Real-time and Continuous Floor Fingerprint Identification

Satoru Fujita*

Computer and Information Sciences, Hosei University, 3-7-2 Kajino-cho, Koganei, Tokyo 184-8586 Japan

*Corresponding Author: fujita_s@hosei.ac.jp

Abstract

This paper proposes a method to identify the floor fingerprint captured by a camera on a smartphone with limited computation resources. Images captured by the camera on the smartphone are fine enough to recognize microscopic features on the floor because recent smartphones equip high-resolution cameras, but the images are deformed because a user holds a phone by his/her hand. This paper introduces an algorithm to calculate a homography matrix from a gravity vector to rectify the poorly deformed floor image. Limited computation resource is another issue to identify images in real time. We implement cost-sensitive parts in native language C++, and select proper parameters, such as the number of extracted features, to identify floor fingerprint in a reasonable time. In order to determine the accurate user location, it is necessary to identify the angles and the translation against the reference image and identify the position from which the user take a picture image. We implement all these functions in a prototype system. Experimental results show that an image can be identified in a 0.28 seconds, and the accuracy of the position is within some centimeter errors. The prototype system can continuously track human positions, and can generate a map around the reference image.

Keywords: floor fingerprint, position estimation, feature matching, image rectification, smartphone.

1. Introduction

Position estimation is one of the most important technologies for providing locational services for customers. Although GPS is a widely used system to estimate the position, the usage is mostly limited to outdoor conditions since the signals from GPS satellites can hardly reach to the indoor spaces. Several types of indoor localization systems have been proposed, such as Wi-Fi localization, pedestrian dead reckoning and electro-magnetic fingerprint, but they also have disadvantages against their accuracy and cost. Our idea for the indoor position estimation is based on identification of floor surface textures. There are various floor textures, like wood-type, stone-type and plastic-type, and all these look similar or homogeneous when we look down the floor covered with monotonous textures, but they have intrinsic micro patterns when we look at the floor from a position very close to it. These microscopic patterns would be unique, because they are configured with grimes acquired afterwards even if the floor is printed as an industrial product. We call such microscopic and intrinsic patterns as floor fingerprint, a kind of the fingerprint of object(1)(2). If we have a floor map with the floor fingerprint and can identify the fingerprint reasonably in a reasonably short time, the accurate and low-cost position estimation system could be achieved.

Similar ideas have been proposed in a domain of autonomous driving or self-driving vehicles(3)(4). The vehicles equip many sensors attached at their bodies and control their tracking on the road. Since GPS only provide rough positions that involve ten meters wide errors, other sensors and cameras need to estimate the exact location and distances from the other vehicles. Ranger is a project that captures the ground textures on the road surface with high-speed camera with luminous LED, and identify the texture to determine the exact location within several centimeters errors(3). Ranger succeeds in identifying the images captured at the car moving in the speed of 130 km/h with ten frames per a second. This method can be applied to indoor autonomous carts carrying goods in logistic stations. Whereas the traditional way to control the tracks of the carts uses marker lines drawn on the floor. Ranger enables the vehicles to control without pre-drawn lines and increases the flexibility of their movement.

This paper proposes a method to identify the floor fingerprint captured by a camera on a smartphone with limited
computation resources on the phone\textsuperscript{5}\textsuperscript{6}\textsuperscript{7}. Images captured by the camera on the smartphone are fine enough to recognize microscopic features on the floor because recent smartphones equip high-resolution cameras, but the images might be deformed because a user holds the phone in a free posture and the phone has a tilt against the floor. Another limitation comes from computation resources. Our previous papers reported to achieve the identification speed in 0.5 seconds, but they used GPU on a desktop computer\textsuperscript{6}. We need to solve these poorly deformed image issues and the computer resource limitation. The advantage of using smartphone is the improvement of user interface. The smartphone can continuously capture images, analyze them to identify the location, and display the results on the screen. It is possible to generate a map of floor fingerprint at the same time when the location is identified, like simultaneous localization and mapping (SLAM)\textsuperscript{9}.

This paper, first, describes related work for detecting image features and image rectification. Then, we propose how to identify fingerprint on smartphone with limited computation resources. We evaluate its capability and performance through experiments and discuss the future directions.

