Vol.2017-MBL-82 No.21
Vol.2017-UBI-53 No.21
2017/3/9

線分マッチングに基づく位置推定アルゴリズムの GPU 並列演算による高速化

徳永悠希[†1]，高橋淳二[†1]，兼子朋也[†1]，花見唯[†1]，戸辺義人[†1]

概要：本稿では，画像を用いたスマートフォンやロボット向けの位置姿勢推定の手法を提案する．カメラで撮影した画像の線分と事前に CAD データから生成した 3D ワイヤフレームマップの線分情報をマッチングすることで位置推定を行う．マッチングの際には，3D ワイヤフレームマップから大量の様々な視点の 2D 画像データベースを作成し，入力画像をこれと比較する．マッチング処理を高速化するために，GPU による並列化を行った．12000 枚のデータベースを用いた実証実験では，CPU 処理で 85 秒かかっていたが GPU 処理で 0.4 秒と処理時間を 200 分の 1 にできた．

キーワード：位置推定，線分特徴，2D-3D マッチング，GPU 並列演算

Parallel Computing with GPU for Speeding up of Localization Algorithm based on line-segment matching

Yuki TOKINAGA[†1], Junji TAKAHASHI[†1], Tomoya KANEKO[†1], Yui HANAMI[†1], and Yoshito TOBE[†1]

Abstract: This manuscript proposes a novel localization method based on client-server model for an indoor mobile robot and a mobile device such as smartphone. The localization process proceeds as matching calculation between a client sensing image and an image in a 2D-image-database. The 2D-database consists of massive amount of images with various viewpoints generated from 3D wireframe map. In order to speed up the matching process we adopt parallel computing technique using a GPU board. In an experiment using a database that has 12000 number of images, the processing time of CPU and GPU were 85 seconds and 0.4 seconds, respectively. The GPU computing contributed for 200 times of speeding up.

Keywords: localization, line-segment-feature, 2D-3D matching, parallel computing with GPU

1. はじめに

近年では，スマートフォンを始めとしたセンサを搭載した携帯機器が多く開発され広く普及している．また，ロボット技術の発達に伴い，人間と居住空間を共にする自律移動ロボットの導入が期待されている．従来のロボットといえば，工場のような限定された環境で動作する産業用ロボットが主体であったが，人間の居住空間のような複雑な環境で，ロボットが安全に自律的な活動を行うためには， 高精度な位置姿勢推定が必要不可欠である．

屋内環境においては，環境側に Wi-Fi アンテナや Bluetooth，赤外線センサ，マーカ，iBeacon などを設置し，それらの電波強度の分布から位置推定を行う環境構造化が有効である．しかし，機器の設置，整備に伴う人的・経済的コストが大きくなりやすく実用的ではない．

本研究では，スマートフォンやロボットなどのクライアントから得たセンサ情報とサーバ上に事前に用意したマップを用いることで，高精度かつ低演算コストな 6 自由度の位置姿勢推定技術の確立を目的とする．事前マップは様々なセンサ入力にも普遍的に対応できるよう構造物線分をランドマークとする．本手法には，環境側にセンサ・ビーコンなどの設置をする必要がないこと，また位置解決における演算処理をサーバ側で実行するクライアント側での演算負荷が小さい，という利点がある．

本論の構成は以下である．2 章にて関連研究との違いについて述べる．3 章にて提案するクライアントサーバモデルを用いた位置推定システムの全体像と線分マッチングによる位置推定アルゴリズム，GPU 並列化によるアルゴリズムの高速化について述べる．4 章にて位置推定システムの評価実験として，L 字型ブロックを用いての精度実験，廊下における精度実験，そして CPU と GPU の位置推定アルゴリズムの演算時間の比較について述べる．5 章にて本稿をまとめる．

2. 関連研究

画像を用いた位置推定の研究では，画像の局所特徴量を用いたものが多く研究されてきた．古くから知られている代表的な特徴量には，Lowe の Scale Invariant Feature Transform (SIFT)や Herbert Bay らの Speed-Up Robust Features (SURF)がある[1] [2]．近年では，Alcantarilla らの KAZE や

†1 青山学院大学
Aoyama Gakuin University

Vol.2017-MBL-82 No.21
Vol.2017-UBI-53 No.21
2017/3/9

そのロバスト性の向上と高速化がなされた AKAZE 特徴がよく用いられている[3] [4]．しかし，局所特徴量はテクスチャのない場面では検出しづらいという問題がある．また，環境の変化に弱いため長期の利用には向かないという欠点がある．そこでこれらの局所特徴量に変わりうるものが線分情報である．局所特徴量が検出されづらい環境でも，線分が多く検出されやすく，特に人工物の多い環境ではそれが顕著である．また環境の変化にロバスト(頑強)であるため，安定性がある．

廣瀬らは線分検出器として Line Segment Detector (LSD) を用いることで 2 次元線分と 3 次元線分の対応付けを安定化させ，実時間での SLAM を行なっている[5] [6]．SLAM は，一つ前のシーンのデータを用いて自己位置推定と地図作成を繰り返すため，誤差が蓄積し，増大していく．事前にマップを構築する必要がないので，未知環境においてもロボットのナビゲーションに利用することができるが，長距離の運用では誤差の蓄積が増加することや，端末の演算負荷が大きくなるという問題がある．

