Online Creation of Panoramic Augmented-Reality Annotations on Mobile Phones

A novel application lets users create panoramic images in real time on a mobile phone and annotate the physical environment using an augmented-reality interface. Annotations can be accurately mapped to the correct objects, despite varying user positions.

Tobias Langlotz
Graz University of Technology

Daniel Wagner
Qualcomm Austria Research Center

Alessandro Mulloni and Dieter Schmalstieg
Graz University of Technology

Jim Spohrer first envisioned the idea of superimposing georeferenced information using augmented reality (AR) in his 1999 essay on the WorldBoard.[1] This idea has recently gained popularity with applications such as Layar (http://layar.com), which use camera phones equipped with a compass and GPS as an inexpensive, albeit crude, platform for AR. However, GPS sensors and compasses have limited accuracy and can't provide precise pose information. Furthermore, these sensors have update rates of approximately 1 Hz, so overlays onto the live video image in a mobile phone's viewfinder are roughly placed, sometimes resembling a directional hint rather than an overlay matched to an exact location.

Here, we present a novel system that improves compass accuracy using vision-based orientation tracking, enabling accurate object registration. However, vision tracking can only work in relation to an image database or 3D reconstruction, which must either be predetermined or constructed on the fly. We thus employ a natural-feature mapping and tracking approach for mobile phones that's efficient and robust enough to track with three degrees of freedom. By assuming pure rotational movements, the system creates a panoramic map from live video on the fly and simultaneously tracks from it (see Figure 1).

We also investigate how to annotate the environment directly on the mobile phone. Previous authoring tools were mostly bound to desktop computers or could operate only at the accuracy of the employed mobile sensors. Our approach lets users create annotations at that moment and store them in a self-descriptive way on a server for later re-identification. We identify the label positions using template matching against the panoramic map, so users can register annotations with the environment even if their current position differs slightly from the original position.

Consider the following example. Peter creates a panoramic map and labels objects of interest (see Figure 2). The system transmits to the server the annotations, Peter's GPS location, and a description of the annotated area's visual appearance. Later, Mary wants to retrieve Peter's annotations. Her phone, using GPS information, notifies her when she's close to the locations

Published by the IEEE CS ■ 1536-1268/12/$31.00 © 2012 IEEE

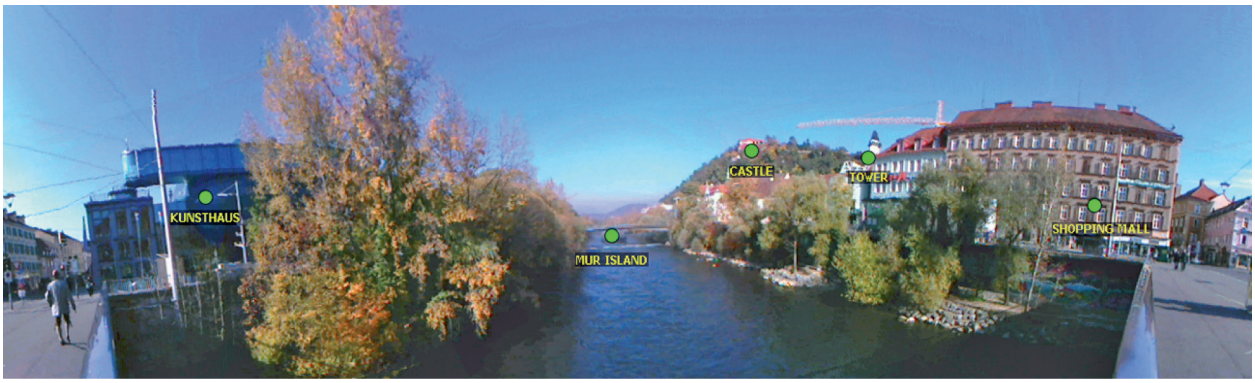


Figure 1. Our vision-based system presents an improvement over regular compass-based annotation systems. By creating and storing panoramas, it can locate and visualize annotations with pixel accuracy.

