特別実験報告書 題目

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

指導教員

菅谷 保之

報告者

田川 幸汰

豊橋技術科学大学 情報・知能工学系

令和 8 年 2 月 9 日 提出

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、テクスチャに乏しい場合が多い。そのため、画像情報に基づいて三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という問題がある。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, 6]、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、必ずしも精密な自己位置推定や幾何学的に高精度な 3 次元モデル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、テクスチャの乏しい屋内環境において、画像ベースの SLAM による高精度な三次元モデル生成や自己位置推定が困難であるという課題に対し、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用途を対象として、実用上十分な精度で安定した自己位置推定を実現可能な手法を確立することである。

本研究では、自己位置推定の際に高精度な三次元形状復元を前提とせず、二次元マップと全方位画像から構築可能な簡易三次元モデルを用いる。具体的には、二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、カメラ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。本手法は、マーカーやセンサー、無線機といった新たな設備を必要とせず、既存の建物環境をそのまま利用可能である。

提案手法は、公共施設やオフィスビルなどの屋内環境におけるナビゲーション用途を想定しており、経路案内や現在位置の把握を主目的とする。このような用途では、必ずしも幾何学的に高精度な三次元モデルや厳密な自己位置推定精度が常に求められるわけではなく、空間の大まかな構造を表現した簡易三次元モデルに基づく自己位置推定であっても、実用上十分な場合が多い。

本研究の貢献は、二次元マップと全方位画像という比較的取得容易な情報のみを用い、特徴点が乏しい屋内環境においても適用可能な、屋内ナビゲーション向け自己位置推定の枠組みを示した点にある。これにより、高精度な三次元形状復元に依存することなく、屋内環境において気軽に導入可能な自己位置推定手法の実現を目指す。

1.3 関連研究

1.3.1 Visual SLAM による自己位置推定と三次元マッピング

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地図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、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, 6]。そのため、画像のみを用いた Visual SLAM によって、屋内環境の高精度な三次元モデルを安定して構築することは、依然として困難な課題である。

(特徴が少ないと不安定になることが示された実験結果や図などが載せれるといい)

本研究で扱う自己位置推定手法は、カメラ画像から特徴点を抽出し、その対応関係に基づいてカメラの位置および姿勢を推定するという点において、Visual SLAM に代表される画像ベースの自己位置推定手法と共通する側面を有する。

しかしながら、Visual SLAM は一般に、連続する画像間で特徴点を追跡することにより、カメラの相対的な移動量を逐次推定し、同時に三次元地図を構築する枠組みである。そのため、推定される自己位置は相対座標系に基づくものであり、外部から絶対座標に関する情報を与えない限り、世界座標系における自己位置は一意に定まらない。

これに対し、本研究の手法では、あらかじめ構築された簡易三次元モデル上のテクスチャに付与された世界座標と、入力画像中の特徴点とを直接対応付けることで、常に世界座標系における絶対的な自己位置を推定する点に特徴がある。すなわち、本研究は、特徴点追跡に基づく相対的自己位置推定ではなく、モデル参照型の自己位置推定を行うものである。

また、本研究では、二次元マップに代表されるように、対象環境の大まかな構造情報が事前に得られていることを前提としている。事前情報を持たずに環境を逐次認識および地図化する Visual SLAM とは前提条件が異なることや、本研究が想定している場面である屋内ナビゲーションにおいて厳密な自己位置推定精度があまり求められないこともあり、両者を単純に推定精度のみで比較することは適切ではないと考える。

以上の前提を踏まえ、本研究では、屋内ナビゲーションの実行を想定し、モバイル端末上で利用可能な Visual SLAM に相当する自己位置推定手法との動作の比較を通じて、提案手法の特性を評価する。ここでの評価は，推定精度の優劣を直接比較することを主目的とするものではなく，壁面や床面が単調で特徴点に乏しい環境においても、自己位置推定が破綻せず安定して動作するかという実用的観点に重点を置く。これにより、屋内ナビゲーション用途において求められる自

己位置推定手法の特性を整理し、実環境において有効に機能するより実用的な自己位置推定手法の在り方について検討する。

1.3.2　画像ベースの自己位置推定手法

3 次元マップ生成を伴わず、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画像ベースの自己位置推定手法も多く提案されている。代表的な手法としては、SfM(Structure from Motion) により構築した 3 次元点群と画像特徴量とのマッチングに基づく手法 [7] や、大規模画像データベースを用いた位置推定手法 [8] 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ことで、SLAM に比べて安定した自己位置推定が可能である一方、高密度かつ高精度な 3 次元モデルの構築が前提となる場合が多く、単調な環境ではモデルの構築が難しいことや、点群を構築するために大規模な事前撮影や計測が必要となる場合が多く、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 次元形状復元を必須とせず、簡易的な三次元環境モデル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、建物を垂直平面などの簡略化した幾何構造として表現し、画像の投影関係を用いて位置を推定する手法が提案されている [13]。また、スパースな 3 次元構造と 2 次元画像特徴の対応付けに基づき、軽量なモデル表現で位置推定を行う研究も報告されている [14]。これらの手法では、高密度な 3 次元モデル生成を前提とせず、モデル構築の容易さと実用性を重視したアプローチが採られている。

これらの研究は、必ずしも高精度な 3 次元形状復元が不要な用途において、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、このような流れを踏まえつつ、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 次元モデルを用いる点に特徴があり、特徴点が乏しい屋内環境における屋内ナビゲーション用途への適用可能性を検討するものである。

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、屋内環境を表現するための基盤として、2 次元マップから簡易な 3 次元ワイヤーフレームモデルを生成する手法について述べる。第 3 章では、生成したワイヤーフレームモデルに対し、全方位画像を用いてテクスチャを割り当てる方法について説明する。第 4 章では、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ついて説明する。第 5 章では、特徴点マッチング結果に基づく自己位置推定手法について説明する。第 6 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 7 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、自己位置推定精度の有効性を検証するために行った実験の結果について述べる。

第2章　簡易モデルを構成するワイヤーフレームの生成

2.1　ワイヤーフレーム生成の方針

本研究におけるワイヤーフレーム生成は、屋内環境の床および壁といった空間構造の骨組みを、比較的少数の頂点と線分で表現することを目的とする。ここで生成されるワイヤーフレームは、後段でテクスチャを付与するための幾何的な基盤であり、高密度な点群や精密メッシュの再現を目的とするものではない。

本研究において生成される簡易モデルは、複雑な形状を高精度に再現することよりも、床や壁、通路といった空間の基本的な構成を簡潔に表現することに重点を置く。これにより、計算コストやデータ管理の負担を抑えつつ，屋内の道案内に必要となる自己位置推定を、必要十分な精度で実現できる点に特徴がある（図 2.1）。

