SPLAT: Spherical Localization and Tracking in Large Spaces

Lewis Baker[*] University of Otago

Jonathan Ventura[†] California Polytechnic State University

Stefanie Zollmann[‡] University of Otago

Steven Mills[§] University of Otago

Tobias Langlotz[¶] University of Otago

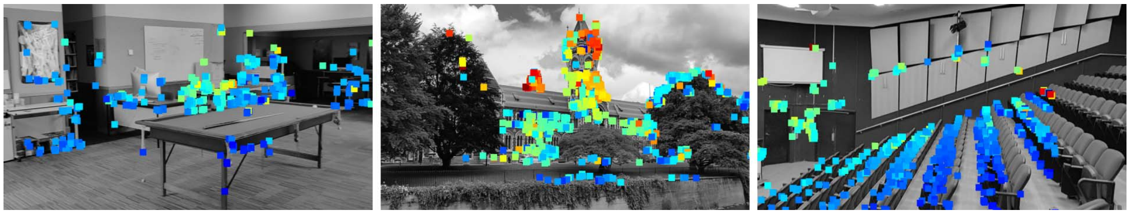

Figure 1: We propose spherical structure-from-motion for localization and tracking in large spaces such as auditoriums, or scenic outdoor locations. 3D points from the system are back-projected and color mapped by depth (blue near, red far).

Abstract

When implementing an Augmented Reality (AR) interface, it is essential to track camera motion in order to precisely register the virtual overlay in the view of the user. However, unlike most indoor AR scenarios, in many outdoor scenarios the user maintains a static position performing mostly rotational movements. Simultaneous Localization and Mapping (SLAM) methods typically used to solve the tracking problem require significant translational camera motion to perform reliably. The magnitude of the required translation is proportional to the size of the scene, exacerbating this problem in large environments such as open places or stadiums. In this paper, we present an alternative SLAM method, which combines spherical Structure-from-Motion and a robust 3D tracking method. We compare our method to ORB_SLAM2 in synthetic and real tests, and show that our method can track more reliably in large spaces, with simpler calculation due to the spherical motion constraint. We discuss this issue in the context of implementing an AR interface for live sport events in stadiums or other open environments, but possible application scenarios for our technique go beyond and can be applied to handheld AR in many outdoor environments.

Index Terms: Computing methodologies—Tracking—; Human-centered computing—Mixed / augmented reality

1 Introduction

Driven by the availability of programming frameworks such as Vuforia, ARKit, and ARCore, we have seen a wider adoption of handheld and mobile Augmented Reality (AR) where mobile devices, typically mobile phones, provide an AR interface to the user. Most applications focus on small indoor environments where digital objects are integrated into our physical environment. Examples include digital enhancements of physical books or magazines, augmentation of board games, Augmented Reality games, or previewing digital furniture augmented within our physical environments. While these environments can be handled well by most AR frameworks, large and open environments still pose challenges and are consequently less often targeted.

Key among these challenges is the precise tracking of a mobile device's location and orientation, which is a necessity for AR. When in outdoor environments or large open areas (e.g. stadiums, large spaces) users often have a fixed position, they are standing in place or even seated, observing the scene or event from a single location. Being fixed in one position means that there is insufficient parallax for conventional Simultaneous Localization and Mapping (SLAM) approaches. One approach is to use panoramic trackers [8, 34] to address these issues. However the motion of the user's device is not purely rotational either, which is problematic for panoramic trackers, which assume a perfect rotation centered around the camera.

We experienced these issues first hand when investigating using an AR interface in a stadium environment [36] with the goal of providing live sports spectators rich real-time visualizations and overlays as they are used in sports broadcasting but this time using an AR interface on a mobile device. However, being in one position is not limited to the scenario where spectators are watching a game or event within a stadium environment. An example where similar usage patterns have been observed are many outdoor AR applications utilizing mobile phones, in particular AR browsers on handheld devices, where users have the tendency to stand in one position and then rotate their device to explore the surroundings [18–20].

In this work, we address the problem of reliably tracking a camera in the situation where users mainly perform rotational movements. We observe that for a static user, their device is generally held at arm's length, so moves on a roughly spherical surface. However, instead of assuming a pure rotation around the camera center (homography-based tracking) or unconstrained movement with a large enough baseline (SLAM), we propose SPLAT, a SLAM system based on spherical Structure-from-Motion (SfM) and 2D tracking [33], paired with a new spherical 3D absolute pose solution. Our system is implemented as a keyframe based tracking and mapping pipeline using ORB features, similar to ORB_SLAM2 [24].

We show that this approach is more robust to restricted parallax through experiments on synthetic data (where ground-truth motion is available). We also demonstrate its effectiveness on real video sequences captured in several large spaces, including a stadium environment during a live sports event.

Our tracker is based on the work of Ventura [33] and extended in the following ways:

- We integrate the SfM methods into a complete real-time simultaneous localization and tracking system suitable for use in Augmented Reality
- We present a method for computing *absolute pose* with a spherical constraint

[*] e-mail: bakelew@gmail.com

[†] e-mail: jventu09@calpoly.edu

[‡] e-mail: stefanie@cs.otago.ac.nz

[§] e-mail: steven@cs.otago.ac.nz

[¶] e-mail: tobias.langlotz@otago.ac.nz

2642-5254/20/$31.00 ©2020 IEEE
DOI 10.1109/VR46266.2020.00006

- We evaluate the performance of our spherical tracking system in various synthetic and real environments.

Our implementation is built from scratch, and loosely based on ORB_SLAM2 [24] with the following contributions:

- We apply spherical constraint to the initialization, tracking, and bundle adjustment

- We present a unique *Keyframe Sphere* data structure which replaces the need for a keyframe culling mechanism.

2 Background

The problem of determining a camera's pose within an environment (localization) and updating it over time (tracking) has been studied for a long time. In the context of AR, the dominant approaches are Simultaneous Localization and Mapping (SLAM) and panoramic tracking. We review these two approaches, highlighting their limitations for the task of localization and tracking in large environments with limited (but not negligible) translational movement.

We focus primarily on visual localization and mapping with RGB cameras. Depth cameras (RGB-D sensors) are not common on mobile devices, and when present have limited range. Inertial sensors are likely to be used in practical applications, but are often combined with visual tracking and localization to overcome sensor drift.