自己位置推定の研究は「局所的位置推定」と「大域的位置推定」の二つに大別できる．ここで，局所的位置推定とは初期値に対する相対的な移動量として自己位置推定を行うことである. 現在研究されている位置推定技術の多くは，この局所的位置推定に分類される．そこで本研究では事前マップを用いることで大域的な位置推定を行う．

風間らは撮影位置が既知な画像群から構成された環境地図から画像検索を行うことで自己位置推定を行う手法を提案した[7]．Tim らは事前に LiDAR データから生成した疎な点群マップを用意し，単眼カメラの位置姿勢を 6 自由度で推定している[8]．彼らの手法は，幾何学的情報のみに依存する. 3 次元幾何情報を直接マッチングすることで, GPU による高価な画像レンダリングの必要性を回避し，十分な精度とフレームレートを示している. Gawel らは LiDAR で取得した点群マップとビジョンセンサから得た疎なキービジョンマップとを構造記述子を用いてマッチングする手法を提案している[9].

これらの手法では事前マップの構築に多大なコストが必要となる．また，事前マップはセンサの計測データに基づいて生成されるためマップの正確さはセンサの精度と計測誤差に依存する．また，計測によく用いられている LiDAR センサそのものは極めて高価である．そこで，本研究では 3D CAD による事前マップの作成を行う．CAD データであれば実環境での測位が不要であり，建造物の設計の際には必ず用意されている．そのため，人的・経済的コストをかけず，センサなどで計測するよりも安定したデータが得られる．

3. U-Map: クライアントサーバモデルに基づく位置推定インフラストラクチャシステム

3.1 システムの全体像

本研究では，クライアントサーバモデルによる位置推定インフラストラクチャシステム（U-Map: Universal-Map）を構築する．サーバで事前マップの管理と位置解決演算を行い，クライアントでの演算負荷を小さくできるという特徴がある．また，U-Map を利用できるクライアントは，RGB カメラの他に, LiDAR や KINECT などの様々なセンサを搭載したデバイスであり，様々なセンサ入力に普遍的に共通する特徴として，建物の構造物線分をランドマークとしてマップを構成しているという特徴がある．

サーバでは構造物線分として CAD モデルから生成した 3D ワイヤフレームモデルをマップとして管理する. クライアントがクエリとしてセンサデータをアップロードすると, サーバでは 3D ワイヤフレームモデルとセンサデータとのマッチングを行い，最もマッチング率が高い箇所をそのセンサデータの取得位置座標として出力し, 位置推定を行う. マッチングの手法はセンサデータそれぞれの形式にそれぞれによって異なる．本稿では，以下，クライアントのセンサが RGB カメラであることを想定し議論を進める．

3.2 2D-3D マッチングによる位置推定

クライアントのセンサが RGB カメラである場合，センサデータは 2 次元画像である．従って，3 次元の位置推定を考えると不良設定問題である．また，サーバで管理されているマップは 3D ワイヤフレームであるためそのままではマッチングできない．そこで本研究では以下の手順により 2D-D3 マッチングによる位置推定を行う．まず，3D ワイヤフレームから，クライアントと同じ画角，画像サイズを持つ任意始点画像を生成して 2D 画像データベース : 2D-DB を作成する．次に，入力された 2D 画像を 2D-DB 内の画像と比較し類似度を計算する．類似度が最も高くなった画像をベストマッチ画像 i_{best} と定める．i_{best} には，画像生成時の視点座標（x_e, y_e, z_e）と注視点座標（x_c, y_c, z_c）が対応しており，これが入力データ取得時のクライアントデバイの位置・姿勢となる．位置解決プロセス全体のアルゴリズムフローを図 1 に示す．次節以降で，2D-DB の生成法，線分を用いたマッチング手法，GPU 並列演算によるマッチング処理の高速化について述べる．

3.3 2D-DB: 2D 画像データベース

3D ワイヤフレームマップは, ASCII フォーマットの PLY ファイル (*.ply) にて管理している. PLY 形式は, PCL (Point Cloud Library) を始め多くの研究者や団体によって利用されている 3 次元データ表現形式であり，3 次元空間上の頂点，線分，面とそれぞれの色属性や，ユーザ定義による属

Vol.2017-MBL-82 No.21
Vol.2017-UBI-53 No.21
2017/3/9

性を保持することができる．一般的な 3 次元 CAD ソフトのほとんどが PLY 形式ファイルの出力をサポートしている．本研究では，CAD ソフトとして Rhinoceros 5.0 を利用した．今回作成した 3D ワイヤフレームの CAD ソフト上での表示画面を図 2 に示す．

任意視点の 2D 画像の生成には OpenGL をベースとしたプログラムを作成した．プログラムでは，3D マップが表現された PLY ファイルを読み込み，指定した視点・注視点・画角・画像サイズを持つ画像を生成し保存する．3D ワイヤフレームを元として生成するため 2D 画像は線分のみの画像となる．縦幅 10[m]，横幅 5.0[m]，高さ幅 1.0 [m]の範囲で，0.02 [m]の刻み幅で視点を指定した場合，2D-DB は約 10 万枚程度の画像ファイルとなる．

