Low-Cost Asset Tracking using Location-Aware Camera Phones

David Chen[a], Sam Tsai[a], Kyu-Han Kim[b],
Cheng-Hsin Hsu[b], Jatinder Pal Singh[b], Bernd Girod[a]

[a]Department of Electrical Engineering, Stanford University, Stanford, CA 94305
{dmchen, sstsai, bgirod}@stanford.edu

[b]Deutsche Telekom R&D Laboratories, 5050 El Camino Real 221, Los Altos, CA 94022
{kyu-han.kim, cheng-hsin.hsu, j.singh}@telekom.com

ABSTRACT

Maintaining an accurate and up-to-date inventory of one's assets is a labor-intensive, tedious, and costly operation. To ease this difficult but important task, we design and implement a mobile asset tracking system for automatically generating an inventory by snapping photos of the assets with a smartphone. Since smartphones are becoming ubiquitous, construction and deployment of our inventory management solution is simple and cost-effective. Automatic asset recognition is achieved by first segmenting individual assets out of the query photo and then performing bag-of-visual-features (BoVF) image matching on the segmented regions. The smartphone's sensor readings, such as digital compass and accelerometer measurements, can be used to determine the location of each asset, and this location information is stored in the inventory for each recognized asset.

As a special case study, we demonstrate a mobile book tracking system, where users snap photos of books stacked on bookshelves to generate a location-aware book inventory. It is shown that segmenting the book spines is very important for accurate feature-based image matching into a database of book spines. Segmentation also provides the exact orientation of each book spine, so more discriminative upright local features can be employed for improved recognition. This system's mobile client has been implemented for smartphones running the Symbian or Android operating systems. The client enables a user to snap a picture of a bookshelf and to subsequently view the recognized spines in the smartphone's viewfinder. Two different pose estimates, one from BoVF geometric matching and the other from segmentation boundaries, are both utilized to accurately draw the boundary of each spine in the viewfinder for easy visualization. The BoVF representation also allows matching each photo of a bookshelf rack against a photo of the entire bookshelf, and the resulting feature matches are used in conjunction with the smartphone's orientation sensors to determine the exact location of each book.

Keywords: Visual search, location sensing, inventory management

1. INTRODUCTION

Accurately tracking a large set of assets is a difficult, time-consuming task for many people and institutions. Asset inventories are typically maintained by manual updates or deployments of barcode or radio-frequency identification (RFID) technologies. Manual updates are costly in terms of human labor and error-prone. Expensive barcode or RFID systems, on the other hand, may only be affordable to large institutions and require physically attaching a marker to each asset. Given these disadvantages, a more efficient and affordable asset tracking system is desired.

Modern smartphones are equipped with auto-focus cameras, orientation sensors, and high-speed network access. These capabilities of smartphones make them an attractive choice for automated asset tracking. In our asset tracking system, the user simply snaps a few photos with a smartphone of the assets that should be indexed. The pictures are transmitted from the phone to a processing server. On the server, robust image-based features are extracted from the query photos and matched against a large visual database of previously indexed assets or products. Matching these robust features enables automatic identification of assets in the query photos. Subsequently, meta data such as the name, description, price, and manufacturer for each asset can be easily retrieved from the Internet and transferred into the user's inventory database. When the smartphone receives the response from the server, the recognized assets can also be highlighted in the phone's viewfinder. Suppose

a librarian is interested in creating an inventory of all the books in the library. Fig. 1 shows the response of our system to a query snapshot of a bookshelf. Each recognized book spine is identified and highlighted in the smartphone's viewfinder, to provide instant feedback to the librarian about the books being added into the inventory.

Figure 1. Screenshot of a smartphone's viewfinder showing an identified book asset.

In asset tracking, the location of each recognized asset is equally as important to know as the identity of the asset. As the user is taking photos, orientation sensors on the smartphone provide valuable information about the surrounding environment. For example, digital compass readings indicate which direction in a room the user is facing, while accelerometer readings indicate recent vertical and horizontal movements. These sensor readings are transmitted to the processing server, where a location inference algorithm analyzes the readings to determine the location where each query photo is taken. Initial location estimates based on sensor readings can be further refined by matching objects in the query photos against previously photographed objects with known location and pose. The location information is combined with the asset identification and meta data in the user's inventory database.

Despite the advantages of using smartphones for asset tracking, there are several important challenges. For image analysis, assets shown in a query photo may only have small areas of visibility, making it difficult to extract a sufficient number of image features for recognition. From the perspective of any single asset, the image features for all the other assets shown in the query photo essentially act as clutter, and current feature-based image matching methods often fail in high levels of clutter. For location tracking, the location inference algorithm must be able to cope with a large amount of noise in the smartphone's sensor readings. Ambiguity in the phone's movements is not always resolvable from the sensor readings alone, so assistance is sometimes required from other modalities like the smartphone's camera. This paper makes the following new contributions to successfully address the aforementioned challenges:

- We design and implement an asset tracking system using smartphones, with a lightweight mobile agent running on the smartphone and the recognition and location inference algorithms running on the server.
- We use books on shelves as a concrete example to demonstrate the practicality of the proposed asset tracking system. In particular, we develop a spine segmentation algorithm, apply robust feature-based image matching to recognize each segmented spine, and accurately estimate each spine's pose in the query photo. Sensor readings from the smartphone and image matching help us localize each recognized book spine first within a room and then within a bookshelf.

2. RELATED WORK

2.1 ASSET TRACKING

Automated asset tracking has been considered by several groups in the literature. Barcode systems[1] simplify the task of creating an inventory compared to manual data collection, and the technology is easy to use with little training and fairly accurate in controlled settings. However, barcode systems require specialized equipments, namely barcode scanners, and require markers to be placed on every asset being tracked. These two requirements significantly increase the deployment expense. It is also sometimes not possible to place markers of sufficient

resolution on assets with non-flat surfaces or assets with very small surfaces, like thin book spines. Another disadvantage of barcode systems is that they only support line-of-sight and close-range scanning.

