Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding

Ben Glocker, Jamie Shotton, Antonio Criminisi, and Shahram Izadi

Abstract—Recovery from tracking failure is essential in any simultaneous localization and tracking system. In this context, we explore an efficient keyframe-based relocalization method based on frame encoding using randomized ferns. The method enables automatic discovery of keyframes through online harvesting in tracking mode, and fast retrieval of pose candidates in the case when tracking is lost. Frame encoding is achieved by applying simple binary feature tests which are stored in the nodes of an ensemble of randomized ferns. The concatenation of small block codes generated by each fern yields a global compact representation of camera frames. Based on those representations we define the frame dissimilarity as the block-wise hamming distance (BlockHD). Dissimilarities between an incoming query frame and a large set of keyframes can be efficiently evaluated by simply traversing the nodes of the ferns and counting image co-occurrences in corresponding code tables. In tracking mode, those dissimilarities decide whether a frame/pose pair is considered as a novel keyframe. For tracking recovery, poses of the most similar keyframes are retrieved and used for reinitialization of the tracking algorithm. The integration of our relocalization method into a hand-held KinectFusion system allows seamless continuation of mapping even when tracking is frequently lost.

Index Terms—Camera relocalization, tracking recovery, dense tracking and mapping, marker-free augmented reality

1 Introduction

Development of systems for simultaneous localization and mapping (SLAM) has gained enormous momentum in recent years. Seminal works on MonoSLAM [1], [2] and PTAM [3] enabled real-time performance for sparse feature methods using commodity hardware. Today, we find numerous works on real-time dense tracking and mapping with RGB [4], [5] and RGB-D cameras [6], [7] which allow accurate 3D reconstruction of the physical world.

Recent advances in this area include object-level SLAM [8], [9], scalable representations [10], [11], [12], dynamic scene reconstruction [13], [14], [15], and SLAM on mobile devices [16], [17], [18]. Real-time SLAM has led to exciting applications such as environment-aware augmented reality (AR) [7], [19], [20], [21]. Obtaining knowledge about the 3D geometry of physical objects in a scene combined with the ability to sense the depth in real-time allows, for example, realistic occlusion handling and accurate fusion of real and virtual objects [22].

The underlying processing pipelines of different SLAM systems are quite similar. The camera motion is tracked in a frame-to-frame (or frame-to-model) fashion where the pose update is determined by computing a relative transformation between a (partially) reconstructed world (*i.e.* the map) and a set of features or 3D point clouds obtained from the live camera frames. For RGB-D settings, the camera transformation can be for instance computed by employing a robust version of the iterative closest point (ICP) algorithm as it is implemented in the KinectFusion approach [6]. Given the estimated camera pose, new measurements are integrated into the map yielding an updated and refined reconstruction of the scene. Existing 3D reconstruction pipelines mainly differ in the details how tracking [7], [23], [24] and mapping [10], [11], [25] are implemented.

1.1 Camera Relocalization in Real-Time SLAM

In order to acquire an accurate map of the scene, reconstruction pipelines rely on a steady stream of successfully tracked frames. Tracking failure can have severe consequences. Integrating measurements with incorrect poses yields implausible, invalid geometry and might destroy already reconstructed parts. Indeed, if tracking failure can be detected, at least map corruption can be prevented. However, in AR applications in which the pose of the camera is required to correctly overlay virtual objects onto the real world, tracking failure leads to an abrupt and unpleasant end of the user's experience.

The causes for tracking failure are versatile. Rapid camera motion and sudden change of viewpoint are probably the predominant ones where image-based camera tracking fails. In addition, and particularly relevant to systems where the map is spatially restricted to a limited area of the world, tracking which relies on the reconstructed map fails every time the camera points outside this restricted domain. This can frequently happen in AR scenarios where the user is in control of a hand-held or head-mounted camera.

To this end, it is of great practical importance to integrate a camera relocalization module which allows instant recovery from tracking failure. Incorporated into the 3D reconstruction pipeline, such a module allows seamless

- *B. Glocker is with the Biomedical Image Analysis Group, Department of Computing, Imperial College London, SW7 2AZ, United Kingdom. E-mail: b.glocker@imperial.ac.uk.*
- *J. Shotton, A. Criminisi, and S. Izadi are with Microsoft Research, Cambridge, CB1 2FB, United Kingdom. E-mail: {jamiesho, antcrim, shahrami}@microsoft.com.*

Manuscript received 28 Jan. 2014; revised 4 Aug. 2014; accepted 10 Sept. 2014. Date of publication 25 Sept. 2014; date of current version 1 Apr. 2015. Recommended for acceptance by M. Gandy, S. Julier, and K. Kiyokawa. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TVCG.2014.2360403

continuation of mapping even when camera tracking is frequently lost. This avoids the frustrating task of restarting an entire scan from scratch due to tracking failure. Also, this makes AR applications more robust.

We have recently proposed a relocalization module [26] which can be easily integrated into existing reconstruction pipelines. Our approach is inspired by different components of previous work, resulting in an efficient and scalable algorithm that provides a solution to the main causes of tracking failure in systems such as KinectFusion. This article is an extended version this earlier work. Here, we provide an extensive performance evaluation and a more detailed description of our method including visual examples. Comparison to more baselines including sparse feature methods and thorough exploration of the effects of various parameters on the relocalization performance will hopefully provide valuable insights to our approach.

1.1.1 Related Work

Relocalization has been widely studied in the context of real-time SLAM. While approaches exist that require an offline training phase (*e.g.* [27], [28]), below we focus on methods which are capable of *online* real-time performance. One can roughly categorize existing approaches into two categories, though hybrid [29] and more exotic variants exist [30], [31].

The first category are landmark-based approaches (LbAs) [32], [33], [34]. During successful tracking, fiducial landmarks or features, also called *keypoints*, are extracted from the camera images, encoded by a descriptor, and stored in a database together with their 3D locations. When tracking is lost, landmark candidates are detected in the incoming frame and based on descriptor similarity putative matches are established between those candidates and stored keypoints. In a recent work [33], the authors employ a 3D test based on depth information to rapdily rule-out false matches. The combination of the perspective 3-point algorithm and RANSAC [35] is then commonly employed to determine the pose of the camera.

We denote the second category as image-based approaches (IbAs) [36], [37], [38]. Indeed, all methods discussed here rely on image information, however, the main difference to the first category is that IbAs make use of global image matching and do not require explicit landmark detection. During successful tracking, compact representations of whole images are generated and stored together with the corresponding camera poses. Those frame/pose pairs are commonly referred to as *keyframes*. When tracking is lost, the compact representation of the incoming frame is compared to the ones of all keyframes. The poses of the most similar keyframes are retrieved and then used to directly reinitialize the tracking algorithm.

