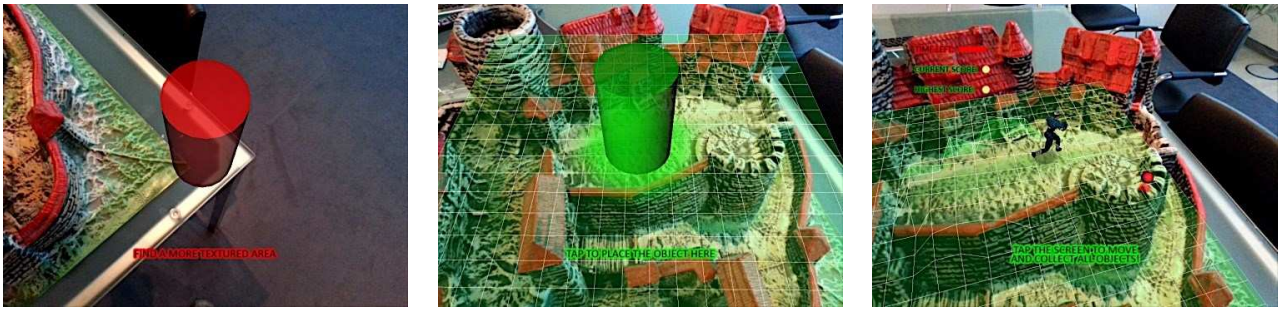
User Friendly SLAM Initialization

Alessandro Mulloni[1], Mahesh Ramachandran[2], Gerhard Reitmayr[1,3],
Daniel Wagner[1], Raphael Grasset[3], Serafin Diaz[2]

[1] Qualcomm Austria Research Center, Vienna, Austria
[2] Qualcomm Research Center, San Diego, CA
[2] Christian Doppler Laboratory for Handheld Augmented Reality, Graz, Austria



(a) (b) (c)

Figure 1: Our interface uses a novel tracking system to support user-friendly SLAM initialization. (a-b) When users scan the environment, our interface interactively communicates if a location is bad (a) or good (b) for SLAM initialization. Hidden from the user, our tracking system initializes a SLAM map as soon as possible. (c) When a user select a good location, an application can start instantly.

ABSTRACT

The development of new Simultaneous Localization and Mapping (SLAM) techniques is quickly advancing in research communities and rapidly transitioning into commercial products. Creating accurate and high-quality SLAM maps relies on a robust initialization process. However, the robustness and usability of SLAM initialization for end-users has often been disregarded. This paper presents and evaluates a novel tracking system for 6DOF pose tracking between a single keyframe and the current camera frame, without any prior scene knowledge. Our system is particularly suitable for SLAM initialization, since it allows 6DOF pose tracking in the intermediate frames before a wide-enough baseline between two keyframes has formed. We investigate how our tracking system can be used to interactively guide users in performing an optimal motion for SLAM initialization. However, our findings from a pilot study indicate that the need for such motion can be completely hidden from the user and outsourced to our tracking system. Results from a second user study show that letting our tracking system create a SLAM map as soon as possible is a viable and usable solution. Our work provides important insight for SLAM systems, showing how our novel tracking system can be integrated with a user interface to support fast, robust and user-friendly SLAM initialization.

Keywords: Augmented Reality, SLAM.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Artificial, augmented, and virtual realities

1 INTRODUCTION

Monocular Simultaneous Localization and Mapping (SLAM) has rapidly advanced from early PC-based research prototypes to commercial applications on mobile devices. We have also recently demonstrated a keyframe-based SLAM system that runs at 30Hz on off-the-shelf mobile devices. However, as SLAM technology is transferred from the research labs to the hands of untrained end-users, fundamental questions on its usability and its robustness arise.

This paper investigates the usability of the initialization process of keyframe-based monocular SLAM systems. Such systems typically require an initial baseline between two frames to initialize a map. Before such a baseline has formed, SLAM tracking is not possible, and the content and logic of the AR application cannot be triggered. Consequently, in most SLAM-based applications, users are asked to perform an appropriate motion to initialize SLAM but they do not receive any interactive feedback from the application during this process. Users can also fail in performing such motion, which often results in failure in the initialization of the tracker.

Since a good Augmented Reality (AR) experience depends on robust and transparent tracking integrated into the application, badly designed SLAM initialization has a negative impact not only on tracking quality but also on the overall user experience. In contrast, well-designed and user-friendly SLAM initialization should guide users through the process and allow them to quickly dive into the AR experience with minimal time and effort invested on initializing the tracker.

SLAM initialization consists of two tightly integrated components: first, a tracking system that initializes a SLAM map as soon as a sufficient baseline between two keyframes has formed; second, a user interface that interactively supports users in pointing their camera at an area with sufficient detail for the tracking system (e.g., feature points) and in performing a motion that provides sufficient baseline. In this paper, we provide novel contributions to both components, with a particular focus on how users can be best guided in providing a sufficient baseline for SLAM initialization.

We present a novel tracking system that allows 6-degree-of-freedom (6DOF) pose tracking between a single keyframe and the current camera frame. The method works by tracking 2D points from an initial keyframe and jointly optimizing the 3D point locations and relative camera pose for each successive frame, with

respect to the initial keyframe. Our method is not designed to be a replacement for SLAM but to complement it with tracking under single-keyframe conditions. The proposed method is therefore well suited for pose tracking from application startup till the SLAM system has successfully initialized its map.

We then discuss how our proposed tracking system can support interactive guidance of users during SLAM initialization. Since we can track the 6DOF motion of the user's camera, we look at how users can be guided to perform "good" motions to initialize the SLAM map robustly. Results from a first pilot study suggest that users tend to disregard visual guidance but they still provide sufficient baseline for SLAM through their natural motions. We therefore reconsider our design by hiding the need for a good motion from the users, and relying on our tracking system to automatically generate a SLAM map as soon as possible. The results from a user study show that users successfully initialize robust SLAM maps with such an interface design. Furthermore, users of our interface never fail to initialize SLAM after location selection - by design of the interface. Overall, insight from our user study indicates that our interface (Figure 1) can allow for robust and user-friendly SLAM initialization into any AR application based on SLAM. The contribution of this paper is two-fold: First, we present a novel tracking system that can estimate a 6DOF camera pose from a single keyframe without any prior scene knowledge. Second, we evaluate it and show it is suitable for a new SLAM initialization approach leading to more user friendly SLAM systems.

2 Related Work

There is a rich body of previous work on SLAM from a computer vision perspective. In the following, we give an overview of previous art on SLAM tracking. We then highlight the state of the art in providing user guidance for SLAM initialization.

2.1 SLAM Tracking and SLAM Initialization

Even before the advent of robust model-based tracking for AR, early research prototypes of SLAM tracked natural features using Kalman filters [4]. Filter based SLAM systems could run in real time, but due to processing constraints they could track only a limited number of features (a few dozens in practice). Klein's PTAM system [9] was the first SLAM system that decoupled tracking and mapping and could thereby use optimization methods (Bundle Adjustment [19]) as opposed to recursive filters for more accurate mapping and richer maps. At the same time Klein's system could track a much larger number of points resulting in significantly increased tracking robustness. Later, Strasdat et al [18] showed that increasing the number of features is more beneficial than increasing the number of frames and that a keyframe-based approach is hence preferable. Since the introduction of PTAM, almost all newly introduced SLAM systems are optimization-based rather than filter-based, including recent dense systems such as Newcombe's DTAM [13] or the system by Graber et al [7].

