「マルチメディア，分散，協調とモバイル
(DICOMO2019)シンポジウム」 令和元年7月

スマートフォンを用いた床指紋照合

藤田 悟[1]

概要：本研究は，床面の画像から特徴点抽出を行い，床指紋として画像照合することで，屋内でも利用可能な位置推定システムを実現することを目的としている．これまでの研究成果として，特徴点の効果的抽出手法や，床画像データベースとの高速照合技術，撮影画像の 3 次元歪を重力ベクトルや ARCore を利用して補正する技術などを確立してきた．しかし，実際の照合には GPU を搭載したデスクトップ PC を利用しており，スマートフォン単体で実現した場合の性能や課題について明らかではなかった．そこで，本報告では，スマートフォン上に全機能を実装し，これに加えて，撮影位置を cm 単位で正確な推定する手法について検討した．そして，native コードを利用した実装を行うことで，高速に床指紋照合できることを実証した．撮影位置についても，重力センサーを用いて端末の姿勢認識をすることで，高精度な推定が可能であることを示した．

Floor Fingerprint Identification on Smart Phone

SATORU FUJITA[1]

1. 序論

床表面は，人の目から見ると，一見，どこを見ても同じような表面が続いているように見えるが，実際には，微細な模様や汚れ，傷などが多数存在していて，それぞれに微妙に異なるテクスチャーを保有している．このような床表面に現れる固有のテクスチャーから特徴点を抽出し，唯一に同定することができれば，床画像から，特定の位置を推定することが可能になる．著者らは，この技術を，床指紋照合と名付け，特徴点の効果的抽出手法や，床画像データベースとの高速照合技術について研究を進めてきた [4][5]．床画像は，一般的に強い特徴点が少なく，また，似たような模様が多数存在するため特徴点のマッチングを正しく行うことが難しく，総じて照合が困難である．これに対して，過去の研究成果の中で，ORB[11] を改良した特徴点の抽出手法や，RANSAC[2] を用いた高速なマッチング手法に関して提案し，実際に 87.5m の廊下を用いて，床指紋照合による位置推定実験を実施し，正解率 99.5%，1 画像当たりの照合時間 0.5 秒の結果を得て，この位置推定手法の実用性を示してきた．

同様の技術は，ロボットや自動車業界でも注目されている [3][8][13]．ヤマハ発動機はカートのような低速自動走行車両を実現する技術として，車両底部に設置されたカメラで撮影した路面画像をデータベース画像と特徴点マッチングすることで，車両の位置を特定する VGL(Virtual Guide Line) を開発し，マーカーなしで 20km/h の車両の自動走行を実現している [3]．この技術は Ranger という技術を応用したもので，カメラの高速シャッターに同期した LED 照明を用い，道路面に垂直に向けられたカメラ画像を用いて，130km/h で走行する車両から撮影された道路表面画像を毎秒 10 枚の速度で，特徴点マッチングして位置推定する機能を有している [8]．

車両底部に設置したカメラは，画角や撮影距離が一定であることや，照明を調整することで，安定した品質の画像を取得することができるが，スマートフォンを用いて，街角を行く歩行者が撮影した画像から位置推定する状況では，別の課題が存在する．ひとつは，撮影時のスマートフォンの位置や傾きが自由であることからくる画像の歪である．スマートフォンを傾けて保持して撮影した場合には，画像に 3 次元の透視投影の効果による歪が生じ，認識精度が低下する．そこで，スマートフォンに備わっている加速度センサーを用いて重力方向ベクトルを求めることにより，画

[1] 法政大学情報科学部
Computer and Information Sciences, Hosei University

像の傾斜歪を補正する方法 [7] や，ARCore を利用した補正方法 [6] を検討し，報告してきた．しかし，これらの実装は GPU を装備したデスクトップ PC 上に行っており，リソースの限られたスマートフォン上での評価は行われてこなかった．