The notion of keyframes often also appears in the context of LbAs. Here, keyframes are particular camera frames from which keypoints have been extracted and stored. In our work, however, we commonly associate keyframes with IbAs and whole image matching.

Both categories come with advantages and drawbacks. The main advantage of LbAs is their ability to recover the pose from novel views. As long as a sufficient number of keypoints can be recognized, the camera pose can be determined. Depending on the visual characteristics of the scene and possible artifacts such as motion blur, it might not always be possible to obtain sufficiently many matches. Also, the construction of the keypoint database in real-time settings can be challenging. Often a costly online training phase is required which demands additional resources such as a background thread or extra GPU computations [34]. Another limitation of LbAs lies in their inherent sparse representation of the scene. Some approaches are limited to store only a few thousand unique points, as discussed in [32], [37]. The optimal choices for a suitable pipeline of robust detection [39], description [40], [41], [42] and matching [43] is certainly a challenge on its own.

Instead of map locations, in IbAs a set of keyframes represents the reconstructed scene. This allows direct retrieval of pose proposals for tracking initialization. The main challenges are related to the online determination of keyframes and the definition of efficient frame similarity measures. For the first part, often heuristics are employed such as thresholds on distances in pose space. For instance, a tracked frame is added to the set of keyframes only if the camera translation and orientation are sufficiently different from the previous keyframe [36], [38]. This heuristic might yield non optimal scene coverage. The second issue regarding efficient similarity evaluation is commonly tackled by using compact representations such as heavily downsampled images and normalized intensity differences [36], [37], [38]. With increasing number of keyframes, the time needed for the search of the most similar ones can be a limiting factor of IbAs in real-time settings. The fact that tracking can only be recovered from views which have been approximately visited before has been recently approached by utilizing synthesized views [36]. However, rendering such views can be costly and defining an optimal sampling strategy in pose space is non trivial.

1.1.2 Contributions

In light of prior work, we developed a simple yet powerful relocalization approach [26] which falls into the IbA category. Our method makes use of randomized ferns which have been previously used in the context of keypoint-based relocalization [34]. The way in which we make use of ferns is quite different and will be described in Section 2. With our relocalization approach we make the following contributions:

1) Efficient frame encoding scheme allows compact representation of keyframes, and fast retrieval of pose proposals for tracking recovery;
2) Automatic discovery and sampling of keyframes by exploring the space of appearance during tracking and avoiding spatial sampling heuristics;
3) Scalability to large scenes through a small memory footprint, large model capacity, and minimal computational demands;

Besides those properties which are essential for the integration of our method into 3D reconstruction pipelines, we demonstrate the robustness of our method with respect to different settings of internal method parameters. After discussing the technical details, we investigate

the relocalization performance in an experimental evaluation in Section 3. We then discuss in Section 4 a practical application of marker-free AR realized with the KinectFusion system equipped with our relocalization module. In Section 5, we discuss current limitations and future work.

2 Keyframe-based Relocalization via Randomized Ferns for Frame Encoding

Our relocalization approach is based on the idea of generating compact codes for camera frames. Those codes can then be used to efficiently determine frame dissimilarities either to judge whether a frame is novel enough to be considered as keyframe or to retrieve the poses of most similar keyframes when tracking is lost.

We start by explaining how randomized ferns can be used for generating compact codes.

2.1 Frame Encoding

Given an RGB-D image frame $I : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}^4$, we define the function that provides pixel values as $I_c(\mathbf{x}) \in \mathbb{R}$ where $\mathbf{x} \in \Omega$ is the pixel location in image channel $c \in \{R, G, B, D\}$. For convenience we introduce the notation $I(\theta) = I_c(\mathbf{x})$ with $\theta = (c, \mathbf{x})$.

In order to generate compact code representations for RGB-D image frames, we first define a fern $F = \{f_i\}_{i=1}^{n}$ as a set of n consecutive nodes f_i where each node represents a binary test parametrized by a pair (θ_i, τ_i). Each test is evaluated on the image data as

$$f(I, \theta, \tau) = \begin{cases} 1 & \text{if } I(\theta) \geq \tau \\ 0 & \text{if } I(\theta) < \tau. \end{cases} \quad (1)$$

Here, τ is a threshold on the image pixel value $I(\theta)$. Evaluating all f_i of a fern F in consecutive order provides a binary code block $b_F = f_1 \dots f_n \in \mathbb{B}^n$. Given a conservatory $C = \{F_k\}_{k=1}^{m}$ of m diverse ferns, the concatenation of individual code blocks yields a global single code $b_C = b_{F_1} \dots b_{F_m} \in \mathbb{B}^{mn}$. This mechanism allows us to generate (non unique) codes for any RGB-D image frame. A particular binary image code b_C^I depends on the total number of ferns m, the number of nodes n in each fern, the binary test parametrization (θ_i, τ_i) of each node, and of course on the visual content of the image frame I.

The idea of using ferns for generating codes for image patches has been already introduced in [44]. This has been applied to the task of camera relocalization in [34]. In both works, the idea is to learn compact codes for efficient keypoint recognition instead of encoding whole image frames. At test time, in those works the conservatory of ferns is utilized as a classifier to find putative matches between incoming frames and a learned keypoint database.

Although inspired by the promising performance of those fern-based approaches, we use ferns in a way which is quite different from previous works. We employ whole frame encoding for keyframe-based relocalization with a test procedure that allows us to simultaneously compute frame dissimilarities between a new frame and all previously stored keyframes. In the following we define the frame dissimilarity in terms of the hamming distance.

2.2 Frame Dissimilarity via Hamming Distance

Given compact representations b_C^I and b_C^J for two camera frames I and J, the frame dissimilarity is defined as the BlockHD as

$$\text{BlockHD}\big(b_C^I, b_C^J\big) = \frac{1}{m} \sum_{k=1}^{m} b_{F_k}^I \equiv b_{F_k}^J, \quad (2)$$

where the equivalent operator $\equiv$ returns 0 if two code blocks are identical and 1 if there is at least one bit difference. The BlockHD is simply counting the number of differing code blocks. The normalization with respect to the number of ferns m maps the distance to the $[0, 1]$ interval. This is a nice property which eases parameter tuning of an algorithm.

