



Tracking and co-location of global point clouds for large-area indoor environments

Nick Michiels¹ · Lode Jorissen¹ · Jeroen Put¹ · Jori Liesenborgs¹ · Isjtar Vandebroeck² · Eric Joris² · Frank Van Reeth¹

Received: 1 September 2023 / Accepted: 21 April 2024
© The Author(s) 2024

Abstract

Extended reality (XR) experiences are on the verge of becoming widely adopted in diverse application domains. An essential part of the technology is accurate tracking and localization of the headset to create an immersive experience. A subset of the applications require perfect co-location between the real and the virtual world, where virtual objects are aligned with real-world counterparts. Current headsets support co-location for small areas, but suffer from drift when scaling up to larger ones such as buildings or factories. This paper proposes tools and solutions for this challenge by splitting up the simultaneous localization and mapping (SLAM) into separate mapping and localization stages. In the pre-processing stage, a feature map is built for the entire tracking area. A global optimizer is applied to correct the deformations caused by drift, guided by a sparse set of ground truth markers in the point cloud of a laser scan. Optionally, further refinement is applied by matching features between the ground truth keyframe images and their rendered-out SLAM estimates of the point cloud. In the second, real-time stage, the rectified feature map is used to perform localization and sensor fusion between the global tracking and the headset. The results show that the approach achieves robust co-location between the virtual and the real 3D environment for large and complex tracking environments.

Keywords Tracking · Colocation · Large-area · Point cloud · Registration

1 Introduction

Several application domains are investigating how they can incorporate co-located XR technology. Co-located setups allow users to share the physical and the virtual space simultaneously. Examples of application domains are entertainment, art and manufacturing. For entertainment and art, XR promises the seamless blending of the virtual and the real world to craft socially engaging and immersive experiences. In industry, such as the manufacturing industry, it is a powerful tool to create a realistic and immersive environment for operator training. Realistic XR experiences can be used to improve operator safety in the context of dangerous chemicals and nuclear waste, and promote well-being and inclusiveness in the search for new talent.

A big challenge for all these systems is scaling up to larger environments. Manufacturing environments, for example, range from workstations to assembly lines and all the way up to entire facilities. In co-located environments, the physical position of the user in the real world has a completely consistent mapping to the position in the virtual world. This means the user can freely navigate, turn around

✉ Nick Michiels
nick.michiels@uhasselt.be

Lode Jorissen
lode.jorissen@uhasselt.be

Jeroen Put
jeroen.put@uhasselt.be

Jori Liesenborgs
jori.liesenborgs@uhasselt.be

Isjtar Vandebroeck
isjtar@crew.brussels

Eric Joris
eric.joris@crew.brussels

Frank Van Reeth
frank.vanreeth@uhasselt.be

¹ Expertise Centre for Digital Media, Hasselt University-Flanders Make, Wetenschapspark 2, 3590 Diepenbeek, Belgium

² CREW, CREW Brussels, Vandernootstraat 23/8, Sint-Jans-Molenbeek 1080, Belgium

and retrace his steps while the virtual representation remains in sync. This is a challenging task and a mismatch can potentially lead to harmful user collisions with the physical world. Current XR technology relies on two common solutions: outside-in and inside-out tracking (Fang et al. 2023).

With outside-in tracking, the localization of the headset is facilitated by external devices. The extra sensors involved can be cameras, optical base stations or even acoustic and magnetic trackers. Although these approaches generally achieve very good accuracy, it can be costly to scale them up to work in large environments. This stems from the necessity to endow the environment with relatively expensive sensors and the fact that larger areas quickly require an ever larger amount of them to maintain full coverage.

An alternative, that has gained traction recently due to the resurgence of consumer VR, is inside-out tracking technology. It tracks the ego-motion of cameras on a headset to find its position in the environment. Inside-out tracking gives the users more freedom and increases their mobility without requiring a rapidly growing array of sensors. However, the precision of these systems is impaired by small inaccuracies that are accumulated over time, causing drift in the localization of the headset (McGill et al. 2020). When scaling up the tracking area, the drift will become too large after a while to achieve proper co-location. Some existing approaches have tried to improve this, but require an inordinate amount of markers (Podkosova et al. 2016) or impractical equipment, such as motion capturing with a set of inertial sensors positioned on the human body (Yi et al. 2023).

The main objective of this work is to tackle the drift issue that is inherent in current inside-out tracking systems. This will be a systems article, in which we describe our efforts to engineer an open and accessible system that achieves accurate global tracking and alignment, with respect to a reference global point cloud, for large-scale indoor environments. The system is cost-effective, easy to use and imposes as few restrictions on the environment as possible. In this article, the system is mainly designed and integrated for a VR environment, but we show that it is generic and could be integrated with other XR systems as well.

A key contribution of our approach is to decouple the mapping and localization stages. We work under the assumption that large parts of the scene do not change all that much. We first capture the scene with a laser scanner, which will act as a global ground truth reference frame. The global point cloud is used to calculate the transformation and drift deformations to align the initial SLAM map with it. After the registration phase, the user can be localized with high precision in the global coordinate frame of the scanned point cloud. During the VR experience, the recovered global poses are then fused with the interactive poses of the headset to mitigate the drift buildup of the headset. In contrast to other techniques, our approach only requires a sparse set of

markers during the registration phase, which can be removed for the live localization phase. For scenarios that require very precise alignment, our work proposes an optional further refinement step. This step calculates an additional drift compensation factor, which accounts for the difference in pose between the globally aligned SLAM keyframes and rendered-out versions of the global point cloud.

A nice side effect of the alignment process is that content creators can work directly in the global coordinate frame. With the increasing importance of digital twin creation, the easy authoring of content in a global reference frame will become increasingly important. Although one could argue that laser scanning is not always practical, in industries such as the manufacturing industry the use of laser-scanning is already widespread. We envision applications where content creators for XR can trust that the virtual objects they place in the world will end up in the right location, unlike current approaches that work by placing 3D objects in a warped SLAM map (e.g. spatial anchors).

2 Related work

An extensive comparison of XR-headsets and their respective tracking solutions is done by Fang et al. (2023). Many of the earlier systems are built with outside-in tracking, which means that they use external devices to track a user inside a physical room. Examples of these solutions use cameras (Furtado et al. 2019), projective beacons (Niehorster et al. 2017) or even acoustic trackers (Wang and Gollakota 2019). These systems can provide accurate tracking but are expensive and tedious to configure when scaling up to large environments.

Another category of approaches aligns the camera images with a point cloud of the underlying scene for localization purposes (Feng et al. 2019; Li and Lee 2021; Ren et al. 2023). These approaches use deep learning to learn descriptors for matching correspondences between the images and the point cloud. They solve a variant of the Perspective-n-Point (PNP) problem to estimate the pose of each image, combined with RANSAC to reduce the impact from outliers. Most of these approaches provide good accuracy, but do not run at real time frame rates.

Recently, there has been a migration towards solutions that rely solely on the integrated sensors without the need for an external camera setup. This push was mostly driven by the advent of affordable consumer VR headsets, such as the Oculus Quest. Other examples of devices with inside-out tracking are the Microsoft HoloLens and the Magic Leap. Some further works of interest demonstrate co-location for SLAM-tracked VR headsets with hand tracking (Reimer et al. 2021) and motion capturing with a combination of inertial sensors and a camera (Yi et al. 2023). They obtain

3D positional data by implementing real-time room mapping and positional tracking that is based on visual-inertial simultaneous localization and mapping (viSLAM) techniques. SLAM techniques (Durrant-Whyte et al. 1996) try to simultaneously find out the location of a user in an unknown 3D space, while also trying to build up an internal map or representation of that environment. Constructing these internal representations inherently suffers from limited precision of computing devices and sensor noise. As a consequence, whenever the user retraces his steps to an area he has seen before, there can be a gap between the information recorded in the database and the newly observed situation in the real world. This problem is called drift in the literature and it accumulates over time (Bar-Shalom et al. 2001). This means that certain objects will, over time, move away from their true positions by a considerable amount, i.e. one or several meters.

Systems based on SLAM can be integrated in a cheap, low latency, untethered and portable device and provide sufficiently accurate tracking for small environments such as living rooms. However, drift problems prevent them from scaling well to larger environments. To give an example demonstrating the difficulties involved, McGill et al. (2020) showed that in a small environment of 6 m², the mean positional accuracy of the Oculus Quest is about ± 0.04 m, with a considerable drift in accuracy over time. When scaling to larger environments this drift will accumulate considerably, deteriorating the co-location with the real world. An additional problem for practitioners, is that tracking implementations on proprietary HMDs are closed-source black-box solutions that make it hard to integrate them with other software.

