修士論文

令和３年度

研究題目

360 度画像を用いた SfM の精度に関する研究

学生証番号　086105

氏名　西島　啓斗

提出日　令和４年１月１４日

指導教員　蚊野　浩

京都産業大学大学院
先端情報学研究科

要旨

Structure from Motion（SfM）は，3 次元シーンを撮影した複数の画像から，シーンに関する 3 次元情報と撮影に用いたカメラの位置情報を推定する技術である．通常のカメラを用いた SfM により，屋外の情景や観光地などの広い 3 次元空間を 3 次元モデル化することが可能になっている．通常のカメラは画角が 40 度程度であり，広い空間を SfM するためには非常に多数の写真画像を撮影する必要がある．この問題を解決するには，通常より視野の広い画像を使う SfM が有効である．

最近では，360 度の視野を手軽に撮影できる 360 度カメラが容易に入手できる．SfM に 360 度画像を利用すれば，1 枚の画像で撮影できる範囲が広いことによる撮影枚数の低減や，360 度画像の良い性質による SfM の復元精度の向上などの効果が期待できる．本研究では，360 度画像を用いた SfM を研究・実装し，通常画像を用いる SfM との比較，検討を行った．

360 度画像を用いた SfM と通常画像による SfM の精度を比較するために，シミュレーションによる実験を行った．3 次元点群のシミュレーションデータを，通常の画像面と単位球面に投影し，その位置に適切なノイズを加えた画像を作成した．それらの画像について，提案手法を MATLAB のプログラムで実装し，カメラ移動量の復元を行った．復元された移動量と真値との誤差を評価し，精度を検証した．結果として 1）360 度画像を用いた SfM の精度は，通常画像を用いた SfM よりも改善されていた．また，2）360 度画像を用いた SfM では，カメラの移動方向と SfM の復元精度に関係が見られなかったのに対して，通常画像を用いた SfM ではカメラを前に動かした場合では，上，右に動かした場合よりも復元精度が悪かった．1）のようになる理由は，360 度画像は全方向の観察点を利用してカメラ移動量の計算を行うことができることであると考えた．2）のようになる理由は，カメラを前後に移動させて撮影した通常画像では，上下や左右に移動させて撮影した通常画像に比較して，3 次元点を投影した観察位置の変化が少ないからである，と考えた．360 度画像の場合には，通常画像のような方向性がないため，カメラをどのように移動させても，同じ程度の精度で SfM を実行することができると考える．

提案手法が実画像に対しても動作することを示すために，3DCG ソフトで作成したシーンの 360 度画像を用いる場合と，実際の 360 度カメラ（リコー社の THETA）で撮影した 360 度画像を用いる場合の 2 つで SfM を行い，動作を検証した．入力画像から ORB 特徴を抽出し，それを 2 画像間でマッチングさせ，提案手法で SfM を行った．その結果を可視化し，正しく動作していることを確認した．

目次

第1章 序論

人間は3次元のシーンを観察した時，脳内の処理によって様々な情報を導き出し，そのシーンを理解することができる．人間の目と脳の機能をカメラとコンピュータに置き換えて，同様な情報処理能力を人工的に実現しようとする学問分野をコンピュータビジョンと呼ぶ．コンピュータビジョン技術の研究開発は画期的に進歩しており，実用化が進んでいるものに，例えば，自動運転に利用されるビジョン技術や，スマートフォンのカメラ撮影に利用されるコンピュテーショナルフォトグラフィなどがある．

カメラで撮影した写真画像は3次元世界を2次元画像に投影している．そのため3次元世界の幾何学的な情報などが縮退しており，1枚の写真画像だけから，元の3次元世界の立体構造を復元することは，一般に困難である．一方，2台のカメラを配置したステレオカメラを用いれば，視野で捉えたシーンの空間的な情報を計算することができる．このような技術をステレオビジョンという．ステレオビジョンは，性能が既知の2台以上のカメラを用いて，かつ，カメラ間の位置関係が既知の場合に，観察したシーンの空間情報を推定する技術である．

コンピュータビジョンの技術が発展するに伴い，移動する1台のカメラで撮影した複数の画像や映像から，まず，カメラの移動に関する情報を推定し，次いで，カメラの移動情報を利用してシーンの空間情報を推定する，という技術が研究されるようになった．この技術を Structure from Motion（SfM）という．近年ではコンピュータの性能が向上したことにより，大量の画像を対象とした SfM が可能になり，大きな空間の 3 次元モデルを再構築できるようになっている [1].

また，移動ロボットのナビゲーションに利用される技術として SLAM（Simultaneous Localization and Mapping）がある．SLAM は移動ロボットに搭載した画像センサーや慣性センサーを用いて，移動ロボットの移動情報と環境の地図情報を同時に推定する技術である．SLAM の中でも，画像情報を使う SLAM を画像 SLAM（Visual SLAM）と呼ぶ．画像 SLAM も SfM 技術の一つであり，実時間で処理が行われる高速 SfM と考えることができる．このように SfM 技術は，広域空間の 3 D モデリングや，移動ロボットのナビゲーションなど，いくつかの分野で研究と実用化が進められている．

コンピュータビジョンに利用するカメラは通常のカラーカメラを用いることが多い．通常のカメラはピンホールカメラでモデル化することができる．通常のカメラに標準レンズを装着する場合，画角は 46 度程度である．広角レンズであれば 90 度ぐらいになるが，それでも，全天 360 度に対して，面積にして 15%程度しかカバーすることができない．一方，RICHO THETA や GoPro MAX といった 360 度写真を撮影できるカメラを用いれば，全天 360 度をワンショットで撮影することができる．近年，これらの 360 度カメラは，多数の機種が発売されており，容易に入手できるようになっている．本研究では，

360 度画像を利用して SfM を行うことで，精度の向上や，撮影の手間の削減が達成できるのではないかと考え，360 度画像を使った SfM について研究した．

本論文の構成は次のとおりである．第 2 章では SfM の基礎となる理論や技術の説明を述べる．第 3 章では SfM を 360 度画像に適用する手法を述べる．第 4 章では作成した SfM プログラムを用いて 360 度画像と通常画像それぞれで SfM を行い，復元結果を比較した実験結果を示す．最後に第 5 章で結論を述べる．

第2章 SfM に関する従来技術と従来研究

第 2 章では，SfM の基礎となる通常のカメラを用いた 2 画像間での 3 次元復元について説明する．まず，通常のカメラを用いた SfM について説明し，また関連技術として，画像特徴と 360 度画像を説明する．

2.1 通常のカメラを用いた 2 画像間の 3 次元復元

2.1.1 ピンホールカメラモデル

通常のカメラを用いた画像撮影のモデルである，ピンホールカメラモデルについて説明する．カメラを用いた撮影における幾何学的な性質は，座標系の変換によって説明することができる．図 2.1 に示すように，座標系としてワールド座標系，カメラ座標系，画像座標系の 3 つを利用する．ワールド座標系とカメラ座標系は 3 次元の直交座標系，画像座標系は 2 次元の直交座標系である．

一般にワールド座標系は，ユーザが任意に設定する座標系である．SfM では基準となるカメラを決めて，基準カメラのカメラ座標系とワールド座標系を合わせる場合が多い．カメラ座標系はカメラごとに設定する座標系で，カメラをモデル化するピンホールカメラの投影中心を原点とし，カメラの水平方向をX_C軸，垂直方向をY_C軸，奥行き方向をZ_C軸とする．画像座標系は，各カメラが撮影する画像上に設定する 2 次元座標である．画像座標系として，縦横の長さを画素単位で数える通常の画像座標と，カメラ座標系における$Z_C = 1$の平面として定義される正規化画像座標がある．

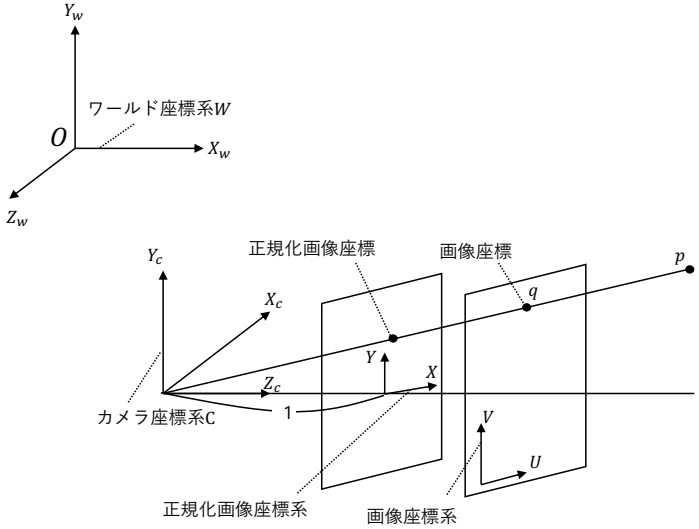

図 2.1 ワールド座標系，カメラ座標系，画像座標系の関係

ある 3 次元の点pを画像として撮影することは，ワールド座標系，カメラ座標系，画像座標系の間で変換を行うことである．座標系の間の変換は主に 3 段階に分けて考えることができる．変換の手順を 2.1.1-1 から 2.1.1-3 で説明する．

なお，本論文では座標値の表記に縦ベクトルを用いることを基本とする．また，座標値を同次座標表現することで，ピンホールカメラによる座標系の変換を行列演算の形で表すことができ，関係を見通しよく表すことができる．同次座標は通常の座標ベクトルの最後に 1 を付け加えたものである．また，同次座標を表す記号に対しては，記号の上にチルダを加えることで明記する．例えば，2 次元座標$\boldsymbol{m}=\begin{pmatrix}x\\y\end{pmatrix}$，3 次元座標$\boldsymbol{M}=\begin{pmatrix}X\\Y\\Z\end{pmatrix}$の同次座標は，次のようになる．

$$\tilde{\boldsymbol{m}}\sim\begin{pmatrix}x\\y\\1\end{pmatrix},\tilde{\boldsymbol{M}}\sim\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}$$

なお，同次座標は定数倍の違いを許して等しいという性質があり，このことを記号‘〜’で表す．

2.1.1-1 ワールド座標系からカメラ座標系への変換

ワールド座標系の点$\boldsymbol{M}_w$からカメラ座標系の点M_cに変換について考える．点$\boldsymbol{M}_c$は，ワールド座標系の原点から回転行列$\boldsymbol{R}$だけ回転し，並進ベクトル$\boldsymbol{t}$だけ平行移動したカメラで撮影した点であるとする．$\boldsymbol{M}_w$から$\boldsymbol{M_c}$への変換は剛体変換で表すことができ，変換の式は$\boldsymbol{M}_c=\boldsymbol{R}\boldsymbol{M}_w+\boldsymbol{t}$と表される．このときの$\boldsymbol{R}$と$\boldsymbol{t}$をカメラの外部パラメータと呼ぶ．$\boldsymbol{R}$，$\boldsymbol{t}$はそれぞれ次のような行列とベクトルである．

$$\boldsymbol{R}=\begin{pmatrix}r_{11}&r_{12}&r_{13}\\r_{21}&r_{22}&r_{23}\\r_{31}&r_{32}&r_{33}\end{pmatrix},\boldsymbol{t}=\begin{pmatrix}t_1\\t_2\\t_3\end{pmatrix}$$

同次座標を用いると，この変換は以下のようになる．

$$\tilde{\boldsymbol{M}}_c\sim\boldsymbol{D}\tilde{\boldsymbol{M}}_w$$

ここで，$\boldsymbol{D}$は外部パラメータからなる以下の 4×4 の行列であり，外部パラメータ行列と呼ぶ．

$$\boldsymbol{D}=\begin{pmatrix}\boldsymbol{R}&\boldsymbol{t}\\\boldsymbol{0}^T&1\end{pmatrix}=\begin{pmatrix}r_{11}&r_{12}&r_{13}&t_1\\r_{21}&r_{22}&r_{23}&t_2\\r_{31}&r_{32}&r_{33}&t_3\\0&0&0&1\end{pmatrix}$$

ここで，$\boldsymbol{0}^T$のTは行列の転置を表し，$\boldsymbol{0}_{3\times1}^T=\boldsymbol{0}_{1\times3}$となる．

2.1.1-2 カメラ座標系から正規化画像座標系への変換

カメラ座標系に変換した点$\boldsymbol{M}_c$を，カメラ座標系で$Z_c=1$に存在する仮想的な画像平面上の点$\boldsymbol{m}_c$へ投影する．この変換を，同次座標で行うと次の式になる．

$$\tilde{\boldsymbol{m}}_c\sim\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix}\tilde{\boldsymbol{M}}_c$$

この変換を，正規化画像座標系への透視投影変換という．

2.1.1-3 カメラ座標系から画像座標系への変換

次に，正規化画像座標系の点$\boldsymbol{m_c}$から，画像座標系の点$\boldsymbol{m}$への変換を考える．この変換は行列$\boldsymbol{A}$を点$\boldsymbol{m_c}$に掛けることで行う．同次座標を用いると，変換は以下のような式になる．

$$\tilde{\boldsymbol{m}} \sim \boldsymbol{A}\tilde{\boldsymbol{m}}_c$$

このとき，行列$\boldsymbol{A}$は以下のような3×3の行列である．

$$\boldsymbol{A} = \begin{pmatrix} \frac{f}{\delta_u} & 0 & c_u \\ 0 & \frac{f}{\delta_v} & c_v \\ 0 & 0 & 1 \end{pmatrix}$$

