Indoor Localization Using Camera Phones

Nishkam Ravi, Pravin Shankar, Andrew Frankel,
Ahmed Elgammal and Liviu Iftode
Department of Computer Science, Rutgers University, Piscataway, NJ 08854
{nravi, spravin, afrankel, elgammal, iftode}@cs.rutgers.edu

Abstract

Indoor localization has long been a goal of pervasive computing research. In this paper, we show the feasibility of determining user's location based on the camera images received from a smart phone. In our system, the smart phone is worn by the user as a pendant and images are periodically captured and transmitted over GPRS to a web server. The web server returns the location of the user by comparing the received images with images stored in a database. We tested our system inside the Computer Science department building. Preliminary results show that user's location can be determined correctly with more than 80% probability of success. As opposed to earlier solutions for indoor localization, this approach does not have any infrastructure requirements. The only cost is that of building an image database.

1 Introduction

Most of the early systems for indoor localization [34, 11, 5, 20] focussed mainly on location accuracy and involved the use of custom hardware. This implied heavy deployment costs and labour requirements. Of late, the focus has shifted to minimizing infrastructure requirements without compromising substantially on accuracy [16, 27, 8]. The reason is well understood: since location information only serves as a parameter to location-based services, the cost of deploying localization systems should be a minute fraction of the total cost of provisioning location-based services.

Camera-equipped mobile phones are being put to many uses as an interesting study indicates [14]. The ability to "capture and send" digital images on the move, opens up possibilities beyond just socialization. In this paper, we explore the possibility of determining user's location indoors based on what the camera-phone "sees". The camera-phone is worn by the user as a pendant(Figure 1), which captures images periodically and sends them to a web server over GPRS. The web server has a database of images with their corresponding location. Upon receiving an image, the web server compares it with stored images, and based on the match, estimates user's location. We accomplish this with off-the-shelf image matching algorithms, by tailoring them for our purpose. We also implemented an algorithm that improves location accuracy by taking into account the trajectory of the user. We built an image database for the Computer Science building with nearly 10 pictures per "corner" to account for real-life issues such as varying heights of the users, different angles that may correspond to the same image, etc. Our experimental results indicate that room-level accuracy can be achieved with more than 90% probability, and meter-level accuracy can be achieved with more than 80% probability. The error can be further reduced by using more sophisticated image matching algorithms, which is not the subject of this paper.

The key advantage of using this approach is that it does not require any infrastructure. Neither custom hardware, nor wireless access points are required. Physical objects do not have to be

"tagged" and users do not have to carry any device apart from what they already do: a mobile phone. The only cost involved is that of building an image database. This is negligible as compared to installing hardware or tagging physical objects. Another benefit of this approach, is that it determines user's orientation along with location.

The disadvantage of using this approach is energy consumption on the mobile phone. We discuss this in detail in Section 3.6 and describe how we minimize energy consumption without compromising on accuracy. Also, location privacy is a key issue. Since location is determined by a central server, it is necessary to use privacy-preserving mechanisms. We describe two such mechanisms in Section 3.8. Another downside of our approach is that it is probabilistic in nature. Since image matching algorithms are not perfectly accurate, there is room for error. The key contribution of this paper is the demonstration that with off-the-shelf image matching algorithms, the error can be reduced to less than 20%.

Rest of the paper is organized as follows. In Section 2 we discuss related work. Section 3 describes our approach, the issues faced and their solutions. Results are presented in Section 4. We conclude in Section 5 with directions for future work.

2 Related Work

2.1 Indoor Localization

A number of indoor positioning systems have been built. In ActiveBadge [34] an IR badge worn by the user emits a unique IR signal periodically. Sensors installed at known positions pick up the signal and update the position of the badge in a centralized database. ActiveBadge provides room level positioning and incurs significant installation and maintenance costs. Active Bat [11] provides centimeter level positioning by using ultrasound receivers, which are installed on ceilings, and ultrasound transmitters which are carried by users/devices. Again, the cost of deployment is significant. In Cricket [20] ultrasound transmitters are installed at known coordinates to emit signals that are received by user's mobile device to estimate location based on time-of-flight. Cricket achieves meter level positioning. It requires installation of special beacons and users have to carry special receivers.

Microsoft's EasyLiving [15] project uses cameras installed in rooms to track humans using vision techniques. The cost of installing cameras in every room makes deployment difficult. Privacy is is a big issue as users are continously *watched*. Radar [5] operates by recording and processing signal strength information at multiple WiFi basestations. It uses signal propagation modeling to determine user's location with 5 meters accuracy. Although no additional hardware is required, WiFi coverage is assumed. Similarly, Place Lab [16] works by listening for the transmissions of radio sources such as 802.11 access points, fixed Bluetooth devices, and GSM cell towers. A beacon database provides location information based on the IDs of beacons. PlaceLab can provide user location with upto 15 meters of accuracy. PlaceLab is a very practical, high-coverage and low-cost location determination system in that no additional hardware is required. However, presence of beacons, corresponding receivers and beacon database is assumed.