2.3 Precision/Recall of BlockHD

In contrast to the well-known bit-wise hamming distance which counts the differing bits of two codes, the block-wise version has an interesting property related to the length of the blocks. Varying the length which is determined by the parameter n directly impacts the precision/recall characteristics of BlockHD. The probability that code blocks b_F^I and b_F^J are equivalent decreases with increasing bit length due the increasing number of binary tests. Remember that the frames I and J need to produce same feature test responses for identical codes. Longer code blocks increase precision but negatively affect recall. Two frames with a low BlockHD and long code blocks are very likely to be very similar. However, one might also miss some similar frames due to image noise or just by chance because of the hard thresholding in the feature tests (cf. Eq. (1)). For shorter codes the probability is higher that two frames with different visual content yield a low BlockHD. The recall, however, intuitively increases in this case, though the precision might be low. In case of 1-bit blocks which corresponds to ferns with only one node, the BlockHD is equivalent to the bit-wise HD. In the following we describe the mechanisms that allows us to efficiently compute the frame dissimilarity.

2.4 Harvesting Keyframes

The key difference to previous fern-based approaches is in the way we utilize the output of the ferns. Remember a fern with n nodes can generate 2^n unique codes. We associate each fern F with a code table T_F with 2^n rows. Each row can store a set of frame identifiers (ids)[1] and all sets are initially empty. In addition to the fern-specific code tables, we also define one global table P taking id/pose pairs as input elements. This set is also initially empty and will be used to globally store the camera poses of keyframes.

Let us now assume a steady stream of tracked camera frames with pairs $(I, H)_{id}$ where $H \in SE(3)$ is the camera pose composed of rotation and translation. Here, the id is assumed to be unique for each frame/pose pair. For each incoming frame we can generate the code b_C^I and add its id to the m sets from the corresponding rows which are linked with the individual code blocks. Additionally, we would add the pose H with key id to the global table P.

1. This is different to classification ferns [34], [44] where empirical distributions over keypoint classes are stored in the code tables.

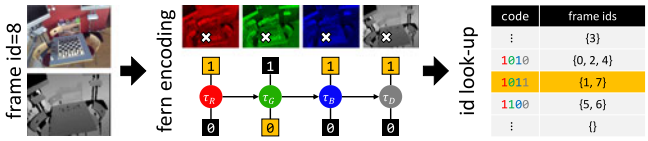

Fig. 1. Frame encoding: Fern-based frame encoding takes an input RGB-D image and generates small code blocks for each fern based on simple binary feature tests evaluated at randomized, but fixed, image locations. A code block is associated with a row of a code table which stores the ids of keyframes with equivalent codes. In harvest mode, the id of an incoming frame is added to the row if the minimum dissimilarity κ_I is above a threshold. For tracking recovery, most similar keyframes are retrieved from the tables and the corresponding poses are used for reinitialization of the tracking algorithms.

Assuming a number of tracked frames have been already encoded and stored using this strategy. Now, every time we are about to add a new id to a set in a row of a code table T_F, we can also immediately read out the previously stored identifiers (*cf.* Fig. 1). We know that those must correspond to frames which have an equivalent code block b_F. In fact, if we simply count those co-occurrences of previously stored frames along the m rows in which we would add the new id, we can *simultaneously* compute the dissimilarities between the new frame and all previously stored frames. Assuming the count of co-occurrences for two frames I and J using the above procedure is denoted as q_{IJ}, then we can equivalently to Eq. (2) compute their dissimilarity by

$$\mathrm{BlockHD}\left(b_C^I, b_C^J\right) = \frac{m - q_{IJ}}{m}. \tag{3}$$

In addition, for every incoming new frame I we can determine the minimum BlockHD with respect to all previously stored frames as

$$\kappa_I = \min_{\forall J} \mathrm{BlockHD}\left(b_C^I, b_C^J\right) = \min_{\forall J}\left(\frac{m - q_{IJ}}{m}\right). \tag{4}$$

The value κ_I provides useful information about the novelty of new frame compared to stored keyframes. A low value reflects that a very similar frame is already present, while a high value indicates a novel view from a pose which should probably be stored as a keyframe. Based on this observation, we propose a strategy for online harvesting of tracked frames and automatic determination of keyframes. Based on the value κ_I and a predefined threshold t, we decide whether an incoming frame is added or discarded. It should become clear that such a threshold influences how densely the observed scene is covered by keyframes (see Fig. 2). Intuitively, a compromise is desired which avoids redundant information to be added to the scene representation while the coverage should also be sufficiently dense. Note, that our harvesting strategy is completely driven by the visual content of the camera frames and spatial sampling heuristics based on pose offsets [36] are avoided.

2.5 Tracking Recovery via Pose Retrieval

When tracking fails for an incoming frame, we make use of the previously harvested keyframes to retrieve pose proposals for tracking recovery. For the incoming frame we perform the same encoding procedure as in harvesting mode which provides dissimilarities to all previously stored keyframes. Now, instead of determining the value

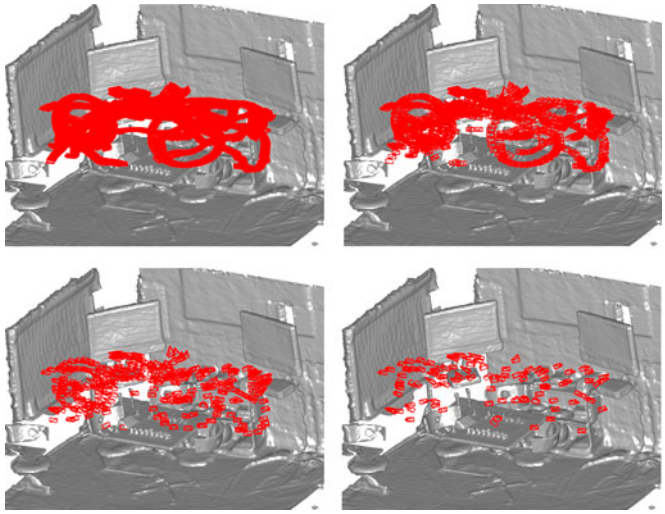

Fig. 2. Harvesting keyframes: The top-left image shows a reconstructed scene with 4000 tracked frames. Each camera pose is shown as a red frustum. When simulating harvesting from this stream of frames, 836 are accepted as keyframes for $\kappa_I > 0.1$ (top-right). Increasing the threshold to $\kappa_I > 0.2$ yields 314 keyframes (bottom-left) and for $\kappa_I > 0.3$ only 105 are being accepted (bottom-right). We evaluated the effect of the keyframe coverage on the relocalization performance and the results are summarized in Fig. 8.