Antilatency¹ is an off-the-shelf tracking system that claims to be scalable to large environments. However, the setup of the tiled floor is tedious and is not straightforward for non-squarely shaped environments. Because the system is based on active technology within the tiles, the cost of the system drastically increases when scaling up.

The aforementioned tracking methods rely on natural image features to identify landmarks in the environment to retrieve a known calibrated position within the tracking area. When natural image features are not a strict requirement for the application at hand, a set of artificial features can be introduced consisting of bar codes, QR codes, ArUco (Garrido-Jurado et al. 2014) or alternatively designed custom markers (Jorissen et al. 2014). Maesen et al. (2013) and Podkosova et al. (2016) have laid the groundwork for this approach with their fundamental research on large-area tracking, but evaluation has only been performed in very constrained environments. A main drawback is that

they rely on a dense amount of markers to achieve global localization, which is impractical in most relevant industrial environments.

SLAM methods can be categorized into smoothing, filtering and keyframe methods (Kazerouni et al. 2022). Keyframe methods are the most efficient for a given computational budget (Strasdat et al. 2010). ORB-SLAM (Campos et al. 2021) is a keyframe-based SLAM implementation with support for inertial integration, making it robust even in situations where tracking would normally be lost. It has support for mono, stereo and RGBD configurations, including fish-eye lenses. It uses natural image features to track corresponding points between images to build its internal representation of the environment.

Our work will build further on the strengths of multiple methods. We rely on the flexibility and efficiency of ORB-SLAM for localization, but improve the global accuracy of the map building by exploiting ground truth information of the 3D laser scan, without the need of complex deep-learning based descriptors. This approach can drastically reduce or even entirely remove the amount of required markers during tracking.

3 Motivation and overview

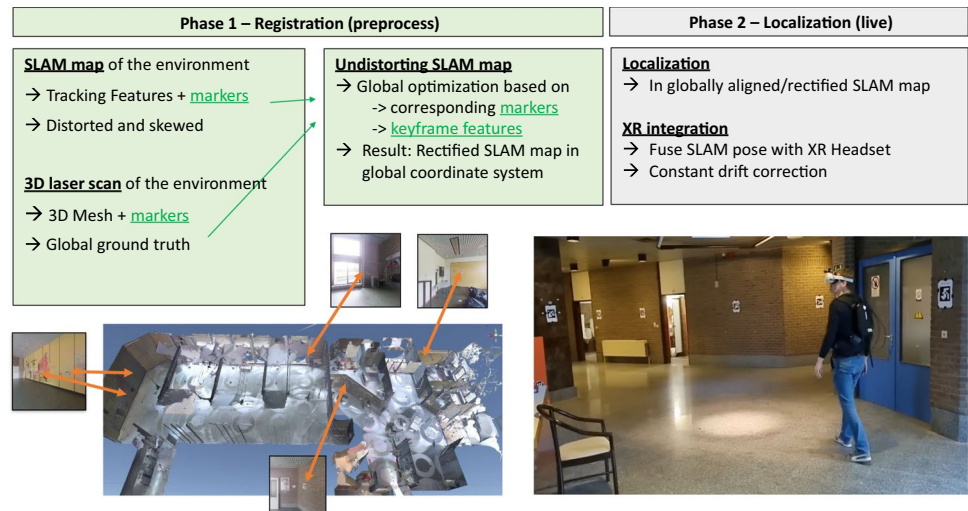
Our approach contains two stages, that are illustrated in Fig. 1. In the pre-processing stage, a feature map is constructed for the entire surroundings by capturing it with a conventional SLAM setup. One can argue that, for a subset of application domains, large parts of the environment do not often change. This motivates us to put a sparse set of ground truth markers in the environment, so that an optimization step can be performed on the feature map to globally align it with the real world, compensating the deformations caused by drift. The ground truth poses for the markers are captured using a laser scan of the environment. We use the Leica BLK360 and RTC360 scanners for this. A nice advantage of the laser scan is its certified accuracy, which makes its acceptance easier in some sectors such as industrial applications. Furthermore, laser scanning environments are already extensively used for XR content in the context of digital twin creation and authoring. Note that only a sparse number of markers are required and that they can be removed after the pre-processing stage. While we could technically do entirely without the markers and rely solely on the point cloud, we find they make it easier to obtain a first guess for the alignment process. Using the information from the laser scan, the drift error in the SLAM map is corrected and the resulting output is an undistorted map of the scene, which is aligned with global frame of the point cloud.

The second stage is the actual live VR experience, where the built-up feature map from the previous stage is used to

¹ <http://antilatency.com>.

Fig. 1 Schematic overview.

Phase 1: Global registration and rectification of the SLAM tracking map with the point cloud environment based on marker and keyframe correspondences. Phase 2: Real-time localization within the globally aligned map and integration of the global pose with the XR headset for drift correction



obtain an accurate global position for localization. Here, the user wears a consumer VR headset with inside-out tracking and walks around in a large environment. Current VR headsets are plug and play, allowing them to directly start tracking in an unknown environment. With their integrated IMU, they are also quite adept at tracking rapid or abrupt local movements. On the other hand, they do not have prior knowledge about the environment and are therefore sensitive to drift. We therefore constantly filter the real-time pose of the XR headset using a sensor fusion approach, where the drift is regularly corrected by the global pose obtained in the undistorted map of the previous stage. In the actual implementation we choose not to filter the pose directly, but instead the drift. The reason for this is that this quantity changes more slowly (see Sect. 5.2). The procedure results in a system that can achieve robust co-location for large indoor environments.

The following sections will go over the details of each implementation step in detail.

4 Global registration phase

The goal of this pre-processing stage is to align the SLAM map, required for localization, with the layout of a real environment. This will allow for easier content creation and authoring of the virtual environment, directly in the coordinate system of the real world.

4.1 Acquisition of the initial map

To get an initial estimate of our environment, we step through it with an Intel Realsense D455 camera module and run a readily available SLAM algorithm. We build further on to the ORB-SLAM3 open-source SLAM implementation. Although this type of SLAM tracking is commonly used in

commercially available XR headsets with inside-out tracking, these are closed-source implementations that do not grant access to the cameras or the internal SLAM map. We use it in its RGBD mode, on a Zotac VR GO 4.0 backpack PC. Using ORB-SLAM, both construction of the map and subsequent localization remained efficient for large maps.

4.2 Marker detection

To link the initial map to the real world, the detected positions of the markers should also be stored in the map. Unfortunately, an important drawback of ORB-SLAM3 is the absence of marker detection. We therefore extended the framework to include efficient and interactive ArUco marker detection. The detected marker information is stored alongside the standard information in the keyframe data of the SLAM map. Figure 2 demonstrates this integration. The process of marker detection only needs to be carried out during the pre-processing phase. Afterwards, the markers can be removed again.

4.3 Acquiring laser scan and global reference markers

We use Leica BLK360 and RTC360 laser scanners to acquire dense 3D point clouds. An example of a laser-scanned point cloud is shown in Fig. 3. The point cloud is captured from a set of static scans, denoted with red dots in Fig. 3, which are automatically registered by the scanner software using an iterative closest point algorithm (ICP) (Besl and McKay 1992). The accuracy of registration is measured using bundle error, a metric that quantifies the average deviation of all overlapping matching points of all point clouds after the registration process. This metric depends on the device used, the percentage of overlap between scans, and the distribution of scan positions. Typically, when using a calibrated