WALRUS [8] provides room-level positioning by using wireless networking and microphones. Wireless data and ultrasound pulses are generated from PCs in each room and a PDA carried by the user listens for both signals. Ultrasound pulses are generated from speakers attached to the PC and do not carry any data. Wireless interface on the PC provides a synchronizing pulse along with information about each room. Although no additional hardware is required, WALRUS assumes the presence of a PC in each room and WiFi coverage. Similarly, Audio Location [27] determines user

location by using microphones that listen to sounds made by the users themselves such as finger clicking. It is a low-cost system that achieves centimeter level accuracy. However, it assumes the presence of microphones in the environment and may not work properly in a noisy environment.

Camera-phone based localization does not require any infrastructure apart from a database of images. The users do not have to carry any special device, and the physical objects do not have to be tagged. This has clear advantage over systems like Cricket, ActiveBadge, ActiveBat, EasyLiving and other systems that require special hardware. It has even lower insfrastructure requirements than Radar, Place Lab and other WiFi-based systems, as no radio sources are assumed. It can be compared with WALRUS and Audio Location in terms of insfrastructure requirement. These approaches assume microphones in the environment, while camera-phone based localization assumes a database of images.

2.2 Camera Phone Research

Work is being done in using camera phones as interaction devices by tagging physical objects with visual codes and using vision techniques to extract and interpret the information stored in these visual codes [24, 6, 23, 26, 31]. Localization could also be possibly achieved with this method. However, physical objects would have to be tagged. Sarvas et al [25] had demonstrated the use of camera phones for getting meta-information about physical objects using a human-in-the-loop approach. They demonstrated their approach for getting meta-information about outdoor landmarks. We go a step further and demonstrate the feasibility of using camera phones for indoor localization.

2.3 Robot Localization

A large body of work on vision-based robot localization exists in the artificial intelligence(AI) community [30, 19, 10, 28]. Our work has been inspired by robot localization to some extent. However, there are certain key differences. First, most of the work in the AI community has focussed on the use of landmarks for positioning. The robot determines its position based on the coordinates of the landmarks that are visible. The reason for using landmarks for robot localization is that the applications are often limited to a single room or maze, and therefore the scope for determining the position of the robot based on the images of walls and objects in the room is limited. Our approach does not make use of landmarks for localization. We are able to estimate the location of the user based on the images of walls and objects, although at a coarser granularity level.

Second, robot localization is a much more difficult problem than human localization, because the robot takes its action based on where it is. The famous RoboCup soccer contest [1] is a good example of this. The actions of the robot, such as going towards the ball, or kicking the ball are entirely dependent on where the robot is. Centimer-scale acccuracy is desired. Besides, the algorithm needs to be computationally very efficient to minimize the reaction time of the robot. Most of the research is driven by these issues. Human indoor localization on the other hand, is resilient to localization errors and delays.

Third, most of the work in the AI community is of purely theoretical nature. Experimentation is limited and is done with robots that are very small in size (usually less than a feet tall). These robots dont see the environment the way a human being can by virtue of being tall.

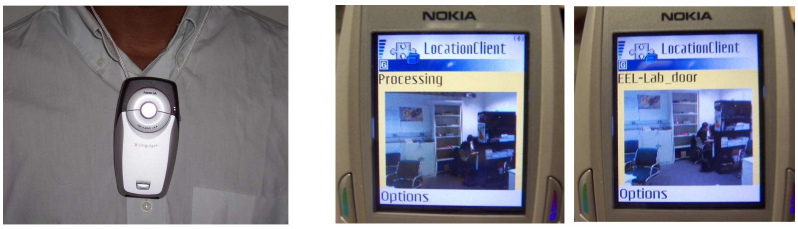

Figure 1: Left: User wearing the phone as a pendant, Right: Snapshots of the client running on the phone

2.4 Augmented Reality

Some augmented reality systems employ vision techniques for augmenting physical objects with meta-information [2, 33, 3]. The goals as well as the approach are quite different. We use image matching algorithms, while AR community uses object recognition algorithms for getting information about objects in sight. Object recognition cannot be applied for localization, as the image of a corner of a room may not have a well-defined object in it. Our approach exploits similarity between two images which is easier than recognizing objects inside an image.

