Bag of World Anchors for Instant Large-Scale Localization

Fernando Reyes-Aviles, Philipp Fleck, Dieter Schmalstieg, and Clemens Arth

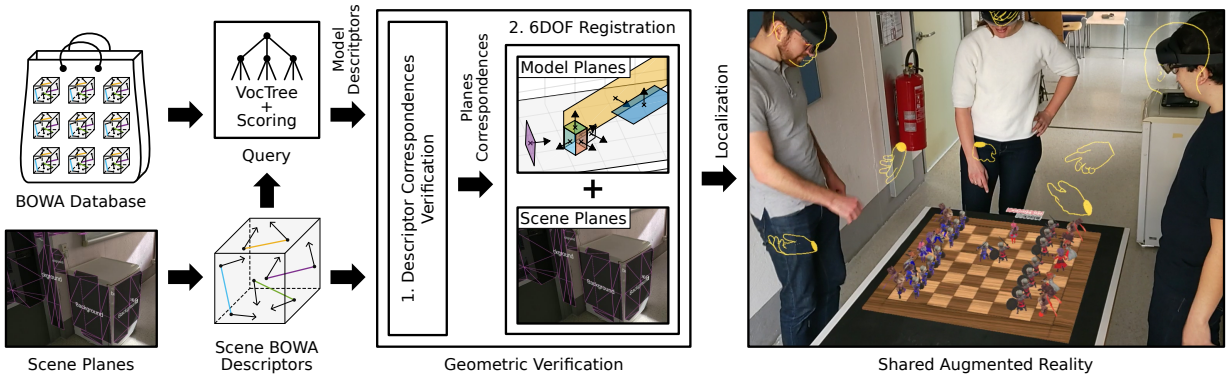


Fig. 1: Overview of our *Bag of Wor(l)d anchors* (BOWA) approach for instant large-scale localization in an analogy to the Bag-of-Words (BOW) approach by Nister and Stewenius [37]. While a single scene is described as a collection of BOWA descriptors inferred from geometric primitives, the collectivity of these descriptions for multiple environments is organized in a tree-like database structure to maintain high scalability. Given an actual environment captured with a mobile device featuring geometric primitive detection, a set of candidate environments from the database is inferred by indexing into the database. A subsequent verification step identifies the correct environment, while the result of 6DOF pose estimation can ultimately be used to register the device, respectively the user, in a single shared augmented reality space.

Abstract—In this work, we present a novel scene description to perform large-scale localization using only geometric constraints. Our work extends compact world anchors with a search data structure to efficiently perform localization and pose estimation of mobile augmented reality devices across multiple platforms (*e.g.*, HoloLens 2, iPad). The algorithm uses a bag-of-words approach to characterize distinct scenes (*e.g.*, rooms). Since the individual scene representations rely on compact geometric (rather than appearance-based) features, the resulting search structure is very lightweight and fast, lending itself to deployment on mobile devices. We present a set of experiments demonstrating the accuracy, performance and scalability of our novel localization method. In addition, we describe several use cases demonstrating how efficient cross-platform localization facilitates sharing of augmented reality experiences.

Index Terms—Camera localization, Correspondence problem, 3D registration, Augmented Reality, Computer vision, Cross-platform, Collaborative, Structural modeling.

1 Introduction

Self-localization on mobile devices is a key enabling technology for augmented reality (AR). Recent commercial AR solutions, such as Apple's ARKit, Google's ARCore, and Microsoft's Mixed-Reality Toolkit (MRTK), provide this functionality out of the box. These solutions build on hardware-accelerated code for simultaneous localization and mapping (SLAM) to deliver incremental 6DOF pose tracking as well as a map of the observed environment (a so-called *world anchor*). The world anchors can be searched by the SLAM code to re-establish the pose (in case tracking is lost) or stored for *pose detection* from scratch at a later time. Multiple world anchors can be searched for similarities with the currently observed environment to provide *place detection*, *i.e.*, the identification of the user's environment (*e.g.*, the current room).

If a unified large-scale reconstruction of the global environment is available, place detection and pose detection can be combined into a single operation, *global localization*. However, previous methods for global localization are expensive in terms of storage and computation. For example, the file size of Microsoft Azure world anchors is commonly in the 20-30 MB range, making it expensive to load, store or transmit these datasets. Besides, world anchors are derived from device-specific sensing capabilities (*e.g.*, Lidar sensor) and therefore tied to a specific platform. As a consequence, world anchors cannot be easily exchanged between mobile devices of different vendors. Instead, cross-platform global localization services are usually provided as paid cloud services, which come with significant disadvantages. Apart from vendor lock-in and potential loss of privacy, cloud services require constant network connectivity and introduce additional latency. This situation makes the creation of shared applications between heterogeneous AR devices cumbersome or impossible.

Reyes-Aviles *et al*. [38] recently introduced the concept of compact world anchors (CWA), which enable pose detection from pairwise relationships between geometric entities (planes, cylinders and spheres), which are detected in SLAM maps. The resulting CWA data structure is tiny (kilobytes instead of megabytes), and the computations for pose detection are cheap.

However, CWA has some limitations. First and foremost, it can only provide pose detection, but not place detection. The right CWA dataset for the current scene needs to be available ahead of time. It can possibly distinguish a small number of geometrically unique places, if multiple CWA datasets are simply concatenated. However, its power to

- *Fernando Reyes-Aviles is with VRVis Competence Center in Vienna, Austria.E-mail: fernando.reyes-aviles@icg.tugraz.at*
- *P. Fleck, D. Schmalstieg and C. Arth are with Graz University of Technology, Austria. E-mail: {philipp.fleck, schmalstieg, arth}@icg.tugraz.at.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

discriminate places does not scale well. The ability of CWA to work across multiple platforms and users has not yet been investigated.

In this work, we aim to continue the research direction that started with CWA. To this end, we introduce a new scene representation nicknamed *bag of wor(l)d anchors* (BOWA), which is created from a novel geometric descriptor with enhanced resilience compared to CWA. Hence, our novel scene representation inherits the lightweight representation of CWA, but allows both pose detection and place detection. Our core contributions are:

- A robust feature descriptor that lends itself to both pose detection and place detection
- A scalable data structure for global localization with constant runtime
- An extensive evaluation of the performance of BOWA, both in terms of correspondences matching success rate, as well as in terms of cross-platform performance on a sizable indoor environment
- A set of collaborative applications to demonstrate cross-platform, multi-user applications that would have been hard or impossible to build before

We present data showing that BOWA can find the correct scene within a few milliseconds on a mobile or head-worn device, while yielding centimeter-level accurate pose results.

2 Related Work

Our work combines aspects of broad areas, including SLAM, pose detection, and scene understanding. We provide a brief survey of the most important concepts and instances of prior work, while we refer the reader to surveys [6, 31–33, 40] for more extensive coverage.

2.1 Simultaneous localization and mapping

With sufficient parallel processing power, keyframe SLAM became feasible [23], ushering in a wave of real-time SLAM methods. Among them, volumetric integration [21, 35] based on depth sensors may have been the next leap ahead. Although many extensions have been proposed, those approaches are still considered the state of the art for non-semantic SLAM.

Leading commercial solutions for AR, such as ARKit, ARCore and MRTK, feature hardware-supported SLAM systems (*i.e.*, using custom sensors and processors), which are also capable of storing their maps as world anchors, in a proprietary, binary format. Microsoft calls them *Azure Anchors*, Google, *ARCore Cloud Anchors* [3], and Apple, *ARWorldMap* [1]. The former two methods can be shared across devices and vendors by means of a cloud service, while Apple's format is confined to stay within its ecosystem. The size of the anchors and the need to invoke a paid cloud service make it impractical to quickly share anchors between mobile devices. Likewise, curating collections of world anchors to cover large areas is tedious. Yet another impediment for AR developers stems from the fact that anchors are created by automatically scanning the user's environment and cannot be restricted to contain only certain objects, such as offered by third-party tracking tools, like Vuforia's area targets [4].

2.2 Pose detection

Standalone (non-incremental) pose detection is a key ingredient of both conventional model-based pose tracking and of SLAM systems. Assuming a model of an object or place, an initial pose must be identified from a single image [22,41], or the pose must be re-established instantly after a tracking failure [18,47]. Such pose detection is typically done by establishing 2D-3D relationships between the interest points in the image and known features of the model, or by index data structures, such as Ferns [18], derived from the features. Spurious matches are eliminated with probabilistic sampling methods, such as RANSAC [15]. For optimal performance, both features and index structures are usually hand-crafted and closely tied to the sensing hardware, making it difficult to achieve cross-platform operation. In either case, traditional pose detection methods operate on a small model and may require a reasonable pose guess as a starting point. Technically, CWA [38] is also a pose detection method, as it operates on a small model of no more than a few world anchors and does not offer any place detection ability.