ここで，fはカメラの焦点距離，c_u, c_vはカメラ座標系における，カメラの光軸と画像面との交点座標を画像座標系で表した位置（画像中心），δ_u, δ_vはそれぞれ横と縦の画素サイズである．

以上のように，行列$\boldsymbol{A}$はカメラの焦点距離，画像中心，画素サイズからなる行列である．この行列$\boldsymbol{A}$を内部パラメータ行列と呼び，行列$\boldsymbol{A}$を構成している3つの要素をピンホールカメラの内部パラメータと呼ぶ．

以上の変換をまとめて行う式は，以下のように表される．

$$\tilde{\boldsymbol{m}} \sim \boldsymbol{A}(\boldsymbol{R} \quad \boldsymbol{t})\tilde{\boldsymbol{M}}_w$$

ここで，$\boldsymbol{A}(\boldsymbol{R} \quad \boldsymbol{t})$をまとめると3×4の行列$\boldsymbol{P}$になり，座標変換は次のような式になる．

$$\boldsymbol{P} = \boldsymbol{A}(\boldsymbol{R} \quad \boldsymbol{t}) = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix}$$

$$\tilde{\boldsymbol{m}} \sim \boldsymbol{P}\tilde{\boldsymbol{M}}_w$$

この行列$\boldsymbol{P}$を透視投影行列，またはカメラ行列と呼ぶ．行列$\boldsymbol{P}$を，同次座標で表現した3次元ワールド座標に掛けることで，2次元の画像座標に変換できる事がわかる．透視投影行列を用いることで，画像の撮影を行列の演算の形で表すことができる．

2.1.2 エピポーラ幾何と$\boldsymbol{E}$行列，$\boldsymbol{F}$行列

2台のカメラで撮影した画像の間の関係を，エピポーラ幾何を用いて表すことができる．エピポーラ幾何の説明を図2.2に示す．ある3次元点$\boldsymbol{M}$が2台のカメラで撮影されたとする．このとき，2つのカメラの投影中心と点$\boldsymbol{M}$でできる平面を，エピポーラ平面と呼ぶ．また，エピポーラ平面と2つの画像面が交わる線をエピポーラ線，2つのカメラのカメラ中心を結んだ線が画像面と交わる点をエピポールと呼ぶ．

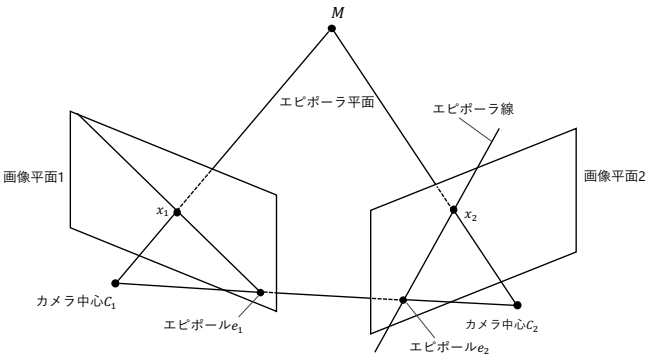

図 2.2 エピポーラ幾何

図 2.2 のように，3 次元点$\boldsymbol{M}$が画像平面 1 において$\boldsymbol{x_1}$の位置に観察されたとする．このとき，画像平面 2 における対応する観察点$\boldsymbol{x_2}$は，エピポーラ線上に存在する．$\boldsymbol{x_1}, \boldsymbol{x_2}$が正規化画像座標系での同次座標であるとき，任意の$\boldsymbol{x_1}, \boldsymbol{x_2}$の間で

$$\boldsymbol{x_2^T E x_1} = 0 \tag{2.1}$$

の関係が成り立ち，この 3×3 の行列$\boldsymbol{E}$を基本行列（Essential Matrix，$\boldsymbol{E}$行列）と呼ぶ．$\boldsymbol{E}$行列はカメラ間の位置関係（$\boldsymbol{R}$と$\boldsymbol{t}$）のみで決まり，正規化画像座標系における対応点同士の関係を表現している．$\boldsymbol{E}$行列に内部パラメータの情報は含まれていない．

$\boldsymbol{x_1}, \boldsymbol{x_2}$に内部パラメータ行列$\boldsymbol{A}$を掛けると，通常の画像座標を得る．2 つの画像が内部パラメータ行列$\boldsymbol{A}$を持つ 1 台のカメラで撮影されたとすると，画像座標は$\tilde{\boldsymbol{m}}_1 = \boldsymbol{A x_1}, \tilde{\boldsymbol{m}}_2 = \boldsymbol{A x_2}$となる．この画像座標間の関係を，以下の式で表すことができる．

$$\tilde{\boldsymbol{m}}_2^T \boldsymbol{F} \tilde{\boldsymbol{m}}_1 = 0 \tag{2.2}$$

行列$\boldsymbol{F}$も 3×3 の行列であり，基礎行列（Fundamental Matrix，$\boldsymbol{F}$行列)と呼ばれる．$\boldsymbol{F}$行列には，カメラの内部パラメータと外部パラメータ両方の情報が含まれている．$\boldsymbol{E}$行列と$\boldsymbol{F}$行列には以下の関係がある．

$$\boldsymbol{E} = \boldsymbol{A}^T \boldsymbol{F} \boldsymbol{A}$$

$\boldsymbol{E}$行列や$\boldsymbol{F}$行列によって画像座標間の関係が表されている．２枚の画像上の，２つの画像座標が空間上の同じ点に対応していればこの条件を満たし，これをエピポーラ拘束と呼ぶ．

2.1.3 8 点アルゴリズム

通常のカメラに関するエピポーラ拘束条件として式(2.2)がある．この拘束条件を利用して，$\boldsymbol{F}$行列を推定するアルゴリズムを説明する．

ここまでは 1 組の対応点について考えたが，対応点の組が N 組ある場合を考える．まず N 組のうち，i番目の対応点について考える．i番目の対応点が$\tilde{\boldsymbol{m}}_i = \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix}, \tilde{\boldsymbol{m}}_i' = \begin{pmatrix} u_i' \\ v_i' \\ 1 \end{pmatrix}$であるとき，エピポーラ拘束条件は$\tilde{\boldsymbol{m}}_i'^T \boldsymbol{F} \tilde{\boldsymbol{m}}_i = 0$となる．この式は，行列$\boldsymbol{F}$の 9 要素を列ベクトルとして，次のように変形できる．

$$\boldsymbol{u}_i^T \boldsymbol{f} = 0 \tag{2.3}$$

ここで，$\boldsymbol{u}_i$と$\boldsymbol{f}$は以下のようなベクトルである．

$$\boldsymbol{u}_i = (u_i{'}u_i \quad u_i{'}v_i \quad u_i{'} \quad v_i{'}u_i \quad v_i{'}v_i \quad v_i{'} \quad u_i \quad v_i \quad 1)^T$$
$$\boldsymbol{f} = (f_{11} \quad f_{12} \quad f_{13} \quad f_{21} \quad f_{22} \quad f_{23} \quad f_{31} \quad f_{32} \quad f_{33})^T$$

N 組の対応点が与えられた場合，$\boldsymbol{u}_i$を積み重ねた次の連立方程式を解くことで$\boldsymbol{F}$行列を求めることができる．

$$\boldsymbol{U}\boldsymbol{f} = 0$$

ここで，$\boldsymbol{U}$は$\boldsymbol{u}_i$を積み重ねた N×9 の行列

$$\boldsymbol{U} = \begin{pmatrix} u_1{'}u_1 & u_1{'}v_1 & u_1{'} & v_1{'}u_1 & v_1{'}v_1 & v_1' & u_1 & v_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_N{'}u_N & u_N{'}v_N & u_N{'} & v_N{'}u_N & v_N{'}v_N & v_N' & u_N & v_N & 1 \end{pmatrix} \tag{2.4}$$

である．これにより，エピポーラ拘束条件の式から，対応点の組の数だけ式がある連立方程式を得る．

2 つの画像の複数の対応点から$\boldsymbol{F}$行列を求めるアルゴリズムに，式(2.3)を用いる 8 点アルゴリズムがある．$\boldsymbol{F}$行列は 9 個の要素があるが，定数倍の自由度があるため，8 組以上の対応点があれば$\boldsymbol{F}$行列を計算できる．対応点が 8 組ちょうどの場合は，単に連立 1 次方程式を解けばよい．対応点が 9 組以上ある場合は，最小二乗法を用いて$\boldsymbol{F}$行列を求める．対応点が N 組ある場合，エピポーラ拘束条件の式から，以下のようになる．

$$\boldsymbol{F} = \underset{\boldsymbol{F}}{\operatorname{argmin}} \sum_{i=1}^{N} (\tilde{\boldsymbol{m}}_i{'}^T \boldsymbol{F} \tilde{\boldsymbol{m}}_i)^2$$

この式を，$\tilde{\boldsymbol{m}}_i{'}^T \boldsymbol{F} \tilde{\boldsymbol{m}}_i = \boldsymbol{U}\boldsymbol{f}$であることから書き換えると

$$\boldsymbol{f} = \underset{\boldsymbol{f}}{\operatorname{argmin}} \sum_{i=1}^{N} (\boldsymbol{U}\boldsymbol{f})^2$$
$$\sum_{i=1}^{N} (\boldsymbol{U}\boldsymbol{f})^2 = \boldsymbol{f}^T \boldsymbol{U}^T \boldsymbol{U}\boldsymbol{f}$$
$$\boldsymbol{f} = \underset{\boldsymbol{f}}{\operatorname{argmin}}\, \boldsymbol{f}^T \boldsymbol{U}^T \boldsymbol{U}\boldsymbol{f}$$

となる．ここで，$\boldsymbol{f}$のスケールは任意である．$\boldsymbol{f} = 0$となるのを避けるために拘束条件として$\|\boldsymbol{f}\| = 1$を加えて解く場合，$\boldsymbol{f}$は$\boldsymbol{U}^T\boldsymbol{U}$の最小固有値に対応する固有ベクトルである．あるいは$\boldsymbol{U}^T\boldsymbol{U}$を特異値分解し，最小特異値に対応する特異ベクトルとして求めることもできる．

上記の計算法を通常の画像座標にそのまま用いると，計算誤差の影響が大きくなる．計算精度を上げる手法として Hartley の正規化 8 点アルゴリズムがある．これは画像の座標原点を，通常の左上から特徴点の重心へ移し，画像のスケールを$-\sqrt{2}$～$\sqrt{2}$の範囲に拡大縮小する正規化の処理を行い，そして線形解法を適用する解法である [2] [3].

本研究では MATLAB のコンピュータビジョンツールボックスを用いた．このツールボックスには$\boldsymbol{F}$行列を推定する関数 estimateFundamentalMatrix がある．この関数では，下

枠内手順 2.1 のように，Hartley の正規化 8 点アルゴリズム [3]を用いて$\boldsymbol{F}$行列を推定している．

手順 2.1 MATLAB の正規化 8 点アルゴリズムの実装
1．スケールと画像原点について正規化を行う
2．対応点$\boldsymbol{x}, \boldsymbol{x}'$から式(2.4)のように制約行列$\boldsymbol{U}$を作る
3．行列$\boldsymbol{U}$を特異値分解し，最小特異値に対応する特異ベクトルを行列$\boldsymbol{F}_{3\times3}$に変形する
4．$\boldsymbol{F}$を特異値分解する．3 つある特異値の 3 番目を 0 にすることで，rank2になるという拘束を満たすよう調整する
5．正規化をもとに戻す
6．行列$\boldsymbol{F}$をノルムで割って 1 に正規化する

本研究では$\boldsymbol{F}$行列の推定に 8 点アルゴリズムを用いたが，より精度の高いアルゴリズムとして FNS 法などが存在する [4].

2.1.4 アウトライアと RANSAC

画像から特徴点を検出し，対応付けを行うとき，異なる特徴を間違って対応付けてしまうことがある．間違って対応付けされた特徴点をアウトライア（外れ値）と呼ぶ．反対に，正しく対応付けされた特徴点をインライアと呼ぶ．アウトライアが存在すると，$\boldsymbol{F}$行列の推定精度に悪影響を与える．そのため，$\boldsymbol{F}$行列を計算する前にアウトライアを除去する必要がある．アウトライアを除去するアルゴリズムとして，RANSAC（Random Sample Consensus）がある．MATLAB の関数 estimateFundamentalMatrix では，$\boldsymbol{F}$行列を計算する際に外れ値を除去するオプションとして RANSAC を使用できる．

2.1.5 $\boldsymbol{F}$行列からカメラモーションを求める

求めた$\boldsymbol{F}$行列から，２枚の画像を撮影したカメラの回転行列$\boldsymbol{R}$と並進ベクトル$\boldsymbol{t}$を求めることができる．このために，まず，$\boldsymbol{F}$行列から内部パラメータ行列$\boldsymbol{A}$を用いて$\boldsymbol{E}$行列を求める．

$$\boldsymbol{E} = \boldsymbol{A}^T \boldsymbol{F} \boldsymbol{A}$$

$\boldsymbol{E}$行列と 2 画像の対応点の組から特異値分解を用いてカメラの動き（$\boldsymbol{R}$と$\boldsymbol{t}$）を計算する．$\boldsymbol{E}$行列は次のように分解できることがわかっている．

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$$

ここで，$[t]_\times$は並進ベクトルの各要素T_x, T_y, T_zを用いる以下のような行列である．

$$[\boldsymbol{t}]_\times = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$

この行列は歪対称行列と呼ばれる．歪対称行列は$[t]_\times^T = -[t]_\times$という性質を持っている．また，この行列を用いてベクトルの外積を表すことができる．あるベクトル$\boldsymbol{a}, \boldsymbol{b}$があったとき$\boldsymbol{a}$の歪対称行列を$\boldsymbol{b}$に掛ける演算$[\boldsymbol{a}]_\times \cdot \boldsymbol{b}$は，$\boldsymbol{a}$と $\boldsymbol{b}$の外積$\boldsymbol{a} \times \boldsymbol{b}$と等しくなる．

$\boldsymbol{F}$行列を$\boldsymbol{R}, \boldsymbol{t}$に分解するアルゴリズムは以下のようになる [3].