3 Approach, Issues and Solutions

3.1 Location Query

The user wears the mobile phone as a pendant (Figure 1). A client running on the phone clicks images periodically, sends them to a web server over GPRS and receives location updates. No human involvement is required. The time interval between two consecutive image clicks is determined by two factors: desired update frequency and energy consumption. If the client clicks and sends images frequently, the energy consumption on the phone increases. If the client sends images very infrequently, the location updates would be received infrequently resulting in poor system performance. The lower bound on the time period is determined by the latency of sending an image over GPRS and the time taken by the web server to respond. In Section 3.6, we discuss how an optimal time period is chosen.

3.2 Database Creation

Database creation is a tedious process. A straightforward way to create an image database is to hire a database creator who walks around in the building clicking images, labeling them with location and organizing them in a database. To make this process semi-automatic, we wrote a Java client that runs on the phone and records video as the user walks around, simultaneously sending the images to the web server. The web server extracts features from the received images on the fly and stores the features in a database. The image may or may not be stored. The images/features in the database are tagged with location, manually afterwards. To make the process of tagging images with location semi-automatic, we are working on implementing a speech recognition interface on the phone. The database creator would announce his location as he walks around with the camera phone hanging as a pendant, and the speech recognition interface would send the location to the web server along with the images. The database would need to be updated periodically as the furniture inside the building changes with time.

Figure 2: Multiple low-resolution images corresponding to the same corner

3.3 Number of Images

During location query, the user wears the mobile phone as a pendant and receives location updates. The images taken by the phone depend on the height of the user as well as the angle from which they are taken. During image database creation, we take multiple images of the same *corner* to accomodate for different heights and angles. Figure 2 shows 5 pictures corresponding to the same corner of the room. On an average we store 10 images per corner. The success of the system depends on the amount of effort put into database creation. More images per corner increases the chance of success. At the same time, a very large number of images can slow down the image comparison process and increase the response time. The optimal number of images in the database would therefore be determined by the desired accuracy and response time. There is a third factor that would govern the number of images in the database, which is the desired frequency of location updates. High update frequency requires high coverage which implies a dense image database to accomodate for small changes in the location of the user. At the same time, in order to obtain updates frequently, the phone would have to send query images frequently, which would imply higher energy consumption on the phone. We discuss this in more detail in Section 3.6.

3.4 Location Representation

Indoor location can be represented in different ways, depending on the applications at hand. For our experiments, we used relative coordinates which are easy to construct. Since every image is labeled with its location discretely, we also use human understandable labels, such as {Room 334, North-East Quadrant, Facing North}, in addition to relative coordinates. The location corresponding to an image, is the location of the camera at the time when the image was taken. Figure 3 shows two images corresponding to the same corner, but with different locations. While the first one was taken with the camera in the south-east quadrant of the room, the second one (closer to the object) was taken with the camera in the north-east quadrant of the room. This approach of labeling images also takes care of the distance of the user from the object in sight.

3.5 Location Determination

When the web server receives a query image, it compares it with the images stored in the database. Every image that matches with the query image is assigned a weight which reflects the degree of similarity between the two images. By image comparison, we imply feature comparison. We use three off-the-shelf algorithms for image comparison: Color Histograms [29, 18, 32], Wavelet Decomposition [12] and Shape Matching [13]. Each algorithm assigns a weight to the image. The total weight of an image is calculated as a linear combination of the weights assigned by each algorithm. If the weight of the best match is less than a certain threshold value, the query image is discarded. This is necessary to prevent wrong location updates from being sent to the client. Once

Figure 3: Two low-resolution images with different locations corresponding to the same corner. First image was taken with camera in the south-east quadrant of the room. Second image was taken with camera in the north-east quadrant of the room.

the weight of the images in the database with respect to the query image are known, the following methods can be used for location determination:

Naive Approach: In this approach, the images in the database are organized in a flat manner. The location of the user is the location of the image that matches the query image with maximum weight. In other words, the location of the user is the one that maximizes the probability of seeing the query image.

Heirarchical Approach: In this approach, the images in the database are organized heirarchically. The images corresponding to a floor are grouped together, the images corresponding to a room are grouped together and so on. When the system determines that the user has entered a particular room, subsequent searches are performed only on the images of that room, until the system discovers that the user has exited the room. There are two advantages of this approach: (1) the search gets localized and hence response time decreases, and (2) the probability of error decreases, because the system has fewer images to confuse the query image with. The disadvantage of this approach is that if the system incorrectly determines the room the user is in, subsequent searches would get affected and produce wrong results.

History-based Approach: In this approach, the web server keeps track of the trajectory of the user. Based on the past locations of the user, a better estimate of the current location can be derived. In other words, the location of the user is determined not from a single query image, but multiple query images received over a certain period of time. When the server receives a query image, it looks at the last *n-1* query images. The current location of the user is the one that maximizes the probability of *seeing* the n query images in the shortest period of time.

