Location-based augmented reality on mobile phones

Rémi Paucher and Matthew Turk
Computer Science Department
University of California, Santa Barbara
rpaucher@cs.ucsb.edu | mturk@cs.ucsb.edu

Abstract

The computational capability of mobile phones has been rapidly increasing, to the point where augmented reality has become feasible on cell phones. We present an approach to indoor localization and pose estimation in order to support augmented reality applications on a mobile phone platform. Using the embedded camera, the application localizes the device in a familiar environment and determines its orientation. Once the 6 DOF pose is determined, 3D virtual objects from a database can be projected into the image and displayed for the mobile user. Off-line data acquisition consists of acquiring images at different locations in the environment. The online pose estimation is done by a feature-based matching between the cell phone image and an image selected from the precomputed database using the phone's sensors (accelerometer and magnetometer).

The application enables the user both to visualize virtual objects in the camera image and to localize the user in a familiar environment. We describe in detail the process of building the database and the pose estimation algorithm used on the mobile phone. We evaluate the algorithm performance as well as its accuracy in terms of reprojection distance of the 3D virtual objects in the cell phone image.

1. Introduction

There have been many augmented reality systems using various approaches for visual augmentation of the scene. Most of these methods fall into one of two categories: pose computation and object recognition. Many in the AR community use object recognition to provide information about the recognizable objects viewed by the user. However this approach works only if the user is looking at specific known objects; it does not support augmented reality when there are no special objects in the scene. A more flexible approach is to determine the 6 DOF pose of the sensor, which enables the projection of any 3D virtual object from a database into the image the user is viewing at any location in the 3D scene.

Our work uses an approach of this type, computing the sensor pose in an indoor environment and augmenting the scene by projecting 3D information into the image. Among many images of the environment taken at different locations in the environment offline (creating a local database for use in the AR system), the algorithm selects the database image which is the most similar to the live cell phone image. The next step is to match the two images to find point correspondences, based on features extracted in both images. Then the pose between the two images can be computed so that the position and the orientation of the cell phone camera can be known. 3D virtual objects from the database are then projected in the image according to the computed pose as depicted in Figure 1.

Using mobile phones for augmented reality has both advantages and drawbacks. First, cell phones with cameras are becoming ubiquitous, so this is the most convenient platform on which to do augmented reality. AR applications can also benefit from cell phone sensors such as accelerometers and magnetometers, which can improve the quality of augmented reality and facilitate user tracking. However, despite rapid advances in mobile phones as a computing platform, their performance for real-time imaging applications is still very limited. Their computation power is equivalent to that of a typical computer perhaps ten years ago. They also have relatively slow memory access and a small cache. Many do not support floating-point numbers natively, making some computations quite slow.

Many applications tracking cell phone position and orientation send data to a remote computer that does the computations and sends back the results to the mobile device. This approach is not well adapted to augmented reality applications because of bandwidth limitations and the overhead and additional power consumption of data communication. For these reasons, we chose in this work to do all the computation on the mobile device. Considering the trend in cell phone computation power, it will most likely be feasible to develop real-time AR applications processed locally in the near future.

This work aims at showing the feasibility of doing indoor

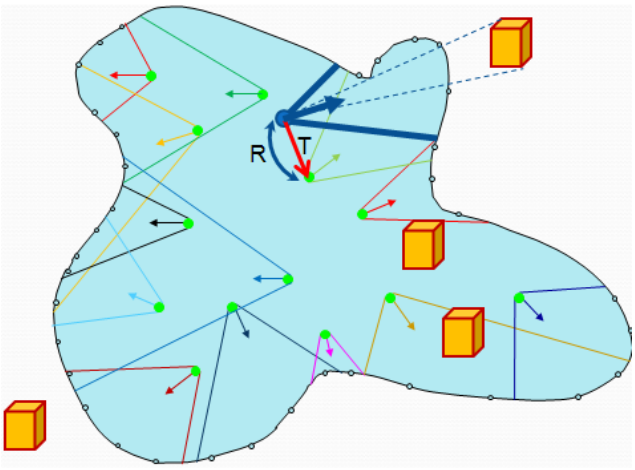

Figure 1. Application overview. The database contains several images taken at different locations (the green dots) in the indoor environment (blue blob). The arrows are the optical axes and the angles represent fields of view. The pose (translation T and rotation R) between the cell phone image (bold blue lines) and a database image is computed by image matching. Then the 3D virtual objects (represented by yellow cubes) are projected into the cell phone image. Here, only one virtual object is seen by the cell phone.

localization and augmented reality on the fly using only the cell phone's sensors and processing. Such a system could be used, for example, in a museum by visitors, to view tour directions and information about pieces of art when pointing their phones at them.

