# Chapter 5
# Image-Based Positioning of Mobile Devices in Indoor Environments

**Jason Zhi Liang, Nicholas Corso, Eric Turner and Avideh Zakhor**

## 5.1 Introduction

Indoor positioning allows for many commercially viable applications, such as navigation, behavior and movement tracking, and augmented reality (AR). These applications all require the user's location and orientation to be reliably estimated. Positioning is noticeably more challenging indoors than outdoors since GPS is typically unavailable in interior environments due to the shielding effect of structures. As a result, much research has been focused on relying on other types of signals, or in our case, images as a basis for positioning.

A variety of sensors are capable of performing indoor positioning, including image [1], optical [2], radio [3–7], magnetic [8], RFID [9], and acoustic [10]. WiFi-based indoor positioning takes advantage of the proliferation of wireless access points (AP) and WiFi capable smartphones and uses the signal strength of nearby WiFi beacons to estimate the user's location. A few drawbacks are that APs cannot be moved or modified after the initial calibration, and that a large of number of APs are required to achieve reasonable accuracy. For instance, 10 or more wireless hotspots are typically required to achieve submeter accuracy [4]. The most debilitating drawback of WiFi positioning is its inability to estimate the user's orientation, which is necessary for AR applications. Other forms of indoor positioning that rely on measuring radio signal strength such as bluetooth, GSM, and RFID, also share the same strengths and weaknesses of WiFi-based indoor positioning.

There have also been previous attempts at indoor image-based positioning [1]. An image-based positioning system involves retrieving the best image from a database

J.Z. Liang (✉) · N. Corso, E. Turner · A. Zakhor
Department of EECS, UC Berkeley, Berkeley, CA, USA
e-mail: jasonzliang@eecs.berkeley.edu

N. Corso
e-mail: ncorso@eecs.berkeley.edu

E. Turner
e-mail: elturner@eecs.berkeley.edu

A. Zakhor
e-mail: avz@eecs.berkeley.edu



J. Choi and G. Friedland (eds.), *Multimodal Location Estimation of Videos and Images*,
DOI 10.1007/978-3-319-09861-6_5

that matches to the user's query image, then performing pose estimation on the query/database image pair in order to estimate the location and orientation of the query image. The authors in [1] take advantage of off the shelf image matching algorithms, namely color histograms, wavelet decomposition, and shape matching and achieve room level accuracy with more than 90 % success probability, and meter-level accuracy with more than 80 % success probability for one floor of the computer science building at Rutgers University. This approach however, cannot be used to determine the absolute metric position of the camera, nor its orientation. Thus, it cannot be used in augmented reality applications where precise position and orientation is needed.

In this chapter, we demonstrate an image-based positioning system for mobile devices capable of achieving submeter position accuracy as well as orientation recovery. The three stages of that pipeline are: (1) preparing a 2.5D locally referenced image database, (2) image retrieval, and (3) pose recovery from the retrieved database image. We also present a method to estimate confidence values for both image retrieval and pose estimation of our proposed image-based positioning system. These two confidence values can be combined to form an overall confidence indicator. Furthermore, the confidence values for our pipeline can be combined with that of other sensors such as WiFi in order to yield a more accurate result than each method by itself.

Our pipeline can be summarized as follows:

1. Database Preparation, shown in Fig. 5.1a: We use a human operated ambulatory backpack outfitted with laser scanners, cameras, and an orientation sensor (OS), as seen in Fig. 5.2, to map the interior of a building in order to generate a locally referenced 2.5D image database complete with SIFT features [11–13]. By locally referenced image database, we mean that the absolute six degrees of freedom pose of all images, i.e., $x$, $y$, $z$, yaw, pitch, and row, are known with respect to a given coordinate system. By 2.5D, we mean that for each database image, there is a sparse depthmap that associates depth values with image SIFT keypoints only.
2. Image Retrieval, shown in Fig. 5.1b: We load all of the image database SIFT features into a k-d tree and perform fast approximate nearest neighbor search to find a database image with most number of matching features to the query image [14–16].
3. Pose Estimation, shown in Fig. 5.1c: We use the depth of SIFT feature matches along with cell phone pitch and roll to recover the relative pose between the retrieved database image in step (2) and the query image. This results in complete six degree of freedom pose for the query image in the given coordinate system [17].

In Sect. 5.2, we describe our approach for generating sparse depthmaps during database preparation. In Sect. 5.3, we will go over image retrieval and pose estimation. Section 5.4 includes estimation of confidence values for both image retrieval and pose estimation are estimated. In Sect. 5.5, we show experimental results, characterizing the accuracy of our pipeline. Section 5.6 includes conclusions and future work.