図 2.1: 異なる 3 次元モデル表現の比較（左：点群モデル, 中央：フォトグラメトリモデル, 右：本研究で用いる簡易モデル）

2.1.1　入力情報と前提条件

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
- 2 次元マップと 3 次元座標との 4 点以上の対応関係
- 屋内環境内でテクスチャを取得したカメラ位置

以下ではこの前提のもと、2 次元マップ上で定義された屋内空間の構造情報と、環境の 3 次元座標情報との対応関係に基づき、ワイヤーフレームを半自動的に構成する手法について述べる。ここでいう半自動とは、2 次元マップ上での初期境界指定および対応点の入力を人手で行い、それ以降の幾何構造生成を自動化することを指す。

2.2　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点を 3 次元空間の床面上へ写像するために、2 次元座標間の対応関係に基づくアフィン変換を用いる。床面は水平であると仮定し、床上の 3 次元座標は常に $Z = 0$ に固定されるものとする。この仮定のもとでは、2 次元マップと床面との関係は平行移動・回転・スケーリングによって十分に表現可能であるため、本研究では射影変換ではなくアフィン変換を採用する。

まず、2 次元マップ上の点と、それに対応する床面上の 3 次元座標を、4 点以上与えることができると仮定する。これらの対応点を用いて、2 次元マップ座標系から床面上の 2 次元座標系へのアフィン変換行列を推定する。2 次元マップ上の点を $\mathbf{p}_i = (x_i, y_i)^\top$、対応する床面上の点を $\mathbf{q}_i = (X_i, Y_i)^\top$ とすると、両者の関係は以下のアフィン変換で表される。

$$\begin{pmatrix} X_i \\ Y_i \end{pmatrix} = \mathbf{A} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \tag{2.1}$$

推定されたアフィン変換行列 $\mathbf{A}$ を用いることで、2 次元マップ上の座標を、床面上の 3 次元座標（$Z = 0$）として一意に定めることができる。4 点を超える対応点が与えられる場合には、同一の変換行列を用いて、すべての床境界点を床面上へ写像する。

最後に、得られた床境界の頂点列が閉ループを形成しているかを確認する。閉ループになっていない場合には、始点と終点を接続することで多角形になるように修正する。さらに、頂点列の並び順を判定し、反時計回りとなっている場合には順序を反転させることで、床境界が時計回りとなるように修正する。

2.3　床境界における観測点に基づくサンプリング

床テクスチャ生成のために、床境界上のすべての点を手動で指定することは、作業量が多く現実的ではない。そこで本研究では、床面ポリゴンの境界上において、カメラ配置を考慮しながら床境界を適切な間隔で分割し床境界点を自動的に生成する手法を用いる。

床境界は閉じた多角形として与えられ、隣接する 2 点 $\mathbf{p_0}$ と $\mathbf{p_1}$ により各境界エッジが定義される。まず、各エッジの始点 $\mathbf{p_0}$ は必ず床境界点として採用する。

次に、各カメラ位置 $\mathbf{c}$ を境界エッジ上に正射影する。エッジ方向ベクトル $\mathbf{d}$ および射影係数 t は次式で与えられる。

$$\mathbf{d} = \mathbf{p_1} - \mathbf{p_0} \tag{2.2}$$

$$t = \frac{(\mathbf{c} - \mathbf{p_0})^\top \mathbf{d}}{\mathbf{d}^\top \mathbf{d}} \tag{2.3}$$

射影点 $\mathbf{p}_{\mathrm{proj}}$ は次式で表される。

$$\mathbf{p}_{\mathrm{proj}} = \mathbf{p_0} + \mathbf{t}\mathbf{d} \tag{2.4}$$

このとき、射影係数 t が $0 \leq t \leq 1$ を満たし、かつ

$$\|\mathbf{c} - \mathbf{p}_{\mathrm{proj}}\| \leq \mathbf{d}_{\mathrm{th}} \tag{2.5}$$

である場合に、射影点は当該エッジ上に存在すると判定する。

得られた射影点を用いて、床境界上のサンプリング位置を決定する。カメラから床テクスチャを歪みなく取得するためには、床境界がカメラに対してなるべく正面に位置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が存在する場合には、それらの間隔に応じて床境界点を追加する。具体的には、射影点間の距離が十分に大きい場合には、射影点から一定距離内側に床境界点を追加する。一方、射影点間の距離が小さい場合には、その中点を床境界点として追加する。これにより、カメラ視点に対して適切な位置に床境界点を配置することができる。

（この部分はわかりづらいので、床境界点を生成する方法を図で示す）

最後に、生成された床境界点列全体について隣接点間の距離を確認し、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床境界点を統合する。この処理により、床境界全体が過度に偏ることなく、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する。

2.4 床面および壁面の幾何構造

床面は、前節までで得られた床境界頂点列 $\{\mathbf{P_i}\}$ によって囲まれた単純多角形として定義される。ここで、各床境界頂点は床面上に存在するものとし、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 $\{\mathbf{P_i^{ceil}}\}$ 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 $\mathbf{P_i} = (\mathbf{x_i}, \mathbf{y_i}, \mathbf{0})$ に対し、対応する天井境界頂点は $\mathbf{P_i^{ceil}} = (\mathbf{x_i}, \mathbf{y_i}, \mathbf{H})$ として与えられる。

床面の幾何構造は一般に凹形状を含む複雑な多角形となるため、本研究では、境界形状を保持した制約付き Delaunay 三角形分割を用いて床面を三角形メッシュ化する。この手法は、Shewchuk によって提案された 2 次元品質メッシュ生成手法として広く用いられている [15]。具体的には、床境界頂点を頂点集合とし、隣接する床境界頂点同士を拘束辺として与えた平面分割問題として定式化する。このとき、三角形の最大面積を制約条件として与えることで、過度に大きな三角形が生成されることを防ぎ、安定した床面メッシュを得る。

得られた三角形メッシュに対しては、上方（$+z$ 方向）から見たときに、すべての三角形が時計回りとなるように頂点順序を統一する。これは、後段で行う描画処理や法線計算において、面の向きを一貫させるためである。

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 $\mathbf{P_i^{ceil}}, \mathbf{P_{i+1}^{ceil}}$ を結ぶことで、一枚の壁面を表す四角形を定義する。

このようにして生成された床面および壁面の幾何構造は、本研究における簡易 3 次元モデルのワイヤーフレーム表現を構成し、後段で行うテクスチャ付与および自己位置推定処理の基盤として用いられる。

第3章　簡易モデルへのテクスチャ割り当て

3.1　全方位画像から透視投影画像生成

3.1.1　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。一般的な透視カメラを用いる場合、多方向のテクスチャ情報を得るためには、カメラ姿勢を変更しながら多数の画像を撮影する必要があり、撮影および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、カメラ中心を基準とした全方向の視線情報を一枚の画像として保持しており、幾何学的変換を行うことで、任意の視線方向に対応する透視投影画像を生成できる。この特性により、複数視点から撮影した場合と同等の画像群を、高い効率で取得することが可能となる。(複数方向の透視投影画像を生成する図を示した方がわかりやすい)