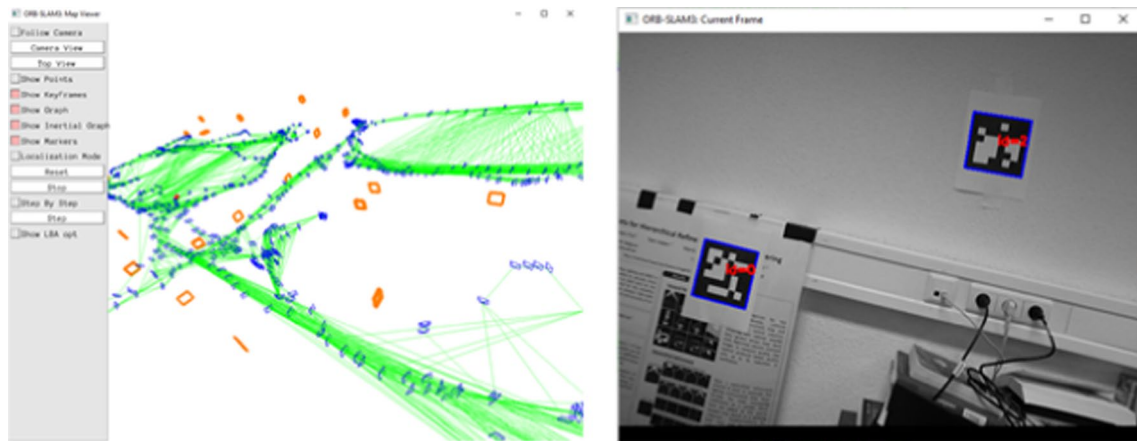
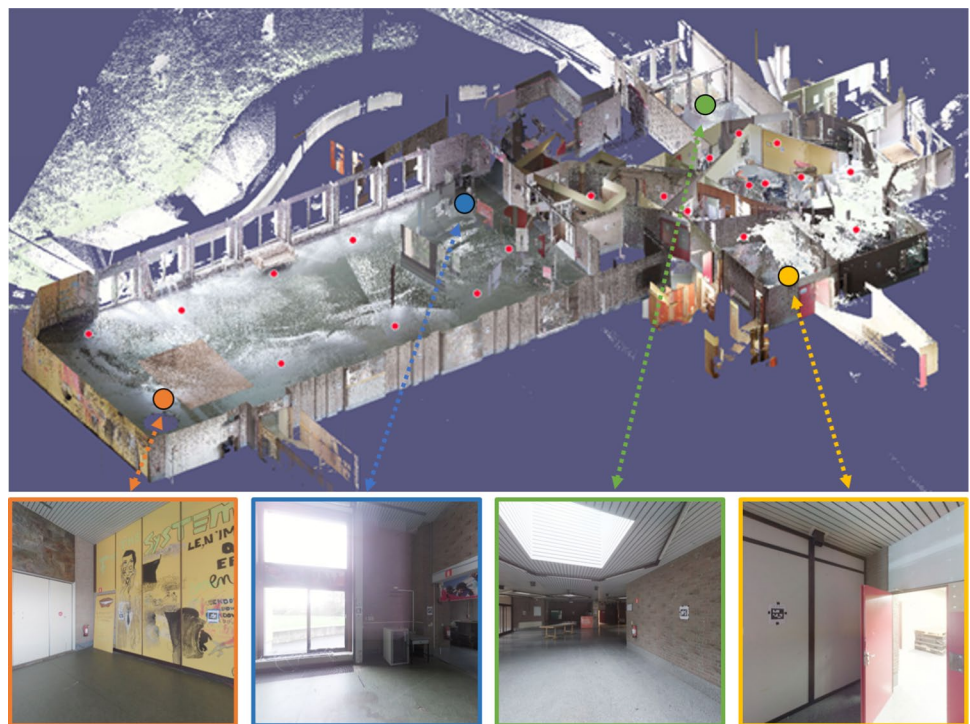


Fig. 2 We have integrated the ArUco marker detection directly in the ORB-SLAM3 framework. Here, the orange markers are the positions of the markers in the SLAM map

Fig. 3 Illustration of a large point cloud acquired by the laser scans. Top: reconstructed point cloud using laser scanning. The red dots illustrate the positions of the laser scanner. Each acquired position contains a high-resolution panoramic image as well; Bottom: examples, in the form of cube maps, of the high-resolution panoramic image for four different scanner poses. The colored dots show their respective poses in the point cloud. Our system estimates the 6DOF reference poses of the markers, as detected in the various cube maps, with respect to the scanned point cloud



scanner of this type and following the recommended capturing procedure, this metric of registration is within the range of millimeters or lower. We leverage this point cloud in the pre-processing stage to obtain the global pose of the reference markers. The 3D laser scan of the environment is stored in the open E57 vendor-neutral point cloud format and is treated as a global ground truth in the global coordinate system. Such global reference markers are required to be able to align the tracking and localization map from the previous section with the real-world global environment needed for the VR experience. These global reference markers serve as

a point of correspondence between the acquired tracking and localization map coming from SLAM and the real world. The proposed solution is based on ArUco markers, as they can be tracked at real-time rates when integrated into the SLAM algorithm.

To acquire the global ground truth positions of the ArUco markers, we have integrated tools to detect them in the 3D scan. This is done by exporting the cube map images of the omnidirectional images coming from the laser scanner, which are also stored in the E57 point cloud format. Cube maps (Greene 1986) are a type of texture mapping that uses

six square textures to represent the surfaces of a cube, providing a way to efficiently encode omnidirectional data. In our implementation, each cube map image has a resolution of 2048×2048 pixels. The ArUco marker detection algorithm then operates on the individual cube map images. For each of these images, the E57 file contains a corresponding global 6DOF pose (rotation and translation) in the global point cloud coordinate system. These poses are illustrated with the red dots in Fig. 3. As can be seen, the ArUco markers are visible in these cube map images and as such the pose of that cube map image with respect to the ArUco marker can be calculated. This marker pose P_{m2c} , in combination with the global pose P_{g2c} of the camera with respect to the global point cloud, can be used to extract the 3D positions of the marker corners. Here, m2c stands for marker to camera, whereas g2c stands for the camera pose with respect to the global point cloud coordinate system.

The four 3D positions $[X, Y, Z]$ of the corners in the marker coordinate system are related to the 2D projection of the corners $[x, y]$ in the image of the cube map by the following formula:

$$\lambda \cdot [x, y, 1] = K \times P_{m2c} \times [X, Y, Z, 1]_{marker} \quad (1)$$

Here, P contains the intrinsic parameters of the camera, which in this case is the projection matrix of the cube map image that contains the focal length and the center of projection. This projection matrix P represents the transformation of 3D points in the camera coordinate system to projected 2D points in the image of the camera. λ is the arbitrary scaling after perspective transformation. The 2D corner positions are obtained by the ArUco marker detection algorithm. The 3D corners (also called object points) are defined in the local coordinate system of the object. The object points are in this case the 3D corners of the marker and for clarity reasons this will be denoted as the marker frame or marker coordinate system. As the dimensions of the real markers are known, the marker object points are known as well. In our case, the marker size is 16.75 cm. The four 3D corners in the marker frame are $[0, 0, 0]$, $[16.75, 0, 0]$, $[16.75, 16.75, 0]$ and $[0, 16.75, 0]$. P_{m2g} is the unknown 4×4 transformation matrix that needs to be calculated. It represents the transformation of 3D points in the marker frame to 3D points in the global frame. The set of four 3D-2D correspondences of the corners is used to solve a Perspective-n-Point problem and to retrieve the P_{m2c} pose of the marker with respect to the camera. The reprojection error is used to evaluate the quality of the recovered pose and filter out bad poses, typically expected to be below one pixel.

To retrieve the actual global 3D positions of the markers in the point cloud, an additional transformation of the camera to the global coordinate system is required. Now, all parameters except the global 3D positions of the four marker

corners are unknown. These can be calculated by inverting the equation above and deprojecting the 2D corners to 3D. Or more straightforward, by transforming the local marker 3D positions first to the camera frame and then subsequently into the global frame:

$$[X, Y, Z]_{global} = P_{c2g} \times P_{m2c} \times [X, Y, Z]_{marker} \quad (2)$$

Here, the pose of the camera in the global coordinate system is defined by P_{c2g} , which is the pose of each individual cube map of the scanner. By applying this to all the cube map images of the scanner for all the different ArUco markers, the outputs are the 3D positions of each marker aligned with the global point cloud. These will serve as ground truth for the next steps of the global tracking system. The accuracy of the acquired 3D marker positions depends on the registered pose of scans, the resolution of the cube map images, and the 2D detections of the ArUco markers. The first two factors are defined by the laser scan device software, as discussed above. The latter is determined by the ArUco detection algorithm, which is generally pixel precise, provided the marker is sufficiently close to the scan positions, typically not more than 3 ms away. Furthermore, we filter out markers that are too far away. Assuming that the detection does not deviate by more than one pixel, we can calculate the maximum displacement of the reconstructed 3D marker corner between two pixels at a range of 3 ms. The offset between the deprojected 3D points of two neighboring pixels, given a resolution of 2048×2048 for a cube map tile at a range of 3 ms, is approximately 2.3 mms.

