Mobile Augmented Reality based 3D Snapshots

Peter Keitler*, Frieder Pankratz*, Björn Schwerdtfeger*, Daniel Pustka*, Wolf Rödiger*, Gudrun Klinker*, Christian Rauch†, Anup Chathoth†, John Collomosse⋆, Yi-Zhe Song‡

*Fachgebiet Augmented Reality
Technische Universität München
Garching b. München
Germany
E-Mail: keitler@in.tum.de

†Vodafone Group Services GmbH
– Vodafone Group R&D Germany
München
Germany
E-Mail: christian.rauch@vodafone.com

⋆CVSSP
University of Surrey
Guilford
Surrey
United Kingdom
E-Mail: J.Collomosse@surrey.ac.uk

‡Dept. Computer Science
University of Bath
Bath
Avon
United Kingdom
E-Mail: yzs20@cs.bath.ac.uk

Abstract: In this paper, we present a mobile augmented reality application that is based on the acquisition of user-generated content obtained by 3D snapshotting. To take a 3D snapshot of an arbitrary object, a point cloud is reconstructed from multiple photographs taken by a mobile phone. From this, a textured polygon model is computed automatically. Other users can view the 3D object in the environment of their choosing by superimposing it on the live video taken by the cell phone camera. Optical square markers provide the anchor for virtual objects in the scene. To extend the viewable range and to improve overall tracking performance, a novel approach based on pixel flow is used to recover the orientation of the phone. This dual tracking approach also allows for a new single-button user interface metaphor for moving virtual objects in the scene. The Development of the AR viewer was accompanied by user studies and a further summative study evaluates the result, confirming our chosen approach.

Keywords: Mobile augmented reality, 3D snapshotting, tracking, user interface

1 Motivation

We present a mobile augmented reality (AR) platform based on user-generated content. The core idea is to enable a user of our system to generate a 3D model of arbitrary small or mid-sized objects, based on photographs taken with their mobile phone camera. Thereupon, another user can inspect the object integrated in their natural environment, e.g. their home, using our mobile AR viewer application. We refer to this capture and viewing process as "3D Snapshotting"

3D models can be generated on-the-fly, using a pair of photographs of an object from different perspectives and reconstruct its 3D structure from them. This results in a dense 3D point cloud.

Figure 1: Mobile AR viewer

However, they cannot be rendered directly, especially on mobile platforms, since graphics hardware is optimized for polygon models. One challenge for 3D snapshotting is therefore to reduce model complexity by automatically computing a textured mesh without significantly deteriorating aesthetic quality.

Another user can then inspect an object created by 3D snapshotting using the AR viewer installed on their mobile phone. It overlays the object on the live video, as shown in Figure 1. An optical square marker serves as an anchor for the object in the natural environment of the user. An extension based on pixel flow enhances the robustness of tracking and enlarges the usable viewing range.

This poses interesting questions regarding the usability of such a system. Can mobile phones perform sufficiently well to satisfy user's expectations with respect to the described AR application? How can users interact with the 3D scene most effectively? A demonstrator running on an HTC Touch Diamond has been developed to investigate those issues.

Wagner [2] and Henrysson [5] investigated various technical aspects of mobile AR applications. One of the fiercest barriers for developing AR tracking systems for mobile phones is their limited computing power. Therefore, porting existing tracking methods to mobile phones is quite challenging. A successful port of the ARToolKit marker tracking library was done by Wagner et al., released under the name ARToolKitPlus [12].

Another restriction of mobile platforms is the lack of a floating point unit (FPU). This forces the developer to either rely on a computationally expensive software implementation or to use fixed point arithmetic instead, which however introduces restrictions in the range of representable values and also in the achievable precision. Drab et al. [3] introduced a motion detection algorithm called

projection shift analysis (PSA), designed particularly for mobile devices with limited resources. The algorithm computes the shift in x and y direction between consecutive frames of a video, using only integer arithmetic.