Peter annotated. A map view lets her reach a spot close to where Peter was when he created the annotations. After Mary points the phone upward, the phone creates a new panorama to efficiently match Peter's annotations to the environment. Mary's phone displays the corresponding annotation as soon as it detects a particular annotation's supporting area. Mary can now create additional annotations.

Panoramic Mapping and Tracking

The system uses a simultaneous mapping and tracking approach, operating on cylindrical panoramic images. Its algorithm is conceptually comparable to simultaneous localization and mapping (see the "Related Work in Augmented Reality" sidebar). However, we don't create a 3D map of the environment; instead, we limit the map to a 2D panorama. This lets users operate the application from any assumed standpoint—they needn't walk to designated hotspots. It also corresponds well to the way in which people explore an environment—that is, by finding an interesting location and then looking around. Furthermore, the system runs at real-time rates of up to 30 Hz on mobile phones and can be deployed spontaneously, because it doesn't require any preparations.

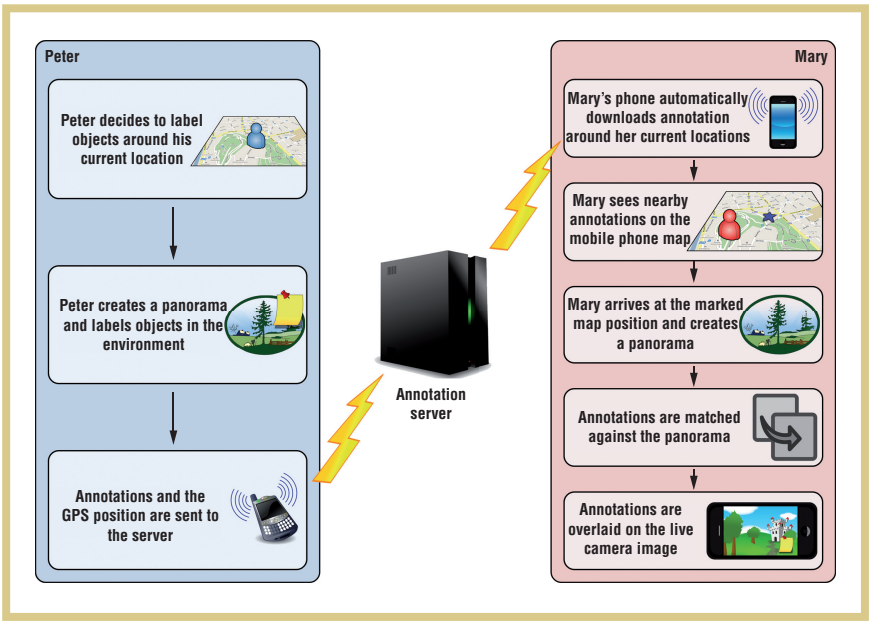


Figure 2. The workflow of the panoramic augmented-reality (AR) annotation system involves two users. Peter creates annotations, and later on, Mary browses through them.

We briefly introduce our method here (more detailed information appears elsewhere[2]).

Panoramic Mapping

Our panoramic mapping method assumes that the camera undergoes only rotational motion. Under this constraint, there are no parallax effects, and we can map the environment onto a closed 2D surface. Although a perfect rotation-only motion is unlikely for a handheld camera, our method can tolerate enough error for casual operation. Mapping errors tend to be negligible, especially outdoors, where distances are usually large compared to the mobile phone's translational movements. A detailed analysis of the effect of violating the pure rotation requirement with respect to the distance of the mapped objects appears elsewhere.[3]

Related Work in Augmented Reality

We can divide previous related work on augmented reality (AR) into two areas: annotation authoring and approaches for tracking mobile devices in large-scale environments in real time.

Current AR applications on mobile phones augment the world with annotations bound to physical objects using the current GPS position and orientation information from an accelerometer and a digital compass. These kinds of applications resemble Spohrer's WorldBoard,[1] a georeferenced information display using AR on a handheld device. The Touring Machine was the first prototypical mobile AR system to demonstrate the advantage of augmenting information over physical objects by using a backpack AR system and head-mounted display.[2] Later work by Rob Kooper and Blair MacIntyre showed how to link the display of georegistered information with AR to online information sources such as the Web.[3] However, these prototypes achieved acceptable registration performance using bulky equipment or stationary infrastructure and weren't intended for daily use.