3.1.2　透視投影画像変換

本研究では、全方位画像を球面上の輝度分布として扱い、透視投影画像の各画素に対応する視線方向を幾何学的に定義することで、全方位画像から任意視線方向の透視投影画像を生成する。

透視投影画像の幅および高さをそれぞれ W_p, H_p とし、画像上の画素座標を (u, v) とする。透視投影面を $z = 1$ に固定すると、画素 (u, v) に対応する視線ベクトル $\boldsymbol{d}$ は次式で与えられる。

$$\boldsymbol{d} = \begin{pmatrix} (u - W_p/2)\,\Delta x \\ (v - H_p/2)\,\Delta y \\ 1 \end{pmatrix} \tag{3.1}$$

ここで、$\Delta x, \Delta y$ はそれぞれ水平方向および垂直方向の画素間隔を表し、水平画角 Θ、垂直画角 Φ を用いて次式で定義される。

$$\Delta x = \frac{2\tan(\Theta/2)}{W_p}, \quad \Delta y = \frac{2\tan(\Phi/2)}{H_p} \tag{3.2}$$

生成したい透視投影画像の視線方向を表す回転行列を $\boldsymbol{R}$ とすると、回転後の視線ベクトル $\boldsymbol{d}'$ は次式で与えられる。

$$\boldsymbol{d}' = \boldsymbol{R}\boldsymbol{d} \tag{3.3}$$

回転後の視線ベクトル $\boldsymbol{d}' = (d'_x, d'_y, d'_z)^\top$ を球面座標系へ変換し、方位角 Θ および仰角 Φ を次式で求める。

$$\Theta = \arctan 2(d'_x, d'_z) \tag{3.4}$$

$$\Phi = -\arctan\left(\frac{d'_y}{\sqrt{d'^2_x + d'^2_z}}\right) \tag{3.5}$$

全方位画像（正距円筒画像）の幅および高さをそれぞれ W, H とすると、球面座標 (Θ, Φ) に対応する全方位画像上の画素座標 (x, y) は次式で与えられる。

$$x = W\left(\frac{\Theta}{2\pi} + \frac{1}{2}\right) \tag{3.6}$$

$$y = H\left(\frac{1}{2} - \frac{\Phi}{\pi}\right) \tag{3.7}$$

3.2 座標系の定義

3.2.1 カメラ座標系

各座標軸の関係を図 3.1 に示す。

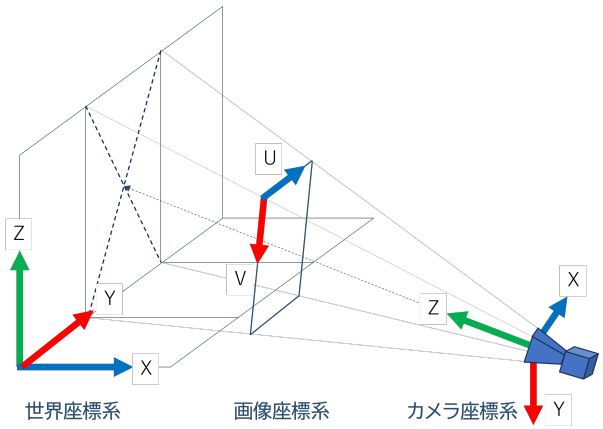

図 3.1: 世界座標系とカメラ座標系、画像座標系の関係

カメラ座標系はカメラの焦点位置を原点とし、光軸方向を z 軸、水平方向右向きを x 軸、鉛直方向下向きを y 軸と定める。モデル床面を X-Y 平面、法線方向を Z 軸とする世界座標系で表された三次元点 $\boldsymbol{p}_w$ は、カメラの回転行列 $\boldsymbol{R}$ と並進ベクトル $\boldsymbol{t}$ を用いて、次式によりカメラ座標系上の点 $\boldsymbol{p}_c$ に変換される。

$$\boldsymbol{p}_c = \boldsymbol{R}\boldsymbol{p}_w + \boldsymbol{t} \tag{3.8}$$

3.2.2 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、鉛直方向を v 軸と定める。カメラ座標系上の 3 次元点 $\boldsymbol{p}_c = (x_c, y_c, z_c)^\top$ は、カメラ内部パラメータ行列 $\boldsymbol{K}$ を用いて、次式により画像座標系へ射影される。

$$\boldsymbol{p}_s = \boldsymbol{K}\boldsymbol{p}_c \tag{3.9}$$

ここで $\boldsymbol{p}_s = (u_s, v_s, w_s)^\top$ とすると、透視投影画像上の画素座標 (u, v) は正規化により次式で得られる。

$$u = \frac{u_s}{w_s}, \quad v = \frac{v_s}{w_s} \tag{3.10}$$

3.2.3　カメラ内部パラメータの設定

理想的な透視投影画像を仮定し、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 $\boldsymbol{K}$ は次式で表される。

$$\boldsymbol{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

ここで f_x、f_y はピクセル単位の焦点距離を表す。全方位画像の幅 W_e と出力透視投影画像の幅 W_p、水平視野角 θ および垂直視野角 ϕ から $f_x = \frac{W_p}{2\tan(\theta/2)}$、$f_y = \frac{H_p}{2\tan(\phi/2)}$ とあらわされる。また c_x、c_y は光軸中心を表し、透視投影画像の中心に設定する。

3.3　複数の透視投影カメラによる全方位カメラ位置姿勢推定

点特徴および線特徴を使ってカメラ位置を推定するため、直交射影に基づく点特徴と線特徴を用いたカメラ姿勢推定 [**?**] を行う。これは、直交射影に基づく点の共線性誤差と線の共面性誤差に対して同一の定式化を行うことで点特徴と線特徴を同時に扱い、カメラ姿勢を推定する手法である。

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は以下の通りである。

- $\boldsymbol{p}_a$：世界座標系で表現された空間点の座標。透視投影画像に映らない点は除外する。
- $\boldsymbol{v}_a$：$\boldsymbol{p}_a$ に対応する画像上の特徴点座標。
- $\boldsymbol{d}_a$：世界座標系で表現された直線 L_a の方向ベクトル。
- $\boldsymbol{r}_a$：世界座標系で表現された直線 L_a 上の点の座標。
- $\boldsymbol{n}_a$：L_a に対応する画像上の直線の法線ベクトル。