2 The 3D Reconstruction

We reconstruct a 3D mesh model of an arbitrary object, using a pair of photographs supplied by the user. These photographs are taken under similar lighting conditions, from different points of view, and are typically captured as JPEG images using their camera phone. SIFT features [7] are identified within the photographs, and matched to identify parts of the object common to both viewpoints. The fundamental matrix, encoding the geometry between the views, is recovered by analyzing the SIFT point matches via MAPSAC [10]. Camera intrinsic parameters are estimated either using the EXIF data captured in the JPEG file [11], or using a calibration rig. We input these data with standard dense reconstruction algorithms to infer the 3D positions of the matched SIFT points. Additional 3D points are recovered using dense match propagation, yielding a cloud of 5k reconstructed 3D points for a typical small or mid-sized object. We create a triangular mesh over the 3D points by backprojecting those points to their 2D positions in one of the original photographs, and performing a Delaunay triangulation. Texture for each 3D mesh triangle is cut from the corresponding 2D triangle within the original photograph. At this stage the mesh is simplified by merging adjacent mesh triangles where color and surface normal are near-identical. A graph is constructed where each node corresponds to a mesh face, and weights correspond to color and surface normal similarity. Graph cut [9] is applied to perform simplification of the mesh. Simplification is essential to reduce the rendering complexity for the mobile AR viewer. The profiling results in Table 1 show that even with simple objects, the application is mostly occupied with rendering. The important stages of the reconstruction process are depicted in Figure 2. Due to complexity of the 3D reconstruction and triangulation algorithms, the reconstruction is executed as a web service instead of on the mobile phone. Modern cellular bandwidth enables photographs and mesh geometry to be easily transferred.

3 AR Viewer Requirements

For a first informal user study, we developed an AR viewer based solely on optical square marker tracking, as described in Section 4. The main focus of the study was to observe how the participants interact with the phone and the test application, to gather requirements for the tracking system and application interface (see Section 5). The HTC Touch Diamond has a Qualcomm® MSM7201A™ 528 MHz CPU and runs Windows Mobile. The integrated camera provides an RGB565 video stream of 240x320 pixel at 15 fps. The 2.8 inch touch screen display has a resolution of 480x640 pixel (260 dpi).

The participants where not told how the marker tracking system worked and what flaws it had. They were just informed that a virtual object would appear on top of the marker. The intention was to observe how the users would naturally interact with the system. Five participants had to arrange

Figure 2: Steps of the reconstruction process. SIFT features are used to match images (upper left), a dense point cloud is reconstructed from matching points (bottom left), finally a textured mesh is produced (right).

simple furniture objects in a room. More concretely, the tasks were to align them with real objects, orient them in a certain direction or inspect them from different perspectives.

The participants, most of them seeing an AR application for the fist time, did not know about the flaws of optical marker tracking and moved the mobile phone too fast in the beginning. This resulted in strong motion blur causing the marker tracking to fail, an effect that was even amplified by the long shutter times of the integrated phone camera.

Another big problem occurred in conjunction with big objects such as tables and chairs. Since the field of view of our phone camera is rather small, the virtual objects occluded almost the complete display, making the participant loose the context provided by the real world. When the marker was close to the participant, he had to move the phone to inspect the whole object which resulted in the loss of the marker tracking because the marker went out of scope. In case the marker was further away from the participant, the field of view wasn't as much of a problem, but the pose of the marker began to jitter, depending on its size, which the participants felt as very troublesome. Similar effects have already been reported by Freeman [4] in conjunction with ARToolKit and explained mathematically by Schweighofer [8]. In the end, the participants developed strategies to compensate for this, for example, standing completely still in front of the marker, interacting only with the methods provided by the AR viewer (see Section 5). Thus the application became less intuitive. From this first user study, we also learned that the participants preferred an inaccurate but stable pose over an unstable pose.

During the user study, we also noticed that users typically do not translate the phone but rather rotate it around axes lying in the wrist, elbow or hip, see Figure 3. This offers the opportunity for simplified pose estimation when the marker is absent. Summarizing the results, the study suggests that optical square marker tracking alone is not enough to provide a satisfactory user experience.