3.4 入力画像の前処理（キャリブレーション・線分検出・膨張処理）

クライアントデバイスの RGB カメラで写真を撮影した場合，レンズ収差による歪みが発生する．この歪みは OpenGL をもとに生成した 2D-DB の画像には存在しないものであるためそのままではうまくマッチングできない．そこで，Zhang の手法によるキャリブレーションを行う．

入力画像にキャリブレーションを適用し歪みを補正した後，線分検出を行う．線分検出には Gioi らの Line Segment Detector (LSD)¥cite{LSD}を用いる．LSD は画素ごとに周波数方向を算出し，近傍画素でほぼ同一方向のものをグループに加え，矩形領域に近似して線分検出を行う．図¥ref{fig:LSD}に LSD による線分検出結果の一例を示す．線分検出では，グレースケール画像から輝度が急激に変化している部分を線分(輪郭)として検出する．しかし，RGB 画像であってもグレースケール画像に変換するため，画像によっては検出が上手くいかない場合がある．そこで，RGB(赤，緑，青)の要素ごとに線分検出を行い，それらを統合することで線分検出精度を向上させる．図 3 に Canny 法と LSD 法それぞれでの線分検出結果を示す．LSD 法はより細かい線分を検出できていることが分かる．

カメラキャリブレーションを適用してもレンズディストーションを完全に補正することはできない．また，3D ワイヤフレームマップから生成するデータベース画像は離散的なものにしかならない．そのため，カメラ画像と完全に一致することがなく，最適解近傍の画像であってもマッチング率が極端に低くなってしまうという問題がある．そこで，カメラ画像の線分に膨張処理を行う．線分を膨張させることで，補正しきれないカメラディストーションに対応し，離散的なデータベースを補完することができる．また，データベースの刻み幅を極限まで小さくする必要がないた

