Representative Feature Descriptor Sets for Robust Handheld Camera Localization

Daniel Kurz* Thomas Olszamowski† Selim Benhimane‡

metaio GmbH

Figure 1: Our approach finds representative feature descriptor sets based on matches between synthetic views of an object (left). The resulting descriptor set improves feature-based localization in arbitrary orientations (center). In addition, we present a gravity-aware version of the proposed method tailored for the localization of objects at a known orientation with respect to gravity, such as an augmented game board (right).

Abstract

We present a method to automatically determine a set of feature descriptors that describes an object such that it can be localized under a variety of viewpoints. Based on a set of synthetically generated views, local image features are detected, described and aggregated in a database. Our proposed method evaluates matches between these database features to eventually find a set of the most representative descriptors from the database. Using this scalable offline process, the localization success rate is significantly increased without adding computational load to the runtime method. Moreover, if camera localization is performed with respect to objects at a known gravity orientation, we propose to create multiple reference descriptor sets for different angles between the camera's principal axis and the gravity vector. This approach is particularly suited for handheld devices with built-in inertial sensors and enables matching against a reference dataset only containing the information relevant for camera poses that are consistent with the measured gravity.

Comprehensive evaluations of the proposed methods using a large quantity of real camera images, a variety of objects, different cameras and different kinds of feature descriptors confirm that our approaches outperform standard feature descriptor-based methods.

1 Introduction and Motivation

In the last years, Augmented Reality (AR) became increasingly popular where the main reason for its success is the pervasiveness of camera-equipped handheld devices such as mobile phones. The range of available applications is nowadays manifold. Location-based AR browsers overlay information on points of interest (POIs) in the camera image based on GPS, inertial sensors and a digital compass. Other applications use fiducials, such as markers or QR-codes, to recognize objects and potentially determine their pose for providing registered overlays in the camera image. Recently, handheld AR applications also started using 3D natural feature tracking [9] which enables the localization and tracking of arbitrary 3D objects in real-time. However, the majority of commercial handheld AR applications performing visual tracking use a planar object as tracking reference, such as magazines, packaging boxes, or a board for AR-enabled games, as shown in Figure 1 (right).

Our proposed method requires a model of the object we aim to localize which enables creating synthetic views of it. For a planar object a fronto-parallel image of the object is sufficient, while for general 3D objects, a textured model of these is needed. While the presented method is universally applicable, for simplicity, the paper will explain and evaluate the method for planar objects where synthetic views can be generated using simple image warping.

A common approach to localize the camera with respect to an object with a known geometry and visual appearance uses 2D-3D correspondences gained by means of local feature descriptors, such as SIFT [16], that are extract both from the current camera image and the underlying reference model. As the reference descriptors of a planar object are usually based on a fronto-parallel image, localization algorithms work best for this viewpoint and the localization rate decreases with steeper camera angles. In addition to appearance changes as a result of perspective distortions, also aliasing artifacts may lead to a decreased performance for viewpoints different from the reference view. However, in particular in consumer applications, it is important that the tracking works even under steep angles. Therefore, we propose a method that significantly improves camera localization as needed for video-see-through AR.

Based on synthetic views of the real object, we present a general approach for determining a set of representative feature descriptors which can be used with arbitrarily oriented objects. Thereafter, we extend this approach tailored for handheld AR applications where the object has a static and known orientation with respect to gravity and inertial sensors allow for measuring the gravity vector in the camera coordinate system. Our evaluation results attest our approaches lead to clearly increased localization rates for out-of-plane rotations of planar objects without increasing the amount of data or increasing the complexity during runtime.

*e-mail:daniel.kurz@metaio.com
†e-mail:thomas.olszamowski@metaio.com
‡e-mail:selim.benhimane@metaio.com

2 Related Work

Visual camera pose localization for handheld AR usually uses either feature descriptors, e.g. SIFT [16], or feature classifiers, e.g. FERNS [18], to gain point correspondences. On mobile phones, simplified versions of both have been used in real-time applications to localize the camera with respect to planar objects [20].

PTAM [9] uses feature descriptors to estimate the camera pose with respect to an unknown potentially three-dimensional environment, which is mapped, i.e. reconstructed, in parallel. For 3D objects with a known geometry and appearance, approaches exist that use a textured CAD model to online create synthetic views to match against for frame-to-frame tracking [2]. Thereby, inertial sensors are used to provide a rough orientation for initialization and to aid frame-to-frame tracking during fast camera movements.

In contrast, our method creates synthetic views of the object that we want to track in an offline stage. During runtime, we solely rely on a static set of representative reference feature descriptors resulting from our proposed method.