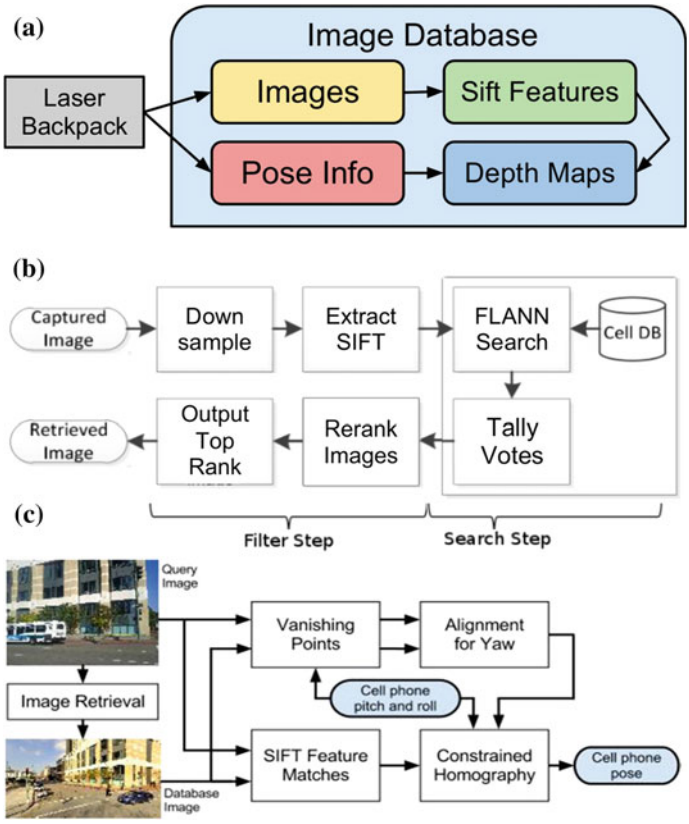


**Fig. 5.1** Overview of our indoor positioning pipeline. The pipeline is composed of **a** database preparation, **b** image retrieval, and **c** pose estimation steps

## 5.2 Database Preparation

In order to prepare the image database, an ambulatory human operator first scans the interior of the building of interest using a backpack fitted with two 2D laser scanners, two fish-eye cameras, and one OS as shown in Fig. 5.2. The database acquisition system requires two laser scanners, namely the pitch and yaw scanners in Fig. 5.2. Measurements from the backpack's yaw and pitch laser range scanners are processed by a scan matching algorithm to localize the backpack at each time step and recover its six degrees of freedom pose [18]. Specifically, the yaw scanner is used in conjunction with a 2D positioning algorithm in [11–13] to recover x, y and yaw, the OS is used to recover pitch and roll, and the pitch scanner is used to recover z [11]. Since the cameras are rigidly mounted on the backpack, recovering the backpack pose essentially implies recovering camera pose. Figure 5.3a shows the recovered path of the backpack within a shopping center, while Fig. 5.3b shows

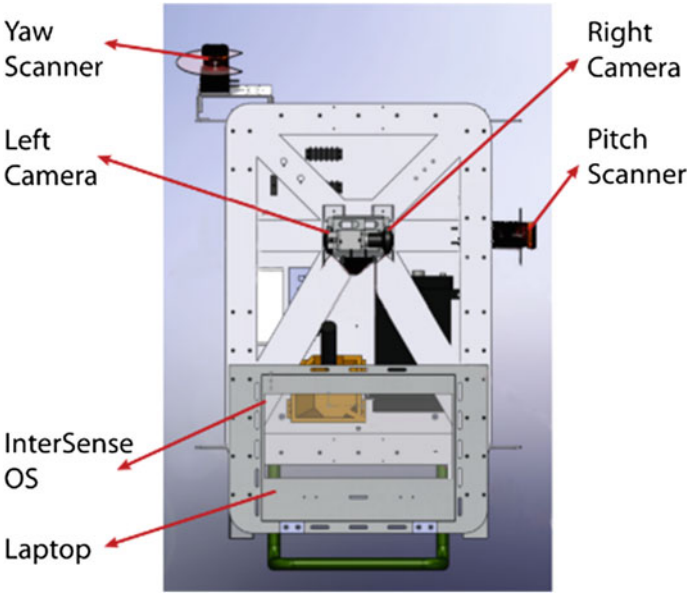


**Fig. 5.2** Diagram of the data acquisition backpack

the surrounding wall points recovered by the backpack by projecting the yaw scans onto the ground plane [19]. These wall points can be connected interactively via commercially available CAD software to produce an approximate 2D floorplan of the mall as seen in Fig. 5.3c. The recovered pose of the rigidly mounted cameras on the backpack are then used to generate a locally referenced image database in which the location, i.e., $x$, $y$, and $z$, as well as orientation, i.e., yaw, pitch, and roll, of each image is known within one coordinate system.