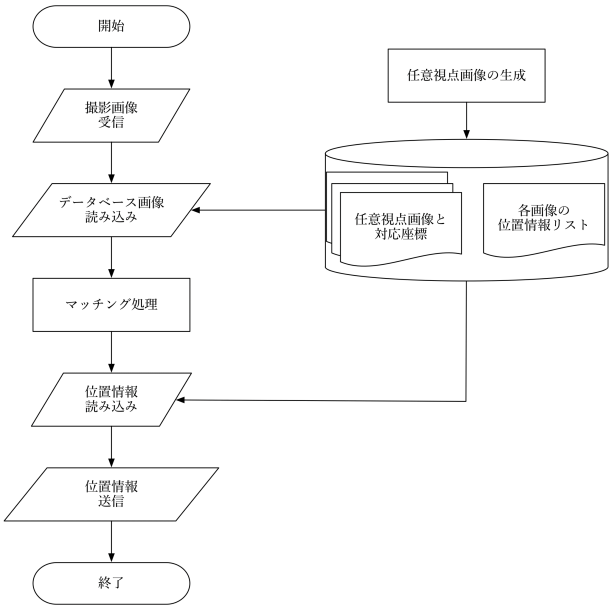

図 1 位置解決プロセス全体のアルゴリズムフロー

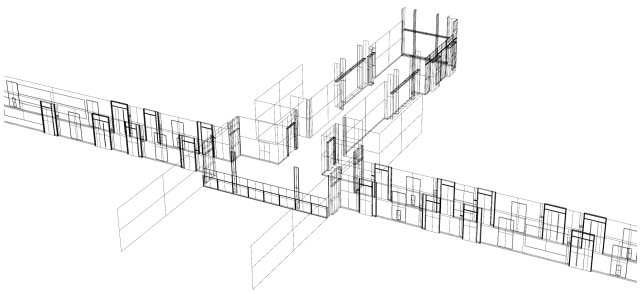

図 2 3D ワイヤフレームマップの CAD 画面表示
（青山学院大学相模原キャンパス O 棟 5 階廊下）

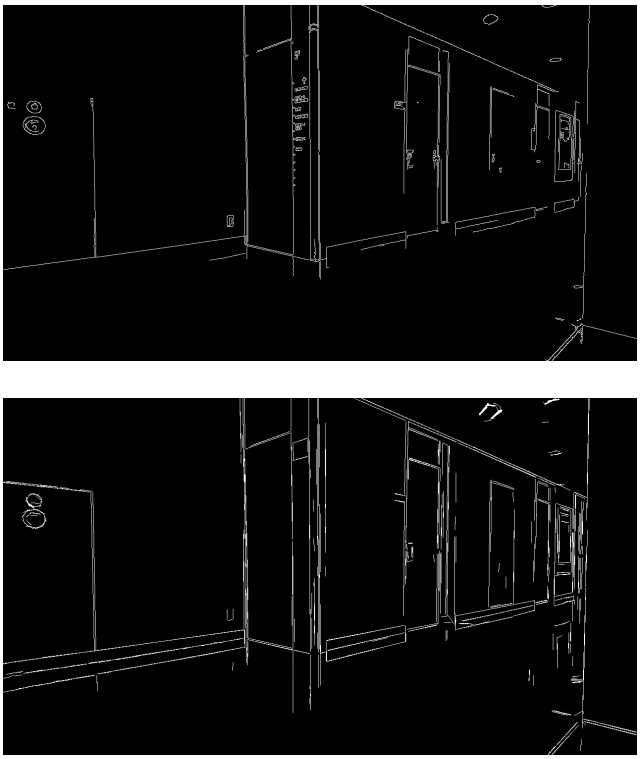

図 3　Canny 法(上)と LSD(下)による線分検出結果の比較

Vol.2017-MBL-82 No.21
Vol.2017-UBI-53 No.21
2017/3/9

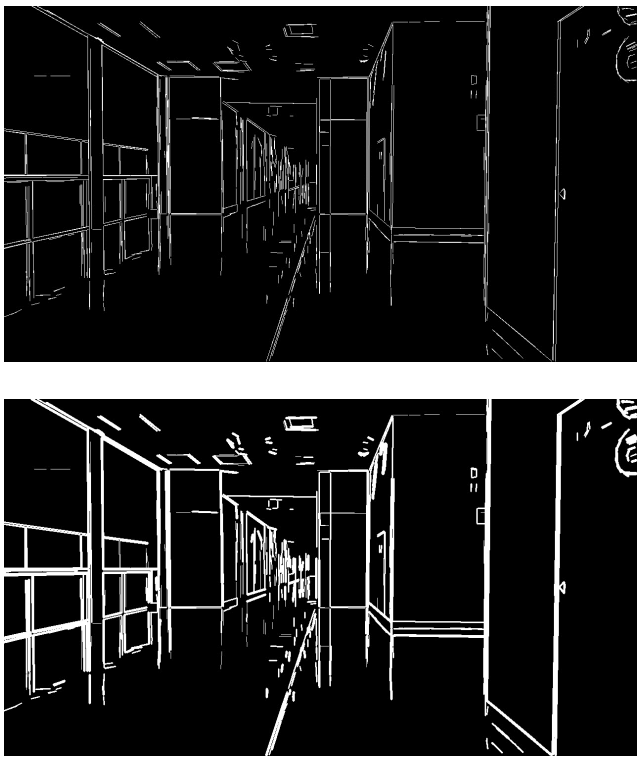

図 4 線分検出結果と膨張処理結果

め, データベースのサイズが縮小し探索時間が短縮される. 図 4 に線分の膨張処理を行った結果を示す.

3.5 線分画像のマッチング

クライアンの撮影画像に前処理を適用した処理画像と 2D-DB 内の画像のマッチング処理について説明する. 便宜上 2D-DB 内の画像を i (i=1,2,3,...) と表現する. マッチングは, 2 つの画像の論理積を取ることで行う. 処理画像と i 番目の DB 画像との論理積によって得られた画像を Ri とし, 要素が非ゼロとなるピクセル数を A^i_{lc} とする. また i 番目の DB 画像の要素が非ゼロとなるピクセル数を A^i_{db} とする. このとき, マッチング率 m は以下のように定義できる,

$$m^i = A^i_{lc} / A^i_{db}. \quad (1)$$

そして, ベストマッチ画像を以下の方法で選択する,

$$i_{best} = \arg\max_i m^i. \quad (2)$$

図 5 にある入力画像に対して DB 画像との論理積を取ったさいの結果画像を示す.

3.6 GPU 並列化による処理の高速化

本手法では, 位置推定の探索範囲が広がるにつれ, データベース画像の枚数と探索時間が増大していくという問題がある. そのため, マッチングには高速な処理が求められる. そこで, 本研究では処理の高速化を実現するために, Compute Unified Device Architecture (CUDA)を用いる.

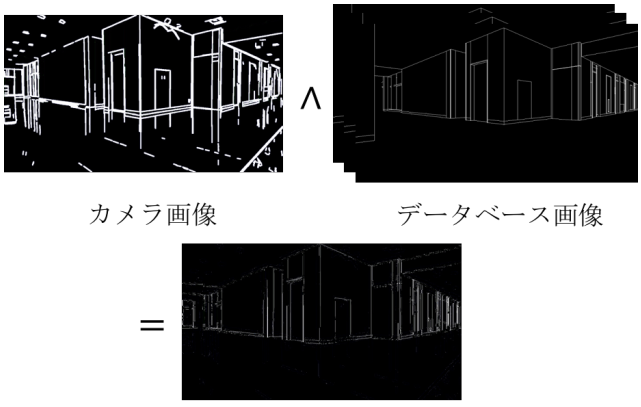

図 5 入力線分画像と DB 画像（m^i 最大), 論理積画像

CUDA とは NVIDIA が提供する GPU 向けの C 言語の統合開発環境である。GPU の性質上, 一度に大量のデータに対して単純な処理を行う画像処理とは非常に相性が良い.

本研究では, システムの起動時に画像データベースを GPU 上のメモリに展開する. システムにカメラ画像が入力されると GPU は画像のマッチング処理とマッチング率の演算を行う. 各データベース画像に対するマッチングでは, 画素ごとの論理積演算を並列処理することで高速化を実現している.

