



図 5.2: 全方位画像(10番)と前後の透視投影画像

## 5.2 簡易モデル生成

### 5.2.1 データセットおよびパラメータ設定

図 5.3 に、ワイヤーフレーム生成の入力として用いた、簡易モデルの床面の境界点を示す。これらの点は、2次元マップ上で手動により入力されたものである。



図 5.3: ワイヤーフレームの床面の境界点

床面は三角形メッシュとして分割し、各三角形の最大面積を  $2.0 \text{ m}^2$  に制限した。側面は四角形メッシュとして分割し、天井高は  $2.3 \text{ m}$  とした。サンプリング間隔については、間隔の違いがテクスチャ割り当て後の外観および自己位置推定結果に与える影響を確認するため、 $2 \text{ m}$ 、 $3 \text{ m}$ 、 $4 \text{ m}$  の 3 種類を設定した。

### 5.2.2 ワイヤーフレーム生成結果

図 5.4 に、これらのコーナー点を結ぶ辺に沿って所定のサンプリング間隔で補間点を生成することで構築したワイヤーフレームモデルを示す。サンプリング間隔は左から  $2 \text{ m}$ 、 $3 \text{ m}$ 、 $4 \text{ m}$  である。床面の三角形メッシュは、コーナー点および補間点に基づいて自動的に三角形分割されていることが確認できる。一方、壁面の四角形メッシュについては、サンプリング間隔の変化に伴い生成されるメッシュ形状が変化していることが観察される。

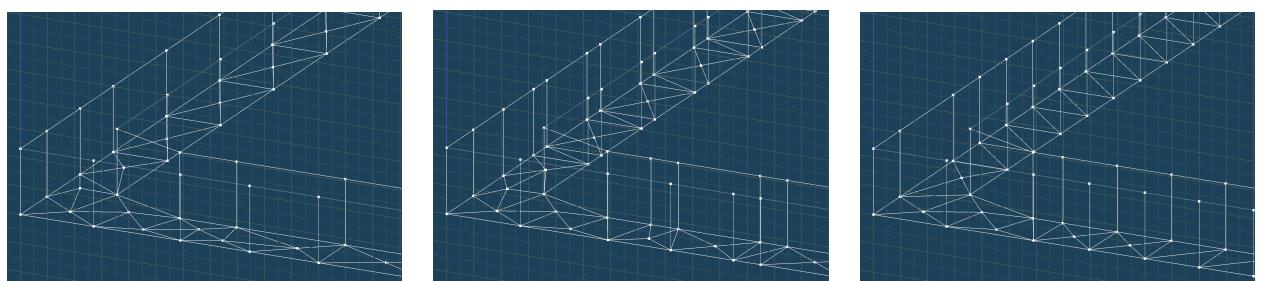


図 5.4: ワイヤーフレームの床面の境界点（サンプリング間隔：左から  $4 \text{ m}$ ,  $3 \text{ m}$ ,  $2 \text{ m}$ ）

### 5.2.3 テクスチャ割り当て結果

図 5.5 に、前節で生成したワイヤーフレームモデルに対して、全方位画像からテクスチャを割り当てた結果を示す。なお、壁面ではなく奥行きを有する面については、誤った特徴点座標が自己位置推定に用いられることを防ぐため、あらかじめテクスチャ割り当ての対象から除外している。