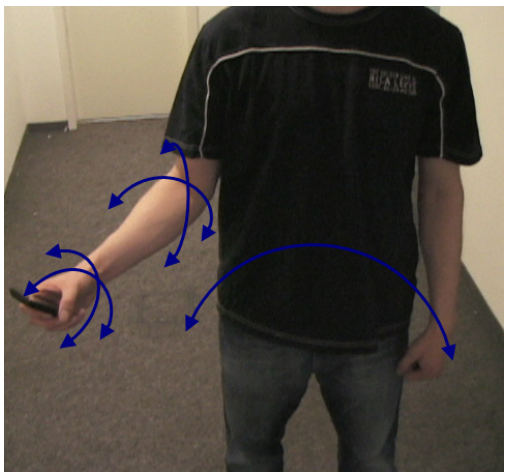

Figure 3: Commonly used rotation axes

4 The Tracker

Based on the experiences of the formative user study described in Section 3, a tracking system suited for mobile devices was developed. The drawbacks of the marker tracker that had already been used for the study were then compensated for by integrating the efficient projection shift analysis (PSA) algorithm to recover the orientation of the mobile phone even without a visible marker.

4.1 Marker Tracker

The tracking algorithm is functionally similar to known methods [2], we therefore concentrate on some optimization details. The first steps in the marker tracking pipeline are a conversion of the source image from color to grayscale, reduction of the resolution and a thresholding, resulting in a binary image suitable for corner detection. To improve the performance of the system, the conversion of the RGB565 formatted camera image into an 8 bit gray scale image was optimized using a look up table. The RGB565 color value of a pixel is interpreted as an unsigned short value and used as an index to get the corresponding 8 bit gray value from the look up table. 64kb additional memory is required for the look up table, but a speed up of 24% was observed. For more detailed time measurements see Table 1. Furthermore, the resolution is cut in half from 320x240 to 160x120 pixels. This allows for efficient implementation of linear interpolation by calculating the mean values of 2x2 pixel blocks for each pixel in the smaller result image. The lower resolution not only improves the performance of marker tracking, it also reduces the effects of noise and motion blur and allows more stable tracking results. The scaled grayscale image is not directly discarded since it is also used by the PSA algorithm (see below).

Then, quadrangles are found using OpenCV functions [1]. OpenCV has been partially ported to Windows Mobile for this purpose. After detecting a rectangle, its corners are refined with sub-pixel accuracy by fitting lines to the edges of the marker. Finding and refining the quadrangles takes up about 71% of the total time needed for marker tracking. The OpenCV functions for

finding contours and approximating polygons aren't that computationally expensive and take only 23% of the needed time. 73%, however, is spent optimizing the corners of the quadrangle with sub-pixel accuracy, in particular for the interpolation between pixels for the refinement of edges. While the interpolation of one pixel value takes only 0.0027 ms, this function is called hundreds of times during one marker detection cycle.

The next steps are to detect the ID of the marker coded in the interior of the rectangle and to compute a unique marker orientation in 2D from it. Having uniquely determined the four corner points, an initial guess of the marker pose in 3D can be computed and optimized using the Levenberg-Marquardt algorithm. This takes up about 19% of the overall time needed for the marker tracker.

A fixed point implementation is used for decimal arithmetic in most cases. Floating point numbers are only used where high precision is necessary, e.g. in the pose estimation. To improve computational accuracy, it does not have a static 16.16 bit format but allows for the adaption of the number of decimal places according to the problem at hand. Quaternion operations for example highly benefit from this technique.

To reduce the jitter of the pose, a double exponential smoothing predictor[6] is used. First, the positions of the four corner points of the marker are smoothed to reduce jitter when the marker becomes relatively small. After that, the pose results from the marker tracker are extrapolated based on a linear motion model and smoothed to reduce the lag of the pose that is introduced by the smoothing of the corner points.