RFID's are also used nowadays for tracking assets. Lampe and Strassner[2] demonstrate an RFID-based tool tracking system, where an RFID chip is attached to each tool and an RFID reader is integrated into a toolbox. While RFID's enable non-line-of-sight scanning, they still have a limited scanning range. To cope with the range limitation, Patil et al.[3] propose to install an RFID reader on a robot which periodically sweeps along some programmed routes within a warehouse. Due to their need to conserve power, RFID's have limited processing power and memory. Ibach et al.[4] propose to attach a more powerful and thus more expensive embedded system to each asset for accurate asset tracking. These embedded systems continuously measure the signal strength of multiple nearby WiFi access points, and exchange the measurement results among themselves in order to derive their own locations. Like the barcode solution, the RFID and WiFi systems[2–4] require using specialized hardware and attaching external tags to the assets. The high cost of deploying these systems may discourage many small and medium-sized institutions or companies from using them.

Our asset tracking system is different from the aforementioned systems in three aspects. First, our system uses commodity smartphones and personal computers that the users already own, while other systems require purchasing new specialized equipments. Second, our system requires no external tags or markers, thereby eliminating the time and effort needed to physically attach a tag or marker to each individual asset. Third, since smartphones are being used, we can easily display detailed information about the inventory on the smartphone's viewfinder, even as the inventory is being dynamically constructed or updated.

2.2 VISUAL SEARCH

Many mobile visual search systems have been successfully developed in the past few years. Applications include landmark recognition[5] where a mobile application serves as a virtual tour guide; product cover recognition[6–9] where users snap pictures of product cases to automatically retrieve a list of current prices from different vendors and additionally media samples for CD and DVD covers; and video snapshot recognition[10] where visual search seamlessly links what people watch on their TV's at home and on their mobile devices when away from home. These systems achieve accurate image-based recognition using local features such as SIFT,[11] SURF,[12] and CHoG[13] which can be matched robustly despite photometric and geometric distortions. Equally important for large-scale image search are data structures which efficiently index local features, and the most notable of these are the vocabulary tree[14] and its recent extensions.[15,16] To facilitate fast search, vocabulary tree systems use an inverted index to quickly judge similarity between each database image and the query image. Very large databases can greatly benefit from inverted index compression,[17] because a compressed index avoids time-consuming memory congestion on a server.

Our feature-based asset recognition methods exploit these latest advances in mobile visual search. Robust local features enable us to reliably identify the assets when there are different illuminations, viewpoints, cropping, and clutter. Different from prior visual search systems, the proposed system segments out individual assets in the query photograph prior to feature extraction, which is extremely beneficial for improving the ratio of inlier to outlier features, as we will show in our experimental results.

2.3 LOCATION SENSING

There has been a large volume of work in localization using smartphones. First, an energy-efficient location sensing solution is developed[18] which uses low-power sensors to achieve location sensing operations that are ordinarily power-intensive, like determining Global Position Systeming (GPS) coordinates. Second, ambient sources like sound and light have been exploited to identify where a customer is currently shopping.[19] This is an interesting approach for tracking a person's daily activities, but it might not be suitable for asset tracking in locations where sound and light are not particularly informative, like in a library. Third, there have been several approaches for achieving highly accurate indoor localization using radio frequency (RF) signals.[20] These approaches are also applicable to smartphones that can communicate with existing WiFi infrastructure, but relying only on WiFi for highly accurate localization requires significant efforts in surveying the spectrum.[21]

In contrast to the previous methods, our location tracker uses sensors installed on all modern smartphones to provide accurate localization of assets. Each smartphone nowadays is equipped with a WiFi transceiver, an

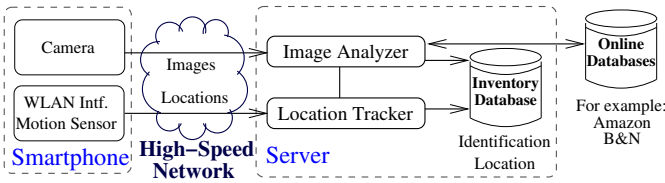


Figure 2. Using smartphones to update asset inventory.

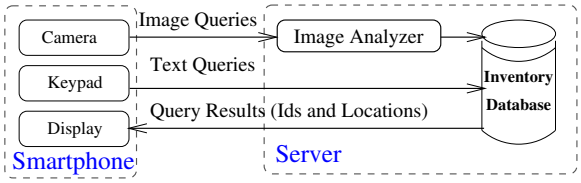


Figure 3. Using smartphones to query asset inventory.

accelerometer, and a digital compass. As we will show in Sec. 3, these sensors allow our system to provide highly accurate location information for each asset without requiring any additional expensive equipment or infrastructure.

3. SYSTEM ARCHITECTURE

3.1 OVERVIEW

In our asset tracking system, the smartphone runs a lightweight mobile agent that provides a graphical user interface and communicates with the server over a high-speed network such as a wireless local area network (WLAN). The system in Fig. 2 illustrates how users build and update the asset inventory. When a query photo is sent from the smartphone to the server, the image analyzer on the server recognizes the individual assets in a query photo through robust feature-based image matching against an online database. The online database contains product images from a vendor like Amazon or Barnes-and-Nobles. Also on the server, the location tracker combines readings from various smartphone sensors to determine the locations of the assets.

Once an inventory has been constructed, the system in Fig. 3 shows how a user can query the asset inventory. The most common queries are text-based queries. The query text strings are matched against meta data of assets stored in the inventory. For example, a user may type in the book name as the query text to find his/her `Computer Networks` textbook. Our mobile agent also supports image-based queries, using the same underlying recognition technology employed during inventory construction. A new query image taken with the smartphone is sent to the image analyzer on the server to find a closely matching image in the inventory database. This second mode of querying saves the user from the hassle of typing in any text.

3.2 IMAGE ANALYZER

The image analyzer identifies the assets in a query image by comparing the query image against images in an online database. Robust asset recognition is very challenging as images may be taken from various viewpoints, at different scales, under diverse illumination conditions, and with varying image quality. Images taken by smartphones typically contain more noise and blurring than images captured with high-quality digital cameras. Local scale- and rotation-invariant image features[11,12] enable image matching in the presence of many different geometric and photometric distortions. Although such features are also partially resilient against clutter, they do not cope well with images containing many different objects in unknown orientations. From the perspective of any single object, the other objects in the image act as clutter, so the ratio of inlier to outlier features is fairly low for a single object of interest. To address these problems and achieve high recognition accuracy for photos containing many different objects or assets, our image analysis algorithm is carried out in two stages: (i) asset segmentation and (ii) asset recognition.