本報告では，床指紋照合のすべての機能をスマートフォン上で実装し，性能チューニングと，性能評価を行う．これに加えて，スマートフォンの傾きを計算に入れて，撮影位置を詳細に決定する手法について検討を行う．

2. 関連研究

2.1 物体表面の画像照合

物体表面の画像照合は，一般の画像照合と同様に，ORB[11] や AKAZE[1] などを用いてミクロな特徴点を抽出して，特徴量を計算し，特徴点間のマッチングを行い，その中から妥当なマッチングペア (inlier) を抽出するというステップで実現する．最終的に，inlier 数や inlier の outlier に対する割合に対して閾値を設定して，照合の成否を判定する．ここで，物体表面画像の照合の難しさは，特徴点の弱さにある．一般の画像特徴点は，エッジや突起などに起因する明確な明度の違いのある個所を同定して，特徴量を計算する方式が取られるが，床面や道路面には，そのような強い特徴点は少ない．そこで，弱い特徴点をいかに安定して取得できるかが，物体表面の画像照合で重要なポイントになる．

FIBAR では，ネジの頭部分の梨地仕上げを識別するために，スマートフォンのカメラ部分に，外部光の拡散フィルタと表面の突起を強調するための黒色のリングを装着して，画像を撮影している．撮影時のネジの傾きや，レンズとネジの頭部との距離も一定に保たれ，画像歪も小さく抑えられている [12]．Ranger は，車両底部に搭載したカメラによって，道路表面の画像を撮影し，路面の照合を行う．カメラの横には高輝度な LED ライトが用意され，カメラのシャッターと同期して路面を強く照らすことで，高速走行時にも，ブレの少ない安定した画像を撮影することに成功している．カメラは，下向きで道路面との距離や傾きは一定に保たれているため，特徴点マッチングの照合については，Homography 変換に必要な 4 点の特徴点ではなく，2 点の特徴点から照合することができるようになり，RANSAC による確率的な照合を高速に実現してる [8]．Zhang らも，移動体の底部に LED ライトとカメラを装着して，等距離かつ傾きのない画像を撮影して，床や道路面の撮影を行い，位置推定システムを構築している．特徴点マッチングの照合については，投票方式によりヒストグラムを作成し，マッチングに成功した点が集中する場所を推定位置とする方式を用いて，照合にかかる計算コストを抑えている [13]．

上記のような研究と比較すると，本報告の環境は，利用者によるスマートフォンを利用した撮影環境であり，光源の調整や，撮影角度や距離を一定に保つことも困難な環境になる．そこで，特徴点検出のための閾値を下げて，より多くの特徴点を抽出し，その中から適切な照合ペアを検出する必要がある．照合コストを抑えるためには，RANSAC に用いる特徴点数を抑える必要があり，事前の撮影角度の補正が必要と考える．歪補正についての関連研究については，次節に述べる．

2.2 歪補正手法

Kurz らは，重力ベクトルをを用いて，水平な平面画像の 3 次元歪を補正する手法 GREED (gravity-rectified feature descriptors) を提案した [9][10]．この手法では，端末の座標系で表現された重力ベクトル $\boldsymbol{g} = (g_x, g_y, g_z)^T$ を，地表面のグローバル座標系 $\boldsymbol{z} = (0, 0, 1)^T$ に対応させるように，ホモグラフィ行列 H を求める．まず，重力ベクトル $\boldsymbol{g}$ を法線とする平面を $\boldsymbol{g_1}$ と $\boldsymbol{g_2}$ という直交ベクトルで表現する．

$$\boldsymbol{g_1} = (-g_z, 0, g_x)^T \tag{1}$$

$$\boldsymbol{g_2} = \boldsymbol{g} \times \boldsymbol{g_1} \tag{2}$$

この時，ホモグラフィ行列 H_{kurz} は，次のように与えられるとした．

$$H_{kurz} = [\boldsymbol{g_1} \ \ \boldsymbol{g_2} \ \ \boldsymbol{z}]^{-1} \tag{3}$$