4.2 Pixel Flow

The projection shift analysis (PSA) algorithm, originally designed to use the motion of the phone as an interaction technique, was adapted to compute the orientation of the phone [3]. The algorithm computes the horizontal and vertical shift between two consecutive frames. First, it calculates the sums of gray values for every row and column in the current image, resulting in two vectors of accumulated gray values, one for the horizontal and one for the vertical shift. These vectors are compared with the vectors from the previous image to determine the shift of the current image. The optimal shift is found by minimizing the sum of squared differences (SSD) over all possible shifts. To increase the performance and stability of the algorithm, the SSD is only calculated for a predefined range of shifts. Tests suggest a range of 50% of the image size. This eliminates the risk of detecting huge shifts caused by similar background colors and patterns. Only integer operations are needed.

This produces stable and fine grained results most of the time but the algorithm is still vulnerable to motion blur. Thus, the PSA is invoked by default on the image with halved resolution already used by the marker tracker. Depending on the SSD value and a predefined threshold, the image is maybe scaled down again by 50% and PSA is repeated. This is done two more times, if necessary, up to one eighth of the original resolution in the worst case. This can be implemented efficiently by just scaling down the row and column intensity vector results of the previous PSA execution instead of scaling down the whole image.

4.3 Fusion

As long as the marker is being tracked by the system, the marker pose is considered to be accurate. Once the marker is lost, the horizontal and vertical shift of the current image with respect to the previous image is used to estimate the orientation change of the mobile phone around its X and Y axes. The last known marker pose is incrementally updated using these values. Translation of the phone is assumed to be negligible compared to changes in orientation (as noted in our earlier study). The conversion from shift in the image to the rotation of the phone is done by using two factors, one for the X and one for the Y axis. They describe the degree of rotation corresponding to one pixel shift. When choosing these factors, it has to be considered that depending on the objects and their distance to the camera, one degree of rotation can result in different pixel flows. Objects further away from the camera result in a higher pixel flow. To take this into account, the factors are constantly being adjusted as long as the marker is still visible. This way, the system automatically adapts to the current scene.

The advantages of this approach are that the users don't need to keep the marker in the visible area all the time. They can interact more naturally with the application and their environment, even when the marker is lost. Another advantage is that the results of the pixel flow can be used for other purposes, e.g. an alternative interaction method for the application (see Section 5).

PSA also has some drawbacks, too. If the background is flat or consists of repeating patterns (such as checker board), the it produces inaccurate results. Movements in the background are problematic, too. Since the pixel flow isn't 100% accurate, a drift will accumulate over time, resulting in an instantaneous jump of the pose as soon as the marker becomes fully visible again.

5 The Augmented Reality Viewer

The user interface of the AR viewer consists of a toolbar overlay. It contains buttons to load, select and erase virtual objects, to change settings, to toggle between view and object manipulation mode as well as several tools to arrange objects in space. In particular, this comprises the operations move, rotate, lift and scale. They were identified in our first user study described in Section 3.

Drop shadows are rendered at the bottom of the objects to improve the sense of immersion. While being tracked, the square marker is highlighted with a green selection to provide visual feedback. During user interaction with one particular object all other objects are rendered transparently to emphasize the current selection. In object manipulation mode, two different interaction techniques are provided:

- *Touch manipulation:* Once a function has been selected, the user has to drag a line on the screen with their finger. The relative movement on the screen is used to directly manipulate the attributes of the selected object, e.g. the position in the plane defined by the marker (move) or the height (lift), using a predefined factor for each function. These factors where defined using extensive testing.

- *Flow manipulation:* Using the touch screen for user input requires both hands most of the time. Using the pixel flow provided by the tracking system, we can implement an interaction method for only one hand. First, the desired manipulation function is selected from the toolbar, preferably using the thumb. Then, instead of using the touch screen to, e.g. move an object, the user has to press a central button on the phone while moving the phone in the according direction(s). Again, the relative movement is used as input for the selected manipulation function, except that other conversion factors are used.

Early testing with some users also showed that imprecise results from the pixel flow, e.g. if downscaling was disabled, have a greater impact on the object manipulation than on the augmentation itself. While a drift of the augmentation was tolerated by most of the users, flow manipulation, for example while moving an object, is highly sensitive to errors.