2.3 Localization using bag of words

Conventional pose detection based on linear search through a feature set does not scale well to address the needs of global localization. Apart from the obvious challenges in searching through a growing database in constant time, individual features are not discriminative enough to distinguish both place and pose at the same time. Therefore, Nister and Stewenius [37] proposed the *bag of words* (BOW) approach, which – in a nutshell – considers how rare an observed feature is as a way to distinguish places. This idea can be implemented with a so-called vocabulary tree, which groups similar features for quick searching.

This idea became very popular in the following years. Agarwal *et al*. [7] used vocabulary trees to create reconstructions of sights in different cities from publicly available image collections. Irschara *et al*. [20] focused on the task of localization from 3D reconstructions, partitioning reconstructions based on virtual views. Arth *et al*. proposed a BOW approach for real-time localization on mobile devices [9, 10]. Widespread toolkits like ORB-SLAM2 [34] continue to rely on the BOW approach, although research today focuses more on deep learning methods. In this paper, we show how the BOW approach can boost the scalability of geometric feature detection.

2.4 Deep-learning-based localization

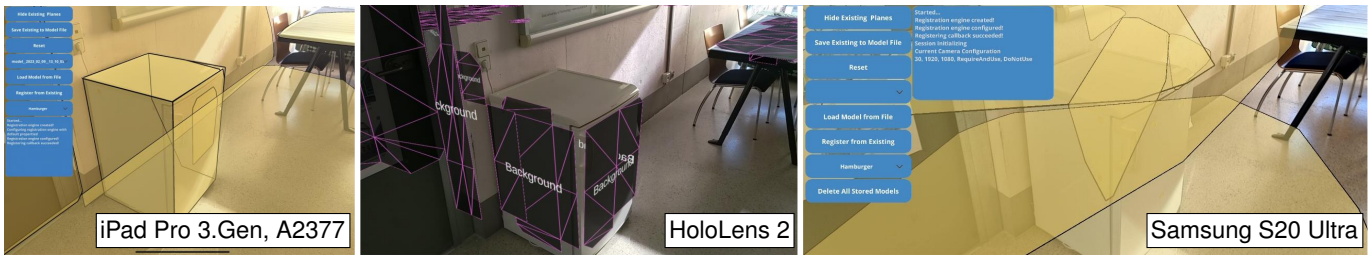
Recently, many deep-learning methods focusing on visual place recognition have been proposed [8, 17, 30, 45]. The key to solving the visual place recognition problem is an efficient image retrieval (*i.e.*, finding the most similar image in the database). However, at the core of these methods reside convolutional neural network (CNN) architectures for feature extraction from RGB images, dense descriptors creation and matching. Other approaches leverage dense 3D point clouds [14,24,48] to tackle the localization problem in unstructured, dynamic environments, where local features are not discriminative enough and global scene descriptors only provide coarse information. These approaches bring great advancement to the computer vision community; however, most (if not all) of them are not applicable reasonably on mobile hardware (*e.g.*, iPad Pro, HoloLens 2, Magic Leap 2) at the moment of writing this paper. Besides, contrary to the dense descriptors these approaches utilize, our proposed place recognition method leverages geometric primitives, from which we can compute low-dimensional descriptors only.

2.5 Semantic SLAM and scene understanding

Semantic SLAM methods build maps from higher-level primitives than visual point features. One of the first attempts was SLAM++ [39], which incorporates semantic information per object using polygonal models. Follow-up approaches, including QuadricSLAM [36] and others [12, 25, 26, 49], use more specific, complex object instances rather than generic shapes. Later work includes deep learning as well, such as DROID-SLAM [46]. In general, these methods aim to increase the efficiency of map search by composing maps of objects with a higher level of abstraction than point features, but they do not attempt to organize large maps for optimal search and detection, which is the goal of this work.

While there are a lot of deep-learning approaches targeting pixel-wise semantic segmentation (*e.g.*, identifying an object class per pixel), there is relatively little work on real-time primitive detection. Recent work includes the approach of Sommer *et al*. [43] and an AR related approach of Stanescu *et al*. [44]. Another similar objective is pursued by offline reconstruction methods aiming to estimate room layouts. Among these approaches, Cabral and Furukawa were the first to describe a system for reconstruction of piece-wise planar floor plans from images [11]. More recently, deep learning has been applied to the problems, including LayoutNet [50], PlaneNet [29] and subsequently PlaneRCNN [28]. In the latter work, geometric properties of a scene are inferred by processing single RGB images. Overall, extracting geometric information is a very vivid topic in computer vision. However, none of these methods is concerned with detection problems.

Commercial solutions (ARKit, ARCore and Microsoft's Scene Understanding SDK) also include the ability to detect horizontal and vertical planes. Empirically, horizontal planes tend to work much more



(a) (b) (c)

Fig. 2: Comparison of plane detection results on a real scene for (a) ARKit, (b) Scene Understanding SDK and (c) ARCore. Observing, for example, the table on the right of the pictures, the plane detection results are quite accurate owing to the depth perception capabilities of the devices for (a) and (b), while the results using imagery only (c) are useless for our approach.

reliable than vertical ones. None of the toolkits offers the detection of geometric primitives other than planes, like cylinders or spheres. To the best of our knowledge, the iPad Pro and HoloLens 2, make use of depth cameras for plane estimation, while the capabilities of Android's ARCore are very limited, even on high-end devices (*e.g.*, Samsung S20 Ultra with depth camera). Unfortunately, this makes ARCore devices almost useless for our goals. See Figure 2 for a comparison between different device classes. However, we consider this a temporary disadvantage, and we demonstrate cross-platform operation of BOWA for the remaining platforms.

3 Scene Description

The original CWA approach was only tested on small models obtained using either (a) InfiniTAMv3 and structural modeling [44] or (b) ARKit, and using exhaustive correspondence search between the scanned scene primitives and a stored world anchor. At the core of CWA resides a 3-vector descriptor

$$\mathbf{F}_{\mathrm{CWA}}(\mathbf{p}_1,\mathbf{p}_2) = (\angle(\mathbf{n}_1,\mathbf{d}),\ \angle(\mathbf{n}_2,\mathbf{d}),\ \angle(\mathbf{n}_1,\mathbf{n}_2)). \tag{1}$$

This approach delivers very compact, purely geometric descriptor sets, but has several limitations: First, the reference world anchors (and thus the identity of the current place) must be known ahead of time. Second, the environment must be small enough so that a naive search is acceptable. The distinctive power of a 3-vector may be insufficient in the presence of similar, repetitive or partially observed plane configurations present in a larger environment. Establishing false correspondences may lead to incorrect results. In addition, exhaustive search may be too slow for larger environments. Third, pose detection has so far been demonstrated only within a single platform. The planar structures provided by an underlying framework (*e.g.*, Scene Understanding SDK, ARKit, or ARCore) are similar, but not identical, potentially leading to problems during matching.

3.1 Distinctive feature descriptor

To overcome these limitations, we introduce a novel 14-dimensional descriptor vector ($\mathbf{F}_{\mathrm{BOWA}}$), and we combine multiple such descriptors into a world anchor using a BOW-like approach[1].

$\mathbf{F}_{\mathrm{BOWA}}$ includes not only angle differences between the normals of pairs of oriented points (see Figure 3a), but also the sizes and the relative orientation and position between pairs of planes (see Figure 3b). We only considered planes for the development of this new descriptor due to the fact that no commercial SLAM solution provides detection of cylinders, spheres, *etc*. At the time of writing this paper, Microsoft's Scene Understanding SDK, ARKit and ARCore only provide planar surface detection. However, this does not limit our approach in any way, as we will discuss at the end of this work.

To estimate the first four features of our descriptor, we use the *point pair feature* (PPF) concept introduced by Drost *et al*. [13]. From two given planes $\mathbf{m}_1$ and $\mathbf{m}_2$ with surface normals $\mathbf{n}_1$ and $\mathbf{n}_2$, we compute the geometric centroids $\mathbf{p}_1$ and $\mathbf{p}_2$ of their point sets (see Figure 3a).

[1]Technically, our method should therefore be called "Bag of Words As World Anchors", but we did not like the acronym BOWAWA.