Jose Montiel and Andrew Davison created a visual compass based on single-camera simultaneous localization and mapping (SLAM).[4] They used an extended Kalman filter formulation of the tracking problem to compute orientation from dynamically acquired landmark features. Their approach creates a sparse 3D reconstruction of the environment, so the system isn't restricted to rotations. Gerhard Reitmayr and his colleagues describe a SLAM system for sharing dynamically generated annotations with a remote observer.[5] Georg Klein and David Murray recently introduced a variant of SLAM-based tracking that can run on mobile phones.[6] However, all these SLAM systems work only in small areas, and the maps aren't designed to store annotations permanently.

Only a few related works focus on creating annotations directly in an AR view. Early work on in situ authoring placed virtual objects in the real scene and to support users through triangulation from different views.[7] Jun Rekimoto and his colleagues presented Augment-able Reality, which lets users annotate an environment prepared with barcode markers referring to contextual information.[8] More recently, Jason Wither and his colleagues showed how to add depth to annotations using aerial maps.[9] Later, they used a laser range finder to automatically calculate the depth information from a given position and orientation, allowing better label placement.[10]

Envisor, on the other hand, uses a vision-based approach for orientation tracking.[11] It tracks the camera orientation in real time and simultaneously creates an environment map by calculating the optical flow between successive frames. These measurements are refined with more computationally expensive landmark tracking to avoid the drift that frame-to-frame feature matching introduces. Although the results of this approach are similar to our approach, Envisor can't run on phones due to high computational costs, because the method requires extensive GPU processing to run in real time.

REFERENCES

1. J.C. Spohrer, "Information in Places," *IBM Systems J.*, vol. 38, no. 4, 1999, pp. 602–628.
2. S. Feiner et al., "A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring The Urban Environment," *Personal and Ubiquitous Computing*, vol. 1, no. 4, 1997, pp. 208–217.
3. R. Kooper and B. Macintyre, "Browsing the Real-World Wide Web: Maintaining Awareness of Virtual Information in an AR Information Space," *Int'l J. Human-Computer Interaction*, vol. 16, no. 3, 2003, pp. 425–446.
4. J. Montiel and A. Davison, "A Visual Compass Based on SLAM," *Proc. 2006 IEEE Int'l Conf. Robotics and Automation* (ICRA 2006), IEEE Press, 2006, pp. 1917–1922.
5. G. Reitmayr, E. Eade, and T. Drummond, "Semi-Automatic Annotations in Unknown Environments," *Proc. IEEE Symp. Mixed and Augmented Reality*, IEEE CS, 2007, pp. 67–70.
6. G. Klein and D. Murray, "Parallel Tracking and Mapping on a Camera Phone," *Proc. IEEE Symp. Mixed and Augmented Reality*, IEEE CS, 2009, pp. 83–86.
7. Y. Baillot, D. Brown, and S. Julier, "Authoring of Physical Models Using Mobile Computers," *Proc. 5th Int'l Symp. Wearable Computers* (ISWC 01), IEEE CS, 2001, pp. 39–46.
8. J. Rekimoto, Y. Ayatsuka, and K. Hayashi, "Augment-able Reality: Situated Communication through Physical and Digital Spaces," *Proc. 2nd IEEE Int'l Symp. Wearable Computers* (ISWC 98), IEEE CS, 1998, p. 68.
9. J. Wither, S. DiVerdi, and T. Höllerer, "Using Aerial Photographs for Improved Mobile AR Annotation," *IEEE/ACM Int'l Symp. Mixed and Augmented Reality* (ISMAR 06), IEEE CS, 2006, pp. 159–162.
10. J. Wither et al., "Fast Annotation and Modeling with a Single-Point Laser Range Finder," *Proc. 2008 7th IEEE/ACM Int'l Symp. Mixed and Augmented Reality*, IEEE Press, 2008, pp. 65–68.
11. S. DiVerdi, J. Wither, and T. Höllerer, "Envisor: Online Environment Map Construction for Mixed Reality," *Proc. IEEE Virtual Reality Conf. 2008*, IEEE Press, 2008, pp. 19–26.

