The Visual Computer
https://doi.org/10.1007/s00371-023-02777-2

ORIGINAL ARTICLE

Localization and tracking of stationary users for augmented reality

Lewis Baker[1] · Jonathan Ventura[2] · Tobias Langlotz[1] · Shazia Gul[1] · Steven Mills[1] · Stefanie Zollmann[1]

Accepted: 6 January 2023

Abstract
In augmented reality applications it is essential to know the position and orientation of the user to correctly register virtual 3D content in the user's field of view. For this purpose, visual tracking through simultaneous localization and mapping (SLAM) is often used. However, when applied to the commonly occurring situation where the users are mostly stationary, many methods presented in previous research have two key limitations. First, SLAM techniques alone do not address the problem of global localization with respect to prior models of the environment. Global localization is essential in many applications where multiple users are expected to track within a shared space, such as spectators at a sporting event. Secondly, these methods often assume significant translational movement to accurately reconstruct and track from a local model of the environment, causing challenges for many stationary applications. In this paper, we extend recent research on Spherical Localization and Tracking to support relocalization after tracking failure, as well as global localization in large shared environments, and optimize the method for operation on mobile hardware. We also evaluate various state-of-the-art localization approaches, the robustness of our visual tracking method, and demonstrate the effectiveness of our system in real-life scenarios.

Keywords Localization · Tracking · Augmented reality · Virtual reality

1 Introduction

Augmented reality (AR) provides virtual graphics and visualizations to users, accurately aligned with the real world. To achieve this, it is necessary to determine the viewing position and direction (pose) of the user and track their movement in real time. Most commonly, simultaneous localization and mapping (SLAM) approaches [1,2] are used to map the environment while simultaneously estimating the pose with 6 degrees of freedom (6-DoF; 3 rotation and 3 translation). Most traditional SLAM approaches assume that the user will perform a significant amount of translational motion in order to initialize a 3D map through point triangulation. However, there are also 3-DoF SLAM approaches that assume that the user will perform purely rotational motion (i.e., 3-DoF) [3–5]. They do not require a translational motion; in fact, they assume a perfect rotation around a center point with no offset (zero translation).

Unfortunately, for many practical AR scenarios the user will remain mostly stationary (e.g., AR visualizations for seated spectators at sporting events [6]) or will remain in one place [7] while viewing the AR content around the location (e.g., AR Browsers [8]). Grubert et al. found that when reviewing AR browser usage "most of the users were experiencing the application while standing at the same position (78%), combined with rotations (90%)" [7]. In fact, previous work already highlighted the benefits of the purely rotational movement in several application areas [9,10]. In all these cases, the user's translational motion is limited, making 6-DoF SLAM initialization error-prone if not impossible. However, prior work always assumed that the users perform a perfect rotation. But specifically when the AR interface runs on mobile devices such as phones or tablets, the device is generally handheld at some distance from the body. This means the performed motion is not purely rotational either, violating the assumptions of 3-DoF trackers. Unfortunately, this error has often been neglected in the literature. Pure rotational models also have the disadvantage that depth and parallax effects cannot be rendered, so there is a need for localization and tracking methods that can cope with the limited translational motion of a stationary user.

✉ Stefanie Zollmann
stefanie.zollmann@otago.ac.nz

[1] Departments of Computer Science and Information Science, University of Otago, Dunedin, New Zealand

[2] Cal Poly, San Luis Obispo, USA

Published online: 02 February 2023

Recently, there has been research into methods for reconstruction and pose estimation under spherical motion constraints [11]. Spherical motion can be placed in between the 6-DoF and typical 3-DoF tracking methods, as it constrains the translation vector, without eliminating it completely. This provides distinct advantages over 3-DoF systems in that 3D visualizations can be viewed with motion parallax, and placed at varying depths, while also allowing for mapping with global optimization through bundle adjustment and loop closure (as is typically used in 6-DoF SLAM methods to reduce drift [12]). Recently, there has been work into applying these constraints in stationary capture scenarios such as structure-from-motion (SfM) [11], spherical localization and tracking for AR (SPLAT) [13], and reconstruction from panoramic videos [14,15].

Real-world applications of augmented reality require robust solutions to both localization and tracking, and we address this challenge for the case of stationary spherical motion. This restricted case is a good approximation to the motion of users who are situated in a large-scale environment—seated in a stadium in our primary use case. Some previous works propose methods for tracking within pre-computed SfM models of an environment [16,17]; however, they do not address the challenges that are common in many real-life use cases, such as reconstruction from stationary spherical motion, and localization and tracking in large dynamic environments. In this paper, we address these issues by evaluating existing methods and developing a novel approach for global localization for stationary users of AR interfaces. The main contributions of this paper are:

- Design of an overall system which incorporates both state-of-the-art localization, and spherical tracking for AR (Sect. 3).
- Extension of our existing spherically constrained tracking approach [13], with added support for relocalization and global localization (Sect. 4).
- Evaluation of the complete system for accuracy and robustness of tracking (Sect. 5.1).
- A mobile implementation, and performance evaluation (Sects. 5.1.3 and 5.1.4).
- A technical evaluation of state-of-the-art localization approaches in a sports-spectator scenario (Sect. 5.2).

Together these contributions demonstrate the ability to localize, track, and re-localize consumer devices situated in large-scale environments.

2 Related work

Tracking in the context of AR refers to the process of determining the pose of a camera as it moves. For many applications, it is also required that this camera pose is computed in real time. This is particularly true in the case of AR, where the camera pose is used to render visualizations from the user's perspective as they move their device.

2.1 SLAM-based tracking

SLAM has been a popular approach to monocular tracking and can be used in AR scenarios to compute the position and orientation of the viewer in real time. Research in SLAM originated in the field of robotics, with early work by Davison et al. [18] which was later extended to MonoSLAM [19]. Their method uses an extended Kalman filter (EKF) to update camera pose and landmark locations based on measurements from the image. They note that the EKF update can be expensive when there are many features in the map, and address this by tracking a small number of dominant features.

Klein and Murray proposed to separate the mapping and tracking into separate tasks for allowing real-time performance in AR applications [1]. With their Parallel Tracking and Mapping (PTAM) approach, the authors specifically target small AR workspaces. This work was one of the first to take a multi-threaded approach to the SLAM problem.

More recently, ORB-SLAM has been published which offers many improvements over PTAM [12]. ORB-SLAM has more capabilities than PTAM and is designed with the similar idea of using multiple threads to increase computational efficiency on multi-core devices. A recent survey of SLAM suggests that SLAM systems should cover three basic components: initialization of a 3D map, tracking the motion and further map updates, while an accurate and stable solution would additionally cover global map optimization and relocalization [20].

2.2 Rotation-based tracking

The previously discussed works all share a common theme in that they aim to track a camera assuming it has 6 degrees-of-freedom. That is, the camera pose can be described by a rotation R and translation $\mathbf{t}$. However, some tracking methods reduce computation by estimating pose with fewer parameters, such as via homography estimation [21]. Using a homography to estimate the camera pose is accurate provided that either the scene to be tracked is planar, or the motion being modeled is a rotation.

There are two driving factors behind the use of homographies. Firstly, it is fair to assume that the scene may be planar. For example, tracking a camera for field sports can utilize the homography assumption, provided they are able to segment the planar sports field from the image [22], or tracking from obviously planar objects such as books [23] or fiducial markers [24]. While not all of these works explicitly apply homography-based tracking, they clearly demonstrate