Feature descriptors are designed to enable matching of corresponding features in different images of the same object or scene. Thereby it is crucial that the descriptor is invariant to changes in the camera viewpoint. Invariance to scale is usually achieved by means of image pyramids that represent the camera image in scale-space, e.g. [16]. To be invariant to in-plane rotations, one or multiple orientations are computed for every feature. This orientation assignment can either be based on image intensities in the neighborhood of the feature [16] or based on inertial sensor data, if the orientation of the object or scene with respect to gravity is known [11].

Different approaches try to detect affine regions [17] in an image which in addition to a position provide an anisotropic scaling for features to account for the effect of perspective foreshortening resulting from out-of-plane rotations. The determined size and orientation of a feature is eventually used to normalize the image patch around the feature before description.

If auxiliary information is available, this again can be used to compensate for out-of-plane rotations. Provided with the depth of the camera pixels, the 3D normal vector of a feature can be determined to create a viewpoint-invariant patch [21] of the feature. For horizontal surfaces, the gravity vector measured with inertial sensors enables the rectification of the camera image prior to feature description [10]. Additionally, the descriptor layout should be designed to be invariant to changes up to a certain extent. Approaches exist that employ learning to find ideal feature descriptor layouts within a defined design space [3] based on a ground truth dataset containing corresponding image patches of features under greatly varying pose and illumination conditions.

If such data is not available, rendering techniques, such as image warping, can be employed to create a multitude of synthetic views of a feature (or object). For descriptors providing a low invariance to viewpoint variations or in-plane rotations but enabling very fast descriptor matching, such warpings are used to create different descriptors for different viewpoints and/or rotations to support larger variations [4]. Other approaches use synthetic views to determine repeatable keypoints and then compute a custom descriptor which is based on the empirical distributions of normalised intensities of the pixels around the keypoint over multiple observations [19]. Image retrieval methods can benefit from such synthetic views by creating a specific scalable vocabulary tree for every canonical viewpoint [5]. Query images can then be processed in parallel with all trees on a multiprocessor server followed by a selection of the best match across all trees.

Our proposed method also first creates reference descriptors for a variety of (in-plane and out-of-plane) rotations using synthetic views. However, instead of keeping all of them or combining them, we automatically determine the most representative subset of these feature descriptors and finally only use this as reference. This not only results in faster matching, but also reduces the amount of data that needs to be downloaded from a server, e.g. in AR browsers, such as junaio[1]. Also our approach is independent of the features and descriptors considered and it is even general enough to be applied on feature classifiers that are explained later.

Different methods aim at camera localization based on structure from motion 3D point clouds. For every 3D point there usually exist multiple descriptors resulting from the images used for reconstruction of this point. As these descriptors are highly redundant, they can either be clustered [8] or averaged [14] to reduce the amount of reference data. To further reduce the dataset and speed up matching, greedy approaches to an NP-hard set cover problem are used. If an image retrieval step is used to initialize the localization [8] the approach tries to find the minimal subset of views, that covers the 3D point cloud. Another approach aims at finding a minimal subset of points that covers all camera images [14].

Compared to these methods, our approach does not attempt to solve a set cover problem. We are not interested in a minimal set of features that covers a set of views, but in a subset of a given size that provides the most representative description of an object. Thereby, we do not cluster or average descriptors belonging to the same physical point but simply let the algorithm decide how many and which descriptors for a certain feature result in the most matches and therefore are expected to provide best localization results.

Feature classifiers also aim to identify for a given image feature the corresponding reference feature in a database. This can be formulated as a classification problem, where every reference feature is a class, and the classifier determines the class with the highest probability for a given current feature. An offline training phase is required, where the classifier is trained with different possible appearances of a feature, usually gained by randomly warped patches. Randomized Trees [13] use these to estimate the probabilities over all classes for every leaf node, while the inner nodes contain binary decisions based on image intensity comparisons. Ferns [18] are also based on binary intensity difference tests. However, instead of a tree, they use a non-hierarchical structure which provides better scalability. While the previous methods aim at training all possible visual appearances of a feature as a single class, which provides invariance to changes in the viewpoint, other approaches [7] train the appearances from different viewpoints as different classes. Thereby, the classifier is not only able to determine the identity of a feature, but also its pose.

Thanks to the training stage provided with different appearances of a feature, classifiers in general provide a good invariance to out-of-plane rotations. However, the probabilities require a lot of memory, which makes them unfeasible for a large amount of features on mobile devices. Our approach also benefits from different synthetically created appearances of a feature in an offline stage, but does not increase the memory consumption.