These features describe the angular differences between the normals ($\mathbf{n}_1$ and $\mathbf{n}_2$) of the oriented points ($\mathbf{p}_1$ and $\mathbf{p}_2$) and the length of the vector $\mathbf{d}$ between them,

$$\mathbf{F}_{ppfs} = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_1,\mathbf{d}),\ \angle(\mathbf{n}_2,\mathbf{d}),\ \angle(\mathbf{n}_1,\mathbf{n}_2)), \tag{2}$$

where $\mathbf{d} = \mathbf{p}_2 - \mathbf{p}_1$, and $\angle(a,b) \in [0,\pi]$ denotes the angle between two vectors. The next four features encode the size of the pair of primitives,

$$\mathbf{F}_{sizes} = (A_2, A_1, R_2, R_1), \tag{3}$$

where A_1 and A_2 denote the surface area of planes $\mathbf{m}_1$ and $\mathbf{m}_2$ respectively, and R_1 and R_2 denote the aspect ratio computed as

$$R = \frac{\max(\mathbf{L})}{\min(\mathbf{L})}, \tag{4}$$

where $\mathbf{L}$ is a vector that contains the edge lengths of the minimum bounding rectangle around all observed points of a given plane. The last six features of our descriptor encode the relative orientation and position between pairs of primitives,

$$\mathbf{F}_{pose} = \left(Q_w, Q_x, Q_y, Q_z, t_x, t_z\right), \tag{5}$$

where $\mathbf{Q}$ is a quaternion which denotes the relative rotation from one plane to another, and $\mathbf{t}$ is the relative position from one plane to another. For example, in Figure 3b, we set $\mathbf{p}_1$ to be the origin of a local coordinate system which contains planes $\mathbf{m}_1$ and $\mathbf{m}_2$. We align the normal $\mathbf{n}_1$ with the z axis and rotate the point $\mathbf{p}_2$ about the z axis to let it lie on the xz plane. We denote the 3D points after the transformation as follows:

$$\mathbf{p}_1 = [0,0,0]^\top, \qquad \Pi_{xz}^\top \mathbf{p}_2 = 0 \ \therefore\ \mathbf{p}_2 = [x,0,z]^\top. \tag{6}$$

We write the equation of the plane $\mathbf{m}_1$ and the normal $\mathbf{n}_1$ after the transformation as follows:

$$z = 0 \ :\ \mathbf{m}_1, \qquad \mathbf{n}_1 \cdot z = 1 \ \therefore\ \mathbf{n}_1 = [0,0,z]^\top. \tag{7}$$

Finally, our descriptor for two planes $\mathbf{m}_1$ and $\mathbf{m}_2$ is

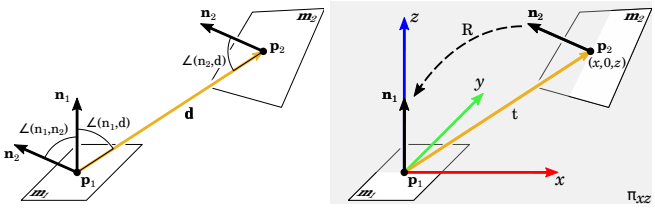
$$\mathbf{F}_{\mathrm{BOWA}}(\mathbf{m}_1,\mathbf{m}_2) = \left(\mathbf{F}_{ppfs}, \mathbf{F}_{sizes}, \mathbf{F}_{pose}\right), \tag{8}$$

3.2 Descriptor validation

$\mathbf{F}_{\mathrm{BOWA}}$ descriptors are computed using pairs of primitives. Because not every combination of two primitives yields a meaningful descriptor, we established some rules based on the geometric properties of the features and the observations made during the evaluation of the proposed approach on real-world scenarios.

The first four features (see Equation 2) are computed from pairs of oriented points (see Figure 3a). We extract such points by calculating the geometric centroids of planes, *i.e.*, by computing the average of the 3D points which define the minimum bounding rectangle of a plane.

Parallel or co-planar planes cannot be used, because the three angles $\angle(\mathbf{n}_1,\mathbf{d})$, $\angle(\mathbf{n}_2,\mathbf{d})$, and $\angle(\mathbf{n}_1,\mathbf{n}_2)$ of Equation 2 would be equal. Other



(a) (b)

Fig. 3: Geometric features extracted from pairs of primitives to create $\mathbf{F}_{\text{BOWA}}$ descriptors. (a) Example of the four features used to create $\mathbf{F}_{ppfs}$ in Equation 2. (b) Example of the local coordinate system used to create the $\mathbf{F}_{pose}$ features in Equation 5.

problematic cases are planes with very large surface area that must be fully observable, which is only possible in a very small fraction of the localization attempts. Therefore, we do not compute descriptors from parallel or co-planar planes, planes with areas $A > 7$ m^2. We also discard planes with a high aspect ratio $R > 8$, which are usually the product of partial observations or occlusions. These thresholds were empirically determined during our evaluations on real-world scenarios.

4 Scene Recognition

Having to know the world anchor corresponding to the current place ahead of time is a major bottleneck for practical applications. We introduce a fast search structure to automatically identify the right BOWA out of a larger number of possibilities. To this aim, we employ a vocabulary tree similar to the work of Nister and Stewenius [37] to solve this problem. In doing so, we simultaneously solve place and pose detection, while retaining the superior efficiency of geometric feature descriptors.

4.1 Vocabulary tree creation

In a classical BOW approach, a search tree is created from a large number of descriptors by repeatedly splitting them according to an arbitrary dimension of the feature vector. The SIFT features used in the work of Nister and Stewenius [37] have 128 dimensions, but all dimensions are uniform values defined over the same domain. A tree is built using SIFT descriptors extracted from existing image databases containing millions of images.

In contrast, the dimensionality of 14 used in $\mathbf{F}_{\text{BOWA}}$ is relatively low. Alas, creating a tree for $\mathbf{F}_{\text{BOWA}}$ is not trivial, since we cannot take advantage of existing databases. Collecting a large enough dataset of planar descriptions of real scenes (we expect it would have to contain thousands of scenes) is impossible time-wise. Therefore, we build our vocabulary tree from $\mathbf{F}_{\text{BOWA}}$ descriptors extracted from synthetic, random scene models. As shown in Figure 4, our scene creation engine builds room-sized models with an arbitrary number of box-like structures, somewhat akin to messy storage spaces. Using a larger number of these models, we can extract a sufficient number of $\mathbf{F}_{\text{BOWA}}$ descriptors to build a vocabulary tree with a suitable parameter set for the branching factor and the tree depth.

A potential drawback of an approach relying on synthetic exemplars is that we cannot guarantee that the feature space, respectively, the leaves of the resulting vocabulary tree, cover the descriptor space well. However, we can at least ensure that the numbers used to build the tree are in a plausible range. For example, using 2,500 synthetic models with each model having between 20 and 50 planes and between 200 and 2,000 $\mathbf{F}_{\text{BOWA}}$ descriptors, we can build a vocabulary tree, with six levels and a branch factor of ten; with a good retrieval performance.

4.2 Database training and online recognition

The training of the vocabulary tree is essentially an indexing task. Given a BOWA set, *i.e.*, N sets of $\mathbf{F}_{\text{BOWA}}$ descriptors for N rooms, the vocabulary is trained by traversing each descriptor of each BOWA instance through the tree and recording the weighted paths the descriptors followed inside the tree. In other words, for each j-th BOWA instance in the database, we compute a database vector d_i^j defined as

$$d_i^j = m_i^j \omega_i, \quad i = 1, \ldots, \frac{k^{L+1}-k}{k-1}, \quad 1 \le j \le N, \tag{9}$$

where k defines the branch factor (number of children of each node i) of the tree, L refers to the number of levels of the tree, m_i^j denotes the number of descriptors of the given j-th BOWA instance with a path through node i, with the weights ω_i defined as

$$\omega_i = \ln \frac{N}{N_i}, \tag{10}$$

where N is the number of BOWA instances in the database and N_i is the number of BOWA instances in the database with at least one descriptor vector path through node i. We also create a descriptor-to-leaf index by storing the leaves where the $\mathbf{F}_{\text{BOWA}}$ descriptors fall into. We use such an index to later identify scene-to-model correspondences between a single BOWA and our query scene (see Figure 5). After the vocabulary tree is created, we build an inverted index [42]. Such an index associates a given BOWA instance with as many leaf nodes as $\mathbf{F}_{\text{BOWA}}$ descriptors it contains. This index is used during the online recognition phase for efficient hierarchical scoring. It allows us to find all BOWA instances, in the database, in which a query $\mathbf{F}_{\text{BOWA}}$ descriptor occurs.