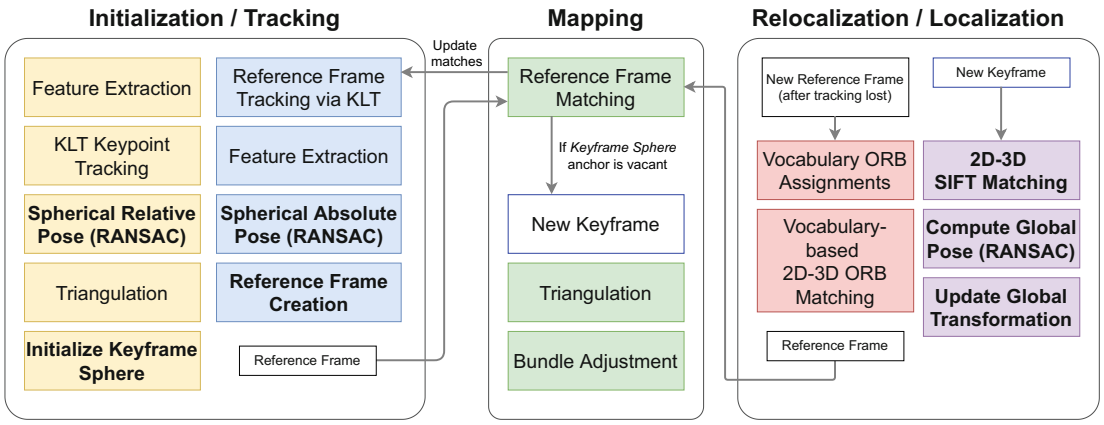

Fig. 1 High-level overview of our tracking and mapping system. The various tasks of the system are grouped by color, and each execution thread is surrounded by rounded rectangles. The key difference to ORBSLAM is highlighted as bold text. The first task (yellow) handles initializing a 3D map and the keyframe sphere, before handing off to the tracking task (blue). This task tracks keypoints from the last reference frame and creates new reference frames when required by the keyframe sphere. The mapping thread has a single task (green) which updates the 3D-to-2D matches of the latest reference frame asynchronously, triangulates new points, and updates the keyframe sphere. Our third thread handles relocalization after tracking failure (red) and global localization (purple). Specific details of the system are explained in Sect. 4

a use case for tracking from planar objects. Secondly, in many AR scenarios and in particular, outdoors we only face a mainly rotational movement which allows us to use homographies [8]. Consequently, there have been several methods that focused on tracking rotations. Many of them can be understood as rotational SLAM algorithms as they work in unknown environments, but instead of building 3D maps of the environment, they commonly build 2D maps (panoramic maps) to represent and track features in the environment [3–5]. Common to all these approaches is that they assume a perfect rotational movement around a center point. Of course, this is almost never the case in particular when rotating a mobile device at arm's reach. While in theory, the error introduced by assuming a zero translation, is relatively small [5], evaluations under more realistic scenarios have shown the error to be up to 4 degrees for the main rotation axis even under optimal conditions, with large scenes containing distant objects and while rotating the handheld phone around an axis with a minimal offset [3]. Assuming a similar error along all axis potentially prevents a seamless AR overlay as rotational tracking errors are known to be more critical when producing precisely registered AR overlays [25]. In this work, we focus on stationary users mainly performing rotational movements, but we extend our previous work using a spherically constrained tracking that considers an offset from the center of rotation [13].

3 Approach overview

While the previously discussed approaches to tracking are important for AR, a key limitation to these systems is their inability to provide global context to the visualizations they enable. This is primarily caused by how the co-ordinate systems are established when initializing the tracking systems—usually one frame of the input is used to represent the origin of the co-ordinate system, and all pose data that are output from the tracking uses this as a reference.

In many application scenarios, it is important to maintain global context while tracking the users. This can be achieved through tracking within a known co-ordinate system such as GPS, or a pre-built model of the environment that is shared by all users of the system. There are many approaches to the localization problem ranging from place recognition [26], structure-from-motion (SfM) approaches which localize images while reconstructing the environment [27,28], geometry-based methods which require prior geometric models (such as line models) of the environment [29], and deep-learning approaches [30,31]. However, many of these approaches do not target real-time performance and thus would require adjustment to be used for real-time AR tracking, particularly on mobile devices.

When localizing from pre-computed SfM models, it is important to consider that the appearance of the pre-computed model may differ from how the environment is presented at the time of tracking. For example, an SfM model may be computed using several thousand photographs all taken on one day within a short time-frame. Lighting and weather conditions can drastically affect the appearance of the environment and make localization through feature matching difficult. While these approaches have shown some robustness to dynamics [32], robustness under these conditions often involves capturing the SfM model under varying conditions which is impossible in the case of a sports sta-

dium, where dynamics caused by thousands of spectators are unique at each event.

While these works focus on the localization of single images, there has also been work on combined localization and tracking from SfM models [16]. In this paper, we argue that creating a local map of the environment during tracking can be beneficial for robust tracking with fast relocalization, but global localization to a pre-computed model is also important. For these reasons, we propose a global localization and tracking system that locally maps features as they are seen, while globally localizing to a pre-computed SfM model using as few frames as possible.

From the related work in the previous section, we can see a trend toward research focusing solely on either tracking or localization, with few works developing systems that incorporate both. For many real-world AR applications, both aspects are important—localization is required to align virtual content with the real world; tracking rather than continuous re-localization provides real-time performance; and some re-localization is required in the case of tracking failures. In this section, we detail a complete localization and tracking system specifically targeting AR applications.

Our overall system consists of five tasks, distributed into three threads. The tasks include initialization, tracking, mapping, relocalization and global localization components. The key differences between our system and state-of-the-art ORB-SLAM are that we introduce (1) a spherical relative pose estimation for the initialization step, (2) a keyframe sphere instead of arbitrary keyframes and a selection of reference frames on the keyframe sphere, (3) a spherical absolute pose estimation for the tracking step and (4) a global localization step (key differences are highlighted in bold in Fig. 1).

We distribute the work of these four components into three separate threads for concurrent processing. Each thread handles a subset of the tasks of the overall system distributed in a way that prioritizes performance in the main thread. In our implementation, we perform both initialization and tracking on the same thread, as these are sequential processes. Initialization (Sect. 4.2) handles the construction of an initial map using two frames from the system and needs to be completed only once. From here, this main thread of execution purely handles the tracking task (Sect. 4.3) which determines the camera pose for each frame of the video.

The mapping task (Sect. 4.4) is handled entirely in its own thread, as this is the most computationally expensive operation. The mapping task takes potential keyframes (reference frames) from the tracking task via a thread-safe queue. These incoming frames are matched to existing map points to detect loops, and new points are triangulated, followed by a global bundle adjustment optimization. The tracking task can see the new and updated map points via the reference frame.

Our third thread handles both global localization (Sect. 4.6) and relocalization from tracking failure (Sect. 4.5). The global localization task of this thread is a new addition which is not handled by ORB-SLAM, and provides a greater context to the tracking result, transforming the tracking pose data to a known co-ordinate space. The purpose of the global localization task is to process keyframes as they are created by the mapping thread to determine their pose in the global co-ordinate system (i.e., with respect to a pre-computed SfM model). The relocalization task handles recovery from tracking failure by re-establishing matches with the local SLAM map. These two tasks are handled again with thread-safe queues, with priority given to the relocalization task resulting in a responsive recovery when tracking is lost.

4 Globally localized tracking for AR

Having discussed several state-of-the-art localization methods, we outline a complete AR tracking system that incorporates these global localization principles with a SLAM-based tracking system. Our system is applicable to situations where a model of the environment has been computed in advance through structure-from-motion (SfM). In this case, our approach is able to provide tracking data within the original co-ordinate system of the SfM model. If these data are not available, the tracking data are returned within an arbitrary local co-ordinate system.

Our method uses real-time tracking based on the SPLAT [13], combined with global localization based on a Bag-of-Words approach [33,34]. We considered two approaches to the problem of providing global context to the tracking system and briefly discussed them here to justify the method we propose.

One approach is to use a pre-computed SfM model solely for tracking, without remapping the environment. This can support real-time tracking of SfM models pre-computed from panoramic images, without the need for further mapping [16]. This can cause issues when dynamics such as lighting, shadows and other spectators change the appearance of the environment from that of the pre-computed model. Other approaches involve registering a local SLAM map to an existing global model [17].

An alternative is to create a local map of the environment as it is currently observed. The local map can then be aligned to the pre-computed SfM model to determine the transformation between the two. In this case, fewer features need to be matched between the current view and the SfM model, as this static transformation can be computed once using two successfully localized frames. Using this approach, the tracking can be performed using the local map, putting less strain on the robustness of the feature matching component.

As the main application scenario of this paper is to track live sport spectators, it is very likely that any pre-computed SfM model will differ in appearance to the current appear-

ance of the environment. For this reason, we take the second approach in an attempt to create a robust tracking system under these conditions. Our approach to tracking is described in detail in the following subsections and is an extension of our previous work on SPLAT [13] with the following key differences:

- Use of a separate processing thread for relocalization.
- Vocabulary tree-based reference frame matching.
- Global localization of keyframes using a pre-computed SfM model.
- Transformation of pose output to represent tracking in this co-ordinate system.

4.1 The keyframe sphere

We use the keyframe sphere approach of SPLAT [13] to subdivide the space of possible camera poses in a way that is tailored to a spherically constrained keyframe SLAM system. This is achieved by generating a fixed number of anchor points approximately uniformly across the surface of a sphere and assigning keyframes to these positions.

To reduce the overhead of excessive keyframes, we use fewer anchor points than previously [13]. We reduce the number of anchor points from 1000 to 500, finding no degradation of robustness. We also set a threshold requiring that a new keyframe's camera center must be within some small distance of an anchor point. As we have fewer keyframe anchors, we also increase our distance from 25 to 75% of the distance between two neighboring anchors to ensure that enough keyframes are created.