of the minimum distance, we directly determine the ids of the k nearest keyframes, *i.e.* the most similar ones. We can then read-out the corresponding pose proposals from the global table P. Depending on the underlying tracking and mapping approach, we can for example use those k poses to reinitialize the tracking algorithm. It is also possible to employ a weighted averaging using the frame similarities as weightings to compute a novel pose proposal. This is similar to the idea proposed in [36]. If reinitialization is unsuccessful for all proposed poses, we discard the current frame and repeat the same procedure for the next incoming camera frame until tracking is recovered. In the meantime, the mapping process is paused and automatically resumed after recovery. Remember we assume a user controlled, hand-held camera, and it is likely that at some point the user will move the camera to a pose which is similar enough to a previously tracked one. One could also imagine to employ visual guidance which could help the user to move the camera into areas of good keyframe coverage. In Fig. 3 we illustrate some examples of the pose retrieval process. We show incoming test frames (only RGB is shown) on the left, and the five nearest neighbors from the stored keyframes. We also plot the corresponding camera poses in the reconstructed scene for a geometric interpretation of those results. Visually, the retrieved keyframes are very similar to the query image and similarly are their poses in most cases. If among the five nearest neighbor poses there is at least one that allows the tracking to resume, we can successfully recover the pose of the camera and continue the mapping process.

2.6 Fern Construction

We have not discussed the details for the fern parameters introduced in Section 2.1. In particular, the binary test parameters (θ_i, τ_i) seem crucial for obtaining useful compact frame representations. However, and maybe not too

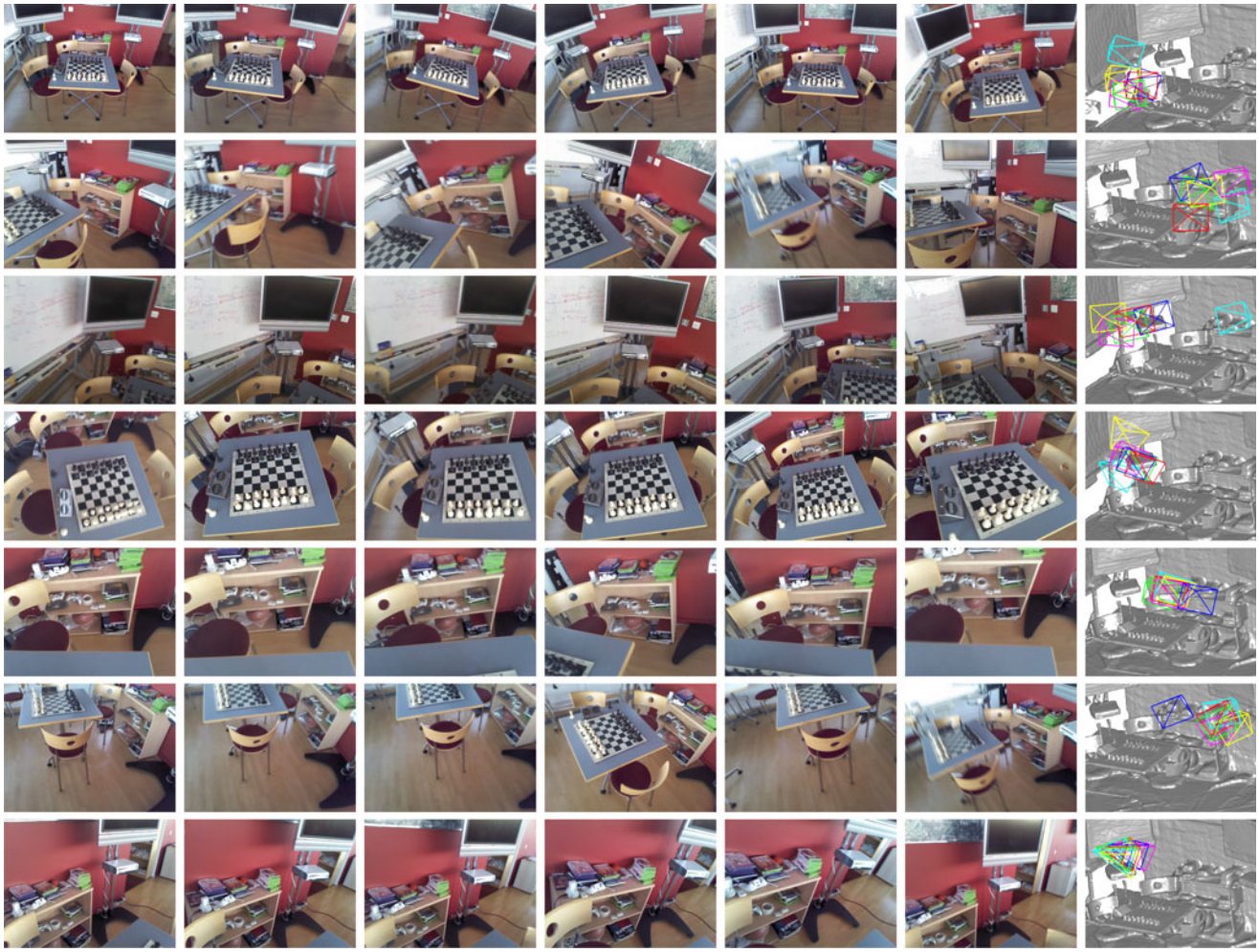

Fig. 3. Pose retrieval: The left column shows the RGB image of the query frame for which tracking has failed. The second to sixth column show the RGB images of the retrieved keyframes in ascending order with respect to frame dissimilarity (BlockHDs). The plots in the most right column illustrate the corresponding camera poses. The yellow frustum is the 'ground truth' pose of the incoming frame. The poses of the retrieved keyframes are shown in red (nearest), green (2nd), blue (3rd), magenta (4th), and cyan (5th). In most cases, the retrieved poses are sufficiently close to 'ground truth'. An interesting case of visual ambiguity resulting in an outlier keyframe is shown in the third row. The 5th nearest keyframe is visually very similar, however, its camera pose (shown in cyan) is almost perpendicular to 'ground truth'. This is a limitation of keyframe matching approaches, but not a problem as long as at least one of the retrieved poses is sufficiently close to the real one.

surprising, we found that randomness injected into the construction of the ferns yields overall best performance. Similar findings are reported in related approaches [34], [44], [45]. Our default construction strategy for the entire conservatory of ferns is explained in the following. The impact of parameters such as the number of ferns m, the keyframe acceptance threshold t, or the number of neighbors k are investigated in our experiments.