Monocular SLAM systems are dominating in AR because of the simpler system of a single camera. However, to obtain 3D estimations from a single camera, features such as points or lines need to be triangulated between two frames with known camera pose. The pose can be obtained by tracking an initial reference target, for example a black rectangle, or by estimating the relative motion using epipolar geometry between two frames. The former approach is not widely applicable, while the later process can be fragile due to multiple confounding factors.

To estimate the relative motion between to frames, a set of point correspondences or similar features need to be obtained. Thus the two frames must have significant overlap to enable the system to reliably establish correspondences. The camera motion between the two frames must also allow for accurate triangulation, which requires large enough triangulation angles between the two camera centers and the 3D points. The best motion is pivoting around a fixed point in the scene, so that the camera translates and produces a larger baseline angle while continuously observing the same features. In contrast, motions without translation (i.e., pure rotation) preclude any initialization of the scene.

PTAM and related systems address this issue by presenting a user with an explicit initialization step. At first, an initial keyframe is manually selected, while the system tracks correspondences using a variant of KLT and shows the correspondences as colored lines between the first and current video frame. Simultaneously, the system continuously estimates the epipolar geometry and triangulates the 3D points in the scene. If successful, the system is initialized with this set of 3D points, and tracking and mapping proceeds with this initial map. Other related approaches maximize some measure of the relative motion estimation quality [6].

SLAM systems that use RGBD cameras, stereo cameras or structured-light cameras do not have this limitation, because the 3D depth of features can be estimated directly in a single instant. Furthermore, due to the known baseline of stereo systems or the known depth values, scale of the 3D maps is also known.

2.2 Point estimation in monocular SLAM systems

The initialization problem is related to the question of how to estimate the 3D locations of observed features in the SLAM system. If the camera can be tracked through other means and pose is continuously available, a 3D location can be directly triangulated from observations in two suitable frames. However, in incremental filter-based systems the information is slowly accumulated over time. The later approach is closely related to our system, as we provide camera motion from the very first frame and continuously estimate the 3D map from there.

A classic approach is to represent the 3D features with an inverse-depth parameterization, either during an explicit initialization step [2] or throughout the whole filtering process [12]. Our system uses similar formulation to the depth-bundles described by Pietzsch [16], where only the inverse depth of each feature is estimated.

2.3 User Interfaces for SLAM Initialization

User interfaces for initializing SLAM tracking have been poorly explored in Augmented Reality. Existing methods in research typically rely on favorable operation by computer vision researchers or trained users, who know how to move the camera in order to successfully initialize a SLAM map. SLAM techniques such as DTAM or KinectFusion generally provide feedback during mapping and tracking through a simple exocentric view and an iterative reconstruction of the 3D scene. However, they do not give users any interactive feedback during initialization. PTAM provides trails of detected features points as visual feedback during motion, but such feedback is not intuitive for untrained users. These techniques are valid as far as the technology stays in the research labs, but more user-friendly guidance and feedback are necessary as soon as a SLAM-based system is in the hands of an untrained end-user.

Some previous research work considers the problem of guiding users through tracking problems, but not specifically for a SLAM tracking system. Jachnik et al. [8] use a radar view and color-coding to interactively guide users in capturing a hemispherical map over a planar marker. Users are not guided in their motions but they are informed on the areas that are still to be mapped. Coffin et al. [3] present the results of two user studies on visual guidance interfaces for pose recovery after tracking failures. Their

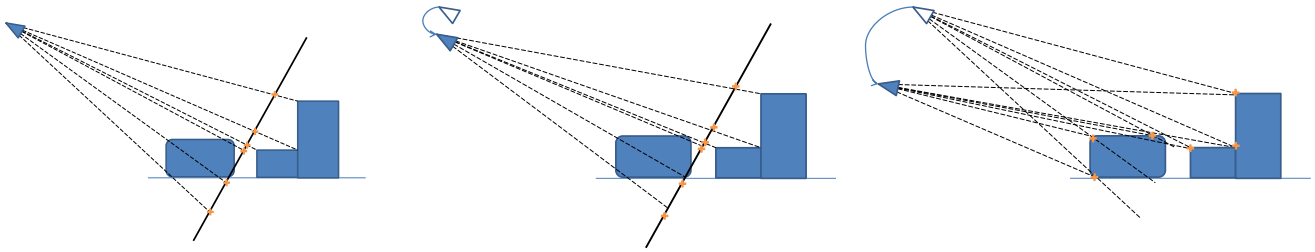


Figure 2: Our tracking method. Left: We start by assuming that all feature points have a depth of one unit. Middle: For small motions the scene is tracked as a plane. Right: For larger motions, we jointly update the depth of all points and the camera pose.

results are not directly applicable to our research, as they assume the availability of previously-recorded keyframes or of a model of the environment. Oda et al. [15] address the problem of space management and interference avoidance in a multi-user AR system via a technique called *redirected motion*. This technique cannot be generalized to our problem, because it does only offset the rendering of the virtual objects and it has no impact on the trajectory of the physical camera.

Several companies recently published SLAM-based products and hence explored different methods for improving the usability of SLAM initialization. All of them consider only sideward translational motions for the initialization. Metaio's InstantMapping [11] uses a pictorial 2D animation of the motion to execute. Dekko's SLAM system [5] asks users to overlay the edges of two 2D images, which forces users to execute a translational motion for the initialization. Minecraft [1] from 13th Lab provides only textual information of the motion to execute. In contrast to our work, all these systems do not provide interactive feedback or guidance during the process.

3 Tracking

Our tracking system is able to provide immediate 6DOF tracking without any prior knowledge on the scene, using image sequences from one single monocular camera. The standard method of calculating a relative pose with respect to a starting point is to apply triangulation methods, such as the 5-point [14] or 8-point [10] methods, which are commonly used in wide-baseline 3D reconstruction. However, these standard relative-pose methods rely on a minimal baseline for solving; cases of zero-motion or rotational-only-motion cannot be solved.

The method described in this section (see Figure 2) does not suffer from this weakness, since it is optimization-based: At each frame, it executes a "Mini Bundle Adjustment" step that jointly optimizes the point distribution as well as the relative camera pose between the current frame and the initial keyframe. Initially, all points detected in the first keyframe are set to have a distance of unity from the camera center. As the camera moves, the depth of all points and the relative pose are continuously updated: Points move forward and backwards along rays originating from the first camera center and passing through the respective observations in the first keyframe. Hence, we need to estimate only a single parameter per point, which makes the method efficient enough to run in real time on a mobile phone. For each camera frame, the triangulation angle of all features is calculated. When a feature passes a certain angle threshold, it is marked as robust. Once enough features are robust, the method automatically promotes the current camera frame to a keyframe and initializes the SLAM map using the two keyframes.

3.1 Tracking Method

The initialization starts with a camera frame I_0. The associated camera pose is $C_0 = I^{4x4}$, the identity transformation. 2D feature points $p_i = (u_i, v_i)$ are extracted from the camera frame. Each point is associated with an initial depth $z_i = 1$. The depth is generally stored and processed as inverse depth $w_i = 1/z_i$. The 3D location of the point X_i corresponding to the feature point p_i is then $X_i = (u_i, v_i, 1, w_i)^T$ in homogeneous coordinates.