The paper is organized as follows: In Section 2 we discuss the related work. Section 3 will describe the preprocessing steps to learn the environment and the database building process. In Section 4 we will present how the additional cell phone sensors help in pose estimation. Our algorithms for image matching and pose computation on the cell phone side will be described in Section 5, and our results are presented in Section 6. We conclude with a discussion on the results and further work.

2. Related work

Early localization systems used radio signals to determine the user location and relied on a remote server [8] [11]. These systems came with a significant infrastructure cost.

Much work has been done in the robotics community on localization [18] [14] [6] [17]. However, the robots usually have fewer degrees of freedom, and much of the work relies on landmarks. Also many of these systems do not provide metric localization, because they do not deal with a 3D model.

The augmented reality community often uses markers in the environment because of the low computation power requirements and their robustness [16] [15] [19]. Some works use them for localization [9] [10] [13]. However this is an invasive solution since objects have to be tagged with these codes. Our approach does not require one to place any additional content in the environment. Much of the work in the AR community uses object detection and recognition techniques rather than pose estimation.

SLAM approaches like [5] enable to localize the user by building a map on-the-fly. However this kind of approach is more appropriate to small unfamiliar workspaces and provide much less precise maps.

6 DOF pose estimation has become possible on mobile phones due to the increasing computation power available on these devices [20]. However, for large and unconstrained environments pose estimation may be infeasible due to the difficulties of efficiently matching a given image with a database of information about the complete environment. In this work, we consider only database images that are likely to be seen by the camera at its estimated position. Therefore the approach is easily scalable to wide areas, unlike approaches that search the whole database [12]. Arth et al. [1] worked on the problem of localization on cell phones by feature matching, using a potential visible set (PVS) approach to search only relevant images. Their method searches all the point sets adjacent to the camera's location. The embedded sensors on today's cell phones enable us to go a step further and produce an estimate of the camera pose before any processing of the image.

3. Database building

The first step in the application is to acquire data in advance about the environment; e.g., to go through a museum and create a database that will be used subsequently by AR applications in the building. This involves carrying a camera rig through the space and preprocessing this data on a host computer. The data comprises images of the environment taken at different locations. For each image, we store the pose of the camera (its absolute rotation and position) and its intrinsic parameters. Then we extract SURF features [2] in each of the images and store their positions in the image as well as their descriptors. On the cell phone application, we want the user localization to be metric; this requires us to have metric information in the database. Therefore we use a stereo camera for the database images, which produces the depth for each pixel of the image. As a result, we also store the 3D position of the features in each image. For a reason to be explained below, we also store the 3D position of the center of the images.

Among all the detected SURF features, we have to choose the most robust ones; i.e., those that are likely to be detected in the cell phone image and be close to the new feature in terms of descriptor similarity. In order to do that, we track each feature over several frames and keep only the

ones that were successfully tracked over all the frames. The criterion to keep a feature from one frame to the next one is that the feature position remains close to its previous position and that the distance between the descriptors is short enough. In practice, we track the SURF features while the stereo camera remains still. This eliminates many of the features detected because of the noise in the image.

4. Cell phone sensors and pose estimation

Once the database has been collected and processed, the indoor environment is ready to support location-based AR on the mobile phone. As we will see in Section 5, having a coarse estimate of the pose makes the image retrieval step easier and accelerates the pose estimation algorithm. In this work, we used the N97 phone from Nokia. It has several sensors to help us with the pose estimation such as GPS, an accelerometer, a magnetometer and a rotation sensor.

First, when considering cell phone pose estimation, we have to define an absolute coordinate system. As a right-handed cartesian coordinate system, we chose the system $(\overrightarrow{E}, \overrightarrow{g}, \overrightarrow{N})$, where $\overrightarrow{E}$ is East, $\overrightarrow{g}$ gravity and $\overrightarrow{N}$ North.

In this section, we will show how to use the cell phone sensors to produce a camera pose estimation in the defined coordinate system.