To implement this, we use a simple shortest-path/nearest-neighbours approach. Upon receiving the *nth* query image, the web server carries out a search (image comparison) based on the last n query images. Let k_i denote the set of top 10 images that match with the *ith* query image, and let $location(k_i^j)$ denote the location of the *jth* image in that set. The weighted euclidean distance between the *jth* image of set k_i and the *mth* image of set k_{i+1} is given by :

$$d_i^{j,m} = w * |location(k_i^j) - location(k_{i+1}^m)|.$$

Where w is inversely proportional to the weight of image k_i^j.

Path is the set of images (one corresponding to each query image) such that the sum of the weighted euclidean distances between these images is the minimum among all possible combinations. In other words, these images define the shortest possible path (scaled by the weight of images) that the user could have traversed, and also the most likely.

$$Path = \{\forall_{i=1}^{n} k_i^j | D_i^{j,k} = d_i^{j,k} + D_{i+1}^*, D_i^* = min_{j,k} D_i^{j,k}\}$$

The location of the image in set k_n that belongs to the set $Path$ is returned as the current location of the user:

$$current\_location = location(k_n \cap Path).$$

We use the *sliding-window* approach. The window of n images to be considered for calculating the shortest-path slides by 1 with every new query image. The intuition behind using the shortest-path/nearest-neighbours approach is the following: assuming that the user does not abruptly increase her walking/running speed above a certain limit, the current location of the user should not be too far away from her past locations. Shortest-path/nearest-neighbours captures that notion. We use weighted distance in order to give higher priority to images with higher weight (i.e images which match better with the query image). This is the reason for using a multiplicant w, which is inversely proportional to the weight of the image. In effect, instead of estimating the location of the user based solely on the similarity with the query image (as in the Naive and Heirarchical approaches), the location is estimated by first estimating the most likely path that the user could have traversed. The most likely path is the one that maximizes the probability of "seeing" the last n query images. The shortest-path weighted by the degree of similarity can be shown to be within close bounds of the most likely path. Note that every time a new query image is received, the top 10 matches corresponding to each of the last n query images are used, instead of directly using the last n estimated locations, in order to account for possible error in the estimation of past locations. We used n = 5 in our implementation. The first location update is sent after 5 query images have been received. This is the *stability delay* of our approach.

Better algorithms can be used for estimating the location of the user based on past locations by maintaining additional state on the web server. The AI community has been working on developing such algorithms. Among the more popular ones are Monte Carlo localization, Markov localization and Kalman filtering. Gutmann et al [10] present an interesting experimental comparison of these algorithms. It would be interesting to see how these algorithms compare with ours for camera-phone based localization.

3.6 Energy Optimization

Our localization approach comes at the cost of mobile phone energy. There are three ways of optimizing energy consumption:

(1) Network energy is several times more expensive than CPU energy. Instead of sending images directly to the web server, feature extraction can be performed on the phone and the features communicated to the web server. Since feature extraction is a computationally intensive process, extensive experimentation is required to evaluate the trade-offs. Let *cpu_energy* be the average energy spent on the phone for extracting features of an image, *feature_energy* be the average energy spent on the phone for communicating the features to the web server, and *network_energy* be the average energy spent on the phone for communicating an image to the web server. Performing feature extraction on the phone is useful only if : $cpu\_energy + feature\_energy < network\_energy$. This needs to be experimentally verified. We are in the process of implementing feature extraction on the phone which will enable us to carry out these experiments.

(2) Size of an image depends on its resolution. A high resolution image is bigger in size than one with low resolution. By sending a low resolution image to the web server, significant amount of energy can be saved and response time can be decreased. This, however, can affect the accuracy of the result. From our experiments, we discovered that as long as the images stored in the database are similar in resolution to the query image, the image comparison algorithms remain unaffected. Therefore, we take low resolution images both during database creation as well as location querying. Our images are usually 5 KB in size. This reduces energy consumption without compromising accuracy.

(3) As mentioned before, the client on the phone periodically sends query images to the web server and obtains location updates. Energy consumption is directly proportional to the number of query images sent by the client. One way to reduce energy consumption is to send query images infrequently. This would imply infrequent location updates which may affect system performance. In order to decrease the query frequency without affecting the frequency of location updates, we build some intelligence into the web server. We identify *critical images* in the image database which correspond to critical points in the building. For example, the corner of a long corridor (as shown in Figure 5) is a critical point, while the middle of a corridor is not. While the user is walking in the corridor, the web server can *guess* his new location based on his last location and speed, assuming that the user does not change speed or direction abruptly. But when the user reaches the corner, the web server cannot predict which way the user will go. In other words, intersections are critical points.