この手法では，H_{kurz} によるホモグラフィ変換結果の表示位置や表示倍率が思った結果にならない場合があり，Kurz らの論文でも，経験的に，式 (4) のような微修正を行って利用している [9][10]．

$$H_{scaled} = [\boldsymbol{g_1} \ \ \boldsymbol{g_2} \ \ \sqrt{|\boldsymbol{g}|}\boldsymbol{z}]^{-1} \tag{4}$$

一方，著者らは，表示位置や表示倍率について，より正確に定義したホモグラフィ行列の計算方法を示した [7]．このホモグラフィ行列を用いることで，前後に傾斜した端末から撮影された画像が，カメラを水平な位置に保って撮影した画像に変換されることを確認した．ただし，端末の地表面からの高さが一定という仮定があり，また，端末の左右方向の傾きについては，考慮していないという課題があった．そこで，Google の開発した ARCore を用い，端末の高さを推定し，より正確なホモグラフィ行列を計算する手法を開発した [6]．この計算手法では，端末の前後の傾斜に加え，左右の傾斜についても考慮されており，期待通りの画像補正に成功した．

3. スマートフォン上の床指紋照合

3.1 全体構成と実装

スマートフォン上での床指紋照合システムを構築するための第一の課題は性能である．画像処理ライブラリである OpenCV は，android 環境でも利用可能であるが，その利用方法に課題がある．すなわち，OpenCV の本体は C++ で書かれていて，android 用に Java とのインタフェースが提供されていることである．このインタフェースを通して Java から OpenCV を呼び出すと，頻繁にデータ配列のコピーが発生し，性能を劣化させる．そこで，性能がクリティカルな部分は，Java で実装しないで，C++ を用いた native 関数として実装する必要がある．以下に，床指紋照合に必要な機能別に実装上の課題と選択した実装手法について説明する．

3.1.1 画像取得

OpenCV には，JavaCameraView クラスが用意されていて，フレームごとに OpenCV の Mat クラスのインスタンスとして画像を取得することができる．ただし，この画像は android のプレビュー画像であり，解像度は低い．例えば，本研究で利用する Pixel3 については，プレビュー画像の解像度は 1920x1080 である．プレビューモードでは，手振れなどの影響を受けやすいため，念のため，静止画撮影モードで画像取得する方法も実装したが，結果的にはプレビュー画像でも解像度的に充分であることが示されたため，本論文では JavaCameraView のプレビュー画像を入力として扱う．

3.1.2 歪補正

画像照合の前段階で必要な機能は，3.2 節で詳説する画像の 3 次元歪の補正である．重力ベクトルから，ホモグラフィ行列を計算し，床を直上から見下ろした画像に変換する必要がある．この機能はホモグラフィ行列を計算した後は，OpenCV の画像変換関数を一度呼び出すたけなので，C++ と Java のインタフェースによるコスト低下の影響が低いため，Java で実装する．

3.1.3 画像フィルタ

変換後の画像を Gaussian 差分フィルタなどで輝度調整を実施する．画素にアクセスする度に C++ と Java のインタフェースを介するため，Java 実装の場合，著しく低速になった．よって，C++ による native 実装を行う．ただし，実験によっては，画像フィルタがなくても，十分な照合性能が得られたことから，必要に応じて画像フィルタの機能自体のオン・オフを切り替える．

3.1.4 特徴点抽出

著者らの床指紋の研究では，ORB を拡張し，特徴点を広く分散させて抽出する B-ORB を実装している [5]．このライブラリは，ORB のライブラリとの結びつきが強いため，Java で実装することは難しいため，そのまま，C++ の実装を行う．

3.1.5 特徴点マッチングペアの抽出

特徴点マッチングをした後の検出ペアの中から正例のペアを抽出する機能であり，照合性能と速度性能に影響する重要な機能である．今回，android 上で様々な調整を行ったため，複数のアルゴリズムについて Java で実装を行い，最後に C++ による native 実装に置き換えるなどして，性能比較実験を行った．