For any subsequent camera frame I_t at time t, the positions of all features are first localized in the camera image: The points X_i are projected into the current frame I_t using the last known camera pose C_{t-1}. New measurements of the 2D location in frame I_t are made by searching around the predicted location to get a Normalized Cross Correlation (NCC) peak and for each point X_i a 2D location m_i is observed. From the 2D locations m_i and the corresponding 3D locations X_i, the camera position C_t is estimated. In a subsequent step, the 3D locations X_i are then also refined using the locations m_i.

The camera location C_t with pose (R_t, T_t) as well as the inverse depth w_i of each point is optimized using a Gauss-Newton non-linear refinement scheme on the following optimization problem

$$\hat{R}_t, \hat{T}_t, \hat{w}_i = argmin \sum_i \left(m_i - p_i(R_t, T_t, w_i)\right)^2 \quad (1)$$

where the projection function is

$$p = proj\left(\begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}_t \begin{pmatrix} u \\ v \\ 1 \\ w \end{pmatrix}\right) = proj\left(R\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + Tw\right). \quad (2)$$

The function $(u, v) = proj(X)$ is the usual camera projection operation mapping a 3D point X to the image co-ordinates (u, v), either through a perspective camera, or a camera model including radial distortion. The Jacobians with respect to R, T and w are:

$$J_{R,T} = \frac{\partial proj(X)}{\partial X} \frac{\partial \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}}{\partial (R,T)} \begin{pmatrix} u \\ v \\ 1 \\ w \end{pmatrix}, \quad (3)$$

$$and\ J_w = \frac{\partial proj(X)}{\partial X} T \quad (4)$$

When the camera translation T of C_{t-1} is too small, the Jacobian J_w of the observation m_i with respect to the inverse depth w_i of the 3D point degenerates to zero, which precludes an accurate estimation of the inverse depth. Therefore, during the Gauss-Newton iteration, we test the information matrix $J_w^T J_w$ of the inverse depth parameter for each point. If this value is too small, its inverse is set to 0 which avoids unreliably updating the depth coordinate. In practice, since the Jacobians J_w are not scale invariant, it is difficult to set an appropriate threshold to test if this value is small. Hence an equivalent measure is used, which is the triangulation angle subtended by the two camera centers at the 3D point location. The inverse depth (and 3D location) is updated only if this angle is above a threshold (e.g. 5 degrees).

At the same time, the camera pose can always be estimated, because the camera parameter part of the optimization is always

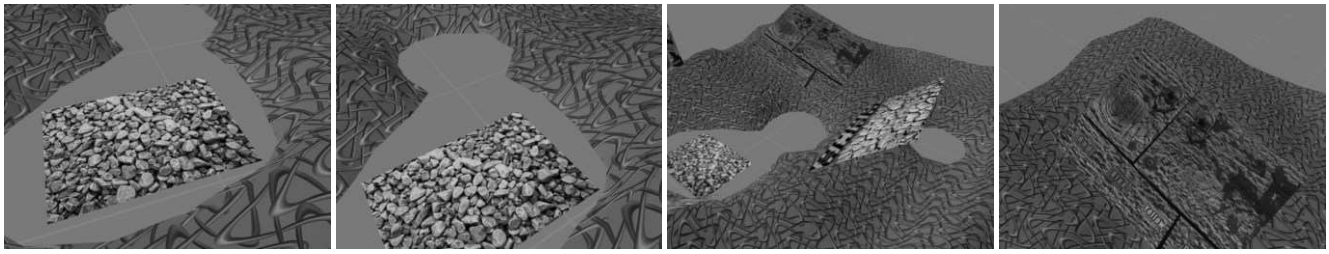


Figure 3: Example images from our ground truth testing sequence. Left image: frame 0 (sequence start); Second image: frame 13 (SLAM initialization successfully completes and the SLAM map is built); Third image: frame 226; Fourth image: frame 866.

well constrained: the points X_i have valid depth estimates at every frame. Optimizing and updating both camera pose parameters and the inverse depth coordinates w_i of all points results in a new camera pose C_t and new depth estimates for those points for which the Jacobians did not vanish.

This approach has the following desirable properties:

- Camera tracking is instantly possible from the first frame onwards, because points always have a known – assumed – depth value. For small motion, the scene can be approximated with a plane; therefore tracking errors are not noticeable.
- 3D point estimation happens as soon as enough translation is present in the camera motion.
- The 3D point estimation respects the epipolar constraint because it jointly estimates camera motion and all possible 3D points.

No special user interaction is necessary. The user can move the camera in any direction as long as the original scene stays visible. The system can automatically initialize the 3D map as long as part of the original scene is visible.

3.2 Pose verification and initialization

The tracking method presented in the previous section can converge to false structure estimates in the case of specific motions: When the epipole is close to the principal point, such as in the case of forward motion, there are multiple local minima for the camera motion that are very close to the global minimum corresponding to the true motion. In such cases, depending on the initialization, the optimization can converge to one of the local minima. These local minima and their visualizations are described in detail in [2].

Since the false local minima are close to the global optimum, tracking actually works well as long as the viewpoint doesn't change too much with respect to the first keyframe. Since the map is usually initialized with a baseline sufficiently small to not create problems due to wrong local minima, the tracking method does not suffer noticeably in this case. However, a SLAM map built from these false estimates would create problems for the SLAM tracker under subsequent large viewpoint changes. It is therefore important to verify and eventually correct the estimated structure.

To guard against these failure cases, we have implemented a pose verification step based on two-view epipolar geometry. We take the correspondences between the two frames and estimate the Essential matrix using eight-point algorithm in a RANSAC procedure. We solve for the relative pose from the Essential matrix by picking the solution (among the four possible ones) that satisfies cheirality. To handle the problematic case of planar scenes where the Essential matrix is ambiguous (determined only up to 3DOF), we additionally estimate a Homography in a RANSAC procedure (and decompose it) if the earlier step fails. The homography transformation is uniquely defined for planar scenes, and decomposing the homography results in a relative pose estimate between the two frames.

The relative pose obtained from the Essential matrix (or the Homography matrix) is then used to triangulate the points and eventually initialize the map. This also serves as an independent verification of the relative pose estimated by the tracking method at the end of the initialization stage.

3.3 Evaluation

We evaluated the performance of the 6DOF initialization tracker using ground truth data. We used a synthetic sequence that was produced using Autodesk Maya. We generated a 3D scene with textures and also synthesized a trajectory of a moving camera that looks at the scene. Color and Depth images were rendered at 640×480 pixels resolution from 1000 viewpoints along the trajectory (see Figure 3). This approach provided us with the ground truth poses for the camera motion as well as a true depth map for each image. Using this data, we can calculate the 3D coordinate of every pixel in every image of the sequence. In this section we present results on the accuracy of the camera pose and point structure estimation. Additionally we provide timings measured on a PC and mobile phone.