2.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a tracking approach originating from robotics. A robot typically moves through its working environment and while moving, the robot builds up a spatial map of the environment which it uses for tracking and localization [7]. While the original approaches often used direct range sensors [7] or multiple cameras [5], later research focusing on AR scenarios showed variations of SLAM approaches that work in real-time using a single camera only (e.g. MonoSLAM [6], PTAM [17]). Since then, many approaches for monocular SLAM systems have been introduced that differ mainly in what kind of features are taken or how they are matched. Examples include the use of invariant image features (e.g. ORB_SLAM2 [23,24]), dense image registration (e.g. LSD-SLAM [9]), objects as landmarks (e.g. SLAM++ [30]), and the use of deep learning to estimate pose [16] or compact scene representations [1].

However, despite the differences in feature handling and matching (or the lack thereof [9]), most monocular visual SLAM systems rely on the same basic concept which is to have a reasonable parallax at some stages (e.g. initialization) to compute correct depth information. In our work we use ORB_SLAM2, one of the most widely used SLAM systems, to represent a general SLAM system. ORB_SLAM2 is a keyframe based SLAM system, meaning that a sparse set of frames are chosen for map update, while the others are used for pose tracking only. It is capable of tracking moving cameras in a variety of environments, and mapping the environment. The method supports loop closure when returning to previously mapped areas, bundle adjustment of the computed map, and re-localization to recover from tracking failure [23,24].

While all these SLAM systems have shown potential for tracking in various types of indoor and outdoor environments, they are unable to perform reliably in very large spaces when the user's camera motion is small. This is a significant problem for AR in large open spaces, such as a stadium, as their environment will be much larger than the typical office or home that these systems perform well in. Furthermore, live event spectators such as sport spectators will often be constrained to a fixed position or seat, moving only small distances throughout an event which causes initialization problems due to the small baselines.

2.2 Panoramic Tracking

Tracking of purely rotational movements has also been considered previously within the research community. While one can use integrated hardware sensors such as compass and gyroscopes, these are in practice not reliable enough to deliver precise information for tracking the rotation of a mobile device. One of the first works for using vision information for optical tracking was presented by Montiel and Davison [21]. Their approach of a visual compass uses an Extended Kalman Filter formulation of the general tracking problem applied to purely rotational tracking [8]. Later works such as Envisor [8] or Panoramic Mapping and Tracking [22,34,35] provided different algorithms for the same problem by building a panoramic map. The former optimizes the algorithm for using the GPU to achieve real-time performance while the latter optimizes the handling of the panoramic map and the feature matching to achieve real-time performance on a mobile device. Common to all these approaches is that they require purely rotational movements and a violation of those (e.g. when not rotating around the camera centre) will introduce errors and ultimately tracking failure.

Panoramic trackers assume that the camera's motion is purely rotational (or, less commonly in AR, that the scene is planar). Under these circumstances the images of the scene are related by homographies, and so can be mapped to a sphere (or plane) [32]. This approach has two main drawbacks for our application. The first is that while the camera movement is largely rotational, there is a translational element. This is introduced by the camera position being usually offset, and the fact the the centre of rotation is not the phones' camera centre in particular when held an arms length from the user. This translation element is restricted (i.e. to arms reach), but may be large enough that the assumption of (nearly) pure rotational motion is violated. The second is that the visualization of the event may rely on a 3D interpretation of the scene, which panoramic trackers do not provide. Rather than forming a 3D model, panoramic trackers represent the scene as directional rays from the viewer's location.

Hybrid solutions also exist that combine panorama tracking and SLAM. Pirchheim et al. [26] describe a method to track through periods of rotational motion in a general SLAM system. However, they assume that there is a sufficient translational movement at some point in the sequence to create the map. Gauglitz et al. [12] introduce a method for tracking through both general and rotational motions but do not build and localize from a global map.

2.3 Limitations for Users in Large Open Environments

Augmented Reality applications often assume that the user is in a relatively small enclosed space, such as an office or workshop [17], or moves freely through a larger environment [27]. In both of these cases, the motion of the user is a significant fraction of the size of the scene, providing a good baseline for triangulation-based reconstruction. Alternatively, the user might view distant objects from a single view point [19]. In this case the motion of the user is insignificant compared to the size of the scene and panoramic tracking is viable. We are concerned with the intermediate case – where the user's motion is small, but not negligible, compared to the distance to the scene.

This case arises in our application of Augmented Reality for sports spectators, where a large stadium environment (on the order of 100m) is combined with limited motion of the user's device (on the order of 1m). Treating the motion as pure rotation leads to an angular error on the order of 0.5°, which corresponds to tens of pixels error on a typical HD display. A similar situation arises when a seated user with a head mounted display (translations on the order of 0.1m) views a large indoor scene (on the order of 10m).

The case of a seated, or otherwise stationary user does, however, offer some advantages. The key advantage for our purposes is that the user will be likely returning to the same position multiple times, and will quickly be able to scan and observe their entire environment.

This means that some of the key advantages of typical SLAM systems, such as loop closure, and regular addition of new keyframes, become less useful. A SLAM system that is constrained to spectator type motion would need to keep track of fewer keyframes, and would rarely need to perform loop closure procedures.

Apart from the focus on our specific use-case of live spectators in a stadium, we believe the applicability of our approach goes further including many AR applications that use handheld devices in outdoor environments. As Grubert et al. showed, when using AR browsers or related applications, stationary position and rotation-only movement is the dominant movement pattern [14].

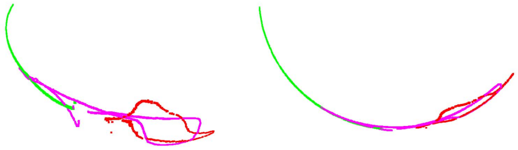

Figure 2: Different spherical movements captured by a mobile phone user in a stadium environment (color coded for different paths). Front (left) and Top view (right).

3 Approach—Spherical Simultaneous Localization and Mapping

As we have seen, existing approaches use specific constraints, parallax, or pure rotation that are often violated in the real world. In this work, we present a new approach of spherical localization and tracking that has, so far, not been widely considered. By doing so we address the problems that arise when using state-of-the art SLAM for mostly pure rotational movements. Our system represents a SLAM approach based on spherical Structure-from-Motion [33].