>Per default, each fern consists of $n = 4$ nodes with one node per RGB-D channel, yielding $F = \{f_R, f_G, f_B, f_D\}$. This also defines the parameters c_i within the set $\{\theta_R, \theta_G, \theta_B, \theta_D\}$. We randomly sample image locations $\mathbf{x}_i$ at which the binary tests are applied from a uniform distribution over the image domain Ω. Note, those locations are sampled only once at the time of the fern construction. Here, we sample one location per fern, such that $\mathbf{x}_R = \mathbf{x}_G = \mathbf{x}_B = \mathbf{x}_D$. This has the effect that a code block b_F corresponds to feature responses from all image channels obtained at the same fixed location. The feature test thresholds τ_i are uniformly sampled such that $\tau_R, \tau_G, \tau_B \in [0, 255]$ for RGB and $\tau_D \in [800, 4000]$mm for depth.

3 Experimental Evaluation

It is not straightforward to assess the overall performance of a real-time relocalization system. The best way is probably to actually test the system when it is running in a live, real-world environment. However, quantitative evaluation of a live system becomes difficult, in particular, if one is interested in how certain system parameters affect performance.

Keeping the limitations of simulated tests in mind, in the following we explore the performance of our system by relative comparisons. We employ the publicly available '7-scenes' dataset[2] for simulation purposes. All scenes in this dataset were recorded from a handheld Kinect camera at 640×480 resolution. We used KinectFusion [6] (with care to avoid loss of track) to obtain the 'ground truth' camera poses. For each scene, several sequences were captured by different users, and split into two distinct evaluation sets. One set is then used as a steady stream of tracked frames

2. http://research.microsoft.com/7-scenes/

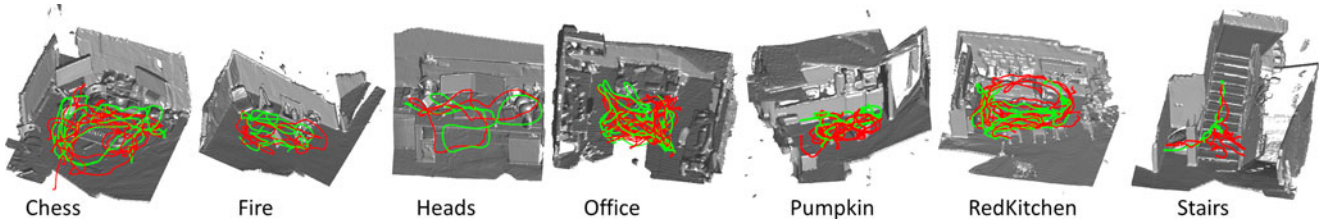

Fig. 4. RGB-D dataset: For each of the seven scenes, we have recorded several sequences of tracked RGB-D camera frames. The frame trajectories used for simulating the online harvesting are shown in red. The frames used for evaluating tracking recovery are shown green.

TABLE 1
Main Results: Summary of the Relocalization Performance Evaluated on the RGB-D Dataset '7-Scenes'

Scene	Spatial Extent	#Frames Harvest	#Frames Recovery	Tiny Image Baseline (k=5) NN	Tiny Image Baseline (k=5) WAP	Tiny Image Baseline (k=5) kNN	Our Results (m=500, t=0.2, k=5) NN	Our Results (m=500, t=0.2, k=5) WAP	Our Results (m=500, t=0.2, k=5) kNN	Our Results (m=500, t=0.2, k=5) Keyframes	All-in-One kNN
Chess	3m^3	4k	2k	70.8%	72.4%	80.6%	68.4%	74.8%	**85.3%**	317	82.0%
Fire	4m^3	2k	2k	56.6%	56.9%	60.9%	57.4%	53.5%	**72.0%**	141	66.0%
Heads	2m^3	1k	1k	49.0%	50.2%	61.0%	47.9%	40.4%	**79.8%**	91	76.1%
Office	5.5m^3	6k	4k	60.8%	61.3%	65.8%	61.7%	50.2%	**74.7%**	674	72.7%
Pumpkin	6m^3	4k	2k	54.6%	56.1%	59.7%	47.8%	53.1%	**62.8%**	302	62.5%
RedKitchen	6m^3	7k	5k	46.1%	46.8%	49.3%	42.0%	41.4%	**54.1%**	588	52.7%
Stairs	5m^3	2k	1k	25.2%	27.3%	29.9%	21.4%	13.9%	**34.1%**	68	32.7%
Average				51.9%	53.0%	58.2%	49.5%	46.8%	**66.1%**		63.5%

Percentages correspond to the number of successfully recovered frames (within 2 cm translational error and 2 degrees angular error). The best performance is obtained with our relocalization approach and the kNN pose proposal strategy. Similarly good performance for scalability test on the 'All-in-One' representation indicates promising scalability properties for our compact encoding scheme.

for simulating harvesting of keyframes, the other set is used for performance evaluation. The frames exhibit ambiguities (*e.g.* repeated steps in 'Stairs'), specularities (*e.g.* reflections in 'RedKitchen'), motion blur, lighting conditions, flat surfaces, and sensor noise. It should be noted that the 'ground truth' poses do contain some errors which occur from a slight tracking drift and model distortions. An overview of the dataset is shown in Fig. 4 and some more details about each scene are given in Table 1. The varying difficulties of the scenes are reflected in the errors, consistently across different approaches.

3.1 Overall System

We integrated our relocalization module into the KinectFusion pipeline (see Fig. 5) which relies on model-based ICP camera tracking [6]. In order to detect ICP tracking failure (and success), we employ a plausibility check on the magnitude of camera motion and on the ICP residual error. If the relative motion or residual is too large, ICP reports tracking loss and relocalization is initiated. A short demonstration of this system is shown in a video[3] which highlights the importance and the performance of our relocalization module. In particular, the system is able to immediately recover tracking even when the camera frequently leaves and re-enters the limited reconstruction volume.

3.2 Performance Evaluation

Our main metric for comparing different settings and approaches is the percentage of frames for which the pose was successfully recovered. Here, we define recovery to be successful if the final estimated pose after ICP camera tracking is within 2 cm translational error and 2 degrees angular error compared to ground truth. In order to explore the effect of the randomness in our method we perform three complete runs for all experiments. In each run, the conservatory of ferns is constructed from scratch. The reported performance values of our method are averages over three runs. The stability of the performance for different runs is discussed in Section 3.6.4.