6 Evaluation and Results

Based on the functionality described so far, we performed a second informal summative user study with 8 subjects. Again, the focus of the study was on the interaction of the participants with the mobile phone and the AR viewer. Like in our first user study, the participants where not told at first how the tracking system worked and what flaws it still had. This time the participants had to place different objects onto other objects, view an about 2 meters tall virtual advertising column and identify 3 posters that where placed around the top of it, thereby forcing them to use the pixel flow tracking.

The overall satisfaction when using the application increased drastically, partly because of the stabilized marker position, but mostly because the participants did not have to pay much attention to the marker and could freely move the phone without worrying about loosing it. The participants were still standing in front of the marker, rotating the phone about an axis in the wrist, elbow or body, approving our assumption about the movements of the user.

But with the freedom the new system granted, some users tried to view the top of the advertising column and move around the object at the same time. The tracking system doesn't support such movements, so the users didn't get the result they expected. After explaining to them the restrictions of the new tracking system, the users quickly adapted to it. They either went around the marker, looked at the marker for a few moments to let the system register the new position or they used the interaction methods provided by the application to rotate the advertising column.

Most of the participants didn't notice the drift at first or where not bothered by it. Only a few participants repeatedly checked whether the object was still at the marker position.

The user study showed that the flow manipulation is particularly suited for translating the object in the X/Y (move) and Z (lift) directions, but unusable for rotation since the required movement of the phone (rotating the phone around its Y axis) would make the object leave the field of view. Scaling the object using the flow manipulation is possible, but since it requires the phone to move, the point of view changes and so the users where not sure anymore to which size they wanted to scale the object since they lost their reference points.

Component	*Time [ms]*
AR Viewer	100.0
Camera	12.10
Tracking system	13.0
Image conversion	2.20
Marker tracker	9.85
Find quadrangles	7.00
Find contours (OpenCV)	1.00
Approximate polygons (OpenCV)	0.63
Refine Quadrangles	5.10
Interpolate pixel values	2.26
Identify marker	0.0058
Estimate pose	1.90
Pixel flow	0.87
1/2 resolution	0.86
1/4 resolution	0.00098
1/8 resolution	0.00033
Rendering	74.91
Draw background	17.56
Draw object	57.35

Table 1: Typical time measurements for AR viewer rendering a model with 1125 triangles at 10.0 FPS

Even though the touch manipulation was at a much better stage of implementation at this time, the users preferred the flow manipulation for moving the object. Only when the given task required precision, the users switched to the touch manipulation.

Acknowledgements

This topic has been initiated at the Vodafone Group R&D academic flagship conference in November 2007. The research activities have been supported and supervised by Vodafone Group R&D.

References

[1] G. Bradski. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[2] W. Daniel. *Handheld Augmented Reality*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, 2007.

[3] S. A. Drab and N. M. Artner. Motion detection as interaction technique for games & applications on mobile devices. In *Pervasive Mobile Interaction Devices (PERMID 2005) Workshop at the Pervasive 2005*, Munich, Germany, May 2005.

[4] R. Freeman, S. Julier, and A. Steed. A method for predicting marker tracking error. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–4. IEEE Computer Society, 2007.

[5] A. Henrysson. *Bringing Augmented Reality to Mobile Phones*. PhD thesis, Linköping UniversityLinköping University, Department of Science and Technology, The Institute of Technology, 2007.

[6] J. J. LaViola Jr. An experiment comparing double exponential smoothing and kalman filter-based predictive tracking algorithms. In *VR '03: Proceedings of the IEEE Virtual Reality 2003*, page 283, Washington, DC, USA, 2003. IEEE Computer Society.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[8] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2024–2030, 2006.

[9] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[10] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24:271–300, 1997.

[11] N. S. University, N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM Transactions on Graphics*, pages 835–846. Press, 2006.

[12] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In H. G. Michael Grabner, editor, *Computer Vision Winter Workshop*, February 2007.