3 Proposed Method

For arbitrary objects, our proposed method requires a textured 3D model allowing for the creation of synthetic views of the object to be localized. For planar objects, a fronto-parallel image of the object is fully sufficient and synthetic views can be created using image warping. Therefore, we will explain and evaluate our approach for planar objects, while it is universally valid.

Given a reference image $I(u,v)$ of the object, we use image warping to create a set of synthetic views $I(\mathbf{H}_i(u,v))$, as illustrated in Figure 2 on the right. The perspective warpings we use correspond to images taken by virtual cameras that are located on a hemisphere centered around the planar object. To ensure a uniform sampling, we use the vertices of IcoSpheres as camera locations, similarly as in [7]. Figure 3 displays three such IcoSpheres with different resolutions, i.e. number of virtual cameras. The camera's view vector

[1] http://www.junaio.com

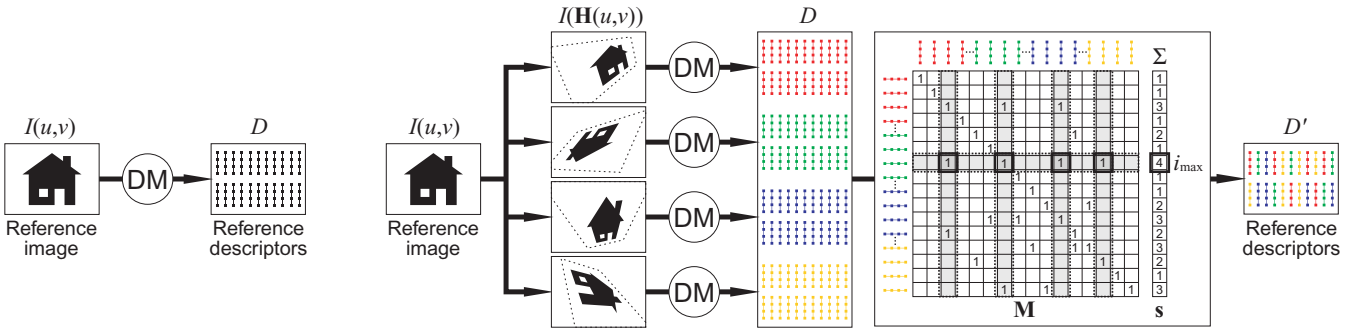

Figure 2: Workflow of the proposed off-line process. The regular approach (left) uses one image and creates descriptors for it. Our proposed method (right) first creates warpings, then descriptors for every warping, then matches them and based on that selects a subset of the descriptors.

is then defined to look at the object center and the up-vector initially is the normal of the object. Based on these two vectors, an orthonormal coordinate system is defined representing the orientation of the virtual camera. If the intrinsic parameters of the real camera that will be used are known, they are used to eventually compute the homography $\mathbf{H}_i$ corresponding to the virtual camera pose. Otherwise, an assumption of the intrinsic parameters is used. The resulting images are shown for a "Peach" photo in Figure 6. For each such image, we detect and describe image features with a description method (DM) and aggregate all descriptors d_i together with the index of the corresponding view v_i in a set of descriptors D and view indices V.

Now, we are looking for a subset $D' \subset D$ of the descriptors in the database that provides a sufficient amount of matches among all synthetic views. Therefore, we first match every descriptor $d_i \in D$ against all subsets $D_j = \{d_k | v_k = j\}$ of descriptors from every synthetic view. A match connects a descriptor with its nearest neighbor if the second nearest neighbor is sufficiently far away as in [16]. After having matched all descriptors in the database, we discard all wrong matches, i.e. where the feature position warped back to the original reference image differs by more than $\sqrt{1.5}$ pixels. For all remaining (correct) matches, we update the feature positions as the average over all matched features, which results in a sub-pixel precise position that showed to improve localization results. The matches are then expressed as a binary $(n \times n)$ matrix $\mathbf{M}$, where $n = |D|$ is the overall number of descriptors. The matrix entry $\mathbf{M}_{i,j}$ carries a 1 if the i-th descriptor has been matched with the j-th descriptor and otherwise a 0. Now let the vector $\mathbf{s}$ contain the sum of every row, i.e. $\mathbf{s}_i$ represents the number of matches for d_i.