2. Image Registration and Rectification

2.1 Image Registration

The general approaches to detect the identical objects in different images use feature-based matching algorithm that detects micro features at a small fragment of the image, finds matching pairs of the features between the images and verifies the correct matching pairs consistent to physically possible alignment. ORB\textsuperscript{9} and AKAZE\textsuperscript{10} are efficient algorithms to detect micro features, implemented in OpenCV libraries\textsuperscript{11}. They find and select notable featuring points, like corners and edges of objects appearing in the image, and generate feature descriptions that represent scale-invariant and rotation-invariant features for a small image fragment in a binary encoded digit. Since the main target of these algorithms is object detection in an image, corner and edge detection is very important and useful but the target for floor fingerprint is slightly different, because we would like to detect micro weak features on relatively monotonous floor surface. When applying these algorithms to the floor fingerprint identification, we need to make the threshold parameter for determining notable feature points lower and make the number of detected feature points larger. We also implemented an extension of ORB, called Balanced ORB, which could find more points with weak features on the plain surface but on the edge\textsuperscript{6}.

The next step for the image registration is a matching step where the feature descriptions in both images are compared with each other, and it finds the most suitable matching in terms of Hamming distances of their descriptions. After the matching, the matching pairs are verified to find a set of correct matching pairs. This process is the most important and cost-sensitive process for image registration. A random sample consensus, called RANSAC, is often used for selecting correct data from a data pool involving many noisy and irrelevant data\textsuperscript{12}. Assumed that a data pool involves correct data with a probability of $w$, and a problem requires $k$ independent data to generate a model, then the average number of iterations $E(k)$ to find a correct data becomes $1/w^k$. The function findHomography() in OpenCV randomly picks up four matching pairs $1/w^4$ times, generates homography matrix that maps the source points to destination points, and validates this homography matrix by applying it to other matching pairs and checking if the other matching pairs are correctly transformed by this matrix. This process works well if the probability of $w$ is reasonably high, but there are a very small set of correct pairs in floor image matching, because features are weak and most matching pairs are incorrect. In a sample case, $w$ is less than 0.03, then it requires more than 1,000,000 random samplings for finding a correct pair. Ranger avoids this situation by fixing camera position where the optical axis is vertical to the ground surface, and the distance between the camera and the surface is constant. In this case, there are no perspective distortion, and the scales of images are always the same. As a result, two matching pairs are sufficient enough to generate a transform matrix, and then 1,000 random samplings are good enough to verify.

2.2 Image Rectification

Whereas Ranger achieves high-speed matching for images captured by the camera whose posture against the ground is fixed, the posture in our case is uncontrollable because smartphone is held by hand. Images captured by hand held devices are distorted in a perspective view. If our system tries to find matching for such raw images, RANSAC requires four matching pairs and then the verification is very slow for finding the correct four pairs from a pool that involves correct pairs with probability of 0.03. Image rectification at a preprocessing stage is another approach to remove distortion from images. Smartphone
equips an accelerometer that measures acceleration of body movement and also measures a gravity vector from the device. Kurz introduced a way to determine homography matrix $H$ from the gravity, and rectified perspective floor images into non-perspective ones at preprocessing time. He named this scheme as gravity-rectified feature descriptors, called GREFD$^{13}$.$^{14}$. When the gravity vector is represented as $g = (g_x, g_y, g_z)^T$ in the local coordinate, it corresponds to $z = (0, 0, 1)^T$ in the global coordinate. Then, let $g_1$ and $g_2$ be two vectors defining a plane with a normal $g$. These two vectors can be defined as follows:

$$g_1 = (-g_z, 0, g_x)^T$$

$$g_2 = g \times g_1$$

Then the $(3 \times 3)$ matrix $H$ is defined as

$$H = (g_1 \, g_2 \, z)^T$$

After calculating $H$, they applied camera intrinsic parameter $K$ to $H$. Here, the most important thing is that the matrix $H$ can be calculated by using gravity $g$. GREFD succeeded in warping a floor image into ones in an orthogonal coordinate and comparing it with a corresponding flat floor image using RANSAC with a parameter $k = 3$. This scheme is not applicable to any planes except for the horizontal plane, but it is very efficient when we process floor images.