3.2 重力ベクトルを用いた歪補正

文献 [6] で ARCore を利用した画像の 3 次元歪の補正手法を提案した．ARCore を利用する場合，ARCore がカメラデバイスを占有してしまうため，独自に開発した部分からの画像取得が難しく，以前の研究では，ARCore が画面表示するための関数から，プレビュー画像を抜き出す手法を利用していた．今回，OpenCV とも共存する必要があるため，処理が煩雑になる．そこで，ARCore は使わず，重力ベクトルを用いた歪補正手法に立ち戻って開発することにした．

重力ベクトルを用いる 3 次元歪手法については，文献 [6] の考察で簡単に述べたが，ARCore を用いた歪補正手法を応用して，期待する歪補正を行うことができる．欠点は，ARCore と違って端末の地上からの高さを推定する機能を持たないことである．これについては，端末の高さ h を通常の利用局面から 1.1m に固定して対応する．以下に，歪補正の計算手法を説明する．

まず，Kurz らと同様に，端末の座標系で表現された重力ベクトル $\boldsymbol{g} = (g_x, g_y, g_z)^T$ を，地表面のグローバル座標系 $\boldsymbol{z} = (0, -1, 0)^T$ に対応させるように，変換行列を求める．ARCore の座標系は，垂直方向上向きが y 軸と定められているため，これと合わせて，端末の local 座標系から見た重力ベクトル $\boldsymbol{g} = (g_x, g_y, g_z)^T$ の向きを $-y$ 軸とし，地面に水平な x 軸と z 軸を持つような global 座標系の基本ベクトル $\boldsymbol{b_x}$, $\boldsymbol{b_y}$, $\boldsymbol{b_z}$ を，端末の local 座標系を用いて，次のように定義する．

$$\boldsymbol{b_y} = (g_x/G, -g_y/G, g_z/G)^T \tag{5}$$

$$\boldsymbol{b_x} = (g_z/G', 0, -g_x/G')^T \tag{6}$$

$$\boldsymbol{b_z} = \boldsymbol{b_x} \times \boldsymbol{b_y} \tag{7}$$

ただし，$G = \sqrt{g_x^2 + g_y^2 + g_z^2}$, $G' = \sqrt{g_x^2 + g_z^2}$ である．この時，ホモグラフィ行列 H_0 は，上記の 3 個の基本ベクトルから構成できる．

$$H_0 = [\boldsymbol{b_x} \ \ \boldsymbol{b_y} \ \ \boldsymbol{b_z}] = \begin{pmatrix} \frac{g_z}{G'} & \frac{g_x}{G} & \frac{-g_x g_y}{G'G} \\ 0 & \frac{-g_y}{G} & \frac{-G'}{G} \\ \frac{-g_x}{G'} & \frac{g_z}{G} & \frac{-g_y g_z}{G'G} \end{pmatrix} \quad (8)$$

一方，逆に global 座標系からみた local な座標系の基本ベクトルの変換行列 E は，H_0 の逆行列となる．

$$E = H_0^{-1} \quad (9)$$

この基本行列は ARCore の姿勢行列と等価であり，文献 [6] に示す ARCore を用いた 3 次元歪と同様の計算を行える．すなわち，この基本行列に沿って端末の local 座標系の座標を $\boldsymbol{s} = (s_x, s_y, s_z)^T$ とし，端末の位置座標を $\boldsymbol{p}$ とすると，カメラから見える物体の座標 $\boldsymbol{o}$ は，式 (10) で表される．

$$\boldsymbol{o} = \boldsymbol{p} + E\boldsymbol{s} \quad (10)$$

一方，端末のスクリーン座標 (u, v) を，u 軸が右向き，v 軸が下向きに定義し，端末のスクリーンの画素数を，縦 r_y，横 r_x であるとし，端末から，1m 離れた面を撮影した時の縦方向の実世界の長さを l とすると，s_x, s_y, s_z と u, v の関係は，次のように表現できる．

$$\begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} = L_1 \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} s_z \quad (11)$$

$$L_1 = \begin{pmatrix} -\frac{l}{r_y} & 0 & \frac{r_x l}{2r_y} \\ 0 & \frac{l}{r_y} & -\frac{l}{2} \\ 0 & 0 & 1 \end{pmatrix} \quad (12)$$