$\boldsymbol{v}_a$ は透視投影画像上で $\boldsymbol{p}_a$ に対応する画素を選択することで求める。また、画像座標 (v_x, v_y) は光軸中心を原点とした座標に平行移動する：

$$v_x = x - \frac{W}{2}, \quad v_y = y - \frac{H}{2} \tag{3.12}$$

さらに焦点距離 f を追加し、$\boldsymbol{v}_a = (v_x, v_y, f)$ とする。

$\boldsymbol{d}_a$ は 3 次元モデルの頂点座標を用いて、以下の式で求める。

$$\boldsymbol{d}_i = \frac{\boldsymbol{p}_{i+1} - \boldsymbol{p}_i}{|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i|} \tag{3.13}$$

$\boldsymbol{n}_a$ は画像上の 2 点 $(\boldsymbol{v}_i, \boldsymbol{v}_{i+1})$ から外積を計算して求める：

$$\boldsymbol{n}_i = \frac{\boldsymbol{v}_i \times \boldsymbol{v}_{i+1}}{|\boldsymbol{v}_i \times \boldsymbol{v}_{i+1}|} \tag{3.14}$$

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自己位置推定を反復的に行うことで、点特徴および線特徴に基づく目的関数を最小化する。最終的に得られるカメラの回転行列 $\boldsymbol{R}$ と並進ベクトル $\boldsymbol{t}$ は、世界座標系の三次元点をカメラ座標系に変換するための変換行列である。

3.3.1 座標系変換と投影

メッシュ頂点の世界座標 $\boldsymbol{v}_{\text{world}}^{(i)} \in \mathbb{R}^3$（$i = 1, 2, 3$）を，カメラの回転行列 $\boldsymbol{R}$ および並進ベクトル $\boldsymbol{t}$ を用いて，次式によりカメラ座標系へ変換する。

$$\boldsymbol{v}_{\text{cam}}^{(i)} = \boldsymbol{R}\boldsymbol{v}_{\text{world}}^{(i)} + \boldsymbol{t} \tag{3.15}$$

変換後の頂点座標を用いて，メッシュの重心方向ベクトル $\boldsymbol{c}_{\text{cam}}$ を次式で求める。

$$\boldsymbol{c}_{\text{cam}} = \frac{1}{3}\sum_{i=1}^{3} \boldsymbol{v}_{\text{cam}}^{(i)} \tag{3.16}$$

本研究では，各メッシュの重心方向にカメラ視線を向けることで，当該メッシュが透視投影画像の中心付近に投影されるよう制御する。これにより，射影歪みを抑えた状態でテクスチャを取得することが可能となる。

この目的のため，メッシュ重心方向 $\boldsymbol{c}_{\text{cam}}$ を視線方向とする射影用回転行列 $\boldsymbol{R}_{\text{proj}}$ を構成する。具体的には，重心方向を前方軸とし，上方向ベクトルとの外積により左右方向を定めることで，右手系の直交基底を形成する。得られた直交基底を列ベクトルとして並べることで，射影用回転行列を定義する。

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。

$$\boldsymbol{v}_{\text{proj}}^{(i)} = \boldsymbol{R}_{\text{proj}}^{\top}\boldsymbol{v}_{\text{cam}}^{(i)} \tag{3.17}$$

回転後の 3 次元点をカメラ内部パラメータ行列 $\boldsymbol{K}$ を用いて画像座標系へ投影する。得られる画素座標が画像範囲内に収まらない場合には，画像中心からの最大距離に基づいて，透視投影画像のサイズおよび投影スケールを調整する。この処理により，メッシュが画像外に切り落とされることを防ぎ，単一の透視投影画像からメッシュ全体のテクスチャを確実に取得できる。

3.3.2 テクスチャ候補の評価

距離が遠いテクスチャや、正面方向から撮影していないテクスチャはテクスチャの視覚的品質劣化の原因となるため、テクスチャ割り当ての条件として、以下の物理量を計算する。

- メッシュ重心までの距離

$$d = \|\boldsymbol{c}_{\text{cam}}\| \tag{3.18}$$

- メッシュ重心方向ベクトル $\boldsymbol{c}_{\text{cam}}$ とメッシュ法線ベクトル $\boldsymbol{n}$ のなす角

$$\begin{aligned} \boldsymbol{n} &= \frac{(\boldsymbol{v}_{\text{cam}}^{(2)} - \boldsymbol{v}_{\text{cam}}^{(1)}) \times (\boldsymbol{v}_{\text{cam}}^{(3)} - \boldsymbol{v}_{\text{cam}}^{(1)})}{\|(\boldsymbol{v}_{\text{cam}}^{(2)} - \boldsymbol{v}_{\text{cam}}^{(1)}) \times (\boldsymbol{v}_{\text{cam}}^{(3)} - \boldsymbol{v}_{\text{cam}}^{(1)})\|} \\ \theta &= \arccos\left(\frac{\boldsymbol{c}_{\text{cam}} \cdot \boldsymbol{n}}{\|\boldsymbol{c}_{\text{cam}}\|\,\|\boldsymbol{n}\|}\right) \end{aligned} \tag{3.19}$$

これらの評価量を用いて、各メッシュに対して距離が近く、かつ正面に近い視点から取得された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。

3.4　テクスチャの射影変換

透視投影画像から得られるメッシュ対応領域は、透視歪みを含む任意形状の多角形として画像座標系上に表現される。この領域は、撮影条件や視点方向に依存して形状が変化するため、そのままでは、後段でテクスチャ画像上の画素座標から対応する三次元位置を一意に復元することが困難となる。

そこで本研究では、透視投影画像上で得られたメッシュ領域を、元のメッシュ形状に対応した正規化された画像座標系へ射影変換する。これにより、テクスチャ画像上の画素座標とモデルが保持する三次元形状との対応関係を維持する。

三角形メッシュの場合は、三頂点の対応関係に基づいてアフィン変換を適用し、元の三角形形状を保ったテクスチャを生成する。一方、四角形メッシュの場合は、四頂点の対応関係に基づく射影変換を適用し、透視歪みを補正した長方形テクスチャを生成する。

(実際に多角形変形前後の画像を載せたい)

さらに、変換後の画像に対してマスク処理を行い、メッシュ領域のみを抽出することで、元のメッシュ形状と整合したテクスチャ画像を得る。

3.5　テクスチャの視覚的品質の改善

複数の視点から取得されたテクスチャを同一メッシュに割り当てる場合、視点差や撮影条件の違いにより、テクスチャ境界に不連続が生じ、四角的品質が悪化してしまう。本研究では、この不連続を低減するため、隣接する 2 つのテクスチャに対してブレンド処理を行う。

各テクスチャには視線方向に基づく左右情報が付与されており、これを用いて左右関係を判定する。左側テクスチャを I_{L}、右側テクスチャを I_{R} とする。

ブレンドは画像全体ではなく、画像幅 W の中央付近に設定した、x 方向の狭い領域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。

$$I(x, y) = w_{\mathrm{L}}(x)\, I_{\mathrm{L}}(x, y) + w_{\mathrm{R}}(x)\, I_{\mathrm{R}}(x, y), \tag{3.20}$$