Likewise, in the training step, during the online phase we obtain the weighted paths followed inside the tree, *i.e.*, the vector

$$q_i = n_i \omega_i, \tag{11}$$

where n_i is the number of $\mathbf{F}_{\text{BOWA}}$ descriptors of the query scene with a path through node i. Since a vocabulary tree is an approximate search structure, the general performance of a BOW approach is measured by considering a successful recognition, if the query scene is within the P most dominant results returned. We use the normalized query and database vectors,

$$\mathbf{q} = \frac{q_i}{\|q_i\|}, \qquad \mathbf{d}^j = \frac{d_i^j}{\|d_i^j\|}, \quad 1 \le j \le N, \tag{12}$$

to compute a relevance score s using an L_p-norm as

$$s^j(\mathbf{q}, \mathbf{d}^j) = \|\mathbf{q} - \mathbf{d}^j\|_p^p = 2 + \sum_i \left(|q_i - d_i^j|^p - |q_i|^p - |d_i^j|^p \right). \tag{13}$$

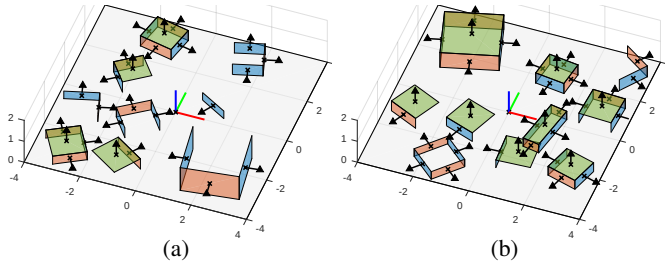
Note that the lower the score s, the higher the relevance. To score efficiently we only compute the score s for those BOWA instances in which a query $\mathbf{F}_{\text{BOWA}}$ occurs. To find those BOWA instances we use the inverted index that we compute during the offline phase. We thus retrieve a list of relevance from which we can identify those P most promising BOWA instances.

4.3 Pose estimation and verification

From our sorted list, we now verify each BOWA in ascending order of score s, *i.e.*, descending order of relevance. In analogy to the calculation of a homography on image queries to identify the right candidate, we use a pose estimation check to validate the BOWA candidate in question (see Figure 5).

First, we obtain a list of potential correspondences by determining the descriptors from the BOWA candidate and the current query scene, which fall into the same leaves, through a simple intersection, *i.e.*, $\mathcal{M}_i(\mathbf{m}, \mathbf{m}')$ and $\mathcal{S}_j(\mathbf{s}, \mathbf{s}')$. Second, we measure the similarity δ_i between all these descriptor pairs as

$$\begin{aligned}
\delta_1 &= |\|\mathbf{d}_m\|_2 - \|\mathbf{d}_s\|_2|, & \delta_2 &= \|\alpha_m - \alpha_s\|_2, \\
\delta_3 &= \frac{|A_{2_m} - A_{2_s}|}{\max(A_{2_m}, A_{2_s})}, & \delta_4 &= \frac{|A_{1_m} - A_{1_s}|}{\max(A_{1_m}, A_{1_s})}, \\
\delta_5 &= \frac{|R_{2_m} - R_{2_s}|}{\max(R_{2_m}, R_{2_s})}, & \delta_6 &= \frac{|R_{1_m} - R_{1_s}|}{\max(R_{1_m}, R_{1_s})}, \\
\delta_7 &= |\langle \mathbf{Q}_m, \mathbf{Q}_s \rangle|, & \delta_8 &= \|\mathbf{t}_m - \mathbf{t}_s\|_2,
\end{aligned} \tag{14}$$



(a) (b)

Fig. 4: Example of synthetic models used to build our vocabulary tree for scene recognition. Comparing (a) and (b), the number of primitives in (b) is apparently higher, subject to a larger set of $\mathbf{F}_{\text{BOWA}}$ descriptors.

where $\alpha = (\angle(\mathbf{n}_1,\mathbf{d}),\ \angle(\mathbf{n}_2,\mathbf{d}),\ \angle(\mathbf{n}_1,\mathbf{n}_2))$, $\mathbf{Q} = \big(Q_w, Q_x, Q_y, Q_z\big)$, $\mathbf{t} = (t_x, t_z)$. We further consider only the *primitives correspondences* $\mathbf{s} \leftrightarrow \mathbf{m}$ and $\mathbf{s}' \leftrightarrow \mathbf{m}'$ when all δ_i are below a minimum value. This step is particularly important, as our vocabulary tree is created with a synthetic set of models. Therefore, also $\mathbf{F}_{\text{BOWA}}$ descriptors which do not give correct correspondences may fall into the same leaves.

We use eight different measurements, because the 14 features are defined over different domains. The length of the vector $\mathbf{d}$ is measured in meters, while the angles $\alpha \in [0,\pi]$ are measured in radians. Areas A of the planes are measured in m^2, while the aspect ratios R denote a proportional relationship between width and height of the planes. The quaternions $\mathbf{Q}$ denote rotations, and the vectors $\mathbf{t}$ denote a displacement of points measured in meters.

Third, we use a voting scheme for robust matching (*i.e.*, outlier rejection). We gather all the *potential* primitive correspondences $\mathbf{s} \leftrightarrow \mathbf{m}$ in an accumulator space $\mathcal{A}$. If a $\mathbf{s} \leftrightarrow \mathbf{m}$ pair receives at least Z votes, we consider it as correspondence. The metric Z, given by

$$Z = \mu - \sigma \cdot \tau, \quad \mu = \frac{1}{m}\sum_{i=1}^{m} \mathcal{A}_i, \quad \sigma = \sqrt{\frac{\sum_{i=1}^{m}(\mathcal{A}_i - \mu)^2}{m-1}}, \qquad (15)$$

indicates the number of standard deviations σ by which the $\mathcal{A}_i$ differ from the mean value μ of the accumulator $\mathcal{A}$. The variable τ is a threshold that we use to control the lower bound of the metric Z.

If we have at least three correspondences available for a candidate BOWA, we can estimate the $\mathbf{T} = [\,\mathbf{R} \mid \mathbf{t}\,] \in \mathbf{SE}(3)$ which registers the candidate BOWA to the query scene using a closed-form method to solve a linear system with 12 unknowns [38]. Similarly to the verification step used there, we measure the rotational and translational error by computing the normal deviation error and the plane-to-plane distance error.

Finally, the first BOWA which achieves at least three matched primitives and has a low registration error to the scene is selected as the final solution, and we can stop the search.

5 Experimental Results

In the following, we evaluate multiple aspects of our approach, such as the overall descriptor matching performance, as well as the perceived localization accuracy in homogeneous and heterogeneous device configurations. We captured four sets of models of 25 different scenes with a third generation iPad Pro (iPad) and a HoloLens 2 (HL2), accumulating a database of 50 iPad (two sets of 25, S1 and S2) and 50 HL2 (two sets of 25, S1 and S2) models. To establish a ground truth, we manually labeled all possible correspondences of geometric primitives between those models.

5.1 Descriptor matching performance

First, we conducted a matching performance evaluation on all device combinations (*i.e.*, iPad-iPad, HL2-HL2 and iPad-HL2) and compared our $\mathbf{F}_{\text{BOWA}}$ descriptors with $\mathbf{F}_{\text{CWA}}$ descriptors. In the heterogeneous case, we match the corresponding models of one device class to the two models of the other one (*i.e.*, S1 from iPad to S1 from HL2, S1 from iPad to S2 from HL2, *etc.*).

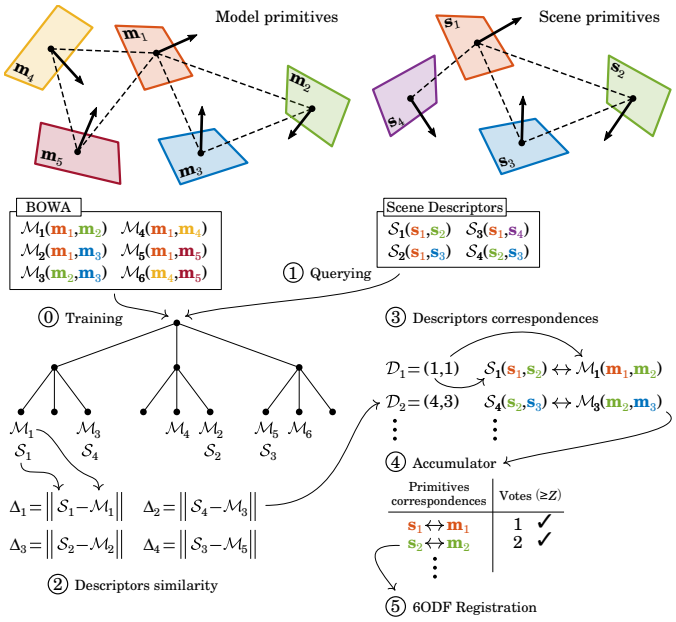


Fig. 5: Minimal example of the post-verification step of our scene recognition method. For a vocabulary tree trained with a single BOWA (step 0), we store the leaves where the $\mathcal{M}$ descriptors fall into. After querying (step 1), we measure the $\mathcal{S}$-to-$\mathcal{M}$ similarity using the distance function in Equation 14 (step 2). We create a list of *descriptors correspondences* $\mathcal{D}_m = (i,j) \rightarrow \mathcal{S}_i \leftrightarrow \mathcal{M}_j$ (step 3). We extract the $\mathbf{s} \leftrightarrow \mathbf{m}$ primitives correspondences from the $\mathcal{S}_i \leftrightarrow \mathcal{M}_j$ and accumulate them for voting (step 4). Finally, iff we have at least three matched primitives, we compute the 6DOF registration (step 5).