We use a cylindrical mapping model, which doesn't suffer from discontinuities (as with cubic environment maps). When the mapping process starts, the first camera frame is projected into the map and serves as a starting point for tracking. We assume that the phone is held with zero pitch and roll during the first frame. For mobile phones with a linear accelerometer, roll and pitch can be automatically inferred to initialize the application. For subsequent camera

frames, projecting only those parts of the image that haven't yet been mapped preserves the compute cycles.

The system organizes the map into tiles, and it only considers a tile for tracking after the tile is completely filled with pixels. We used a run-length-encoded coverage mask to achieve pixel-accurate bookkeeping for the mapping. This lets us quickly sort out map pixels that don't require updating. As a result, every pixel of the map is written only once, and usually only around 1,000 pixels are mapped per frame, which guarantees high frame rates.

Panoramic Tracking

We track the camera orientation needed for the mapping process with an efficient and accurate method using the map as it's being built. We apply an active search procedure based on a constant-velocity motion model to track keypoints from one frame to the next. Keypoints in the map are compared against their counterparts in the camera image. We subdivide the map into 32×8 cells, and once we've completely mapped a cell, we extract its keypoints using a corner detector.

The tracking approach is generally drift-free, but errors in the mapping process still accumulate, so the map isn't 100-percent accurate. As a result, our method allows loop closing, which can minimize errors that accumulate over a 360-degree horizontal rotation.

The motion model provides a rough estimate for the camera orientation in the next camera frame, which the system then refines based on normalized cross-correlation (NCC) template matching. Based on the estimated orientation, the system projects keypoints from the map into the camera image and matches an 8×8 pixel-wide patch against the projected keypoints using NCC.

As long as tracking succeeds, we store the camera frames at quarter resolution together with their estimated pose. When tracking fails, we compare the current camera image against all stored keyframes and take the pose of the best match as the coarse guess to reinitialize the tracking process. In practice, tracking quickly restarts within 45 milliseconds (on an ASUS P565 phone) as soon as the user points the camera in a previously observed direction.

Annotation Detection and Tracking

In our previous work on panoramic mapping and tracking,[2] we saved the created map together with 2D map locations of annotations so that another user could reload the map and explore the annotations. Because we didn't store the keyframes together with the map, we used the PhonySIFT (scale-invariant feature transform) approach to register the loaded map with the camera images.[4] This required the user to be close to where the map was originally created—within 20 to 100 cm, depending on the distance of the object in the camera frame. If the standpoint deviated too much, PhonySIFT wouldn't always register the map or correctly align the annotations with the physical objects. This sensitivity to the standpoint, together with the elevated memory requirements for storing and transmitting a complete map, was a major limitation.

With our new method, users don't need to rely on previously created maps for tracking, because they can always build a new map on the fly. Instead of describing the annotations using a position in a previously created map, we store them in a self-descriptive way, suitable for robust redetection in a new map.

With our method, users don't need to rely on previously created maps for tracking, because they can always build a new map on the fly.

However, SIFT (or similar descriptors) isn't suitable for storing keypoints surrounding the annotation in the camera image. Although we have an efficient SIFT-based solution for tracking on mobile phones,[4] building a support search structure on the entire 2048×512-pixel panorama and maintaining it every time a cell gets updated is currently too slow to run in real time on a phone. Furthermore, because of the support area's size, SIFT can be problematic when matching small objects (those that are less than 50×50 pixels).

So, instead of matching points of interest against the camera image, we match them against the panoramic map. This lets us search in regions that have been seen but are no longer in the camera view. We can decouple object detection from the current camera view and run it in the background.