The only parameter our method needs is the desired size of the final set of feature descriptors f. While the number of descriptors we have in our initially empty set D' is less than this number, we determine the index of the best descriptor (with the most matches) in the database as $i_{max} = \text{argmax}_i(\mathbf{s}_i)$. The i_{max}-th descriptor is then added to the final set of descriptors $D' = D' \cup \{d_{i_{max}}\}$. Afterwards, our proposed method sets the i_{max}-th row to zero, such that $d_{i_{max}}$ cannot be picked as best descriptor again. Now that $d_{i_{max}}$ has been added to the set of representative descriptors D', we consider those descriptors that $d_{i_{max}}$ has been matched with, as covered by D'. Therefore, we do not need any additional descriptors in D' which also match against these descriptors. By setting all columns j of $\mathbf{M}$ that have a non-zero element in the i_{max}-th row to zero, we remove these matches from $\mathbf{M}$. Thereby, our method scales well with an increasing number of synthetic views. After updating $\mathbf{s}$, the procedure is repeated until the desired number of descriptors is reached, i.e. until $|D'| = f$. Figure 2 illustrates the proposed method in the right subfigure for a planar object. The resulting representative feature descriptor set D' is finally used in the same way as D in the regular approach shown in the left of Figure 2.

It seems natural, that when matching the feature descriptors of a live camera image against the set of reference descriptors, the best matches will involve those reference descriptors that originate from a virtual camera pose similar to the one of the real camera. To empirically confirm this correlation, we divide the camera poses on IcoSphere4 in six different bins, as shown color-coded in Figure 3, and store with every reference descriptor the corresponding view bin. For a sequence of real camera images taken at increasingly steep angles, we plot the histogram of the view bins of those reference features that were successfully matched as inliers. As can be seen, the steepness of the used reference views clearly correlates with the steepness of the current camera. This finding leads to the gravity-aware approach which will be explained in the following.

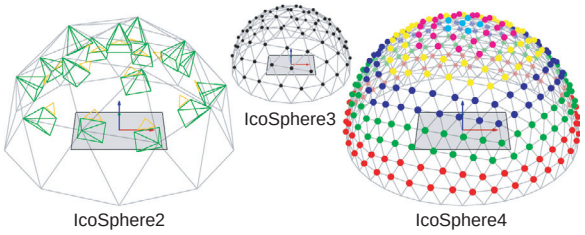

Figure 3: The three IcoSpheres whose vertices we use for uniformly sampling 16 (IcoSphere2), 71 (IcoSphere3) or 301 (IcoSphere4) camera positions on a unit hemisphere.

4 Gravity-Aware Method

Many handheld AR applications are using objects with a known orientation with respect to gravity for camera localization. Examples include magazines or game boards lying on a desk, navigation prints on the floor, vertical billboards and posters, building façades for large-scale outdoor AR or cars for marketing applications.

Off-the-shelf handheld devices are usually equipped with inertial sensors which provide a measurement of the normalized gravity vector $\mathbf{g} = [g_x, g_y, g_z]^\top$ in the camera coordinate system. Based on this vector, the angle between the camera's principal axis and the gravity can be computed as $\gamma_c = \cos^{-1}(-g_z)$. As this angle is correlated with the corresponding angle γ_r of those synthetic reference views that provide the most inlier matches, cf. Figure 4, we aim to narrow the set of reference descriptors to match against based on this angle. We propose to create multiple representative feature descriptor sets D'_g for different camera orientations with respect to gravity, i.e. for certain ranges of γ_r. During runtime, we then only use the reference descriptor set that corresponds to the current measured camera orientation angle γ_c. Thereby, the same amount of reference descriptors to match against can contain much

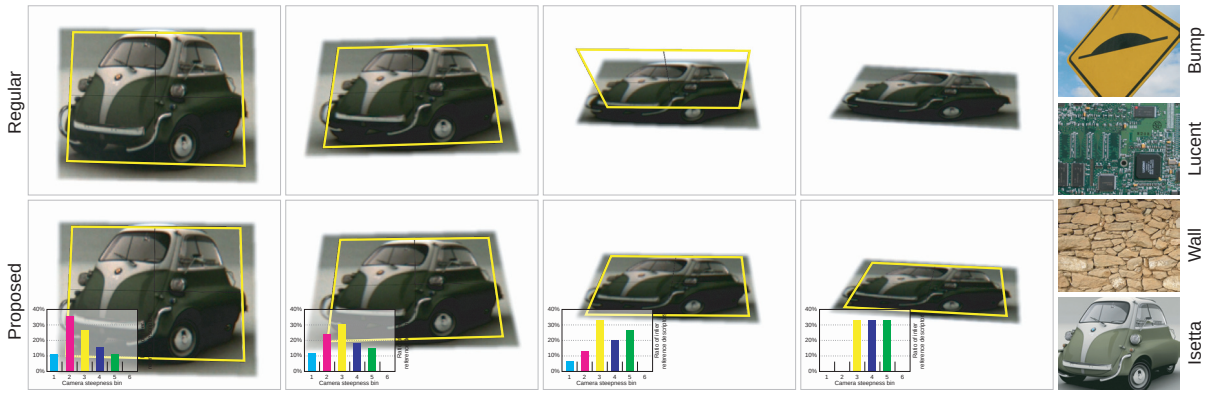

Figure 4: Four frames of the evaluation dataset and the localization result of the regular approach (upper row) and our proposed method (lower row). In particular for steep camera angles, our method provides better results than the regular method. The histograms visualize the distribution of the view bins of the reference features that led to the localization, which is clearly correlated with the steepness of the camera.