4. 評価実験

実装した U-Map システムを 3 つの実験により評価する. 1 つ目の実験は理想環境において本提案手法がどの程度の精度をだせるかを確かめるもので, 簡略化のために一辺 12cm の L 字ブロックを対象環境とする. 2 つ目の実験は実環境として大学建物内の廊下を対象環境とする. 3 つ目の実験は GPU 並列化による効果を検証するもので, DB の画像数に対して, CPU 演算と GPU 演算ではどのような差が出るかを検証する.

4.1 L 字ブロックを用いた評価実験

4.1.1 実験条件

クライアントデバイスが対象環境のセンシングをしている実験のシーンを図 6 に示す. 撮影の際には, スマートフォンをカメラ三脚で固定し, 撮影エリアには目的以外の線分が検出されないよう暗幕を引いた. スマートフォンの位置・姿勢はモーションキャプチャにより計測する. 図中の L 字ブロックやスマートフォンに付着している白い小球はモーションキャプチャのマーカである. スマートフォンは Nexus 5 をモーションキャプチャは OptiTrack 精度±2[mm] を使用した. スマートフォンによる撮影は, 注視点を L 字ブロックの中心に向けながら図 7 のように 1 周しながら行った. なお, 三脚でスマートフォンを固定してい

Vol.2017-MBL-82 No.21
Vol.2017-UBI-53 No.21
2017/3/9

るため高さは一定である． L 字ブロックを撮影した画像と視点座標の位置をセットとして，計 20 の入力画像データセットを作成した．

2D-DB 画像は，図 7 のように，撮影視点を移動していることを想定し，5 度刻みで 1 周 72 方向，中心からの距離が 31 パターンの合計 2232 枚の画像を生成した．

4.1.2 L 字ブロック環境でのマッチング結果

ある入力画像に対するベストマッチング結果の一例を図 8 に示す．図中で青は入力画像，緑は DB 画像，白は論理

図 6　L 字ブロックを対象環境とした実験の一場面

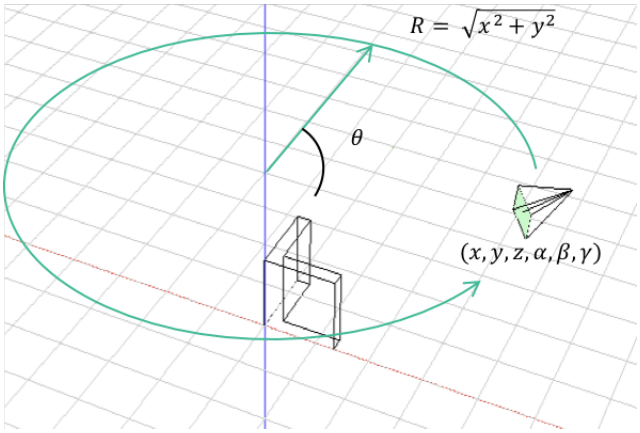

図 7　カメラの移動軌跡

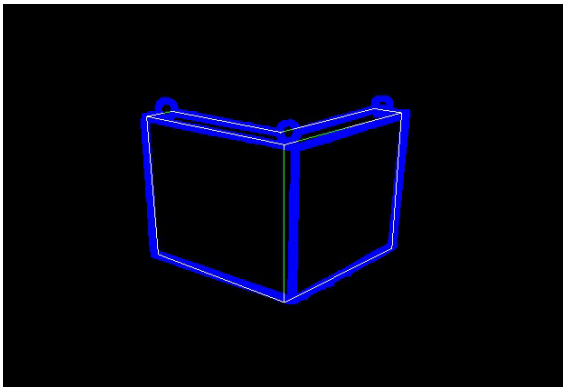

図 8　論理積を取った結果の画像

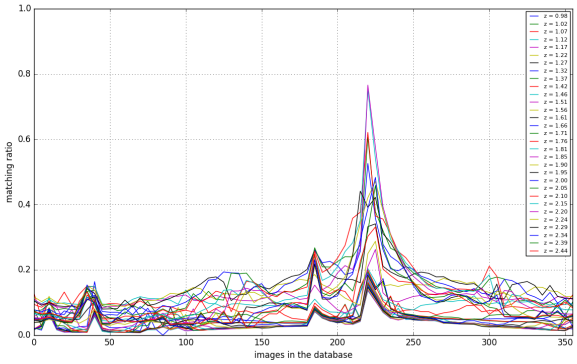

図 9　ある入力画像とすべての 2D-DB 画像の論理積演算後のマッチング率

積演算後の非ゼロピクセルである．この結果では，マッチング率 m=0.78 であった．すべての 2D-DB 画像とある入力画像のマッチング率を計算し結果をプロットしたグラフを図 9　に示す．グラフ中では L 字ブロックとスマートフォンの距離の違いを z で表現している．

4.1.3 L 字ブロック環境での位置推定の精度

モーションキャプチャを用いて取得した実際のカメラ位置と提案手法により推定した位置の差を取ることで，位置推定の精度を評価した．その平均誤差は 0.062 [m]であった．角度の誤差はデータベースの生成間隔よりも小さくなっていることから，生成間隔を狭めることで，より高精度な位置推定ができると考えられる．