Using the idea of *critical images* we can decrease the frequency of sending query images without affecting frequency of location updates. When the web server sends back a location update, it piggybacks the value of the time period after which the phone should click and send another image. The value of the time period is decided by the web server based on the position of the user in the building. Let c denote the location of the nearest critical point toward which the user is headed, and let l denote the current location of the user. If the current speed of the user is v (and assuming the user does not change speed or direction abruptly), the value of the time period t is given by: $t = |c - l|/v$, where v is calculated from the last two locations of the user.
The web server keeps sending back location updates by *guessing* the next location of the user from his current location and speed, until it receives a new query image. Note that this approach is applicable to only buildings with a certain structure. More specifically, to buildings with corridors. Our Computer Science department building is suitable for using this optimization.

3.7 Implementation

Figure 4 shows the various components of our system. *Client* and *CreateDB* were implemented in Java using MMAPI [4] which is supported on Symbian OS phones. We used the Nokia 6620 mobile phone. We implemented off-the-shelf algorithms for feature extraction and image comparison, namely: Color Histograms, Wavelet Decomposition and Shape Matching. All the three location determination approaches described in the paper were implemented and tested. For implementing *Naive* and *Heirarchical* approaches only the *Database Creator* had to be modified in order to organize the images in the database appropriately. We implemented the *history-based* energy optimization mechanism, which decides the value of the next time period after which the phone should send another query image. We provide this energy optimization as an optional feature which the user can turn on or off. The reason for not hardcoding this feature into our system is that it is vulnerable to abrupt changes in speed and direction of the user. For example, if the user decides to turn back while going in a certain direction, the web server will keep sending wrong location updates until it

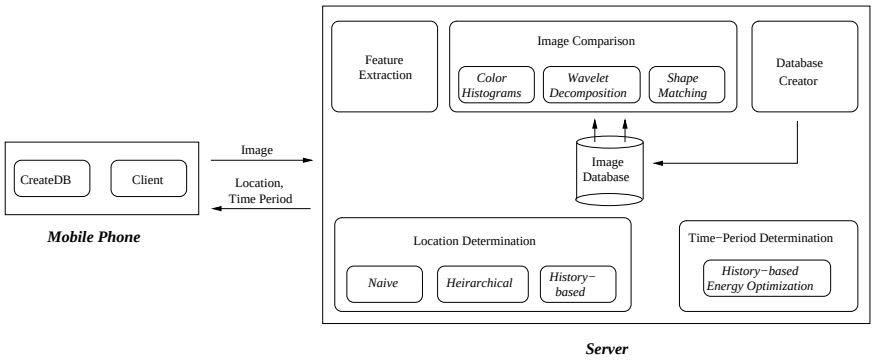

Figure 4: Camera-Phone based indoor localization system architecture

receives a new query image. All the server-side components were implemented in C++ for sake of performance.

3.8 Privacy

Maintaining location information for all users in one central place has the advantage that the users have to trust only one entity, but the disadvantage that everyone is vulnerable to this entity. Practise shows that most people exhibit *crowd* mentality when it comes to taking risks. For example, if many people take service from a web site in return for their credit card information, it is more likely that a new user will join in, as opposed to disclosing credit card number to a hitherto unknown web site. Reputation is an agent of trust and one of the most effective. Therefore, there is reason to believe that centralizing trust may work in favor of privacy and not against it. But at the same time, the possibility of the trusted entity defaulting cannot be ruled out, which in our case is the web server.

We have not yet built any privacy mechanisms into our system, but the following two mechanisms can be easily incorporated:

(1). Instead of sending one query image, the client would send multiple images, where one image would correspond to user's actual location while others would be arbitrary. The web server would send back location corresponding to every image, and the client would pick the one corresponding to his actual location. This would make it hard for the web server to discover the user's location, but not impossible. Sending extra images, however, would increase the energy consumption on the phone. This can be worked around by using a trusted agent as an intermediary between the phone and the location server. Instead of sending the images directly to the server, the phone would send the image to the trusted agent. The trusted agent would insert extra images and send them to the server. The server would send back location information corresponding to all the images to the trusted agent. The trusted agent would pick the location corresponding to the *right* image and communicate it back to the phone.

(2). In our current implementation, during image database creation, the web server extracts the features from the incoming images on the fly and stores them in the database along with corresponding location. The images themselves are not stored. This approach can be extended to preserve privacy in the following manner : during location query, the client would extract features on the phone and send them to the web server instead of sending the image. The web server would store location of every image in an encrypted form which can only be decrypted on the user's phone.