ただし、

$$w_{\mathrm{L}}(x) + w_{\mathrm{R}}(x) = 1 \tag{3.21}$$

を満たす。

ブレンド領域内では，x 方向に沿って連続的に変化する重みを用いるため、シグモイド関数に基づき右側テクスチャの重み $w_{\mathrm{R}}(x)$ を次式で定義する。

$$w_{\mathrm{R}}(x) = \frac{1}{1 + \exp(-k\,\tilde{x})}, \tag{3.22}$$

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 は重みの遷移の急峻さを制御するパラメータである。左側テクスチャの重みは

$$w_{\mathrm{L}}(x) = 1 - w_{\mathrm{R}}(x) \tag{3.23}$$

として与える。

ブレンド領域外では、一方のテクスチャのみを用いることで、テクスチャ全体の解像感および幾何的整合性を維持する。このように、境界付近に限定した方向性を持つブレンドを行うことで、テクスチャ境界に生じる輝度不連続を効果的に低減できる。

第4章　入力画像とテクスチャの特徴点マッチング

4.1　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、画像間で対応する特徴点を検出し、それらの対応関係を用いて、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構築することで、カメラの位置および姿勢を推定する。なお、本研究で使用するテクスチャは側面テクスチャに限定する。床面は同一テクスチャが連続して用いられることが多く、特徴点の識別が困難であるため、対応付けの対象から除外する。

この対応付けを実現するため、本章では特徴点検出および特徴点マッチングの処理を行う。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、撮影条件や環境の変化に対する適応性には限界がある。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。その一方で、従来手法と比較して計算コストが大きいという特徴がある。

4.2　従来手法による特徴点マッチング

4.2.1　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変な特徴点を得る手法である [17]。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する手法であり、Fast Explicit Diffusion を用いることで高速な処理を可能としている [18]。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。本研究では、入力画像および簡易モデルに対応するテクスチャ画像をグレースケール化した後、それぞれに対して特徴点検出および記述子計算を行う。また、データベースとなるテクスチャ画像に対しては、特徴点を事前に計算しておくことで、マッチング時の計算負荷を低減している。

4.2.2　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 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。

4.2.3　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、Lowe の比率テストを適用し、第一近傍と第二近傍の距離比が一定以下となる対応点のみを採用することで、曖昧な対応を除去する。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。

4.3　学習ベース手法による特徴点マッチング

4.3.1　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 は畳み込みニューラルネットワークを用いて、画像中の特徴点位置と対応する記述子を同時に推定する手法である [19]。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，特徴点検出および記述子生成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。

(superpoint に関する図)

本研究では、入力画像および簡易モデルに対応するテクスチャ画像をグレースケール化し、SuperPoint に入力することで特徴点検出および記述子計算を行う。画像サイズが大きい場合には画像を複数のタイルに分割し、それぞれに対して特徴点検出を行った後、検出結果を統合することで、画像全体の特徴点を抽出している。

4.3.2　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 は特徴点集合をグラフ構造として扱い、自己注意機構を用いて特徴点間の対応関係を推定する学習ベースのマッチング手法である [20]。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。

(superglue に関する図)

4.3.3　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 ratio test 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、曖昧な対応を除去する。そのあとは同様に、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。

第5章　特徴点マッチングに基づく自己位置推定

5.1　テクスチャ画像座標から世界座標への変換

第4章で取得した簡易モデルのテクスチャ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、これらの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。

5.1.1　2次元座標から3次元座標への変換

。それぞれの世界座標系における3次元座標を左上から順に時計回りに P_0, P_1, P_2, P_3 とすると，平面基底ベクトルは次式で表される。

$$\boldsymbol{B}_1 = P_1 - P_0, \quad \boldsymbol{B}_2 = P_3 - P_0 \tag{5.1}$$

同様に、対応するテクスチャのUV座標を $p_0, p_1, p2, p_3$ とし、UV空間における幅および高さを次式で定義する。

$$w = p_1^{(u)} - p_0^{(u)}, \quad h = p_3^{(v)} - p_0^{(v)} \tag{5.2}$$

入力画像上で検出された2次元特徴点をピクセル座標 (x, y) とし，画像サイズを (W, H) とすると、特徴点は次式によりUV空間上の正規化座標 $(u, v) \in [0, 1] \times [0, 1]$ に変換される。

$$u = \frac{x/W - p_0^{(u)}}{w}, \quad v = \frac{y/H - p_0^{(v)}}{h} \tag{5.3}$$

得られた正規化係数 u, v を、世界座標系における平面基底ベクトルの線型結合として用いることで、対応する3次元座標 $\boldsymbol{P}$ を次式により計算する。

$$\boldsymbol{P} = P_0 + u\boldsymbol{B}_1 + v\boldsymbol{B}_2 \tag{5.4}$$

これにより、テクスチャ画像上の2次元特徴点を、対応する四角形メッシュ平面上の世界座標系3次元特徴点へ変換することができる。

5.2　自己位置推定

前節で取得した入力画像上の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PnPL問題に対する大域最適解の計算手法を用いる[21]。一般的な反復的最適化手法では、初期値に依存して局所最適解に陥

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、非反復的に大域最適解を求めることが可能であり、計算時間の削減と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。松下らの手法では、画像上の 2 次元特徴点に焦点距離を加えたベクトル (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) を入力として用いる。このとき、虚数解を除いたすべての解候補に対して、目的関数値 J と対応する回転行列 $\boldsymbol{R}$ および並進ベクトル $\boldsymbol{t}$ が出力される。

得られた複数の解候補の中から、本研究では以下の方針に基づいて最終的な自己位置推定結果を選択する。具体的には、各解候補について姿勢誤差を評価し、その値が最も小さい解を最終結果として採用する。これは、姿勢誤差が大きい推定結果は、画像中の対応点が局所的に偏っているなど、特徴点マッチングに不備がある可能性が高いと考えられるためである。

1. 大域最適解法により得られた解候補のうち，天地が反転している解を除外する。本研究では，カメラ座標系の Y 軸が世界座標系の Z 軸の負方向を向くという幾何学的制約に基づき，天地反転の有無を判定する。

2. 残った各解候補について，回転行列 $\boldsymbol{R}$ および並進ベクトル $\boldsymbol{t}$ から，カメラ中心位置 $\boldsymbol{C}$ を

$$\boldsymbol{C} = -\boldsymbol{R}^\top t \tag{5.5}$$

   により算出する。

3. 現在位置 $\boldsymbol{C}_{\mathrm{gt}}$ が既知である場合，推定されたカメラ中心位置 $\boldsymbol{C}$ 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。

$$e_{\mathrm{pos}} = \|\boldsymbol{C} - \boldsymbol{C}_{\mathrm{gt}}\|_2 \tag{5.6}$$