Asset segmentation accurately divides each captured image into several regions, where each region is likely to contain a single asset. Designing a *general* asset segmentation algorithm is an open problem that is outside the scope of this work. In this paper, we aim to design and implement a practical asset tracking system with the assumption that a user would provide the specific asset type as an *oracle* to our system. This assumption is quite reasonable, e.g., a library is far more likely to be interested in books on the bookshelves than any other category of assets. Leveraging on the properties of a given asset type, we can design a *specific* segmentation algorithm. Effective segmentation enables the recognition engine to focus on identifying each segmented object, rather than trying to identify the group of objects together.

Following asset segmentation, asset recognition matches the segmented image regions corresponding to different assets against a visual database of individual assets. Even after segmentation, there are strong geometric and photometric differences between each asset's appearance in the query photo and the appearance of the corresponding asset in the database. Thus, local features like SIFT and SURF are used to achieve robust matching in the presence of such visual differences. For scalable search in a large database, the features are quantized in a vocabulary tree or another appropriate approximate nearest neighbor (ANN) search structure. The quantized query features can then be quickly compared against quantized database features using an inverted file system associated with the ANN search structure.

3.3 LOCATION TRACKER

Accurately tracking the location of each asset creates several challenges. First, most assets such as books or toys are placed indoor. Even though Global Positioning System (GPS) provides fairly accurate localization (e.g., < 5 meters) in outdoor environments, the same technology exhibits poor localization performance for indoor environments. Second, in indoor environments, depending on the size of the asset to be tracked, the location information may need to be very fine-grained. For example, to localize a book, it is not sufficient just to know which room contains the book; we specifically need to know which bookshelf and which rack within that bookshelf holds the book. Third, physical location information in the form of latitude and longitude coordinates might not be helpful for a user to pinpoint the location of assets. Instead, a semantic description of the location (e.g., living room, north wall, eye level) could be much more informative.

To overcome the challenges mentioned above, our location tracker makes use of an existing WiFi infrastructure and inexpensive motion sensors built in smartphones. WiFi networks are already pervasive in most indoor environments, including residential areas, campus buildings, and business facilities. WiFi is also becoming more widely available in some outdoor environments such as urban centers. During training, the user specifies which room he/she is standing in while carrying the smartphone, and the location tracker profiles the WiFi fingerprints from multiple nearby WiFi access points. This profile information is then stored in the server for future location look-ups. Subsequently, when the smartphone is building an asset inventory, the location tracker scans local WiFi networks and matches the acquired WiFi fingerpoints against the fingerpoints stored in the server to quickly and reliably identify the user's current location.

WiFi-based localization lets us determine which rooms contain which assets. For finer granuality in localization, the motion sensors on the smartphone can be employed. The digital compass, or even a digital gyroscope in more advanced smartphones, provides information about which direction the smartphone is pointing. When an asset is photographed, recording the digital compass measurements around that time instant yields a record of which direction the user should be facing in order to see the asset. Similarly, the accelerometer on the smartphone provides information about recent phone movements. From the pattern of recent horizontal and vertical accelerations, the offset of the phone's position from a previous anchor point can be inferred. These smartphone sensor measurements by themselves are not adequate for calculating location but should be combined with the WiFi-based location results.

4. BOOK TRACKING SYSTEM

4.1 PREVIOUS WORK ON BOOK SPINE RECOGNITION

Recognizing books on bookshelves has been previously considered by several groups. Lee et al.[22] study the problem of automated shelf reading, where they scan through bookshelves in libraries to ensure books are placed in the right order. They apply color quantization on each book spine and use mean, minimum, and maximum color indices as the features for content-based book spine matching. Their book spine matching algorithm, however, assumes that the books are perfectly aligned with the vertical axis. Furthermore, they assume that the order of the books on the shelves is known beforehand, which is not the case when books are being added into the inventory database for the first time. Quoc and Choi[23] develop a system for book recognition, which consists of three components: book region extraction, book segmentation, and title extraction. For title extraction, they use optical character recognition (OCR) to read the book title. OCR is not robust, however, against the severe photometric and geometric distortions typically encountered in smartphone imagery. Crasto et al.[24] invent a projector-camera system to continuously monitor books in a bookshelf. Per users' requests, the system plays

a movie of the bookshelf appearance in the near past. Their system relies on color histograms for book spine recognition and thus is sensitive to illumination changes. They also employ two cameras and a geometry model for book spine segmentation. The cameras in their system are fixed in location and can only cover one or two bookshelves at most, and moving the cameras to a different location would require some tedious re-calibration.

Our book tracking system differs from these previous systems in several important aspects. First, we employ robust local features for accurate, scalable book spine recognition in presence of challenging geometric and photometric distortions. Second, our system leverages smartphones and requires no system calibration when visiting a new location. Third, our system seamlessly combines image-based book spine recognition with location tracking to automatically build location-aware inventory databases.

4.2 BOOK SPINE SEGMENTATION

Book spine recognition is more challenging than recognition of the front covers of books and CD/DVD cases, a problem we have previously researched.[9,25,26] First, a book's spine has much smaller surface area than its cover, yielding fewer visual features that can be utilized for robust recognition. Second, each query photo of a bookshelf contains many book spines, and from the perspective of a single spine, all the other spines' visual features represent background clutter. Since books can be randomly placed on the bookshelf, the order in which the books appear is not known beforehand, so the database must contain individual book spines images. Trying to directly match a photo containing many spines against a database of individual spines would result in low recognition accuracy. Segmenting out the individual spines in the query photo is thus crucial for accurate recognition.