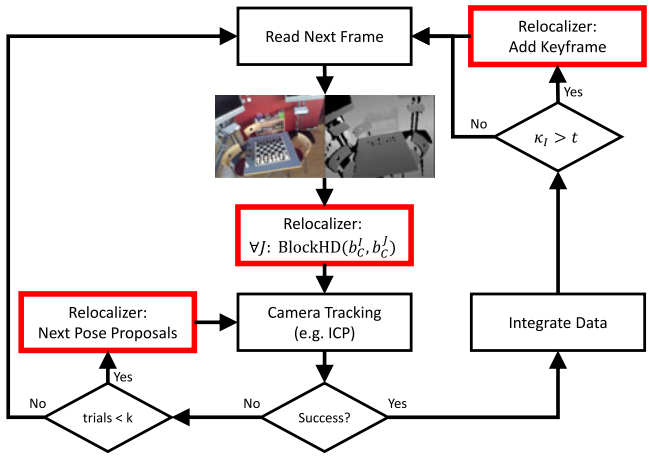

Fig. 5. Pipeline integration: The diagram illustrates the integration of our relocalization module into a standard 3D reconstruction pipeline such as KinectFusion. Every incoming frame is pushed through our encoding and frame dissimilarity mechanism. Depending on whether camera tracking is successful, we can forward tracked frames to the map integration and keyframe extraction procedures, or we initiate pose proposal retrieval for tracking recovery.

3. http://www.doc.ic.ac.uk/ ~bglocker/videos/ferns.mp4

3.3 Frame Processing

Per default, our relocalization method operates on downsampled, smoothed images with a resolution of 40×30 (factor 16) and Gaussian blur of $\sigma = 2.5$ pixels (after downsampling). The same pre-processing is for instance used in [37], [38] and we use this setting when comparing to baselines. In order to explore the effect of this pre-processing we performed a set of experiments with varying downsampling factors $(1, 2, 4, 8, 16)$. In each case, we keep the Gaussian blur with $\sigma = 2.5$ pixels for the respective image resolutions of 640×480, 320×240, 160×120, 80×60, and 40×30. The results for those tests are reported in Section 3.6.5.

3.4 Pose Proposal Strategies

There are several strategies of using pose retrieval for relocalization. One way is to simply initialize the tracking algorithm with the nearest neighbor (NN) pose, *i.e.* the one from the keyframe with smallest BlockHD. It is also possible to retrieve a set of kNN proposals and initialize the tracking with each of those. Besides proposing directly the poses of keyframes, we can also interpolate a proposal via weighted averaging over the kNN poses [36].

In the following, we will compare relocalization performance for different strategies, namely NN, kNN, and weighted average pose (WAP). In contrast to NN and WAP where we provide a single proposal, kNN corresponds to a multi-proposal approach where the poses of the k closest keyframes plus their WAP are used for initializing ICP. If ICP reports success for more than one pose, the one with lowest residual error is selected for continuing normal tracking and mapping. We also investigate the influence of k in a separate experiment. In particular, we explore values 5, 10, 20, and 40 for the k nearest neighbor pose proposal and pose averaging. The results for varying values of k and the impact of including WAP are reported in Section 3.6.6.

3.5 Baselines

We compare our method to another image-based approach denoted as 'tiny image baseline'. This IbA baseline represents whole image matching approaches [36], [37], [38]. Keyframes are stored with their camera poses, after downsampling to 40×30 pixels and applying a Gaussian blur of $\sigma = 2.5$ pixels [37]. For pose retrieval with best possible accuracy (at the expense of speed), we use brute-force matching against *all* available stored frames using the normalized distance over RGB-D as defined in [36]. It should be noted that this exhaustive search is impractical for real-time systems but it should give an upper bound on the performance. In practice, keyframe sampling heuristics are usually employed to keep the number of keyframes at a reasonable level [36]. Such heuristics, however, do not guarantee sufficient coverage of a scene. In order to eliminate this factor of insufficient keyframe density, we opt for the brute-force search strategy. We use the same pose proposal strategies as for our method, *i.e.* NN, kNN, and WAP with $k = 5$. The comparison with respect to this baseline is reported in Section 3.6.1.

In addition, we also compare the performance of our real-time keyframe method with two landmark-based approaches. Both of them require an offline training phase and thus, are not suitable for integration into real-time SLAM systems. Those methods can be considered for relocalization tasks when a previously scanned scene is revisited, and there is sufficient time for training in between. The comparison, however, allows us to get an estimate of how close we can get to the performance of those offline methods. The first LbA method employs the fast ORB feature descriptor [46] and a database of 3D points which is built during the training phase. Relocalization then becomes a problem of 2D-3D feature matching which is solved using RANSAC and the perspective 3-point method. The second method is a random forest approach for directly estimating correspondences between 2D image points and 3D locations in a canonical representation of the reconstructed scene. The details for both methods are given in [28], and a comparison with our method is discussed in Section 3.6.2.

3.6 Results

In the following we investigate different properties of our relocalization system. Besides quantifying the actual performance for pose recovery in comparison to baselines, we also explore the impact of different parameters and the scalability to larger scenes. We also report detailed timings and computational impacts of the individual components of our method.

3.6.1 Baseline Comparison

Our main quantitative results for the comparison to the keyframe baseline are summarized in Table 1. Both, our approach and the baseline perform best in terms of successfully recovered frames when using the kNN proposal strategy. Overall, our approach is able to recover from significantly more test frames compared to this baseline. In all the experiments, we set $k = 5$ such that in total six poses (including the WAP) are used for initializing ICP. Using WAP alone as a single proposal does not perform too well for both our method and the baseline (see Table 1). However, adding it as an additional pose proposal in the kNN approach seems beneficial. This is investigated in more detail in Section 3.6.6. In Fig. 6 we show some more detailed statistics of the angular and translational error distributions for our method and the baseline when using the kNN strategy.

The given percentage of successful frames is always with respect to the total number of tested frames as listed in the column 'Recovery'. The column 'Harvest' indicates the number of frames used for simulating keyframe harvesting, or in case of the baseline directly the number of keyframes. The number of accepted keyframes for our approach is given in the column 'Keyframes'. Note that these quantitative results correspond to a simulated setting where it is assumed that all 'Harvest' frames are observed before the 'Recovery' frames are tested.

3.6.2 Comparison With Landmark-Based Approaches

In Fig. 7 we compare our method and also the 'tiny image baseline' with the performance of two LbAs which have been discussed in [28][4]. Those methods require an offline

4. The error thresholds in [28] were set to 5 cm and 5 degrees, so numbers reported here are updated for 2 cm and 2 degrees.