4.1. Position

The accelerometer outputs the second derivative of the position, so it is theoretically feasible to obtain the position by double integrating the accelerometer data. However our experiments showed that the data outputted by this sensor is too noisy to get an estimation of the position. Figure 2 shows the results of an experiment comparing the ground truth 2D trajectory and the trajectory estimated with the accelerometer data, while the user walked holding the phone upright. The graph shows a bird's eye view, with an equal number of points in both curves. An accurate trajectory estimate would overlap the rectangular ground truth; in contrast, the accelerometer-based position estimate was wildly off.

Another solution is to use the GPS data which gives the location of the user with a few meters error. Depending on the application, that can be adequate. However, if the system is used indoor there is usually no GPS signal available, so the position cannot be estimated with the cell phone sensors. Therefore in our tests, if there is no GPS signal available, we initialize the user location with the last computed position (i.e., the position computed for the previous frame). Where a GPS signal is not available, we assume the user does not walk more than two meters between two pictures, so that we can initialize his location to the previous computed location.

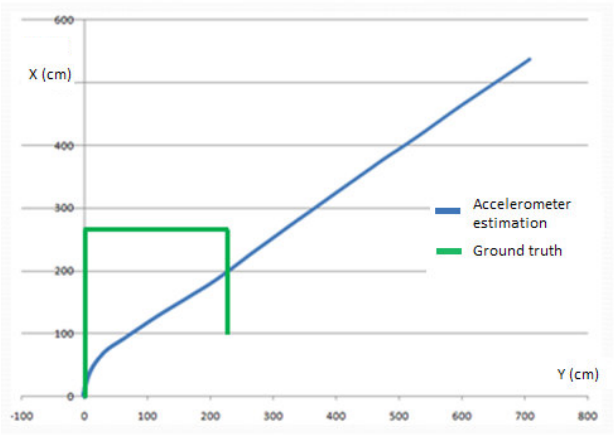

Figure 2. Accelerometer accuracy for trajectory estimation. The 2D position (top-down view) of a walking user holding the cell phone is estimated using the accelerometer data (blue line) and compared to ground truth (green line). The accelerator-only estimate is not useful.

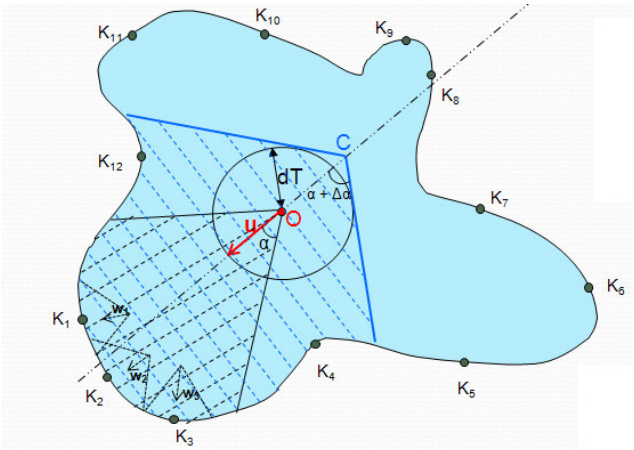

Figure 3. The first criterion of the database search. K_i are the 3D centers of the database images, w_i their optical axes. O and $\mathbf{u}$ are the estimated location and rotation, dT and $\Delta\alpha$ their uncertainties. According to the first criterion, only the database images which have centers inside the blue cone are searched.

4.2. Orientation

The accelerometer sensor outputs three channels that represent the acceleration along the three cell phone axes. Besides measuring acceleration due to the user, the accelerometer also measures the gravity acceleration. Thus if the user is not moving, the three gravity components projected in the cell phone reference can be measured by the sensor. This enables us to obtain the tilt of the phone, that is, two parameters out of three of the cell phone rotation matrix. The same information can also be obtained from the rotation sensor. This sensor gives three angles that rep-

resent the amount of rotation around the three cell phone axes. These angles can be used to retrieve the three components of the gravity vector in the cell phone reference. The advantage of using this sensor rather than the accelerometer is that the outputs of this sensor are not sensitive to user movements. However the sensor is less precise than the accelerometer because the angles are quantized to 15 degrees.