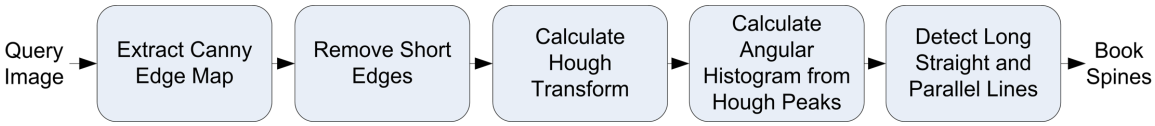


Figure 4. Operations for block spine segmentation.

To perform the segmentation, we exploit the fact that books are often densely packed together on a bookshelf and have roughly the same orientation, as exemplified by the query photos shown in Fig. 5(a,e). Fig. 4 shows a block diagram of the operations performed for segmentation. First, a Canny edge map is extracted from the query image.[27] In the Canny edge map, the crevices between book spines appear as long and straight edges, while the text and graphics on the spines appear as short and curved edges. Connected components are detected in the edge map, and all components containing fewer pixels than a lower threshold are removed. Removal of these short edges improves the subsequent steps for line boundary detection.

Next, a Hough transform is calculated from the edge map with short edges removed. Fig. 5(b,f) displays the Hough transform calculated for two query images in Fig. 5(a,e). In the Hough plots, squares denote the locations in the (ρ, θ) space where peaks in the Hough transform occur. A large group of peaks around a common angle corresponds to nearly parallel lines between book spines. To estimate the dominant angle θ_{spines} of the spines, the Hough peak occurrences are binned in an angular histogram as shown in Fig. 5(c,g). The bin attaining the highest count in the angular histogram yields the spines' dominant angle θ_{spines}.

Given the dominant angle θ_{spines} of the book spines, we search for long straight lines with angle near θ_{spines}. Long straight lines can be detected from the Canny edge map.[28] Each such line having an angle lying in the range $(\theta_{\text{spines}} - \Delta\theta, \theta_{\text{spines}} + \Delta\theta)$ is considered as a potential boundary between two books. In our experiments, $\Delta\theta = 15^o$ was found to be a good parameter. Fig. 5(d,h) shows that our segmentation method successfully finds the line boundaries between books. Regions between the line boundaries are segmented out as individual spines. As will be shown in Sec. 5, spine segmentation significantly improves recognition performance compared to a scheme that skips this step, because segmentation substantially boosts the ratio of inlier to outlier features for each single spine.

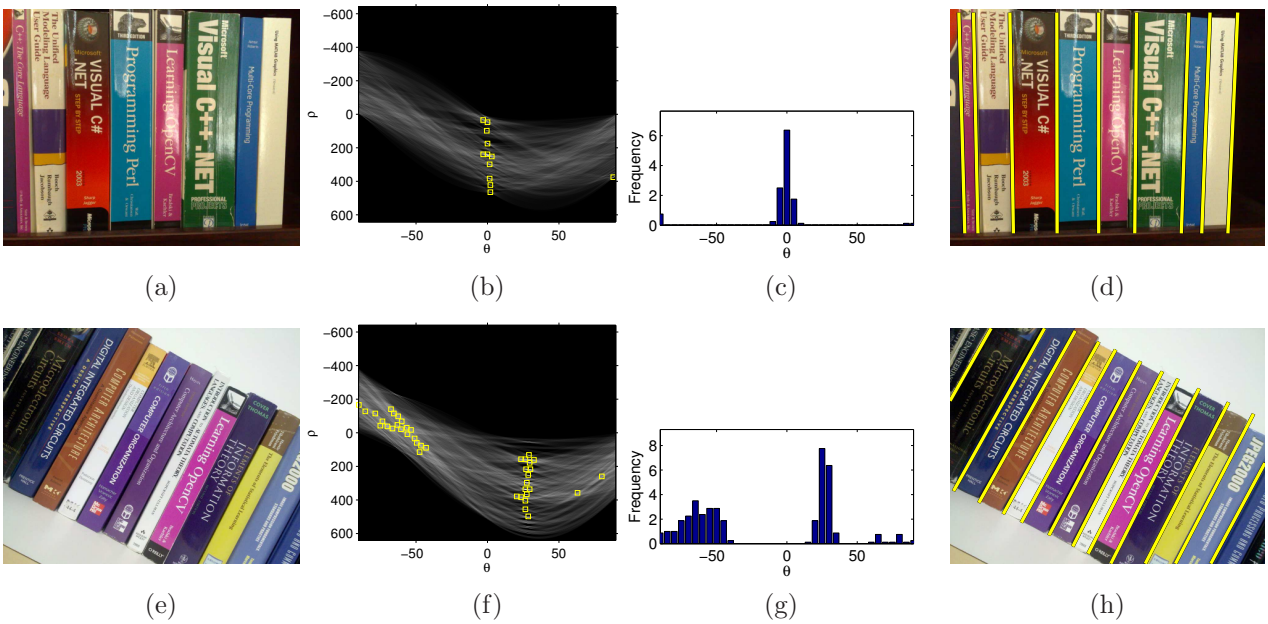


Figure 5. (a,e) Original bookshelf images, (b,f) Hough transforms with peaks denoted by squares, (c,g) Angular histogram for Hough peaks, (d,h) Detected line boundaries between book spines.

4.3 BOOK SPINE RECOGNITION

Each of the segmented spines is individually matched against a database of book spines. Fig. 6 shows the steps used in the recognition process. First, a set of robust local features, commonly referred to as a bag-of-visual-features (BoVF), is extracted from each query spine. Since achieving low query latency is very important for interactive mobile applications, we employ SURF features[12] which are much faster to compute than SIFT features[11] but offer comparable performance.[25,29] Recognition with BoVF also provides robustness against imperfect segmentation.

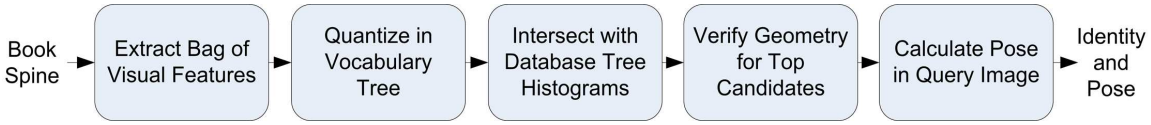


Figure 6. Operations for block spine recognition.