The results of the iPad-HL2 experiment are shown in Figure 6, while the results for the homogeneous device configurations are given in the supplemental material. While $\mathbf{F}_{\text{CWA}}$ has a poor performance in spaces 7, 17, 21 and 23, it gives us a similar amount of true-positive correspondences as $\mathbf{F}_{\text{BOWA}}$ in spaces 1, 11 and 24 and only two false-positives. We further observed that the information encoded in the $\mathbf{F}_{\text{CWA}}$ (see Equation 1) reliably detects pairs of primitives, but is rather weak in outlier rejection. In contrast, $\mathbf{F}_{\text{BOWA}}$ uses its richer geometric features for significantly more robust matching results. An example of this is shown in Figure 7.

To further assess the matching performance, we exhaustively matched the $\mathbf{F}_{\text{CWA}}$ and $\mathbf{F}_{\text{BOWA}}$ descriptors across all device configurations, model combinations and scenes. The result of this experiment is shown in Figure 8. While the diagonal (*i.e.*, matches of corresponding models of the same scene) is barely observable for $\mathbf{F}_{\text{CWA}}$, it is clearly visible for $\mathbf{F}_{\text{BOWA}}$, indicating a considerably superior matching performance.

5.2 Vocabulary tree of BOWA descriptors

In this experiment, we wanted to evaluate the retrieval performance, *i.e.*, identify a plausible number P of candidates. We trained the vocabulary tree with the normalized descriptors of the 25 scenes using a BOWA created from a first device and query it with a BOWA captured with a second device from the other device class. Following this approach, the tree was trained with 13×HL2 and 12×iPad, and queried with 13×iPad and 12×HL2.

In Figure 9, left, the retrieval performance for both CWA and BOWA is shown using their respective ROC curves. The correct BOWA candidate is found more than 90% within the first 10% candidates in the database, while the performance of CWA is clearly inferior. It is also worth noting that, even if CWA is able to identify the correct corresponding model from the database, it is often unable to find enough primitive correspondences to perform any final localization. In analogy, this would correspond to the case where in image retrieval the correct

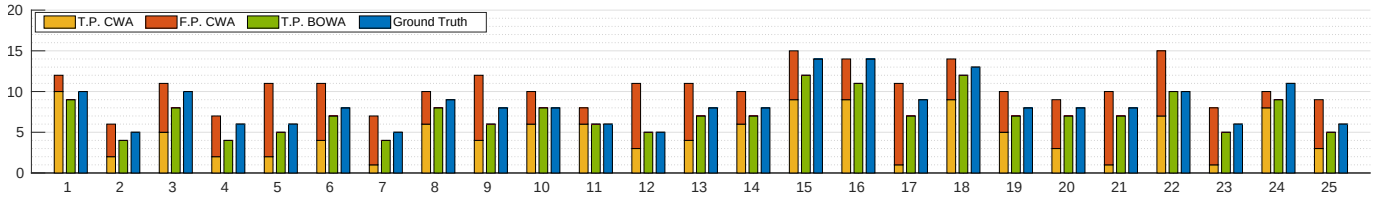
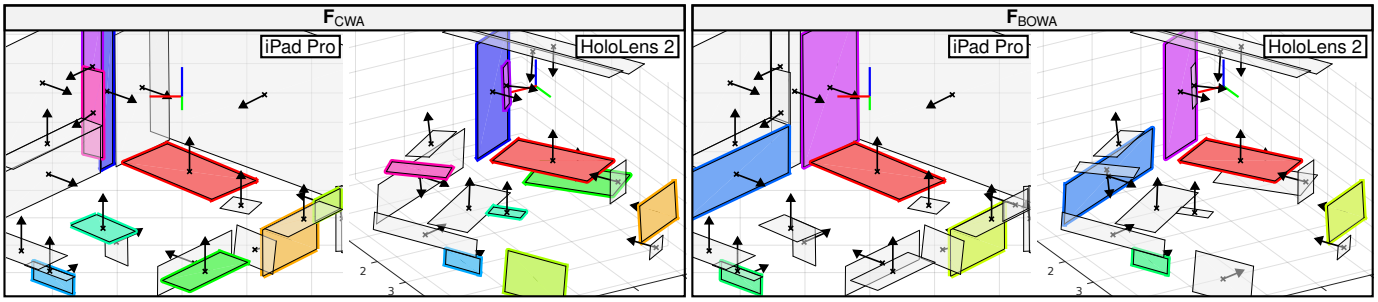
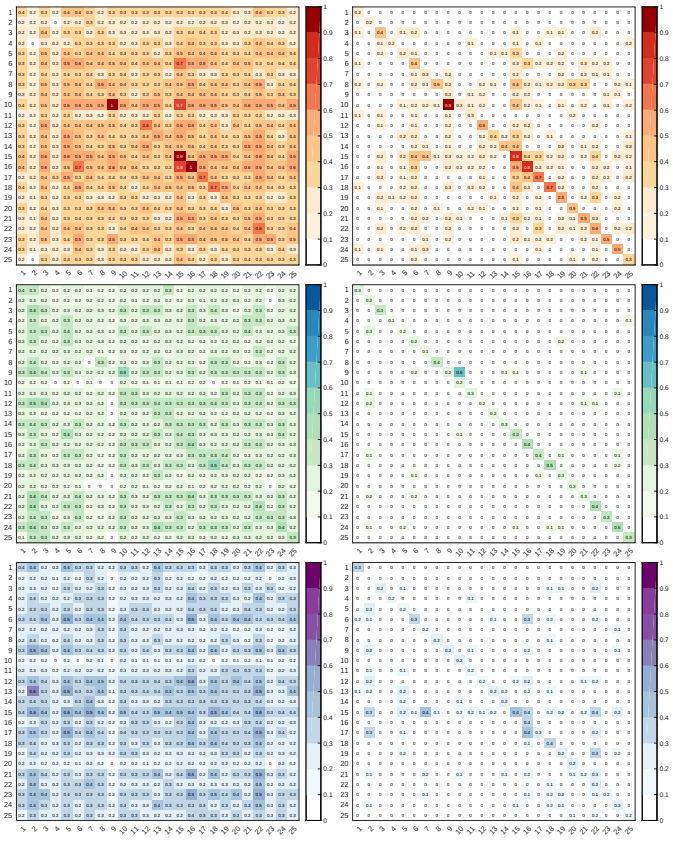


Fig. 6: Evaluation of the performance of the 14-vector descriptor $\mathbf{F}_{\text{BOWA}}$ and the 3-vector descriptor $\mathbf{F}_{\text{CWA}}$ for 25 different scenes for an iPad-HL2 device combination. Each group of bars shows in blue the ground-truth correspondences (manually labeled), the $\mathbf{F}_{\text{BOWA}}$ true-positive correspondences in green, the $\mathbf{F}_{\text{CWA}}$ true-positive correspondences in yellow, and the false-positive correspondences in red. We show the average results of four runs, *i.e.*, A. iPad S1 *vs*. HL2 S1, B. iPad S1 *vs*. HL2 S2, C. iPad S2 *vs*. HL2 S1 and D. iPad S2 *vs*. HL2 S2. The results for the individual tests A, B, C and D are given in the supplemental material. Note the absence of false-positive correspondences for $\mathbf{F}_{\text{BOWA}}$. Due to the increased descriptiveness of $\mathbf{F}_{\text{BOWA}}$, the number of wrong correspondences was zero for all the tests.



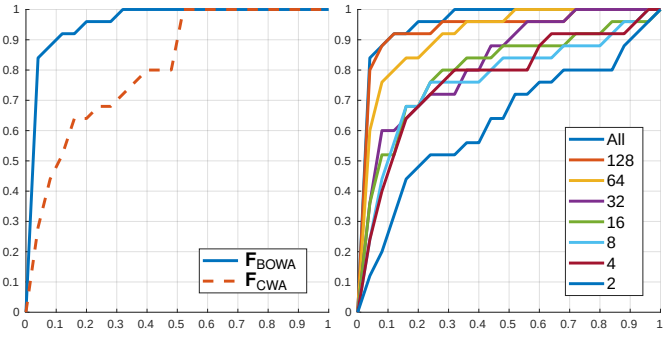
(a) (b) (c) (d)

Fig. 7: Example of the inter-device performance of $\mathbf{F}_{\text{CWA}}$ and $\mathbf{F}_{\text{BOWA}}$ using a model captured with and iPad Pro (a) and (c), and a model captured with a HoloLens 2 (b) and (d). The images in (a) and (b) show the highlighted correspondences found using $\mathbf{F}_{\text{CWA}}$. The images in (c) and (d) show the highlighted correspondences found using $\mathbf{F}_{\text{BOWA}}$. In (a) and (b) we obtained several false positives using $\mathbf{F}_{\text{CWA}}$ descriptors, *e.g.*, the purple and green planes. In (c) and (d), in contrast, we did not obtain false-positive correspondences using $\mathbf{F}_{\text{BOWA}}$ descriptors.