Contrary to existing solutions, our approach does not assume a large parallax, fixed position, or pure rotational movement. Instead, our approach uses a spherical constraint introduced by Ventura [33]. The spherical constraint relaxes the fixed position of a camera and instead allows a position which is on the surface of a sphere, and a viewing direction parallel to the sphere normal (Figure 2 shows different near-spherical user movements).

This has several advantages that we want to explore. Firstly, as already shown by Ventura, the spherical constraint allows us to use a minimal representation of the camera pose based on three rotation parameters which only requires three point correspondences to compute the relative pose between two cameras in contrast to using five correspondences in the case of unconstrained movements. More importantly, we argue that this spherical constraint is a better approximation of the user's movement when stationary in a large environment such as in our sports stadium scenario.

In his work, Ventura only investigated the spherical constraint for offline 3D reconstruction but did not implement a full tracking system or a real-time spherical SLAM approach. In the following we present our implementation of a SLAM based system using this spherical constraint while later evaluating its performance using synthetic and realistic data, and comparing it against state-of-the-art SLAM. As part of our solution, we also propose a method for computing the absolute pose with two 3D-to-2D point correspondences with a spherical constraint.

Our spherical SLAM system uses a new data structure to store keyframes (Sect. 3.1) and, as with many SLAM systems, has three distinct stages: Initialization (Sect. 3.2), Tracking (Sect. 3.3), and Mapping (Sect. 3.4). Fig. 4 depicts an overview of the whole system.

3.1 The Keyframe Sphere

We propose a new method for subdividing the space of possible camera poses which is tailored specifically to a spherically constrained keyframe SLAM system. We compute N points evenly distributed across the surface of a sphere using the method of [29]. These points act as anchors for the keyframe selection, with the goal of achieving an even distribution of keyframes. This kind of careful keyframe selection can help a SLAM system in scalability and performance, with N acting as an upper limit on the possible number of keyframes.

Through experimentation, we found good results with $N = 1000$ anchor points. We also set a threshold that requires that a new keyframe's camera centre must be within some small distance of an anchor point. We set this distance threshold to be equal to 25% of the distance between two neighboring anchors. Setting larger values here can cause issues, such as two similar keyframes being assigned to different neighboring anchor points.

3.2 Initialization

We perform an automatic initialization step before we can start with the tracking. For our application scenario this would be a valid assumption, since we can assume that users would scan part of the area with their mobile devices. Our system automatically initializes a map and begins tracking when enough spherical motion has occurred, using the process described below.

Feature extraction and tracking Our feature extraction method uses ORB features [28]. We extract 2,000 keypoints and descriptors in the initial frame. To match keypoints between successive frames, we find matches using a pyramidal KLT feature tracker [2]. At each frame, we use these 2D matches to make an estimate of the current pose relative to the initial frame.

Relative pose estimation We assume that the camera moves on a mostly circular path with a constant radius from the origin. Also, we assume that the camera viewing direction is parallel to the normal of the sphere that is given by this radius. These assumptions allow to simplify the pose estimation. The camera pose can then be described as: $[\mathbf{R}|\mathbf{t}]$ where $\mathbf{t} = [0, 0, -1]^T$, and the camera center $\mathbf{c} = -\mathbf{R}^T\mathbf{t}$. We use the spherical relative pose estimation of Ventura [33] in combination with Pre-emptive RANSAC to discard outlier tracks [25], and to determine an estimate of the current pose.

Triangulating the initial map Using the relative pose estimate from the previous step, we determine if the angular motion is great enough as determined by our *Keyframe Sphere* structure. If the poses of the start and end frame fall within the distance thresholds of two different anchor points, we proceed with initialization. We then extract features in the final initialization frame, and match them to the initial frame using the KLT tracks. Finally, we use the matches and relative pose to perform triangulation to compute the 3D points.

3.3 Tracking

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

Feature extraction and tracking The extraction of ORB features is very similar to the extraction from the initialization step. Again we extract 2,000 keypoints and descriptors in the initial frame. We use KLT feature tracking [2] to keep track of matches from the most recent reference frame, or keyframe. This simplifies feature matching in the mapping thread when new map points must be triangulated.

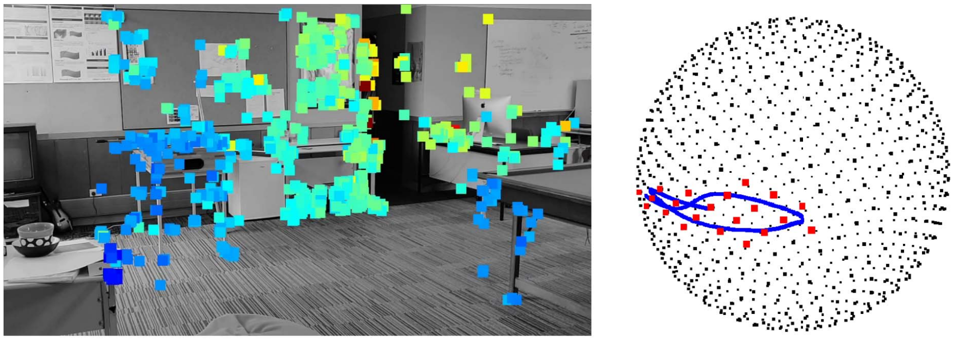

Figure 3: Example output of our system. Left: tracked features overlaid with the current input frame. Right: Tracked poses (blue) alongside, occupied (red), and unoccupied (black) keyframe anchor points.