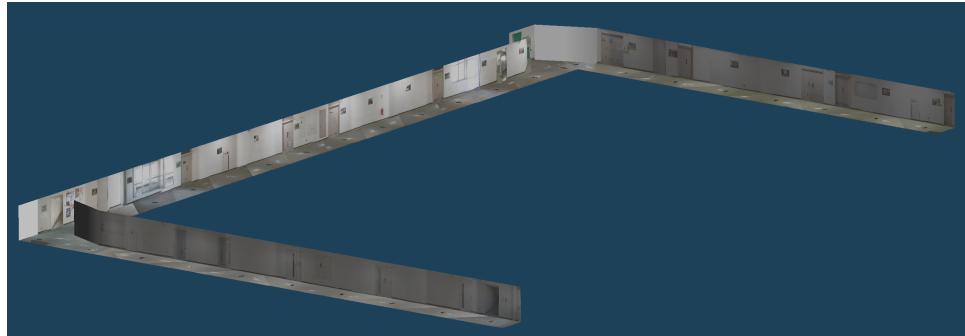


図 5.5: 簡易モデルの全体図

図 5.6 に、サンプリング間隔の異なるワイヤーフレームごとに、拡大表示した簡易モデルを示す。いずれのサンプリング間隔においても、テクスチャは大きな位置ずれを生じることなく、各面に対して適切に割り当てられていることが確認できる。また、ブレンド処理を行った場合には、隣接するテクスチャ間の接続が滑らかになり、視覚的品質の向上に寄与していることがわかる。

サンプリング間隔の違いによる全体的な再現度の差は大きくないものの、サンプリング間隔が 4 m の場合には、テクスチャを大きく拡大して生成する必要があるため、引き延ばしによる劣化が確認される。一方で、3 m の場合には、面ごとに割り当てられるテクスチャの大きさにはばらつきが生じる点が確認された。これらの点から、視覚的品質の観点では、2 m のサンプリング間隔が最も適していると考えられる。



図 5.6: 簡易モデルの拡大図（サンプリング間隔：左から 4 m, 3 m, 2 m）

## 5.3 特徴点マッチング手法の比較実験

### 5.3.1 実験条件

前節で生成した簡易モデルのテクスチャと、実環境で撮影された入力画像との間で特徴点マッチングを行い、その精度を評価した。入力データには、大学構内 C 棟 5 階の廊下環境において、屋内ナビゲーション端末を保持して歩行撮影した動画を用いた。歩行経路と、入力動画を一部切り抜いたものを表 5.7 に示す。動画の解像度は  $1920 \times 1080$  ピクセルであり、これを 1 fps (frame per second) の間隔で静止画として切り出し、評価用画像セットとした。

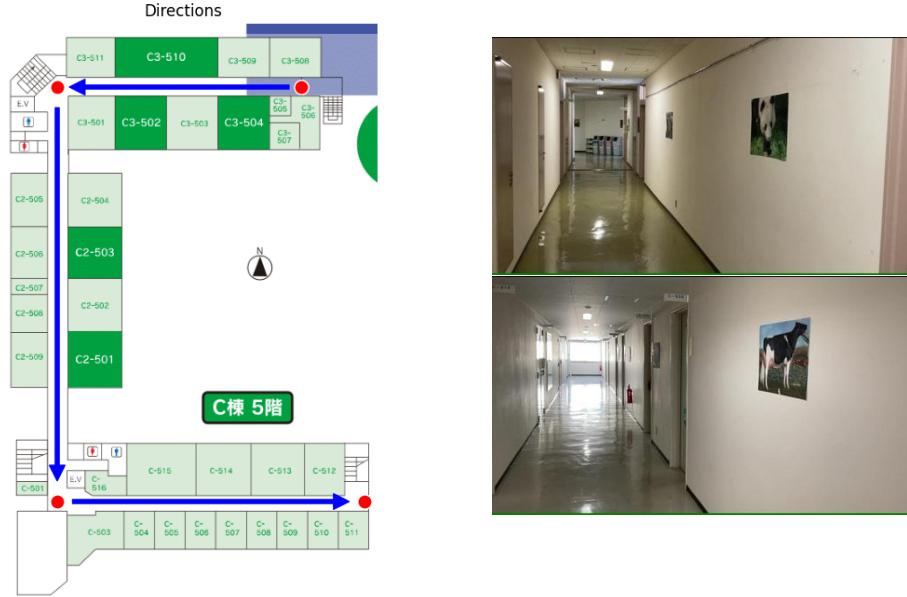


図 5.7: 歩行経路と入力画像

屋内ナビゲーションを目的とした走行環境においては、カメラが壁面や掲示物に対して正対する状況は限定的であり、進行方向に対して壁面を斜め方向から観測する頻度が高いと考えられる。したがって本実験では、視点変化や変形に対する頑健性を確保することを最優先としつつ、ナビゲーション用途としてのリアルタイム性も考慮してパラメータ調整を行った。表 5.2 に SIFT および AKAZE 検出器の主なパラメータを示す。