(a) (b)

Fig. 8: Heatmaps of exhaustive evaluation for all scenes, *i.e.*,. (a) all *vs*. all of $\mathbf{F}_{\text{CWA}}$, (b) all *vs*. all for $\mathbf{F}_{\text{BOWA}}$; Each cell shows the percentage of total number of found correspondences, *i.e.*, true-positives plus false-positives. First row: Results of the iPad-iPad test. Second row: Results of the HL2-HL2 test. Third row: Results of the iPad-HL2 test.



(a) (b)

Fig. 9: Vocabulary tree retrieval performance with six levels and a branch factor of ten. (a) ROC curves for $\mathbf{F}_{\text{CWA}}$ and $\mathbf{F}_{\text{BOWA}}$, showing the percentage (y-axis) of BOWA instances that make it into the top x percent (x-axis) of all BOWA instances stored in the tree. It is crucial that the corresponding correct BOWA falls at the top of the result, as verification can only be done for a small fraction of the database when the database grows large. Hence, we are mainly interested in where the curves meet the y-axis. (b) ROC curves with an increasing number of $\mathbf{F}_{\text{BOWA}}$ descriptors.

corresponding database image is identified; however, the subsequent geometric validation check fails due to having too few matches or too many wrong matches.

The graph on the right in Figure 9 shows the retrieval performance for an increasing number of query $\mathbf{F}_{\text{BOWA}}$ descriptors. Note that we achieve 90% performance within the first 10% candidates in the database using only 128 query $\mathbf{F}_{\text{BOWA}}$. The mean number of $\mathbf{F}_{\text{BOWA}}$ obtained from the query scenes is 354, 226 median, 1034 max. and 68 min.

For an experiment that discusses the general performance of the retrieval process on a larger-scale synthetic dataset, the reader is referred to the supplementary material.

	PC		iPad Pro		HoloLens 2	
Step	mean	median	mean	median	mean	median
1. Load VocTree	45.9	45.2	53.8	52.6	804.4	805.3
2. Scene $\mathbf{F}_{\text{BOWA}}$	2.66	1.78	1.36	0.84	13.63	9.61
3. Scene $\mathbf{F}_{\text{BOWA}}$ query	19.3	19.1	16.8	16.8	124.7	124.4
4. Load anchor	0.03	0.02	0.08	0.05	0.85	0.57
5. Verify $\mathbf{F}_{\text{BOWA}}$	0.10	0.10	0.35	0.14	0.49	0.43
6. 6DOF	0.54	0.36	0.58	0.34	3.90	1.74

Table 1: Mean and median runtimes on PC, iPad and HL2 in *ms* for the different steps of the scene recognition and pose estimation process, in order: (1) loading the vocabulary tree; (2) computing the $\mathbf{F}_{\text{BOWA}}$ from the primitives observed in the scene; (3) vocabulary tree traversal time plus the scoring runtime; (4) loading the pre-computed $\mathbf{F}_{\text{BOWA}}$ model descriptors; (5) measuring the similarity of the scene $\mathbf{F}_{\text{BOWA}}$ *vs*. the P candidates (see Equation 14) obtained from the step (3); (6) computing the model to scene registration. Note that (4), (5) and (6) may be repeated more than once, depending on the number of common leaves obtained from the initial query (3), and the number of primitive correspondences (5).

5.3 On-device runtime

In Table 1, the runtimes for the different steps of the scene recognition and pose estimation process are shown. We implemented our algorithm on a PC with an Intel Core i7-7700HQ CPU at 2.80 GHz and 16 GB RAM, a third generation iPad Pro and a HoloLens 2. We saved all the required data in binary, yet uncompressed files. The total file size of the 25 anchors is 569 KB, the centers of the tree use 15 MB of storage, the file size of the d_i vectors is 100 MB, 4 MB for the ω_i weights and 400 KB for the descriptor-to-leaf index.

Table 1 shows that loading all the files takes a lot of time, however, this is on the one hand due to a naive implementation, on the other hand it is a one time operation upon start-up. The rest of the steps take only a few milliseconds, which gives an instant localization and user experience. The individual results for the 25 rooms are given in the supplemental material.

5.4 Perceived localization accuracy

Multi-user applications in AR require reasonable accuracy regarding tracking and localization. The bigger the displacement among the participants, the harder it becomes to have useful interactions. For example, a shared virtual object of interest can be manipulated by either of the users. In a perfect world, the virtual object's pose would be precisely shared among all participants. However, tracking systems and algorithms are not perfect; therefore, we expect a displacement of a virtual object among the participants. This displacement becomes even more noticeable when a world-registered digital twin of a physical object is shown. For densely packed real-world objects (*e.g.*, cable sockets in a network switch), even small displacements can cause profound disagreement between human participants.

The internal setup of the HL2 is tuned to generate and maintain a representation of the environment at first sight. Although this representation is altered and updated over time, it is persistent in memory across device and application restarts. Alterations of the reconstruction over time also result in updates of geometry detected through the Scene Understanding SDK. Consequently, our scene representation looks slightly different if captured at different points in time. Contrary to the HL2, the iPad creates an environment model ad hoc, which is static once captured, and does neither alter nor receive updates to detected geometric primitives as long as the application is not closed. Again, since iPad creates a reconstruction anew every time, our scene representation also looks slightly different every time the application is started. In summary, neither the underlying reconstructions nor the created geometric scene representations are deterministic.

We desire pose estimation to be close to perfect (*i.e.*, positional error below 1 cm and angular error below 1°), provided the geometry of models resembles the real world structures close enough. According to several studies [5], HoloLens 2 and iPad achieve a reconstruction accuracy of ± 2 cm, and the internal tracking quality of both devices is also limited to ± 2 cm. Overall, the expectation for the perceived error between two users is therefore about 4 cm *per device*. We challenge this assumption by evaluating the perceived localization accuracy using BOWA on three HoloLens 2. Participants were asked to place a virtual sphere at the tip of a physical cone within one of our office scenes, as shown in Figure 10. This procedure was repeated ten times.

In Figure 11, left, the resulting placement error is visualized confirming our initial expectations. The average displacement error for homogeneous HL2 configurations is 3.72 cm, while, for HL2-iPad configurations, it is about 7.56 cm with a maximum displacement of 13.47 cm in a single case. Detailed numbers for all runs are given in the supplementary material. In Figure 11 on the right, a picture of a similar run using HL2 and iPad is shown, taking the image of the iPad with the HL2 to have augmentations from both devices in one picture. Note that, during our tests, the correct scene was detected, respectively, the localization was successful in all 50 out of 50 evaluation runs.

6 Application Examples

In this section, we present application examples which demonstrate the versatility of BOWA across multiple AR platforms, namely, a recording tool, a collaborative object manipulation tool, and a collaborative chess game. All of our examples use Unity software and our BOWA library to create, respectively, use the framework for model creation and instant global localization.

6.1 Scene recording and modeling

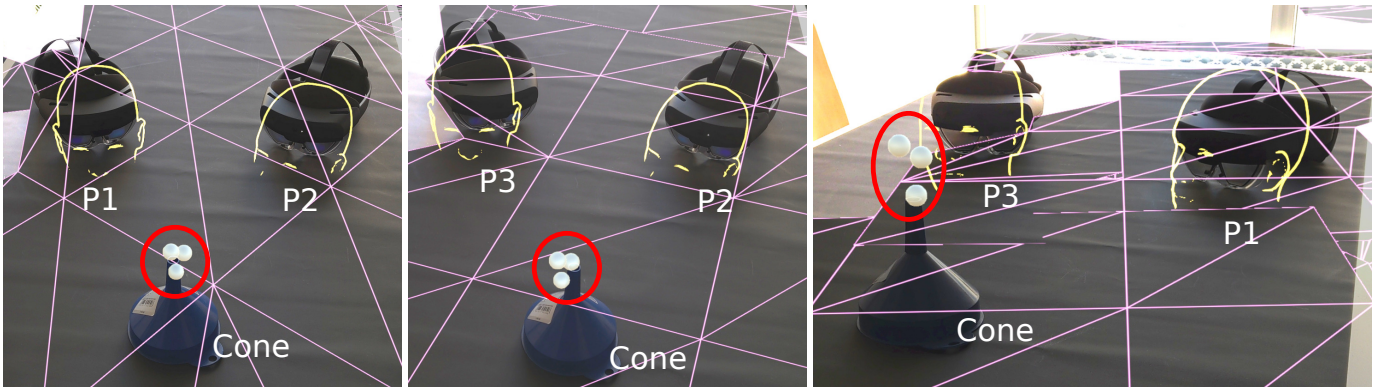
In order to collect a compelling set of environments, we implemented a tool to record BOWA descriptors of a scene and verify new BOWA models in terms of their localization accuracy. This tool runs on all platforms, *i.e.*, Android, iOS and HoloLens 2. Figure 2 depicts the application running on those platforms within the same physical environment. As opposed to the 2D UI on the iPad, on the HoloLens 2 the interaction is triggered by voice commands. All models used for offline experimental evaluations were acquired using the application on individual platforms. The application is also used to repeatedly test the localization and 6DOF pose estimation.