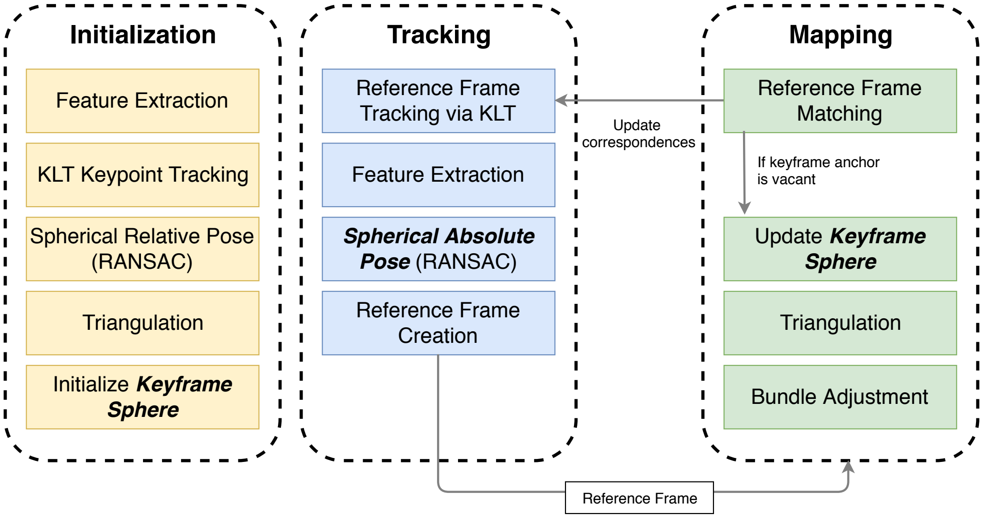

Figure 4: High level overview of our tracking and mapping system. The first component handles initializing a 3D map, and the Keyframe Sphere before handing off to the tracking component. The tracking component tracks keypoints from the last reference frame, and creates new reference frames when required by the Keyframe Sphere. The mapping thread updates the 3D-to-2D matches of the latest reference frame asynchronously, triangulates new points, and updates the Keyframe Sphere.

3D-to-2D feature matching We next use the extracted ORB features to obtain 3D-to-2D correspondences. We use the KLT tracks to maintain references to 3D map points that were found in the last reference frame. When the mapping thread finds new matches in the reference frame, the tracking thread receives these asynchronously via correspondences to the KLT tracks.

Absolute pose estimation To estimate the pose for each frame, we use a novel Perspective-2-Point (P2P) method with a spherical pose constraint within a pre-emptive RANSAC scheme to determine the current pose, and an inlier set of matches [25]. Details of this P2P solution are covered in Sect. 3.5.

Reference frames Once a frame is successfully tracked, we decide whether or not 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 merging the observations. This is equivalent in some respects to loop closure when loops are small, and drift is relatively small. Reference frames are essentially the result of keyframe pre-processing, for keyframe creation. They may or may not become keyframes as described in the following paragraph.

Keyframes A reference frame will become a keyframe if it's anchor point in the keyframe sphere is unoccupied. In this case, the KLT tracks from the previous reference frame are triangulated, and bundle adjustment takes place. Since the tracking thread is acquiring many matches through KLT 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.

3.4 Mapping

Once a frame has been tracked successfully, it is sent to the mapping thread to use the newly visible feature points in the image to update the map. The mapping thread has three main stages:

Guided matching When new keyframes are stored, the tracking thread follows features from the latest reference frame using KLT tracking as before. The mapping thread uses these results to guide the matching of keypoints between these two frames.

Updating the map The previously computed matches are then triangulated in a similar manner to the initialization phase. If more than 50 map points were triangulated from the matches, then the keyframe is considered successful, and is added to the *Keyframe Sphere* data structure as described below.

Bundle adjustment In the mapping thread, after triangulation, bundle adjustment takes place to optimize the 3D point locations, and keyframe camera poses. To enforce a spherical constraint here, we set the parameters for the camera translation to be fixed in the optimization. Once this process is complete, we remove keyframe references to outlier 3D points using the same reprojection threshold we set in our tracking thread. We then update the positions of the remaining points, and camera poses of the keyframes. We found that running a small number of iterations (one to two) 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 as used in [33] at the cost of some accuracy.

3.5 Spherical Perspective-2-Point Solution

One of the contributions of this work is a method for computing *absolute pose* under the spherical constraint. This is a key component of a tracking system which allows for computing the pose of a camera relative to a pre-computed 3D map using correspondences between the observed image points, and the 3D map point locations. This is widely known as a Perspective-*n*-Point (PnP) problem. PnP solvers are often used in SLAM systems to determine the pose of the camera relative to the map. In the unconstrained case, solutions require at least three 3D-to-2D correspondences to determine the pose [10].

Previous work by Ventura [33] defines spherical motion as having a fixed translation, causing the camera to remain on the surface of an imaginary sphere. This section covers our solution to this problem using 2 correspondences and a known translation vector $\mathbf{t}$.

The 3D points $\mathbf{X}^w = \{X_1^w, ..., X_n^w\}$, and their corresponding observations $\mathbf{x}^c = \{x_1^c, ..., x_n^c\}$ are related by a known translation $\mathbf{t}$, unknown rotation $\mathbf{R}$, and unknown scale factor λ_i:

$$\mathbf{R}X_i + \mathbf{t} = \lambda_i x_i \tag{1}$$

Since we assume that the camera motion is spherical, we can fix $\mathbf{t} = [0, 0, -1]^T$, so our objective is to compute $\mathbf{R}$. Our approach is to first compute the scale factors λ_i which can be used to unproject a homogeneous 2D observation x_i^c into a 3D point X_i^c in the camera co-ordinate system. Finally, we can find $\mathbf{R}$ by aligning the two 3D point sets using Horn's method for absolute orientation [15].

We also know that the distances between each pair of 3D points must be the same in both the world co-ordinate system, and the unprojected (scaled) camera co-ordinate system. So we can produce the following equations

$$||X_1^w - X_3^w||^2 - ||\lambda_1 x_1^c - \lambda_3 x_3^c||^2 = 0 \tag{2}$$

$$||X_2^w - X_3^w||^2 - ||\lambda_2 x_2^c - \lambda_3 x_3^c||^2 = 0 \tag{3}$$

We can also assume that our camera is facing outwards, with the camera axis parallel to the normal of the sphere. This assumption can give us one correspondence for free – that the world origin $[0, 0, 0]^T$, located behind the camera, should always project to the center of the image. So, let $X_3^w = [0, 0, 0]^T$, $x_3^c = [0, 0, 1]^T$, and $\lambda_3 = -1$.