4. 現在の視線方向 $\boldsymbol{f}_{\mathrm{gt}}$ が既知である場合，世界座標系における視線ベクトル $\boldsymbol{f}_{\mathrm{world}}$ を次式で定義する。

$$\boldsymbol{f}_{\mathrm{world}} = \boldsymbol{R}^\top (0, 0, 1)^\top \tag{5.7}$$

5. 推定された視線方向 $\boldsymbol{f}_{\mathrm{world}}$ と基準となるカメラ視線方向 $\boldsymbol{f}_{\mathrm{gt}}$ とのなす角を，姿勢誤差 e_{rot} として次式で定義する。

$$e_{\mathrm{rot}} = \cos^{-1}\left(\frac{\boldsymbol{f}_{\mathrm{world}} \cdot \boldsymbol{f}_{\mathrm{gt}}}{\|\boldsymbol{f}_{\mathrm{world}}\|\|\boldsymbol{f}_{\mathrm{gt}}\|}\right) \tag{5.8}$$

6. 位置誤差 e_{pos} および姿勢誤差 e_{rot} が，それぞれ所定の閾値以下となる解のみを有効な解候補として残す。

7. 有効な解候補の中から，姿勢誤差 e_{rot} が最小となる解を，最終的な自己位置推定結果として採用する。

第6章　自己位置推定結果を用いた屋内ナビゲーション

6.1　屋内ナビゲーションの方針

6.1.1　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起因する遅延により、推定位置が端末の実際の位置とずれる可能性がある。さらに、屋内環境では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用いる可能性が高く、ネットワーク通信による処理遅延が生じる場合には自己位置推定のリアルタイム性に大きな影響が出る可能性がある。

6.1.2　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。推定された自己位置情報は、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、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と考えられる。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基準情報として用いつつ、状況に応じた補完手段を導入する方針とする。屋内環境における自己位置推定の安定性を向上させるため、本研究ではモバイル端末に搭載された自己位置推定機能を併用する。具体的には、ARKit による自己位置推定結果を併用し、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。

6.1.3　ARKit による自己位置推定

ARKit は、Apple 社が提供するモバイル端末上で動作する自己位置推定技術であり、Visual-Inertial Odometry（VIO） と呼ばれる技術を用いる。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 [22]。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、リアルタイム性が確保されやすい一方、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 [23]。

ARKit では、ワールド座標系が内部的に定義されており、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、重力と反対方向を Y 軸、カメラの向いている方向を Z 軸とし、Y 軸と Z 軸の外積が X 軸となる。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として表現される。

本研究では、この 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 6.1 に示す。

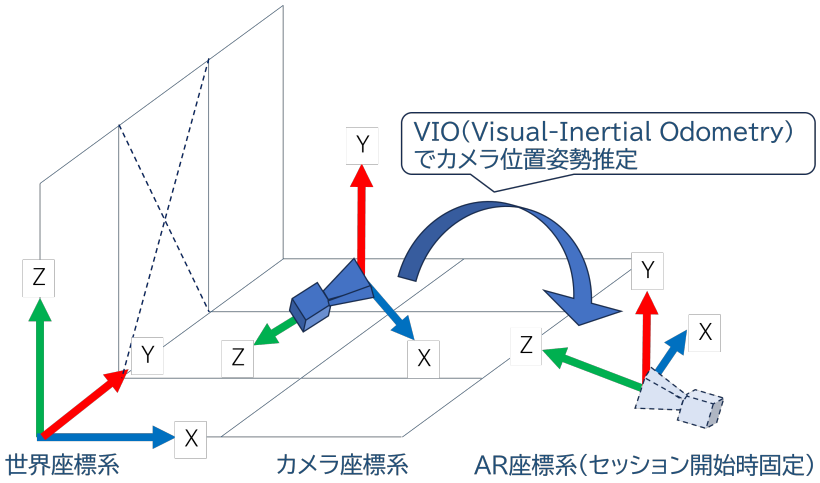

図 6.1: 世界座標系とカメラ座標系、AR 座標系の関係

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 のカメラ座標系では、X 軸は右方向、Y 軸は上方向、Z 軸は手前方向が正となる。図 6.2 にその定義を示す。

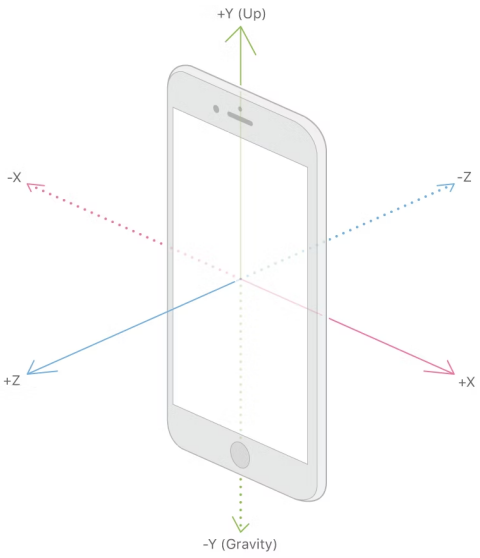

図 6.2: ARKit カメラ座標系の定義

自己位置推定結果を補完するためには、ここで示した座標系の違いを考慮した上で、各座標系間の相互変換が必要となる。

6.2　自己位置推定手法の使い分けと比較方針

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検討するため、性質の異なる 3 つの自己位置推定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づける。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。

以上の三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より実用的な自己位置推定手法を理論的に検討する。