To create a sparse depthmap for the database images, we first temporally subsample successive captured images on the backpack while maintaining their overlap. We then extract SIFT features from each pair of images and determine matching feature correspondence pairs through nearest neighbor search. In order to ensure the geometric consistency of the SIFT features, we compute the fundamental matrix that relates the two sets of SIFT features and removes any feature pairs which do not satisfy epipolar constraints.

We then triangulate matching SIFT keypoint pairs in 3D space. As seen in Fig. 5.4, for each pair of SIFT correspondences, we calculate the 3D vectors that connects the camera centers of the images to the respective pixel locations of their SIFT features. In doing so, we make use of the database images' poses and intrinsic parameters to ensure both vectors are correctly positioned within the same world coordinate frame. Next, we determine the depth of the SIFT features by finding the intersection of these rays and computing the distance from camera center to the intersection point. We

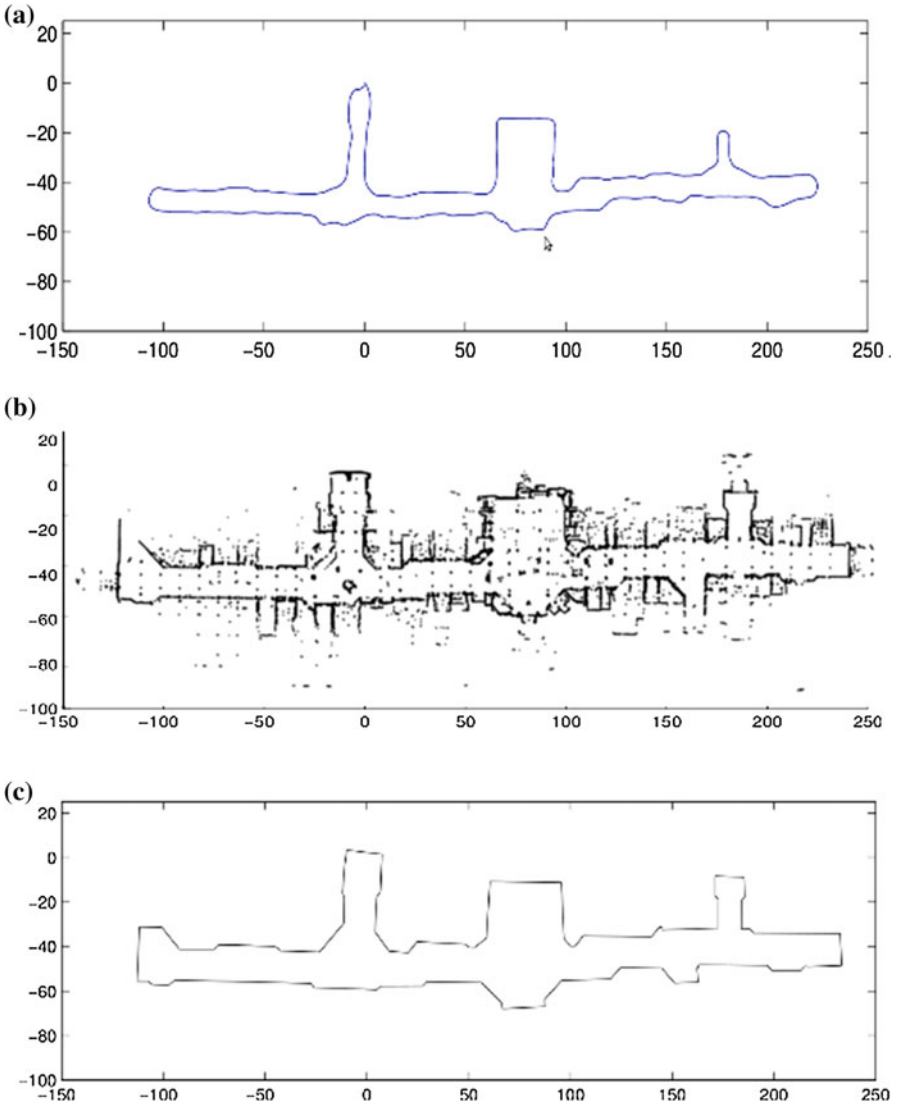


**Fig. 5.3** **a** Recovered path of backpack traversal. **b** Wall points generated by backpack. **c** 2D floorplan recovered from wall points

use the following to determine the intersection point or the point mutually closest to the two vectors:

$$x = \left( \sum_{i=1}^{2} I - v_i v_i^T \right)^{-1} \left( \sum_{i=1}^{2} \left( I - v_i v_i^T \right) p_i \right) \tag{5.1}$$