4.4 Global registration

This step will use the acquired global reference markers of the previous section and apply a full global optimization to mitigate the drift. First, an initial registration step is applied to the SLAM map to roughly align the SLAM feature map with the global point cloud. This step is necessary to provide a guess to the optimization algorithm, to ensure convergence to a decent solution. Second, the global registration is further refined using a global optimization step. Both steps will be detailed further in the next two paragraphs.

Because the ArUco markers are detected in both the global point cloud and the SLAM map, a sparse set of 3D correspondences are known. Hence, using the known 3D correspondences, their respective rotation and translation can be derived. Suppose that these two point sets $\{p_i\}$ and $\{p'_i\}$; $i = 1, 2, \dots, N$ are related by

$$p'_i = R \times p_i + T + N'_i \quad (3)$$

where R is a rotation matrix, T a translation vector and N_i a noise vector. This transformation can be calculated using least-squares fitting by applying singular value

decomposition (SVD) as shown by Arun et al. (1987). The idea is to first center both point clouds around the mass by calculating the mean and minimize their relation above. This can be done by calculating the 3×3 covariance matrix and decomposing it into a rotation matrix using SVD. The result is a roughly aligned SLAM map. However, a key aspect here is that they are only roughly aligned, as the inherent drift in the SLAM map keeps it from aligning perfectly.

Next, the registration is refined by fully optimizing the entire set of SLAM map properties. Fortunately, the rough alignment of the previous paragraph is sufficient to start a full global optimization process based on Sparse Bundle Adjustment (SBA) (Triggs et al. 2000), where all parameters are refined based on known ground truth markers. SBA is a global optimization technique that is already used in SLAM algorithms to take into account loop closing. Its goal is to minimize a global error in the SLAM map. Typically it optimizes for the keyframe poses and the 3D feature map points, based on 2D observations of the features in the 2D images. In addition, we have extended the regular approach by integrating the ground truth markers in the process. To this end, the 2D detections of the markers are also integrated into the keyframes of SLAM. Thus, for each keyframe we know its relative marker positions. On the other hand, their corresponding 3D ground truth marker positions in the global map are known as well. Finally, we incorporate all this information by inserting an extra cost to the global optimization algorithm. To calculate this cost, we use the ground truth P_{m2c} transformations (transformation of the 3D marker positions to the 2D keyframe image positions) for each known marker, as extracted in Sect. 4.3. These are used as a fixed parameter in the system. The reprojection error of the known 3D marker corners onto the 2D images is added to the cost function that needs to be minimized. Integrating this extra reprojection error and running sparse bundle adjustment will result in a refined SLAM tracking map that aligns with the global point cloud map. Concretely the cost function of SBA is extended with a second term as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{C}, \mathbf{X}, \mathbf{M}) = & \sum_{k=1}^K \sum_{i=1}^N \rho(\|\mathbf{p}_{ki} - \Pi(\mathbf{C}_k, \mathbf{X}_i)\|^2) \\ & + \sum_{k=1}^K \sum_{j=1}^O \rho(\|\mathbf{m}_{kj} - \Pi(\mathbf{C}_k, \mathbf{M}_j)\|^2) \end{aligned} \tag{4}$$

where $\mathcal{L}(\mathbf{C}, \mathbf{X}, \mathbf{M})$ is the cost function, representing the total reprojection error over all keyframe poses C , 3D map points X and ground truth 3D marker points M acquired by the laser scan, with K , N and O respectively the number of keyframes, map points and marker points. Each keyframe pose C_i includes the position and orientation of the i th camera in the world coordinate system. Traditionally a robust kernel ρ is applied to the squared reprojection error to mitigate the influence of outliers. It ensures that the cost function is not

overly sensitive to large errors, which are likely caused by incorrect correspondences. Here we use the Huber kernel with a delta parameter of $\sqrt{7.815}$ for both terms, similar to ORBSLAM-3. On top of the existing reprojection error of the 3D map points and the 2D keyframe observations, we add a second term to the cost function that calculates the reprojection error of the ground truth 3D markers corner positions M_j with respect to the 2D marker corner observations m_{kj} of each keyframe k . $\Pi(\mathbf{C}_k, \mathbf{M}_j)$ is the projected ground truth marker position M_j onto keyframe with pose C_k . This term is similar to $\Pi(\mathbf{C}_k, \mathbf{X}_i)$, which projects the 3D map points X_i onto the keyframe with pose C_k , and is compared with the 2D map point observation P_{ki} for that specific keypoint. The weight of the marker pose error is equally contributing to the cost function with respect to the 3D map points, however the total number of marker corners is less than the total number of map points per keyframe.

4.5 Further keyframe refinement

The use of markers, as explained in the previous section, already achieves a good alignment between the SLAM frame and the point cloud frame. When moving further away from the markers, however, small displacements can still occur. Optionally, if the downstream task requires more precise 2D–3D alignment, further refinement can be achieved by matching the images of the acquired SLAM keyframes and their respective poses, coming from the SLAM detections of the registered map, to the global point cloud. This is feasible by rendering out the dense point cloud to the respective keyframe poses. An example of ground truth keyframe and rendered keyframe is shown in Fig. 4. When overlaying the set of ground truth keyframes and their rendered-out counterparts, additional displacements and inaccuracies can be observed, as shown in Fig. 5 in the middle column. The image shows the keyframe image blended with the rendered equivalent. One can clearly observe ghosting artifacts due to misalignments.

Then, to quantify the misalignment, features matching is applied to both the ground truth keyframe image and the rendered counterpart, allowing us to obtain 3D-correspondences between the 2D ground truth keyframe and the 3D point cloud. For this we need the 3D point cloud positions of the rendered pixels as well, as shown in Fig. 4 to the right. The difference in terms of rotation and translation between the current keyframe pose obtained with the global registration from the previous section and the expected counterpart is estimated using Efficient Perspective-n-Point (EPnP) (Lepetit et al. 2009), in combination with RANSAC (Fischler and Bolles 1981), to get the actual ground truth global pose with respect to the point cloud. The EPnP algorithm is capable of extracting the pose of a keyframe based on the observed 3D-2D correspondences,

Fig. 4 Keyframe rendering. An example of a ground truth keyframe (left) and a rendered keyframe from the point of view of an estimated keyframe pose of SLAM (middle) and his accompanying 3D point cloud positions (right)



Fig. 5 Left: ground truth keyframe images as captured by the camera. Middle: overlay of the ground truth keyframe images with the rendered keyframe as estimated by SLAM. Misalignments are visible in the form of ghosting. The matched 2D features are visualized, which are used to quantify the displacement and to extract a ground truth keyframe pose. Right: similar overlay, but now rendered with the extracted ground truth keyframe pose, resulting in better alignment of the keyframe. Notice that the impact of moved objects between the acquisition of the 3D scan and the SLAM map keyframe is limited, as sufficient other features can be found. This is for example the case in the second example where the table has been moved



in our case observed correspondences between the global point cloud and the real target camera image:

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ 1 \end{bmatrix} = P_{cam} C_k \begin{bmatrix} X_{scsan} \\ Y_{scsan} \\ Z_{scsan} \\ 1 \end{bmatrix} \tag{5}$$

were $[X_{scsan}, Y_{scsan}, Z_{scsan}]$ is the set of 3D positions of the detected features acquired by rendering the reference 3D scan to the 2D feature points. $[x_{cam}, y_{cam}]$ is the set of projected pixels in the tracking camera. In this case we set the detected pixels as the corresponding pixels in the real

camera view, as this is the expected ground truth projected position. P_{cam} is the intrinsic projection matrix of the camera and C_k is the unknown camera pose that will be extracted.

Figure 5 shows the ground truth keyframe image as captured by the camera on the left. An overlay of the ground truth image on top of the rendered image of the estimated SLAM keyframe pose is shown in the middle. Here, the detected features and their respective offset is illustrated. On the right a similar overlay is shown, but now with a render of the corrected ground truth pose for the keyframe. One can clearly see that both are correctly aligned and there are no longer ghosting artifacts or displacements between the features.

The estimated ground truth poses of the keyframes are fed into the SBA global optimization algorithm of the previous section as an additional ground truth transformation in the system. Concretely, for each newly acquired ground truth keyframe pose, we have fixed the camera pose C_k of its corresponding keyframe k in both terms of the cost function as given in Eq. (4). As such, the global error and drift from the real world is further reduced. In the future, this type of image-to-pointcloud refinement will make it possible to remove the need for markers entirely.