After segmentation, we know the exact orientation of each book spine. Using this knowledge, we can extract upright SURF features[12] instead of oriented SURF features. Since orientation assignment is an expensive operation, upright SURF features are much faster to compute than oriented SURF features. Furthermore, upright SURF avoids errors in orientation assignment because the orientation of the spine has been predetermined. Fig. 7 shows an example of oriented SURF and upright SURF keypoints for a segmented spine. Notice that the locations and scales of the keypoints are identical in both cases, but on the left, individual orientations are assigned according to the local dominant gradient, while on the right, orientations are assigned to be the same angle. We show in Sec. 5 that upright SURF substantially boosts feature matching performance compared to oriented SURF for segmented book spines.

Figure 7. Segmented spine with (a) Oriented SURF keypoints and (b) Upright SURF keypoints.

For fast search through a large database of book spines, each query spine's BoVF is quantized through a vocabulary tree.[14] Soft binning is used to mitigate quantization errors.[15] The query spine's quantized BoVF form a tree histogram, counting how often each node in the vocabulary tree has been visited. A similarity score between the query tree histogram and each precomputed database tree histogram is calculated by histogram intersection, an operation that can be performed efficiently using an inverted index.[14] Thereafter, a shortlist of the 50 top-ranked database spines are further subjected to rigorous geometric verification with the ratio test[11] and affine model RANSAC[30] to find spatially consistent feature matches. The best matching database spine is then selected.

To estimate the pose of the book spine in the query image, we exploit two complementary sources of information. First, during geometric verification, feature correspondences between the query spine and the best matching database spine enable us to generate a transformation model from the database spine's coordinate frame into the query spine's coordinate frame. By mapping the corners of the database spine into the query spine's coordinate frame using the transformation model, we can estimate the pose of each spine in the query image. Second, during segmentation, the line boundaries surrounding a query spine yield a different estimate of the spine's pose in the query image. Suppose the feature-based estimate predicts the spine's corner points to be $\{p_{1,i} = (x_{1,i}, y_{1,i}), i = 1, 2, 3, 4\}$, while the line-based estimate predicts the corner points to be $\{p_{2,i} = (x_{2,i}, y_{2,i}), i = 1, 2, 3, 4\}$. We found the following rule of combining the two estimates to yield good results.

$$
\begin{aligned}
p_{c,i} &= \alpha p_{1,i} + (1-\alpha)\, p_{2,i} \qquad i = 1, 2, 3, 4 \\
\alpha &= \frac{N_1}{N_1 + \beta L_2} \\
N_1 &= \text{number of feature correspondences from geometric verification} \\
L_2 &= \text{total length in pixels of two line boundaries of book spine from segmentation} \\
\beta &\approx 0.1
\end{aligned}
$$

When more features are matched during geometric verification, the feature-based estimate $p_{1,i}$ is more reliable and more weight is given to $p_{1,i}$. Likewise, when the line boundary found during segmentation is longer, there is greater confidence in the line-based estimate $p_{2,i}$ and more weight is assigned to $p_{2,i}$ accordingly. Adaptively combining the two pose estimates is found to reduce the estimation error compared to using either estimate alone.

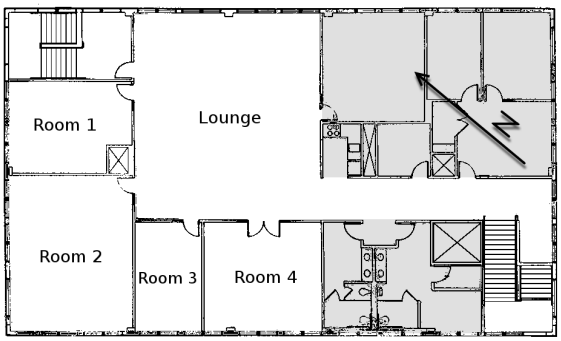
4.4 LOCATION TRACKING

Assets are organized by their locations in the inventory database. Books can be partitioned in the database by the room, the side of the room, and the rack which holds a book. We use WiFi fingerprints, motion sensor readings, and visual features to identify the location of each book when updating the database.

4.4.1 LOCATION SENSING

We created an experimental environment in a community center shown in Fig. 8(a), using the four rooms and the lounge marked by the non-gray regions. When the user starts to take photos of the bookshelves, the client uses the WiFi fingerprint to identify which room the user is in. The orientation of the smartphone is determined from a digital compass fingerprint obtained for each room individually, as shown in Fig. 8(b). Although the compass readings vary between rooms, within a room, the four different directions have sufficiently different signatures for the directions to be discerned from one another.

To further localize each book, we also need to obtain the physical location of the smartphone in front of the bookshelf. We scan the bookshelf from top rack to bottom rack. For each rack, we take photos from the left to right and then from right to left for next rack, alternatively. We use the accelerometer sensor readings to roughly track the phone's position. See Fig. 9(a) for a temporal trace of accelerometer sensor data that was captured scanning a three-rack bookshelf. In the process, we took three photos for every rack in the bookshelf. As can be seen from the figure, the raw sensor data from the client is fairly noisy. Thus, we classify the movements to either vertical or horizontal by testing the acceleration variance shown in Fig. 9(b). When the vertical variance



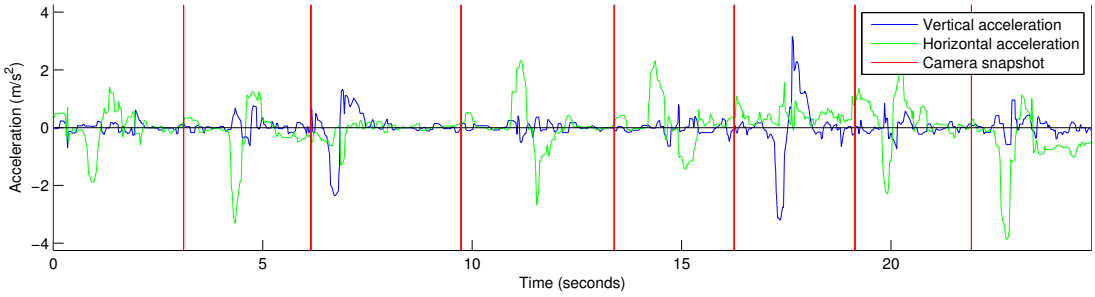
(a)