これに基づいて，点 o は，式 (13) に展開できる．

$$\boldsymbol{o} = \boldsymbol{p} + EL_1 \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} s_z \quad (13)$$

次に，カメラ端末からの視線が地表面と交差する点の座標を $\boldsymbol{o_0}$ とする．この $\boldsymbol{o_0}$ をカメラの local 座標系で表すと，$(0, 0, s_{0z})^T$ になる．端末と地表面の高度差を h とし，$\boldsymbol{e_z} = (e_{zx}, e_{zy}, e_{zz})^T$ とすると，s_{0z} は，式 (14) で計算することができる．

$$s_{0z} = -h/e_{zy} \quad (14)$$

以上により，交点 $\boldsymbol{o_0}$ は，式 (15) で表される．

$$\boldsymbol{o_0} = \boldsymbol{p} + \boldsymbol{e_z} s_{0z} \quad (15)$$

次に，ホモグラフィ変換を行いたい床面の領域を視線と床面との交点 o_0 を中心に矩形領域として設定する．矩形領域の座標 (u', v') は，それぞれ，u' 軸が右向きとし，$(e_{zz}, 0, -e_{zx})^T$ 方向にとり，v' 軸が下向きとし，$(e_{zx}, 0, e_{zz})$ 方向にとる．単位ベクトル化するため，$d = \sqrt{e_{zz}^2 + e_{zx}^2}$ とすると，床面上の点 $\boldsymbol{o}$ は式 (16) で表現される．

$$\boldsymbol{o} = \boldsymbol{p} + TL_2 \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \quad (16)$$

$$L_2 = \begin{pmatrix} \frac{l}{r_y} & 0 & \frac{r_x l}{2r_y} \\ 0 & \frac{l}{r_y} & -\frac{l}{2} \\ 0 & 0 & 1 \end{pmatrix} \quad (17)$$

$$T = \begin{pmatrix} \frac{e_{zz}}{d} & \frac{e_{zx}}{d} & -\frac{e_{zx}}{e_{zy}} h \\ 0 & 0 & -h \\ -\frac{e_{zx}}{d} & \frac{e_{zz}}{d} & -\frac{e_{zz}}{e_{zy}} h \end{pmatrix} \quad (18)$$

床面上の点において，式 (13) と式 (16) は等しくなることから，式 (19) を導くことができる．

$$EL_1 \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} s_z = TL_2 \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \quad (19)$$

よって，床面の矩形領域の座標系と，それが対応するスクリーン座標の関係は，式 (20) で表現できることがわかる．

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} s_z = L_1^{-1} E^{-1} TL_2 \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \quad (20)$$

ホモグラフィ行列 H_{gr} は，式 (21) を満たす行列として定義される．

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} s_z = H_{gr}^{-1} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \quad (21)$$

最終的に，重力ベクトルを用いた歪補正のためのホモグラフィ行列 H_{gr} は，式 (22) にて計算できる．

$$H_{gr} = (L_1^{-1} E^{-1} TL_2)^{-1} \quad (22)$$

しかし，このままでは，端末を傾けて撮影した画像の中央付近の画像を射影変換することになるため，解像度の低い画像となる．解像度の高い画像を得るには，撮影した画像の中央ではなく，距離が近くて大きく映し出されている画像底辺のすぐ上にあたる部分画像を用いるべきである．これを実現するため，L_2 の中心を底辺に平行移動させた L_2' を定義する．式 (22) の L_2 を，この L_2' に置き換えることで，目的の底辺の直上の部分の画像変換が可能になる．

$$L_2' = \begin{pmatrix} \frac{l}{r_y} & 0 & \frac{r_x l}{2r_y} \\ 0 & \frac{l}{r_y} & -\frac{l}{2} + d \\ 0 & 0 & 1 \end{pmatrix} \quad (23)$$