5 Live global localization phase and XR integration

At this point, the SLAM map is registered and aligned with the global 3D scan of the environment, and can be used to localize the camera. When performing SLAM in localization mode only, the system returns the pose of the camera with respect to the global frame, and thus aligned with the real environment. However, although the localization is done in real time, it does not suffice for smooth tracking in a XR experience. On the one hand, there is a VR headset that achieves position updates at a rate of 90 Hz but is closed-source, proprietary and prone to drift. On the other hand, our global alignment algorithm does correct drift but can only do this at a rate of about 15–20 Hz. To get the best of both worlds, an additional sensor fusion is required. This fusion continuously applies a drift correction based on the global position updates to the VR rendering system, allowing it to achieve optimal interactive frame rates with as little drift as possible.

5.1 Camera calibration

To achieve this, an external global localization camera used for SLAM is mounted on the VR headset. Initial tests were done using an RGB Ximea sensor, which we later replaced with the Realsense RGBD sensor. The Meta Quest 2 headset was used for the VR integration. Figure 6 shows the camera mounted on the VR headset, with the help of a custom-designed 3D printed bracket. The relative transformation between the external camera and the VR headset is needed to compensate for the difference between the viewpoint of the VR user and that of the tracking camera. This calibration is achieved by exploiting the known 3D positions of the VR controllers. By capturing a set of correspondences between the 3D positions of the VR controllers and the 2D projections of the controllers onto the Realsense camera, we can solve the following system and extract the transformation matrix $cam2XR$:



Fig. 6 Our XR camera setup with a Realsense D455 strapped to an Oculus Quest 2 HMD by a custom 3D-printed mounting bracket

$$[x, y] = P_{cam} \times cam2XR^{-1} \times [X, Y, Z]_{controller} \quad (6)$$

This process is illustrated in Fig. 7.

5.2 Sensor fusion with VR headset

After calibration, the poses of the tracking camera are integrated with the VR headset for continual drift correction. The key insight is to rely on the headset pose for the sensitive head movements and on the globally aligned poses to correct drift periodically. Both poses are combined using Kalman filtering (Kalman 1960) in which the drift is assumed to be constant.

The integration and visualization of this sensor fusion are done in a native plugin for Unity. The plugin connects to the external tracking camera and runs the adapted ORB-SLAM3 algorithm in combination with the globally aligned and rectified map. As explained in Sect. 4, the ORB-SLAM3 SLAM map is acquired and globally aligned with the real environment in a preprocessing step. In this live stage, ORB-SLAM3 is tracking in the preprocessed and globally aligned feature map in localization-only mode. The plugin will return the pose of the camera in this globally aligned coordinate system. Two approaches for communicating with the SLAM plugin have been realized. The first one lets the Unity render instance request a global pose on demand, as illustrated in Fig. 8a on the right. The other one runs in broadcast mode, where the global pose is sent to any client that might need it, as illustrated in Fig. 8b. In broadcast mode the external SLAM tracking instance can also broadcast its global poses over the network. We have implemented this over the Open Sound Control (OSC) protocol (Wright and Freed 1997). For example, in broadcast mode, the Unity VR experience receives new global poses at a regular pace. For each new global pose, it keeps track of the changes compared to the previously received one. This change of global pose, in combination with the change in VR headset movement, is used to measure drift between the two systems. This problem of headset drift is shown in Fig. 9. The measured drift is then filtered using Kalman filtering, to reduce the visually displeasing effects of sudden instantaneous drift corrections.

Fig. 7 Using the VR controllers to calculate the transformation between the Realsense camera and the HMD

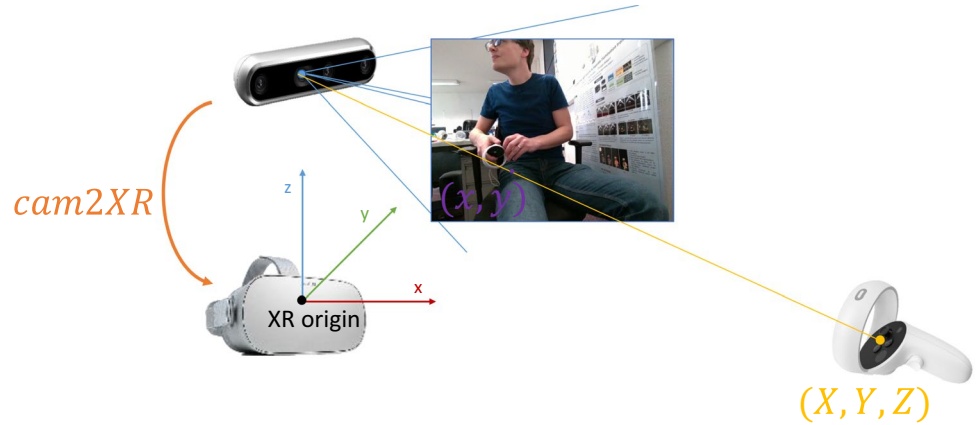
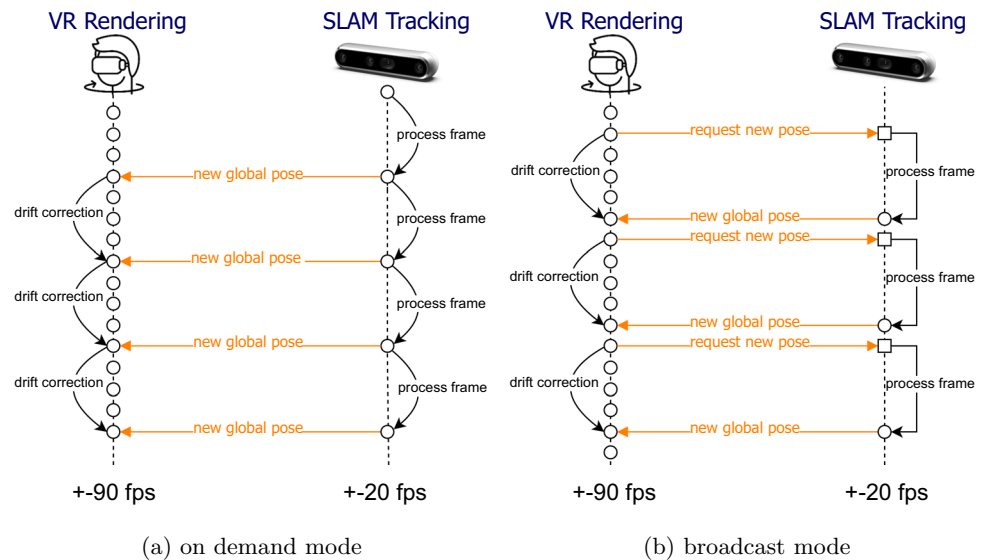


Fig. 8 Two different approaches for headset integration. **a** On-demand-mode where the headset is responsible for requesting poses from the SLAM tracking instance. **b** Broadcast mode where the SLAM tracking instance broadcasts the camera pose when ready



Instead of Kalman filtering the global pose directly, which is by nature non-linear, we opt to filter the measured drift between the two systems. This quantity evolves more slowly over time and is therefore more amenable to prediction. The drift D is calculated as follows:

$$D = pose_{global} \times pose_{XR}^{-1} \tag{7}$$

If there is no drift, the drift matrix D will stay constant. Given that the drift exhibits linear behavior over time, we apply standard Kalman filtering techniques. However, to effectively filter the drift matrix D , it must first be decomposed into a 3D translation vector and a 4D quaternion representing the rotation. Both translation and rotation measurements are filtered using Kalman filtering. To smoothly filter the evolution of the drifted rotation and translation over time, we configure both the transition and measurement

matrices as identity matrices. Optimal results were achieved by setting the process noise covariance to 0.001 and the measurement noise covariance to 0.9. The resulting filtered drift is combined with the XR headset pose to compensate for the drift and to get the final camera pose for rendering:

$$pose_{compensated} = D \times pose_{XR} \tag{8}$$

Because the XR pose is updated at a high frame rate of more than 90Hz, the newly compensated $pose_{compensated}$ will be updated at the same high frequency. On the other hand, D will be updated only when a new global tracking position is received.