3.3.1 Pose Accuracy

Figure 4 shows a comparison of the translation and rotation parts of the estimated camera pose with respect to ground truth. For this test, we disabled the automatic map building that normally occurs in our system after a sufficient baseline has formed and a stable initial map has been created. Instead of extending the SLAM map, the tracker continues to estimate the camera pose and the 3D feature locations. Eventually the tracking fails because the feature count drops below a minimum threshold, due to feature track loss and no new features getting added. When the tracking fails, we restart the tracking from the next frame. The pose estimated by the tracker (which consists of rotation and translation vectors) is compared with ground truth. Since the tracking uses the coordinate system of the first keyframe as reference, which is in general different from the reference coordinate system of ground truth, we first bring the ground truth poses into the reference coordinate system of the tracker. Additionally, the scaling factor for the camera translations and 3D point locations changes continuously for each frame. To facilitate a comparison with ground truth, we compensate for the scale factor by scaling the estimated camera translation vector at each frame to have the same length as that of the ground truth.

In Figure 4, when the tracking starts, the reference frame is re-initialized and hence the camera pose is known very well with respect to the first keyframe. As the camera moves, there is some drift due to the initial planar assumption of scene points and because the point locations have not converged. As the camera moves further to create sufficient baseline it allows for reliable point triangulations, and the poses are estimated well and the error with respect to ground truth goes down. Beyond a certain

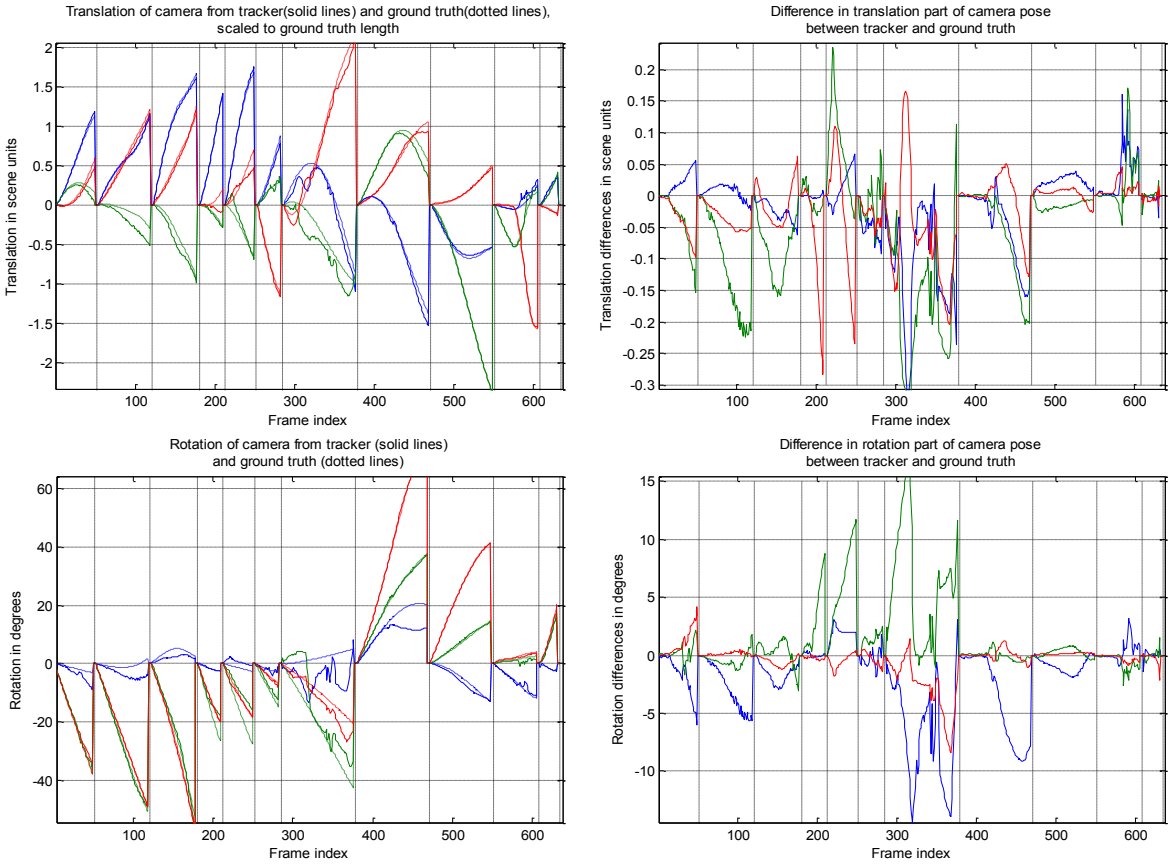


Figure 4: Accuracy of our tracking system. Left: translation (top) and rotation (bottom) parts of camera pose estimated by tracker (solid lines) and ground truth (dashed lines). Right: difference between ground truth and pose estimated by tracker. Translation (top) and rotation (bottom).

baseline, the pose accuracy falls due to loss of features and error in the point tracks. This happens because tracking accuracy suffers under large view point changes, since the feature tracking part of the tracker has not been designed for this purpose. The feature point localization in our tracker is optimized to track efficiently only over small viewpoint changes. The pose error in the graphs increases when the camera moves because of the error in feature localization (image pixel noise) due to distortions in appearance warping caused by inaccurate patch normal. It is not caused by the pose computation step through the mini-BA. It is important to consider that under normal circumstances, for this specific test sequence, the SLAM system creates a map at approximately 17 frames after the first keyframe, which results in more stable tracking from that point onwards. When the camera tracking is lost due to feature loss, the tracker is re-initialized again, which is why the pose changes to the identity transformation again. This explains why the pose values of the camera drop to zero in the figure, whenever the tracker is reinitialized. Figure 4 (right) plots the difference between the ground truth and the estimated pose. When the tracker succeeds, the pose error is approximately within 5 degrees and 3% of the extent of translational camera motion. In a few trials the tracker pose was very inaccurate due to poor feature tracking. In all the plots of Figure 4 the places when the tracker is reinitialized is indicated by dotted vertical grid lines.

3.3.2 Point Structure Accuracy

We also evaluated the accuracy of the point structure that is jointly estimated by the tracker together with the camera pose. We calculated the true 3D coordinate of every feature point selected by the tracker using the ground truth information of the test sequence. For every frame, we fitted the estimated scale to best match the scale of the ground truth. For this, we calculated the ratio of estimated point depth to ground truth point depth and took the median of all measurements. The left graph in Figure 5 shows how point depth accuracy changes over a sequence of 22 frames (horizontal axis): Each single point is represented by a different colored line. The vertical axis represents the relative accuracy in percentage, as deviation of the estimated distance from the true distance relative to true distance. The graph is capped to +/- 20% in order to highlight details for inlier points. Same as in 3.3.1 we configured the tracker to run longer than usual by using stronger convergence criteria: While under normal operation the map would have been initialized in frame 13 tracking continued here until frame 22.