4.2 Initialization

Our system automatically initializes a map and begins tracking when enough spherical motion has occurred, using the process described below. The overall initialization step is based on spherical SfM [11], but with a focus on faster computation by only triangulating from two frames.

Feature extraction and tracking We extract 1000 ORB features [35] in the initial frame. To match keypoints between successive frames, we find matches using a pyramidal KLT feature tracker [36]. At each frame, we use these 2-D matches to make an estimate of the current pose relative to the initial frame with a spherical constraint [11].

Relative pose estimation To determine the relative pose for initialization, we use the spherically constrained relative pose estimation introduced in [11]. We assume that the camera moves on a mostly circular path with a constant radius of 1 unit from the origin, and that the viewing direction of the camera is in alignment with the normal of the unit sphere. These assumptions reflect the offset from the center of rotation for stationary users (e.g., when holding a device at arm's reach) and allow us to simplify the pose estimation. The camera pose is given by $[\,\mathsf{R} \mid \mathbf{t}\,]$ where $\mathbf{t} = [0\ 0\ -1]^\mathsf{T}$, and the camera center $\mathbf{c} = -\mathsf{R}^\mathsf{T}\mathbf{t}$. We use the method of [11] for both computing and decomposing an essential matrix to compute the spherically constrained relative pose. In combination with Preemptive RANSAC to discard outlier tracks [37], this determines a relative pose between the first two camera frames.

Triangulating the initial map We use the keyframe sphere structure to determine whether the angular motion is sufficient for initialization. If the poses of the start and end frame are assigned to two different keyframe anchors, we proceed with initialization. Features in the final initialization frame are matched to the initial frame using the feature tracks, and triangulation performed to compute the 3D points.

4.3 Tracking

After the map has been initialized with two keyframes, and the 3D points from triangulation, we use these data as input for our tracking method. The tracking step can be described by the following components:

Feature extraction and tracking Similar to the feature extraction from the initialization phase, we have an upper limit of 1000 keypoints and descriptors. We again use KLT feature tracking [36] to keep track of matches from the most recent keyframe. We then compute a convex hull [38] surrounding the tracked keypoints and find new ORB features outside this mask.

We enforce a 1000 keypoint limit which includes both the KLT tracks (and their already known descriptors), as well as the newly detected features. New features are extracted in each frame; however, it could be optimized to only compute new descriptors in reference frames, as it is only here where matching descriptors to the existing 3D points occurs.

3D-to-2D feature matching We next use the extracted ORB features to obtain more 3D-to-2D correspondences. The KLT tracks maintain references to 3D map points found in the last reference frame, which in many cases is enough to track. However, we must also find potential correspondences between the newly detected features, and points which have already been mapped to avoid triangulating duplicate points.

In a separate thread, we perform brute-force matching between the features in the current reference frame (the origin of the current KLT tracks) and the other keyframes. When new matches are found, the correspondences are updated in the reference frame, which can be accessed asynchronously (via their corresponding feature tracks) in the main thread.

Absolute pose estimation To estimate the pose for each frame, we use the Perspective-2-Point (P2P) method from [13] which uses a spherical pose constraint within a preemptive RANSAC scheme to determine the current pose, and an inlier set of matches [37]. Again, here the spherical pose

constraint reflects the offset from the center of rotation for stationary users.

Reference frames Once a frame is successfully tracked, we decide whether it will become a reference frame. The reference frame is updated when a tracked frame falls within the keyframe sphere distance threshold of a new anchor point. When a reference frame is created, the mapping thread matches its keypoints to all neighboring keyframes and merges the observations. This is equivalent in some respects to loop closure when loops are small, and drift is relatively small.

Keyframes A reference frame will become a keyframe if its anchor point in the keyframe sphere is unoccupied. In this case, the feature tracks from the previous reference frame are triangulated, and bundle adjustment takes place. Since the tracking thread is acquiring many matches through feature tracks to this frame, the new map points are automatically assigned to the current tracked frame as soon as they are ready. This allows new map points to be added asynchronously.

4.4 Mapping

The main purpose of the mapping step is to process potential keyframes from the tracking task. For this step, we match incoming frames to existing map points and detect loops as well as perform a global optimization step on all existing map points. This means that once a frame has been tracked successfully by the tracking task, it is sent to the mapping thread to use the newly visible feature points in the image to update the map in three main stages consisting of (1) reference frame matching, (2) triangulation to update the map and (3) bundle adjustment.

Reference frame matching When new keyframes are stored, the tracking thread follows 2D features from the latest reference frame using KLT tracking. The mapping thread uses these results to guide the matching of keypoints between these two frames. For this purpose, we project known 2D keypoints from the last reference frame into the current frame to help guide the matching.

Additionally, the tracked features in the new keyframe are matched with existing 3D points in the map using a vocabulary-based approach [34]. The use of a feature vocabulary reduces the amount of processing required to match points compared to brute-forced approach. This is the same matching method we use for relocalization, which we describe further in Sect. 4.8.4.

Triangulation The computed matches are then triangulated in a similar manner to the initialization phase using the pose data of both map frames (the newly added and the existing reference frame). If more than 50 map points were successfully triangulated from the matches, then the new keyframe is added to the keyframe sphere.

Bundle adjustment Bundle adjustment takes place after triangulation to optimize the 3D point locations and keyframe camera poses. To enforce the spherical constraint, the camera translation is fixed in the optimization [11]. Once this process is complete, we remove keyframe references to outlier 3D points using the same reprojection threshold set in the tracking thread. The positions of the remaining points and camera poses of the keyframes are then updated. We found that running a small number of iterations each time a keyframe is added is a good way to keep map updates frequent. This also allows for faster performance than full-bundle adjustment at the cost of some accuracy. The use of fewer iterations has also been shown to be beneficial where fast computation is needed [39]. In our experiments, we use two iterations of bundle adjustment between map updates.

4.5 Relocalization

In the case of tracking failure we relocalize by matching the current (untracked) frame with the existing keyframes The keyframe features are more likely to resemble the current appearance of the environment than those in the SfM model. This allows fast and robust relocalization in cases where the environment is very different from the prior model.

In our implementation, we detect tracking failure when the number of inlier 2D-to-3D correspondences falls below a threshold of 30. The global localization thread then prioritizes localizing the currently untracked frame using a purely ORB variant of the BoW approach of Sect. 4.8.4. While this process runs, the tracking thread uses the same KLT approach as Sect. 4.2 to re-initialize the reference frame and resume tracking as normal.

4.6 Global localization

In order to align the pose to the global coordinate system of the SfM model, we first localize at least two keyframes with respect to the SfM model. Next, we compute the scale factor between the local tracked coordinate system and the SfM model, and finally we compute a transformation that describes the rotation and translation differences between the systems.

Keyframe localization First, we localize two keyframes F_1 and F_2 from the tracking system to the COLMAP model. In our implementation, we extract 2000 SIFT features [40] from F_1 and F_2, and perform brute-force matching with the existing SIFT features from the SfM model, using a distance ratio test to discard ambiguous matches. We then compute the global rotation and translation, R_i^w and $\mathbf{t}_i^w$, using P3P [41] within a Preemptive RANSAC loop [37]. In our system implementation, we used this simple feature matching approach to localize keyframes. However, we further investigate and evaluate the state-of-the-art approaches in Sect. 4.7.

Fig. 2 Two views of a spherical trajectory computed with our tracking system. The result is rendered within the meshed COLMAP reconstruction of our Rugby Stadium dataset to demonstrate the effect of global localization and scaling

Computing the transformation We now have a global pose R_i^w, $\mathbf{t}_i^w$, and a local pose R_i^l, $\mathbf{t}_i^l$ from the SPLAT tracking for two of the keyframes. To compute the difference in scale, we take the distance between the centers of F_1 and F_2 in both co-ordinate systems where the camera center $\mathbf{c}_i$ of keyframe F_i is defined as

$$\mathbf{c}_i = -\mathsf{R}_i^{\mathsf{T}}\mathbf{t}_i. \tag{1}$$

We then compute the scale s as the ratio of the two distances

$$s = \frac{|| \mathbf{c}_1^w - \mathbf{c}_2^w ||}{|| \mathbf{c}_1^l - \mathbf{c}_2^l ||}. \tag{2}$$

Finally, we compute the transformation between the coordinate systems using one of the localized keyframes F_i. We first define the 4×4 scaled local pose P_i^l as

$$\mathsf{P}_i^l = \begin{bmatrix} \mathsf{R}_i^l & s\mathbf{t}_i^l \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3}$$

and the global pose P_i^w as

$$\mathsf{P}_i^w = \begin{bmatrix} \mathsf{R}_i^w & \mathbf{t}_i^w \\ \mathbf{0} & 1 \end{bmatrix}, \tag{4}$$

and compute the 4×4 transformation T as

$$\mathsf{T} = \mathsf{P}_i^{l-1}\mathsf{P}_i^w. \tag{5}$$

Using T and s, the pose results from each frame of tracking can then be transformed into the global coordinate system of the SfM model. The resulting trajectory is spherical and localized within the SfM model as depicted in Fig. 2.