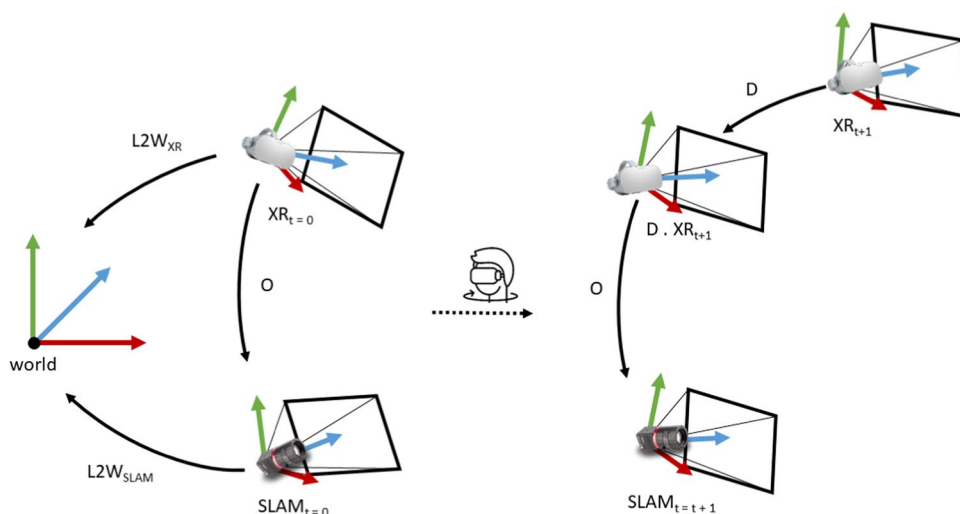


Fig. 9 Sensor fusion and drift correction. In our approach, two systems are running in parallel. First there is our proposed global localization, based on the registered SLAM map, running on the images of the RGBD camera at relatively low framerates of about 20 fps. Second, there is the headset, tracking at a framerate of about 90 fps. The former is globally accurate and aligned with the point cloud, the latter

is more interactive but does not align with the point cloud and drifts over time. At $t = 0$, an initial alignment O is performed. At $t = t + 1$, when a new global pose is retrieved by SLAM, the initial offset O does not longer suffice, as the headset could have been drifted by a factor of D . This factor is estimated, filtered and compensated for

5.3 Sensor fusion with standalone IMU system

In this approach we use the IMU sensor data of the Realsense D455 camera. The IMU data is acquired at 100 Hz, where the RGBD data together with the global pose processing runs at about 20 Hz. Based on the IMU readings, we extract a 6DOF pose, inspired by the approach of Lang et al. (2002). The IMU pose is not aligned with to global environment and is inherently susceptible to drift build-up. The key idea of this approach is very similar to the previous section. The high-frequency IMU pose is used for subtle and sensitive movements, while the lower-frequency global pose is used to align with the real world and perform a smooth drift correction. Provided that the drift buildup is constant, corrections should be imperceptible. This approach allows the adoption and integration of the tracking system for other use cases, such as augmented reality applications.

backpack PC. The reference point clouds are obtained using Leica BLK360 and Leica RTC360 laser scanners.

To evaluate the global registration and alignment, we have captured three different environments: *xrhuis*, *crew* and *makerspace*. An overview of the details and properties of these are given in Table 1. An essential objective for the downstream task is proper alignment of the 3D point cloud, or any other augmented 3D object in the coordinate system of the 3D point cloud, to the 2D view of the camera, and by extension to the point of view of the XR device. As such, a key evaluation is measuring how well the 3D world aligns with the SLAM tracking camera.

For each of the three datasets, we have registered the SLAM map to the point cloud with three types of registration which we denote ‘aligned’, ‘markers’ and ‘pointcloud’. Each of them corresponds with a different level of accuracy as detailed in Sect. 4.4. The first uses the rough alignment

6 Results and evaluation

The global registration and localization are implemented as an extension to the open source ORBSLAM-3 (Campos et al. 2021) framework. Mapping and localization are done in RGBD mode using a Realsense D455 camera, on a Lenovo Legion 5 Pro laptop (AMD Ryzen 7 5800 H and GeForce RTX 3070). We use a resolution of 640×480 pixels. For the localization phase, we implemented the sensor fusion on a Meta Quest 2 headset, tethered to a Zotac VR GO 4.0

Table 1 Properties of the three datasets used for evaluation

	xrhuis	crew	makerspace
Approximate tracking area (m ²)	160	250	312
Number of rooms/corridors	5	9	4
Laser scanner	RTC360	BLK360	BLK360
Number of points (millions)	497	205	1058
Number of keyframes	800	800	800
Number of test frames	2425	7211	4833
Cloud-to-cloud bundle error (mm)	0.8	6.8	2.5
Cloud-to-cloud overlap (%)	70	56	64

Table 2 Quantitative results of the alignment precision for three categories of alignment: rough alignment (aligned), marker-based alignment (markers) and refined alignment based on pointcloud-keyframe matching (point cloud)

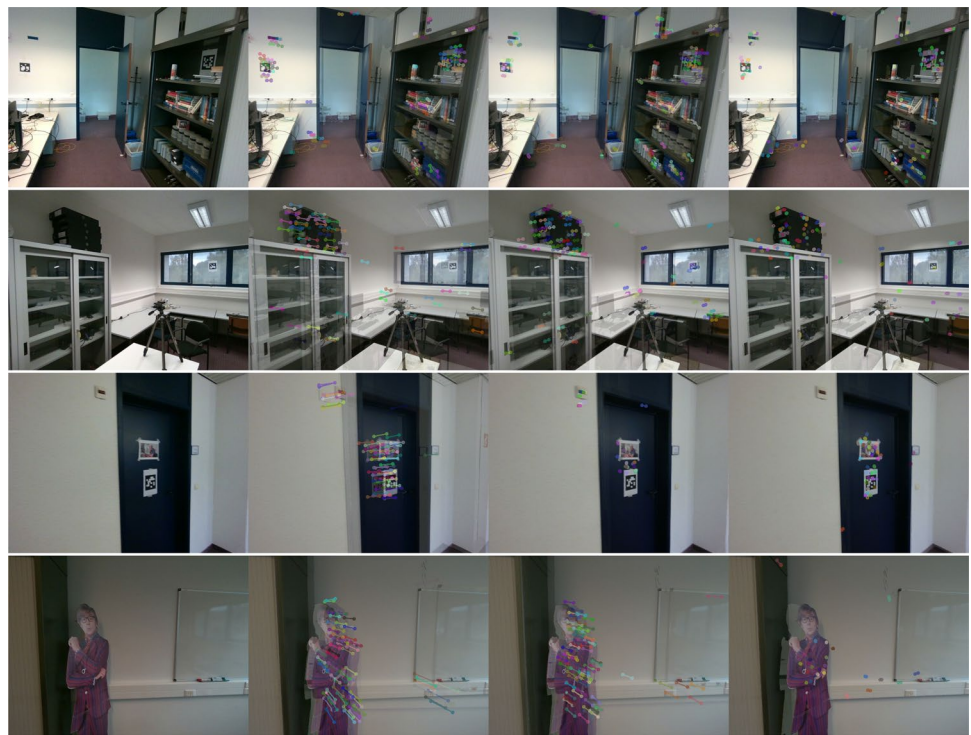
	Measurement	Type of registration											
		Aligned				Markers				Point cloud			
		μ	σ	P50	P90	μ	σ	P50	P90	μ	σ	P50	P90
Xrhuis	Pose angle error (degrees)	0.65	0.61	0.53	1.18	<u>0.51</u>	0.70	<u>0.36</u>	1.04	<u>0.62</u>	<u>0.54</u>	0.46	1.20
	Pose distance error (cm)	11.80	5.12	12.12	17.41	5.18	<u>3.76</u>	4.69	<u>8.95</u>	<u>5.03</u>	4.19	<u>4.16</u>	9.35
	2D distance error (pixels)	18.17	10.73	14.84	32.68	5.68	7.11	3.60	10.54	<u>3.47</u>	<u>2.40</u>	2.63	6.52
Crew	Pose angle error (degrees)	3.08	9.35	2.03	2.91	1.63	6.40	0.93	2.56	<u>0.92</u>	<u>3.50</u>	<u>0.55</u>	<u>1.80</u>
	Pose distance error (cm)	55.60	57.13	39.52	104.17	18.93	30.13	10.61	53.55	<u>7.45</u>	<u>23.51</u>	<u>5.59</u>	<u>12.28</u>
	2D distance error (pixels)	65.52	54.61	51.84	112.51	26.91	38.95	17.42	56.99	<u>5.88</u>	<u>11.43</u>	4.58	9.15
Makerspace	Pose angle error (degrees)	0.49	2.02	0.32	0.81	<u>0.39</u>	<u>0.52</u>	<u>0.26</u>	<u>0.74</u>	0.62	0.78	0.43	1.26
	Pose distance error (cm)	6.23	10.85	5.45	9.92	5.64	<u>3.83</u>	5.09	<u>9.56</u>	<u>5.34</u>	4.33	<u>4.15</u>	10.97
	2D distance error (pixels)	8.82	7.18	7.76	14.60	7.31	<u>5.88</u>	5.20	15.14	<u>6.52</u>	6.43	5.40	11.56

For each metric the table shows the mean, standard deviation, 50th percentile and 90th percentile

Underline denotes the best performing score per measurement per dataset

Bolt denotes the best performing score per dataset for the 2D distance error measurements, which has the most significant impact for the end-user

Fig. 10 Example results on a test sequence of the *xrhuis* dataset. For the columns from left to right: (first) ground truth keyframe image; (second) overlay of the roughly aligned frame with the ground truth; (third) overlay of the marker-based aligned frame with the ground truth; (fourth) overlay of the keyframe-refined frame with the ground truth. For each blended pair, the figure shows feature matches and their respective distances



of Eq. (3). The second is the main registration based on the ground truth reference markers, as discussed in Sect. 4.4. The last one is an additional refinement on the map based on the keyframe-to-pointcloud matching approach of Sect. 4.5.