Equations 2 and 3 can be solved for λ_1 and λ_2 respectively using the quadratic formula, resulting in four solutions. Finally, $\mathbf{R}$ is computed by applying Horn's method to the two 3D point sets $\mathbf{X}^w$ and $\mathbf{X}^c$ [15].

4 Synthetic experiments

Our initial evaluation focuses on synthetic experiments with controlled parameters in order to determine the theoretical limits of SPLAT, and the other tracking algorithms. Brückner *et. al.* suggest a framework for comparing relative pose estimation algorithms with controlled 3D point data generated in a cuboid [4]. Our experiments are similar in that we generate a 3D space of a known size and structure. However, we generate points in a 3D sphere rather than a cuboid, to have a finer level of control over the space size, and the distance between the generated points and the simulated cameras.

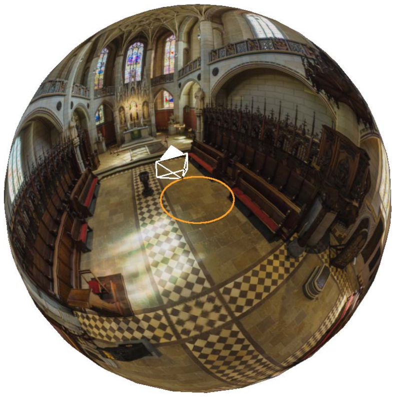

Figure 5: Example of our synthetic video setup. The world texture is projected onto a sphere with a variable radius (5 in this case), and a camera (white) moves in a circle (orange) within this sphere.

4.1 Data preparation

We take a high resolution texture and map it to a sphere within a 3D rendering software. To avoid unrealistic seams, the texture we use is a high resolution equirectangular photograph of a church[1] (Fig. 5).

Generating videos To simulate camera motion, we focus on the simple case of a camera moving in a circular arc with a fixed speed and radius. We run our experiment over 1,000 frames, (0.36° of rotation per frame), and an arc radius of 1 unit.

To determine how the tracking algorithms perform in spaces of various sizes, we fix the circular arc radius of the camera, and its movement and internal calibration parameters. We then vary the radius of the textured world sphere at regular intervals ranging from 2 to 50. At each interval, we encode a lossless 30 fps video, resulting in 49 videos which we use as input for the tracking systems.

Error evaluation Since we are interested in determining the reliability of these systems, we look to evaluate a *tracking rate*, as a measure of tracking robustness. In our experiment, we define tracking rate as the ratio of the longest sequence of successfully tracked frames with respect to the total number of frames input to the system. We used this error metric (error metric A) to analyze the suitability of three tracking algorithms.

Normalizing parameters All tracking systems we tested had certain parameters which can influence when the system would report a tracking failure. In all systems we set the number of detected ORB features to 2,000, and the minimum number of 3D-to-2D match inliers to 15. In both SPLAT and the Homography tracker, we set the inlier reprojection threshold to 5.0 pixels.

4.2 Synthetic results

In our synthetic tests, we found that all three methods handled the challenges of varying space sizes very differently. Our approach was able to reliably track in all tested space sizes. Our results are summarized in Fig. 6.

ORB_SLAM2 Our results from this experiment showed that ORB_SLAM2 was able to track reliably with small space radii, but as the space increased in size, tracking began to fail. Between radius 2 and 5, tracking was very reliable. Between 6 and 14, tracking was

[1] Original image captured by Jürgen Matern 2018, shared under the Creative Commons Attribution-Share Alike 4.0 International license.

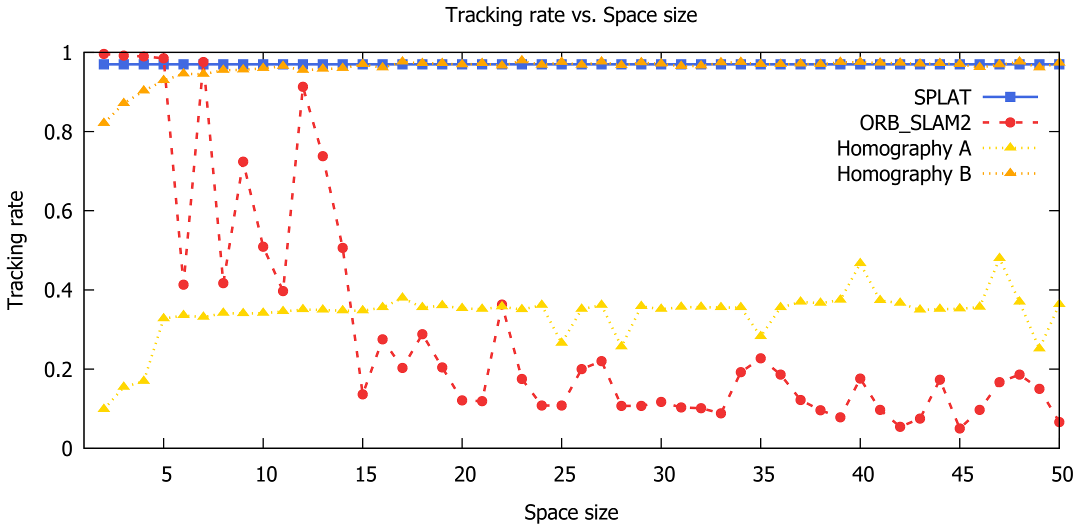

Figure 6: Tracking rate of our spherical tracker (blue), ORB_SLAM2 (red), and a homography-based tracker (with two metrics: yellow and orange) on our synthetic video sequences with different 3D sphere radii.

sometimes reliable, but would occasionally lose track. And due to the nature of the circular video sequences, there are no opportunities for the algorithm to relocalize. From a space radius of 15 and higher, ORB_SLAM2 was usually able to initialize, but would lose tracking nearly immediately, resulting in very low tracking rates.

Homography Tracking We also compared our system to a simple homography tracking system which we base on a simple panorama stitching pipeline[2]. We modified it to perform sequential matching, and use the same number of ORB features as the other algorithms. Since this method is essentially performing homography estimation between sequential frames with no propagation of existing matches (through feature tracking), it performed very poorly when using our tracking rate metric A described in Sect. 4.1 (yellow line, Fig. 6). We include another more forgiving tracking rate metric B, which is the percentage of sequential frame-pairs where homography estimation was successful (orange line, Fig. 6). In both error metrics, the homography estimation was less reliable when the space size was smaller.