more descriptors representing the object in an orientation similar to the one of the camera. For the camera positions sampled using IcoSphere4, cf. Figure 3, we create six view bins which are visualized with different colors. As in the original approach, we create feature descriptors for all virtual cameras. The matching stage is then carried out for every view bin g, by matching all descriptors belonging to the g-th bin against all descriptors from all bins. The resulting matching matrix $\mathbf{M}$ is a $(m \times n)$ matrix where m is the number of descriptors in the g-th view bin and n is the overall number of descriptors in all bins. The algorithm then proceeds as described in section 3 by iteratively finding the best descriptor, adding it to D', removing the related matches from $\mathbf{M}$ and updating $\mathbf{s}$ until the desired number of descriptors is reached. For every live camera image, we then compute γ_c based on the measured gravity vector and only match against the reference descriptor set with the average $\overline{\gamma_r}$ closest to γ_c.

We found empirically, that our method provides better results when computing the matches of the descriptors of a view bin with all descriptors from all bins than matching only within each view bin. Thereby, a descriptor is considered good not only if it can be matched with many descriptors within the same bin, i.e. gravity angle range, but also with descriptors from different bins which helps avoiding aliasing problems at the borders between two bins.

In comparison with Gravity-Rectified Feature Descriptors (GREFD) [10], this approach does not require warping the current camera image, which can be expensive on handheld devices. Instead we create warpings of the reference image in an offline step. More importantly, our proposed gravity-aware method can be applied to surfaces at arbitrary but known orientations with respect to gravity, while GREFD is limited to horizontal surfaces.

5 Evaluations and Results

In order to evaluate the impact of our proposed methods, we compare the success rate of a template localization algorithm using four different feature description methods. The "Regular" version detects and describes both reference and current features in the provided images without any additional processing. The "GREFD" version uses regular feature detection and description for the reference image. However, for every current camera image it uses Gravity-Rectified Feature Descriptors [10] that rectify the camera image based on gravity measurements before detecting and describing features. Against these baseline versions, we compare our proposed method in two different configurations. The "Proposed" version uses regular descriptors for the current camera images but employs the approach explained in section 3 with the views of IcoSphere4 to determine a set of representative feature descriptors for the reference image. Finally, our "Proposed Gravity" method creates six of such representative feature descriptor sets for the view bins of IcoSphere4 as discussed in section 4. We compare two kinds of feature detectors and descriptors, namely SURF [1] using the OpenCV implementation, where the descriptor entries have been linearly transformed from the range $[-1.0, 1.0]$ to integers in $[0, 255]$, and our custom 48-dimensional feature descriptor that we will refer to as "Mobile48". It is also based on histograms of image gradients but in contrast to SURF optimized to perform in real-time on mobile devices. All approaches use image pyramids to achieve scale invariance of the feature descriptors. The number of reference descriptors to match against in a single frame is fixed to 250 for all methods, descriptors and tests.

Despite the feature detection and description, all versions use the same procedure for template localization. For every reference feature descriptor, the nearest neighbor among all current feature descriptors is determined using exhaustive search. If the ratio between the distance to the nearest and the second nearest neighbor is below a certain threshold, the two descriptors are matched. All matches, which represent 2D-2D correspondences between the reference image and the planar object in the current camera image, are used to determine a homography mapping the current image to the coordinate system of the reference image using PROSAC [6]. This homography is then refined in a non-linear optimization using all inlier matches. Finally the homography is used in combination with the calibrated intrinsic camera parameters to determine the camera pose with respect to the planar object. Note that the algorithm does not perform any frame-to-frame tracking, i.e. it does not use any information from the previous frame but re-initializes in every frame.