	NE	SE	SW	NW
Room 1	-1.89	-0.32	0.72	-2.73
Room 2	-0.92	0.63	2.44	-1.90
Room 3	-0.89	0.90	2.30	-2.12
Room 4	-0.13	1.25	-2.73	-1.65
Lounge	-1.62	-0.57	0.64	3.00

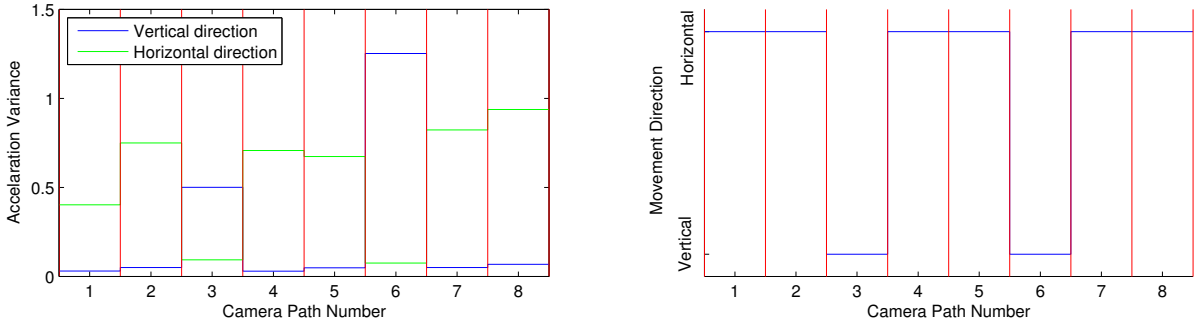
(b)

Figure 8. (a) Floor plan of the community center. We use the four rooms and the lounge for location sensing. (b) Compass sensor data obtained from the smartphone facing the four different wall directions for different rooms in the community center.

is larger than the horizontal variance, we can reliably classify the camera path as a vertical movement. Thus, we know the rack where the photo is taken by observing the history of vertical movements. These sensor-based position estimates can be further refined using the feature-based rack identification method discussed in the next section.



(a)



(b) (c)

Figure 9. (a) Accelerometer measurements obtained from the smartphone while scanning a bookshelf with three racks. We took three pictures for each rack. A long vertical line represents a time instant when a photo is taken. (b) Measurements of the horizontal and vertical acceleration variance between camera shots. When the vertical direction acceleration variance is larger, we classify the movement as vertical, and vice versa. (c) The final classification result of our algorithm.

4.4.2 BOOKSHELF RACK IDENTIFICATION

In conjunction with the sensor-based estimates described in the previous section, a photo of a rack, such as those shown in Fig. 5(a,e), can be matched against a photo of the entire bookshelf. This process is illustrated

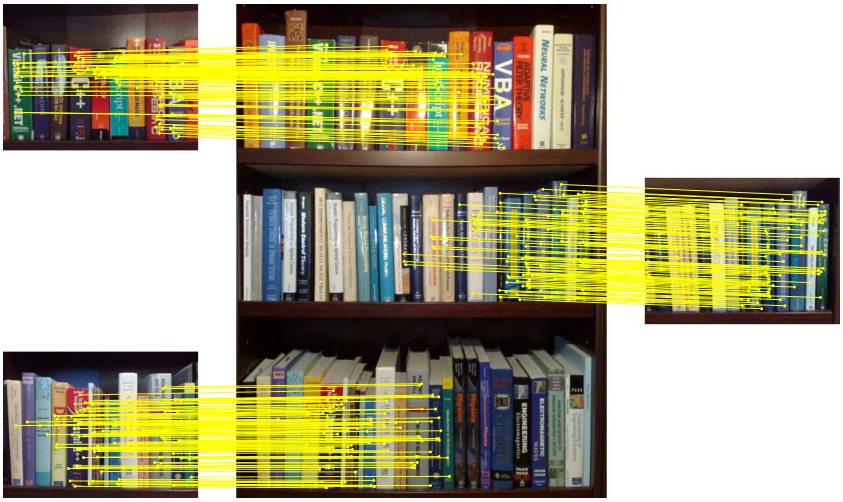


Figure 10. Feature-based matching of three photos of individual racks to a photo of the entire bookshelf. Feature point correspondences are connected by lines.

in Fig. 10, where the photo of the entire bookshelf appears in the middle and three photos of three separate racks appear on the sides. Robust local features are extracted from the images, and the features in each rack photo are matched against the features in the bookshelf photo using the same geometric verification process mentioned in Sec. 4.3. Feature correspondences between the rack photos and the bookshelf photo are denoted by lines in Fig. 10. As can be seen, the abundance of feature correspondences enables us to reliably identify the exact location in the bookshelf for each group of books.

5. EVALUATION

We have implemented a mobile book recognition system that employs the techniques discussed in this paper. A demo video is available online.* Fig. 1 shows a screenshot of the viewfinder of a smartphone running our client software. Our client software has been developed for both the Symbian OS and the Android OS. The demo video illustrates how a user's inventory database can be gradually constructed. Each time the user snaps a query photo, our system searches an online database of book spines, recognizes and localizes each spine in the photo, and adds the recognized spines to the user's inventory database. When the recognition result is sent to the smartphone, the user can conveniently select and view the recognized spines. A selected spine has its boundary highlighted and its title displayed in the smartphone's viewfinder, enabling easy visualization of the new books being added to the inventory database.

In our mobile client, the location tracking functionality is active as the user constructs the inventory database. To show the performance of the accelerometer-based movement classifier, we create a scatter plot of the horizontal and vertical acceleration variance in Fig. 11 to show that a clear division exists between the two types of movements. In our tests, we did not observe any false classifications. A photo of a bookshelf rack can also be matched against a previously taken photo of the entire bookshelf (see Fig. 10) to store information about which rack each book resides in. The inferred location is combined with the book recognition results in the inventory database.