### 3. Floor Fingerprint Identification Implemented on Smart Phone

#### 3.1 Implementation Issues in Proposed Mechanism

Our research goal is to generate an implementation of real-time identification of floor fingerprint on smartphone held by user’s hand. The top problem is the performance issue with limited resources on smartphone. OpenCV project releases android versions of OpenCV libraries that can be easily installed and accessed from Java application development platform. Described more in detail, core parts of OpenCV are implemented in compiler language C++, and bridge classes that use Java native interface are provided for the development platform. It is useful if a programmer makes a small program that does not care of the performance, but the bridge interface pays higher cost to transfer array-type data between two languages. Our policy for implementation is so simple that cost-critical parts should be written in C++. For example, an extension of ORB algorithm is written in C++, because the feature detection step is a bottle neck for the performance of the total system, the extension strongly depends on the original ORB implementation and it accesses a lot of inner data of ORB, then it could be hard to be written through bridge interface. On the other hand, rectification for perspective distortion, described in the next section, is written in Java, since the rectification accesses sensor data provided in Java, generates a transformation matrix, and then uses the bridge interface for warping the image once. It accesses the bridge interface only once, so that the loss at the bridge is very small. Verification process for selecting consistent matching pairs is considerable. The process accesses a lot of inner data and cost-critical, but the specification of this process frequently changes, since the best verifying algorithm has not been fixed. We needed to implement two versions of the algorithm; a testing version in Java and a release version in C++.

Another issue for effectively using OpenCV is parameter selection. The default parameters are useful for notable object detection in images, but we need to find many weak features in relatively smooth surface. For example, the default number of feature points detected by ORB algorithm is set to 500, but our previous research that run on desktop computer was set to 10,000. The larger number leads to larger possibility to find correct identification, but the computation cost for matching features is roughly proportional to a square of the number. The section of the experiments discusses such parameter tuning in detail.

The current scheme for identifying floor fingerprint is implemented as follows;

(a) Preprocessing image

Rectification for removing perspective distortion based on gravity vector is implemented in Java.

(b) Feature detection

An extension of ORB, called Balanced ORB, is implemented in C++.

(c) Matching features

Brute-force matcher is provided in OpenCV. GPU implementation cannot be used on smartphone.

(d) Verification of matching pairs

Algorithm for finding analogous triangles in both images based on RANSAC is implemented in Java and C++.

(e) Warping image

Controlling warp and merge of images is implemented in Java. Actual warping is provided in OpenCV.

#### 3.2 Image Rectification by Gravity Sensor

Although our previous report explained how to remove perspective distortion from an image captured by a smartphone with a gravity sensor, the perspective distortion was limited to the pitch angle of the phone, but was not dealt
with the roll angle(7). This paper improves the rectification that can deal with both pitch and roll angles of the smartphone(8).

We, first, rewrite the gravity vector expressed in the local coordinate into ones in the global coordinate, as described in GREFD, but the vertical axis from the ground is the y-axis in our global coordinate but the z-axis defined in the GREFD. Our coordinate is the same as that used in the ARCore library by Google. When the gravity vector, which is the opposite direction vector to the y-axis in the global coordinate, is given as \( \mathbf{g} = (g_x, g_y, g_z)^\top \) in the local coordinate, this y-axis basis vector \( \mathbf{b}_y \) in the global coordinate is expressed in the local coordinate as:

\[
\mathbf{b}_y = \left( \frac{g_x}{G}, \frac{g_y}{G}, \frac{g_z}{G} \right)^\top
\]

where \( G = \sqrt{g_x^2 + g_y^2 + g_z^2} \) \( (4) \)

\( \mathbf{b}_x \), which is the x-axis basis vector, is not uniquely determined in the local coordinate, but if it is assumed that its y element is zero in the local coordinate, it is expressed as:

\[
\mathbf{b}_x = \left( \frac{g_x}{G}, 0, \frac{-g_z}{G} \right)^\top
\]

\( (5) \)

Finally, \( \mathbf{b}_z \), which is the z-axis basis vector, is calculated as the external product of \( \mathbf{b}_x \times \mathbf{b}_y \):

\[
\mathbf{b}_z = \mathbf{b}_x \times \mathbf{b}_y = \left( \frac{-g_y g_z}{G^2}, \frac{g_x g_z}{G^2}, \frac{-g_x g_y}{G^2} \right)
\]

\( (6) \)

Therefore, homography matrix \( \mathbf{H}_0 \) from the global coordinate to the local coordinate is described in the following equation:

\[
\mathbf{H}_0 = \begin{pmatrix}
    \mathbf{b}_x & \mathbf{b}_y & \mathbf{b}_z
\end{pmatrix}
\]

\[
= \begin{pmatrix}
    g_x/G & g_y/G & -g_z/G \\
    0 & -g_y/G & g_z/G \\
    -g_x/G & g_z/G & g_x/G
\end{pmatrix}
\]