手順 2.2 MATLAB 関数 relativeCameraPose のアルゴリズム

1. $\boldsymbol{E}$行列を特異値分解する $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$
2. 特異値行列 $\boldsymbol{D}$ から，1 番目と 2 番目の特異値の平均を e とする
3. 3 つの特異値を$e, e, 0$として，$\boldsymbol{E} = \boldsymbol{U}\, diag(e, e, 0)\, \boldsymbol{V}^T$とする
4. 行列 $\boldsymbol{E}$ を特異値分解する $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$
5. 2 つの行列$\boldsymbol{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ $\boldsymbol{Z} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$を作成

- 回転行列

6. $\boldsymbol{U}, \boldsymbol{W}, \boldsymbol{V}$から取りうる可能性のある回転行列 2 つを作成する

$$\boldsymbol{R1} = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T$$

$$\boldsymbol{R2} = \boldsymbol{U}\boldsymbol{W}^T\boldsymbol{V}^T$$

7. 回転行列であることから，行列$\boldsymbol{R1}, \boldsymbol{R2}$の行列式が 1 になるよう符号を調整する

- 並進ベクトル

8. 並進ベクトルについて，歪対称行列を$\boldsymbol{U}, \boldsymbol{Z}$から作成

$$[\boldsymbol{t}]_\times = \boldsymbol{U}\boldsymbol{Z}\boldsymbol{U}^T$$

9. 歪対称行列から並進ベクトルを求める

この時点で，$\boldsymbol{R}, \boldsymbol{t}$の組み合わせは，それぞれにプラスとマイナスがあり，合計 4 つが存在する．この内，物理的に実現可能な組み合わせを選ぶ．組み合わせの中には，3 次元点がカメラの後ろ側に存在することになるものもあり，これは物理的に不可能である．
MATLAB の関数では，4 つの組み合わせの中で，物理的に実現可能なものだけを残すことで符号を決定している．また，並進ベクトル$\boldsymbol{t}$の大きさは決定することができないので，単位ベクトル化した$\boldsymbol{t}$を求める．

2.1.6 特徴点の 3 次元空間における位置を求める

以上で 2 つのカメラ間の並進ベクトルと回転行列が求まった．内部パラメータは既知であることから，特徴点の 3 次元空間における位置を求めることができる．3 次元位置は投影中心から観察点への視線を考え，2 つの画像からの視線が交わる点が観察点の 3 次元位置であるとすることで求められる．

2.2 Structure from Motion

複数枚の画像が撮影され，画像に写っている特徴点の対応がわかっているとする．この対応点の情報のみから，特徴点の 3 次元位置とカメラの動きを推定することを Structure from Motion（SfM）と呼ぶ．2.1 節で述べた手順を繰り返し行えば良い．SfM の手順は以下のようになる．

手順 2.3 SfM の処理の流れ
1. ２枚の画像から特徴点を検出し，特徴点を対応させる
2. 対応した点から $\boldsymbol{F}$ 行列を推定する
3. $\boldsymbol{F}$ 行列からカメラの回転 $\boldsymbol{R}$ と並進 $\boldsymbol{t}$ を推定する
4. 特徴点の 3 次元位置を計算する
5. 複数視点からの復元結果の誤差が最小になるよう最適化を行う

内部パラメータが既知の場合，$\boldsymbol{F}$行列を求める際に必要な対応点の組の数を考える．回転は自由度 3 であり，平行移動はスケールが決まらないことから，自由度は 2 である．以上より，外部パラメータの自由度（変数の数）は 3＋2＝5 自由度となる．よって，内部パラメータがわかっている場合は，最低で 5 つの対応点の組があれば，$\boldsymbol{F}$行列を計算することができ，カメラの動きと物体の 3 次元構造を計算できる．ただし，画像だけを用いている場合には，原理的にスケールを定めることはできない．

2.2.1 バンドル調整

多数の画像から SfM を行う場合，全ての画像を同時に処理するのではなく，2 枚の画像で SfM を行い，それを順番に積み重ねていく．$\boldsymbol{F}$行列や$\boldsymbol{E}$行列の推定を，複数枚の画像から連続して行うと誤差が蓄積する．蓄積した誤差の影響を小さくするために，バンドル調整を行う．バンドル調整とは，カメラの内部パラメータ，外部パラメータ，復元した 3 次元点の座標を微小に変化させ，誤差が最も小さくなるように調整することである．

具体的には，以下のコスト関数Errを最小化するような内部パラメータ，外部パラメータ，３次元座標を求める．コスト関数は次のようになる．

$$Err(\boldsymbol{P}_1 \dots \boldsymbol{P}_m, \boldsymbol{q}_1 \dots \boldsymbol{q}_n) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left\{\left(u_{ij} - \hat{u}(\boldsymbol{P}_i, \boldsymbol{q}_j)\right)^2 + \left(v_{ij} - \hat{v}(\boldsymbol{P}_i, \boldsymbol{q}_j)\right)^2\right\}$$

ここで$\boldsymbol{P}$は内部パラメータと外部パラメータからなる透視投影行列，$\boldsymbol{q}$は復元された 3 次元点，(u_{ij}, v_{ij})は画像の特徴点座標，$(\hat{u}(\boldsymbol{P}_i, \boldsymbol{q}_j), \hat{v}(\boldsymbol{P}_i, \boldsymbol{q}_j))$は$\boldsymbol{q}$を画像面に再投影して得られた 2 次元座標である．

バンドル調整の計算には，非線形最小二乗法を利用する．非線形最小二乗法を解くためのアルゴリズムとして Levenberg–Marquardt 法がある．非線形最小二乗法を解くライブラリとして，Ceres Solver や MATLAB の lsqnonlin 関数がある．

2.3 画像特徴

画像中のコーナーのように顕著な変化が生じている箇所を特徴点という．また，その特徴点の内容を記述するものを特徴量という．特徴点を検出する方法として，FAST や DoG がある．また，特徴量を記述する特徴記述子には特徴をベクトルとして表す SIFT やバイナリコードとして表す BRIEF がある．特徴検出と特徴記述子を組み合わせて，画像から特徴点を検出，記述する手法として，SIFT や AKAZE，ORB が存在する．

特徴をバイナリコードとして表現する方法の利点として，ベクトル表現の場合と比較してメモリ消費が少ない，類似度を高速に計算できる事が挙げられる．これらの特徴点を用いて SfM の処理を進める方法を，特徴点ベースの方法と呼ぶ．

2 枚の画像があったとき，それぞれの画像の 2 つの特徴点が同じ部分を示しているかを判定する手法を特徴点マッチングという．これにより，異なる画像の間で同じものを写している点を対応付けることができる．特徴量がベクトルの場合，特徴点の類似度はユークリッド距離として求められる．バイナリコードで表現されている ORB の場合は，ハミング距離を類似度とする．

2.4 360 度画像の表現形式

360 度画像を撮影できるカメラが一般的に入手できるようになっており，図 2.3 に示すような RICHO THETA，GoPro MAX や Insta360 などがある．これらは，180 度以上の視野を持つ魚眼レンズを 2 つ組み合わせることで，360 度視野の画像や動画を撮影できるようにしている．撮影した画像や動画は，2 つの魚眼画像から変換され，正距円筒画像の形式で保存される．

図 2.3 360 度カメラの例
左：RICHO THETA Z1　中央：GoPro MAX　右：Insta360 ONE X2

2.4.1 正距円筒画像

通常の透視投影画像は 180 度より大きな視野を表現することができない．360 度画像を保存するための形式として，正距円筒図法（Equirectangular）がある．これは，画像の横軸に経度，縦軸に緯度をとり，2 軸が等間隔かつ直交するような形式である．正距円筒図

法で保存された画像を，正距円筒画像と呼ぶ．正距円筒画像では，緯度が赤道から縦方向に離れるほど，像が横方向に引き伸ばされる．図 2.4 に正距円筒図法の例を示す．赤道上で正円であるオレンジの円が，上下方向では横に引き伸ばされている事がわかる．

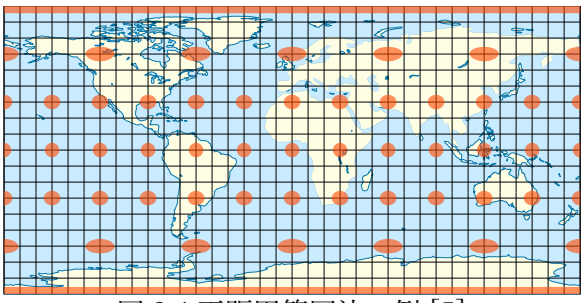

図 2.4 正距円筒図法の例 [5]

正距円筒画像では座標を緯度φと経度θの 2 つで表す．画像の中央を原点として，θ は$-\pi$~π，φは-π/2~π/2 の範囲をとる．図 2.5 に正距円筒画像の座標系と撮影例を示す．

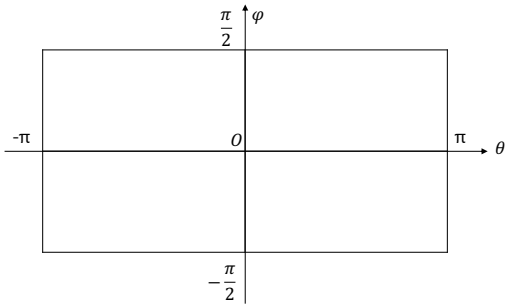

図 2.5 正距円筒画像
左：正距円筒画像の座標　右：正距円筒画像の例

2.4.2 単位球面上に像を写すカメラモデル

360 度画像を用いた SfM の計算を考えやすくするため，360 度カメラで撮影された正距円筒画像を単位球面モデルに変換して扱う．そのために，正距円筒画像から検出された特徴点の座標を，単位球面上の 3 次元座標に変換する．これは，正距円筒形式の世界地図の一点を，地球儀上の一点に変換することに相当する．単位球面モデルは半径 1 の単位球を考え，その球面がカメラの撮像面であるとする．図 2.6 に単位球面モデルの図を示す．単位球面モデルでは焦点距離が 1，画像中心が座標原点である．

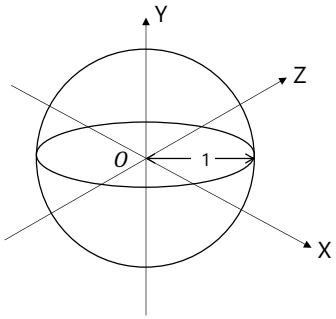

図 2.6 単位球面モデル

2.5 SLAM

SfM と関連する技術として，Simultaneous Localization And Mapping（SLAM）がある． SLAM は加速度センサや画像センサからのデータを用いて，自身の移動情報と周辺地図を同時に推定する技術であり，自動運転などの分野に用いられている．特に，画像センサのみを用いて SLAM を行う技術を画像 SLAM（Visual SLAM）と呼ぶ．画像 SLAM の主な研究には，PTAM [6]や ORB-SLAM [7]，LSD-SLAM [8]などがある．

これらの手法は周辺地図の復元方法によって，主に 2 種類に分かれる．1 つ目は特徴点ベースの方法と呼ばれるもので，PTAM や ORB-SLAM が該当する．この方法では画像から特徴を抽出し，特徴の 3 次元位置を復元している．2 つ目は直接法と呼ばれるもので，LSD-SLAM が該当する．直接法では画像の輝度勾配を利用し，画素単位で 3 次元位置を復元している．

第3章 360 度画像による SfM

第 3 章では， 360 度画像を用いて SfM を行う手法について述べる．

3.1 360 度画像を用いた 2 画像間の 3 次元復元

3.1.1 画像特徴の抽出

画像特徴は正距円筒画像から検出する． 特徴検出には MATLAB の関数 detectORBFeatures を用いる．この関数では特徴検出器に Oriented FAST，特徴記述子に ORB を用いる．