Figure 5: Corridor corner: Critical Point

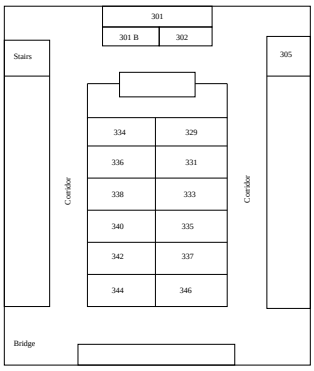

Figure 6: Map of the third floor of Computer Science building

The reason for not storing images on the server side is to eliminate the possibility of discovering the location of an image manually. Discovering the location of an image based on its features is hard and computationally expensive.

The downside of using any privacy mechanism that aims at hiding the location of the user from the web server is that it prohibits the use of *history-based* location determination approach as well as the *history-based* energy optimization approach.

4 Experiments and Results

Figure 6 shows the map of the third floor of the Computer Science building. We created an extensive image database for the marked areas, which includes 16 rooms, staircase, a bridge and the corridors. The goal of the experiments was to test the feasibility of our approach. Our approach has the following two characteristics: (1) it is probabilistic in nature (2) it works with discrete data points as opposed to continuous data points. We therefore sought the answers to the following two questions : (1) How successful is our approach in achieving room-level accuracy?, and (2) How successful is our approach in estimating the orientation and location of the user anywhere in the building?

We conducted three main experiments. The experiments were carried out on two users with five trials per user. In the first experiment, the user would wear the phone as a pendant and enter the room as the camera took a picture. The image database would comprise of only pictures corresponding to user standing at the door of a room and facing inside, with 10 images per door. This experiment was conducted to find out the probability of success for room-level accuracy. Only the Naive approach was used for this experiment. Heirarchical and History-based approaches do not apply to this scenario. Figure 7 shows the pictures of a few rooms that we experimented with. Figure 8 shows an example query image that matches best with Image 3 in Figure 7.

In the second experiment, the image database comprised of only inside-room pictures of all rooms. The experiment was conducted with the user wearing the phone as a pendant and walking around inside rooms, sending a query image every 4 seconds. The value of 4 seconds was chosen arbitrarily and would eventually depend on the application. Figure 9 shows inside-room pictures of a few rooms. Figure 10 shows example of a query image that matches best with Image 4 in Figure 9. Most of the rooms in the Computer Science building are around 4mx7m. We report

Figure 7: Low-resolution pictures of a few rooms taken from the door

Figure 8: Low-resolution query image matches with Image 3

user's location to quarter-room level accuracy (North-East, North-West, South-East, South-West), and report one of the 4 orientations (facing North, facing South, facing East, facing West). The database consists of 10 images per corner to account for varying user heights and angles, and 16 corners per room (4 corners per quarter corresponding to the 4 orientations). For this experiment, all the three approaches were compared, namely: Naive, Heirarchical and History-based.

The third experiment was conducted with the complete database, as the user walked around on the 3rd floor. In all, we have around 300 locations (i.e corners) covered in the database. This includes corridors, staircase and the bridge in addition to rooms. We call this *corner-level* accuracy. Again, all the three approaches were compared. The history-based energy optimization mechanism was used to decide the value of the time period after which the user should send the next query image. When the user is inside a room, this is always 4 seconds.

Table 1 shows the results for the three experiments. The results correspond to the ten trials. When the weight of the best match corresponding to a query image is below a certain threshold, the query image is discarded. This is important to account for corners not covered in the image database. Results show that room-level accuracy as well as quarter-room-level accuracy can be successfully achieved 90% of the time. For quarter-room-level accuracy, History-based approach shows significant improvement over the Naive approach and the Heirarchical approach performs the best among the three. The reason for the superior performance of the Heirarchical approach is that when a query image is receieved, the search is carried out only over the images of the room the user is in, thereby reducing the probability of error.

Naive approach achieves corner-level accuracy 50% of the time. The History-based approach performs significantly better and achieves corner-level accuracy 80% of the time. Heirarchical approach shows an erroneous behavior which is due to the fact that if the room of the user is incorrectly determined, all subsequent searches result in incorrect location until the user exits the room. It is therefore not possible to quantify the probability of success of the Heirarchical approach as one error can propagate and lead to extremely bad results.

From the results, it can be concluded that for applications that require only room-level accuracy, the Naive approach would suffice. Corner-level accuracy corresponds to the most general case: the

Figure 9: Low-resolution pictures of different corners

Figure 10: Query image matches with Image 4

user walking inside rooms, across rooms and inside corridors. History-based approach is clearly the most suitable for such applications. Heirarchical approach performs well most of the time but completely fails sometimes and is therefore not reliable.

We also measured the robustness of the three approaches. Robustness is measured by counting the number of erroneous images it takes to result in an incorrect location. A higher number implies better robustness. Robustness is a useful measure to account for anomalies, such as user's hand covering the camera lens, or the camera getting deflected towards the roof or the floor. Table 2 shows the value of robustnes for nine trials. The average value of robustness for the History-based approach is 2.67. For Naive and Heirarchical approaches, robustness would be 1.