The last parameter to compute is the amount of rotation around a vertical axis. To fully determine the rotation we need additional information from the magnetometer. The 2D magnetometer (or digital compass) outputs the angle from the projection of the North vector onto the cell phone plane and one cell phone axis. This angle gives us one parameter, which is enough to compute the full orientation of the cell phone because we already have two parameters given by the accelerometer/rotation sensor. By expressing the components of all three cell phone axes from the gravity components and the magnetometer angle, we obtain the full cell phone rotation matrix.

5. Cell phone application

5.1. Database search restriction

Once the augmented reality application captures an image, it searches the database for the most relevant stored image. The database can contain an arbitrarily large number of images. Searching among all the images is not feasible, since the matching step is a time-expensive step run on the cell phone. Most of the database images do not need to be searched because the portion of the environment they represent is not even visible to the user. In Section 4 we showed how we could obtain an estimate of the camera pose from the cell phone sensors. Using this estimate, we can discard images that are not likely to being seen by the user; for example, images that represent a scene behind the user. We used two criteria to select only the relevant images.

First, for a database image to be considered, its center has to be seen by the cell phone camera. We can determine this since we stored the 3D point of the center of all the database images. In case the 3D point of the center is not inside the camera's field of view, the overlap region (if there is some) between the two images is not large enough for the pose estimation to be precise enough. Because there is an uncertainty on the phone orientation estimated by the sensors, we have to extend somewhat the field of view of the cell phone camera in which we search for the 3D point. In the same way, we assume that the user location is close to the estimated location (thanks to GPS or the previously computed location). The larger the uncertainties, the larger the number of images to be searched. Figure 3 illustrates this first criterion.

The second criterion prevents bad image matching configurations. The matching step is done using features that are rotation-invariant to some extent. The matching process will fail if there is too much rotation between the two images. Thus we do not search for images that have an orientation that is too different from the camera. This criterion also helps to remove many irrelevant images, such as those with centers that are not seen by the cell phone camera despite being inside its field of view (because of possible walls, or the possible non-convexity of the environment).

Using these two criteria restricts the search significantly. For each database image, we load in memory its 3D center point and the absolute pose of the stereo camera that captures the image. This information is about 36 bytes for each image. The image descriptors are loaded on demand. However the two criteria have some limitations. The best way to restrict the search would be, on top of using these two criteria, to group all images from the same room together, so that only the images taken in the same room as the user will be searched.

5.2. Image retrieval

Among the images that have been selected to be searched, we select the best matching candidate by using a classic feature-based approach. The SURF features are detected in the cell phone image and the descriptors are computed at each of these points. Then for each possible image of the database, the features from the cell phone image are matched to the ones from the database image. For each feature of the cell phone image, the nearest neighbor in the database image features set is searched. Then we select only the good matches, that is, the matches that have a low enough distance between each other, and also the ones for which the ratio between the second best distance and the best distance is high enough. By this method, we obtain sets of good matches between the cell phone image and each of the possible database images. We then select the image from the database that has the highest number of good matches with the cell phone image. This method gives good results in general. The robustness can be increased by adding an outlier removal step that would remove incorrect matches before counting the number of good matches for each image.

The nearest neighbor search of descriptors is performed using a KD-Tree structure. Thus we store a KD-Tree for each image during the preprocessing step. Search is implemented using the Best Bin First technique [3], a technique which is about 90% as accurate as linear search but 100 times faster.

5.3. Outlier removal process

Once the selected database image has been matched with the cell phone image, we obtain a set of candidate matches, from which incorrect matches need to be removed. The most common way to remove outliers between two views

is using a RANSAC [4] loop computing the fundamental matrix at each iteration. Depending on the algorithm used, 5, 7 or 8 points are needed to compute the fundamental matrix. We considered that algorithms that need fewer than 8 points are too time-consuming, even if they are more precise. However the 8-point algorithm uses a linear criterion that gives very poor results in the presence of noise, which is why using it inside a RANSAC loop gives unusable results. This leads both to discarding inliers and to accepting outliers. We cannot use this because the pose computation algorithm is very sensitive to outliers, so it needs to deal only with inliers. Therefore we first remove the most obvious wrong matches by approximating the transformation as planar. Thus we first fit a homography between the two sets of points using the Least Median of Square algorithm. Then the points that lie too far from their image by the homography are considered as outliers. In other words, only points that roughly satisfy a planar criterion are selected. The chosen threshold is of course high, so that depth changes are allowed. This step removes most of the outliers and is fast because it requires only four matches to estimate the homography at each iteration. We can then use the 8-point algorithm combined with RANSAC to remove the final outliers. Since matches have been filtered before this step, we need only a small number of RANSAC iterations.