To test recognition accuracy, we constructed a database of 2148 book spine images, simulating a database of books covering a particular subject. The book spine images are taken at the Stanford Engineering Library. SURF features are extracted from each database spine image. The set of all database features is used to train a vocabulary tree with 6 levels and 10^6 leaf nodes. For query images, we took 50 photos of bookshelves with a

*Demo Video: `http://msw3.stanford.edu/~dchen/CIBR/Book-Spines-Android.wmv`

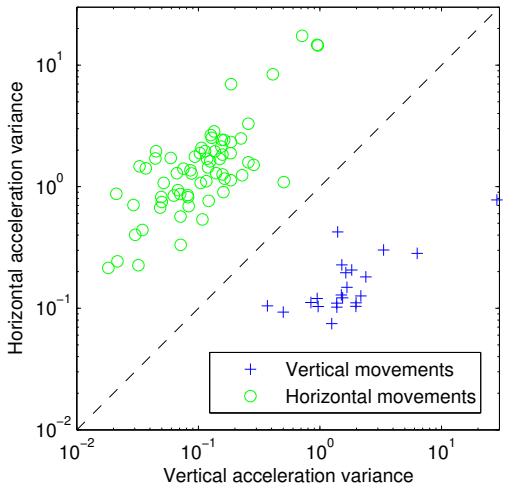


Figure 11. Scatter plot of the horizontal and vertical acceleration variances.

smartphone, showing books in different orientations and illuminations.[†] Each query image has a resolution of 1024×768 pixels.

We compare three different recognition schemes. First, *SURF* refers to extracting oriented SURF features from the entire query image, without any spine segmentation, and querying the features against the online database. After vocabulary tree scoring, a shortlist of the 50 top-ranked database spines are verified geometrically. This scheme suffers from the fact that from the perspective of one spine, the features for all the other spines in the query image act as clutter and create a very low ratio of inlier to outlier features. The recognition accuracy, defined as the number of correctly recognized spines divided by the total number of spines in the query images, for the *SURF* scheme is shown in Table 1. Out of 407 spines in the 50 query images, only 31 are correctly recognized, yielding a very low accuracy of 7.6 percent.

Table 1. Comparison of different recognition schemes.

Scheme	Total Spines	Recognized Spines	Accuracy	Processing Time
SURF	407	31	7.6 %	2.72 sec
Segmented SURF	407	294	72.2 %	1.76 sec
Segmented Upright SURF	407	337	82.8 %	1.65 sec

In contrast, the *Segmented SURF* scheme segments out each book spine in the query image before extracting oriented SURF features. Each spine's BoVF can then be individually queried against the online database using vocabulary tree scoring followed by geometric verification. Since the amount of clutter is significantly reduced from the perspective of any single spine, this scheme substantially boosts the ratio of inlier to outlier features and therefore greatly improves recognition accuracy. Out of 407 spines in total, *Segmented SURF* correctly recognizes 294 spines, resulting in an accuracy of 72.2 percent, which is much higher than the 7.6 percent obtained without segmentation. Another advantage of the *Segmented SURF* scheme is that it reduces the processing time to 1.76 sec per query image, compared to the *SURF* scheme which takes 2.72 sec. Timing was measured on a Linux duo-core server running at 2.00 GHz. The time savings are due to the extraction of fewer features in total and hence fewer features being subjected to time-consuming geometric verifications. After segmentation, SURF features are detected only in the interior of the segmented spines, not along the spine edges, crevices between books, and in the surrounding bookshelf frame.

Finally, the *Segmented Upright SURF* scheme also performs spine segmentation prior to feature extraction, but it extracts upright SURF features as opposed to oriented SURF features (see Fig. 7). Since the orientation

[†]Query Images: `http://msw3.stanford.edu/~dchen/CIBR/BookSpineQuery`

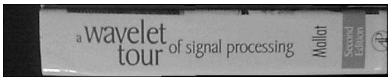
of the spine is known from segmentation, upright SURF avoids the orientation assignment errors incurred by oriented SURF. Table 1 shows that *Segmented Upright SURF* achieves the best performance of the three schemes: an accuracy of 82.8 percent is attained, a 10 percent improvement over *Segmented SURF*. Because orientation assignment is a costly portion of feature keypoint detection, this scheme also has the fastest processing speed.

For comparison against our feature-based recognition methods, we performed a preliminary experiment where we tried to recognize the segmented spines by optical character recognition (OCR). OCR has been proposed in prior work for automatic book title extraction.[23] We employed the open-source Tesseract OCR engine.[31] Because the spines often contain non-text graphics, uneven illumination, and varying fonts, the OCR engine almost always gave unrecognizable text outputs. Fig. 12 shows a few segmented spines and their OCR readings from Tesseract. It can be seen that the OCR readings are very noisy, and even post-processing operations like spelling correction or nearest Levenshtein distance matching against a dictionary are unlikely to yield the correct results. A more thorough evaluation of OCR-based spine recognition with images captured by smartphones is planned for our future work.

Java I':¿.s¡——Iiz—I gs

Nsumu Nsrwokxs

' *1 snscrnomcnzrnc EE Q;A W WAVIQS – :



Slgllll pI'(¿(.(JS.II'Ig E

Figure 12. Segmented spines and their OCR readings from Tesseract.

6. CONCLUSIONS

Asset tracking can be a very tedious and costly operation. We have developed an effective asset tracking system using smartphones, where a user simply uses a smartphone to snap photos of various assets for generating and updating an inventory. Robust identification of assets visible in the query photos is achieved through feature-based image matching. The location of each recognized asset is efficiently inferred from the smartphone's sensor measurements, with possibly additional assistance from image-based analysis. As a practical demonstration of asset tracking, we have created a mobile book tracking system with which any smartphone user can easily manage his/her book assets. The system processes the photo of a bookshelf to automatically identify the books and add the books into the inventory. Segmentation of the individual book spines prior to feature-based image matching is found to be crucial for accurate recognition. Having determined the orientation of each spine during segmentation, we can also extract more discriminative image features to further boost recognition performance. Our book tracking system has been implemented for a couple of popular smartphone operating systems and has an intuitive, appealing graphical user interface that highlights identified book spines in the smartphone's viewfinder. Leveraging both smartphones with ever-expanding capabilities and state-of-the-art visual search techniques, we can build cost-effective and easy-to-deploy asset tracking systems for objects other than books.