We worked with low-resolution JPEG images of size 5 KB. Our experiments with high-resolution images (128 KB in size) resulted in approximately the same success rate as low-resolution images. By working with low-resolution images we could save substantially on energy as well as response time. Table 3 shows the average response time (i.e latency) for our experiments, as well as the average energy consumption on the phone for sending a query image and receiving one location update. The computation time on the location server is 100-120 msec which is a fraction of the total response time. Response time is dominated by the time taken to send the image to the server and is therefore subject to the bandwidth delivered by the cellular data service, which in our case is GPRS. GPRS delivers a bandwidth of 20-40Kbps, depending on user speed. 3G delivers a bandwidth of around 400Kbps at pedestrian user speeds and over 2Mbps in fixed locations. With the advances in the telecommunications technology, significantly better bandwidths are expected. In order to handle simultaneous requests from multiple clients, multiple instances of the server can be run.

The energy consumption was measured by connecting a constant 3.6V external power supply to the phone and measuring the value of current. Energy(E) and current(I) are related as $E = VIt$ where V is the voltage and t is time. The Nokia phone does not allow an external power supply to be connected, so we carried out these experiments with a Sony Ericsson P900 phone. Since Sony Ericsson P900 does not support MMAPI, we disabled the component that takes pictures. The energy values shown in the table correspond to the total energy consumption on the phone from the instant the camera starts transmitting the picture over GPRS till it receives the location

Table 1: Probability of Success for the Three Experiments

Approaches	Naive	Heirarchical	History-Based
Room-level accuracy	93%	N/A	N/A
Quarter-room-level accuracy	83%	96%	94%
Corner-level accuracy	50%	Non-deterministic	80%

Table 2: Robustness of the three approaches (number of erroneous images that result in incorrect location)

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Average
History-based	3	2	2	5	0	3	3	2	4	**2.67**
Naive	1	1	1	1	1	1	1	1	1	**1**
Heirarchical	1	1	1	1	1	1	1	1	1	**1**

update. When the phone is not being used, it consumes around 4mJ per second.

What is currently missing from our experiments is a thorough user study. A user study would serve two purposes. First, it would help in evaluating the percentage gain in energy obtained by using the history-based energy optimization approach, and second, it would help in evaluating the usability of the system.

5 Conclusions and Future Work

In this paper we showed that it is feasible to achieve indoor localization using just camera phones and off-the-shelf image comparison algorithms. No additional infrastructure is required. More importantly physical objects do not have to be tagged and users do not have to carry any extra device. The only effort is that of building an image database which can be partially automated as discussed in the paper.

We described our approach, pointed out the issues and discussed preliminary solutions. We were able to attain room-level accuracy with more than 90% chance of success and corner-level accuracy with more than 80% chance of success, using a history-based location determination approach. The latency of receiving location updates over a GPRS connection is less than a second, which would be even lower for a 3G connection.

We discovered that the image comparison algorithms remain unaffacted if resolution of the images in the database is the same as the resolution of the query image. By using low resolution images, we were able to reduce energy consumption and response time significantly. We also implemented a history-based energy optimization mechanism, which needs to be evaluated through a user study.

Results can be improved by using more sophisticated image comparison algorithms and overloading them with better planning algorithms which can estimate current location based on past locations. Accelerometers can also help in improving accuracy. Prior work shows that activity, direction as well as the the speed of the user can be estimated from accelerometer data [9, 7, 17, 22, 21]. This can be especially helpful when using the history-based approach for location determination.

Table 3: Energy Consumption and Response Time

Image Size	Avg. Response Time	Avg. Energy Consumption
5KB Image	720 msec	630mJ
128KB Image	4100 msec	3600mJ

We have experimented with a triaxial accelerometer and have been able to determine simple user activities like walking, running, situps, standing etc with more than 95% accuracy using meta-level classifiers. However, currently our system does an offline analysis of the data recieved from the accelerometer. It would be interesting to extend the system to do a real-time analsysis and experimentally measure the improvement in accuracy. Using accelerometer, however, would imply additional cost and overheads.

Effectiveness of the history-based energy optimization mechanism depends entirely on the building structure and user trajectory. A thorough user study (preferably over a number of buildings) needs to be carried out in order to evaluate it. A user study can also help in evaluating the usability of the system and selecting the most energy-efficient privacy mechanism.

Remaining Challenges: Our system is prone to failure in presence of moving objects such as humans, since we do not carry out object recognition. This restricts the applicability of the system to certain kinds of buildings, especially the less crowded ones. The system would work inefficiently in public buildings such as museums. We are currently working on this problem

We have not tested our system under varying lighting conditions, because of the constant lighting conditions inside the building where we tested our system. The results may get affected if the lighting conditions change over time. The effect of sunlight can be partially taken care of by constructing the image database with two sets of images, one corresponding to day-time, and another corresponding to night-time.