At frame 0 all points are initialized with the same depth of one scene unit; hence most points are very inaccurate. As the camera moves most points quickly converge towards true depth. After three frames, most points are accurate within 1-2%. While most points remain in this range a few points soon become outliers (drifting far away in frame 3) and others become less accurate until they converge again in frame 18. The black dashed line shows the average of absolute error of robust points (here points are selected as robust if their absolute error is smaller than 3× the average of absolute errors of all points), which stays well below 2% from frame 3 onwards. In AR, the absolute pose and structure accuracy is often not as important as the reprojection accuracy. Hence, we also conducted a screen-space error evaluation shown in the right half of Figure 5. Points have medium accuracy at first (average error: 2.8 pixels in frame 1). Yet, as the camera moves points quickly become more accurate (average errors: 1.7, 1.9 and

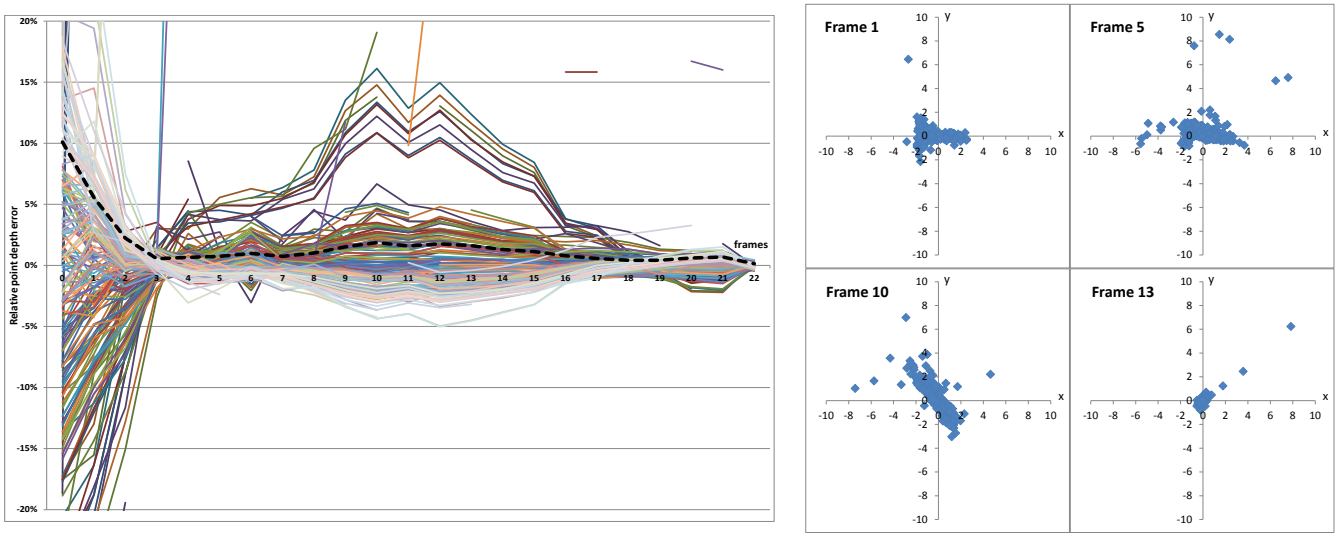


Figure 5: Left: Accuracy of point depth estimates on a sequence of 22 frames; Right: Projection accuracy in pixels in frames 1, 5, 10 and 13.

0.3 pixels in frames 5, 10 and 13) until the map is initialized in frame 13.

3.3.3 Processing Time

We measured the execution times of each part of our tracking system on a mobile phone as well as on a PC. We ran our benchmarks on a Samsung Galaxy S3 with a Qualcomm Snapdragon S4 CPU running at 1.5Ghz. Even though this CPU has two cores, the tracker runs only on one core. On the PC we benchmarked on an HP 8460p notebook with a Core i5-2540M CPU running at 2.6GHz. Feature tracking uses the FastCV library [17] which uses SIMD acceleration on the phone, but not on the PC.

Table 1 shows timings for PC and mobile phone. In the first frame, keypoint detection extracts 440 keypoints from the camera image. In the successive frames features are first tracked to update their current location in the camera image. Then Bundle Adjustment optimizes the camera pose and the depth of all features with respect to the first frame. Next, we run a check to verify if enough features have been triangulated with a minimum baseline angle between the first frame and the current camera image. Since the angles are already available from Bundle Adjustment this step takes less time than can be measured reliably and is therefore not listed in Table 1. If this check succeeds we run the pose verification step based on a robust Essential matrix check and, if necessary, also a Homography check.

Table 1. Timings on PC and mobile phone.

Task	Time on PC	Time on phone
Key point detection*	3ms	6.5ms
Feature tracking	5-7ms	8-10ms
Bundle Adjustment	1-1.5ms	3-4ms
Essential matrix check*	1ms	4ms
Homography check*	0.5ms	1.5ms

* These steps do not run at every frame

Table 1 shows that the performance of our tracking system allows a stable tracking at 30Hz on PC and mobile phone. Tracking and Bundle Adjustment take only ~9ms on the PC and ~15ms on the mobile phone. The final verification step adds another 1.5ms on the PC and 5.5ms on the mobile phone. This leaves sufficient time to build a SLAM map once the baseline is large enough, without any noticeable delay to the user.

4 User Interface #1: Interactive Feedback

As we discussed in the previous sections, the process of initializing a SLAM system requires users to capture two frames with a sufficient baseline. A dedicated user interface can be used to improving users' performance of such process. Since our tracking system can instantly track user movements with six degrees of freedom, *before* the SLAM system is initialized, we can incorporate this aspect in the design of our user interface.

4.1 Interface Requirements

The design of a user interface for user-friendly SLAM initialization requires:

- Choosing a motion that is suitable for both users and tracker.
- Guiding the user in correctly performing such motion.
- Providing interactive feedback to the user on his current performance in executing the motion as well as how to correct an erroneous execution.

Furthermore, for a wider applicability, the design of such a user interface should be minimally obtrusive for end-applications, and easily integrated with an application's look-and-feel and logic.

Motion. Choosing a good motion for user-friendly SLAM initialization depends on two potentially contrasting requirements. A first requirement is to be *good for the tracker*, in the sense that enough feature points should be visible by the device camera from the start to the end of the motion, in order to be tracked; the amount of translation between start and end of the motion should also be sufficiently large to form a good initial baseline. A second requirement are standard interface-design principles: the motion should be *easy to explain*, *easy to perform* (e.g., consider physical fatigue and manual dexterity) and *easy to learn* for a user.

Guidance. Instructions must be presented through the mobile device held by the user, ideally providing continuous feedback on how the user should move the device and visualization of how he has moved it since the start of the process. Different modalities can be used in this context, such as visual, audio or tactile (vibrotactile actuators). Visual guidance requires users to simultaneously look at the screen while physically moving the screen itself; this needs coordination but puts instructions and motion in a single unified coordinate frame, which reduces the mental mapping the user must perform.

Feedback. Interactive feedback should provide information about temporal and spatial aspects (e.g., start, end, current progress status). Feedback should also inform about the quality of the execution of the motion by the user, for example in textual or in pictorial form.

4.2 Interface Design

An ideal motion for the tracker allows continuous tracking of all feature points from start to end, while producing a large baseline between start and end positions. One best case is rotating the device around a pivot (Figure 6, left): during the whole motion all feature points remain continuously in view and can be tracked, and a large baseline is typically reached by the end of the motion. However, the motion is difficult to explain to users and difficult to perform – particularly while sitting – as it requires much physical movement. In contrast, we chose a sideward translational motion (Figure 6, right), also used in other SLAM systems [1][5][11] because it is easy to explain to users while still reasonably good for the tracker: provided that the necessary baseline is not too large, most of the initial feature points will still be in view by the end of the motion.

Figure 6. Two possible good motions for SLAM initialization: (left) rotation around a pivot and (right) sideward translation. While the first is an optimal case for the tracker, it is complicate to explain and to perform. The second motion is simpler for a user but still it provides reasonable information to the tracker.