\( (7) \)

On the other hand, the transformation matrix \( \mathbf{E} \) from the vector in the local coordinate to the global coordinate is expressed as an inverse matrix of \( \mathbf{H}_0 \).

\[
\mathbf{E} = \mathbf{H}_0^{-1}
\]

\( (8) \)

When the position of the smartphone in the global coordinate is \( \mathbf{p} \), that of the object is \( \mathbf{o} \), and the position of the object in the local coordinate is \( \mathbf{s} = (s_x, s_y, s_z)^\top \), then the following equation is accomplished.

\[
\mathbf{o} = \mathbf{p} + \mathbf{E} \mathbf{s}
\]

\( (9) \)

When screen coordinate is \( (u, v) \) where the u-axis is directed to the right and the v-axis is directed to the bottom, the resolutions of the screen is \( r_x \times r_y \), and the length of the real world for the height of the image taken by the camera at the one meter from the object is \( l \) meter, the vector \( \mathbf{s} \) is expressed by \( (u, v) \) as follows.

\[
\begin{pmatrix}
    s_x \\
    s_y \\
    s_z
\end{pmatrix} = \mathbf{L}_1 \begin{pmatrix}
    u \\
    v \\
    1
\end{pmatrix} s_z
\]

\( (10) \)

\[
\mathbf{L}_1 = \begin{pmatrix}
    -1/r_y & 0 & r_x l/2r_y \\
    0 & l/r_y & -l/2 \\
    0 & 0 & 1
\end{pmatrix}
\]

\( (11) \)

The crossing point of the ground and the eye vector \( \mathbf{e}_z = (e_{zx}, e_{zy}, e_{zz})^\top \) is defined as \( \mathbf{o}_0 \). The local coordinate of \( \mathbf{o}_0 \) is expressed as \( (0, 0, s_{0z})^\top \), if the height of the camera is \( h \) and \( s_{0z} \) is defined as follows.

\[
s_{0z} = -h/e_{zy}
\]

\( (12) \)

Therefore, \( \mathbf{o}_0 \) is expressed as follows.

\[
\mathbf{o}_0 = \mathbf{p} + e_x s_{0z}
\]

\( (13) \)

Next, the rectangle area on the floor around the crossing point \( \mathbf{o}_0 \) is determined. The coordinate of the rectangle area is expressed as \( (u', v') \). The direction of \( u' \) is \( (e_{zx}, 0, -e_{zz})^\top \) and that of \( v' \) is \( (e_{zy}, 0, e_{zz})^\top \). The vertical vectors in \( \mathbf{T} \) are divided by \( d = \sqrt{e_{zx}^2 + e_{zy}^2 + e_{zz}^2} \) respectively for making them unit vectors, then the \( \mathbf{o} \) on the floor is expressed as follows.

\[
\mathbf{o} = \mathbf{p} + \mathbf{T} \mathbf{L}_2 \begin{pmatrix}
    u' \\
    v'
\end{pmatrix}
\]

\( (14) \)

\[
\mathbf{L}_2 = \begin{pmatrix}
    1/l_{ry} & 0 & r_x l/2r_y \\
    0 & l/r_y & -l/2 \\
    0 & 0 & 1
\end{pmatrix}
\]

\( (15) \)

\[
\mathbf{T} = \begin{pmatrix}
    e_{zx}/d & e_{zx}/d & -e_{zx}/d2 \\
    0 & 0 & -e_{zy}/d2 \\
    -e_{zx}/d & e_{zz}/d & -e_{zz}/d2
\end{pmatrix}
\]

\( (16) \)

Eq.(10) must be equal to Eq.(14). Therefore, Equations (17) and (18) are obtained.

\[
\mathbf{E} \mathbf{L}_1 \begin{pmatrix}
    u \\
    v \\
    1
\end{pmatrix} = \mathbf{TL}_2 \begin{pmatrix}
    u' \\
    v'
\end{pmatrix}
\]

\( (17) \)

\[
\begin{pmatrix}
    u \\
    v \\
    1
\end{pmatrix} = \mathbf{L}_1 \mathbf{E} \mathbf{E}^{-1} \mathbf{TL}_2 \begin{pmatrix}
    u' \\
    v'
\end{pmatrix}
\]