ここで，移動量 d は式 (24) で決定する．$E_{i,j}$ とは，E の i 行 j 列要素を示す．式の詳細な説明は省略する．

$$d = \sqrt{m_x^2 + m_y^2 + m_z^2} - \frac{lh}{2} \tag{24}$$

$$\begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} = \begin{pmatrix} E_{0,2} * (r_1 - r_0) + \frac{r_1 l E_{0,1}}{2} \\ E_{1,2} * (r_1 - r_0) + \frac{r_1 l E_{1,1}}{2} \\ E_{2,2} * (r_1 - r_0) + \frac{r_1 l E_{2,1}}{2} \end{pmatrix} \tag{25}$$

$$r_0 = -\frac{h}{E_{1,2}} \tag{26}$$

$$r_1 = -\frac{h}{E_{1,2} + lE_{1,1}/2} \tag{27}$$

以上により得られたホモグラフィ行列 H_{gr} を用いて，3 次元的な端末の傾きを補正し，直上から撮影されたような床画像を得ることができる．

3.3 撮影位置の推定

3.2 節のような画像変換を行っているため，実際の撮影位置は，常に変換後の画像の直上にあるのではなく，斜め方向から撮影されている場合がある．位置推定を行うためには，このような画像に対する撮影者の位置の差異を計算する必要がある．これは，式 (16) の右辺第 2 項の L_2 を，式 (23) の L_2' に置き換え，$u' = 0, v' = 0$ と置くことで，視点 p から床面の矩形領域の原点との位置ベクトルの差を計算することで解決する．この手順により，位置情報を照合した画像単位ではなく，cm レベルの位置推定を行うことができるようになる．

4. 性能評価

4.1 基本機能の動作確認

床指紋照合の基本機能の動作を確認する．基本画像に対して，別途撮影した画像を照合した時に，その照合結果を撮影画像のフレームを赤線で，撮影位置を青色のポイントで示すようになっている．図 1 にその例を示す．赤色の矩形が，マッチングを行った撮影画像である．この画像では，基本画像に対して，端末を若干傾けて撮影した場合の例であり，青色のポイントが手前側に現れ，その時の撮影ポイントを表している．照合成功時には，実行にかかった時間が android の Toast 画面として表示されるようになっている．

4.2 実行性能評価

木目調のフローリングの床面に対して，基本性能評価実験を行った．この実験では，事前のフィルタを省略し，B-ORB による特徴点抽出，BruteForceMatcher による特徴点マッチングを行い，その後で，文献 [5] に示す相似三角形を抽出する方法によるマッチングペアの妥当性検証を行った．B-ORB による特徴点抽出数は，デスクトップ PC を利用していた従来の研究では 10000 点としていたが，スマートフォン端末の性能を考慮して，1000, 2000, 5000, 10000 点で実験を行った．実験を行った機種は，Pixel3 64GB モデルである．実験結果を表 1 に示す．それぞれの特徴点数につき，10 回程度の実験の平均値を示す．

図 1　照合結果のサンプル画像

表 1　特徴点数を変えた実行時間の比較 (msec)

特徴点数	1000	2000	5000	10000
合計時間	342.2	244.9	659.2	2353.8
特徴点抽出	224.3	161.0	275.5	506.2
マッチング	34.0	57.5	368.0	1814.9
照合判定	83.8	26.0	15.6	32.7
照合率	68.8%	100%	100%	100%

特徴点数が 1000 点の場合は，16 回の試行に対し，5 回で照合に失敗し，照合に成功した 11 回の平均時間を示している．特徴点が 2000 点以上の実験については，すべての照合に成功している．