5.1 Benchmark with Ground Truth Poses

The results of the aforementioned template localization method can be judged in different ways, where the most reliable approach is to compare them with ground truth. To this end, we employ metaio's publicly available template localization dataset[2] and the accompanied evaluation framework [15]. The dataset consists of 48,000 images of 8 planar objects (print-outs of templates, see Figure 4) taken with an industrial camera. The pose of this camera was very precisely determined by an attached measurement arm and can therefore be considered ground truth. Since the "Proposed Gravity" and "GREFD" versions of the algorithm rely on the gravity vector, we synthesize plausible gravity measurements from the ground truth poses to simulate inertial sensors assuming the templates were located on a horizontal surface as explained in [12].

[2] http://www.metaio.com/research

For every template, there are five image sequences, namely ”Angle“, ”Range“, ”Fast Far“, ”Fast Close“ and ”Lighting“, cf. [15]. While the ”Angle“ sequences contain a good amount of steep camera angles, the other sequences do not and are therefore summarized as ”Others“. Table 1 shows for the versions of the algorithm using ”Mobile48“, the ratio of camera frames within every sequence, in which the planar object could be correctly localized. As can be seen, our proposed methods outperform the regular approach in all sequences. On average, the localization success rate increases by 21.96% over all sequences and by 33.74% for the steep ”Angle“ sequences when using the ”Proposed“ method compared to ”Regular“. For most of the sequences, ”Proposed Gravity“ gives even better results, in particular for the ”Angle“ sequences, where the localization rate increases by nearly 40% compared to ”Regular“.

However, ”GREFD“ is clearly superior for these sequences. But as the impact of ”GREFD“ is very small and sometimes even negative for the ”Others“ sequences without steep camera poses, the overall performance of both ”Proposed“ and ”Proposed Gravity“ is better than that of ”GREFD“.

In addition to the increased ratio, our proposed methods also increase the localization precision. While the average re-projection error of the four template corners is 4.08 pixels in the ”Regular“ approach and 4.12 pixels when using ”GREFD“, it decreases to 3.65 for ”Proposed“ and 3.58 pixels for ”Proposed Gravity“.

Sequence\Method using Mobile48	Regular	GREFD	Proposed	Proposed Gravity
Grass (Angle)	0.0817	0.1458	0.1375	0.1408
Grass (Others)	0.0735	0.0895	0.1138	0.1241
Wall (Angle)	0.2267	0.3308	0.2400	0.2467
Wall (Others)	0.3025	0.3031	0.3315	0.3567
Bump (Angle)	0.3567	0.4675	0.4483	0.4383
Bump (Others)	0.2854	0.3233	0.3158	0.3094
Stop (Angle)	0.3683	0.8258	0.5425	0.6358
Stop (Others)	0.5335	0.5354	0.5996	0.6031
Isetta (Angle)	0.6233	0.8492	0.8125	0.8375
Isetta (Others)	0.5833	0.5798	0.6610	0.6633
Philadelphia (Angle)	0.3242	0.4942	0.3717	0.3683
Philadelphia (Others)	0.3698	0.3547	0.4533	0.4579
Lucent (Angle)	0.2333	0.4458	0.3767	0.3983
Lucent (Others)	0.4287	0.4363	0.5552	0.5467
Mac Mini (Angle)	0.2708	0.3992	0.3942	0.3933
Mac Mini (Others)	0.3081	0.3208	0.4146	0.4156
All sequences (Angle)	0.3106	0.4948	0.4154	0.4324
All sequences (Others)	0.3606	0.3679	0.4305	0.4346
All sequences	**0.3506**	**0.3933**	**0.4276**	**0.4342**
Sequence\Method using SURF	Regular	GREFD	Proposed	Proposed Gravity
All sequences (Angle)	0.3598	0.6198	0.5151	0.5265
All sequences (Others)	0.3857	0.3923	0.4731	0.4730
All sequences	**0.3805**	**0.4378**	**0.4815**	**0.4837**

Table 1: Ratio of correctly localized frames in the localization dataset.

The results of the same evaluation using SURF are shown in table 1 and verify that in particular for the ”Angle“ sequences, localization improves significantly (+46.33%) with ”Proposed Gravity“.

The reason why ”GREFD“ is stronger than ”Proposed Gravity“ under steep angles, is that the problem for ”GREFD“ is easier. 250 descriptors can be used to match a fronto-parallel view (the reference image) with a nearly fronto-parallel view (the rectified current camera image), while our ”Proposed Gravity“ method needs to match a steep view (current image) with one out of unlimited possible views at the same gravity angle, whereof only a couple of views have been sampled in the offline stage, with the same number of descriptors. But both proposed methods are faster than ”GREFD“, as they do not require image warping during runtime.