4.2　実験環境

青山学院大学相模原キャンパス O 棟 5 階廊下にて実験を行なった．O 棟建設時に作成された 2D CAD の図面データと環境の実測により，3 次元 CAD ソフトウェア Rhinoceros で 3D ワイヤフレームマップを作成した（図 2）．3D ワイヤフレームマップより 2D-DB を生成する．画像生成領域は，東側廊下の端を原点として $37.00 < x < 46.00$ [m]，$-0.60 < y < 3.10$ [m], $0.66 < z < 0.73$ [m] とし，x 軸方向，y 軸方向は 0.10 [m] 刻み， z 軸方向は　0.01 [m] 刻みとした．各視点座標において，注視点の水平方向を $0 < \theta < 360$ [deg] の範囲で 45 [deg] ずつ回転させた画像を生成した．

入力画像データセットの作成では，図 10 に示すようにキャスターの上に三脚でスマートフォンを設置し，移動させながら撮影を行った．スマートフォンの位置の真値は精度 2[mm]のレーザ測距計 KINGTOP　SW-E100 を用いて壁や床らの距離を元に算出した．画像の撮影に用いたスマートフォンは Xperia X Performance を用いた．画像のファイルフォーマットは JPG とし画像サイズは 1200×720，垂直方向画角は 48° である．生成された画像ファイル数は 72,000 枚であった．

Vol.2017-MBL-82 No.21
Vol.2017-UBI-53 No.21
2017/3/9

図 10　入力画像データセット作成時の一例

4.3 廊下環境でのマッチング結果

マッチング結果の一例として，入力画像 p2001（図 11）を線分検出した画像を図 12，マッチングアルゴリズムにかけた際の論理積画像を図 13 に示す．図 13 において緑は DB 画像，白は論理積演算後の非ゼロピクセルである．この結果では，ベストマッチング率 m=0.28 であった．すべての 2D-DB 画像とある入力画像のマッチング率を計算し結果をプロットしたグラフを図 14 に示す．また，このグラフを CAD マップに投影したものを図 15 に示す．

図 16 に示すのは，壁にポスタなどが貼ってあり構造物線分以外の線分が多く検出された場合の論理積結果画像である．ポスタの枠線は事前マップの 3D ワイヤフレームモデルには含まれていないため，マッチング処理においてはノイズでしかない．しかしながら，このようにノイズの多い入力データでも論理積演算により正しいベストマッチ画像を算出することができた．