3.1.2 正距円筒画像から単位球面モデルへの変換

正距円筒画像から単位球面上の座標への変換は，2 段階の処理で行う．まず，正距円筒画像の座標値(i,j)から角度(θ,ϕ)を求める．そして，(θ,ϕ)を単位球面上の 3 次元座標(X,Y,Z)に変換する．正距円筒画像の画素数を横$xSize$，縦$ySize$とすると，(θ,ϕ)は次の式で求められる．

$$\theta = \frac{i-1}{xSize-1}\pi$$

$$\varphi = \frac{\pi(-j+ySize)}{-1+ySize} - \frac{\pi}{2}$$

また，単位球面上の 3 次元座標(X,Y,Z)は次の式で求められる．

$$X = \cos\varphi\sin\theta$$

$$Y = \sin\varphi$$

$$Z = \cos\varphi\cos\theta$$

3.1.3 360 度画像に対する$\boldsymbol{E}$行列の計算

単位球面モデルにおいてもエピポーラ拘束が成り立つことが知られている [9]．単位球面上の点を$\boldsymbol{e}_1, \boldsymbol{e}_2$とする．この点をそれぞれ$\lambda$倍，$\lambda'$倍すると，3 次元点$\boldsymbol{M}$になる．このことから，

$$\lambda\boldsymbol{e}_1 = \boldsymbol{M}$$

$$\lambda'\boldsymbol{e}_2 = \boldsymbol{M}' = \boldsymbol{R}\boldsymbol{M} + \boldsymbol{t}$$

上式から，

$$\lambda'\boldsymbol{e}_2 = \lambda\boldsymbol{R}\boldsymbol{e}_1 + \boldsymbol{t}$$

$$\lambda' e_2 - \lambda R e_1 - t = 0$$

従って，$\boldsymbol{e}_2, \boldsymbol{R}\boldsymbol{e}_1, \boldsymbol{t}$が 1 つの平面上にあるので，$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$として

$$\boldsymbol{e}_2^T \boldsymbol{E} \boldsymbol{e}_1 = 0$$

以上から，単位球面上の点の対応点の関係を$\boldsymbol{E}$行列で表す事ができる．

単位球面モデルに変換した特徴点の 3 次元座標を使って，通常画像の場合と同様に$\boldsymbol{E}$行列を 8 点アルゴリズムで求める．処理手順としては最小二乗法で$\boldsymbol{E}$行列を求めた後，特異値の調整を行う．

まず，以下のような行列$\boldsymbol{B}$を作成する．

$$\boldsymbol{B}_{N\times 9}=\begin{pmatrix} x_1'x_1 & x_1'y_1 & x_1'z_1 & y_1'x_1 & y_1'x_1 & y_1'z_1 & z_1'x_1 & z_1'y_1 & z_1'z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N'x_N & x_N'y_N & x_N'z_N & y_N'x_N & y_N'x_N & y_N'z_N & z_N'x_N & z_N'y_N & z_N'z_N \end{pmatrix}$$

また，$\boldsymbol{E}$行列の要素を縦に並べたベクトルを作成する．

$$\boldsymbol{E}=(e_{11}\quad e_{12}\quad e_{13}\quad e_{21}\quad e_{22}\quad e_{23}\quad e_{31}\quad e_{32}\quad e_{33})^T$$

$\boldsymbol{BE}=0$を満たすような$\boldsymbol{E}$を計算する．特徴点の抽出には誤差を含むので，最小二乗法を用いて，$\boldsymbol{BE}$が最小になるような$\boldsymbol{E}$を求める．このとき，$\boldsymbol{E}$行列が 0 にならないように$||\boldsymbol{E}||=1$という制約を追加する．このような制約のある最小二乗法を解く方法として，特異値分解を用いる方法がある．行列$\boldsymbol{B}$の最小特異値に対応する特異ベクトルが最小二乗解となる．

次に，$\boldsymbol{E}$行列の性質を用いて，求めた$\boldsymbol{E}$行列を調整する．ノイズのない環境で計算された$\boldsymbol{E}$行列の 3 つの特異値をS_1~S_3とすると，$S_1=S_2$，$S_3=0$であるという性質がある [3]．この特性に合うよう行列の特異値を調整することで，$\boldsymbol{E}$行列を求める．手順は以下のようになる．2 枚の単位球面画像の対応する観察点の座標を$\boldsymbol{n}=(x,y,z),\boldsymbol{n}'=(x',y',z')$であるとし，対応点が N 組あるとする．

手順 3.1 $\boldsymbol{E}$行列の計算

1. 行列$\boldsymbol{B}_{N\times 9}$を作る

$$\boldsymbol{B}_{N\times 9}=\begin{pmatrix} x_1'x_1 & x_1'y_1 & x_1'z_1 & y_1'x_1 & y_1'x_1 & y_1'z_1 & z_1'x_1 & z_1'y_1 & z_1'z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N'x_N & x_N'y_N & x_N'z_N & y_N'x_N & y_N'x_N & y_N'z_N & z_N'x_N & z_N'y_N & z_N'z_N \end{pmatrix}$$

2. $\boldsymbol{B}$を特異値分解する $\boldsymbol{B}=\boldsymbol{USV}^T$
3. 最小特異値に対応する特異ベクトル，すなわち$\boldsymbol{V}$の最後の列から$\boldsymbol{E}_{3\times 3}$を作る
4. $\boldsymbol{E}$行列の特異値S_1~S_3が$S_1=S_2$，$S_3=0$であるという制約から，
   $\boldsymbol{E}$を特異値分解し，$\boldsymbol{E}=\boldsymbol{U}\,diag(1,1,0)\,\boldsymbol{V}$とする

以上の方法によって求めた$\boldsymbol{E}$行列には，符号の不確定性が残る．すなわち，$\boldsymbol{E}$行列を$\boldsymbol{R},\boldsymbol{t}$に分解したとき，それぞれに符号の異なる複数個の解が求まる．また，$\boldsymbol{E}$行列の計算において RANSAC を用いて対応点の外れ値を除去する．これにより，外れ値の影響による推定誤差を少なくし，頑健な推定を行う．RANSAC は，以下の手順で処理する．

手順 3.2 RANSAC
繰り返し回数：$loop$　結果の$\boldsymbol{E}$行列：$\boldsymbol{E}_{result}$
1. 検出した特徴点からランダムに 8 点選ぶ
2. 選んだ特徴点から$\boldsymbol{E}$行列を求める
3. $\boldsymbol{E}$行列の精度を表す誤差の評価値を求める
4. 誤差の評価値が以前より小さくなっていれば$\boldsymbol{E}_{result} = \boldsymbol{E}$とする
5. 1~4 を$loop$回繰り返す
6. $\boldsymbol{E}_{result}$が最終的な$\boldsymbol{E}$行列となる

3.1.4 $\boldsymbol{E}$行列からカメラモーションを求める

計算した$\boldsymbol{E}$行列を，並進ベクトル$\boldsymbol{t}$と回転行列$\boldsymbol{R}$に分解する．まず，$\boldsymbol{E}$行列から並進ベクトルを計算する．この時点で並進ベクトルの符号を確定させることはできないので，並進ベクトルの候補として$\boldsymbol{t}_1$を求める．$\boldsymbol{E}$行列から並進ベクトルを求める方法は次のようになる．

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$$
$$\boldsymbol{E}^T\boldsymbol{t} = \boldsymbol{R}^T[\boldsymbol{t}]_\times^T\boldsymbol{t} = 0$$
$$\boldsymbol{E}\boldsymbol{E}^T\boldsymbol{t} = 0$$
$$\boldsymbol{E}\boldsymbol{E}^T = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} = (\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \boldsymbol{u}_3)\, diag(\lambda_1 \quad \lambda_2 \quad \lambda_3)\, (\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \boldsymbol{u}_3)^{-1}$$

以上より，$\boldsymbol{E}\boldsymbol{E}^T$の最小固有値$\lambda_{min}$に対応する単位固有ベクトル$\boldsymbol{u}_{min}$が並進ベクトル$\boldsymbol{t}_1$になる [4].

まとめると，$\boldsymbol{E}$行列から並進ベクトルを求める手順は手順 3.3 となる．

手順 3.3 $\boldsymbol{E}$行列から並進ベクトル$\boldsymbol{t}_1$を求める
1. 対称行列$\boldsymbol{E}\boldsymbol{E}^T$を固有値分解する．
2. 最小固有値に対応する単位固有ベクトルを$\boldsymbol{t}_1$とする

この時点で$\boldsymbol{E}$行列と$\boldsymbol{t}_1$の符号は不確定である．通常のカメラであれば，$\boldsymbol{R}, \boldsymbol{t}$の符号の異なる組み合わせのうち，復元した 3 次元点がカメラの前方に多く存在する組み合わせを選べば良い．しかし，360 度カメラでは全方向に 3 次元点が復元されるため，この方法を用いることができない．

$\boldsymbol{E}$行列と$\boldsymbol{t}_1$の符号を決めるために，外積（ベクトル積）を用いる．ある 3 次元点$\boldsymbol{p}$があるとき，カメラ 1 の座標系での座標を$\boldsymbol{p}_1$，カメラ 2 の座標系での座標を$\boldsymbol{p}_2$とする．カメラ 1 と 2 における点$\boldsymbol{p}$の観察点をそれぞれ$\boldsymbol{e}_1, \boldsymbol{e}_2$とすると，$\boldsymbol{p}_1, \boldsymbol{p}_2$はそれぞれ観察点を$L_1$倍，$L_2$倍した位置にあることになる．これは以下の式で表される．

$$\boldsymbol{p}_1 = L_1\boldsymbol{e}_1$$
$$\boldsymbol{p}_2 = L_2\boldsymbol{e}_2$$

$\boldsymbol{p}_2$をカメラ 1 の座標系に変換する式$\boldsymbol{p}_1 = \boldsymbol{R}\boldsymbol{p}_2 + \boldsymbol{t}$に上式を代入すると，

$$L_1\boldsymbol{e}_1 = \boldsymbol{R}(L_2\boldsymbol{e}_2) + \boldsymbol{t}$$

となる．両辺と並進ベクトルの歪対称行列$[\boldsymbol{t}]_\times$の積を取ると，左辺が$L_1[\boldsymbol{t}]_\times\boldsymbol{e}_1$，右辺が$L_2[\boldsymbol{t}]_\times\boldsymbol{R}\boldsymbol{e}_2 = L_2\boldsymbol{E}\boldsymbol{e}_2$となるので，

$$L_1[\boldsymbol{t}]_\times\boldsymbol{e}_1 = L_2\boldsymbol{E}\boldsymbol{e}_2$$

となる．よって，$[\boldsymbol{t}]_\times\boldsymbol{e}_1$と$\boldsymbol{E}\boldsymbol{e}_2$の符号が一致する$\boldsymbol{t}$と$\boldsymbol{E}$の組み合わせ 2 組を残す．次に，回転行列$\boldsymbol{R}$を 2 組の$\boldsymbol{t}$と$\boldsymbol{E}$から計算する．$\boldsymbol{R}$は 2 組のどちらも同じ値になる．最後に，エピポーラ面を上から見たとき，$\boldsymbol{t}\times\boldsymbol{e}_1$と$\boldsymbol{e}_1\times\boldsymbol{e}_2$の向きが同じである組み合わせが正しい$\boldsymbol{t}$と$\boldsymbol{R}$とする．まとめると，$\boldsymbol{E}$と$\boldsymbol{t}_1$の符号を決定するアルゴリズムは手順 3.4 のとおりである．

手順 3.4 手順 3.3 に続いて$\boldsymbol{R},\boldsymbol{t}$を決定する
画像 1 の特徴点：p_1，画像 2 の特徴点：p_2

1. $\boldsymbol{t}_1$から歪対称行列$[\boldsymbol{t}]_\times$を作る
2. $\boldsymbol{a} = [\boldsymbol{t}]_\times p_1, \boldsymbol{b} = \boldsymbol{E}p_1$とする
3. すべての観察点について$\boldsymbol{a}$, $\boldsymbol{b}$の内積を求め足し合わせる $sum \mathrel{+}= \boldsymbol{a}^T\cdot\boldsymbol{b}$
4. sumが 0 以上の場合，$\boldsymbol{t}_2 = -t_1, \boldsymbol{E}_1 = \boldsymbol{E}, \boldsymbol{E}_2 = -\boldsymbol{E}$
   それ以外の場合，$\boldsymbol{t}_2 = -t_1, \boldsymbol{E}_1 = -\boldsymbol{E}, \boldsymbol{E}_2 = \boldsymbol{E}$とする

- 回転行列の計算

1. $\boldsymbol{K} = -\boldsymbol{t}_1\boldsymbol{E}_1$とする
2. $\boldsymbol{K}$を特異値分解する $\boldsymbol{K} = (\boldsymbol{U},\boldsymbol{S},\boldsymbol{V})$
3. 回転行列 $\boldsymbol{R}$を求める $\boldsymbol{R} = \boldsymbol{U}\,diag(1,1,\det(\boldsymbol{u}*\boldsymbol{u}^T))\,\boldsymbol{V}^T$

- $\boldsymbol{t}$の符号を決定する

1. $\boldsymbol{a} = \boldsymbol{R}^T\boldsymbol{t}_1\times\boldsymbol{R}^T p_1$
2. $\boldsymbol{b} = \boldsymbol{R}^T p_1\times p_2$とする
5. すべての観察点について$\boldsymbol{a}$, $\boldsymbol{b}$の内積を求め足し合わせる $sum \mathrel{+}= \boldsymbol{a}^T\cdot\boldsymbol{b}$
3. sumが 0 より大きい場合並進ベクトル$\boldsymbol{t} = \boldsymbol{t}_1$
   それ以外の場合$\boldsymbol{t} = \boldsymbol{t}_2$とする