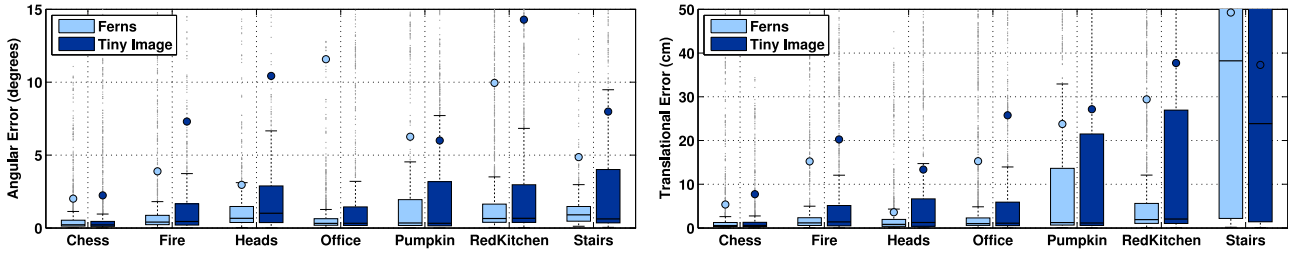

Fig. 6. Error statistics: Angular and translational error distributions are shown for our fern-based method and the 'tiny image baseline' when using the kNN pose proposal strategy. The large translational error for the 'Stairs' scene is explained by the repetitive appearance of stair cases which yields ambiguities that are difficult to resolve.

training phase in which the landmark database respectively the regression forests are constructed. This offline training yields favorably performance on all of the sequences compared to our real-time approach, but limits the applicability of those approaches.

3.6.3 Number of Ferns And Acceptance Threshold

The main comparative results are obtained with our default setting of $m=500$ ferns and a keyframe acceptance threshold of $t=0.2$. In Fig. 8 we compare the performance of our method using the kNN with $k=5$ strategy with varying parameters m and t. We observe that initially adding more ferns improves relocalization while further improvement beyond 500 ferns is marginal. Regarding keyframe acceptance, we find that a lower threshold ($t=0.1$) yielding denser scene coverage does not necessarily improve relocalization. Setting the threshold too high ($t=0.3$) yields a too sparse sampling of keyframes and relocalization performance decreases. A visual example for the coverage of the 'Chess' scene for varying values of t is shown in Fig. 2. The number of accepted keyframes for the three different thresholds are provided in Fig. 8. For our largest scene 'RedKitchen' those numbers are 1368, 588, and 272, and for the smallest scene 'Heads' 218, 91, and 45 keyframes are accepted (on average) under the different thresholds.

3.6.4 Effect of Randomness

In general, the relocalization performance is very stable across different runs in which the ferns have been randomly constructed from scratch. Also, the number of accepted keyframes (overlaid in white) does not vary significantly. The only larger variance is found for the challenging 'Stairs' sequence. In Fig. 9 the variation across three different runs with default fern setting is summarized.

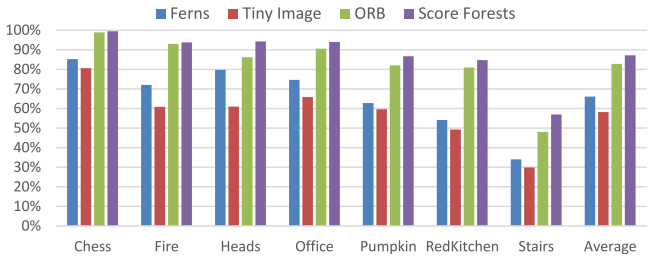

Fig. 7. Comparison with offline LbAs: Compared to our real-time fern-based method, the two offline landmark-based approaches perform favorably. Though, their applicability is limited to scenarios of revisiting a previously reconstructed scene after training.

3.6.5 Effect of Frame Size

Up to a factor of 16, the downsampling of frames has little impact on the overall performance as can be seen in Fig. 10. For higher factors a significant drop in performance can be observed. In practice, we use a factor of 16 yielding frames of size 40×30 which is computationally beneficial for subsequent processing, in particular, when applying the Gaussian blur.

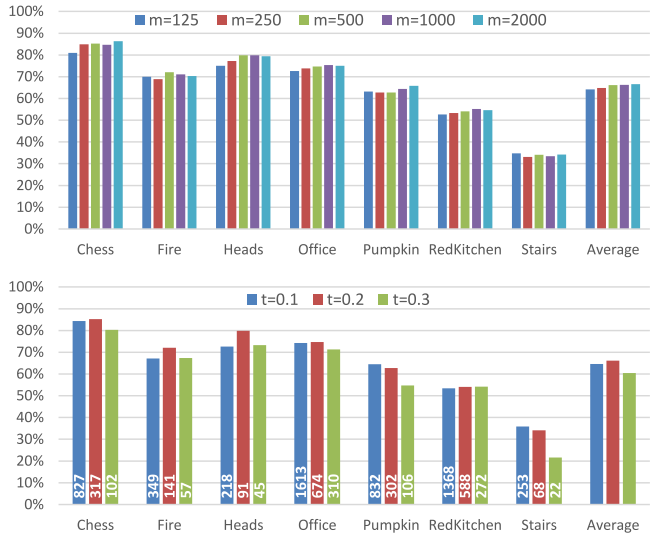

Fig. 8. Effect of number of ferns and harvesting threshold: Relocalization performance when varying the number of ferns m (top graph) and changing the keyframe acceptance threshold t (bottom graph). Initially increasing the number of ferns improves performance, which levels off after $m=500$. The threshold t influences the number (in white), and thus, the density of keyframes. Optimal coverage seems to be obtained with $t=0.2$ (see also Fig. 2 for a visual example).

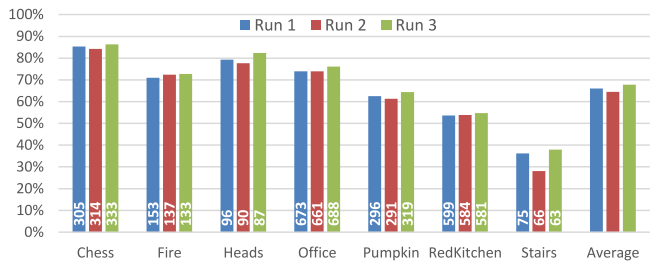

Fig. 9. Effect of randomness: Performance is relatively robust to the effect of randomness in different fern constructions. Only for the challenging 'Stairs' sequence we found some significant variation in performance over different runs.

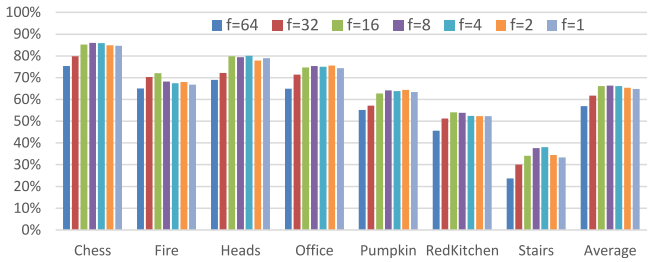

Fig. 10. Effect of frame size: Downsampling with factors up to 16 has little impact on the overall relocalization performance. Performance drops significantly for 32 and 64. We use a factor of 16 which yields a frame size of 40×30 and enables very efficient processing in the subsequent pipeline.