These special restrictions also make several of SIFT's features unnecessary. Because the map is always expected to be more or less upright, and because maps are recreated at similar locations, rotation and scale invariance aren't required. Instead, we identify label positions using NCC, which shares only the brightness and contrast invariance with the SIFT descriptor.

We describe a single annotation using nine templates in a 3×3 configuration (see Figure 3a). Each template is 16×16 pixels, because this configuration provides the best detection rate. Also, as opposed to one large template, small templates independently located in the map can better detect changes in scale and rotation. The 3×3 templates don't need to perfectly reproduce the arrangement in the original map—they just need to roughly form the original arrangement with a tolerance of five pixels in any direction (see Figure 3b). This makes the

Figure 3. The support area of an annotation is described (a) using a 3 × 3 grid of templates encoded using a Walsh transform. (b) The system can match the templates from a slightly different camera perspective.

annotation detection robust to small, nonuniform scaling, such as when an object is seen from a slightly different angle.

Compared to a complete map, which requires approximately one megabyte of storage, each annotation requires only around two kilobytes of storage. Furthermore, we can easily combine annotations from different users by loading all annotations created in a close proximity. Finally, detecting independent annotations is generally more robust to slight offsets in the user position than matching a complete map from a different location.

Walsh Transforms for Faster Template Matching

In a typical scenario, we must match dozens of annotations, described by image templates, against a map that's 2,048 × 512 pixels. Matching numerous templates against an image of this size is slow; we use Walsh transforms as a precheck[5] because they're fast to execute, and because using integral images makes the execution speed independent of the templates' size.[6] Matching multiple templates against the same image scales well, because the same Walsh transform of the image can be matched against an arbitrary number of transformed templates.

Integral images are memory intensive.[6] Furthermore, they're difficult to create for incomplete images such as the panoramic map, which is subject to change by successively adding new pixels to the image. Updating the map would require updating most of the integral image as well.

To solve this, we subdivide the map into tiles. We don't consider matching a tile against annotation templates until it's completely filled. We can then build an integral image for each tile with enough overlap to the right and bottom that we can place the templates at every possible pixel location inside that tile, performing a dense search. For each pixel location, we create eight Walsh transforms, which are then compared against the Walsh transforms of the annotations' templates.

Walsh transforms are fast to compute, but they only give the matching error's lower bound. So, for good matches, we also apply NCC. For each template, we keep the 10 best matching locations together with their NCC score. If at least four of the nine templates have been matched, we check if they form a 3 × 3 arrangement in the map (see Figure 3b). Our tests show that four out of nine provide a good balance between false positives and failed detections. Once this check succeeds, we mark the annotation as detected.

Real-Time Scheduling of Annotation Detection

Because the annotation templates are matched against the map instead of the camera image, we can schedule the matching to guarantee a desired frame rate. Rather than check each finished tile immediately, the system puts them into a queue. During each frame, the system schedules only as much work from the queue as allowed given the time budget. Because the operations are simple and their timings are predictable, we can easily limit the workload to remain within the budgeted amount of time.

Our system can thus run at constant speed on any phone that can perform real-time panoramic mapping and tracking. The annotation speed depends on the phone's processing speed. We benchmarked the detection on an ASUS P565 smartphone. Matching one cell against 12 annotations took approximately 54 ms. Targeting a frame rate of 20 Hz (50 ms per frame), the system can schedule approximately 10 ms for each frame detection.

Once the system has searched all available map cells for annotations, it can use any surplus compute time to search at different scales (for increased scale invariance) until the new map tiles are complete.

Browsing and Creating Annotations

We applied our technique in an AR browser application. In this application, the user initially sees the environment in an aerial map (see Figure 4a). This 2D map view shows the user's current GPS position and highlights nearby annotations. The user can employ the map to navigate the environment and find annotated spots. Once the user walks closer to an annotated spot, the application downloads the annotation data from a server. All annotations in immediate

proximity—as indicated by their GPS tags—are considered, so that inaccuracies in the GPS data don't affect the experience.