表 5.2: 従来手法 (SIFT, AKAZE) のパラメータ設定

手法	パラメータ項目	標準値	本実験設定値
SIFT	nOctaveLayers	3	<b>5</b>
	contrastThreshold	0.04	<b>0.03</b>
	edgeThreshold	10	<b>20</b>
AKAZE	threshold	0.001	<b>0.0005</b>
	nOctaveLayers	4	<b>6</b>

一方、深層学習ベースの手法である SuperPoint および SuperGlue についても、同様に計算効率と精度のトレードオフを考慮した設定を用いた。表 5.3 に SuperPoint および SuperGlue のパラメータを示す。

表 5.3: 深層学習手法 (SuperPoint, SuperGlue) のパラメータ設定

モデル	パラメータ項目	標準値	本実験設定値
SuperPoint	max_keypoints	無制限	<b>1024</b>
	keypoint_threshold	0.005	<b>0.001</b>
SuperGlue	weights	—	<b>outdoor</b>
	sinkhorn_iterations	20	<b>5</b>

SuperGlue の重みパラメータ (weights) については、屋内環境での実験であるものの、屋外データで学習された `outdoor` モデルを採用した。一般に屋内では `indoor` モデルが推奨されるが、本研究の対象である「大学構内の廊下」においては、以下の 3 点の理由から屋外モデルの方が適していると判断した。

1. **環境の構造的特徴:** 一般的な屋内学習データは、狭い部屋に置かれた家具や雑貨などの豊富な模様を頼りに学習されている傾向がある。一方、本実験の環境である廊下は、模様が少なく、長い白壁や天井のラインといった直線的な構造が大部分を占めている。この特徴は、複雑な室内よりも、むしろビルの外観や道路といった屋外環境の構造に近い。
2. **視点変化への強さ:** ナビゲーション中のカメラ映像は、壁に近づいたり、斜め方向から撮影したりと、見え方が大きく変化する。屋外モデルは、建物を様々な角度から撮影したデータで学習されているため、こうした大きな視点変化に対して頑健である。細かい模様に頼りがちな屋内モデルと比較して、廊下のような大まかな空間構造を捉える能力に長けていると考えられる。
3. **予備実験による裏付け:** 実際に本環境のデータを用いて比較実験を行ったところ、屋内モデルを使用した場合よりも、屋外モデルを使用した方が安定して多くのマッチング点が得られる傾向が確認された。

マッチング後の誤対応除去については、各手法の純粋な性能を公平に比較するため、処理条件を統一した。ホモグラフィ行列の推定アルゴリズムには、従来の RANSAC と比較してパラメータ依存性が低く、ノイズに対してロバストな MAGSAC++ を採用した [25]。再投影誤差の許容閾値は、入力が高解像度画像であることを考慮し 5.0 pixel に設定した。また、計算コストの増大を防ぎ推定精度を安定化させるため、検出された全マッチング点を用いるのではなく、マッチングスコア上位の 100 点を選抜して幾何学的検証に入力する構成とした。

### 5.3.2 実験結果

図 5.8 に、従来手法 (SIFT, AKAZE) および学習ベースの手法 (SuperPoint+SuperGlue) を用いた特徴点マッチングの定性的な評価結果の一部を示す。図の左列に示すように、生成モデルのテクスチャと入力画像との間の視点差が小さいケースにおいては、3 手法ともに十分なインライア数を確保し、安定したマッチングに成功した。各手法ごとの詳細な傾向は以下の通りである。