We designed a user interface (Figure 7) to interactively support users in performing this translational motion. We set two key design goals for our interface:

- Interactive visualization of *how well the user is performing the motion* as well as *how to correct a wrong motion*. It is necessary for users to understand not only that they are doing an erroneous motion but also how to correct their motion.
- Unobtrusive interface with *minimal impact on an application's AR experience*. It is important to design an interface that can be easily integrated with any AR experience. It is also important for the interface to be as unobtrusive as possible to prevent slowing down users who are already proficient in initializing SLAM.

We considered various interface designs, ranging from minimal 2D iconic representations to full 3D visualizations. We ultimately chose a 2.5D representation as the best compromise between simplicity for end users and the amount of information shown.

We considered 2D designs based on iconic representations, to visually inform users about the quality of their motion. While this type of interface design is very unobtrusive and easily integrated within an AR experience, we found it difficult to visualize in 2D how a wrong 3D motion can be corrected.

We also considered 3D designs rendering a trace of the user's motion in 3D. However, such visualization becomes complex and hard to interpret, due to perspective effects. We explored this 3D design to the point of designing a mini-game that enforces the motion, for example asking the user to collect different virtual items positioned on a pre-designed trajectory. However, we found that this is very obtrusive for an AR application, as it breaks the application logic with an external mini-game. Overall, we reflect that the users' goal is to be engaged with the AR application, rather than with an initialization mini-game; this design space is therefore not practicable for end-applications.

Ultimately, we chose a 2.5D visualization in which we only visualize sideward (horizontal) translation and full rotation. We considered that the key goal of our interface is to show users positive feedback on translational motions and to clearly highlight an erroneous rotational motion. The final interface design is shown in Figure 7. The interface shows the initial position of the device (gray object) and the current position of the device (green or red object, depending on how well the user is executing the motion). An animated green arrow indicates the direction the user should move the device. The orientation of the object in the visualization interactively matches the device's movements. This design is minimally obtrusive for the AR experience, while communicating full 3DOF warnings in case of rotational motions and interactively reacting to users' corrections to their motion. The cylindrical object in Figure 7 is only meant to illustrate the functioning of our interface, and can be replaced by any other 3D object, e.g. a 3D character used in an end-application.

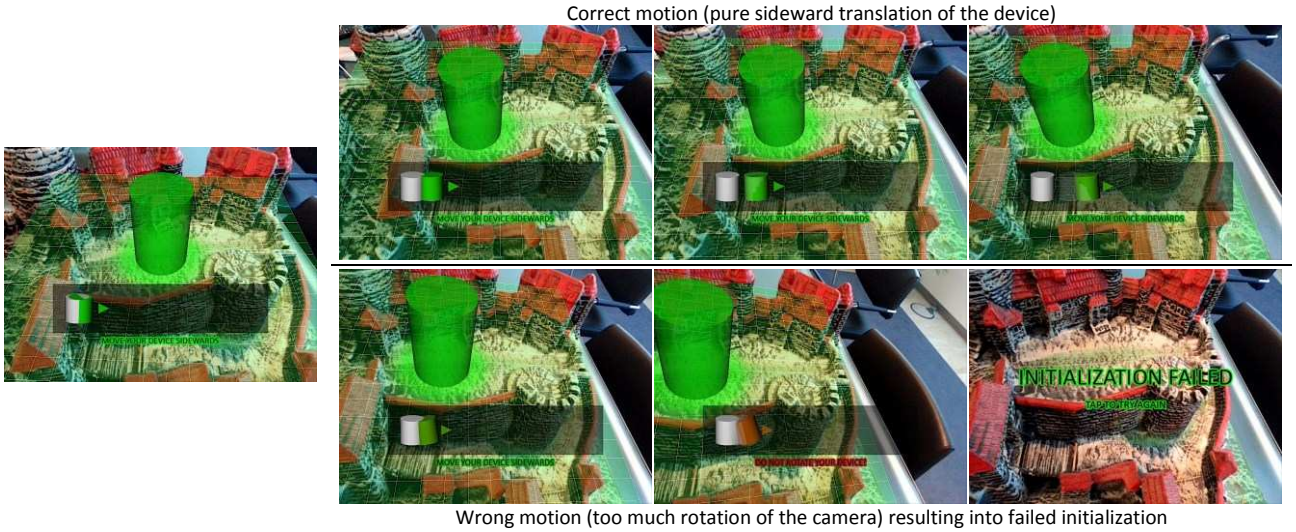


Figure 7. Screenshots of our user interface for guidance during SLAM initialization. Top: the interface when the user is performing a correct motion. Bottom: the interface when the user is erroneously rotating the camera instead of translating it.

4.3 User Study

We conducted a first pilot study of our interface design on the following research question: *does interactive feedback enhance user performance during SLAM initialization*? As a measure of performance, we looked at *successful SLAM initialization*, *user experience* with the initialization process, as well as *user awareness* of how to correctly initialize. These are all fundamental aspects for considering a technology user-friendly.

4.3.1 Study Design

We conducted the pilot within-subjects to comparatively evaluate our interface against a non-interactive interface that shows static instructions on how to move the device (Figure 8, left) – a user interface seen in other SLAM systems [11]. The pilot study had two conditions:

- *Static guidance*: in this condition, the system only shows visual instructions, which explain how to move the device. This type of interface is common in other SLAM systems. The interface for this condition is shown in Figure 8, left.
- *Interactive guidance*: in this condition, the system exploits our tracking system to provide an interactive visualization of the device motion. We also provide interactive feedback on the goodness of the motion. For this condition, we used the user interface described in the previous section (Figure 7).

Study participants were given the task to initialize SLAM tracking and then play a mini-game for 20 seconds. Before SLAM initialization could start, we used a *scanning* interface (see Figure 1, a-b): users were asked to point their device at different locations in the environment, and they were only allowed to initialize the game when at least 75 feature points were in view (in order to prevent initialization at a bad location). The game consisted of controlling the movements of a virtual character and collecting as many items as possible (Figure 1, c). Current and best score were always shown on screen, to increase user engagement with the game. This task was designed so that we could observe user performance initializing SLAM while putting the users' focus on the mini-game and not on the initialization process. We believe this was necessary to observe realistic user behavior when using an end-application based on SLAM.

Figure 8. Left: User interface for the *static guidance* condition. Such type of user interface is common in most SLAM-based applications. Right: setup used for our two user studies.

Four participants (mean age 21, all male) were recruited for the pilot study. Only one participant was familiar with AR, all participants had touch-screen experience. As shown in Figure 8 (right) we intentionally chose a sufficiently textured table for the experiment. We started the experiment by explaining participants that we were testing two different technologies for mobile gaming (called "Technology A" and "Technology B"). We then asked participants to play the game based on one of the two technologies (the order of conditions was balanced between participants). We intentionally did not have any training phase, as we wanted to look at performance in a realistic case of an end-user operating a SLAM-based application (e.g., downloaded from an app store) for the first time. Participants used Technology A for a total of 6 repetitions, initializing SLAM and then – in case of a successful initialization – playing the mini-game for 20 seconds. After this condition, participants then played with Technology B for 6 times, same as for the previous condition. During all trials, we logged software usage on the device and observed user behavior. After participants tried both conditions, we collected any further spontaneous feedback on the usability of the system.