$\boldsymbol{t}$のスケールを決めることはできないため，$\boldsymbol{t}$は単位ベクトルとして求める．

3.1.5　2 画像からの 3 次元位置計算

2 つの特徴点に対応した視線を考え，2 本の直線が最も近くなる点が解であるとする．基準となるカメラをカメラ 1，もう一台のカメラをカメラ 2 とよび，それぞれのカメラにおける空間の点$\boldsymbol{M}$の観察点を$\boldsymbol{p}$，$\boldsymbol{q}$とする．

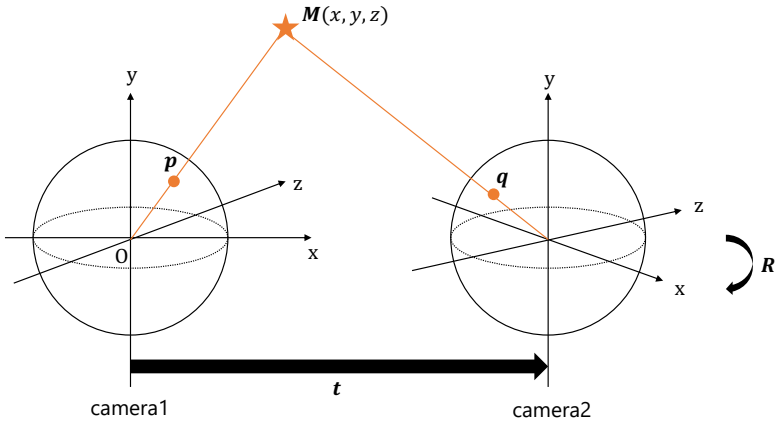

図 3.1　2 つの 360 度カメラで撮影した場合の模式図

図 3.1 において，カメラ 2 はカメラ 1 に対して，x 軸方向に$\boldsymbol{t}$移動して，y 軸回りに$\boldsymbol{R}$回転している．ワールド座標系はカメラ 1 のカメラ座標系と同じであるとする．点$\boldsymbol{M}$の 2 つの観察点$\boldsymbol{p}$，$\boldsymbol{q}$のそれぞれのカメラ座標系における座標値がわかっている場合に，点$\boldsymbol{M}$の 3 次元位置(x, y, z)を求める方法を説明する．図 3.1 において，カメラの中心と観察点を通る直線を引き，2 本の直線が交わるところが点$\boldsymbol{M}$の位置である．実際の 360 度カメラから求めた 2 本の直線は，観察誤差の影響で，3 次元空間上で交わらないことが多いので，2 本の直線が最も近づく 2 点を求め，それらの中点を点$\boldsymbol{M}$の 3 次元座標とする．カメラ 2 が回転運動をしているので，点$\boldsymbol{q}$をワールド座標系へ合わせるために，カメラ 2 に回転の補正を施した後にカメラ 2 からの視線を考える．

2 本の直線上の点をそれぞれ，

$$\boldsymbol{l}_1 = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \alpha \begin{pmatrix} x_{0v} \\ y_{0v} \\ z_{0v} \end{pmatrix}, \quad \boldsymbol{l}_2 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \beta \begin{pmatrix} x_{1v} \\ y_{1v} \\ z_{1v} \end{pmatrix}$$

というベクトル方程式の形で表現する．この 2 点の距離の差が最も小さくなる点を求める．それには，次の式で表される 2 点間のユークリッド距離の 2 乗が最小になるα，βを求めればよい．

$$f(\alpha, \beta) = \left[\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \alpha \begin{pmatrix} x_{0v} \\ y_{0v} \\ z_{0v} \end{pmatrix} - \left\{ \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \beta \begin{pmatrix} x_{1v} \\ y_{1v} \\ z_{1v} \end{pmatrix} \right\} \right]^2$$

上式が最小になる時，$f(\alpha, \beta)$のα，βによる偏微分が 0 であることから，α，βに関する連立方程式を導き，解を求める．ただし，視差量が少ない特徴点に対して，連立方程式の解を計算できない場合がある．そのような特徴点では距離の計算を行わない処理とした．

3.2　バンドル調整

通常の SfM と同じように，復元した 3 次元点座標，カメラの並進量，回転量を調整するためにバンドル調整を行う．本研究での 360 度カメラのモデルでは内部パラメータを考えないので，内部パラメータは調整の対象から外れる．コスト関数は以下のようになる．

$$E(\boldsymbol{D}_1 \dots \boldsymbol{D}_m, \boldsymbol{q}_1 \dots \boldsymbol{q}_n) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left\{\left(u_{ij} - \hat{u}(\boldsymbol{D}_i, \boldsymbol{q}_j)\right)^2 + \left(v_{ij} - \hat{v}(\boldsymbol{D}_i, \boldsymbol{q}_j)\right)^2 + \left(w_{ij} - \hat{w}(\boldsymbol{D}_i, \boldsymbol{q}_j)\right)^2\right\}$$

ここで，$\boldsymbol{D}$は$\boldsymbol{R}, \boldsymbol{t}$からなる推定された外部パラメータ，$\boldsymbol{q}_j$は推定された 3 次元座標，$(u_{ij}, v_{ij}, w_{ij})$は単位球面上の特徴点座標，$(\hat{u}(\boldsymbol{D}_i, \boldsymbol{q}_j), \hat{v}(\boldsymbol{D}_i, \boldsymbol{q}_j), \hat{w}(\boldsymbol{D}_i, \boldsymbol{q}_j))$は外部パラメータ$\boldsymbol{D}$で移動したカメラから撮影した，$\boldsymbol{q}_j$の単位球面上の座標である．実装において，非線形最小二乗法の計算には MATLAB の lsqnonlin 関数を利用した．

3.3 動画からの SfM

最終的に作成したデモプログラムは，MATLAB のサンプルプログラムとして提供されている ORB-SLAM の初期化処理を基にして作成した．そのプログラムでは，入力する動画のフレームを順次処理して，十分にカメラが移動した 2 枚のフレームを見つけ，それを SfM の対象画像としている．1 枚目のフレームは動画の 1 フレーム目とする．2 枚目のフレームは以下の 3 つの条件を満たしたものになる．

1. 特徴点の数が十分な数あること
   特徴点の数が 100 点以上のフレームで処理を続ける
2. 再投影誤差の量が少ない特徴点を残す
   再投影誤差が大きい特徴点は除去する
3. 特徴点の視差量が十分あること
   1 枚目との特徴点の視差を計算する．視差が 2 度以上の点が 50%を超えれば 2 枚目のフレームとする．

3.4 作成したプログラムの説明

作成したプログラムの動作は以下のとおりである．

1. 画像列を用意
2. 2 枚の正距円筒画像から特徴を抽出，対応付け
3. 最初のフレームとの特徴マッチング
4. 特徴点から$\boldsymbol{E}$行列を計算
5. $\boldsymbol{E}$行列からカメラの回転$\boldsymbol{R}$・並進$\boldsymbol{t}$を計算
6. 特徴点の 3 次元位置を計算
7. 復元した 3 次元位置とカメラの回転・並進についてバンドル調整を行う

このうち，画像特徴の検出には MATLAB の関数を用いた．実装したプログラムは，2 枚の 360 度画像から 3 次元復元を行うものである．MATLAB のバージョン R2020a を用いてプログラムを作成した．

第4章 実験と考察

4.1 3DCG 画像を用いた動作検証

3DCG ソフトの Unity で 3 次元シーンを作成し，実装したプログラムが動作することを確認した．3 次元シーンは 3 つの立方体を配置し，特徴点を検出できるようにテクスチャ画像を貼り付けた．作成したシーンの様子を図 4.1 に示す．

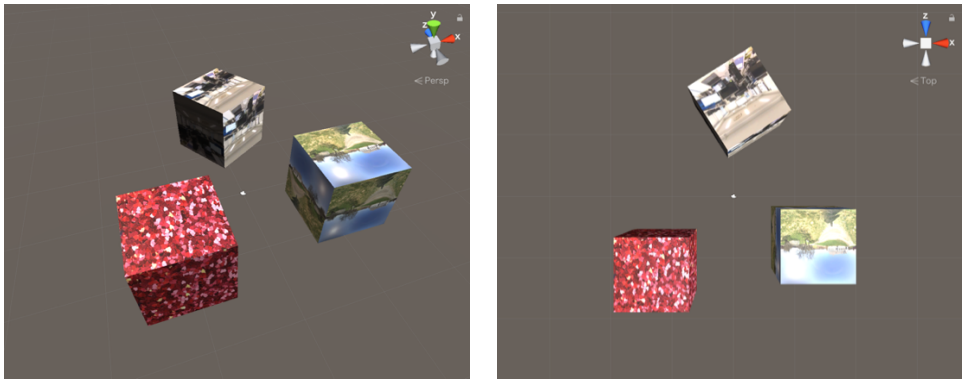

図 4.1 作成した 3 次元シーン
左：斜め上視点からの画像　右：真上視点からの画像

3 次元シーンの原点で基準となる 360 度画像 1 を撮影し，カメラを原点から左方向に移動させ，もう一つの 360 度画像 2 を撮影した．図 4.2 に画像を撮影した位置を示す．

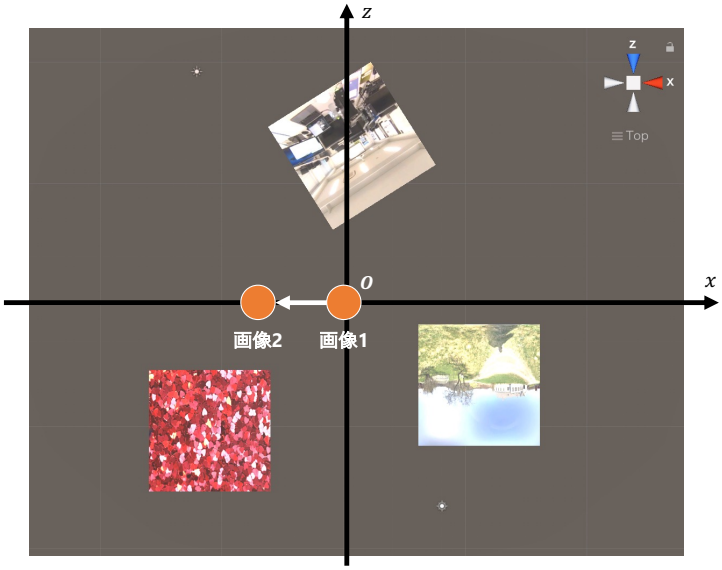

図 4.2 シーン中の画像撮影位置

2 枚の 360 度画像から，作成したプログラムで SfM を行った．3 次元点とカメラ位置の復元結果を図 4.3 に示す．図 4.3 で示すように，カメラから見えている立方体の面の形状が復元されていることがわかる．また，カメラ位置が水平横方向にわずかにずれており，カメラ位置も復元できていることがわかる．

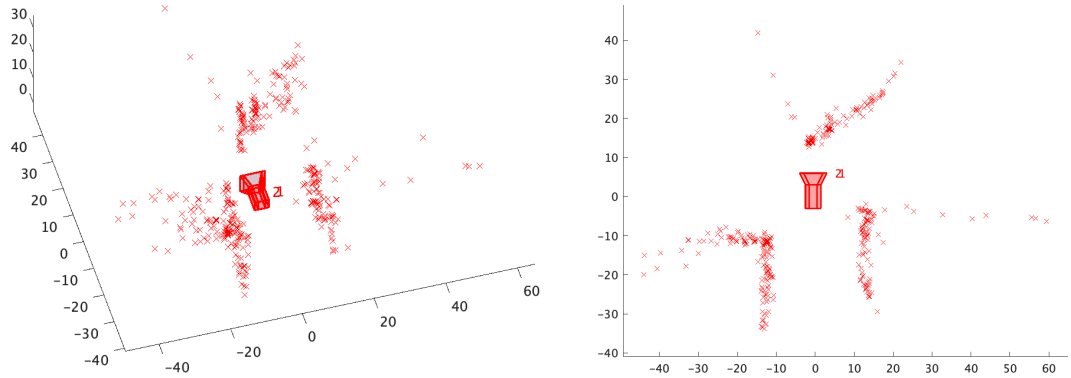

図 4.3 復元された 3 次元点の様子
左：斜め上視点からの復元結果　右：真上視点からの復元結果

4.2 シミュレーションデータでの比較実験

360 度画像を用いた SfM の精度検証を行うために，MATLAB 上でシミュレーションデータを生成し，カメラの回転量と並進量を復元した．そして，提案手法による SfM の復元精度と，ピンホールカメラモデルで撮影した通常画像を用いた SfM の復元精度を比較した．

4.2.1-1 シミュレーションデータの作成方法

シミュレーションデータは，原点から 2000〜5000 の距離に，10 万個の 3 次元点を，一様乱数を用いてランダムに生成した．3 次元点を撮影するカメラは，基準となるカメラ 1 を原点に置き，カメラ 1 から並進，回転運動をさせた位置にカメラ 2 を配置した．この 2 台のカメラで 3 次元点を投影し，SfM する場合についてシミュレーションした．すべての画像で条件が揃うように，復元に用いる観察点の点数は 1000 点に統一した．

4.2.1-2 通常カメラによる投影

通常のカメラで 3 次元点を画像面に投影する場合，内部パラメータと外部パラメータから透視投影変換行列（カメラ行列）を作成し，観察点を計算した．画像サイズは 1024 画素×1024 画素，カメラの画角 90 度になるように内部パラメータを設定した．カメラの内部パラメータを表 4.1 に示す．