- **SIFT:** 視点角度が類似している条件下であれば、撮影距離が離れている場合であってもマッチングが可能であった。しかし、壁面を斜めから見るなど視点変化が大きくなると、マッチングに失敗する事例が多発した。
- **AKAZE:** SIFT と同様の傾向を示したが、SIFT と比較して視点変化に対する耐性がわずかに優れており、より広い角度範囲でマッチングを維持できる傾向が見られた。
- **SuperPoint+SuperGlue:** 従来手法と比較して、視点変化に対する頑健性が著しく向上した。極端にテクスチャが乏しい領域や、ドア等類似構造しか含まない領域といった高難度なケースを除き、ほぼ全てのフレームにおいてマッチングに成功した。

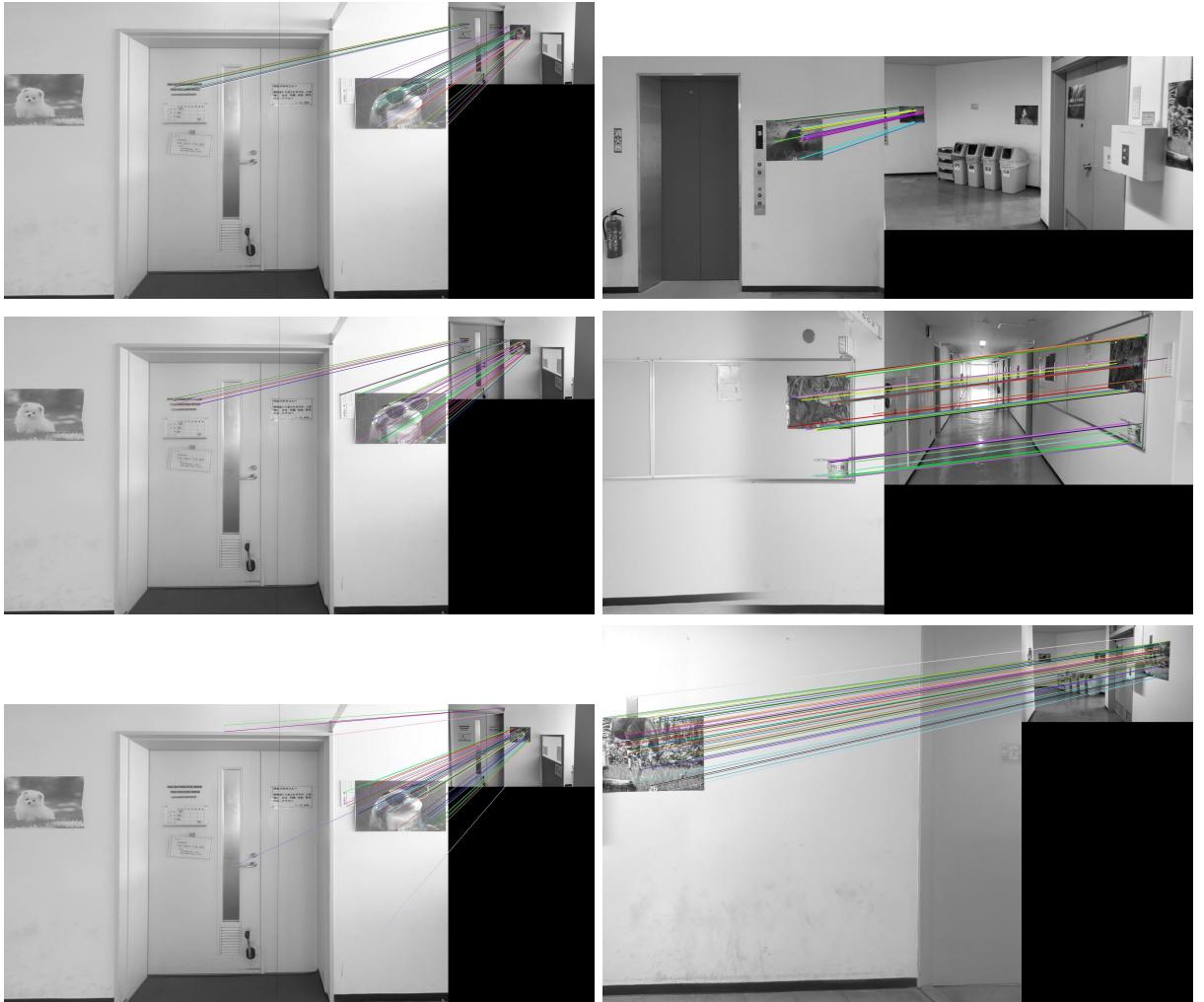


図 5.8: 特徴点マッチング結果の比較

上段から SIFT、AKAZE、SuperPoint+SuperGlue を示し、画像内左はテクスチャ画像、右は入力画像である。

表 5.4 に、各手法におけるマッチング成功枚数および 1 フレームあたりの平均処理時間を見ます。マッチング成功数については、学習ベースの手法 (SuperPoint+SuperGlue) が従来手法 (SIFT, AKAZE) と比較して圧倒的に高い値を示しました。この主な要因は、学習ベース手法が持つ視点変化に対する高い頑健性にある。テクスチャ情報が有効に撮像されているフレームにおいては、角度やスケールの変化に関わらず、ほぼ全てのケースでマッチングに成功していることが確認されました。