The 8-point algorithm works well to remove outliers during the last step of the outlier removal, but because of its poor accuracy it fails to give a precise estimate of the fundamental matrix, even when refining the estimate using all of the inliers. It can only be used to remove outliers but not to estimate the pose.

We also considered removing the outliers using the 3D points coordinates to estimate the 6 DOF pose from three points [7] in a RANSAC loop. However this method is more complex because it requires solving polynomial equations which lead to up to four solutions. We chose not to use it because we obtained similar results with our method in less time.

5.4. Pose computation

At this step we have a set of presumably correct matches between the cell phone and the database images, which enables us to compute the relative pose between the two images. Given 2D-3D matches, we have to find the translation and the rotation between the two cameras. In the following, $\mathbf{c_i}$ and $\mathbf{d_i}$ are 2D points in the cell phone and the database images respectively, $\mathbf{X_i}$ is a 3D point in the database system coordinates, and K_c and K_d are the calibration matrices of the cell phone and the stereo camera, respectively. In this section, we propose two different methods to estimate the pose.

5.4.1 Reprojection minimization

The final goal is that the projected virtual objects match the image content as close as possible. Therefore, the reprojection error of the database 3D points in the cell phone image is the most meaningful criterion to minimize. No matter which method we use, the final pose estimation step has to be the minimization of this criterion:

$$\min_{R,\mathbf{T}} \sum_i \left\| \frac{K_c \left(R\mathbf{X_i} + \mathbf{T}\right)}{\left(K_c \left(R\mathbf{X_i} + \mathbf{T}\right)\right)_3} - \begin{pmatrix} \mathbf{c_i} \\ 1 \end{pmatrix} \right\|^2$$

The minimization is done using the Levenberg-Marquardt algorithm, over six parameters (three for the rotation and three for the translation). Of course the pose has to be well estimated before doing this local minimization. In our tests if the number of matches is high, the estimation does not have to be very precise for the global minimum to be found. A pose estimation using the 8-point algorithm is adequate. The problem is now to initialize the pose before doing the final minimization.

5.4.2 First initialization method

Because the 8-point algorithm does not give good results, we propose an alternative algorithm to initialize the pose. We have an estimate ($\widehat{R}$) of the rotation given by the cell phone sensors. We can still minimize the sum of squares as in the 8-point algorithm, but only over the translation parameters, by setting the rotation equal to the sensors estimate. The translation can only be estimated up to scale when using the 2D points, so the minimization is done on only two parameters. Plus we do not have the essential matrix constraints problems from the 8-point algorithm anymore, because we minimize directly over pose parameters. The $\times$ symbol stands for cross product.

$$\min_{\|\mathbf{T}\|=1} \sum_i \left(\mathbf{T}.\left(\mathbf{c}_i \times \widehat{R}\mathbf{d}_i\right)\right)^2$$

The resolution is done using an SVD decomposition on a 3×3 matrix, which is very fast. This is precise because the rotation is well estimated by the cell phone sensors. This same method can be used inside a RANSAC loop to remove the outliers; it requires only two points to determine the translation and is very fast. We have tested this outlier removal technique, but it does not remove all the outliers, and performs worse in general than the homography fit.

Next we refine the pose by minimizing the Sampson criterion, which is more meaningful than a linear criterion:

$$\min_F \sum_i \left(\frac{\left(\mathbf{c_i}^T F \mathbf{d_i}\right)^2}{(F\mathbf{d_i})_1^2 + (F\mathbf{d_i})_2^2 + (F^T\mathbf{c_i})_1^2 + (F^T\mathbf{c_i})_2^2} \right)$$

where the fundamental matrix F is expressed as a function of the pose parameters: $F = {K_d}^{-T}[\mathbf{T}]_\times R {K_c}^{-1}$.

In this minimization, the translation is represented in spherical coordinates and the rotation via the angle/axis representation. The minimization is thus done over five parameters thanks to the Levenberg-Marcquardt algorithm. This minimization is of course local, so it needs a good initialization. We initialize it with the pose estimation from the previous algorithm.