6.3　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 6.3 に示す。

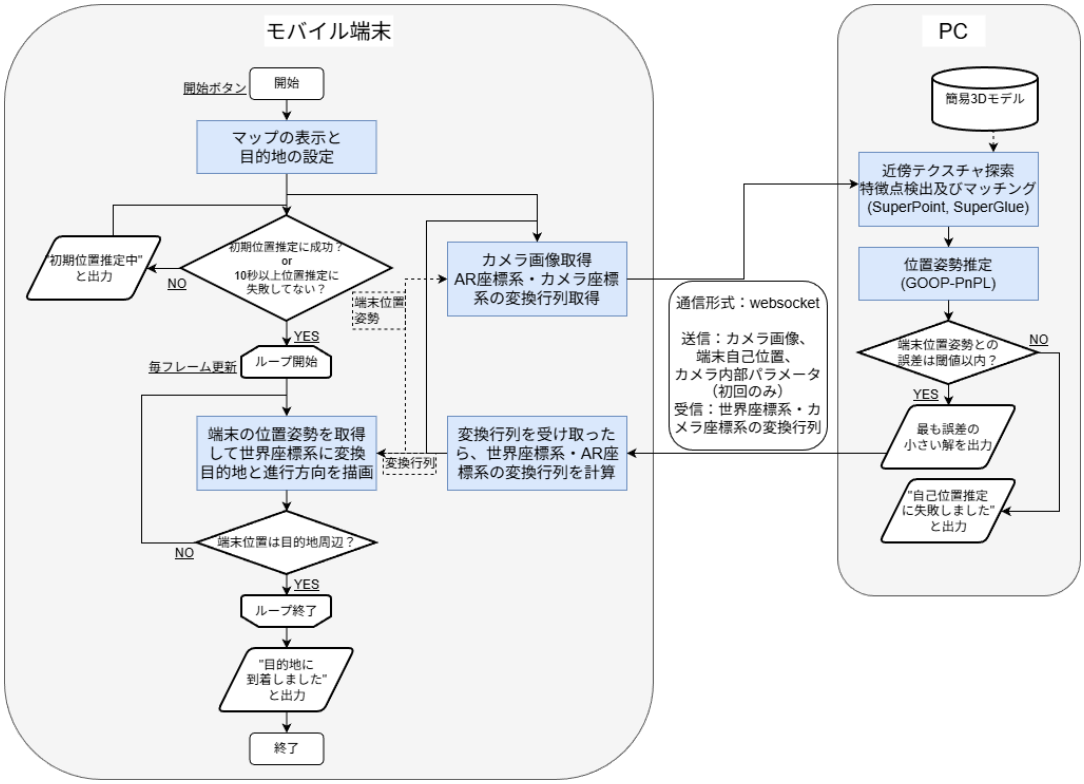

図 6.3: 世界座標系とカメラ座標系、画像座標系の関係

6.3.1 目的地設定

マップ画像上で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。

6.3.2 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 回目以降の推定では、ARKit により更新されたカメラの位置姿勢情報も用い、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。

6.3.3 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 $\mathbf{R}_{\text{world}\to\text{cam}}$ および並進ベクトル $\mathbf{t}_{\text{world}\to\text{cam}}$ と、撮影時における AR 座標系とカメラ座標系の変換行列 $\mathbf{T}_{\text{cam}\to\text{AR}}$ 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、目的地の 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。

6.3.4 変換行列の計算

ARKit 座標系 → 世界座標系の変換行列（$\mathbf{R}_{\text{ar}\to\text{world}}, \mathbf{t}_{\text{ar}\to\text{world}}$）の計算方法を以下に示す。

ARKit では、各フレームのデバイスの姿勢は、カメラ変換行列 (ARFrame.camera.transform) で表される。この変換行列はカメラ座標系から、AR 座標系の変換を表す 4×4 の同次変換行列であり，次のように書ける：

$$\mathbf{T}_{\text{cam}\to\text{AR}} = \begin{bmatrix} \mathbf{R}_{\text{cam}\to\text{AR}} & \mathbf{t}_{\text{cam}\to\text{AR}} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \tag{6.1}$$

ここで，$\mathbf{R}_{\text{cam}\to\text{AR}}$ はカメラ座標系から AR 座標系への回転行列、$\mathbf{t}_{\text{cam}\to\text{AR}}$ は AR 座標系におけるカメラ原点位置を表す並進ベクトルである。

ここで、カメラ座標系 →ARKit 座標系の変換は以下の式 6.2 で表される。

$$\mathbf{X}_{\text{ar}} = \mathbf{R}_{\text{cam1}\to\text{ar}}\mathbf{X}_{\text{cam1}} + \mathbf{t}_{\text{cam1}\to\text{ar}} \tag{6.2}$$

また、世界座標系 → カメラ座標系の変換は以下の式 6.3 で表される。

$$\mathbf{X}_{\text{cam2}} = \mathbf{R}_{\text{world}\to\text{cam2}}\mathbf{X}_{\text{world}} + \mathbf{t}_{\text{world}\to\text{cam2}} \tag{6.3}$$

さらに、ARKit のカメラ座標系 $\mathbf{X}_{\text{cam1}}$ と世界座標系のカメラ座標系 $\mathbf{X}_{\text{cam2}}$ の座標系の違いは以下の式 6.5 で表される

$$\mathbf{X}_{\text{cam2}} = \mathbf{S}\,\mathbf{X}_{\text{cam1}} \tag{6.4}$$

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{6.5}$$

式 6.2、式 6.3、式 6.5 から、世界座標系 →ARKit 座標系の変換は以下の式 6.10 で整理される。

$$\mathbf{X}_{\text{ar}} = \mathbf{R}_{\text{cam}\to\text{ar}}(\mathbf{S}(\mathbf{R}_{\text{world}\to\text{cam2}}\mathbf{X}_{\text{world}} + \mathbf{t}_{\text{world}\to\text{cam2}})) + \mathbf{t}_{\text{cam}\to\text{ar}} \tag{6.6}$$

$$\mathbf{X}_{\text{ar}} = (\mathbf{R}_{\text{cam}\to\text{ar}}\mathbf{S}\mathbf{R}_{\text{world}\to\text{cam2}})\mathbf{X}_{\text{world}} + \mathbf{R}_{\text{cam}\to\text{ar}}\mathbf{S}\mathbf{t}_{\text{world}\to\text{cam2}} + \mathbf{t}_{\text{cam}\to\text{ar}} \tag{6.7}$$

$$\mathbf{X}_{\text{ar}} = \mathbf{R}_{\text{world}\to\text{ar}}\mathbf{X}_{\text{world}} + \mathbf{t}_{\text{world}\to\text{ar}} \tag{6.8}$$

$$\mathbf{R}_{\text{world}\to\text{ar}} = \mathbf{R}_{\text{cam}\to\text{ar}}\mathbf{S}\mathbf{R}_{\text{world}\to\text{cam2}} \tag{6.9}$$

$$\mathbf{t}_{\text{world}\to\text{ar}} = \mathbf{R}_{\text{cam}\to\text{ar}}\mathbf{S}\mathbf{t}_{\text{world}\to\text{cam2}} + \mathbf{t}_{\text{cam}\to\text{ar}} \tag{6.10}$$