表 4.1 内部パラメータの設定

画像サイズ	1024×1024
焦点距離	1023/2
画像中心	((1+1024)/2, (1+1024)/2)
水平画角	90 度

通常画像では 10 万個の 3 次元点のうち，およそ 16000 点が画像面に投影された．このうち，1000 点をランダムに選んで残した．投影される点の座標は小数点以下の有効桁をもつ浮動小数点の値として記録した．

4.2.1-3 360 度カメラによる投影

360 度カメラでの観察点は，3 次元点までの距離を 1 に正規化することで求めた．3 次元点を単位球面上に投影したものを単位球面画像と呼ぶ．360 度カメラでは 10 万個すべての点を画像面に投影することが可能であるが，通常画像と比較しやすいように，単位球面画像に投影される観察点の数を 1000 点にする．そのために，投影されたすべての観察点の中から，ランダムに 1000 点を選びだした．

4.2.1-4 ノイズの付与

画像面に投影した後，それぞれの観察点にノイズを加えた．通常画像に加えたノイズは平均 0，標準偏差 1 の正規分布ノイズである．このノイズを x 軸，y 軸それぞれの座標値に独立に加えた． 標準偏差 1 は画素サイズに相当する．

単位球面画像においては，以下の考え方で，与えるノイズの量を決定した．通常画像の画角が 90 度であることと，単位球の円周が2πであることから，通常画像で撮影している 90 度の範囲は，単位球の円周上で$2\pi/4 = pi/2$の長さになる．$pi/2$の範囲を 1024 画素で撮影すると考えると，1 画素あたりの角度は$N_s = (pi/2)/1024 = 0.00153 \ldots$である．この値が通常画像の 1 画素に対応するものとして，単位球面画像には標準偏差N_sのノイズを与えた．

単位球面画像にノイズを加える方法を述べる．単位球上の観察点において，接平面を考える．接平面を構成する直交する 2 軸に標準偏差N_sのノイズを加えた後，再度単位球面に投影した．ノイズの加え方を図 4.4 に示す．2 軸に同じ量の値を加えることで，点mがm'に移動している．

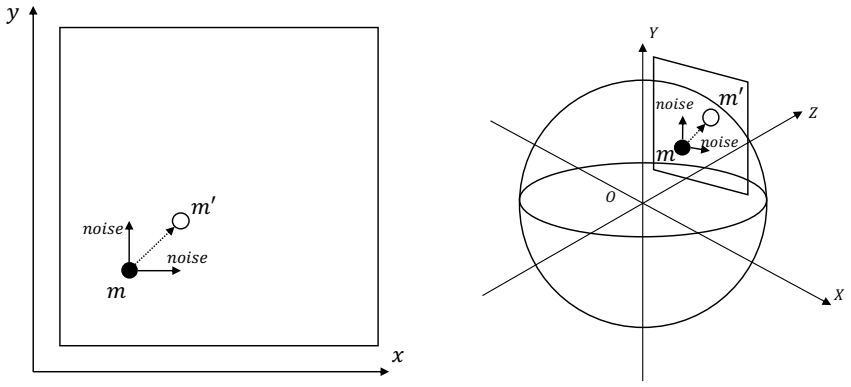

図 4.4 ノイズの加え方
左：通常画像　右：単位球面画像

図 4.5 に投影した画像を可視化したものを示す．左は通常の 2 次元画像であり，右は単位球面の座標に点を投影した単位球面画像である．

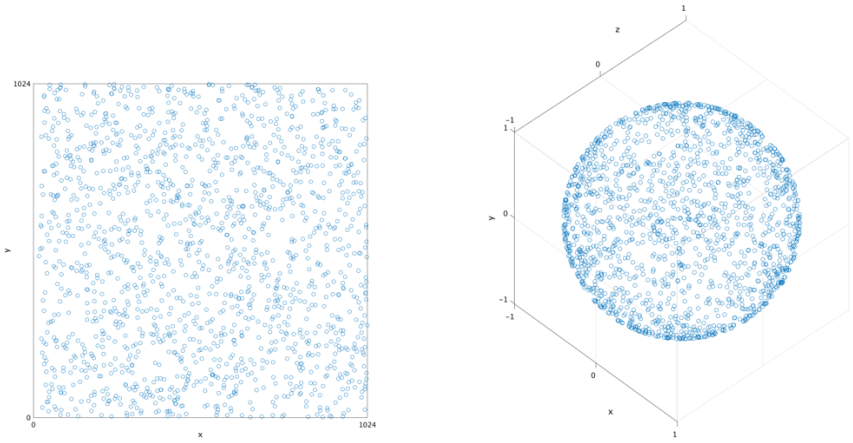

図 4.5 SfM に使った画像
左：通常画像の例　右：単位球面画像の例

4.2.1-5 シミュレーションの方法

上記の方法で作成したシミュレーション画像を用いて SfM を実行する．通常画像については，画像座標から正規化画像座標に変換し，提案手法で並進ベクトルと回転行列を復元した．一方，360 度カメラの場合は，2 つの単位球面画像から提案手法で SfM を行った．

また，もう一つの比較のために追加の単位球面画像を作成した．この画像は，すべての点が投影された単位球面画像から，通常画像と同じ画角に投影されている点を，ランダムに同じ数取り出して作成した．このように作成した単位球面画像からも，本手法で SfM を行い，比較を行った．以下，この単位球面画像を単位球面 1 と呼ぶ．また，既に述べた，全方向の点を写し，観察点の数を通常画像と同じにした単位球面画像を単位球面 2 と呼ぶ．実験に用いる画像の作成順序を図 4.6 に示す．なお，この比較実験では，RANSAC による外れ値の除去やバンドル調整は行なっていない．

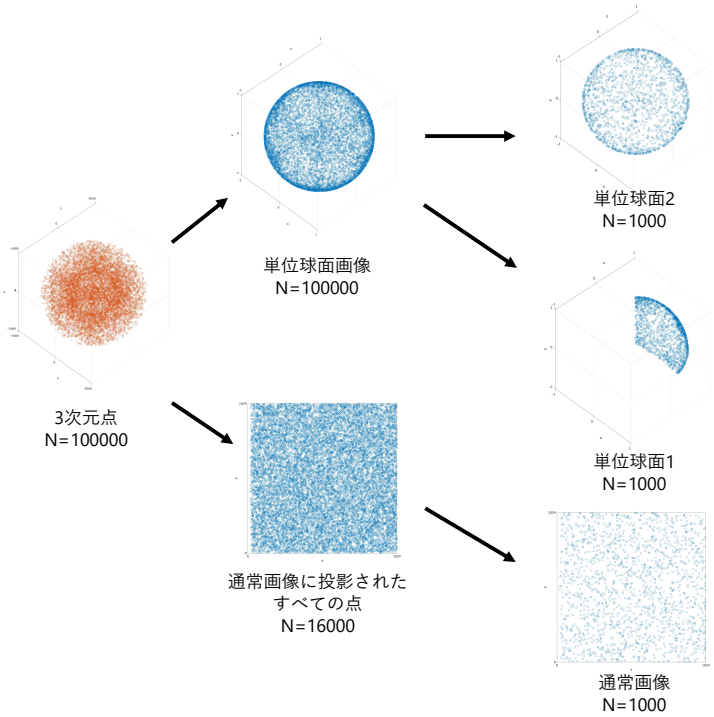

図 4.6 通常画像と単位球面画像を作る流れ（N は点の数）

4.2.2 復元結果の比較方法

通常画像，単位球面 1 と単位球面 2 の 3 画像それぞれで，推定値された並進ベクトルと回転行列についてそれらの真値との誤差を比較する．比較する誤差量として，並進ベクトルでは，ノルムを 1 にした真値と推定値の差をとり，そのノルムを求めた．回転行列はオイラー角に変換し，真値と推定値の 3 つのオイラー角の差をとり，そのノルムを求めた．異なる 3 次元点のデータを生成する試行を 100 回繰り返し，100 回の誤差の平均値と標準偏差を求めた．

4.2.3 カメラの種類・動きと SfM の誤差

カメラが右方向，上方向，前方向に動いたときの誤差の平均と標準偏差を検証した．与えるノイズの標準偏差は 1 に設定した．通常画像と単位球面 1 の画角は，90 度に設定した．基準となるカメラ 1 を原点に置く．カメラ 2 を原点から，①x軸方向に 100 動かした場合，②z軸方向に 100 動かした場合，③y軸方向に 100 動かした場合，3 通りについて，それぞれ 100 回プログラムを実行して，誤差の平均，標準偏差を計算した．結果の誤差の平均を表 4.2，標準偏差を表 4.3 に示す．

表 4.2 各軸方向に動かした場合の誤差の平均

移動量	$\boldsymbol{R}$の誤差の平均			$\boldsymbol{t}$の誤差の平均		
	通常画像	単位球面 1	単位球面 2	通常画像	単位球面 1	単位球面 2
x=100	3.52×10^{-4}	4.45×10^{-4}	2.85×10^{-4}	1.46×10^{-2}	2.13×10^{-2}	4.27×10^{-3}
y=100	3.43×10^{-4}	4.22×10^{-4}	2.67×10^{-4}	1.49×10^{-2}	1.84×10^{-2}	3.95×10^{-3}
z=100	7.21×10^{-4}	6.92×10^{-4}	2.41×10^{-4}	1.61×10^{-2}	1.55×10^{-2}	4.23×10^{-3}

表 4.3 各軸方向に動かした場合の誤差の標準偏差

移動量	$\boldsymbol{R}$の誤差の標準偏差			$\boldsymbol{t}$の誤差の標準偏差		
	通常画像	単位球面 1	単位球面 2	通常画像	単位球面 1	単位球面 2
x=100	1.60×10^{-4}	2.20×10^{-4}	1.12×10^{-4}	8.04×10^{-3}	1.00×10^{-2}	2.20×10^{-3}
y=100	1.41×10^{-4}	2.09×10^{-4}	1.13×10^{-4}	8.83×10^{-3}	1.15×10^{-2}	2.03×10^{-3}
z=100	3.17×10^{-4}	3.39×10^{-4}	1.09×10^{-4}	8.26×10^{-3}	8.45×10^{-3}	2.10×10^{-3}

表 4.2，表 4.3 の結果をまとめてグラフにしたものを図 4.7 と図 4.8 に示す．

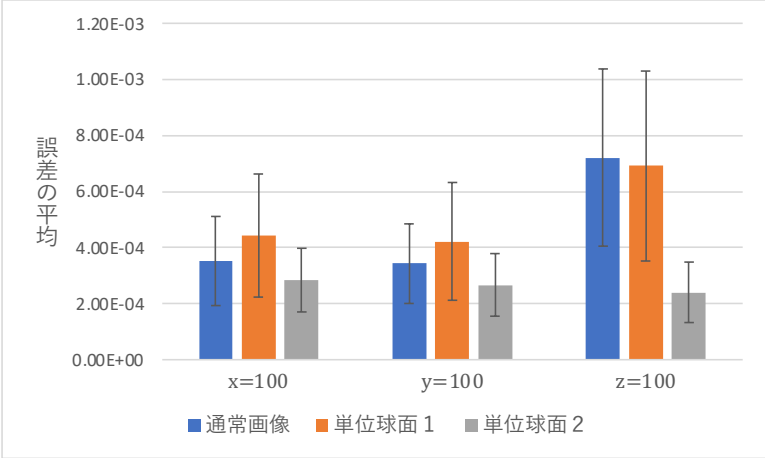

図 4.7 回転行列の誤差の平均値と標準偏差

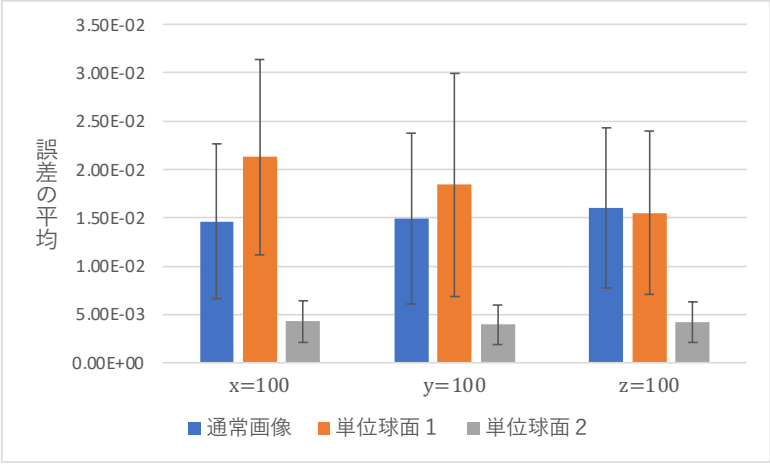

図 4.8 並進ベクトルの誤差の平均値と標準偏差

この実験から次のことがわかる．

① 全方向の点を投影している単位球面 2 を用いた SfM の復元精度が高い．

② 同じ点を投影している通常画像と単位球面 1 を比較すると，$x = 100$と$y = 100$では通常画像の精度が少し高い．$z = 100$ではほとんど同じである．

③ 通常画像と単位球面 1 では，カメラを前方向（z 軸方向）に移動させたとき，右方向・上方向に比べて，$\boldsymbol{R}$の復元精度が悪い．全方向の点を投影した単位球面 2 による SfM ではそのような傾向は顕著でない．