1000 点のモデルでは，照合成功率が 68.8% である．特徴点抽出時間が 2000 点モデルより大きくなったのは，多くの特徴点から 1000 点に絞り込むためのソーティング時間の増加などが起因すると考えられる．また，照合判定時間が大きくなっているのは，RANSAC により，正しいマッチングペアを見つける確率が減少したため，RANSAC の乱数選択ループ数が大きくなったためであると考える．

2000 点のモデルは，総じて実行時間が短く，照合失敗もないことから，今回の実験では，最適なパラメータであったと言える．特徴点数が少ないことから，特徴点抽出時間が短いことは当然として，マッチングにかかるコストも 57.5 msec に抑えられており，全体のコストを引き下げている．

5000 点，10000 点のモデルになるにつれ，特徴点抽出時間，マッチング時間が増大する．10000 点のモデルについては，文献 [5] では，同様のモデルで，デスクトップ PC で約 0.5 秒の実行時間が必要であった．今回，2.35 秒ということで，4 から 5 倍程度の実行時間が必要になっている．多くの時間はマッチングに費やされており，デスクトップ PC での実行時間と比較して，抽出時間が 3.95 倍，マッチング時間が 10.2 倍かかっている．デスクトップ PC では，マッチングに GPU を利用していることから高速化が図ら

図 2　トラッキング

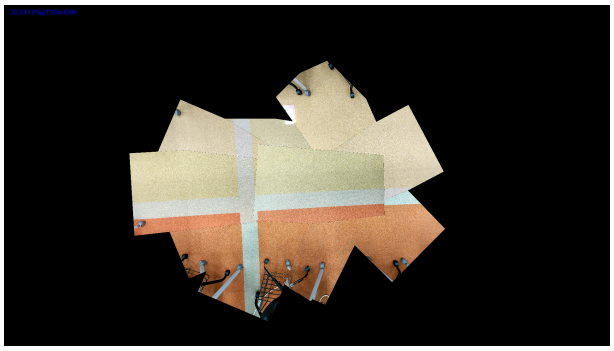

図 3　複数画像の重ね合わせ

れていた一方，スマートフォンでは，CPU ベースでのマッチングため，性能ボトルネックになっていると考えられる．

全体的には，特徴点数を 2000 点とすることで，照合率を落とさずに，0.24 秒で画像照合を行うことができた．これは GPU 付きのデスクトップ PC で特徴点を 10000 点抽出して計算した場合と比べても倍速であり，非常に短い時間である．不照合の時には，より大きな実行時間が必要になるものの，実用レベルの速度が実現できたと考える．スマートフォンの CPU 性能があがり，native 実装することで大きなペナルティなく，床指紋照合を実装可能であることが示された．

4.3　トラッキング

撮影位置の推定結果を，連続することで，人物の歩行経路をトラッキングすることができる．図 2 にトラッキングの様子を示す．青い点が撮影位置であり，時間的な移動を青い線で示している．このように cm 単位で，位置を推定でき，歩行経路を追跡することができることを確認した．

4.4　床画像の収集

床画像の照合ができれば，画像を照合しながら連結することで，床全面の画像データベースの作成が可能になる．現在，この機能は実現できていないが，複数枚の画像を照合後に重ね合わせて表示する機能を実装している．それを，図 3 に示す．この図では，1 枚の基本床面写真を撮影し，それと照合した複数の画像をそのまま重ね合わせている．重ね合わせの精度がわかりやすいように，十字形の模様が入った絨毯の教室の床で実験した．周囲に見えているものは可動式の机や椅子の足であり，縦 3m 横 5m ほどの領域を 10 枚程度の写真を重ね合わせて表示している．十字形を中心に，絨毯の模様である直線もほぼ正確に再現されていることが確認できる．まだ，初期実験段階ではあるが，床指紋による照合で，広域の床画像を照合しながら収集を行えるという可能性を示すことができた．