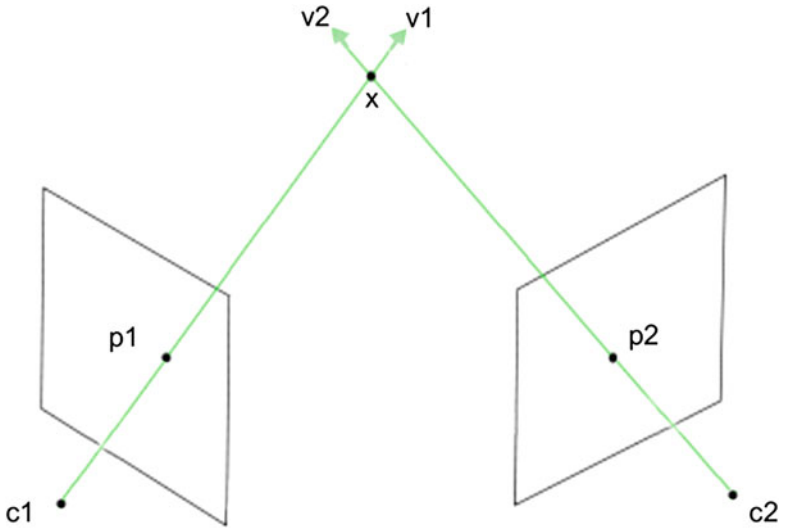


**Fig. 5.4** Triangulation of two matching SIFT features. v1 and v2 are the resulting vectors when the camera centers c1 and c2 are connected to the SIFT features p1 and p2 on the image planes. The two vectors intersect at x

where $x$ is the intersection point, $v_i$ is the normalized direction of the $i$th vector, and $p_i$ is a point located on the $i$th vector. The availability of highly optimized library functions for determining fundamental matrices and performing linear algebra operations means that sparse depthmap generation can be done in a matter of seconds per image. For debugging and visualization purposes, we combine the intersection points of SIFT features from every database image into a single sparse 3D point cloud, shown in Fig. 5.5a, b.

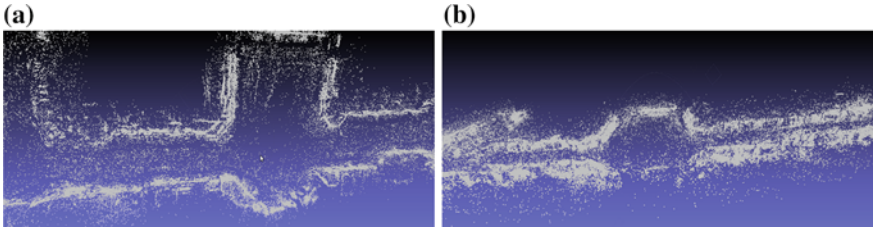


**Fig. 5.5** **a** *Top-down* and **b** *side views* of sparse 3D point cloud generated from triangulation of SIFT feature correspondence of the database images

## 5.3 Image Retrieval and Pose Estimation

The next step of our image-based positioning pipeline shown in Fig. 5.1c is image retrieval, which involves selecting the best matching image from the image database for a particular query image. Our indoor image retrieval system loads the SIFT features of every database image into a single k-d tree [16]. Next, we extract SIFT features from the query image and for each SIFT vector extracted, we lookup its top N neighbors in the kd-tree. For each closest neighbor found, we assign a vote to the database image that the closest neighbor feature vector belongs to. Having repeated this for all the SIFT features in the query image, the database images are ranked by the number of matching SIFT features they share with the query image.

After tallying the votes, we check geometric consistency and rerank the scores to filter out mismatched SIFT features. We then solve for the fundamental matrix between the database and query images and eliminate feature matches that do not satisfy epipolar constraints [14]. We also remove SIFT feature matches where the angle of SIFT features differ by more than 0.2 rad. Since these geometric consistency checks only eliminate feature matches and decrease the scores of database images, we only need to partially rerank the database images. The database image with the highest score after reranking is exported as the best match to the query image. The image retrieval step takes roughly 2–4 s depending on the processing power of the processor used.

As shown in Fig. 5.1c, the last step of our indoor positioning pipeline is pose recovery of the query image. Pitch and roll estimates from cell phone sensors are used in vanishing point analysis to compute yaw of the query image [17]. Once we estimate orientation, SIFT matches are used to solve a constrained homography problem within Random Sample Consensus (RANSAC) to recover translation between query and database images. The method for scale recovery of the translation vector only requires depth values at the SIFT features which are considered inliers from the RANSAC homography. These depth values are present in the sparse depthmaps generated during the database preparation step of Sect. 5.2. We have also found that reducing the size of the query images significantly reduces the number of iterations required for RANSAC homography. This is because the resolution of our database images is significantly lower than that of the query image camera. If the RANSAC homography fails to find inliers, we use the pose of the matched database image as the solution. Depending on the image and speed of the processor, pose estimation requires 2–10 s.