4.7 Discussion

One of the key differences between our approach and existing methods is the use of both a local map and a pre-computed global map. Many approaches perform tracking directly from a pre-computed SfM map [16]. Real-time tracking is achieved by matching features in the current image to the features that comprise the SfM model. This approach provides fast real-time tracking, as there was no need for computational resources to be used on mapping during the tracking process. However, when the environment is dynamic, feature matching can be unreliable as the appearance of the environment may change. This can cause issues when this type of feature matching needs to occur very frequently.

Our approach handles dynamic environments by matching as few frames as possible to the SfM model. While localization under dynamic conditions is possible [32], our system needs only to localize two keyframes in order to compute an approximate transformation between the local and global coordinate systems. Once these frames are localized, we track from a local map which more accurately represents the current appearance of the environment. Another approach that is similar to ours is to track from a locally created SLAM map, which is registered to a '2.5-D' global representation of the environment [17].

For these reasons, our system tracks from a globally registered local map with the intention of improving robustness in dynamic environments. In the following section, we discuss and evaluate the state-of-the art methods for solving the localization problem, to gain better insight into which approaches work best for stationary AR. In Sect. 5.1, we provide results on the accuracy and robustness of the tracking component of our system.

4.8 State-of-the-art localization methods

Robust localization is an important part of our system, as it enables us to provide global context to the pose output from the local tracking component. Previous research on localization with SfM models can be coarsely grouped into three categories: feature matching [42], image retrieval [26] and deep learning approaches [30,31]. We focus on the most promising open-source systems in each category: ESAC (Expert Sample Consensus released with [31]); Active Search released with [43]; and our own custom implementation of a Bag-of-Words (BoW) localizer based on the implementation of Muñoz et al. [34] called fbow. We compare these methods' ability to localize with our SfM models built using COLMAP [27], using COLMAP's vocabulary tree image registration as a localization reference.

Our first method is Active Search [43], which we choose as it is frequently used as a benchmark for image-based localization approaches and is provided open-source. We also investigate ESAC [31] as it appears to be among the most promising and scalable of all the deep-learning approaches to localization, while also being provided open-source by the authors. Our third method is our own implementation of a Bag-of-Words localizer, based on ORB and fbow for image retrieval [33,34] while using SIFT for registration [40]. These

BoW solutions tend to focus on image-retrieval results and do not provide results for the type of accuracy that can be achieved when they are used for full 6-DoF localization, so we implement our own using these libraries.

4.8.1 Data preparation

Our primary application is localization for spectator AR in a sport stadium environment. Due to the limited availability of public stadium datasets, we captured our own in two stadia. The first step in our data processing is to create a sparse reconstruction using the training images of each dataset. For this, we used the open-source COLMAP SfM software from [27] with default parameters.

Cricket ground (CG) Here, we captured a small dataset of 49 images from several positions within one side of the stadium during a single visit. The lighting condition was sunny, and the stadium was at near-full capacity. These data were randomly split into 39 training and 10 test images (approximately 8:2 split).

Rugby stadium (RS) We captured 1125 images from a range of positions primarily from the two opposing main stands, with some taken from ground level. This dataset was captured over two daytime visits to the empty stadium, one overcast and one under sunny weather conditions. This dataset was randomly split into a training set of 900 and a testing set of 225 images (8:2 split). As this stadium has a translucent roof structure, the cloudy and sunny conditions were similar enough to be combined into one reconstruction. We later investigate dynamic scenes in Sect. 5.2.3.

4.8.2 Active search

Active search [43] uses a bi-directional feature matching method. First, a descriptor vocabulary is used to quantize the descriptor space, and words are assigned to each point in the model as well as to each feature in the query image. For each feature $\mathbf{f}$ in the query image, the 3D points which share a node in the vocabulary tree are searched for matches using the typical ratio test [44] resulting in an initial match to a point $\mathbf{P}$.

Then, the 3D points in the neighborhood of this match are prioritized and matched to the features in the inverse direction using a coarser vocabulary. The purpose of this bi-directional matching is to make use of the fact that points in the same 3D region are likely to share similar visibility.

4.8.3 Expert sample consensus

The expert sample consensus (ESAC) approach introduced by [31] is to train a convolutional neural network to learn scene co-ordinates for a given input image using both scene co-ordinate images and 6-DoF pose as ground truth. The localization component uses RANSAC to sample the output scene coordinate images, which naturally encode 2D-to-3D correspondences for pose estimation.

The method was first introduced in [45], where the authors presented a modification to RANSAC which allows the entire pipeline to be differentiable, allowing for gradient-descent end-to-end learning. Their method was later improved in [46], and their most recent system ESAC [31] improves scalability by clustering the dataset and training first a scene classifier, followed by an ensemble of expert networks that are able to operate on the smaller scene clusters.

Pre-processing The ESAC localization approach requires more data to train the CNN, in addition to the pose information from COLMAP. This method uses ground truth scene coordinate images for its own training process. Scene coordinate images are like depth maps; except instead of encoding a depth value (i.e., distance from camera to scene) into each pixel, the full 3D scene coordinate is stored resulting in a $3 \times H \times W$ tensor. It is possible to attain a dense representation of these data from a sparse model through dense MVS reconstruction methods [28]; however, a sparse representation is sufficient [46] and in fact completely optional, as the entire pipeline is capable of learning the scene structure.

Localization After processing the datasets with COLMAP to acquire a sparse reconstruction, the point cloud is then projected into a small representation of the training images using the known pose from the reconstruction ($H = 60$ and $W = 80$, [31]). For each pixel, we encode the 3D coordinates of the nearest point projected to that pixel using a z-buffer. We exclude points behind the camera and leave zeros for empty pixels.

4.8.4 Bag-of-Words localization

We also compare to localization based on BoW techniques. Our system is designed to operate on the output of a typical COLMAP reconstruction and should not be considered state-of-the-art, but represents the expected performance from the localization method with a simple implementation. Our system can be used in two stages, pre-processing and localization.

We also require a vocabulary file containing representative ORB descriptors and use the one provided with ORB-SLAM2 [47]. The BoW approach allows for quick matching between images using an inverted file that contains both image and keypoint indices for each word. We investigate the feasibility of fast image retrieval using ORB, while maintaining robust matching via SIFT.

Pre-processing This stage only needs to be completed once per COLMAP model. The objective of this phase is to create an inverted index file, which stores for each word in the vocabulary, a list of image identifiers corresponding to the training images that contain that word. The purpose of

this file is to act as a database for image retrieval and only needs to be computed once per SfM model.

We first detect 2000 ORB features in each training image and then map the descriptors to words in the vocabulary using the optimized transformation implementation of fbow [34]. Finally, the inverted index is updated by appending the image identifier to the corresponding list in the inverted file for all transformed words in the image.

Localization To localize a query image, we first detect and map ORB features to the vocabulary as before. Then for each word in the query image, we parse the list of training images via the inverted file and accumulate votes for each training image that contains that descriptor. The image with the most votes is accepted as the closest match, from which we begin establishing matches to compute a 6-DoF pose.

To achieve this, we then detect SIFT features in the query image, as they tend to show better matching rates than ORB [48]. Then we apply a FLANN-based matching method [49] to find the two nearest neighbors for each potential match and discard unreliable matches using a ratio test[44]. Finally, we apply an iterative solution to the Perspective-n-Point (PnP) problem based on [50] within a RANSAC scheme [51] to compute the 6-DoF position and orientation of the query image.

5 Evaluation

A key aspect of our approach to AR stems from the ability to track and localize users in an environment accurately and robustly. In the previous section, we outlined some of the key methods in tracking and localization, and their theoretical limitations. In this section, we evaluate these methods in detail to determine the most suitable approaches for globally localized tracking in large dynamic environments with a focus on sports spectating. Specifically, we make use of our own SfM models created from image datasets of large sports stadium environments. Our SfM models are generated from the image datasets using COLMAP [27].

5.1 Tracking results

To evaluate the tracking component of our system, we compared the rate of successful tracking between state-of-the-art monocular ORB-SLAM2 [12,47] and our approach across multiple sequences in two different real-life environments (a sports stadium, and an outdoor basketball court).