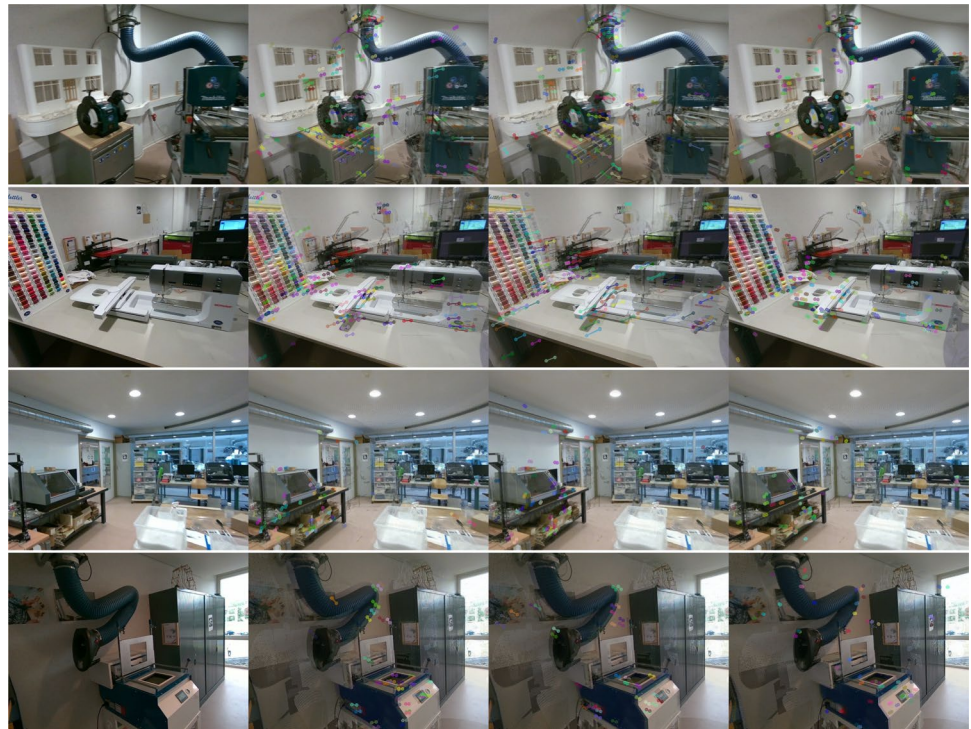
The first type of evaluation is done empirically. We have captured a test sequence for each dataset and estimated the 6DOF poses by running our adapted SLAM algorithm using the maps of the three types of registration. Next, we

implemented a renderer to project the dense point cloud onto the image planes of the estimated poses. If successful, we expect these to align and coincide to the best achievable extent. The results for a selection of these tracked frames are shown in Figs. 10, 11 and 12 for respectively the *xrhuis*, *crew* and *makerspace* datasets. They show the captured image on the left. Columns two to four show an overlay of the captured image with a rendered equivalent as seen from

Fig. 11 Example results on a test sequence of the *crew* dataset. For the columns from left to right: (first) ground truth keyframe image; (second) overlay of the roughly aligned frame with the ground truth; (third) overlay of the marker-based aligned frame with the ground truth; (fourth) overlay of the keyframe-refined frame with the ground truth. For each blended pair, the figure shows feature matches and their respective distances



Fig. 12 Example results on a test sequence of the *makerspace* dataset. For the columns from left to right: (first) ground truth keyframe image; (second) overlay of the roughly aligned frame with the ground truth; (third) overlay of the marker-based aligned frame with the ground truth; (fourth) overlay of the keyframe-refined frame with the ground truth. For each blended pair, the figure shows feature matches and their respective distances



the estimated SLAM pose for respectively the ‘aligned’, ‘markers’ and ‘pointcloud’ registration types. In these examples, one can clearly see an apparent offset, in the form of ghosting, between the target RGB image and the estimated pose for the ‘aligned’ version. For the ‘marker’ version the misalignment is reduced. Note that this is primarily the case for poses close to markers or where markers are clearly visible (this can for example be seen in the second and fifth result of Fig. 11). The ‘pointcloud’ version, which uses the additional keyframe refinement approach shows an even further improvement. One can see that the estimated pose almost entirely coincides with the target image. However, even though the latter performs best in most cases, we have observed that, similar to the markers, the alignment is better in the neighborhood of keyframes that were used for refinement. Moving further away from these keyframes positions decreases the alignment precision. An important conclusion is that the performance is best in the neighborhood of keyframes and markers. We believe that finding better ways to balance the keyframes compared to the point cloud and improving keyframe management, in contrast to the default behaviour of the SLAM framework, will be key for improvements in the future.

To quantify the difference in 2D alignment, we have used feature detection and matching between the target image and the rendered poses for each of the three registration levels. Inspired by the keyframe refinement of Sect. 4.5, we have estimated a ground truth pose for each of the estimated poses, based on the 2D-3D correspondences of the feature matching using Efficient Perspective-n-Point (EPnP) (Lepetit et al. 2009). We use SURF (Bay et al. 2006) features for matching. RANSAC (Fischler and Bolles 1981), is used to get a good set of inliers for each estimated ground truth pose. The inliers of the features are shown in Figs. 10, 11 and 12 as well, where we show the 2D distance between the various matches. For this, one can observe that the difference in distances is also lower for the markers and point cloud results. If we quantify the mean distance of the matched 2D features for all the test images in the sequence, we get results as shown in Table 2. The table shows, for each dataset, the mean, standard deviation, 50th percentile and 90th percentile, for three different metrics. The downstream tasks mainly require accurate projection and alignment of 3D to 2D images, which is the key objective of this work. We argue that for this, a 2D metric is the most relevant metric. However, to give an idea of the 3D accuracy of the poses we have added two other 3D metrics. The three metrics are: the 3D difference in angle in degrees between the ground truth pose and the estimated pose with the registered SLAM (pose angle error); the 3D Euclidean distance between the ground truth pose and the estimated SLAM pose in centimeters (pose distance error); and the 2D distance in pixels between matched features between the ground truth image

as captured by the camera and the rendered point cloud rendered towards the estimated viewpoint (2D distance error). The first two metrics rely on the EPnP solutions which can also introduce some inaccuracies. The latter 2D distance error only relies on EPnP for identifying inliers. A couple of observations can be made. In general it can be observed that the rough alignment performs worst, and the marker-based and keyframe-based alignment improves the result for the *xrhuis* and *crew* datasets. For all three datasets, the keyframe alignment based on the point cloud outperforms the other registrations in types of 2D alignment. The improvement is the most apparent for the *crew* dataset, going from a misalignment of more than 51 pixels towards 5 pixels for half of the datasamples. The least pronounced improvement is for the *makerspace* dataset. Here we see only a marginal improvement of about 3 pixels and even no improvement for the other metrics. For this dataset, we have observed that the initial SLAM map is already acceptable and that few drift deformations are present. We believe that this can be due to the fact that during map building, a large number of loops were detected and, as such, the SLAM system itself was capable of removing a considerable amount of the drift. Unfortunately, this type of behavior is unpredictable and does not scale well to large environments where it is harder or even unfeasible to apply extensive loop closing.