## 5.4 Confidence Estimation

Our confidence estimation system consists of several classifiers that output confidence values for both the image retrieval and pose recovery steps in our pipeline. These classifiers are trained using positive and negative examples from both image

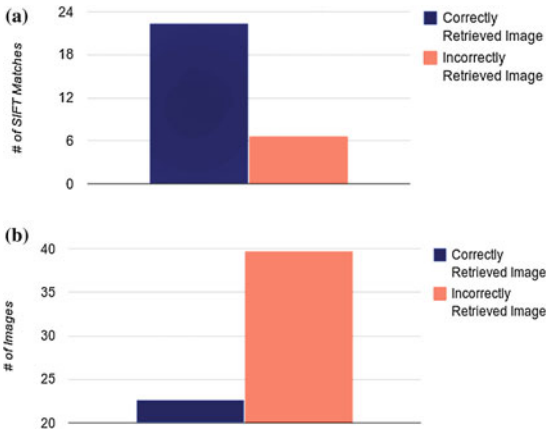


**Fig. 5.6** Comparison of **a** number of SIFT matches after geometric consistency check and **b** vote ranking distribution before geometric consistency check for correctly and incorrectly retrieved images

retrieval and pose recovery stages of our proposed pipeline in Sect. 5.3. We have empirically found a logistic regression classifier to perform reasonably well even though other classifiers can also be used for confidence estimation. In order to evaluate the performance of our confidence estimation system, we create a dataset of over 270 groundtruth images where roughly 25 % are used for validation and the rest for training. To boost classifier performance, 50 out of the 270 images in the validation set are chosen to be "negative" images that do not match to any image database.

To generate confidence values for image retrieval, we train a logistic regression classifier based on features obtained during the image retrieval process. We assign groundtruth binary labels to the images in the training set that indicate whether the retrieved images matches the query images. For a given query image, the retrieval classifier generates both a predicted binary label and a retrieval confidence value between 0 and 1. We have found the following features to be well correlated with image retrieval accuracy [14]: (a) number of SIFT feature matches between query and database image before geometric consistency checks; (b) number of SIFT matches after geometric consistency checks; (c) the distribution of the vote ranking before geometric consistency checks; (d) the distribution of the vote ranking after geometric consistency checks; (e) physical proximity of the top ranking database images in the vote ranking. For example, as shown in Fig. 5.6a, the average number of SIFT matches after geometric consistency checks for correctly matched query images is over three times that of incorrectly matched query images. Likewise, as shown in Fig. 5.6b, the number of database images with at least half the number of votes of the top ranked image before the geometric consistency check is much lower for correctly retrieved images than the incorrectly retrieved ones.

Similarly for pose estimation, we train a separate logistic regression classifier on another set of features that correlate well with the pose recovery accuracy. We assign a groundtruth "True" label if the location error of a training image is below a pre-specified threshold of 4 m, and a "False" label otherwise. As with the image retrieval classifier, our pose estimation classifier generates a predicted binary label

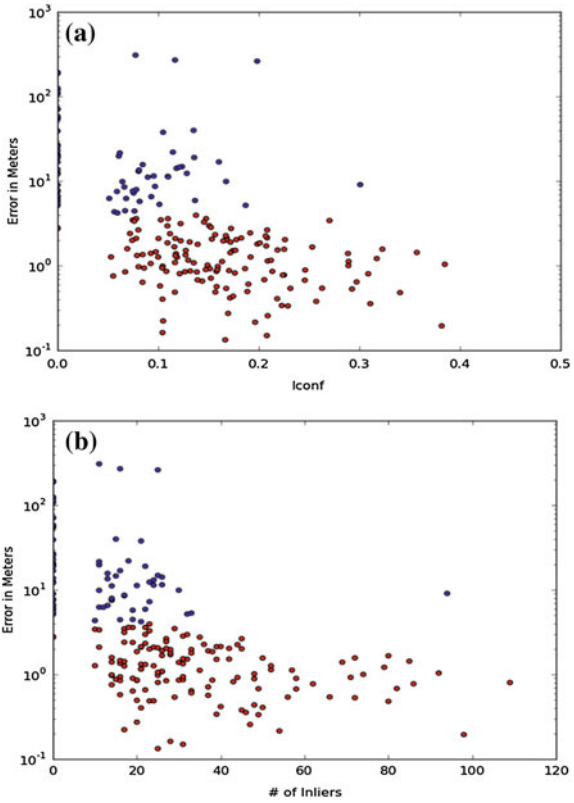


**Fig. 5.7** Scatterplot of a confidence metric used in [17] and **b** number of inliers after RANSAC homography versus location error. *Red* (*blue*) *dots* correspond to images with less (more) than 4 m of location error

and a confidence value between 0 and 1. The features use to train the classifier are: (a) number of inliers after RANSAC homography; (b) reprojection error; (c) number of SIFT feature matches before RANSAC homography; (d) number of RANSAC iterations; (e) a confidence metric in [17] that is used to choose the optimal inlier set. In Fig. 5.7, we use scatterplots to visualize the correlation between some of these features and pose recovery accuracy. Specifically, Fig. 5.7a plots the relationship between the confidence metric used to choose the optimal inlier set and location error of the pose estimation while Fig. 5.7b does the same for the number of inliers remaining after RANSAC homography and location error. The red (blue) dots in the scatterplots correspond to images with less (more) than 4 m of location error. As seen, query images with larger location error tend to have less inliers and a smaller inlier set confidence metric.