At this point, the translation has only been estimated up to scale. To obtain the translation norm, we use point depths from the stereo camera (database). For each match, we can compute the translation norm value using geometric considerations in triangles. Among all these values for the translation norm, selecting the median is the best choice to be sensitive to noisy matches as little as possible.

5.4.3 Second initialization method

The previous method provides an accurate 6 DOF pose estimation. However it is rather slow because it involves an LM minimization. We propose an alternative method to initialize the pose. Let us consider this minimization ($\widehat{R}$ is the estimation of the orientation using the cell phone sensors):

$$\min_{\mathbf{T}} \sum_i \left\| K_c \left(\widehat{R}\mathbf{X}_i + \mathbf{T} \right) - \left(\widehat{R}\mathbf{X}_i + \mathbf{T} \right)_3 \begin{pmatrix} \mathbf{c_i} \\ 1 \end{pmatrix} \right\|^2$$

This criterion is the linearized version of the final criterion we want to minimize. Of course it is less meaningful, because it gives more importance to the points that have a high depth and to the ones that lie far from the image center. But it has the advantage of being much more easily and quickly minimized. Because the rotation is estimated by the cell phone sensors, the minimization is done over only the three translation parameters. The derivatives of this expression are linear, so it requires only one 3×3 matrix SVD decomposition to compute the translation parameters.

This method is faster than the previous one because it uses only one SVD decomposition. In our experiments it is slightly more robust for finding the global minimum.

6. Results and performance

6.1. Augmented reality

In our tests, we picked in the 3D model some 3D points that correspond to something remarkable in the environment. All the 3D virtual objects we used for the AR experiments are planar rectangles. Although any 3D object can be inserted in the database, rectangles make it easier for visualization and error measuring. We built a database of images in a lab environment and picked rectangles in the 3D model. After the user points the cell phone in the direction of these objects, the pose between the cell phone camera and the database image is computed. The visible objects are then projected into the cell phone image according to the pose estimation. Figure 4 shows the augmented picture; the reprojection error is about four pixels for a 640×480 image. As we can see, the final minimization is a mandatory step. As always, there is a trade-off between quality and time. The bigger the images, the more features will be extracted, so the precision will be better but the algorithm will be slower.

We can have even better results when the rotation is lower. In the previous test, the rotation was about 45 degrees. Figure 6 shows a result where rotation is close to 0 degrees and the error is only about two pixels.

6.2. Localization

This application also enables to get the location (and the orientation) of the cell phone in the environment. Figure 5 shows the result of the localization process while the user was walking inside the lab. The position error is in general about 10-15 centimeters. One can notice one green point alone in the middle of the floor plan. The matching process failed here because the cell phone image quality was inadequate.

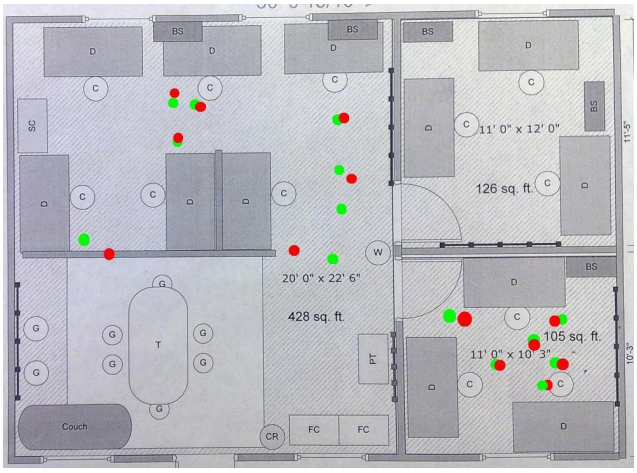

Figure 5. Localization result when the user is walking in the lab. Green points are ground truth and red points are computed positions.