7 Conclusion

In this article, we have presented a VR system that supports co-location with the real world for large environments. The proposed method works by splitting up the process in a separate mapping and localization phase, using the fact that most parts of the environment are not expected to change drastically between sessions. During the pre-processing phase, ArUco markers are detected and used to globally align a complete feature map of the environment. A further refinement step is proposed by matching the ground truth keyframes to the point cloud and using this as extra guidance for the registration. At run time, the global poses are combined with the poses of the XR headset using a sensor fusion approach. The system is initially designed for VR environments, but we have shown that it is generic and could be used for other XR systems as well. Keyframe management is an avenue for future research as more evenly distributed keyframes will help the registration effort, minimizing the likelihood of tracking in areas where the map is still distorted. Using laser scans as a ground truth reference is not practical or cost-effective for all applications. The continuous improvement of state-of-the-art techniques in large-area 3D scanning, such as photogrammetry (Mortezapour et al. 2022), which is becoming increasingly accurate, could reduce the need for laser scans in the future. Future research

in the form of user studies is required to assess the impact of the sensor fusion and drift correction in the VR headset. The proposed system can achieve proper co-location between virtual and physical worlds in the context of large tracking environments.

Authors' contributions N.M. is the lead author of the submission and has worked on preparing the manuscript text and all the technical components and evaluation of the submission. L.J. contributed to all technical components of the submission. J.P. contributed to the writing and polishing of the manuscript text. J.L. contributed to the sensor fusion component. I.V. contributed to the evaluation of the system. E.J. is the principal investigator for CREW. F.V.R. is the principal investigator for Hasselt University. All authors reviewed the manuscript.

Funding This research was funded by the European Union (HORIZON MAX-R, Mixed Augmented and Extended Reality Media Pipeline, 101070072) and the Flanders Make's XRTwin SBO project (R-12528).

Availability of data and materials The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest No, I declare that the authors have no Conflict of interest as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Ethical approval Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of two 3-D point sets. *IEEE Trans Pattern Anal Mach Intell PAMI* 9(5):698–700. <https://doi.org/10.1109/TPAMI.1987.4767965>
- Bar-Shalom Y, Li X, Kirubarajan T (2001) Estimation with applications to tracking and navigation: theory, algorithms and software. <https://api.semanticscholar.org/CorpusID:108666793>
- Bay H, Tuytelaars T, Van Gool L (2006) Surf: speeded up robust features. In: Leonardis A, Bischof H, Pinz A (eds) *Computer vision—ECCV 2006*. Springer, Berlin, pp 404–417
- Besl PJ, McKay ND (1992) A method for registration of 3-d shapes. *IEEE Trans Pattern Anal Mach Intell* 14(2):239–256. <https://doi.org/10.1109/34.121791>
- Campos C, Elvira R, Rodríguez JJG, Montiel JMM, Tardós JD (2021) Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. *IEEE Trans Robotics* 37(6):1874–1890. <https://doi.org/10.1109/TRO.2021.3075644>
- Durrant-Whyte H, Rye D, Nebot E (1996) Localization of autonomous guided vehicles. In: Giralto G, Hirzinger G (eds) *Robotics research*. Springer, London, pp 613–625
- Fang W, Chen L, Zhang T, Chen C, Teng Z, Wang L (2023) Head-mounted display augmented reality in manufacturing: a systematic review. *Robotics Comput Integr Manuf* 83:102567
- Feng M, Hu S, Ang MH, Lee GH (2019) 2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud. *CoRR*. [arxiv:1904.09742](https://arxiv.org/abs/1904.09742)
- Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 24(6):381–395. <https://doi.org/10.1145/358669.358692>
- Furtado JS, Liu HHT, Lai G, Lacheray H, Desouza-Coelho J (2019) Comparative analysis of optitrack motion capture systems. In: Janabi-Sharifi F, Melek W (eds) *Advances in motion sensing and control for robotic applications*. Springer, Cham, pp 15–31
- Garrido-Jurado S, Muñoz-Salinas R, Madrid-Cuevas FJ, Marín-Jiménez MJ (2014) Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit* 47(6):2280–2292. <https://doi.org/10.1016/j.patcog.2014.01.005>
- Greene N (1986) Environment mapping and other applications of world projections. *IEEE Comput Graph Appl* 6(11):21–29. <https://doi.org/10.1109/MCG.1986.276658>
- Jorissen L, Maesen S, Doshi A, Bekaert P (2014) Robust global tracking using a seamless structured pattern of dots. In: De Paolis LT, Mongelli A (eds) *Augmented and virtual reality*. Springer, Cham, pp 210–231
- Kalman RE (1960) A new approach to linear filtering and prediction problems. *Trans ASME-J Basic Eng* 82(Series D):35–45
- Kazerouni IA, Fitzgerald L, Dooly G, Toal D (2022) A survey of state-of-the-art on visual slam. *Expert Syst Appl* 205:117734. <https://doi.org/10.1016/j.eswa.2022.117734>
- Lang P, Kusej A, Pinz A, Brasseur G (2002) Inertial tracking for mobile augmented reality. In: *IMTC/2002. Proceedings of the 19th IEEE instrumentation and measurement technology conference (IEEE Cat. No.00CH37276)*, vol 2, pp 1583–1587. <https://doi.org/10.1109/IMTC.2002.1007196>
- Lepetit V, Moreno-Noguer F, Fua P (2009) EPnP: an accurate O(n) solution to the PnP problem. *Int J Comput Vis* 81(2):155–166. <https://doi.org/10.1007/s11263-008-0152-6>
- Li J, Lee GH (2021) Deepi2p: image-to-point cloud registration via deep classification. *CoRR*. [arxiv:2104.03501](https://arxiv.org/abs/2104.03501)
- Maesen S, Goorts P, Bekaert P (2013) Scalable optical tracking for navigating large virtual environments using spatially encoded markers. In: *Proceedings of the 19th ACM symposium on virtual reality software and technology. VRST'13*. Association for Computing Machinery, New York, pp 101–110. <https://doi.org/10.1145/2503713.2503733>
- McGill M, Gugenheimer J, Freeman E (2020) A quest for co-located mixed reality: aligning and assessing slam tracking for same-space multi-user experiences. In: *26th ACM Symposium on virtual reality software and technology. VRST'20*. Association for Computing Machinery, New York. <https://doi.org/10.1145/3385956.3418968>
- Mortezapour S, Schönauer C, Rüggeberg J, Kaufmann H (2022) Photogrambot: an autonomous ROS-based mobile photography robot for precise 3d reconstruction and mapping of large indoor spaces for mixed reality. In: *2022 IEEE conference on virtual reality and 3D user interfaces abstracts and workshops (VRW)*, pp 101–107. <https://doi.org/10.1109/VRW55335.2022.00033>
- Niehorster DC, Li L, Lappe M (2017) The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research. *i-Perception*

- 8(3):2041669517708205. <https://doi.org/10.1177/2041669517708205>. PMID: 28567271
- Podkosova I, Vasylevska K, Schoenauer C, Vonach E, Fikar P, Broned-erk E, Kaufmann H (2016) Immerseddeck: a large-scale wireless VR system for multiple users. In: 2016 IEEE 9th Workshop on software engineering and architectures for realtime interactive systems (SEARIS), pp 1–7. <https://doi.org/10.1109/SEARIS.2016.7551581>
- Reimer D, Podkosova I, Scherzer D, Kaufmann H (2021) Colocation for slam-tracked VR headsets with hand tracking. *Computers* 10(5). <https://doi.org/10.3390/computers10050058>
- Ren S, Zeng Y, Hou J, Chen X (2023) Corri2p: Deep image-to-point cloud registration via dense correspondence. *IEEE Trans Circuits Syst Video Technol* 33(3):1198–1208. <https://doi.org/10.1109/TCSVT.2022.3208859>
- Strasdat H, Montiel JMM, Davison AJ (2010) Real-time monocular slam: why filter? In: 2010 IEEE International conference on robotics and automation, pp 2657–2664. <https://doi.org/10.1109/ROBOT.2010.5509636>
- Triggs B, McLauchlan PF, Hartley RI, Fitzgibbon AW (2000) Bundle adjustment—a modern synthesis. In: Triggs B, Zisserman A, Szeliski R (eds) *Vision algorithms: theory and practice*. Springer, Berlin, pp 298–372
- Wang A, Gollakota S (2019) Millisonic: Pushing the limits of acoustic motion tracking. In: *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp 1–11
- Wright MJ, Freed A (1997) Open soundcontrol: a new protocol for communicating with sound synthesizers. In: *International conference on mathematics and computing*. <https://api.semanticscholar.org/CorpusID:27393683>
- Yi X, Zhou Y, Habermann M, Golyanik V, Pan S, Theobalt C, Xu F (2023) EgoLocate: real-time motion capture, localization, and mapping with sparse body-mounted sensors

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.