References

[1] RoboCup Soccer, http://www.robocup2005.org/home/default.aspx.

[2] Augmented Reality, http://www.augmented-reality.org/.

[3] Magic Eye Project, http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mue/www/magiceye.html.

[4] Mobile Media API(MMAPI), http://java.sun.com/products/mmapi/index.jsp.

[5] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, 2000.

[6] R. Ballagas, M. Rohs, J. G. Sheridan, and J. Borchers. Sweep and point & shoot: Phonecam-based interactions for large public displays. In *CHI '05: Extended abstracts on Human factors and computing systems*, 2005.

[7] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Proceceedings of the 2nd International Conference on Pervasive Computing*, pages 1–17, 2004.

[8] G. Borriello, A. Liu, T. Offer, C. Palistrant, and R. Sharp. Walrus: wireless acoustic location with room-level resolution using ultrasound. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, 2005.

[9] F. Foerster, M. Smeja, and J. Fahrenberg. Detection of posture and motion by accelerometry: a validation in ambulatory monitoring. *Computers in Human Behavior*, pages 571–583, 1999.

[10] J. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods, 1998.

[11] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. In *Mobile Computing and Networking*, 1999.

[12] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[13] T. Kato, T. Kurita, N. Otsu, and K. Hirata. A sketch retrieval method for full color image database. In *In Proceedings of International Conference on Pattern Recognition*, 1992.

[14] T. Kindberg, M. Spasojevic, R. Fleck, and A. Sellen. The ubiquitous camera: An in-depth study of camera phone use. *IEEE Pervasive Computing*, 4(2), April-June 2005.

[15] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for easyliving. In *VS '00: Proceedings of the Third IEEE International Workshop on Visual Surveillance (VS'2000)*, 2000.

[16] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place lab: Device positioning using radio beacons in the wild. In *Proceedings of the Third International Conference on Pervasive Computing*, 2005.

[17] S. Lee and K.Mase. Activity and location recognition using wearable sensors. *IEEE Pervasive Computing*, pages 24–32, 2002.

[18] W. Niblack, R. Barber, and et al. The qbic project: querying images by content using color, texture and shape. In *In Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, 1993.

[19] T. Ofer and M. Ungel. Vision-based fast and reactive monte-carlo localization, 2003.

[20] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000.

[21] C. Randell and H. Muller. Context awareness by analysing accelerometer data. In B. MacIntyre and B. Iannucci, editors, *The Fourth International Symposium on Wearable Computers*, pages 175–176. IEEE Computer Society, 2000.

[22] N. Ravi, N. Dandekar, P. Mysore, and M. Littman. Activity recognition from accelerometer data. In *Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence(IAAI)*, 2005.

[23] M. Rohs. Real-world interaction with camera-phones. In *Proceedings of the 2nd International Symposium on Ubiquitous Computing Systems*, 2004.

[24] M. Rohs and P. Zweifel. A conceptual framework for camera phone-based interaction techniques. In *Proceedings of the Third International Conference on Pervasive Computing*, 2005.

[25] R. Sarvas, E. Herrarte, A. Wilhelm, and M. Davis. Metadata creation system for mobile images. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, 2004.

[26] D. Scott, R. Sharp, A. Madhavapeddy, and E. Upton. Using visual tags to bypass bluetooth device discovery. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1), 2005.

[27] J. Scott and B. Dragovic. Audio location: Accurate low-cost location sensing. In *Proceedings of the Third International Conference on Pervasive Computing*, 2005.

[28] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *In Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems (IROS)*.

[29] M. J. Swain and D. H. Ballard. Color indexing. *Int. J. Comput. Vision*, 7(1), 1991.

[30] S. Thrun, D. Fox, W. Burgard, and F. Dallaert. Robust monte carlo localization for mobile robots. *Artif. Intell.*, 128(1-2), 2001.

[31] E. Toye, R. Sharp, A. Madhavapeddy, and D. Scott. Using smart phones to access site-specific services. *IEEE Pervasive Computing*, 4(2), 2005.

[32] A. Vellaikal and C. Kuo. Content-based retrieval using multiresolution histogram representation. *Digital Image Storage Archiving Systems*, 2602, 1995.

[33] D. Wagner, T. Pintaric, F. Ledermann, and D. Schmalstieg. Towards massively multi-user augmente reality on handheld devices. In *Proceedings of the Third International Conference on Pervasive Computing*, 2005.

[34] R. Want, A. Hopper, V. Falco, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10, 1992.