To provide insight into the accuracy, we also compare the output pose results to state-of-the-art monocular ORB-SLAM2 [12,47] in a synthetic environment to provide insight into the accuracy of the tracking.

We then show some qualitative results output from our AR prototype to demonstrate how accurate the registration appears in real AR use cases. Finally, we evaluate the computational performance of our approach on different hardware platforms.

5.1.1 Robustness results on real datasets

To evaluate our system, we investigate the robustness of tracking under realistic scenarios. We compare the successful tracking rate of our system to ORB-SLAM2 [12,47] on eight different video sequences. Results using 1000 and 2000 ORB features per frame, are shown in Fig. 3.

Four sequences (Stadium A-D) were captured in a sports stadium from a spectator's perspective, and we used the RS SfM dataset (Sect. 5.2) as our prior global model. The other four sequences (Court A-D) were captured at an outdoor basketball court, again from a spectator's perspective, and include two sequences that deliberately obstruct the camera to test relocalization (Court B, and C). For the court dataset, we captured 211 images from various perspectives around the court and used the same COLMAP process [27] as for the RS data to create a prior SfM model.

5.1.2 Synthetic tests for accuracy evaluation

For evaluating the accuracy of the overall approach on a large scale, we decided to use synthetic data as it allows us to measure deviations from the ground truth pose in more depth.

To compare the accuracy of our system to state-of-the-art, we compare the absolute trajectory error (ATE) and relative pose error (RPE) [52] of our system and state-of-the-art ORB-SLAM2 to synthetic ground-truth. We generate ground-truth data by moving a camera in a circle within a textured sphere and compare trajectories using the *evo* odometry evaluation tools [53]. The ATE is the average difference between two estimated positions at each time point after they have been aligned, while the RPE measures the difference in estimated trajectories over short time intervals.

To determine how the accuracy of the systems scale with the size of the environment, we increase the textured sphere radius to generate sequences from environments of varying scale from 2× up to 50× the radius of the camera motion. In Fig. 4, we plot the mean ATE and RPE for each synthetic video sequence against the 3D sphere size. The RPE metrics were taken at 1-frame intervals, which corresponds to 0.36° of circular motion at radius 1.

5.1.3 Results in AR spectating application

To demonstrate the accuracy of our tracking approach qualitatively, we also implemented an AR rendering system based on OpenGL as well as one using Unity3D. In the case of the RS dataset, we had a textured CAD model of the stadium which could be used as a basis for AR content creation.

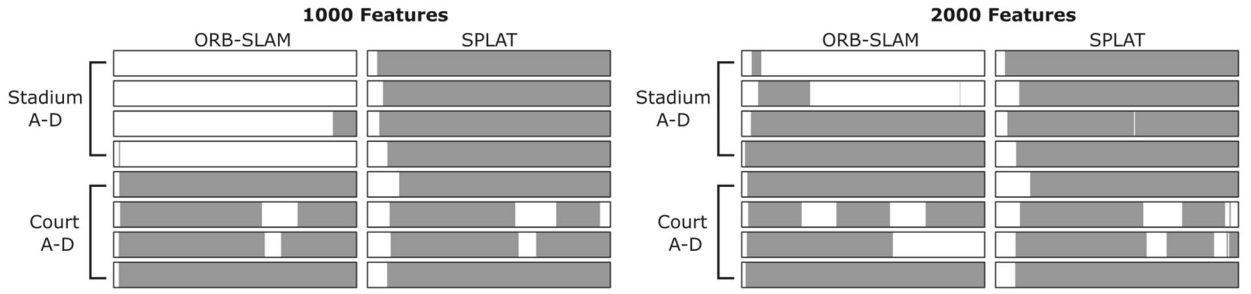

Fig. 3 Comparison of the tracking rate of ORB-SLAM2 [47], and our proposed system with two different configurations for the number of features per frame. Each bar represents a tracking timeline for a test video sequence. White represents an untracked frame. Grey represents a frame tracked in the local SLAM coordinate system

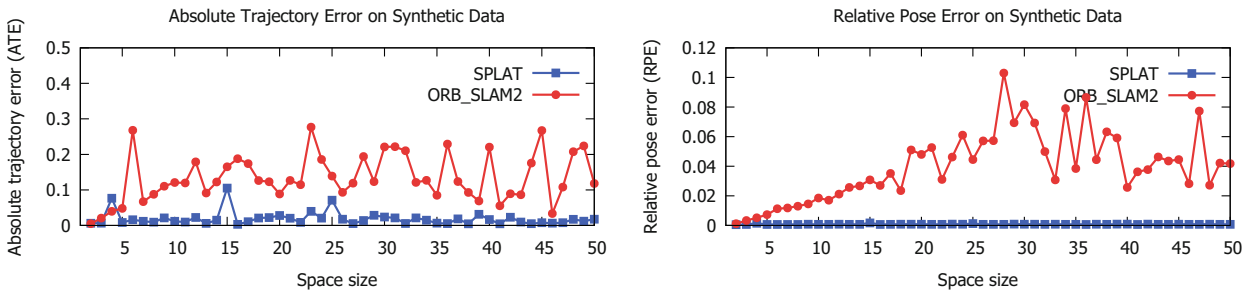

Fig. 4 Pose error from synthetic testing of our tracking system (blue), compared to monocular ORB-SLAM2 (red [12]) with different 3D sphere radii. The radius of the camera motion is fixed at 1 unit, and the size of the surrounding textured sphere was varied from 2 to 50 units

We aligned this CAD model to the SfM model by manually identifying corresponding points and aligning these with 3D modeling software. We then applied the resulting 4×4 transformation to our model matrix when loading the CAD model. The resulting alignment demonstrates that the measured accuracy (as evaluated in Sect. 5.1.2) is sufficient to place 3D content within the stadium environment to visualize game-related content such as heat maps or scoreboards (Figs. 5 and 6).

Fig. 5 Example input frames (top row), and corresponding output frames from our AR renderer (bottom row). Both a CAD model of the environment (lower left) and a heatmap overlay (lower right) are shown to demonstrate the capabilities for AR to display in-place sports statistics

5.1.4 Performance evaluation

In addition to the accuracy and robustness evaluation, we also provide results on the runtime performance of our approach (Table 1). Since AR applications are time critical, we measured the framerate of our method on two hardware platforms. We recorded the minimum, maximum and mean framerate (in frames per second) of our tracking approach using the Stadium-A sequence on a 2016 laptop PC with an Intel i7 6200U processor at 2.3 GHz. We also record the same metrics on an Android smartphone with a Snapdragon 845 (2018). On the smartphone implementation, we used a live feed from the device's camera as input and moved the device in a similar pattern as done in the Stadium sequences. Neither smartphone or desktop implementations make use of the GPU.

The results from our performance experiment are shown in Table 1. We were able to achieve essentially real-time performance on the laptop PC with a mean framerate of 29.5 frames per second. Our Android implementation ran at an average of 12 fps which is not realtime, but still offers an interactive experience for the user. In the future, this could be improved by combining our tracking approach with inertial data from mobile device sensors, which are often available at much faster rates.

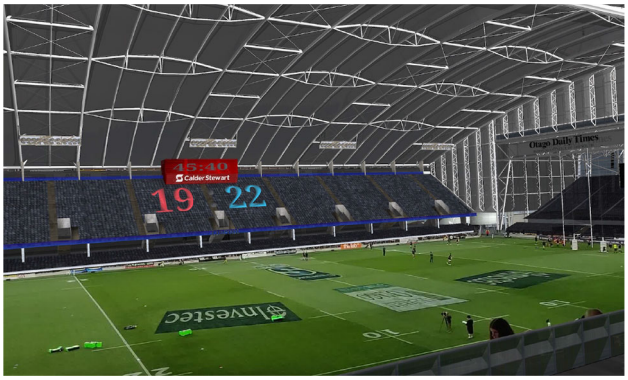

Fig. 6 Example of our AR spectating application showing a virtual scoreboard overlaid with the input frame using the result of the global localization. The AR app is implemented using Unity3D

Table 1 Framerate statistics of our tracking approach on two devices

Device	Min (fps)	Mean (fps)	Max (fps)
i7 6200U	10.417	29.514	45.455
Snapdragon 845	4.999	12.006	29.994

5.2 Localization results

We were also interested in providing more insights into the performance of the different localization options we identified. For this, we tested the methods' ability to localize the test set images against the SfM models. In this section, we compare the computation time of both training (one-off computation), and the localization itself, as well as accuracy metrics when compared to the COLMAP reference method. We also provide and compare qualitative results from renderings of the sparse point clouds overlaid with the input query images.