\( (18) \)

Finally, homography matrix \( \mathbf{H}_{gr} \) for the gravity rectification is expressed as follows.

\[
\mathbf{H}_{gr} = (\mathbf{L}_1 \mathbf{E} \mathbf{E}^{-1} \mathbf{TL}_2)^{-1}
\]

\( (19) \)
Since Eq. (19) transforms the central image by the camera with a tilt angle, the resolution is lower than the part close to the bottom line of the original image. Therefore, another matrix $L_2'$ should be adopted as the transformation matrix for moving the target image to the bottom line of the original image.

$$L_2' = \begin{pmatrix} l/r_y & 0 & r_0 l/2 r_y \\ 0 & l/r_y & -l/2 + d' \\ 0 & 0 & 1 \end{pmatrix}$$ (20)

$d'$ is calculated in the following equations.

$$d' = \sqrt{m_x^2 + m_y^2 + m_z^2} - \frac{lh}{2}$$ (21)

$$\begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} = \begin{pmatrix} \mathbf{E}_{0.2}(r_1 - r_0) + \frac{r_1 l \mathbf{E}_{0.1}}{2} \\ \mathbf{E}_{1.2}(r_1 - r_0) + \frac{r_1 l \mathbf{E}_{1.1}}{2} \\ \mathbf{E}_{2.2}(r_1 - r_0) + \frac{r_1 l \mathbf{E}_{2.1}}{2} \end{pmatrix}$$ (22)

$$r_0 = -\frac{h}{\mathbf{E}_{1.2}}$$ (22)

$$r_1 = -\frac{h}{\mathbf{E}_{1.2} + l \mathbf{E}_{1.1}/2}$$ (23)

Homography matrix $H_{gr}$ with $L_2'$ can transform and rectify the image with perspective distortion into the orthogonal image taken from the position just above the floor.

3.3 Estimation of the Position Which Image Was Taken From

Since an image is taken by a tilted smartphone and rectified to the orthogonal one, the human position holding the smartphone is not right above the center of the image. The position just below the smartphone is calculated by replacing $L_2$ to $L_2'$ in Eq. (14), setting $u' = 0$ and $v' = 0$, and calculating the horizontal distance between eye position $p$ and the origin of the rectangle on the floor. This method can solve the exact human position within some centimeter errors.

4. Experiments

4.1 Capability Test

The capability test checks if floor fingerprint correctly identify the floor surface. The identified floor is expressed as a rectangle with red solid lines in the reference image, and the human position is expressed as a blue point, as shown in Fig. 1. When it succeeds in identifying the fingerprint, the execution times for component functions are represented in a toast screen.

![Fig. 1. A sample of floor fingerprint identification](image)

<table>
<thead>
<tr>
<th>Total</th>
<th>374.0</th>
<th>309.0</th>
<th>276.9</th>
<th>1062.2</th>
<th>2069.3</th>
<th>3391.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectification</td>
<td>113.6</td>
<td>69.2</td>
<td>51.4</td>
<td>109.0</td>
<td>148.3</td>
<td>133.2</td>
</tr>
<tr>
<td>Extraction</td>
<td>235.5</td>
<td>203.0</td>
<td>156.3</td>
<td>414.6</td>
<td>568.4</td>
<td>659.7</td>
</tr>
<tr>
<td>Matching</td>
<td>11.8</td>
<td>24.0</td>
<td>58.3</td>
<td>509.1</td>
<td>1294.9</td>
<td>2526.5</td>
</tr>
<tr>
<td>Validation</td>
<td>12.3</td>
<td>12.5</td>
<td>10.5</td>
<td>28.4</td>
<td>55.7</td>
<td>67.5</td>
</tr>
</tbody>
</table>

4.2 Performance Evaluation

The basic performance evaluation tests were conducted for the wood-style floor. The test scheme used feature extraction by B-ORB, feature matching by a brute force matcher, and matching-pair validation by the analogous triangle matching. Although we set the number of maximum feature points extracted by B-ORB as 10,000 in our early papers, we variably set that from 500 to 10,000 in these tests, because the computation resource is limited on smartphone and the number of feature points has a big impact for the performance. The tests were done on Pixel3 64GB Model, and the results are shown in Table 1. More than ten trials were done for each parameter, and the average values were shown in the table.