図 11 入力画像 p2001 したカメラ画像

図 12 入力画像 p2001 の線分検出結果

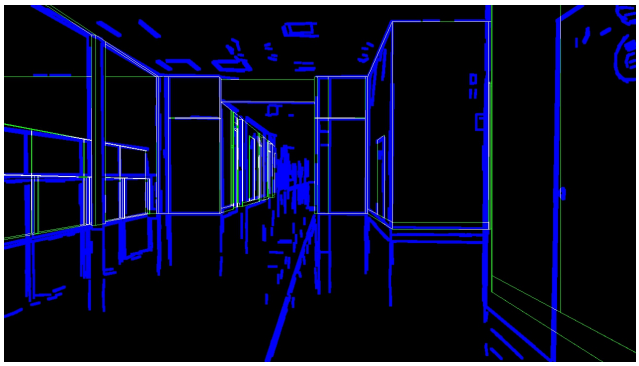

図 13 入力画像 p2001 に対するベストマッチ論理積画像

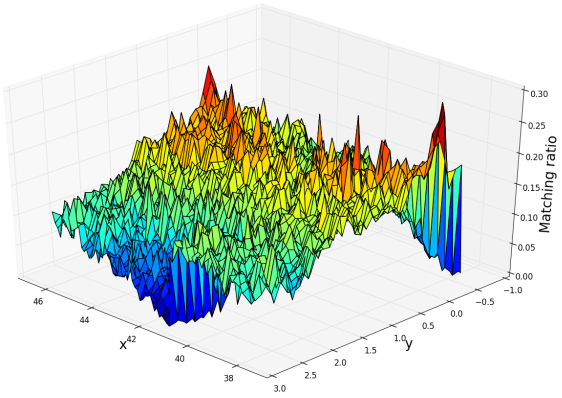

図 14 入力画像 p2001 と 2D-DB 画像の
論理積演算後のマッチング率

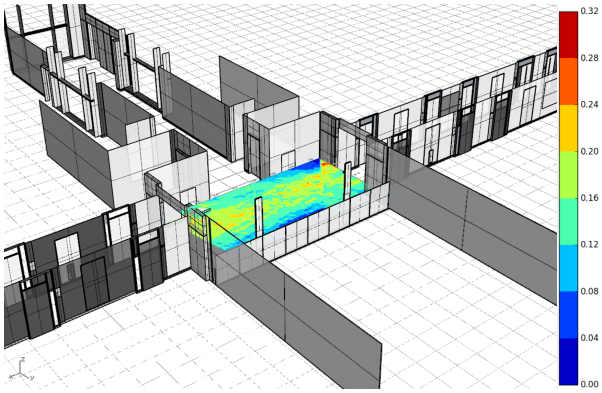

図 15 マッチング率結果グラフの CAD マップへの投影

Vol.2017-MBL-82 No.21
Vol.2017-UBI-53 No.21
2017/3/9

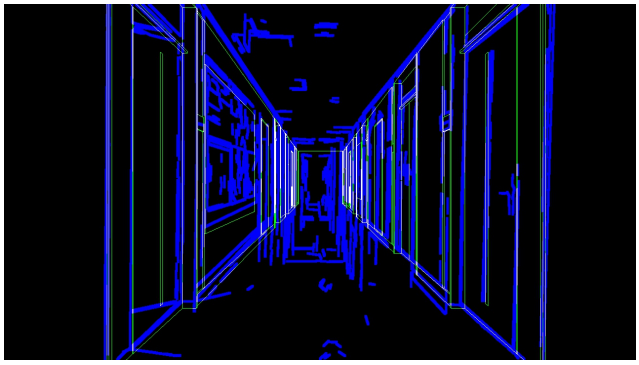

図 16 構造物線分以外の線分が多い場合

4.4 廊下環境での位置推定精度

マッチング率が最も高かった画像に紐づいた位置情報と計測実験から得た真値を比較することで，本手法の精度を評価した．ここで，カメラの位置座標を (x_c, y_c, z_c)，カメラの注視点の座標を (x_f, y_f, z_f)とすると，推定値と真値との各軸ごとの誤差の平均は，$(x_c, y_c, z_c, x_f, y_f, z_f)$=(-0.033, -0.610, -0.005, 0.033, -0.061, -0.005) となった．視点の推定値と真値との誤差としてユークリッド距離を求めたところ誤差の分布は図 17 のようになった．誤差の平均値は　0.12 [m]　であった．

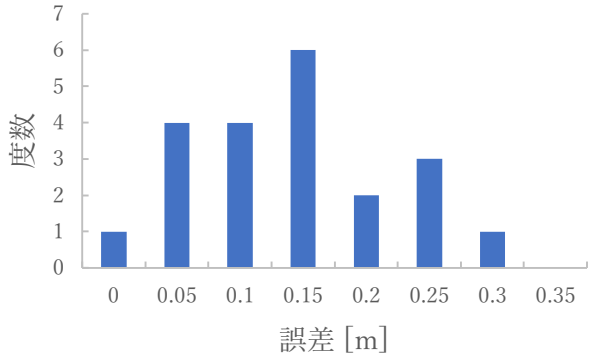

図 17 位置推定誤差の分布

4.5 GPU による処理高速化の評価

画像マッチングの処理に GPU を利用することで，システムの高速化を図っている．CPU（Intel Xeon 3.50 GHz）のみの場合と GPU（Geforce GTX 1080）を用いた場合を比較し，その処理時間を評価した．計測したマッチングプロセスを図 18 に示す．なお，図中の青破線はシステムの起動プロセスを表し，赤破線はマッチングプロセスを表す．それぞれの

CUDA では，処理するデータを GPU のメモリ上に展開することが必要になるため，CPU のみの場合よりもシステムの起動に時間がかかる．図 19 にデータベース画像の枚数に対する CUDA システム起動時間の関係を示す．

図 20 に CPU のみでマッチングを行なった場合と GPU を使用してマッチングを行なった場合の処理時間を比較したグラフを示す．12600 枚の画像の場合で CPU: 90 [s], GPU: 0.4 [s]の処理時間であった．