5.2.1 Computation time

The computation times for each of the methods are reported in Table 2. Localization time refers to the mean time to localize over all query images. Training time refers to the time taken to pre-process the model or images for each localization method. Computation times were measured on a PC with an Intel Core i9-9900KF CPU at 3.60 GHz, 32 GiB memory and a GeForce RTX 2080 Ti video card.

For ESAC, the training time includes initializing the gating network (classifier), initializing and refining 4 experts, and the end-to-end training stage. For Active Search, this includes the pre-processing stages of parsing the SfM data and computing descriptor assignments with the vocabulary. For our BoW method, this includes all steps detailed under pre-processing in Sect. 4.8.4, namely ORB detection, fbow transformation and creation of the inverted index.

Table 2 Computation times from localization approaches on stadium datasets

	Dataset=CG, Test set size=10		Dataset=RS, Test set size=225	
Method	Train	Localize (ms)	Train	Localize (ms)
Active search	2.77 s	158	116 s	221
ESAC (4 experts)	101 h	88	85 h	75.2
ESAC (1 expert)	34 h	92	22 h	145
BoW	1.27 s	603	30.2 s	1559
COLMAP	–	420	–	468

We found that the ESAC approach has the longest training time, which is to be expected from a deep learning approach. However, training time is not particularly critical, and 101 hours is not prohibitively long for a one-off computation. We thus deem all methods to be viable for localization with regard to training time. ESAC [31] reported the fastest localization time with an average of 88 and 75.2 ms on the CG and RS datasets, respectively.

In Table 2, we see that the BoW approach had the fastest training time, likely due to the optimizations of fbow, combined with more compact ORB descriptors. However, the localization time was the slowest, due to the overhead introduced by requiring both ORB and SIFT computation for the query images. For this reason, we conclude that the performance potential from using ORB over SIFT is beneficial in the indexing and pre-processing stages, but attempting to leverage the matching robustness of SIFT in conjunction with the simpler computation of ORB can be detrimental to performance. This suggests that using ORB features for both tasks might produce better results.

We find that all methods produce acceptable localization times, with the slowest being our BoW approach, which is not yet optimized for speed on the localization side. While our method localized the slowest, realistically the localization process would only need to happen occasionally in an overall AR system. For example, localizing with only the initial frames or keyframes of a SLAM system. These methods, therefore, do not need to perform in real time.

5.2.2 Error metrics

The localization results showed potential for most of the methods, especially with ESAC, and Active Search. ESAC results from the RS dataset are shown in Fig. 7.

Out of the 10 CG test cases, 4 were reported successful by our BoW approach. We note, however, that some cases are reported successful but exhibit visible errors when overlaying the points with the image. In our results, we use '# Reported' to refer to the number of images reported to be successfully registered by the method itself.

Fig. 7 Example of accurate localization results from ESAC on the RS dataset. We render the sparse point cloud from COLMAP over the input images using the pose estimate. An accurate localization results in good alignment of the keypoints (white) with their corresponding structures in the image

Due to the potential for false positives, we take the localization results from COLMAP as a reference, $\hat{\mathsf{P}}$, and compare them with the reported successful pose, P, using a geometric error over all co-visible points $\mathbf{x}_1, \ldots, \mathbf{x}_n$,

$$\sum_{i=1}^{n} \frac{|| \mathsf{P}\mathbf{x}_i - \hat{\mathsf{P}}\mathbf{x}_i ||}{n}, \tag{6}$$

where $\mathsf{P} = \mathsf{K}\,[\,\mathsf{R} \mid \mathbf{t}\,]$.

We then flag any result with a geometric error of less than 10 pixels as a successful registration (# Actual). To get an idea of how accurate the successful registrations are with each method, we also compute the mean (Err. M) and standard deviation (Err. SD) of the geometric error over all the true positive cases. All methods were tested with images of the same resolution (480 pixels in the shortest dimension, as required by ESAC). The 10 pixel threshold corresponds to approximately 1.5% of the image width.

To avoid making comparisons between potentially erroneous poses from the COLMAP reference, each reference registration was visually checked for quality. Ideally, we would have a higher-quality reference or ground-truth poses for the test set, which is an area for future work.

The results for all methods on the CG and RS datasets are shown in Tables 3 and 4, respectively.

5.2.3 Dynamic localization

As we have seen in the previous section, the ESAC and Active Search localization approaches both appear promising for localizing sport spectators. However, in our RS dataset, we only evaluated with images of an empty stadium (for both reconstruction and localization). The accuracy results we previously saw from ESAC showed small errors, which suggest a very accurate alignment to the poses from the COLMAP reference, from which that method was trained.

Table 3 Error and success rate of localization approaches on the CG dataset

	Dataset = CG, Test set size = 10			
	ESAC (4 experts)	Active search	BoW	COLMAP (reference)
# Reported	10	0	4	7
# Actual	6	0	3	7
Err. M (px)	1.67	–	6.99	–
Err. SD (px)	0.86	–	3.04	–

Table 4 Error and success rate of localization approaches on the RS dataset

	Dataset = RS, Test set size = 225			
	ESAC (4 experts)	Active search	BoW	COLMAP (reference)
# Reported	225	215	161	221
# Actual	209	203	119	221
Err. M (px)	1.91	3.27	3.43	–
Err. SD (px)	1.68	2.00	1.81	–

We present a qualitative view of how these results would be visualized in a realistic scenario, with reconstruction made from images of an empty stadium, and localization images taken under different dynamic conditions, such as changes in lighting and the presence of spectators. For these tests, we use different images from those previously used, captured during a live rugby game.

As shown in Fig. 8, we can see that both the Active search and ESAC approaches have the potential to produce highly accurate localization. As we do not have ground-truth poses for these additional images, we are unable to perform more evaluation on the accuracy. However, upon visual inspection of the projected points in Fig. 8, both methods produce very similar results with good overall alignment of the sparse model to the image. Though these results appear promising, a more thorough evaluation of the effects of the dynamic elements would be needed in the future.

Dynamic rugby stadium To test our method in a more realistic scenario, we extended a dataset which contains hand-annotated reference points for each image which can be used as an independent reference to measure reprojection error [54]. The dataset is captured in the same rugby stadium environment from previous experiments and is split by three conditions to test this environment in a range of different dynamic complexity: Empty, Semi-crowded and Crowded.

We then ran our experiment again across all four localization approaches and present the resulting reprojection errors in Tables 5, 6 and 7. We used the same error metrics outlined in Sect. 5.2.2, with a threshold of 1.5% of the image width as before (10.8 pixels for Empty cases and 28.8 pixels

Fig. 8 Qualitative results of localization in a dynamic stadium with spectators, and different lighting (Rugby Stadium—Crowded dataset). Both ESAC and Active search produce very similar results when aligning the sparse point cloud with the image using the pose outputs

Fig. 9 Samples from our second Rugby Stadium dataset, extended with three conditions. Left to right: Empty, Semi-crowded, and Crowded

Table 5 Localization results on the empty rugby stadium dataset

	Dataset = Empty, Test set size = 15, Err thresh = 10.8 px			
	ESAC (4 experts)	Active search	BoW	COLMAP
# Reported	15	14	14	15
# Actual	15	14	11	15
Err. M (px)	2.87	3.34	3.77	4.72
Err. SD (px)	0.76	0.87	1.70	1.05

for Semi-crowded and Crowded cases). And example image from each sub-dataset is shown in Fig. 9.

6 Discussion

In this section, we discuss the results of our evaluation on tracking and localization, a comparison of the tracking component with ORB-SLAM2 and the limitations of our system.

6.1 Real-data tracking

When compared to ORB-SLAM2 [12,47], our system is able to track successfully more often. Most notably, ORB-SLAM2 was unable to initialize reliably in the stadium test videos when 1000 ORB features per frame were used, though initialization was successful with 2000 features. However, even with this many features, tracking was lost shortly after initialization on the Stadium A and B sequences.

In the Stadium environment, there are many repetitive features from empty seats in the stands. ORB-SLAM2's failure to initialize could be due to its use of a feature vocabulary that quantizes the descriptor space too coarsely, combined with an over-reliance on feature matching during tracking. In our SPLAT-based system, we only use a feature vocabulary during relocalization and track features frame-to-frame with KLT, which relies less on correct descriptor matching under these challenging conditions.

In the Court environment, both approaches work reasonably well, but interestingly both approaches seem to perform better with 1000 features per frame. This could be due to the relatively low resolution of the images compared to the number of features. An increased number of features can solve some problems, as in the ORB-SLAM2 Stadium cases, but in other scenarios may add unnecessary noise to the feature matching processes by detecting ORB features at less dominant keypoints.