We also perform support vector regression (SVR) on the training set and use the resulting regression model to predict location error of the testing set for our proposed

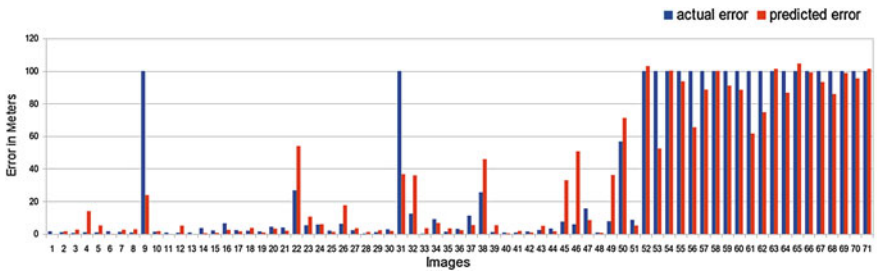


**Fig. 5.8** Plot of actual (*blue*) versus predicted (*red*) location error for images in validation set using SVR regression. For the negative examples in the validation set, we set the actual error to be an arbitrary high value of 100 m

pose recovery method. In doing so, we assign an arbitrarily large location error of 100 m to the negative examples in the validation set. As seen in Fig. 5.8, there is a reasonable correlation between our predicted and actual location error.

We find the predicted binary label of the image retrieval and pose estimation confidence system to be in agreement with the actual groundtruth label 86 and 89 % of the query images in the validation set respectively. Figure 5.9a, b show the distribution of confidence values for image retrieval and pose estimation respectively on the validation set. Green (red) bars represent the confidence distribution of images whose predicted label (do not) match the groundtruth label. To create an overall system confidence score between 0 and 1 and prediction label, we use the following algorithm below:

```
overall confidence = 0.5 * (retrieval confidence + pose confidence);
if overall confidence >0.5 then
  | prediction label = true;
else
  | prediction label = false;
end
```

By comparing the groundtruth and the overall confidence prediction labels for the query images in the validation set, the accuracy of the overall confidence estimation is determined to be 86 %. Figure 5.9c shows the distribution of overall confidence scores for the validation set.

To determine the optimal location error threshold, we set it to values ranging from 1 to 12 m and test the accuracy of the pose estimation confidence system. As shown in Fig. 5.10, the optimal value for the threshold is around 3–5 m.

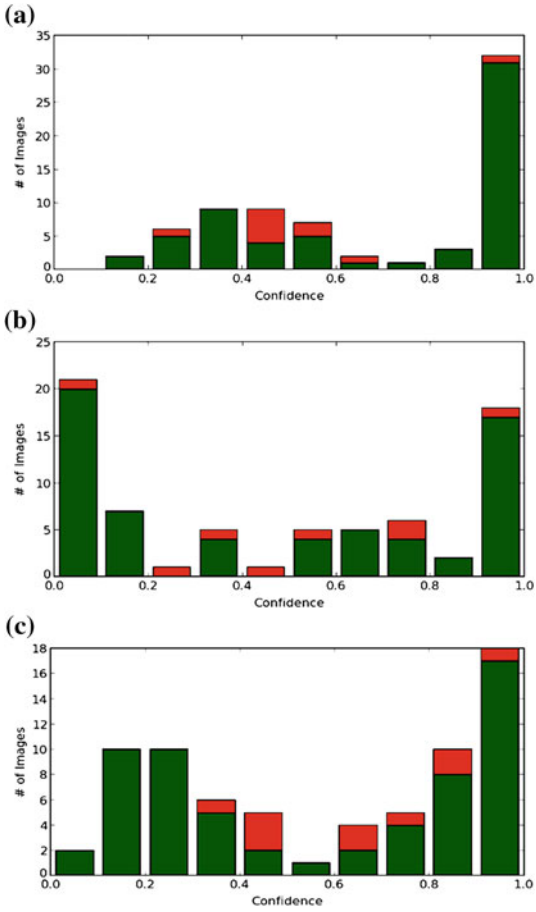


**Fig. 5.9** Confidence distribution for **a** image retrieval, **b** pose recovery, **c** overall system on the validation set; *Red* (*green*) *bars* correspond to incorrectly (correctly) predicted images

## 5.5 Experimental Results

For our experimental setup, we use the ambulatory human operated backpack of Fig. 5.2 to scan the interior of a two story shopping center located in Fremont, California. To generate the image database, we collect thousands of images with two 5 megapixel fish-eye cameras mounted on the backpack. These heavily distorted fish-eye images are then rectified into 20,000 lower resolution rectilinear images. Since the images overlap heavily with each other, it is sufficient to include every sixth image for use in the database. By reducing the number of images, we are able to speed up image retrieval by several factors with virtually no loss in accuracy.