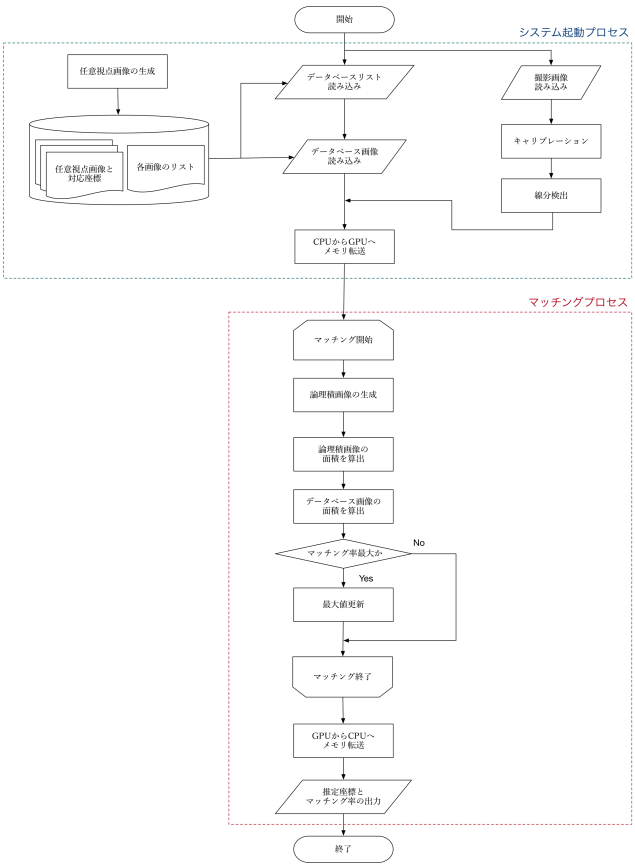

図 18 システム起動プロセスとマッチングプロセス

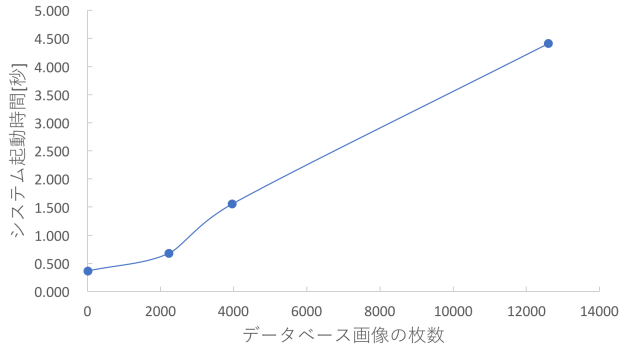

図 19 DB 画像枚数と CUDA システム起動時間

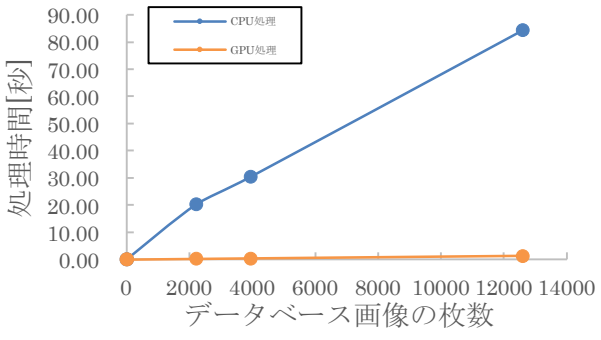

図 20 CPU と GPU の処理時間の比較

5. むすび

本論文では，サーバ上に保持した 3D ワイヤフレームマップとクライアントが取得した画像の線分情報とのマッチングを行うことで位置推定を行うインフラストラクチャシステム: U-Map (Universal Map) を提案した．マッチングでは 3D ワイヤフレームマップから 2D 画像を生成し，撮影画像との論理積を取ることで，位置推定を行うことを可能にした．これにより，従来の位置推定手法では必要とされていた，複数の異なる地点でのセンサデータの取得や，環境へのセンサノードの設置や管理,運用環境の調査･測量,高価な機材の利用なしで．位置推定システムを構築可能であることが示された．実環境における実験では，誤差のユークリッド距離は 0.12 [m]となり，十分な精度が得られることがわかった．GPU を利用することによって，CPU のみの処理と比べて 200 分の 1 の処理時間になることから，高速かつ低演算負荷の処理でシステムを運用することができることが示された．

本研究では，3 次元ワイヤフレームマップから探索範囲の 2 次元画像を生成していたが，探索範囲を縮小するために設置済みの Wi-Fi アクセスポイントを利用することが考えられる．Wi-Fi ルータの情報からおおよその位置が推定できるため，探索範囲を限定することができる．今後，IoT などの普及が進んでいくとともに，どのような場所であっても無線ネットワークが利用できることが考えられる．したがって，このような手法の実現性は十分に高い．

謝辞 本研究は JSPS 科研費 16K21339 の助成を受けた．

参考文献

[1] D.G. Lowe. “Object recognition from local scale-invariant features, " In Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 2, pp. 1150 - 1157, 1999.
[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool,: “SURF: Speeded Up Robust Features, ” computer vision-ECCV Lecture Notes in Computer Science, 2006
[3] Pablo F. Alcantarilla, Adrien Bartoli, and Andrew J. Davison.: “KAZE features, " European Conference on Computer Vision (ECCV). Springer Berlin Heidelberg, 2012.
[4] P. F. Alcantarilla, J. Nuevo and A. Bartoli, “Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces, " British Machine Vision Conf. (BMVC). Bristol, UK, 2013.
[5] R.G.von Gioi, J.Jakubowicz, J-M.Morel, and G.Randall, “LSD:A Fast Line Segment Detector with a false De- tection Control, ” IEEE Trans. on PAMI, vol.32, Is- sue.4, pp.722-732, Apr. 2010
[6] 廣瀬圭佑, 斎藤英雄. "対象環境の線分情報を用いた実時間 SLAM." 画像の認識・理解シンポジウム (MIRU2011) 論文集 2011 (2011): 818-824.
[7] 風間光, 川本一彦, 岡本一志. (2012). 画像検索を用いた図書館内での自己位置推定. 研究報告コンピュータビジョンとイメージメディア (CVIM), 2012(27), 1-8.
[8] Tim Caselitz, Bastian Stede, Michael Ruhnke, Wolfram Burgard. “Monocular camera localization in 3D LiDAR maps, " Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE, 2016.
[9] Abel Gawel, Titus Cieslewski, Renaud Dub¥'e, Mike Bosse, Roland Siegwart and Juan Nieto(2016, October). “Structure-based vision-laser matching, " In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on (pp. 182-188). IEEE.
[10] Zhang, Zhengyou, "A flexible new technique for camera calibration, " IEEE Transactions on pattern analysis and machine intelligence 22.11, pp. 1330-1334, (2000).