Additionally, in contrast to ”GREFD“, both our proposed methods can deal with surfaces in arbitrary orientations while ”GREFD“ is restricted to horizontal surfaces. In order to empirically verify, that ”Proposed Gravity“ can deal with any orientation as long as it is known, we repeated the template localization test with ”Mobile48“ under the assumption that the templates were located on a vertical surface. Therefore, the definition of the gravity vector and the binning of the views on IcoSphere4 had to be adjusted. The results confirm, that in this configuration, where ”GREFD“ does not work at all, ”Proposed Gravity“ gives an overall ratio of 0.4341 correct frames, which is much better than the ”Regular“ approach.

5.2 Benchmark with Mobile Device Data

As the targeted application of our proposed methods are mobile applications running on handheld devices, we also carried out evaluations with images and inertial sensor readings provided by a mobile device (Apple’s iPad2). In these datasets, both the camera images and the gravity measurements have exactly the properties a real application running on a mobile device would use. However, there is no ground truth information available for these datasets. Instead, we compute the Zero-Mean Normalized Cross Correlation (ZNCC) between the reference image and the current image warped with the homography that led to the determined pose and assume that this similarity corresponds to the accurateness of the pose.

We evaluate the ZNCCs after localizing the four planar objects shown in Figure 6 which correspond to real objects that would actually be used in AR applications. The objects were located on a horizontal surface such that they are facing up to provide comparability with GREFD. In contrast to the image sequences used in the previous section where the area around the template is masked out, these sequences do have a background cluttered with office supplies. For each object, we capture 880 images and store them with the corresponding gravity vectors while moving the camera. Some example images with the found pose are shown in Figure 6.

The plots in Figure 5 show the number of frames for which the resulting ZNCC was above a particular threshold as a function of this threshold. While for the ”Magazine“, the regular approach is slightly better than our proposed methods, the other three objects can be localized with higher ZNCCs more often when using our ”Proposed“ method to determine representative feature descriptor sets. Incorporating the measured gravity further improves the results for ”Book“ and ”Peach“. It is obvious, that ”GREFD“ performs much better than any other method in this evaluation, but it is important to keep in mind that it also is much more expensive due to the warping of every live camera image.

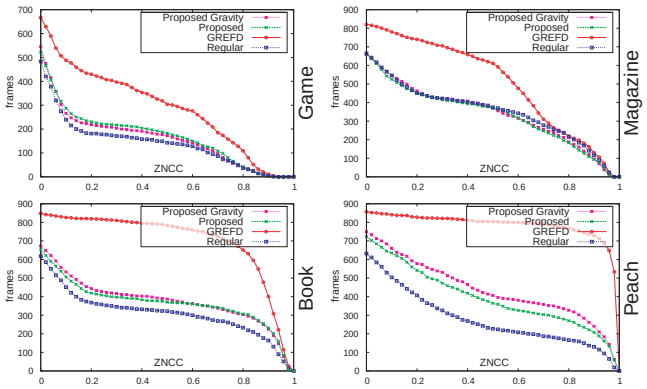

Figure 5: Localization results for horizontal templates under steep angles based on the datasets captured with a handheld device.

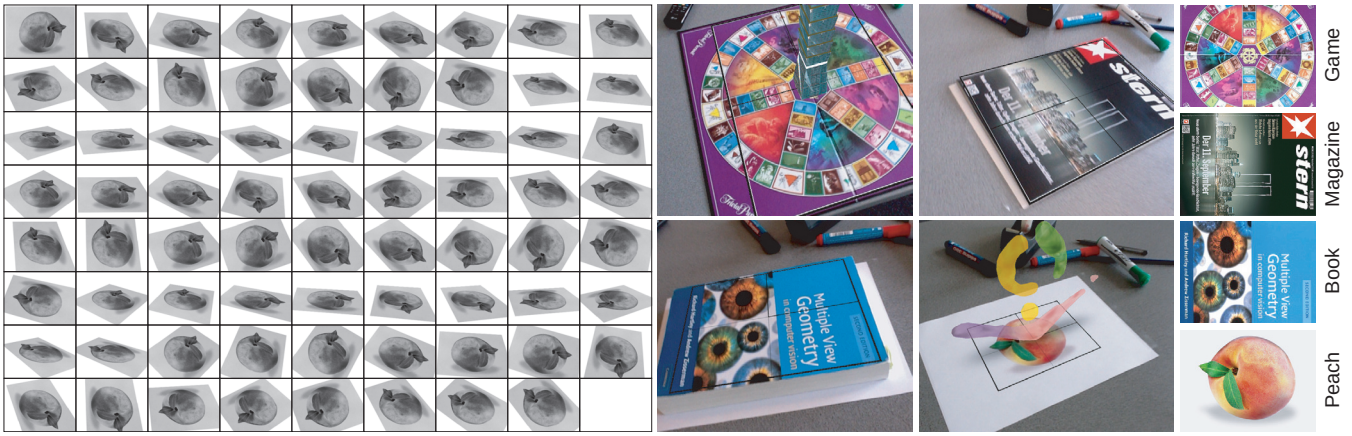

Figure 6: The 71 warpings our method creates for a reference image using the set of warpings IcoSphere3 (left). Selected frames with successful localizations of the handheld dataset (center) using the four objects shown on the right.