Our query image data set consists of 83 images taken with a Samsung Galaxy S3 smartphone. The images are approximately 5 megapixels in size and are taken using the default settings of the Android camera application. Furthermore, the images

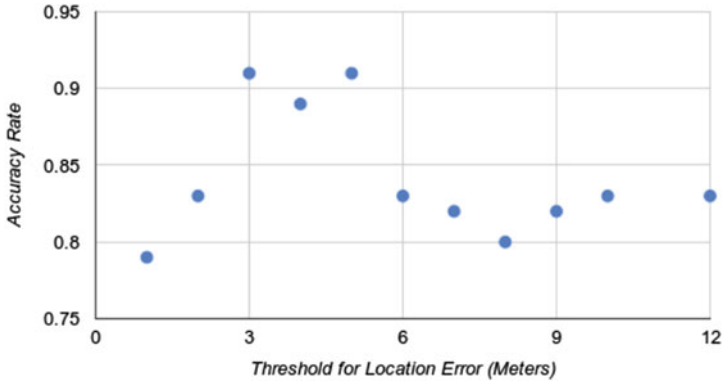


**Fig. 5.10** Scatterplot showing relationship between pose recovery confidence estimation accuracy and the threshold $t_s$ used for location error

consist of landscape photos either taken head-on in front of a store or at a slanted angle of approximately 30°. After downsampling the query images to the same resolution as the database images, i.e., 1.25 megapixels, we successfully match 78 out of 83 images to achieve a retrieval rate of 94 %. Detailed analysis of the failure cases reveal that two of the incorrectly matched query images correspond to a store that does not exist in the image database. Therefore, the effective failure rate of our image retrieval system is 3 out of 80 or less than 4 %. As shown in Fig. 5.11a, successful retrieval usually involves matching of store signs present in both the query and database images. In cases such as Fig. 5.11b where retrieval fails, there are few matched features on the query image's store sign.

Next, we run the remaining query images with successful retrieved database images through the pose estimation part of the pipeline. In order to characterize pose estimation accuracy, we first manually groundtruth the pose of each query image taken. Groundtruth is estimated by using the 3D model representation of the mall, and distance and yaw measurements recorded during the query dataset collection. We first locate store signs and other distinctive scenery of the query image within the


**Fig. 5.11** **a** Successful and **b** unsuccessful examples of image retrieval. *Red lines* show SIFT feature matches

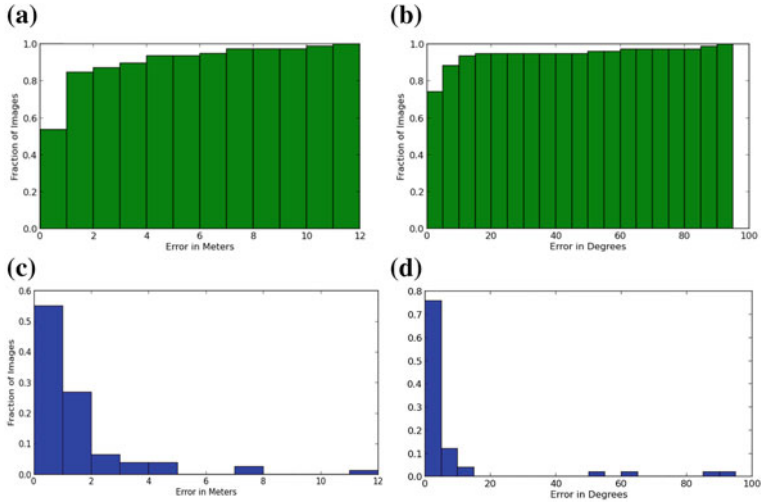


**Fig. 5.12** Cumulative density function of **a** location, **b** yaw error of and probability density function of **c** location, **d** yaw error of our indoor positioning pipeline

3D model to obtain a rough estimate of the query image pose, which is then refined using the measurements. The resulting groundtruth values are in the same coordinate frame as the output of the pose recovery step.