一方、平均処理時間に関しては、本実験の設定下では AKAZE が最も高速な結果となった。学習ベースの手法は推論処理を伴うため計算コストが高い傾向にある。しかし、本実験では評価のために全フレームに対して探索を行っているが、実際のナビゲーション運用時においては、一度自己位置が推定された後は近傍のテクスチャのみを探索対象とする処理を導入することで、計算時間は大幅に削減可能であると考えられる。加えて、学習ベースの手法は低解像度画像に対しても高い特徴記述能力を維持する特性があるため、入力解像度をさらに低減させることによる高速化の余地も残されている。

以上の結果より、視点変化が大きく特徴点の抽出が困難な屋内ナビゲーション環境においては、処理速度の課題を運用上の工夫で吸収可能であることを踏まえると、ロバスト性に優れる学

習ベースの手法が最も適しているといえる。したがって、以降の章では特徴点マッチングに学習ベースの手法を採用する。

表 5.4: 各手法におけるマッチング成功枚数と平均処理時間

手法	マッチング成功枚数 (全 156 枚中)	平均処理時間 [ms]
SIFT	12	4628
AKAZE	20	<b>147</b>
SuperPoint+SuperGlue	<b>89</b>	4112

## 5.4 特徴点マッチングに基づく自己位置推定結果の評価

図 5.9 に、前節で述べた学習ベースの手法による特徴点マッチングを用いた自己位置推定の結果を示す。同図では、マッチングに成功したフレームについて、推定された自己位置および正规化された視線方向ベクトルをプロットしている。なお、実数解が得られなかった場合や、天地反転などの幾何学的に不整合な解しか得られなかった場合は、有効な解が算出されなかったものとみなし、結果から除外している。本実験では、図 5.6 に示したサンプリング間隔の異なる 3 種類の簡易モデルを用いて比較を行った。