6.2 Collaborative 3D object design

Our second example is a collaborative 3D object manipulation tool. It heavily relies on remote Javascript function invocation within Unity software using the RagRug [16] toolkit for AR development. RagRug provides an infrastructure to develop location-aware, distributed applications. The backbone of the framework is formed by MQTT and a local server-side Node-RED instance to span an augmented space shared by multiple users. As there is no exchange of environment information (*i.e.*, imagery, 3D scene captures or even BOWAs) between devices, there is no sensitive data transmitted whatsoever. The core functionality of RagRug in this scenario is to provide 3D model resources for download, to allow collaborative management and handling of 3D objects in the shared space, and to broadcast information about the poses of other users sharing the space in real time[2]. The underlying concept is that the virtual models are part of the persistent space spanned by RagRug, as opposed to the clients having the responsibility of synchronizing local changes across devices.

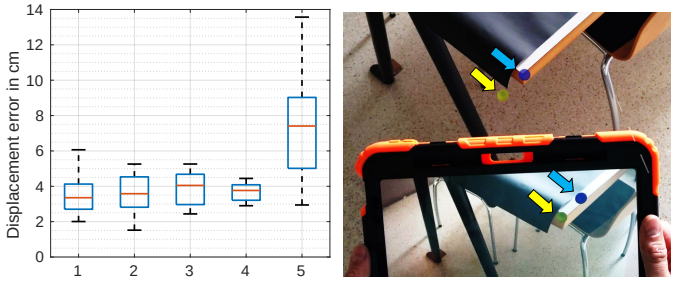
First, several users are registered within the same space based on our BOWA approach. Then, virtual objects are shared between co-present users upon creation. Each participant can manipulate every object's pose. Without external infrastructure to establish inter-participant registration, such features can be expensive to build and maintain, sometimes requiring custom hardware [19]. However, we rely on the matching of BOWA to seamlessly register the participant's spaces. Some exemplary snapshots of this application are shown in Figure 12.

[2]A server-less setup using BOWA with similar functionality could also be implemented by replacing RagRug with peer discovery and subsequent peer-to-peer communication, as well as pre-loaded resources on each device.



(a) (b) (c)

Fig. 10: Multi-user displacement evaluation. We asked three participants (P1, P2, P3) to place a measuring sphere on the tip of a cone to evaluate the spatial displacement. Each sphere is shared among each virtual space. (a) shows the scene from participant's P3 perspective, (b) from the point of view of P1, and (c) from the point of view of P2. The red circle depicts the three measuring spheres. The actual measured displacements amount to $\Delta\overline{P1P2} = 2.41$ cm, $\Delta\overline{P2P3} = 4.11$ cm and $\Delta\overline{P1P3} = 5.63$ cm, where the overall mean is 4.03 cm. Multiple runs of this evaluation follow the same trend within our office test scenes. The measurement was taken approximately at 1 m distance from the local origins.



(a) (b)

Fig. 11: Perceived localization accuracy. (a) Boxplot of the displacement between devices: 1. $HL2_{1-2}$, 2. $HL2_{1-3}$, 3. $HL2_{2-3}$, 4. Mean of 1 to 4, 5. iPad-HL2. (b) Snapshot taken with the HL2 (blue) of the iPad (yellow) to visualize the perceived displacement between the former and the latter.

6.3 Virtual chess

As a last example, we implemented a persistent chess game (Figure 13) using animated 3D models on top of RagRug and Chess.js[3]. On the one hand, it is inspired by the original Battle Chess game from the late 1980s[4] using animated and fighting virtual characters. On the other hand, concerning the interaction aspect, it picks up the concept of playing chess games in public, as contained in several TV series and theater movies located, for example, in New York's Central Park [2].

Multiple users register in the same space using the underlying BOWA framework. A virtual stand-up chess game is placed on a freely accessible table, and every participant who shares the space can just walk up to the table and take a turn. There is no given assignment of black and white to individual participants; however, rules about the movement of characters and color alternation are enforced. Because the management of the game is done within RagRug, remote players can use a (non-AR) browser interface to manipulate the game state.

7 Discussion and Conclusion

In this work, we presented an approach to perform instant large-scale localization and pose estimation based on geometric primitives, *i.e.*, planes. An important aspect of our work is that it works cross-platform on state-of-the-art mobile devices like Lidar-iPad or HoloLens 2, and that it is – to the best of our knowledge – the first approach enabling versatile, mobile, light-weight and cross-platform sharing of AR experiences without vendor lock-in. The performance of our method is also confirmed through our extensive experimental evaluation and the sample applications presented.

[3] https://github.com/jhlywa/chess.js
[4] https://en.wikipedia.org/wiki/Battle_Chess

Extension to Spheres and Cylinders While we have not used spheres and cylinders in our experiments, it is worth noting that these primitives are still supported in BOWA. Both primitive types can be included by putting additional descriptor validation rules in place. Including cylinders is trivially possible by redefining Equation 3 to contain the surface area of a cylinder and changing Equation 4 to define the aspect ratio as the ratio between the diameter and height of a cylinder. The use of spheres is slightly more involved, as it requires not only modifications to Equation 3 and Equation 4, but also obtaining a known orientation. The issue can be resolved by measuring the gravity vector and inferring the main orientation of a scene (*e.g.*, from walls and floor). Under these assumptions, all other dimensions of $\mathbf{F}_{\text{BOWA}}$ can be inferred as well.

Extension to other Platforms Although originally planned, Meta did not include depth-sensing hardware in the Meta Quest Pro. Therefore, core features, such as plane detection in pass-through mode, are not available. Snap's newest generation AR spectacles feature plane detection; however, because of the lock-in to Snap's Lens Studio and privacy concerns, sharing any kind of information (*i.e.*, even loading a model over a network connection) is prohibitive. The only plausible candidate to extend our platform to at this point in time are the Magic Leap 2 headsets, which have also been evaluated. For more details about this evaluation, the interested reader is referred to the supplementary materials. With the drive towards deep-learning-based approaches, we expect more and more features to become available on mobile devices in general. This might include object recognition as well as some semantic segmentation, which we could leverage some time in the future for our approach. As it stands, a major limitation of BOWA is the lack of support for geometric primitive detection on platform levels. Examples of the use of cylinders using the original scenes from CWA [38] can be found in the supplementary material.

Privacy Following an ongoing discussion, it seems that primarily the use of image materials and other metadata, such as WIFI coverage, IMEI numbers or GPS information, might cause issues concerning the privacy of users. Similarly, motion patterns of the user or body parts might be a source of user identification, as discussed *e.g.*, in the work of Liebers *et al.* [27]. The scene representation in BOWA consists of geometric primitives only. Even reconstructing the models gives only little insight into the interior of a space, as can be seen from the examples shown in Figure 7. The localization approach overall does not use any image material or other metadata to prune the search space, nor does it use any kind of information about the user motion or interaction.

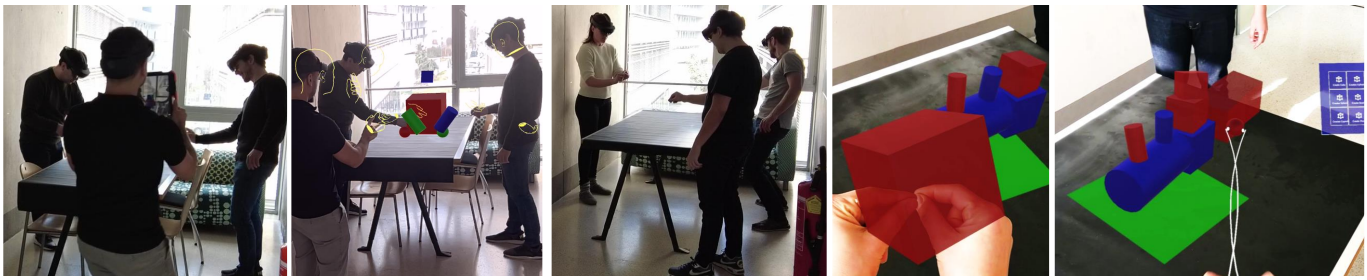


Fig. 12: Three participants use HoloLens 2 devices to collaboratively create the sculpture, while a fourth participant is using the Lidar iPad to record the actions. All participants move within a shared and localized 3D world in real-time, receiving updates of poses and objects accordingly.