5.　考察

スマートフォン上で動作する床指紋照合システムについて述べ，試作機による実験を行った．スマートフォン上に実装するという速度性能面での制約が大きかったが，主な部分を C++ による native 実装を行い，また，特徴点数を限界まで少なくするなどのパラメータチューニングを行うことで，1 枚の画像照合に必要な時間は，約 0.24 秒にまで抑えることができた．秒 4 回の位置推定が可能であり，かなり実用的な性能であると考える．

速度性能に加えて，今回は重力ベクトルによる画像補正方式を完成させ，安定して床面の直上から撮影したような画像を生成することができるようになった．これと同時に，撮影位置を計算する方法も導入し，参照画像上のどの位置から床画像を撮影したかがわかるようになった．この撮影位置同定機能を用いれば，撮影者の位置のトラッキングが可能になる．サンプルを図 2 に示したが，撮影位置を正確に推定し，マップできることが示された．

カメラで撮影しながら床照合ができることから，SLAM と同様の地図を作りながら位置推定を行うシステムも実現できる．実際に図 3 に複数画像の重ね合わせの試作機の実験結果を示した．エッジ画像の処理が不十分であり，現時点で利用価値は低いものの，丁寧に実装することで，地図作成システムに拡張できるものと考えている．

スマートフォンによる床照合システムは，予想以上に性能も高く，精度も高く実装できそうであるという初期実験結果を得た．応用として，屋内位置推定だけでなく，Ranger のように，屋外の地表面を用いた測位システムにも応用できることから，今後も性能と機能の充実を図りたい．

6.　結論

スマートフォン単体で動作する床指紋照合システムについて述べた．速度性能の実験では，1 枚の照合を 0.24 秒で行うことができることが示された．これは，実用的な速度と考えられ，今後の応用を含めて検討する．また，位置の連続的なトラッキングや，照合画像の合成についても可能性を示した．床画像データベースの容易な構築などに活用したいと考える．今後も，応用も含めて，性能と機能の充実を図りたいと考える．

謝辞　本研究は JSPS 科研費 JP17K00138 の助成を受けたものです．

参考文献

[1] Pablo Alcantarilla, Jesus Nuevo, and Adrien Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[2] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, Vol. 24, No. 6, pp. 381–395, June 1981.

[3] 藤井北斗, 渡辺仁. 低速自動走行車両による移動サービスシステム技術紹介. ヤマハ発動機技報, No. 53, 2017.

[4] 藤田悟, 内田薫. 床指紋を用いた位置推定. マルチメディア，分散，協調とモバイルシンポジウム（DICOMO2016）, pp. 1244–1250, 2016.

[5] 藤田悟, 藤田貴大, 内田薫. 床指紋：床の模様に基づく位置推定. 情報処理学会論文誌, Vol. 58, No. 12, pp. 2023–2033, 2017.

[6] 藤田悟, 内田薫. 床指紋照合における傾斜補正手法. マルチメディア，分散，協調とモバイルシンポジウム（DICOMO2018）, pp. 687–693, 2018.

[7] Satoru Fujita, Tomoko Fujita, and Kaoru Uchida. Floor fingerprint verification using a gravity-aware smartphone. In *Proceedings of the 5th IIAE International Conference on Intelligent Systems and Image Processing*, pp. 311–318, 2017.

[8] Kristopher Kozak and Marc Alban. Ranger: A ground-facing camera-based localization system for ground vehicles. In *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2016.

[9] Daniel Kurz and Selim Benhimane. Gravity-aware handheld augmented reality. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pp. 111–120. IEEE, 2011.

[10] Daniel Kurz and Selim Benhimane. Handheld augmented reality involving gravity measurements. *Computers & Graphics*, Vol. 36, No. 7, pp. 866 – 883, 2012.

[11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, Barcelona, 2011.

[12] T. Takahashi and R. Ishiyama. Fibar: Fingerprint imaging by binary angular reflection for individual identification of metal parts. In *Proc. of the 5th Int. Conf. on Emerging Security Technologies*, pp. 46–51, 2014.

[13] Linguang Zhang, Adam Finkelstein, and Szymon Rusinkiewicz. High-precision localization using ground texture. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.