In the case of 500 feature points, the execution time for the feature extraction took more than that in the 2,000 feature case, because it spent more time to sort and select 500 points from the points found by a raw module of feature detections. The validation time also took more, because search iterations increased in RANSAC as a result of decreasing of the probability for finding correct matching pairs.

In the case of 2,000 feature points, execution time was the shortest of all cases. Especially, feature extraction took very small cost against the other cases. Therefore, we concluded this parameter was the most adequate in the experiments. The less feature points made the matching cost decreased to 58.3 milliseconds, and the total cost decreased.

In the cases of 6,000 and 10,000 feature points, the time for feature extraction and matching increased. The total times were five or six times larger than those running on desktop computer with GPU. Especially, feature matching
became ten times slower, and then it was a bottle neck for running on the smartphone.

As the final result, if we adopt matching with 2,000 feature points, image identification successfully works within 0.28 seconds. It is a reasonable performance to be used in location estimation with floor fingerprint identification.

4.3 Stability and Accuracy

We conducted the stability and accuracy experiments for position estimation. We kept standing at the same point, and holding a smartphone by hand, and positions were estimated more than 100 times during 50 seconds. The results are shown in Figures 2 and 3. Blue dots in the image in Fig. 2 were the location estimated. The dots were plotted at very close to each other. Graphs in Fig. 3 are histograms of the x and y axes. The standard deviations are 22.4 and 12.0 dots for x and y axes. Since 1 dot corresponds 0.625 millimeters, estimation errors are around 10 millimeters.

4.4 Tracking

The system can track a route of walking on the smartphone in real time. Fig.4 shows samples of tracking. Blue lines are the footprint. In every 0.3-0.5 seconds, the position was identified on the image of the floor. Fig.4(a) shows a walking on the tile floor, Fig.4(b) shows a walking around a center point and Fig.4(c) shows a walking for visiting three points.

Fig.5 is another sample of tracking across four images. In this case, almost four-times execution time for localization was taken, since the captured image must be compared with four images. Although the current version has no control for limiting and selecting reference images only around the current position, such control will be necessary when the floor fingerprint is applied to larger areas area. Fig.5 has less blue points at the boundary of the images, since identification was sometimes failed at the boundary. The reference images were separately applied to the identification, then the areas around the boundary were hard to detect correct matching pairs. This is a future issue. In these experiments, we walked slowly and held smartphone as stable as we can. The blur is another issue to be solved in future.

4.5 Mapping

The last experiment is to generate a map at the same time of the identification, like SLAM. We first took a floor image, and then continuously identified it to the other images taken afterwards. Fig.6 shows an example of the generated image, where several images are overlapped. The floor is covered with small pieces of tiles that help us to easily check the accuracy of the overlapping. More than 30 images are overlapped to generate a single image. We can find a lot of discontinuity in this image, but a whole structure is
consistent with the actual floor texture. This is an early stage experiment, but it can confirm the capability of our fast identification method of floors.

5. Discussion

This paper proposes a method to identify the floor fingerprint on smartphone and evaluates it in experiments. Although the computer resource limitation on the smartphone is very severe, we achieved to identify an image in 0.28 seconds by implementing main parts in native language C++ and tuning parameters for the number of feature points extracted.

In addition to the performance, we completed to design a rectification algorithm for perspective distortion using a gravity sensor. This algorithm works better than the GREFD proposed by Kurz and the previous method proposed by us, because it can rectify an image from a camera in a posture with roll and pitch angles. We also clarified a method to determine the relative position where the picture was taken. This method enables our system to estimate user’s position within some centimeter errors.

It is possible to simultaneously identify the floor fingerprint of the image and generate a map around the reference image, like SLAM. Although the current version only overlaps images on the reference image and does not adjust errors at the edges connected to the other images, it could be expanded to an automatic map generator in future.

The performance of the floor fingerprint identification on smartphone was better than the expectation. We can identify the location three of four times in a second. The application areas are not only a position estimation for humans but also that for vehicles. In the current version, image blur is a big issue that makes image matching failed, and then we need more study on these remaining issues.

6. Conclusions

This paper proposed a method to identify the floor fingerprint using limited resources on smartphone. Finally, we achieved to identify an image in 0.28 seconds, involving image rectification, feature extraction, feature matching, and verification of matching pairs. It was reasonably fast for applying it to the actual application on the smartphone. We developed a user tracker and a map generator that work in real time on smartphone. In future, we will improve its performance and additional functions.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP17K00138.

References