6.3. Computation time

The implementation was done in Symbian C++ and Open C/C++ on a Nokia N97 cell phone. It is equipped with a single-core ARM 434 MHz processor with 128Mb of RAM. The most limiting step in this application is the SURF algorithm, which takes more than 8 seconds to run on a 640×480 image. This is mostly due to the fact that Symbian only emulates the floating point precision because

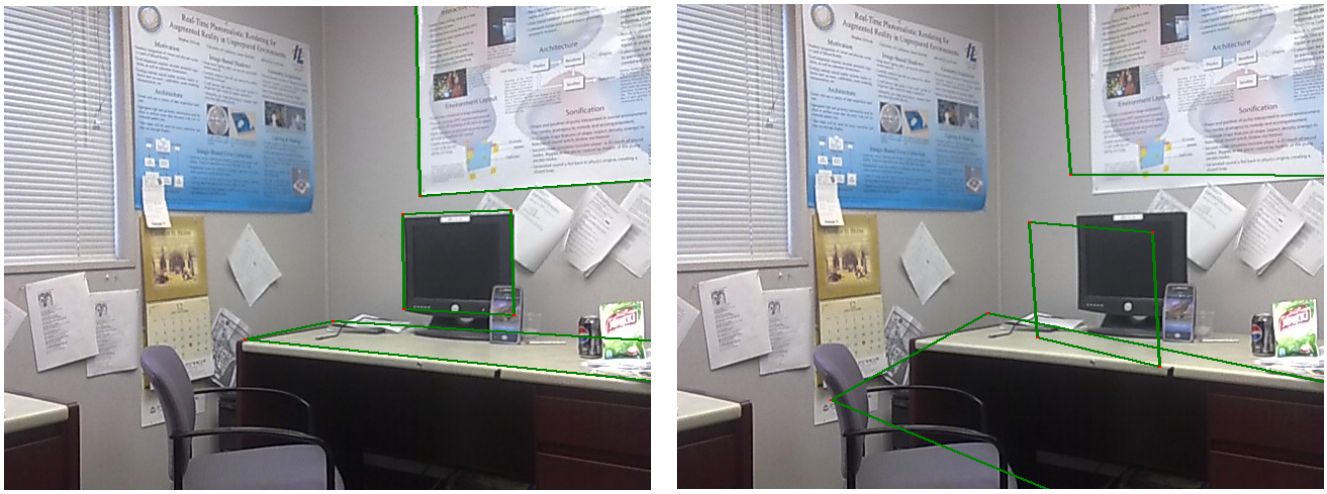

Figure 4. Final augmented picture with (left) and without (right) the final minimization.

Figure 6. Augmented picture when the rotation between the database and the cell phone images is low. Accuracy is better.

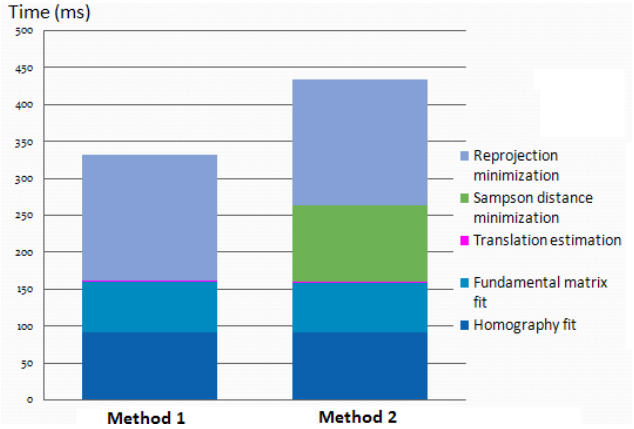

Figure 7. Pose estimation computation time for both initialization methods (5.4.3 and 5.4.2).

it does not natively support it. In our SURF implementation, all the processed data are floating-point numbers, which explains why it takes so much time. This could be reduced by using a fixed-precision version of the SURF algorithm (or using a mobile platform with floating point computation, which are becoming more common). For comparison, the algorithm runs at 10fps for 640×480 images on a 1.73GHz computer.

The second step of the algorithm is the matching process between two sets of descriptors which takes about 1.1 seconds. Again, this time could be significantly reduced if the architecture natively supported floating-point precision.

The computation times for each step of the pose estimation algorithm are displayed in Figure 7. The pose computation takes one third of a second if the fastest method is used, or 430 ms with the other method. All the algorithms also use double precision, which increases execution time.