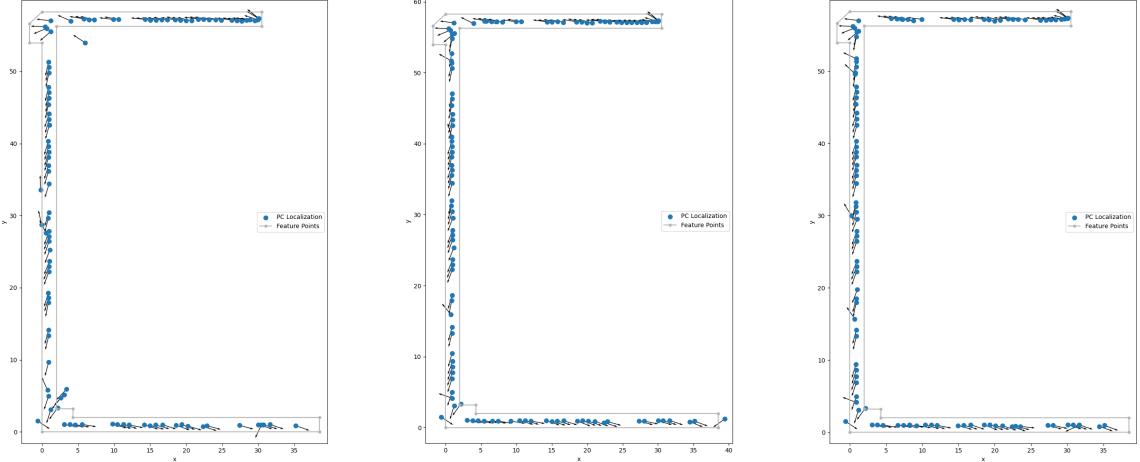


図 5.9: 自己位置推定結果（サンプリング間隔：左から 4m, 3m, 2m）

実験の結果、学習ベース手法の場合いずれのサンプリング間隔においても、多くのフレームで自己位置推定が可能であることが確認された。しかし、サンプリング間隔 4m の場合については、3m および 2m の場合と比較して結果に顕著な差異が見られた。具体的には、特徴点マッチングには成功しているものの、前後のフレーム間で姿勢が不連続に変化したり、推定位置が経路から大きく逸脱したりする誤推定のフレームが多く確認された。全 158 フレームのうち、このような誤推定を除外した有効なフレーム数は、サンプリング間隔 4m で 93 枚、3m で 108 枚、2m で 105 枚であった。3m と 2m はいずれも高い推定成功率を示し、その差はわずかであった。そのため、テクスチャの視覚的品質と自己位置推定の安定性を総合的に評価し、本研究ではサンプリング間隔 2m が最も適していると判断した。

## 5.5 屋内ナビゲーションにおける自己位置推定結果の比較

### 5.5.1 実験条件

本実験では、屋内ナビゲーションにおける自己位置推定手法の差異が、最終的なナビゲーションの成否や精度に与える影響を検証するため、以下の3つの条件を設定し比較を行った。

1. **画像マッチング単独手法:** PCサーバー上での画像特徴点マッチングに基づく自己位置推定のみを用いる条件。
2. **VIO(Visual-Inertial Odometry) 単独手法:** モバイル端末に搭載されたVIOベースの自己位置推定のみを用いる条件。
3. **併用手法:** 両者を併用し、VIOによる追従と画像マッチングによる補正を組み合わせた条件。

歩行経路および経由地は、前節の図5.7に示したものと同一である。ただし、本実験では入力データとして録画済みの動画ではなく、リアルタイムに取得されるカメラ画像を用いた。ナビゲーションの具体的な手順は以下の通りである。

1. **開始処理:** 開始地点にて静止し、初回の自己位置推定を行う。位置が特定され次第、第一の経由地へ向けて歩行を開始する。
2. **経由地判定:** 設定された経由地の半径2m以内に到達した時点で、システムは当該地点への到達と判定し、直ちに次の経由地へのナビゲーションに切り替える。
3. **終了判定:** 最終目的地の半径2m以内に到達した時点でナビゲーションを終了とする。
4. **リカバリ処理:** PCサーバーによる位置推定が10秒間連続して成功しなかった場合、キャッシングが喪失したと判定し、再度初期位置推定処理を実行する。

PCサーバー側で行う2回目以降の自己位置推定においては、計算効率と精度向上のため、モバイル端末から得られる直前の推定位置および視線ベクトルを事前情報として利用する。この際、誤マッチングによる外れ値を排除するため、許容誤差の閾値を位置については1m、姿勢については $10^\circ$ と設定した。また、マッチング対象の絞り込みとして、特徴点マッチングを行う参照画像は、現在の推定端末位置から半径5m以内に存在する画像のみを探索対象とした。