For this experiment we hypothesized to observe higher success rates with interactive guidance than with static guidance. We also hypothesized to observe a better user experience (e.g., less usability complaints, less frustration) and higher user awareness with interactive guidance. Overall, we hypothesized that interactive guidance would not only improve success rate but also provide a better user experience.

For the experiment we implemented an interactive prototype running at 30fps on a Samsung Galaxy S3 with a dual-core Qualcomm Snapdragon S4 1.5 GHz processor. For pose tracking the interactive prototype used the tracking system presented in this paper in conjunction with our keyframe-based SLAM system.

4.3.2 Study Results

In contrast to what we hypothesized, we could not observe differences between the two conditions. Participants performed and behaved very similarly in both conditions.

Overall, participants initialized SLAM successfully in average half of the times with both conditions (mean 46% success rate for static guidance, mean 49% success rate for interactive guidance). We also did not observe any learning effect with either interface: all participants made several successful initialization attempts in the first repetitions as well as several failures in the last repetitions. We could not observe usability issues with the scanning interface.

Given the small sample we looked at the success rate only qualitatively and triangulated it with in-study observations. In both conditions, we observed that participants mainly did not perform the correct motion, but rather stood still and rotated their device sideward. The motion was sometimes too fast and broke tracking at first attempts, but participants seemed to adapt to this and slowed down on subsequent trials. Since the rotation occurred with arms partially stretched, the translation induced by this posture was often sufficient for SLAM to initialize successfully.

Overall, we observed that participants paid little attention to the interface elements for user guidance and rather tried to get past the initialization as soon as possible to engage with the game. When prompted for comparative feedback after the experiment, two of the participants admitted that they did not remember the exact difference between the two conditions they tried. Participants also did not follow the instructions on the screen correctly, but due to their posture and the robustness of our tracking systems they could still successfully initialize SLAM about half of the times.

The results of our first pilot study point against the usage of an interactive widget and in favor of making initialization completely invisible to the user. Our first observations also suggest that typical user movements with the device can be sufficient for initializing SLAM with our tracking system. We brought both these observations into our next iteration of interface design.

5 User Interface #2: Hiding Initialization

The results from the first pilot study suggest that widgets to interactively guide initialization might be disregarded by users, who are rather seeking to engage with the actual game as soon as possible. Furthermore, natural user gestures with the device might

be sufficient for our tracking system to successfully initialize SLAM. We therefore designed a second user interface, which completely hides the initialization process to users, and rather relies on natural user gestures and our proposed tracking system to seamlessly initialize SLAM.

5.1 Interface Requirements and Design

We set two design goals for the next design iteration:

- *Hiding the initialization process* from users, giving the impression that the transition from start screen to game is fast and seamless.
- *Guaranteeing a successful SLAM initialization* to allow for a subsequent robust AR experience.

We kept the scanning interface (Figure 1, a-b) in this next design iteration, as we could not observe usability issues with it in the first pilot study. However, we modified the criteria for allowing users to start the game. In this next design iteration, we continuously run our tracking system in the background when the scanning interface is running. We always create a SLAM map as soon as possible, and reinitialize the tracking system from scratch whenever SLAM tracking is lost. By this approach, we rely on natural user gestures during scanning to be sufficient for SLAM initialization. In the scanning interface, we only allow users to proceed to the game once we have a valid SLAM map.

This design satisfies both our design goals, as it makes initialization completely invisible (by hiding it behind the scanning interface) and only allows a game to start when a valid SLAM map is successfully created.

5.2 User Study

We conducted a comparative study of this new interface design against the *static guidance* condition of the previous pilot study. Our overall goal was to evaluate whether hiding initialization decreases the quality of the user experience with the application, since it causes a slow-down in accessing the game (users cannot get past the scanning interface until a SLAM map is initialized). We also compared the quality of SLAM tracking with the initialized map, since the new interface design does not instruct users in doing an optimal motion for the tracker.

5.2.1 Study Design

The design of this experiment was very similar to the previous pilot study. We had two conditions for this experiment:

- *Static guidance*: this condition is the same as in the previous experiment, and represents a state-of-the-art solution for most other SLAM systems.
- *Hidden initialization*: this condition is the new iteration of our interface design. SLAM initialization is hidden in the scanning interface. The interface still asks users to look for an area with more texture, when SLAM is not initialized.

Study participants were given the same task as in the previous pilot study. With static guidance, the user interface was the same as in the previous study. With hidden initialization, the scanning interface was changed to allow starting the game only once a SLAM map was successfully initialized in the background.

10 participants (mean age 27, SD = 6, 6 male) were recruited. 4 of the participants had no previous experience with AR. All participants (but one) regularly use a touch-screen phone. AR expertise and order of conditions were counterbalanced. There was no repetition of participants from the previous study.

The procedure for this experiment was the same as in the previous study. The two conditions were again introduced to the participants as "Technology A" and "Technology B", and participants played 6 20-second games with Technology A and then another 6 games with Technology B. After the experiment we asked participants to rate the pleasantness of their experience with the technology, their perceived robustness of the technology and their awareness of how to initialize the game on a 5-point Likert scale (where 1 mapped to "totally disagree" and 5 mapped to "totally agree"). We again logged information on system usage and tracking quality on the device.

Based on findings from the previous pilot study, for this experiment we hypothesized to measure an *overall better user experience with hidden initialization* (higher ratings for pleasantness, robustness and awareness). This is because we reflected that the participants would not perceive the hidden initialization procedure; they would therefore have a better experience by being able to jump directly from the scanning interface to the game. We also hypothesized to see *no difference in tracking quality between conditions*, because we believed users would perform similar motions in both conditions – as observed in the pilot study, participants tend not to perform correctly a sideward translation motion.

5.2.2 Study Results

For each condition and participant, we considered the tracking quality (reprojection error) and success rate (ratio of frames successfully tracked) after SLAM was initialized. For our analysis we used the medians of all measurements from the six repetitions.

Our results show that *the difference in tracking quality between conditions is not statistically significant*. In both conditions, we had comparable tracking quality: with static guidance there was a mean reprojection error of 0.9 pixels (SD = 0.4) and with hidden initialization a mean reprojection error of 0.9 pixels (SD = 0.2). A repeated-measure ANOVA shows no that the difference between conditions is not statistically significant ($F(1, 9) = .28$, $p = .61$, $\eta p^2 = 0.030$). The ratio of successfully tracked frames was slightly higher with static guidance (median 98%, mean 96%, SD = 5%) than with hidden initialization (median 96%, mean 90%, SD = 13%). Due to violation of the normality assumption we ran a Wilcoxon Signed-Rank test, which shows that the difference between the two conditions is not statistically significant ($Z = -1.58$, $p = .11$). Initialization with static guidance failed in average 20% of the times (SD = 19%), while it never failed (by design) with hidden initialization. The performance of users with no AR experience was comparable to that of users with AR experience, but we can only consider this qualitatively due to the small sample size.