If a user decides to browse the annotations, he or she can switch to a first-person view (see Figure 4b) to see the current camera image. This automatically triggers the system to start the panoramic mapping and tracking. As the user rotates the phone to explore the environment, the application finds the correct position of the surrounding annotations as the best match of the stored template in the newly created panoramic map. Once the system successfully matches a template, it updates the view by displaying the annotation at the correct position. Furthermore, it updates the preview map by displaying the annotation's position in the miniaturized version of the panoramic image. This helps the user find the annotations from his or her current position.

The process for creating new annotations is similar to exploring annotations. The user moves to a position from which he or she wants to create an annotation. Switching to the first-person view prompts the application to start tracking the orientation and creates a panoramic image of the current environment. The user can now create annotations by simply touching the display at the desired position and entering a textual description or a voice annotation. A self-contained annotation is stored as a 48×48-pixel subimage centered on the chosen point in the panorama image. This subimage is later used for template matching. Besides the annotation itself, the subimage is the only information required for finding the annotation anchor point again.

To share annotations, users can upload them to a server-side database that uses standard Web software and protocols (Apache/Tomcat, MySQL). For better indexing, the system tags each annotation with the current GPS coordinate and user information before uploading it to the server. We file the submitted annotations according to a spatial index (the GPS coordinate), so the system can efficiently respond to queries for information near a particular standpoint. Information about the user's identity and optionally provided tags let us efficiently filter out many annotations.

(a) (b)

Figure 4. Our technique in an AR browser application. (a) A 2D map overview showing nearby annotations and (b) a first-person view of the annotated panorama.

Results

We used our application prototype for a first exploratory field trial to gain user feedback. We recruited eight users (three females and five males), aged 22 to 34, with no previous experience using AR.

We prepared two sets of six annotations for each user in an urban outdoor environment, with labeled objects being 10–200 meters from the user. In each set, we created two annotations from slightly different positions (5 m away). We made the first set the day before the user trials and the second set within 30 minutes of the user tests. So, users had to browse one set that didn't match the current environment conditions (there might be different lighting or shadows) and a second set that matched the current environment conditions but was different for each user. During the test, we asked the users to identify the labeled objects and label some new objects. After the trial, we used a semistructured interview to collect user feedback.

Usability

The test showed for the second sets, users detected 43 out of 48 annotations—that is, the success rate was 89.53 percent when the annotations were recorded under similar environment conditions. The detection rate was lower for annotations created under a different environment condition (27 out of 48; 56.25 percent). There were no false positive detections during any of the tests.

All users could annotate the given objects. Although the users found the display to be very small for clearly identifying objects, nobody perceived this as a real problem. Most of the users' gaze often switched from the display to the real environment for verification. To cope with the small-screen constraint, users proposed adjusting the size of annotation

To cope with the small-screen constraint, users proposed adjusting the size of annotation points and adding a video zoom function.

Figure 5. Appearances of points of interest in maps taken at different times during the day or from different locations. (a) A failed match, (b) object structures (visible as self-shadows) vanish once the object itself is in the shadow, (c) artifacts due to parallax caused by a different user location, and (d) the house matched throughout the entire day, because the major structures were always highly visible.

points and adding a video zoom function for annotating very small objects.

All users agreed that the tracking was stable and fast. They experienced occasional loss of tracking, which was signified by a question mark on the screen, but they consistently recovered quickly by pointing the camera toward a previously visited region. Six out of the eight users stated that as they became more familiar with the applications, they could avoid tracking problems. This was also noticeable as users progressed from a stiff posture to a more relaxed one over time. Users reported that they mostly broke the panorama-based orientation tracking by moving too fast or pointing the phone to the sky.

To detect the annotations, six out of the eight users felt they could improve detection by exploring the neighborhood of an annotation. The remaining users said that the label was at the correct position as soon as they looked toward that position through the camera. None of them noticed any drifting or jumping in the labels once detected.

Finally, the user interface generally received positive comments—especially the panorama preview function, which was employed by all but one user for orientation and to identify unexplored regions. Five of the eight users also took advantage of the preview to locate known positions for reinitializing the tracking. All of the users agreed that the browsing operation was easy and that the tracking was robust.