One apparent limitation of our system is that we can see three instances where tracking appears to become less robust after the relocalization point in Court B and C. While both approaches could successfully relocalize, our approach lost tracking for some frames in some instances. This could be due to the lack of matches to the existing map points after relocalization. During tracking, and before a tracking failure, our system is usually able to maintain many 2D-to-3D matches aided by the robustness of the KLT tracking. However, after a tracking failure fewer matches could be re-established using our vocabulary tree-based relocalization approach. For future

Table 6 Localization results on the semi-crowded rugby stadium dataset

	Dataset = Semicrowded, Test set size = 32, Err. thresh = 28.8 px			
	ESAC (4 experts)	Active search	BoW	COLMAP
# Reported	32	28	10	32
# Actual	29	22	0	12
Err. M (px)	11.39	11.94	–	21.08
Err. SD (px)	5.52	4.48	–	6.22

Table 7 Localization results on the crowded rugby stadium dataset

	Dataset = Crowded, Test set size = 14, Err. thresh = 28.8 px			
	ESAC (4 experts)	Active search	BoW	COLMAP
# Reported	14	10	3	14
# Actual	12	10	0	7
Err. M (px)	12.68	12.28	–	16.07
Err. SD (px)	5.78	6.37	–	6.15

work, this could be improved such as by using an Active search [55]-based relocalization approach to establish more correspondences.

6.2 Synthetic data tracking

The ATE results for ORB-SLAM2 suggest that increasing the space size beyond 10 units introduces large variations in the ATE. Looking at the RPE metric, there is a more steady increase of error up to approximately radius 15, which is similar to the point where previous work found the tracking rate of ORB-SLAM2 began to fail [13]. The large variations beyond this point are likely due to early tracking failure, resulting in fewer successfully tracked frames to compare with the reference poses. Overall, the results suggest that SPLAT has a lower pose error with both metrics. The error remains low with respect to space size, without suffering from the steady RPE error scaling exhibited by ORB-SLAM2.

6.3 Stadium localization

The active search method of [43] produced robust registration with the RS dataset, with a higher success rate compared to our BoW results, and with a faster localization time. However, active search failed completely on the CG dataset likely due to the low number of points in the sparse model.

In our initial testing, many results from active search were reported as 'successfully registered,' while the point renderings were noticeably misaligned. This could be because the provided active search implementation computes both intrinsic and extrinsic camera parameters, and often incorrectly estimates large skew values in the intrinsic matrix. For this reason, we modified the original implementation of active search to use the same PnP solver as our BoW implementation [50,51]. Using this method, we supply a fixed camera matrix and estimate the pose directly, achieving more consistent results.

To compare the accuracy between the three methods we make three comparisons, so we must account for this. We start with a typical threshold of $p = .05$ and apply Bonferroni correction to get an adjusted threshold of $p = .01\dot{6}$. As each method has the potential to succeed or fail on each sample (producing no accuracy output), performing a paired test is impractical. For this reason, we use two-tailed unpaired t-tests here. We compared three groups of accuracy results: ESAC, Active Search and BoW. ESAC (M = 1.91, SD = 1.68) showed significantly lower geometric error when compared to active search (M = 3.27, SD = 2.00) with $p < .0001$ and BoW (M = 3.43, SD = 1.81) with $p < .0001$ in both cases. However, comparing active search to BoW showed no significant difference ($p = .474$).

The smaller error output by ESAC could be due to the fact that our COLMAP poses are actually a reference and cannot be regarded as ground-truth. As active search and BoW compute poses independently of the COLMAP training poses, their error could be attributed to the noise in the point cloud, whereas ESAC is trained specifically to replicate the training poses, and is able to refine the point estimates that were provided in the form of the scene coordinate images. An example of successful localization from the ESAC approach is shown in Fig. 7.

6.3.1 Dynamic stadium localization

The results from this experiment showed that the key state-of-the-art localization approaches performed well when localizing under different environmental conditions to those when the original model was captured. Active Search and ESAC both performed robustly with high localization success rate in all three cases, but ESAC had a higher success

rate while maintaining a similar level of reprojection error to Active Search.

The BoW approach was much less robust in these conditions, and most cases had failed due to too few matching features between the query image and the most similar training image. This result demonstrates that robust localization is heavily reliant on a good feature matching strategy.

COLMAP performed well with the empty stadium, with lower success rate on the Semi-crowded and Crowded datasets showing a higher mean reprojection error. In these cases, COLMAP was able to produce a localization result (shown by high values for '# Reported' metric) with the failure cases not quite meeting the success threshold of 1.5% image width. This is likely related to this method using an internal heuristic for estimating the focal length for the images, rather than relying on a prior calibration.

Overall we found that the state-of-the-art approaches are able to localize within these prior-captured SfM environments even with the addition of dynamic elements, with good robustness.

7 Conclusion

In this paper, we presented our work on globally localized tracking of stationary users for AR. We proposed an overall pipeline that integrates localization from pre-computed SfM models with spherical SLAM-based tracking. We compared our localization and tracking method for stationary users to state-of-the-art ORB-SLAM2 in two large, open environments. Furthermore, we investigated the feasibility of different state-of-the-art localization method for usage in large sports stadium environments.

The results from comparing our system to ORB-SLAM2 suggest that our approach can lead to more robust tracking, particularly in very large spaces such as sports stadia. Through our synthetic testing, we saw that our approach also has the potential to produce more accurate pose results when the true motion is spherical. We also demonstrated the feasibility of using our approach in AR scenarios through implementation of an AR renderer to visualize the registration of the AR content.

In our evaluation of state-of-the-art localization approaches, we found that the active search approach [43] performed well, as did ESAC [31]. We conclude that ESAC may be better suited to server-based localization with GPUs, whereas the active search approach has potential to be applied to localization on mobile device hardware. BoW may be a better choice for on-device computations for venues where the capture of large amount of images is not possible and there is not too much variations in conditions between the captured dataset and the testing conditions.

The main limitation of our approach is its restriction to spherical and stationary movements. The point of our system is to alleviate the tracking issues that arise in these stationary scenarios. While we focus on sports spectating in this paper, there is a typical usage pattern in AR where users are often stationary when using an AR application [7]. Other application scenarios include an audience in a lecture hall and even tourists that use their mobile phones to access information while exploring a single location. However, in all these scenarios it is possible that users may perform stationary motion initially, and switch to general motion later (such as a spectator leaving their seat, for example). However, a more sophisticated error analysis could automatically detect this and switch to more traditional SLAM tracking when sufficiently translational movement is detected. Additionally, as our solution is targeting mobile devices, future work could investigate making use of internal motion sensors readily available to improve the tracking results. Another limitation of our approach is the robustness of tracking after a relocalization has taken place; future work could investigate how to acquire more correspondences to the existing model to improve this.

Acknowledgements We thank Animation Research Ltd, Forsyth Barr Stadium, the Highlanders, Otago Rugby (ORFU) and OptaPerform for their support. We also thank Mike Denham and Craig Tidey from the School of Surveying at the University of Otago for their support in surveying the stadium.