SPLAT Our spherical SLAM method was able to track reliably in all test cases. We found that with a small space radius, our algorithm is somewhat sensitive to the frequency of the map updates – for example, if the keyframe sphere has too few anchor points. We chose $1,000$ anchor points, as this produced around 70 keyframes from a full circle, which is similar to ORB_SLAM2. We also found good results with as few as 250 and as many as $2,000$ anchors, with larger values producing more keyframes at the cost of performance. When tracking in small spaces, small camera movements can result in large disparity, making the movement *appear faster* as a result. We do show, however that our method is much more reliable than ORB_SLAM2 in this scenario, and that homography-based tracking can struggle with smaller space radii. In the next section, we test our method on real data captured from a mobile device, in order to determine how reliably we can track when the movement is not perfectly spherical.

5 Real data experiments

For our real life experiments, we captured a set of ten video sequences with different light conditions, and in environments of various scales, including a large sport stadium. The data was captured with a OnePlus 6 smart phone at a framerate of at least 30Hz and a resolution of 1920×1080.

5.1 Experimental method

During the capture, the mobile phone was rotated around the body of the user at a length of the user's arm. This constraint was not enforced strictly, and all sequences were captured casually from either a fixed seated or fixed standing location. In order to compare our SLAM system to state-of-the-art, we run ORB_SLAM2 in a similar way to our system, configured with the same calibration parameters and number of ORB features.

Since we have no ground truth pose or point data for the real experiments, the experiments in Sect. 5.2 are presented using a tracking rate defined by the states of either tracking or lost tracking. Both systems are configured to avoid *re-initialization* when tracking is lost, and will instead attempt to *re-localize* from the previously initialized map. To measure our pose accuracy, we compare poses output by our system to output from a RealSense T265 Tracking Camera. This device is state-of-the-art hardware, purpose built for real-time motion tracking. This is discussed in Sect. 5.3.

5.2 ORB_SLAM2 Tracking Rate Comparison

In general, the experiments confirm our initial findings from the synthetic experiments. In particular, that ORB_SLAM2 has problems tracking in large spaces. The results from the experiments are depicted in Fig. 7. We tested our method and ORB_SLAM2 on several video sequences in the manner described in Sect. 5.1.

ORB_SLAM2 performed well in large spaces some of the time. In tests Field, and Stadium-1, tracking was robust once initialized. However, in Stadium-2, tracking was lost shortly after initialization. Both of these sequences were taken from a sport spectator perspective within a live rugby game. This could suggest that tracking success is dependent on a successful initialization in large spaces.

SPLAT tracked robustly in all of the sequences, with a slightly delayed initialization in Stadium-1. A delayed initialization in our

[2] https://opencv.org/release/opencv-3-4-3/

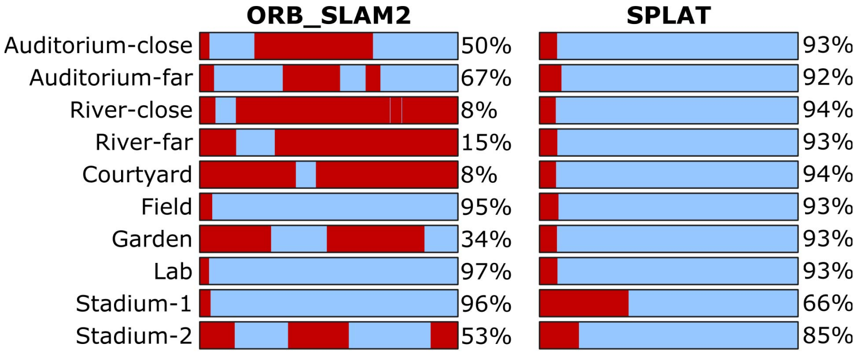

Figure 7: Tracking timelines of ORB_SLAM2 compared to our Spherical SLAM system across ten video sequences at 540p video resolution. Blue represents a successfully tracked frame, red represents an untracked frame. We show the percentage of successfully tracked frames (metric B) alongside each sequence.

system would usually mean that the triangulation resulted in some points behind the camera. These initializations can be caused by moving objects, such as other spectators, or insufficient parallax, and are discarded.

In general, we found high tracking rates with our algorithm. This is desirable for Augmented Reality applications, and would reduce the amount of work needed to be done by a re-localization module. A common use case of our method would be with full integration into a global localization module, and perhaps registering to a prior model of the environment. Such registration can be computationally expensive, so maintaining robust tracking rate is important.

In some of the very large spaces, such as Stadium-1 and Stadium-2, our method takes a little more time to initialize. But once initialized, tracking is robust. The largest space in all of our test videos was River-far, depicting a large building from across a river. This case was able to initialize quickly with SPLAT, but did not perform well with ORB_SLAM2. Notably, ORB_SLAM2 initialized more quickly on River-close, which falls in line with the synthetic results, where initialization was fast on smaller scenes.

5.3 RealSense Accuracy Comparison

To determine accuracy, and to see how well the spherical motion constraint holds true to real AR user motion, we compared our pose outputs to a RealSense T265 Tracking Camera. The RealSense tracker provides a SLAM solution integrating stereo fisheye cameras with inertial measurement. This provides us with reference trajectories independent of the video stream used to compute SPLAT and ORB_SLAM2 tracks, but cannot be considered absolute 'ground truth' as the RealSense solution is subject to drift and other tracking errors. This data was separate from Sect. 5.2, was captured from the stadium stands, and models a spectator viewing a sports event and tracking play on the pitch.

The trajectories from SPLAT and ORB_SLAM2 were aligned to the RealSense tracking path and compared using the TUM evaluation tools [31] which compute the absolute trajectory error and relative pose error. The absolute trajectory error (ATE) is the average difference between two estimated locations at each time after least-squares alignment with a rigid transform, while the relative pose error (RPE) measures the difference in estimated motion of the aligned trajectories over a short period (we use 6 frames, or 0.1s). The RPE gives values for both translational and angular errors. The RealSense reports translations in units of meters, which we adhere to in our results. These results are shown in Table 1.