式 6.10 より、世界座標系と ARKit 座標系の座標変換は式 6.16 で表される。

$$\mathbf{X}_{\text{world}} = \mathbf{R}^{\top}_{\text{world}\to\text{ar}}\mathbf{X}_{\text{ar}} - \mathbf{R}^{\top}_{\text{world}\to\text{ar}}\mathbf{t}_{\text{world}\to\text{ar}} \tag{6.11}$$

$$\mathbf{X}_{\text{world}} = \mathbf{R}_{\text{ar}\to\text{world}}\mathbf{X}_{\text{ar}} + \mathbf{t}_{\text{ar}\to\text{world}} \tag{6.12}$$

$$\mathbf{R}_{\text{ar}\to\text{world}} = \mathbf{R}^{\top}_{\text{world}\to\text{ar}} \tag{6.13}$$

$$= (\mathbf{R}_{\text{cam}\to\text{ar}}\mathbf{S}\mathbf{R}_{\text{world}\to\text{cam}})^{\top} \tag{6.14}$$

$$\mathbf{t}_{\text{ar}\to\text{world}} = -\mathbf{R}^{\top}_{\text{world}\to\text{ar}}\mathbf{t}_{\text{world}\to\text{ar}} \tag{6.15}$$

$$= -\mathbf{R}_{\text{ar}\to\text{world}}(\mathbf{R}_{\text{cam}\to\text{ar}}\mathbf{S}\mathbf{t}_{\text{world}\to\text{cam}} + \mathbf{t}_{\text{cam}\to\text{ar}}) \tag{6.16}$$

6.3.5 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、世界座標系から AR 座標系への変換を行うことで、目的地の AR 空間上の位置を算出する。

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、端末の移動に伴っても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 AR 空間上に表示する。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にアンカーを生成することで表示内容を更新する。

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7章　実験

本節では、簡易モデルを用いた自己位置推定実験に用いた機材、パラメータ、入力データ、および実験結果について説明する。

7.1　実験準備

実験のために計測した点や撮影したカメラ位置、実行端末などについて示す。

7.2　全方位カメラパラメータ推定結果

外部パラメータの推定結果を実測値と比較したところ、カメラ位置の誤差はおおむね 10cm 以下であることを確認した。

7.3　簡易モデル生成結果

壁面や扉付近など特徴量が乏しい領域には、ポスターを貼付して特徴量を追加したモデルを作成し、その効果を検証した。

7.4　特徴点マッチング手法の比較評価

学習ベースの手法は、一般的な手法と比較して特徴が乏しい画像でも比較的安定してマッチングできた。一方、入力画像が複数のテクスチャ画像と類似した特徴を含む場合、どちらの手法でも正しいマッチングが困難であった。

7.5　特徴点マッチングに基づく自己位置推定結果の評価

動画の各フレームを抽出し、特徴点マッチングと自己位置推定を実施した。初回は全テクスチャとマッチングを行い、2 回目以降は直前の自己位置から近傍のテクスチャに限定してマッチングを行った。学習ベース手法では、特徴点マッチングが行われたフレーム数が一般手法の約 5 倍となり、ほぼすべてのフレームで自己位置を推定できた。1 フレームあたりの計算時間は約 0.3 秒であり、特徴点マッチングに時間の大部分がかかっていることが確認できた。

7.6　屋内ナビゲーションにおける自己位置推定結果の比較

本研究では，屋内ナビゲーションにおける自己位置推定手法の違いがナビゲーション結果に与える影響を確認するため，モバイル端末上の AR による自己位置推定のみを用いる場合、PC 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場合，および両者を併用する場合の三つの条件について比較を行った。

まず、モバイル端末の自己位置推定結果 (ARKit) のみを用いた場合，時間の経過とともに推定位置が実際の位置から徐々にずれていくドリフトが発生していることが確認された。この結果は，屋内環境においてセンサ誤差や特徴点追跡の不安定さが累積することで，自己位置推定誤差が増大する可能性を示している。

次に、PC 上での自己位置推定結果のみを用いた場合，推定可能なフレームにおいてはおおむね正確な自己位置が得られているものの，常に自己位置を更新できるわけではなく，特徴点マッチングが成立しない場面では自己位置推定が行われない状況が確認された。そのため，自己位置を一定間隔で連続的に取得することが困難であり，ナビゲーション用途においては不連続な挙動となる傾向が見られた。

一方で、AR と PC による自己位置推定結果を併用した場合には，PC による自己位置推定結果を適宜参照することで，AR のみの場合に見られたドリフトの影響を抑制しつつ，自己位置を連続的に更新できることが確認された。これにより，自己位置推定の安定性と連続性の両立が可能となり，屋内ナビゲーションへの適用において有効であることが示唆される。

第8章　まとめ

8.1　本研究の成果

本研究では、簡易モデルを用いた屋内環境での自己位置推定手法を提案した。提案手法を用いた道案内 AR アプリケーションの実装を通じて、自己位置推定結果が実用的な精度で取得できることを確認した。

8.2　今後の展望

現状は、自己位置推定にテクスチャの点特徴のみを用いている。ワイヤーフレーム構造などから線特徴を用いることができれば、さらなる自己位置推定精度の向上や、点特徴の補完に寄与する可能性がある

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] 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.

[7] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast Image-Based Localization using Direct 2D-to-3D Matching. Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 667–674, 2011.

[8] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Aggregating Local Descriptors into a Compact Image Representation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3304–3311, 2010.

[9] 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.

[10] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. Proceedings of Robotics: Science and Systems (RSS), pp. 1–9, 2014.

[11] Paul Biber and Wolfgang Straßer. The Normal Distributions Transform: A New Approach to Laser Scan Matching. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2743–2748, 2003.

[12] Stephan Kohlbrecher, Johannes Meyer, Thomas Graber, et al. Hector SLAM: Real-Time SLAM with a Single Laser Scanner. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3987–3994, 2011.

[13] Andreas Schindler, Jan-Michael Frahm, and Marc Pollefeys. City-Scale Location Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8, 2007.

[14] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast Image-Based Localization using Direct 2D-to-3D Matching. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 667–674, 2011.

[15] Jonathan R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," in *Applied Computational Geometry: Towards Geometric Engineering*, Proceedings of the First ACM Workshop on Applied Computational Geometry, Lecture Notes in Computer Science, Vol. 1148, pp. 203–222, 1996.

[16] 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.

[17] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, Vol. 60,No. 2,pp. 91–110,2004.

[18] P. F. Alcantarilla,J. Nuevo,A. Bartoli, Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces, Proceedings of the British Machine Vision Conference (BMVC),2013.

[19] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-Supervised Interest Point Detection and Description," arXiv preprint arXiv:1712.07629, 2017. :contentReference[oaicite:0]index=0

[20] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning Feature Matching with Graph Neural Networks," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. :contentReference[oaicite:1]index=1

[21] 松下侑聖, "直交射影に基づく PnPL 問題に対する大域最適解の計算", pp. 4–11, 2024.

[22] Apple Inc., "Managing Session Life Cycle and Tracking Quality," Apple Developer Documentation, https://developer.apple.com/documentation/arkit/managing_session_lifecycle_and_tracking_quality, （参照日：2026 年 1 月）.

[23] Apple Inc., "Understanding World Tracking," Apple Developer Documentation, https://developer.apple.com/documentation/arkit/world_tracking/understanding_world_tracking, （参照日：2026 年 1 月）.