Funding This project is supported by an MBIE Endeavour Smart Ideas grant (UOOX1705) and NSF Award 2144822.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Klein, G., Murray, D., Parallel tracking and mapping for small AR workspaces. In: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225–234. IEEE (2007)
2. Liu, H., Zhang, G., Bao, H., Robust keyframe-based monocular slam for augmented reality, in,: IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 1–10. IEEE (2016)
3. Wagner, D., Mulloni, A., Langlotz, T., Schmalstieg, D., Real-time panoramic mapping and tracking on mobile phones. In: IEEE Virtual Reality Conference (VR), pp. 211–218. IEEE (2010)
4. Langlotz, T., Degendorfer, C., Mulloni, A., Schall, G., Reitmayr, G., Schmalstieg, D.: Robust detection and tracking of annotations for outdoor augmented reality browsing. Comput. Graph. **35**(4), 831–840 (2011)
5. DiVerdi, S., Wither, J., Hollerer, T.: Envisor: online environment map construction for mixed reality. In: IEEE Virtual Reality Conference, pp. 19–26 (2008)
6. Zollmann, S., Langlotz, T., Loos, M., Lo, W.H., Baker, L.: ARSpectator: exploring augmented reality for sport events. In: SIGGRAPH Asia. Technical Briefs, pp. 75–78 (2019)
7. Grubert, J., Langlotz, T., Grasset, R.: Augmented Reality Browser Survey. Graz University of Technology, Tech. Rep. (2012)
8. Langlotz, T., Nguyen, T., Schmalstieg, D., Grasset, R.: Next-generation augmented reality browsers: rich, seamless, and adaptive. Proc. IEEE **102**(2), 155–169 (2014)
9. Langlotz, T., Wagner, D., Mulloni, A., Schmalstieg, D.: Online creation of panoramic augmented reality annotations on mobile phones. IEEE Pervasive Comput. **11**(2), 56–63 (2010)
10. Langlotz, T., Zingerle, M., Grasset, R., Kaufmann, H., Reitmayr G.: Ar record&replay: situated compositing of video content in mobile augmented reality. In: Proceedings of the 24th Australian Computer-Human Interaction Conference, pp. 318–326 (2012)
11. Ventura J.: Structure from motion on a sphere. In: European Conference on Computer Vision, pp. 53–68. Springer (2016)
12. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)
13. Baker, L., Ventura, J., Zollmann, S., Mills, S., Langlotz, T.: SPLAT: spherical localization and tracking in large spaces. In: 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), pp. 809–817. IEEE (2020)
14. Sweeney, C., Holynski, A., Curless, B., Seitz, S.M.: Structure from motion for panorama-style videos. arXiv:1906.03539 (2019)
15. Baker, L., Mills, S., Zollmann, S., Ventura, J.: CasualStereo: casual capture of stereo panoramas with spherical structure-from-motion. In: 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), pp. 782–790. IEEE (2020)
16. Ventura, J., Arth, C., Reitmayr, G., Schmalstieg, D.: Global localization from monocular SLAM on a mobile phone. IEEE Trans. Vis. Comput. Graph. **20**(4), 531–539 (2014)
17. Arth, C., Pirchheim, C., Ventura, J., Schmalstieg, D., Lepetit, V.: Instant outdoor localization and slam initialization from 2.5 d maps. IEEE Ann. Hist. Comput. **21**(11), 1309–1318 (2015)
18. Davison, A.J.: Real-time simultaneous localisation and mapping with a single camera. In: Proceedings Ninth IEEE International Conference on Computer Vision, vol. 2, pp. 1403–1410 (2003)
19. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: real-time single camera SLAM. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
20. Taketomi, T., Uchiyama, H., Ikeda, S.: Visual SLAM algorithms: a survey from 2010 to 2016. IPSJ Trans. Comput. Vis. Appl. **9**(1), 16 (2017)
21. Prince, S.J., Xu, K., Cheok, A.D.: Augmented reality camera tracking with homographies. IEEE Comput. Graph. Appl. **22**(6), 39–45 (2002)
22. Hadian, M., Kasaei, S.: Fast homography refinement in soccer videos. In: 2015 9th Iranian Conference on Machine Vision and Image Processing (MVIP). IEEE, pp. 185–188 (2015)
23. Billinghurst, M., Kato, H., Poupyrev, I.: The magicbook-moving seamlessly between reality and virtuality. IEEE Comput. Graph. Appl. **21**(3), 6–8 (2001)
24. Wagner, D., Langlotz, T., Schmalstieg, D.: Robust and unobtrusive marker tracking on mobile phones. In: 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE. pp. 121–124 (2008)
25. Azuma, R.: Tracking requirements for augmented reality. Commun. ACM **36**(7), 50–51 (1993)
26. Khan, N.Y., McCane, B.: Smartphone application for indoor scene localization. In: Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 201–202 (2012)
27. Schönberger, J.L., Frahm, J.-M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
28. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.-M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV) (2016)
29. David, P., DeMenthon, D., Duraiswami, R., Samet, H.: Simultaneous pose and correspondence determination using line features. In: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, vol. 2. pp. II–II, IEEE (2003)
30. Kendall, A., Grimes, M., Cipolla, R.: PoseNet: a convolutional network for real-time 6-DOF camera relocalization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2938–2946 (2015)
31. Brachmann, E., Rother, C.: Expert sample consensus applied to camera re-localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 7525–7534 (2019)
32. Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic J., et al.: Benchmarking 6dof outdoor visual localization in changing conditions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8601–8610 (2018)
33. Gálvez-López, D., Tardós, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Robot. **28**(5), 1188–1197 (2012)
34. Muñoz-Salinas, R., Medina-Carnicer, R.: UcoSLAM: simultaneous localization and mapping by fusion of keypoints and squared planar markers. Pattern Recognit. **101**, 107193 (2020)
35. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: International Conference on Computer Vision, pp. 2564–2571. IEEE (2011)
36. Bouguet, J.-Y., et al.: Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm. Intel Corp. **5**(1–10), 4 (2001)
37. Nistér, D.: Preemptive RANSAC for live structure and motion estimation. Mach. Vis. Appl. **16**(5), 321–329 (2005)
38. Sklansky, J.: Finding the convex hull of a simple polygon. Pattern Recognit. Lett. **1**(2), 79–83 (1982)
39. Engels, C. Stewénius, H., Nistér, D.: Bundle adjustment rules. Photogram. comput. vis. **2**(32) (2006)
40. Lowe, D. G.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157. IEEE (1999)
41. Gao, X.-S., Hou, X.-R., Tang, J., Cheng, H.-F.: Complete solution classification for the perspective-three-point problem. IEEE Trans. Pattern Anal. Mach. Intell. **25**(8), 930–943 (2003)

42. Li, Y., Snavely, N., Huttenlocher, D.P.: Location recognition using prioritized feature matching. In: European Conference on Computer Vision, pp. 791–804. Springer (2010)
43. Sattler, T., Leibe, B., Kobbelt, L.: Improving image-based localization by active correspondence search. In: European Conference on Computer Vision, pp. 752–765. Springer (2012)
44. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
45. Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C.: DSAC-differentiable RANSAC for camera localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6684–6692 (2017)
46. E. Brachmann and C. Rother, Learning less is more-6D camera localization via 3D surface regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4654–4662. (2018)
47. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
48. Karami, E., Prasad, S., Shehata, M.: Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. arXiv:1710.02726 (2017)
49. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP (1), vol. 2, no. 331–340, p. 2. (2009)
50. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge university press (2003) Shaftesbury Road Cambridge CB2 8EA https://www.cambridge.org/contact-us
51. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
52. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 573–580. IEEE (2012)
53. Grupp, M.: evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo, [Software; odometry benchmarking tools] (2017)
54. Gul, S., Baker, L., Boult, R., Mills, S., Zollmann, S.: Expert sample consensus applied to camera localization for AR sports spectators. In: 2021 36th International Conference on Image and Vision Computing New Zealand (IVCNZ), pp. 1–6. IEEE (2021)
55. Sattler, T., Leibe, B., Kobbelt, L.: Efficient & effective prioritized matching for large-scale image-based localization. IEEE Trans. Pattern Anal. Mach. Intell. **39**(9), 1744–1756 (2016)

Lewis Baker is a PhD graduate at the University of Otago. Lewis has been a member of the Graphics and Vision, and Human Computer Interaction labs at Otago since starting his postgraduate studies, and since then has worked on several projects within the subjects of computer vision, graphics and augmented reality. His main research interests are in tracking and localization, particularly in challenging edge cases such as stationary applications and large environments, which form the basis for his PhD topic.

Jonathan Ventura is an Associate Professor at California Polytechnic State University. He received his Ph.D. in Computer Science from the University of California, Santa Barbara, in 2012. Previously, he was an assistant professor at University of Colorado Colorado Springs and before that a postdoctoral researcher with the Institute for Computer Graphics and Vision at Graz University of Technology in Austria. His main research focus is 3D computer vision for virtual and augmented reality.

Tobias Langlotz is a Professor at the University of Otago. Tobias was previously a senior researcher at the Institute for Computer Graphics and Vision (Graz University of Technology, Austria) where he also obtained his PhD. Tobias main research interest is Vision Augmentations and Computational Glasses utilizing AR technology, spontaneous interaction for wearable AR systems and nomadic mobile telepresence solutions, where he works at the intersection of HCI, Computer Graphics, Computer Vision and Ubiquitous Computing.

Shazia Gul is currently a PhD student at the University of Otago. During her master's thesis, she conducted research on the subject of Computer Vision. Before her doctoral studies, she has developed several mobile applications and obtained an interest in the field of augmented reality and virtual reality. Her research investigates tracking and localization for augmented reality in large dynamic environments, particularly for sports spectating.

Steven Mills is an Associate Professor at the University of Otago, where he gained his PhD in 2000. Between being a student and an academic at Otago he worked in a variety of commercial research and development roles and as a lecturer at The University of Nottingham. His interests lie in computer vision, particularly 3D reconstruction from images and applications with cultural and heritage value.

Stefanie Zollmann is an Associate Professor at the University of Otago in New Zealand. Before, she worked at Animation Research Ltd on XR visualization and tracking technology for sports broadcasting. She worked as postdoctoral researcher at the Institute for Computer Graphics and Vision (Graz University of Technology) where she also obtained a PhD degree in 2013. Her main research interests are XR for sports and media, visualization techniques for augmented reality, but also include capturing for XR and immersive experiences.