Matching

The results of the preliminary user test showed significant differences in matching quality. A further analysis showed that changing light conditions throughout the day caused most of the failed matches.

New or missing shadows can largely change the appearance of objects. Figure 5a shows an example of a failed match—in the morning, the wall behind the sign is half dark and half bright, whereas in the afternoon, the whole background has similar brightness and is shadowed by a tree. Figure 5b shows how object structures (visible as self-shadows) vanish once the object itself is in the shadow. Figure 5c shows artifacts due to parallax caused by a different user location. In contrast, the house in Figure 5d matched throughout the entire day, because the major structures were always highly visible.

Our current detection approach is optimized for no false positives and for speed. In the future, we plan to improve the redetection rate under environmental changes. We'll look into describing annotations as sets of data collected from multiple panoramas with various lighting conditions. It will be interesting to evaluate how many panoramic sources are required for robust results.

Furthermore, the availability of a compass and accelerometer will let us position annotations—albeit with reduced accuracy—even when our vision-based matching fails. Finally, sensor-based tracking, together with georeferenced annotations, will let us improve the template-matching process by narrowing down the search area within the panorama.

ACKNOWLEDGMENTS

This work was sponsored by the Christian Doppler Laboratory for Handheld Augmented Reality, the Austrian Science Fund FWF under contract W1209-N15, and the EU Integrated Project FP6-IST-27571.

REFERENCES

1. J.C. Spohrer, "Information in Places," *IBM Systems J.*, vol. 38, no. 4, 1999, pp. 602–628.

2. D. Wagner et al., "Real-Time Panoramic Mapping and Tracking on Mobile Phones," *Proc. IEEE Virtual Reality Conf.* (VR 2010), IEEE Press, 2010, pp. 211–218.

3. S. DiVerdi, J. Wither, and T. Höllerer, "Envisor: Online Environment Map Construction for Mixed Reality," *Proc. IEEE Virtual Reality Conf.* (VR 08), IEEE Press, 2008, pp. 19–26.

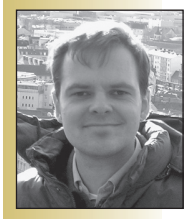
4. D. Wagner et al., "Pose Tracking from Natural Features on Mobile Phones,"

the AUTHORS

Tobias Langlotz is a PhD candidate at the Graz University of Technology. His research interests include handheld augmented reality with a focus on content creation for mobile AR applications. Langlotz received his Diploma degree in media systems from the Bauhaus University in Weimar. He's a student member of IEEE. Contact him at langlotz@icg.tugraz.at.

Daniel Wagner is a principal engineer at Qualcomm Austria Research Center. His research interests include handheld augmented reality, especially vision-based tracking on mobile devices. Wagner received his PhD from the Graz University of Technology. Contact him at daniel.wagner@qualcomm.com.

Alessandro Mulloni is a PhD candidate at the Graz University of Technology. His research interests include user-centric design of interaction and visualization methods. Mulloni received his MSc in computer science from the University of Udine. He's a student member of IEEE. Contact him at mulloni@icg.tugraz.at.



Dieter Schmalstieg is a full professor of virtual reality and computer graphics at the Graz University of Technology. His research interests include all aspects of virtual and augmented reality. Schmalstieg received his Dr. techn. and Habilitation degrees from Vienna University of Technology. He's a member of IEEE. Contact him at schmalstieg@icg.tugraz.at.

Proc. 7th IEEE/ACM Int'l Symp. Mixed and Augmented Reality, IEEE CS, 2008, pp. 125–134.

5. P. Nillius and J.O. Eklundh, *Fast Block Matching with Normalized Cross-Correlation Using Walsh Transforms*, tech. report ISRN KTH/NA/P-02/11-SE, Computational Vision and Active Perception Lab, 2002.
6. P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proc. 2001 IEEE Computer Society Conf. Computer Vision and Pattern Recognition* (CVPR 01), IEEE CS, 2001, pp. 511–518.