### 5.5.2 実験結果

図13に実験結果を示す。まず、自己位置推定にモバイル端末のVIO(Visual Inertial Odometry)のみを用いた場合、時間の経過に伴い、推定位置が真値から徐々に乖離していくドリフト現象が確認された。この要因として、センサノイズの累積に加え、実験環境の視覚的特徴が影響していると考えられる。本実験環境は白壁などのテクスチャに乏しい平面が多く、VIOの処理において十分な数の特徴点を安定して追跡することが困難な区間が存在した。その結果、視覚情報による自己位置の補正が十分に機能せず、累積誤差が増大する結果となった。

次に、PCサーバーの画像特徴点マッチングのみを用いた場合、マッチングに成功したフレームにおいては高精度な自己位置が得られた。しかし、自己位置を連続的かつリアルタイムに取得することは困難であった。その要因として、環境のテクスチャ不足によるマッチングの不成立に加え、特徴点抽出・照合にかかる計算処理時間、および画像データの送受信に伴う通信タイムラ

グが挙げられる。これらが複合的に影響することで、推定結果の更新が断続的になったり、移動に対して提示が遅れたりする現象が生じ、滑らかな移動が求められるナビゲーション用途においては、即時性と連続性の欠如が課題として確認された。

一方、モバイル端末のVIOと画像特徴点マッチングによる推定を併用した手法では、VIOによる連続的なトラッキングを行いつつ、画像マッチングによる高精度な絶対位置情報を適宜参照することで、VIO単独時に見られた累積誤差を効果的に補正できることが確認された。これにより、特徴の少ない環境下においても、自己位置推定における**精度の安定性と時間的な連続性**の両立が可能となり、本システムが屋内ナビゲーションの実運用において有効であることが示唆された。

# 第6章 まとめ

## 6.1 本研究の成果

本研究では、テクスチャ情報の乏しい屋内環境において、既存の二次元マップと全方位画像のみを用いた簡易三次元モデルによる自己位置推定手法を確立した。本研究により得られた主な成果は以下の通りである。

1. **簡易三次元モデルによる実用的な自己位置推定の実現:** 特別な設備や厳密な形状復元を必要とせず、カメラ視点に基づく最適なサンプリングとブレンド処理を導入することで、視覚的品質と推定精度を両立するモデル生成手法を構築した。さらに、生成したモデルに対して深層学習ベースの特徴点抽出を適用することで、従来手法では対応が困難であった特徴の乏しい壁面や視点変化の激しい状況下でも、実用的な精度で自己位置推定が可能であることを示した。
2. **屋内ナビゲーションの実証:** 提案手法を実装した屋内ナビゲーションシステムを開発し、大学構内の実証実験を通じてその有効性を検証した。実験の結果、モバイル端末の VIO は特徴の乏しい領域でドリフトが生じやすく、提案手法による補正是断続的であるというそれぞれの課題が確認された。これらを統合することで、VIO が移動の連続性を維持しつつ、提案手法が累積誤差を定期的に補正するという相互補完効果が得られ、特徴点が検出されにくい区間においてもスムーズなナビゲーションを実現できることを実証した。

## 6.2 今後の展望

本研究の課題および今後の発展として、主に以下の 2 点が挙げられる。

1. **線特徴の活用による頑健性の向上:** 現状の手法ではテクスチャの点特徴のみに依存しているため、白壁など特徴の乏しい領域では精度が低下しやすい。これに対し、本研究の簡易モデルはワイヤーフレームを骨格としているため、画像から検出した線特徴との親和性が高い。柱やドア枠などの幾何構造を直接活用することで、点特徴の乏しい領域における推定精度の向上を図る。
2. **実利用者によるナビゲーション評価:** システムの実用性を検証するためには、多様な環境下で、多様な属性を持つ被験者を対象とした実証実験が必要となる。目的地到達の正確性といった定量的な評価に加え、ナビゲーションの円滑さや、AR 表示の分かりやすさなど、ユーザビリティの観点からシステムを多角的に評価・改善する必要がある。