4.2.4 ノイズの量を変化させた場合

ノイズの大きさと SfM の精度を検証するために，正規分布ノイズの標準偏差を 0.5, 1，2 と変化させた．移動量はx軸方向に 100 とした．通常カメラと単位球面 1 の画角は 90 度とした．100 回実行した結果の平均値と標準偏差を計算した．誤差の平均を表 4.4，誤差の標準偏差を表 4.5 に示す．

表 4.4 ノイズ量を変化させた場合の誤差の平均

SD	Rの誤差の平均			tの誤差の平均		
	通常画像	単位球面 1	単位球面 2	通常画像	単位球面 1	単位球面 2
0.5	1.73×10^{-4}	2.23×10^{-4}	1.38×10^{-4}	7.17×10^{-3}	8.10×10^{-3}	2.08×10^{-3}
1	3.52×10^{-4}	4.45×10^{-4}	2.85×10^{-4}	1.46×10^{-2}	2.13×10^{-2}	4.27×10^{-3}
2	7.79×10^{-4}	9.54×10^{-4}	5.35×10^{-4}	4.32×10^{-2}	7.08×10^{-2}	8.81×10^{-3}

表 4.5 ノイズ量を変化させた場合の誤差の標準偏差

SD	Rの誤差の標準偏差			tの誤差の標準偏差		
	通常画像	単位球面 1	単位球面 2	通常画像	単位球面 1	単位球面 2
0.5	6.65×10^{-5}	1.05×10^{-4}	5.51×10^{-5}	3.77×10^{-3}	4.04×10^{-3}	1.08×10^{-3}
1	1.60×10^{-4}	2.20×10^{-4}	1.12×10^{-4}	8.04×10^{-3}	1.00×10^{-2}	2.20×10^{-3}
2	2.83×10^{-4}	4.70×10^{-4}	2.43×10^{-4}	2.84×10^{-2}	4.26×10^{-2}	4.98×10^{-3}

結果をグラフにしたものを図 4.9 と図 4.10 に示す．

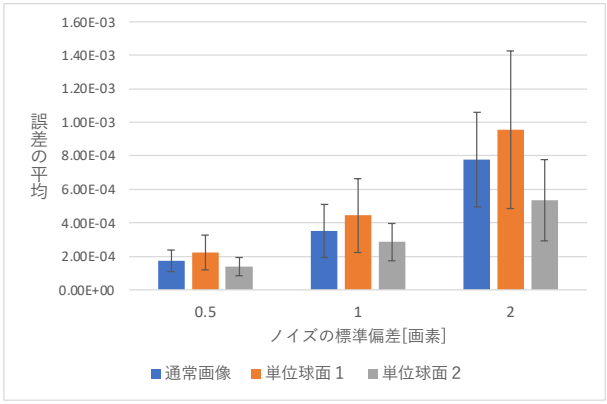

図 4.9 ノイズ量を変化させた場合の$\boldsymbol{R}$の誤差の平均と標準偏差

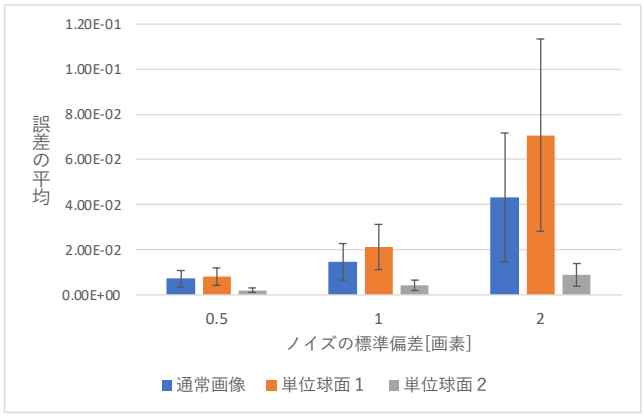

図 4.10 ノイズ量を変化させた場合の$\boldsymbol{t}$の誤差の平均と標準偏差

図 4.9 から，すべての場合において，ノイズの量が増えるほど推定誤差も増えていることがわかる．また，図 4.10 の場合は，通常画像と単位球面の誤差が増えているのに対して，単位球面 2 の誤差があまり増えていないことがわかる．

4.2.5 シミュレーションデータを用いた実験についての考察

360 度画像の単位球面 2 を用いた SfM の誤差は，通常画像の誤差より小さいことがわかった．また，観察点のノイズが増えた場合でも，並進量の誤差の増加量が小さいことがわかった．4.2.3 において，通常画像と視野角を同じにした単位球面 1 では，誤差が通常画像と同じか，悪くなっていた．このことから，360 度画像では全方向の点を使って$\boldsymbol{E}$行列を計算できることが精度向上の要因であると考える．

単位球面 2 において，カメラを右，上方向に動かした場合と前方向に動かした場合で回転の復元結果に差が生じなかった．これは，360 度カメラで観察される点の動きに方向性がないためと考えられる．通常画像では，カメラが前方向に動いた場合，観察点の移動量が比較的小さくなる．一方，360 度カメラでは，観察点の移動量が比較的大きいカメラ横方向の観察点も計算に利用することができる．このことが， 360 度カメラの回転量の復元誤差が移動方向によらず，一定になる原因と考える．

4.3 実写画像を用いた動作検証

4.3.1 実験内容

作成したプログラムで実写 360 度画像から立体構造の復元を行った．また，通常画像と MATLAB 組み込みの関数を用いて，同じ環境を復元し，比較を行った．

360 度画像の撮影にはリコー社の RICHO THETA S を用いた．以下，このカメラを 360 度カメラと呼ぶ．図 4.11 に用いた 360 度カメラを示す．

図 4.11 RICHO THETA S

THETA は撮影画像を正距円筒画像として保存する．THETA の仕様を表 4.6 に示す．

表 4.6RICHO THETA S の仕様

静止画解像度	5376×2688
撮像素子サイズ	1/2.3 CMOS（×2）
レンズ F 値	F2.0

通常画像は Apple 社のスマートフォン iPhone11 Pro Max の超広角カメラを用いて撮影した．以下，このカメラを通常カメラと呼ぶ．表 4.7 に通常カメラの仕様を示す．

表 4.7 通常カメラの仕様

静止画解像度	4032×3024
撮像素子サイズ	1/3.8
レンズ F 値	F2.4
焦点距離	1.54mm 13mm(35mm 換算)
視野角	対角 114° 水平 102° 垂直 86°

SfM に用いる通常カメラの内部パラメータは，MATLAB のカメラキャリブレーターによって取得した．

カメラの移動量を正確に管理するために，移動撮影ステージを用いた．図 4.12 に移動ステージの様子を示す．

図 4.12 使用した移動撮影ステージ

また，表 4.8 に移動ステージの仕様を示す．

表 4.8 使用した移動撮影ステージの仕様

型番	中央精機社 ALS-220-C2P
移動量	±100mm
分解能	0.004mm
位置決め精度	0.02mm
繰り返し精度	0.002mm

使用した移動ステージは横方向の移動を行う X ステージと上下方向の移動を行う Z ステージに分かれている．本実験では，横方向に移動する X ステージを用いた．Z ステージは最上部に固定して実験を行った．

画像の撮影は，以下のように行った．まず，基準となる画像 1 を撮影する．画像 1 の位置から右方向に 100mm カメラを移動させ，画像 2 を撮影した．撮影した画像を図 4.13 に示す．

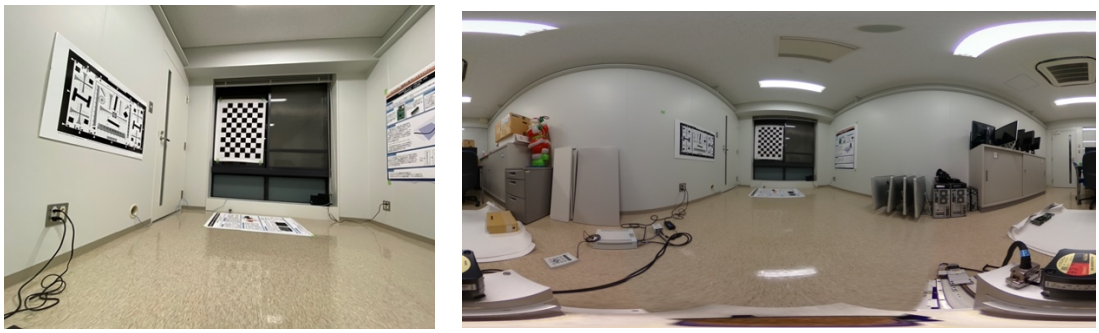

図 4.13 撮影画像の 1 枚

左：通常カメラ　右：360 度カメラ

画像の撮影は室内で行った．壁にポスターなどを貼り付けることで，特徴点の検出ができるようにした．部屋の平面図と画像を撮影したカメラの位置を図 4.14 に示す．メジャーで計測した部屋の長辺は 6350mm，短辺は 2850mm，高さは 2650mm だった．

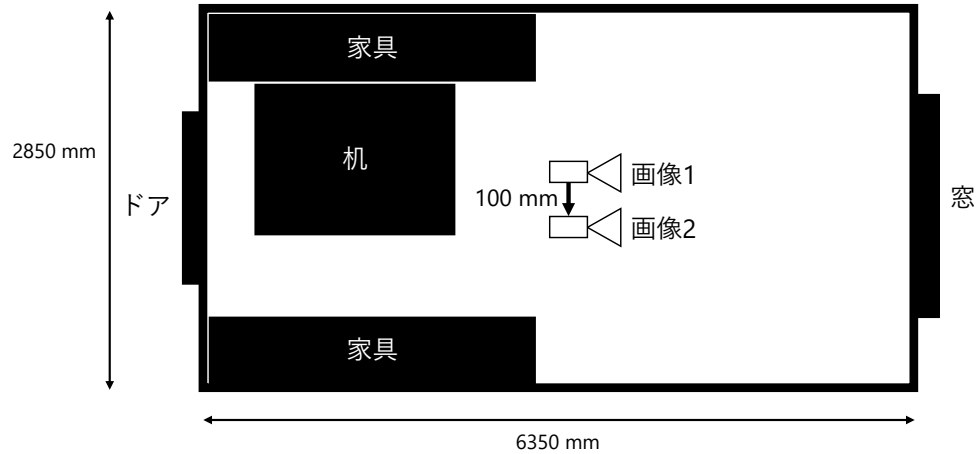

図 4.14 部屋の平面図とカメラ配置

特徴点の検出数は 10000 点に設定した．また，360 度カメラで撮影した画像には移動ステージが写り込んでいる．加えて上下部分は像の歪みが大きい．この領域から特徴検出を行わないように，画像左上を原点として縦方向の特徴検出範囲を$y = 300$~1936に設定した．

本実験では，通常画像の復元に MATLAB の組み込み関数を用いた．$\boldsymbol{E}$行列の計算には，2 画像の対応する観察点から$\boldsymbol{E}$行列を求める estimateEssentialMatrix 関数を，カメラ移動量の復元には，$\boldsymbol{E}$行列から並進ベクトルと回転行列を求める relativeCameraPose 関数を用いた．

特徴点マッチングを行う際，明らかに誤った対応を削除するため，対応点の距離にしきい値を設定した．通常画像は 500 画素，360 度画像は 1000 画素をしきい値として，しきい値より離れている対応点を削除した．

復元結果をわかりやすくするため，特徴検出器で得られる特徴点に加えて，チェッカーボードの点を利用する．写り込んでいるチェッカーボードを，MATLAB のチェッカーボードを検出する関数 detectCheckerboardPoints で検出し，特徴検出器で検出された特徴点に加えた．

3 次元点を復元した結果，復元された点に外れ値が存在した．明らかな外れ値は目視で確認して削除した．また，3 次元点のスケールを決めるためにカメラの移動量を利用した．カメラを真横方向に 100mm 動かしているが，カメラを据え付ける際にカメラのx軸と移動方向にずれが発生する．本実験では，カメラがx軸方向に 100mm 動いたと仮定してスケールを決めた．

4.3.2 実験結果

図 4.15 に通常カメラの復元結果を，斜め上からの視点と真上からの視点で表示したものを示す．

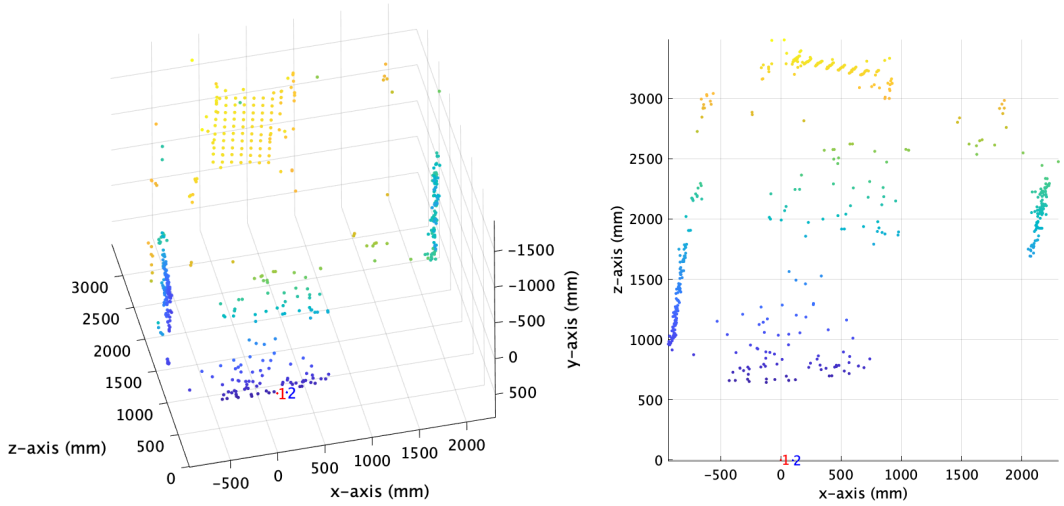

図 4.15 通常カメラの復元結果
左：斜め上からの視点　右：真上からの視点

図 4.15 左を見ると，壁に貼り付けたチェッカーボードが復元されていることがわかる．図 4.15 右では，部屋の 3 つの壁に貼り付けたポスターなどが復元されていることがわかる．また，復元された短辺の距離が約 2800mm となっており，実際の距離と近いことがわかる．3 つの面が傾いているのは，カメラを据え付ける際のずれによるものである．また，画像を撮影したカメラ位置が，原点と右に動いた位置の 2 箇所にプロットされている．