Figure 5.12 summarizes the performance of the pose estimation stage of our pipeline. Figure 5.12a, b show the cumulative distribution functions of location and yaw error respectively while Fig. 5.12c, d show the probability distribution functions of location and yaw error. As we can see, over 80 % of the images have their yaw correctly estimated to within 10° of the groundtruth values. Furthermore, over 55 % of all the images have a location error of less than 1 m. As seen in the example in Fig. 5.13a, when the location error is less than 1 m, the SIFT features of corresponding store signs present in both query and database images are matched by the RANSAC homography [17]. Conversely, in less accurate cases of pose estimation where the location error exceeds 4 m, the RANSAC homography finds "false matches" between unrelated elements of the query and database images. For instance in Fig. 5.13b, different letters in the signs of the two images are matched. In general, we find that images with visually unique signs perform better during pose estimation than those lacking such features.

On a 2.3 GHz i5 laptop, our complete pipeline from image retrieval to pose recovery takes on average 10–12 s to run. On an Amazon EC2 extra-large computing instance, the runtime is reduced further to an average of 4.5 s per image. The individual runtimes for each image is highly variable, with some images taking twice as long as the average time.


**Fig. 5.13** **a** Example of accurate pose estimation on query image. **b** Example of inaccurate pose estimation. Notice how different letters in the same sign are matched

## 5.6 Conclusion

In this chapter, we have presented a data acquisition system and processing pipeline for image-based positioning in indoor environments. Several possible improvements to our image-based positioning pipeline include tracking the position of the user and reducing the amount of noise in the depthmaps by utilizing more images for the sparse depthmap generation process. For future research, we are planning to examine ways to further increase the accuracy of indoor positioning. One method we are exploring is to combine our image-based indoor positioning pipeline with a WiFi-based indoor positioning system. The final position is determined by a particle filter that receives measurement updates from both positioning systems.

## References

1. N. Ravi, P. Shankar, A. Frankel, A. Elgammal, L. Iftode, Indoor localization using camera phones, in *Mobile Computing Systems and Applications* (2006)
2. L.I.U. Xiaohan, H. Makino, M.A.S.E. Kenichi, Improved indoor location estimation using fluorescent light communication system with a nine-channel receiver. IEICE Trans. Commun. **93**(11), 2936–2944 (2010)

3. Y. Chen, H. Kobayashi, Signal strength based indoor geolocation, in *International Conference on Communications* (2002)
4. J. Biswas, M. Veloso, WiFi localization and navigation for autonomous indoor mobile robots, in *International Conference on Robotics and Automation* (2010)
5. G. Fischer, B. Dietrich, F. Winkler, Bluetooth indoor localization system, in *Proceedings of the 1st Workshop on Positioning, Navigation and Communication* (2004)
6. S.S. Chawathe, Low-latency indoor localization using bluetooth beacons, in *12th International IEEE Conference on Intelligent Transportation Systems* (2009)
7. A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, V. Otsason, GSM indoor localization. Perv. Mobile Comput. **3**(6), 698–720 (2007)
8. J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, M. Wiseman, Indoor location sensing using geo-magnetism, in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, pp. 141–154 (2011)
9. S. Schneegans, P. Vorst, A. Zell, Using RFID snapshots for mobile robot self-localization, in *European Conference on Mobile Robots* (2007)
10. H.-S. Kim, J.-S. Choi, Advanced indoor localization using ultrasonic sensor and digital compass, in *International Conference on Control, Automation and Systems* (2008)
11. G. Chen, J. Kua, S. Shum, N. Naikal, M. Carlberg, A. Zakhor, Indoor localization algorithms for a human-operated backpack system, in *3D Data Processing, Visualization, and Transmission*, May 2010
12. T. Liu, M. Carlberg, G. Chen, J. Chen, J. Kua, A. Zakhor, Indoor localization and visualization using a human-operated backpack system, in *International Conference on Indoor Positioning and Indoor Navigation* (2010)
13. J. Kua, N. Corso, A. Zakhor, Automatic loop closure detection using multiple cameras for 3D indoor localization, in *IS&T/SPIE Electronic Imaging* (2012)
14. J. Zhang, A. Hallquist, E. Liang, A. Zakhor, Location-based image retrieval for urban environments, in *International Conference on Image Processing* (2011)
15. D.G. Lowe, Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
16. M. Muja, D.G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in *International Conference on Computer Vision Theory and Applications* (2009)
17. A. Hallquist, A. Zakhor, Single view pose estimation of mobile devices in urban environments, in *Workshop on the Applications of Computer Vision* (2013)
18. N. Corso, A. Zakhor, Indoor localization algorithms for an ambulatory human operated 3D mobile mapping system. Remote Sensing **2013**, 6611–6646 (2013)
19. E. Turner, A. Zakhor, Watertight as-built architectural floor plans generated from laser range data, in *Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission* (2012)
20. E. Turner, A. Zakhor, Watertight planar surface meshing of indoor point clouds with voxel carving, in *3D Vision*, Seattle, June 2013
21. R. Hartley, A. Zisserman, *Multiple View Geometry* (Cambridge University Press, Cambridge, 2004)