REFERENCES

[1] McCathie, L. and Michael, K., "Is it the end of barcodes in supply chain management?," in [*Proc. Collaborative Electronic Commerce Technology and Research Conference, LatAm*], 1–19 (October 2005).

[2] Lampe, M. and Strassner, M., "The potential of RFID for movable asset management," in [*Proc. Workshop on Ubiquitous Commerce*], 9–12 (October 2003).

[3] Patila, A., Munsonb, J., Woodb, D., and Coleb, A., "Bluebot: Asset tracking via robotic location crawling," *Computer Communications* **31**(6), 1067–1077 (2008).

[4] Ibach, P., Stantchev, V., Lederer, F., Weiss, A., Herbst, T., and Kunze, T., "WLAN-based asset tracking for warehouse management," in [*Proc. IADIS International Conference e-Commerce*], 1–8 (December 2005).

[5] Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W.-C., Bismpigiannis, T., Grzeszczuk, R., Pulli, K., and Girod, B., "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in [*Proc. ACM Multimedia Information Retrieval (MIR'09)*], 427–434 (October 2009).

[6] Kooaba, "Product logo recognition on mobile devices." http://www.kooaba.com/technology/labs.

[7] SnapTell, "Media jacket recognition on mobile devices." http://www.snaptell.com/demos/DemoLarge.htm.

[8] Google, "Google Goggles for object recognition on mobile devices." http://www.google.com/mobile/goggles.

[9] Tsai, S., Chen, D., Singh, J., and Girod, B., "Rate-efficient, real-time CD cover recognition on a camera-phone," in [*Proc. ACM Multimedia (ACMMM'08')*], 1023–1024 (October 2008). Technical Demo.

[10] Chen, D., Cheung, N.-M., Tsai, S., Chandrasekhar, V., Takacs, G., Vedantham, R., Grzeszczuk, R., and Girod, B., "Dynamic selection of a feature-rich query frame for mobile video retrieval," in [*Proc. IEEE International Conference on Image Processing (ICIP'10)*], (October 2010).

[11] Lowe, D., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision* **60**(2), 91–110 (2004).

[12] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V., "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding* **110**(3), 346–359 (2008).

[13] Chandrasekhar, V., Reznik, Y., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R., and Girod, B., "Quantization schemes for low bitrate compressed histogram of gradients," in [*Proc. International Workshop on Mobile Vision (IWMV'10)*], (June 2010).

[14] Nister, D. and Stewenius, H., "Scalable recognition with a vocabulary tree," in [*Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*], 2161–2168 (June 2006).

[15] Philbin, J., Isard, M., Sivic, J., and Zisserman, A., "Lost in quantization: Improving particular object retrieval in large scale image databases," in [*Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*], 1–8 (June 2008).

[16] Jegou, H., Douze, M., and Schmid, C., "Improving bag-of-features for large scale image search," *International Journal of Computer Vision* **87**(3), 191–212 (2010).

[17] Chen, D., Tsai, S., Chandrasekhar, V., Takacs, G., Vedantham, R., Grzeszczuk, R., and Girod, B., "Inverted index compression for scalable image matching," in [*Proc. Data Compression Conference (DCC'10)*], 525 (March 2010).

[18] Zhuang, Z., Kim, K.-H., and Singh, J., "Improving energy efficiency of location sensing on smartphones," in [*Proc. of ACM Mobile Systems, Applications and Services*], 315–330 (June 2010).

[19] Azizyan, M., Constandache, I., and Choudhury, R., "SurroundSense: Mobile phone localization via ambience fingerprinting," in [*Proc. ACM International Conference on Mobile Computing and Networking (MobiCom'09)*], 261–272 (September 2009).

[20] Bahl, P. and Padmanabhan, V., "RADAR: An in-building RF-based user location and tracking system," in [*Proc. IEEE INFOCOM'00*], 775–784 (March 2000).

[21] Kim, K.-H., Min, A. W., and Shin, K. G., "Efficient and accurate monitoring of wifi spectrum condition using mobility," in [*Proc. of ACM Mobile Computing and Networking*], (September 2010).

[22] Lee, D., Chang, Y., Archibald, J., and Pitzak, C., "Matching book-spine images for library shelf-reading process automation," in [*Proc. IEEE International Conference on Automation Science and Engineering (CASE'08)*], 738–743 (September 2008).

[23] Quoc, N. and Choi, W., "A framework for recognition books on bookshelves," in [*Proc. International Conference on Intelligent Computing (ICIC'09)*], 386–395 (September 2009).

[24] Crasto, D., Kale, A., and Jaynes, C., "The smart bookshelf: A study of camera projector scene augmentation of an everyday environment," in [*Proc. IEEE Workshop on Applications of Computer Vision (WACV'05)*], 218–225 (January 2005).

[25] Chen, D., Tsai, S., Chandrasekhar, V., Takacs, G., Singh, J., and Girod, B., "Robust image retrieval using multiview scalable vocabulary trees," in [*Proc. Visual Communications and Image Processing (VCIP'09)*], 72570V (January 2009).

[26] Tsai, S., Chen, D., Singh, J., and Girod, B., "Image-based retrieval with a camera-phone," in [*Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'09)*], (April 2009). Technical Demo.

[27] Canny, J., "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence* **8**(6), 679–698 (1986).

[28] Kosecka, J. and Zhang, W., "Video compass," in [*Proc. European Conference on Computer Vision (ECCV'02)*], 476–490 (May 2002).

[29] Takacs, G., Chandrasekhar, V., Tsai, S., Chen, D., Grzeszczuk, R., and Girod, B., "Unified real-time tracking and recognition with rotation-invariant fast features," in [*Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*], 1–8 (June 2010).

[30] Fischler, M. and Bolles, R., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* **24**(6), 381–395 (1981).

[31] Smith, R., "An overview of the Tesseract OCR engine," in [*Proc. Conference on Document Analysis and Recognition (ICDAR'07)*], 629–633 (September 2007).