Despite being comparably robust, hidden initialization was rated significantly less pleasant to use than static guidance (static guidance: mean 4.4, SD = 0.7; hidden initialization: mean 3.5, SD = 1.2) and less robust than static guidance (static guidance: mean 3.9, SD = 0.7; hidden initialization: mean 2.9, SD = 0.8). Wilcoxon Signed-Rank tests show that both differences are statistically significant (pleasant: $Z = -2.27$, $p = .02$; robust: $Z = -2.07$, $p = .04$). In both conditions, participants felt they knew equally well what they had to do (static guidance: mean 4.1, SD = 1.0; hidden initialization: mean 4.4, SD = 0.9). These results are in contrast with our initial hypothesis, as they show that participants found static guidance to be more pleasant to use and more robust, even though both conditions had a comparable tracking quality from our quantitative analysis of software logs.

In order to explain the lower subjective ratings for hidden initialization, we analyzed the subjective feedback collected through interviews after the experiment. Two usability issues are recurrent in the feedback we received: according to participants, hidden initialization exhibited *slower initialization* and *less robust tracking*. Four participants reported that with hidden initialization it was "more difficult to find a starting position", whereas static

initialization was "quicker in showing a green screen". This is definitely true, because of the design of hidden initialization. However, the time necessary for initializing the game was comparable in both conditions: mean 4.2 seconds (SD = 2.0) for static guidance and 4.8 seconds (SD = 1.7) for hidden initialization. A repeated-measures ANOVA test shows that the difference is not significant ($F(1, 9) = .41$, $p = .54$, $\eta p^2 = 0.044$). With static guidance, however, initialization was split into two separate phases: a scanning phase (mean 2.6 seconds, SD = 1.5) and a subsequent initialization motion (mean 1.7 seconds, SD = 0.9). The green screen was shown to users after the scanning phase, indeed in average 2 seconds earlier with static guidance than with hidden initialization, even though the time necessary for starting the actual game was comparable in the two conditions. One participant also reported that it was not clear how to start the game with hidden initialization, which suggests that re-introducing some form of interactive feedback into the hidden initialization interface might resolve these issues with the subjective experience with the interface.

With respect to participants' comments on tracking robustness, a Pearson's correlation test shows that the correlation between tracking success rates and subjective ratings is not statistically significant (pleasant: $r = -.056$, $p = .88$; robust: $r = .075$, $p = .84$). On the other hand, success rates of SLAM tracking were indeed slightly lower with hidden initialization than with static guidance (even though such difference was not significant). Overall, we believe that tracking robustness could be generally improved by finer tuning of the criteria for handover from initialization to SLAM tracking (e.g., baseline, number of tracked points).

6 Discussion and conclusion

In this paper, we discussed how SLAM initialization is made more user friendly by combining novel tracking technology with user-centered interface design. The main insight is that tracking can operate with an approximate map and correct it on the fly during operation. By exploiting this in a simple and fast optimization the initialization system can provide AR feedback to the user that appears correct and convincing. The standard SLAM operation is not affected by any drift or bias in the initialization tracking, because we use a separate pose verification step based only on the 2D-2D correspondences, but not on the 3D estimates.

The evaluation of our tracking system shows that the system does operate well under most conditions. While it does not provide the most accurate estimation possible, it is more than accurate enough to initialize a SLAM map after a short period of tracking. Overall, our proposed tracking system can track robustly and accurately during the short initialization phase of a SLAM-based application, before a sufficient baseline is formed.

We investigated how such novel tracking system can guide users interactively in performing a good motion for SLAM initialization. Our first results show that users tend not to follow the interactive instructions, but they still perform natural motions that form a sufficient baseline for SLAM initialization. This suggests completely hiding the need for an initialization motion in the user interface, and relying on our tracking system to seamlessly initialize a SLAM map as soon as possible. Our results show that this interface design results in successful SLAM initialization with quality comparable to a system that guides users in performing a motion for initialization. While hiding initialization is a successful design choice, user feedback also suggests that some interactive feedback should be still provided to explain users the current state of the initialization.

As an immediate next step for our research we are looking at a further design iteration of our user interface, in which we will refine our latest design to provide more interactive feedback on the state of the system. In particular, the scanning interface should be extended to provide smoother transitions and more information levels between bad-tracking areas (Figure 1, a) and good-tracking areas (Figure 1, b). As the interface design becomes more consolidated and refined, we are looking at conducting larger-scale usability experiments with a larger number of participants, for example using an app-store user study.

Overall, the results from this research have a deep impact on any AR system based on SLAM tracking. At the beginning of this paper, we put under discussion the usability of SLAM systems – in particular the users' ability to initialize a SLAM tracker with the currently available technology and interface designs. Our results significantly advance the insight on how SLAM initialization can be made more user friendly, showing that a tight combination of tracking technology and interface design can result into seamless, but still robust, SLAM initialization. Furthermore, our results show that a usable solution for SLAM initialization can be done with an unobtrusive and subtle design, which can be therefore easily integrated into any AR application that might want to use SLAM.

Acknowledgments. This work was sponsored by the Christian Doppler Laboratory for Handheld Augmented Reality and the Austrian Science Fund (FWF) under contract P-24021.

References

[1] 13th Lab, Minecraft, April 2013. http://13thlab.com

[2] A. Chiuso, R. Brockett, S. Soatto, "Optimal Structure from Motion: Local Ambiguities and Global Estimates", International Journal of Computer Vision, 39(3), 2000.

[3] Coffin, C. Lee, T. Höllerer, "Evaluating the Impact of Recovery Density on Augmented Reality Tracking", Proceedings of ISMAR 2011.

[4] A. J. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera", Proceedings of ICCV 2003.

[5] Dekko. Dekko Monkey, April 2013. http://www.dekko.co/

[6] T. S. Y. Gan and T. W. Drummond, "Vision-Based Augmented Reality Visual Guidance with Keyframes", Proceedings of CGI 2006.

[7] G. Graber, T. Pock, H. Bischof, "Online 3D reconstruction using Convex Optimization", Proceedings of ICCV 2011.

[8] J. Jachnik, R. A. Newcombe, A. J. Davison, "Real-time surface light-field capture for augmentation of planar specular surfaces," Proceedings of ISMAR 2012.

[9] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces", Proceedings of ISMAR 2007.

[10] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections", Nature 293, September 1981.

[11] Metaio, Instant Mapping, April 2013. http://dev.metaio.com/toolbox/instant-mapping/

[12] J. M. M. Montiel and J. Civera and A. J. Davison, "Unified Inverse Depth Parametrization for Monocular SLAM", Robotics: Science and Systems, July, 2006.

[13] R. A. Newcombe, S. Lovegrove. A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-Time", Proceedings of ICCV 2011.

[14] D. Nistér, "An efficient solution to the five-point relative pose problem", Proceedings of CVPR 2003.

[15] O. Oda and S. Feiner, "Interference avoidance in multi-user hand-held augmented reality", Proceedings of ISMAR 2009.

[16] T. Pietzsch, "Efficient Feature Parameterisation for Visual SLAM Using Inverse Depth Bundles", Proceedings of BMVC 2008.

[17] Qualcomm FastCV library, April 2013, https://developer.qualcomm.com/mobile-development/mobile-technologies/computer-vision-fastcv

[18] H. Strasdat, J. M. M. Montiel and A. J. Davison , "Real-Time Monocular SLAM - Why Filter", Proceedings of ICRA 2010.

[19] B. Triggs, P. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon, "Bundle Adjustment - A Modern Synthesis", Proceedings of ICCV 1999.