In our implementation, detecting and describing features from a (480×360) pixels image including conversion to grayscale and the creation of an image pyramid takes on average 31 ms on an iPhone 4. The warping needed for GREFD adds another 99 ms. This makes the feature detection and description in our proposed methods over four times as fast as that of GREFD. Additionally, GREFD is limited to horizontal surfaces while the proposed methods are not.

6 Conclusions and Future Work

We presented a method that automatically determines a set of representative feature descriptors for enabling the localization of real objects under a variety of viewpoints. Our evaluation results attest that our proposal outperforms state-of-the-art methods for feature-based template localization. The range of possible applications of the proposed method is broad as it can be used in any application using any kind of offline-created feature descriptors. Because our method is offline, it does not introduce any computational overhead nor does it increase the amount of data describing an object which is crucial for apps retrieving descriptors via mobile networks.

While all evaluations in this paper use planar objects, the same approach can be applied to 3D objects with a given model. We can also imagine using our method on keyframes for 3D Simultaneous Localization and Mapping (SLAM) or Structure from Motion (SfM) approaches. Adding randomized synthetic noise and blur to the views as in [18] can further increase the representativeness of the descriptor sets with respect to noise and defocus. Our gravity-aware approach uses the angle between the principal axis and the gravity vector. If the object to localize is planar and horizontally aligned, then the gravity angle for every feature can be computed individually which should give better results. For outdoor tracking in urban environments, the range of synthetic views can be drastically reduced assuming that the users are located on the ground and their camera is at an approximately constant height. Exploring further applications of the method will be part of our future work.

Acknowledgements

This research has been partly funded by the European 7th Framework Program, under grant VENTURI (FP7-288238).

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[2] G. Bleser and D. Stricker. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *Computers & Graphics*, 33(1):59–72, 2009.

[3] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *Trans. IEEE PAMI*, 33(1):43–57, 2011.

[4] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *Trans. IEEE PAMI*, 34:1281–1298, 2012.

[5] D. Chen, S. S. Tsai, V. Chandrasekhar, G. Takacs, J. Singh, and B. Girod. Robust image retrieval using multiview scalable vocabulary trees. In *Proc. VCIP*, 2009.

[6] O. Chum and J. Matas. Matching with PROSAC - Progressive Sample Consensus. In *Proc. IEEE CVPR*, 2005.

[7] S. Hinterstoisser, V. Lepetit, S. Benhimane, P. Fua, and N. Navab. Learning real-time perspective patch rectification. *Int. Journal of Computer Vision*, 91(1):107–130, 2011.

[8] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Proc. IEEE CVPR*, 2009.

[9] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proc. IEEE/ACM ISMAR*, 2009.

[10] D. Kurz and S. Benhimane. Gravity-Aware Handheld Augmented Reality. In *Proc. IEEE/ACM ISMAR*, 2011.

[11] D. Kurz and S. Benhimane. Inertial sensor-aligned visual feature descriptors. In *Proc. IEEE CVPR*, 2011.

[12] D. Kurz, S. Lieberknecht, and S. Benhimane. Benchmarking Inertial Sensor-aided Localization and Tracking Methods. In *Proc. TrakMark Workshop*, 2011.

[13] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *Trans. IEEE PAMI*, 28(9):1465–1479, 2006.

[14] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *Proc. ECCV*, 2010.

[15] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab. A dataset and evaluation methodology for template-based tracking algorithms. In *Proc. IEEE/ACM ISMAR*, 2009.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal on Computer Vision*, 60(2):91–110, 2004.

[17] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. Journal Computer Vision*, 65:43–72, 2005.

[18] M. Özuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. IEEE CVPR*, 2007.

[19] S. Taylor and T. Drummond. Multiple Target Localisation at over 100 FPS. In *Proc. BMVC*, 2009.

[20] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proc. IEEE/ACM ISMAR*, 2008.

[21] C. Wu, B. Clipp, X. Li, J.-M. Frahm, and M. Pollefeys. 3d model matching with viewpoint-invariant patches (VIP). In *Proc. IEEE CVPR*, 2008.