## **謝辞**

本研究の機会を与えて下さり、ご指導、ご教示を頂きました菅谷保之准教授に深く感謝致します。

また、論文の添削等さまざまな機会でサポートしてくださった画像情報メディア研究室の皆様方に深く感謝致します。

# 参考文献

- [1] IDC Japan. Digital Twin / IoT / AI 技術調査レポート（概要）. IDC Japan, 2024.
- [2] Global Growth Insights. Digital Twin Technology Market Trends 2024-2033. Global Growth Insights, 2024.
- [3] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [4] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [5] Keisuke Tateno, Federico Tombari, and Nassir Navab. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6243–6252, 2017.
- [6] Y. Liu, J. Zhang, Y. Li, and Z. Zhang. Robust Visual SLAM for Highly Weak-textured Environments. *arXiv:2207.03539*, 2022.
- [7] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *International Journal of Robotics Research*, vol. 35, no. 14, pp. 1309–1332, 2016.
- [8] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast Image-Based Localization using Direct 2D-to-3D Matching. *IEEE International Conference on Computer Vision (ICCV)*, pp. 667–674, 2011.
- [9] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Aggregating Local Descriptors into a Compact Image Representation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3304–3311, 2010.
- [10] Yuan Liu, Wei Dong, Yifan Liu, and Xiaohua Tian. Image-Based Indoor Localization: A Survey. *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2923–2955, 2020.
- [11] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. *Robotics: Science and Systems (RSS)*, pp. 1–9, 2014.
- [12] Paul Biber and Wolfgang Straßer. The Normal Distributions Transform: A New Approach to Laser Scan Matching. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2743–2748, 2003.

- [13] Stephan Kohlbrecher, Johannes Meyer, Thomas Graber, et al. Hector SLAM: Real-Time SLAM with a Single Laser Scanner. IEEE International Conference on Robotics and Automation (ICRA), pp. 3987–3994, 2011.
- [14] Andreas Schindler, Jan-Michael Frahm, and Marc Pollefeys. City-Scale Location Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8, 2007.
- [15] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast Image-Based Localization using Direct 2D-to-3D Matching. IEEE International Conference on Computer Vision (ICCV), pp. 667–674, 2011.
- [16] Jonathan R. Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, in *Applied Computational Geometry: Towards Geometric Engineering*, First ACM Workshop on Applied Computational Geometry, Lecture Notes in Computer Science, Vol. 1148, pp. 203–222, 1996.
- [17] 松下侑聖, 菅谷保之, 直交射影に基づく点特徴と線特徴を用いたハイブリッドなカメラ姿勢推定, 第 27 回画像の認識・理解シンポジウム, 2024 年 8 月.
- [18] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, Vol. 60, No. 2, pp. 91–110, 2004.
- [19] P. F. Alcantarilla, J. Nuevo, A. Bartoli, Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces, British Machine Vision Conference (BMVC), 2013.
- [20] D. DeTone, T. Malisiewicz, and A. Rabinovich, SuperPoint: Self-Supervised Interest Point Detection and Description, arXiv preprint arXiv:1712.07629, 2017.
- [21] P.E.Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, SuperGlue: Learning Feature Matching with Graph Neural Networks, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)\*, 2020.
- [22] 松下侑聖, 直交射影に基づく PnP 問題に対する大域最適解の計算, pp. 4–11, 2024.
- [23] Apple Inc., Managing Session Life Cycle and Tracking Quality, Apple Developer Documentation, [https://developer.apple.com/documentation/arkit/managing\\_session\\_lifecycle\\_and\\_tracking\\_quality](https://developer.apple.com/documentation/arkit/managing_session_lifecycle_and_tracking_quality), (参照日: 2026 年 1 月).
- [24] Apple Inc., Understanding World Tracking, Apple Developer Documentation, [https://developer.apple.com/documentation/arkit/world\\_tracking/understanding\\_world\\_tracking](https://developer.apple.com/documentation/arkit/world_tracking/understanding_world_tracking), (参照日: 2026 年 1 月).
- [25] D. Barath, J. Matas, and J. Noskova, MAGSAC: marginalizing sample consensus, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10197–10205.