The ATE values indicate that ORB_SLAM2 generally gives better alignment to the RealSense data by a small margin, but the relative pose over short periods is more accurately estimated by SPLAT, with better estimation of both translation and orientation over short time-frames (0.1 seconds). Statistical validation of these differences needs to be dealt with carefully since the relative pose errors are not independent of one another for overlapping time segments. To overcome this we take sequential non-overlapping 0.1s (6 frame) periods. Furthermore there are eight comparisons (translation and pose for each sequence), so a Bonferroni correction is applied to our significance threshold. Starting with a threshold of $p = 0.05$ yields a corrected significance threshold of $p = 0.00625$. Two-tailed paired t-tests then indicate a significant improvement in RPE for SPLAT over ORB_SLAM2 for both translation and rotation in the sequence Spectator 1 and for rotation in Spectator 4. Translation in Spectator 3 is close to the corrected threshold ($p = 0.0077$). All other cases have $p-$values larger than the uncorrected threshold.

The differences in ATE are more difficult to interpret, since the ATE between SPLAT and ORB_SLAM2 is generally smaller than the ATE between either tracker and the reference (RealSense) path. This could indicate some over-all drift in the RealSense tracking. The short-term relative poses (as used in the RPE transformation) should be more reliable due to the integration of inertial sensors in the RealSense tracking. Despite these limitations, the RealSense provides a reference for comparison that is computed independently from the SPLAT or ORB_SLAM2 trackers. From this comparison it appears that SPLAT and ORB_SLAM2 provide similar accuracy, with SPLAT providing a statistically significant improvement in relative pose estimation over short periods in some cases.

5.4 Performance

The spherical tracking method runs currently with an average of 25.1 frames per second with $2,000$ ORB features per frame, at a resolution of 960×540. With 750 features, we maintain similar tracking rates at a framerate of 35.6 frames per second. The hardware we use is a modern desktop with an Intel Core i9 processor. The method is so far not optimized for speed.

We profiled the per-frame execution time of our Spherical P2P solver and compared this to OpenCV's P3P solver (*cv::solveP3P*) [3] which is based on the method by Gao et al. [11]. We ran the same video sequence on both algorithms within the same SLAM framework, taking timings before and after the RANSAC computation. We use Pre-emptive RANSAC [25] with 500 samples, and a block size of 50. Due to the unconstrained pose from general P3P solver, the tracking produced unsatisfactory results for an AR application, while our method produced more stable pose output. Both algorithms were able to successfully track to the end of the sequence once initialized, for a total of 443 frames.

The results of a paired two-tailed t-test are shown in Table 2. Spherical P2P shows small but significant speedup ($p \leq 0.0001$). This suggests that using a spherically constrained pose solution in

		SPLAT			ORB_SLAM2		
Data Set	n	ATE (m)	Translation (m)	Angle (deg)	ATE (m)	Translation (m)	Angle (deg)
Spectator 1	1339	0.026 ± 0.017	0.0070 ± 0.0032	0.64 ± 0.35	0.024 ± 0.015	0.0075 ± 0.0034	0.67 ± 0.39
Spectator 2	808	0.016 ± 0.007	0.0076 ± 0.0040	0.68 ± 0.42	0.017 ± 0.009	0.0081 ± 0.0053	0.72 ± 0.48
Spectator 3	519	0.014 ± 0.006	0.0053 ± 0.0041	0.69 ± 0.38	0.013 ± 0.006	0.0058 ± 0.0043	0.71 ± 0.38
Spectator 4	901	0.026 ± 0.016	0.0053 ± 0.0033	1.01 ± 0.57	0.028 ± 0.016	0.0054 ± 0.0034	1.14 ± 0.62

Table 1: Absolute trajectory error (ATE) and relative pose error for SPLAT and ORB_SLAM2 compared to RealSense tracking reference poses. Relative pose errors are computed as translation and rotation differences over 6 frame (0.1s) intervals.

place of a general solution can offer performance benefits, despite our less-optimized overall solution. The mean performance improvement per frame was approximately 6ms, which could account for a reasonable framerate boost.

Method	Mean (ms)	SD	SEM	N
Spherical P2P	22.31	9.68	0.46	443
OpenCV P3P	28.36	10.76	0.51	443

Table 2: Profiling results of OpenCV's implementation for solving P3P problems and general camera motion compared to our constrained Spherical P2P solution.

6 AR Prototype

We integrated the spherical SLAM into an AR prototype. For this purpose, we use the computed poses to place the virtual content at the corresponding location within the real environment. The AR prototype uses the OpenGL for rendering and the Assimp library[3] for model loading. We implemented different prototypes, one for visualizing 3D content for stadium environments (more specifically Rugby, Figure 8, right), one for sightseeing applications visualizing labels corresponding to sights in an urban environment (Figure 8, left) as well as a guide for lecture theaters. For content placement in these AR applications, we make use of the 3D point cloud created by the SPLAT system. By selecting a 3D point in the point cloud with a mouse click, we place content at this location.

Through our AR prototype, we were able to demonstrate some important advantages in visualization that our SLAM system has over a typical homography-based tracker. In many AR scenarios, it is desirable to place virtual objects at different depths in the scene (such as small widgets near the user, and large billboards in the distance). While it is possible to render these objects with homography-based tracking, the objects would exhibit no motion parallax with respect to each other as the user moves. This kind of rendering would only be correct if the user rotated about the camera center, which is unrealistic. With our system, the objects can appear to cross over each other as the user moves, adding an extra depth cue, and resulting in a more convincing visualization. In our demo, we do not address the fact that our system operates at an arbitrary scale. However, localizing tracked frames to an existing map of known scale could provide the information required for scale correction.

7 Conclusion and Future Work

In this work, we proposed SPLAT—a SLAM system based on spherical motion assumption. We presented a method for computing spherically constrained absolute pose relative to a 3D map, and integrated this into a complete tracking solution.

Our method showed more reliable tracking rates compared to state-of-the-art ORB_SLAM2, and we compared the accuracy of both methods to reference data from a RealSense tracking camera. We found that while ORB_SLAM2 had advantages in absolute trajectory errors, our method performed more reliably in synthetic and real experiments. Our experiments showed SPLAT's ability to track reliably in spaces of a range of sizes, and we demonstrated its application through an AR rendering system. When presented with spherical motion data, a general SLAM system can be expected to operate reliably in small spaces, while a panoramic tracker would perform reliably in large spaces. However, our method can be applied to many environments ranging in scale from an office laboratory to a large sport stadium, provided the camera motion is spherical. This has many direct applications, and is useful for situations where the environment scale is unknown.