6.4. Discussion

Each database image generates a 3D point cloud. To have a good registration between all of them, the stereo camera positions and orientations have to be very precise. Only 3D points from one database image are used, so the 3D virtual objects can be described in local coordinates for each image. In other words, we can associate to each image its own 3D objects. In this case, however, the augmentation is more limited because only the 3D objects from one image will be projected in the user image. However having accurate poses for database images is no longer necessary, and we can choose which objects can be projected in the image, which solves occlusion problems due to walls, for example. The accuracy of the algorithm could be improved

if the quality of the images was improved. The stereo camera is very sensitive to brightness and reflective surfaces. To capture good quality images with the stereo camera, the user has to choose manually the parameters (gain, exposure, etc.) that differ for each image. The same goes for the cell phone images. With Symbian APIs, instead of capturing a 640×480 image, the camera actually captures a 480×360 image that is rescaled to 640×480. Thus a lot of precision is lost to the detriment of the matching step. Because of this issue, there is also a large uncertainty in the cell phone calibration parameters that we obtained thanks to Zhang's algorithm [21].

7. Conclusion and further work

In this paper we showed that augmented reality using 6 DOF tracking is feasible on mobile phones. The cell phone is tracked with a precision of about 10-15cm, which translates to error in the augmented images of about a few pixels. For further work, we would like to improve the translation initialization before the final minimization. That could be solved by increasing the frame rate, so that the user movement between two frames would be low enough to be able to initialize the position with the last user position without user speed limitation. Techniques like optical flow or tracking could then be used to avoid searching the database at every frame. Because there is no location estimation from the sensors, the user has to start the application at a known location; otherwise the application has to search all the database images, which is not feasible on the cell phone. This task would be very suitable for a remote server, because the search can be entirely parallelized. This would be helpful to initialize the user location. In addition, we may consider switching from the SURF algorithm to a more lightweight state-of-the-art feature, such as in [20] where real time performance is achieved. Finally, porting the code to an architecture that would have native support for floating-point numbers like the Nokia N900 mobile phone would improve the frame rate drastically.

References

[1] C. Arth, D. Wagner, et al. Wide area localization on mobile phones. In *Proc. IEEE International Symposium on Mixed and Augmented Reality*, pp 73–82, Orlando, FL, 2009.

[2] H. Bay, T. Tuytelaars, et al. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pp 404–417, 2006.

[3] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proc. IEEE Conf. Comp. Vision Patt. Recog*, pp 1000–1006, 1997.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. pp 726–740, 1987.

[5] P. Gemeiner, A. Davison, et al. Improving localization robustness in monocular SLAM using a high-speed camera. In *Proc. of Robotics: Science and Systems IV*, June 2008.

[6] J.-S. Gutmann, W. Burgard, et al. An experimental comparison of localization methods. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[7] R. M. Haralick, C. Lee, et al. Analysis and solutions of the three point perspective pose estimation problem. Tr, 1991.

[8] A. Harter, A. Hopper, et al. The anatomy of a context-aware application. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp 59–68, 1999.

[9] M. Kalkusch, T. Lidy, et al. Structured visual markers for indoor pathfinding. In *Proceedings of the First IEEE International Workshop on ARToolKit*. IEEE, 2002.

[10] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 27, Darmstadt, Germany, 2002.

[11] N. B. Priyantha, A. Chakraborty, et al. The Cricket Location-Support System. In *6th ACM MOBICOM*, Boston, MA, August 2000.

[12] N. Ravi, P. Shankar, et al. Indoor localization using camera phones. In *Proceedings of the Seventh IEEE Workshop on Mobile Computing Systems & Applications*, pp 1–7, 2006.

[13] G. Reitmayr and T. W. Drummond. Initialisation for visual tracking in urban environments. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp 1–9, 2007.

[14] T. Röfer and M. Jüngel. Vision-based fast and reactive monte-carlo localization. In *IEEE International Conference on Robotics and Automation*, pp 856–861, 2003.

[15] M. Rohs. Real-world interaction with camera-phones. In *2nd International Symposium on Ubiquitous Computing Systems*, pp 74–89, Tokyo, Japan, Nov. 2004.

[16] M. Rohs and P. Zweifel. A conceptual framework for camera phone-based interaction techniques. In *Proc. Pervasive*, pp 171–189, 2005.

[17] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 1998.

[18] S. Thrun, D. Fox, et al. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.

[19] E. Toye, R. Sharp, et al. Using smart phones to access site-specific services. *IEEE Pervasive Computing*, 4(2):60–66, 2005.

[20] D. Wagner, G. Reitmayr, et al. Pose tracking from natural features on mobile phones. In *ISMAR '08: Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp 125–134, 2008.

[21] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 1998.