図 4.16 に 360 度カメラの復元結果を，斜め上からの視点と真上からの視点で表示したものを示す．また，図 4.17 に真横からの視点で表示したものを示す．

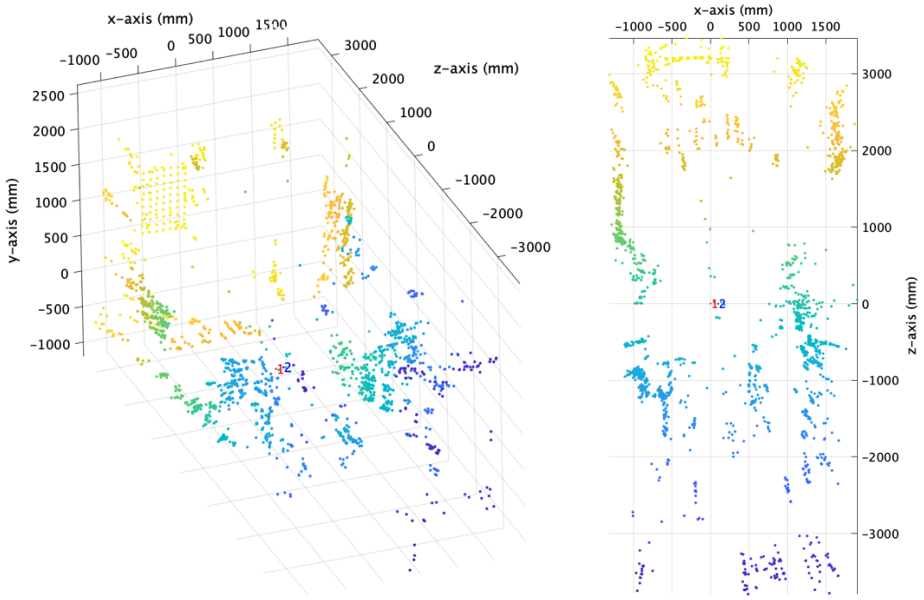

図 4.16 360 度カメラの復元結果
左：斜め上からの視点　右：真上からの視点

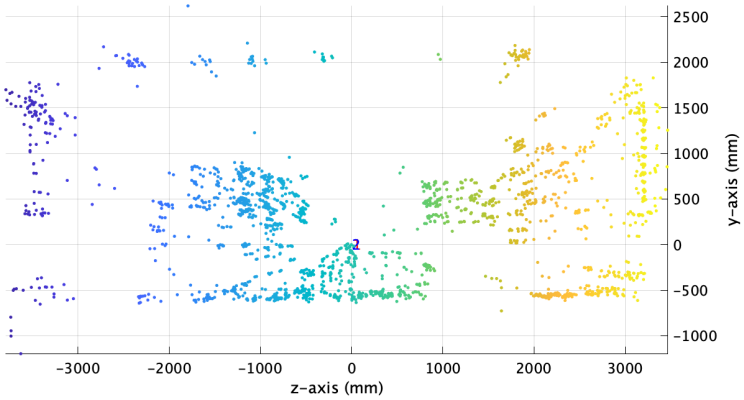

図 4.17 360 度カメラの復元結果（真横からの視点）

図 4.16 左を見ると，上部にチェッカーボードが復元されていることがわかる．図 4.16 右から，カメラを中心として，すべての方向に点が復元されていることがわかる．また，図 4.16 右と図 4.17 から，部屋の形状が復元されていることがわかる．復元された部屋のサイズは短辺が約 2800mm，長辺が約 6400mm，高さが約 2600mm となっており，実際のサイズと近いことがわかる．

復元された回転量を表 4.9 に，並進量を表 4.10 に示す．計算の過程に RANSAC など乱数を用いるアルゴリズムがあり，同じ入力画像を用いて SfM を行っても復元結果が少し異なるため，10 回実行した平均値を示す．また，回転量は回転行列をオイラー角に変換して示す．

表 4.9 回転量の復元結果

	通常カメラ			360度カメラ		
回転軸	x	y	z	x	y	z
平均	-9.62×10^{-4}	-4.81×10^{-4}	-1.02×10^{-3}	-5.83×10^{-4}	5.60×10^{-4}	-2.13×10^{-4}
標準偏差	2.20×10^{-4}	6.52×10^{-4}	2.57×10^{-4}	1.14×10^{-6}	8.17×10^{-6}	2.60×10^{-6}

表 4.10 並進量の復元結果

	通常カメラ			360度カメラ		
移動軸	x	y	z	x	y	z
平均(mm)	100	3.991	-3.513	100	3.338	-1.648
標準偏差	0	0.444	0.293	0	0.007	0.006

表 4.9 と表 4.10 の結果をグラフにしたものを図 4.18 に示す．なお，図 4.18 右ではスケールの決定に用いたx軸の値を省いた．

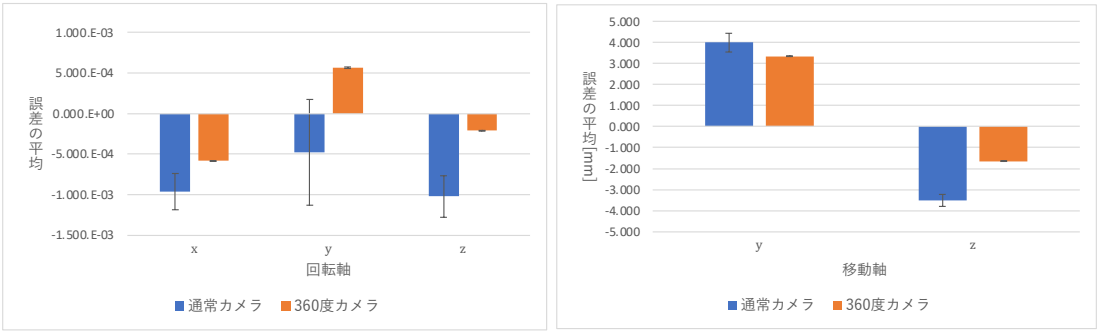

図 4.18 復元結果の比較
左：回転量　右：並進量

カメラの固定を目測で行ったため，カメラの並進方向とx軸はずれている．カメラの正確な移動方向はわからないため，並進量の評価は行わない．回転量の真値は 0 であるから，復元量は 0 に近いほど誤差が少ない．表 4.9 と図 4.21 左を見ると，y の値以外は 360 度カメラのほうが小さくなっている．これは，4.2 節のシミュレーションの結果と一致するものである．

4.3.3 実写画像を用いた実験についての考察

360 度画像を使った復元の場合，通常画像を使った場合と比較して，広い範囲を一度に復元できている．実際の部屋のサイズと比較すると，実際のサイズと近い復元結果になった．

実写画像を用いた復元は，通常カメラの場合より，360 度カメラを用いた場合，回転の誤差が小さいという結果になった．また，2 枚の画像からすべての方向の点を復元できている．同じ広さの空間を復元するとき，通常カメラよりも少ない画像枚数で復元できると考える．

4.4 今後の課題

通常画像の場合と比較して，360 度画像を用いた SfM の精度が良くなる要因は全方向の点を使うことができることであると考えた．精度に影響するその他の要因をより明確にするのが今後の課題である．

本研究で作成したプログラムは 2 枚の画像から SfM を行うプログラムである．既存の SfM のライブラリでは，複数枚の画像から 3 次元構造とカメラ位置を復元することができる．複数枚の 360 度画像を処理できるプログラムを作成するのが今後の目標である．

第5章 結論

本研究では，360 度画像を用いた SfM の精度を通常画像の場合と比較し，検討を行った．はじめに 360 度画像を単位球面モデルでモデル化した場合の SfM の方法を示し．360 度画像を用いた SfM を行うプログラムを作成した．シミュレーションデータや実写画像を用いた実験では，通常画像を用いた場合と比べて，復元精度が向上していることが確認された．精度が向上した理由として，360 度画像では全ての方向の特徴点を利用できることがあると考える．通常画像と 360 度画像で，SfM の精度に影響する要因をより明確にするのが今後の課題である．

謝辞

本研究の遂行にあたり，粘り強く懇切丁寧に指導してくださった蚊野浩教授に深く感謝いたします．また，普段の会話で気分転換に付き合ってくださり，ときに研究について気づきを与えてくださった蚊野研究室のメンバー，大学院の同期の皆様に感謝いたします．最後に，私の学生生活を精神面，金銭面で支えてくださり，２年間の貴重な大学院での研究機会を与えてくださった家族に深く感謝いたします．

参考文献

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz and R. Szeliski, "Building Rome in a Day," IEEE 12th International Conference on Computer Vision, Sept. 29 2009-Oct. 2 2009, pp.72–79., 2009.

[2] 中村恭之, 小枝正直 , 上田悦子, OpenCV によるコンピュータビジョン・機械学習入門, 講談社, 2017.

[3] R. Hartley and A. Zisserman, Multiple View Geometry in computer vision, Cambridge University Press, 2004.

[4] 金谷健一, 菅谷保之 , 金澤靖, 3 次元コンピュータビジョン計算ハンドブック, 森北出版, 2016.

[5] "Equirectangular projection - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Equirectangular_projection. [Accessed 14 1 2022].

[6] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces,," in Proc. IEEE ACM Int. Symp. Mixed Augmented Reality, Nara, Japan, Nov. 2007, pp. 225–234., 2007.

[7] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," IEEE Trans. Robot., vol. 31, no. 5, pp. 1147–1163, 2015.

[8] J. Engel, T. Schops and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in Proc. Eur. Conf. Comput. Vision, Zurich, Switzer- land, Sep. 2014, pp. 834–849., 2014.

[9] A. Torii and et.al., Two- and Three- View Geometry for Spherical Cameras, Proceedings of the 6th workshop on omnidirectional vision, camera networks and non-classical camera, pp.81-88, 2005.

付録

● 特異値分解

コンピュータビジョンにおいてよく使われる行列の分解に，特異値分解がある．特異値分解とは，ある行列$\boldsymbol{A}_{3\times3}$を以下の 3 つの行列の積に分解することである．

$$\boldsymbol{A}_{3\times3} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$$

ここで，$\boldsymbol{U},\boldsymbol{V}$は正規直交行列であり，$\boldsymbol{U},\boldsymbol{V}$の列ベクトルを，それぞれ左特異ベクトル，右特異ベクトルと呼ぶ．また，$\boldsymbol{S}$は以下のような対角行列である．

$$\boldsymbol{S} = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}$$

$\boldsymbol{S}$の対角成分σ_1~σ_3は行列$\boldsymbol{A}$の特異値であり$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$である．

ここでは 3×3 行列の場合を説明したが，一般の m×n 行列にも特異値分解を適用することができる．

● 固有値分解

固有値分解とは，ある正方行列$\boldsymbol{B}$を以下の 3 つの行列の積に分解することである．

$$\boldsymbol{B}_{n\times n} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$$

ここで，$\boldsymbol{U},\boldsymbol{\Lambda}$はそれぞれ以下のような行列である．

$$\boldsymbol{U} = (\boldsymbol{u}_1 \quad \cdots \quad \boldsymbol{u}_n), \boldsymbol{\Lambda} = diag(\lambda_1 \quad \cdots \quad \lambda_n)$$

$\boldsymbol{U}$は$\boldsymbol{B}$の固有ベクトルを列に並べた行列であり，$\boldsymbol{\Lambda}$は$\boldsymbol{B}$の固有値を対角成分にもつ対角行列である．

● 作成したプログラム

本研究で作成した主なプログラムのリストを示す．

プログラム名	プログラム内容
myEquiToSpherePoint.m	正距円筒画像上の座標を単位球面上の座標に変換するプログラム．
myEightPointAlgo.m	８点アルゴリズムで$\boldsymbol{E}$行列を求めるプログラム．
myDecomposeEssentialMatrix.m	$\boldsymbol{E}$行列を回転行列$\boldsymbol{R}$と並進ベクトル$\boldsymbol{t}$に分解するプログラム．
calcPos3d_leastsq.m	$\boldsymbol{R},\boldsymbol{t}$から 2 つの単位球面上の観察点の 3 次元位置を計算するプログラム．
bundleAdjustmentFor360.m	$\boldsymbol{R},\boldsymbol{t}$と 3 次元点座標をバンドル調整で補正するプログラム．