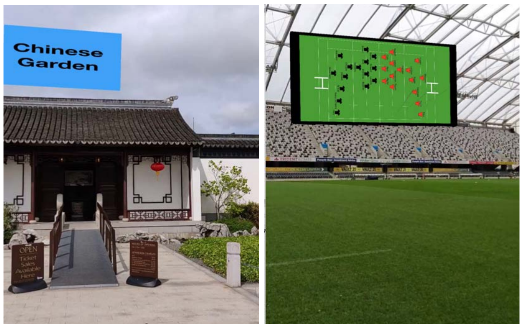

Figure 8: Application scenarios for Augmented Reality Prototype. Left) Sightseeing application. Right) Virtual display board for rugby.

For future work, we want to focus on improving performance of the system as a whole, as our initial profiling shows performance advantages in using a simpler spherical pose computation. We also think that more investigation into the benefits of our Keyframe Sphere data structure could be useful, such as using it to reduce global bundle adjustment overhead, in a similar way to ORB_SLAM2's essential keyframe graph.

There is potential for extending this system to a hybrid method (similar to [13]), combining both spherical and general motion by linking multiple Keyframe Spheres and tracking general motion between locations. A method like this would be useful in tourist applications, where users are moving from point-to-point, occasionally initiating spherical motion when they stop to look around.

Acquiring ground-truth pose data for large scale video sequences is a priority, and would allow for more extensive accuracy evaluations. It is difficult to attain this, since ground-truth must respect the spherical motion assumption. More accurate reference data could come in the form of video sequences registered to pre-computed Structure-from-Motion models, but would be subject to error.

Finally, pairing our tracking system with a global localization method would improve its usefulness in AR. In a sport stadium, for example, this would allow AR content to be easily placed in meaningful locations, and be displayed correctly for multiple spectators.

Acknowledgments

This project is supported by an MBIE Endeavour Smart Ideas grant and NSF Award #1464420. We thank Animation Research Ltd, Forsyth Barr Stadium, the Highlanders, Otago Rugby (ORFU) and OptaPerform for their support.

[3] http://www.assimp.org

REFERENCES

[1] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. CodeSLAM – learning a compact, optimisable representation for dense visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2560–2568, 2018.

[2] J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] M. Brückner, F. Bajramovic, and J. Denzler. Experimental evaluation of relative pose estimation algorithms. In *VISAPP (2)*, pp. 431–438, 2008.

[5] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):865–880, 2002.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):1052–1067, 2007.

[7] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[8] S. DiVerdi, J. Wither, and T. Hollerer. Envisor: Online environment map construction for mixed reality. In *2008 IEEE Virtual Reality Conference*, pp. 19–26, March 2008. doi: 10.1109/VR.2008.4480745

[9] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[12] S. Gauglitz, C. Lee, M. Turk, and T. Höllerer. Integrating the physical environment into mobile remote collaboration. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services - MobileHCI '12*, p. 241. ACM Press, New York, New York, USA, sep 2012. doi: 10.1145/2371574.2371610

[13] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer. Live tracking and mapping from both general and rotation-only camera motion. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 13–22. IEEE, 2012.

[14] J. Grubert, T. Langlotz, and R. Grasset. Ar browser survey. Technical report, Graz University of Technology, December 2011.

[15] B. K. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988.

[16] A. Kendall, M. Grimes, and R. Cipolla. 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.

[17] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10. IEEE Computer Society, 2007.

[18] D. Kurz, P. G. Meier, A. Plopski, and G. Klinker. An outdoor ground truth evaluation dataset for sensor-aided visual handheld camera localization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 263–264, Oct 2013. doi: 10.1109/ISMAR.2013.6671796

[19] T. Langlotz, J. Grubert, and R. Grasset. Augmented reality browsers: Essential products or only gadgets? *Communications of the ACM*, 56(11):34–36, 2013. doi: 10.1145/2527190

[20] T. Langlotz, T. Nguyen, D. Schmalstieg, and R. Grasset. Next-generation augmented reality browsers: Rich, seamless, and adaptive. *Proceedings of the IEEE*, 102(2):155–169, Feb 2014. doi: 10.1109/JPROC.2013.2294255

[21] J. M. M. Montiel and A. J. Davison. A visual compass based on slam. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1917–1922, May 2006. doi: 10.1109/ROBOT.2006.1641986

[22] J. Müller, T. Langlotz, and H. Regenbrecht. Panovc: Pervasive telepresence using mobile phones. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10. IEEE, 2016.

[23] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[24] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[25] D. Nistér. Preemptive ransac for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005.

[26] C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based SLAM. In *2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013*, 2013. doi: 10.1109/ISMAR.2013.6671783

[27] G. Reitmayr and T. Drummond. Going out: Robust model-based tracking for outdoor augmented reality. In *ISMAR*, vol. 6, pp. 109–118, 2006.

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. 2011.

[29] E. B. Saff and A. B. Kuijlaars. Distributing many points on a sphere. *The mathematical intelligencer*, 19(1):5–11, 1997.

[30] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1352–1359, 2013.

[31] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[32] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trend in Computer Graphics and Vision*, 2(1):1–104, 2007.

[33] J. Ventura. Structure from motion on a sphere. In *European Conference on Computer Vision*, pp. 53–68. Springer, 2016.

[34] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. In *2010 IEEE virtual reality conference (VR)*, pp. 211–218. IEEE, 2010.

[35] J. Young, T. Langlotz, M. Cook, S. Mills, and H. Regenbrecht. Immersive telepresence and remote collaboration using mobile and wearable devices. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1908–1918, May 2019. doi: 10.1109/TVCG.2019.2898737

[36] S. Zollmann, T. Langlotz, M. Loos, W. H. Lo, and L. Baker. Arspectator: Exploring augmented reality for sport events. In *SIGGRAPH Asia 2019 Technical Briefs*, pp. 75–78. ACM, 2019.