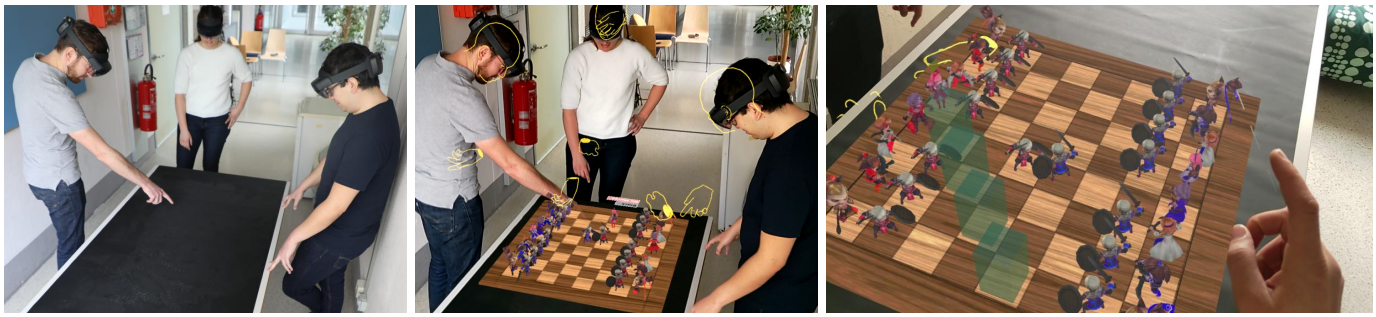


Fig. 13: A virtual chess game with animated characters is placed on a freely accessible table, while all surrounding participants sharing the space can interact with the individual characters on the board. Note that we only enforce the alternation of black and white, as well as the validity of moves with respect to the rules, but do not assign player roles to the individual devices, respectively participants.

Acknowledgments

The authors wish to thank Christina Gsaxner and Georg Krispel for their support. This work was enabled by the Competence Center VRVis. VRVis is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency under the scope of COMET - Competence Centers for Excellent Technologies (879730) managed by FFG. This work was also supported by the European Community's Horizon Europe program under grant agreement no. 101092861 ($THEIA^{XR}$), coordinated by Martijn Rooker, TTControl (TTC), Vienna, Austria.

References

[1] Apple – ARWorldMap. https://developer.apple.com/documentation/arkit/creating_a_multiuser_ar_experience. Last Accessed: 2023-02-17. 2

[2] Chess – Central Park. https://www.centralpark.com/things-to-do/sports/chess-checkers/. Last Accessed: 2023-02-17. 8

[3] Google – Cloud Anchors. https://developers.google.com/ar/develop/anchors. Last Accessed: 2023-02-17. 2

[4] PTC Vuforia – Area Targets. https://library.vuforia.com/environments/area-targets. Last Accessed: 2023-02-17. 2

[5] VGis – Tracking. https://www.vgis.io/2020/04/23/2020-ipad-pro-does-the-lidar-sensor-improve-spatial-tracking/. Last Accessed: 2023-02-17. 7

[6] S. Aarthi and S. Chitrakala. Scene understanding - A survey. *International Conference on Computer, Communication, and Signal Processing: Special Focus on IoT (ICCCSP)*, 2017. 2

[7] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *International Conference on Computer Vision (ICCV)*, 2011. 2

[8] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(06), 2018. 2

[9] C. Arth, M. Klopschitz, G. Reitmayr, and D. Schmalstieg. Real-Time Self-Localization from Panoramic Images on Mobile Devices. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 2

[10] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide Area Localization on Mobile Phones. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009. 2

[11] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[12] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An information-rich 3D model repository. *CoRR*, abs/1512.03012, 2015. 2

[13] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 3

[14] R. Dubé, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena. SegMap: Segment-based mapping and localization using data-driven descriptors. *The International Journal of Robotics Research*, 39(2–3), 2020. 2

[15] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6), 1981. 2

[16] P. Fleck, A. Sousa Calepso, S. Hubenschmid, M. Sedlmair, and D. Schmalstieg. RagRug: A toolkit for situated analytics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 29(7), 2022. 7

[17] S. Garg, T. Fischer, and M. Milford. Where is your place, visual place recognition? In *International Joint Conference on Artificial Intelligence*, 2021. 2

[18] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-time RGB-D camera relocalization. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013. 2

[19] K. Huo, T. Wang, L. Paredes, A. M. Villanueva, Y. Cao, and K. Ramani. SynchronizAR: Instant synchronization for spontaneous and spatial collaborations in augmented reality. In *User Insterface Software and Technology Symposium*. Association for Computing Machinery, 2018. 7

[20] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[21] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(11), 2015. 2

[22] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *International Conference on Computer Vision (ICCV)*, 2015. 2

[23] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 2

[24] W. Li, S. Yu, C. Wang, G. Hu, S. Shen, and C. Wen. SGLoc: Scene geometry encoding for outdoor LiDAR localization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[25] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. GlobFit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4), 2011. 2

[26] K. Lianos, J. L. Schönberger, M. Pollefeys, and T. Sattler. VSO: Visual semantic odometry. In *European Conference on Computer Vision*, 2018. 2

[27] J. Liebers, U. Gruenefeld, and S. Schneegaß. Identifying users by their hand tracking data in augmented and virtual reality. *International Journal of Human-Computer Interaction*, 0(0), 2022. 8

[28] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz. PlaneRCNN: 3D plane detection and reconstruction from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[29] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa. PlaneNet: Piecewise planar reconstruction from a single RGB image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[30] D. Liu, Y. Cui, L. Yan, C. Mousas, B. Yang, and Y. Chen. DenserNet: Weakly supervised visual localization using multi-scale feature aggregation. *AAAI Conference on Artificial Intelligence*, 35(7), 2021. 2

[31] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel. A comprehensive survey of visual SLAM algorithms. *Robotics*, 11(1), 2022. 2

[32] A. Merzlyakov and S. Macenski. A comparison of modern general-purpose visual SLAM approaches. In *International Conference on Intelligent Robots and Systems*, 2021. 2

[33] S. Mokssit, D. B. Licea, B. Guermah, and M. Ghogho. Deep learning techniques for visual SLAM: A survey. *IEEE Access*, 11, 2023. 2

[34] 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), 2017. 2

[35] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 2

[36] L. Nicholson, M. Milford, and N. Sunderhauf. QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM. *IEEE Robotics and Automation Letters*, 4(1), 2018. 2

[37] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2006. 1, 2, 4

[38] F. Reyes-Aviles, P. Fleck, D. Schmalstieg, and C. Arth. Compact world anchors: Registration using parametric primitives as scene description. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2022. 1, 2, 5, 8

[39] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[40] D. Schmalstieg and T. Höllerer. *Augmented Reality: Principles and Practice*. Addison-Wesley Professional, 2016. 2

[41] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[42] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *International Conference on Computer Vision (ICCV)*, 2003. 4

[43] C. Sommer, Y. Sun, E. Bylow, and D. Cremers. Primitect: Fast continuous hough voting for primitive detection. In *International Conference on Robotics and Automation (ICRA)*, 2020. 2

[44] A. Stanescu, P. Fleck, D. Schmalstieg, and C. Arth. Semantic segmentation of geometric primitives in dense 3D point clouds. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018. 2, 3

[45] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. LoFTR: Detector-free local feature matching with transformers. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[46] Z. Teed and J. Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. *Advances in Neural Information Processing Systems (NIPS)*, 34, 2021. 2

[47] H. Wuest, T. Engelke, F. Wientapper, F. Schmitt, and J. Keil. From CAD to 3D tracking - enhancing & scaling model-based tracking for industrial appliances. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2016. 2

[48] Y. Xia, Y. Xu, S. Li, R. Wang, J. Du, D. Cremers, and U. Stilla. SOE-Net: A self-attention and orientation encoding network for point cloud based place recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[49] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song. Semantic SLAM based on object detection and improved octomap. *IEEE Access*, 6, 2018. 2

[50] C. Zou, A. Colburn, Q. Shan, and D. Hoiem. LayoutNet: Reconstructing the 3D room layout from a single RGB image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2