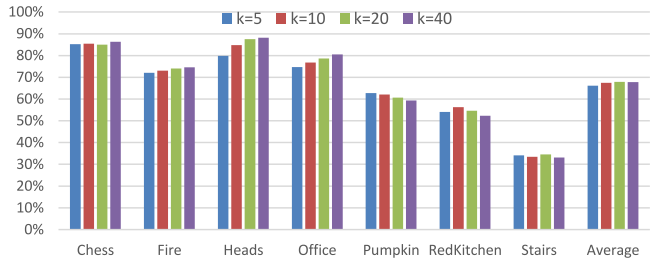

Fig. 12. Effect of number of neighbors: Relocalization performance can be slightly increased when considering more keyframe proposals. This, however, comes with higher computational costs. Overall, our default setting of $k=5$ seems a good compromise.

3.6.6 Nearest Neighbors and Proposal Selection

For $k=5$ we have tested the kNN strategy with and without including the interpolated pose WAP. Without WAP we got consistently worse results, so including it seems beneficial. This is summarized in Fig. 11 where we also report the selection frequencies of the pose proposals for the kNN strategy with WAP. Proposal selection is based on the smallest ICP residual. The WAP proposal is almost as often selected as the nearest neighbor pose.

We further tested different values for k and the results are summarized in Fig. 12. Overall performance can be slightly increased, in particular on the 'Heads' scene, when considering more neighbors. However, the running time for the relocalization process increases linear with the number of proposals. In summary, our default setting of $k=5$ including WAP seems a good compromise between performance and speed (see also Section 3.6.9).

3.6.7 Importance of Color and Depth

We also evaluated the importance of using both color and depth information for the relocalization and compared the performance to two setups where we use either RGB or depth only. For those experiments we use a fern encoding with the same capacity as for the RGB-D version, *i.e.* same number of ferns and 4-bit block codes. Instead of associating each bit to one channel, in the RGB only setup we randomly select a channel per bit when constructing the ferns and those selections are spatially varying. For the depth only variant, all 4 bits are associated with the depth channel, and thus the corresponding feature tests are evaluated on depth only. The results are summarized in Fig. 13. The best performance is obtained when using the full RGB-D information. The RGB only version still performs reasonable and might suggest that our relocalization module could be employed in RGB SLAM systems. The average of successfully recovered frames drops from 66.1 percent to 62.7 percent. Depth information alone is not sufficient to obtain a similarly good performance. The performance measure for the latter one is only 52.5 percent.

3.6.8 Scalability to Larger Scenes

An important aspect of keyframe-based approaches is scalability to large scenes. In particular, this is a challenge for whole image matching approaches such as our baseline where the search for closest keyframes can become impractical when thousands of keyframes are stored. In order to evaluate the scalability of our approach, we performed the following experiment. We constructed a single conservatory of ferns with our default parameters $m=500$, $t=0.2$ and $k=5$. We then used all 26,000 frames from all seven scenes for keyframe harvesting. In total, 2091 frames are accepted as keyframes, which is slightly less than the sum over all keyframes from individual scenes, *i.e.* 2181. In the most right column in Table 1 we report the relocalization performance when using this 'All-in-One' keyframe representation. The overall performance is only slightly worse compared to the scene-specific constructions. We believe this indicates promising scalability properties for our encoding scheme and a model capacity which is sufficient for representing large scenes of more than 30m^3 and thousands of frames.

3.6.9 Timings

Real-time performance of relocalization is essential when being integrated into a tracking and mapping pipeline. In the following we report average timings for individual

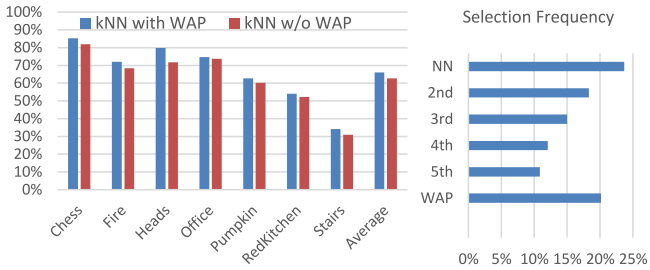

Fig. 11. Proposal selection: Including the weighted average pose in the kNN proposal strategy seems beneficial in terms of relocalization performance. The selection frequency reveals that the WAP is almost as often selected as the NN pose.

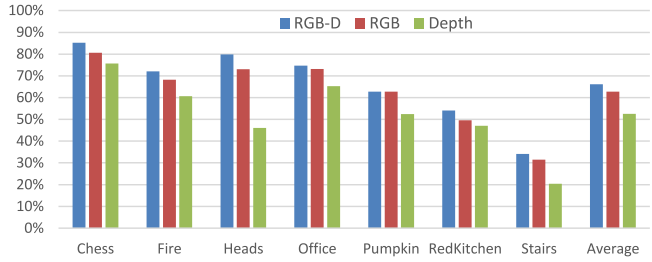

Fig. 13. Importance of color and depth: Best relocalization performance is obtained when using the full RGB-D information. A system using RGB only still performs reasonable, while depth information alone seems insufficient to obtain good performance.

components of our method. All timings have been acquired while running KinectFusion on the 'RedKitchen' scene where a relocalization module based on the default setup and 574 stored frames is assumed to be in place. This allows us to measure the expected impact of keyframe harvesting and recovery under realistic conditions. While the KinectFusion pipeline itself is mostly running on GPU (Nvidia Geforce GTX 580), our relocalization runs currently entirely on a single core CPU (Intel Xeon 2.27 GHz).

The key intervention to the existing pipeline is that every incoming frame is pushed through the frame encoding and dissimilarity computation mechanism. So, even in normal tracking mode our method has an impact on the overall tracking and mapping performance. We found this impact to be very small, with only 3ms for frame encoding including computation of κ_I. A KinectFusion system running at 30 FPS will continue to run at 27 FPS when keyframe harvesting is running in the same computation thread. Of course, the relocalization module could also run in a parallel, dedicated thread. When tracking is lost, we measure 160ms for camera recovery with kNN with $k=5$ including 6 runs of ICP (for the k+1 proposals). The total impact of relocalization during tracking recovery is about 165ms, which keeps the system reasonably responsive. For comparison, the timings increase slightly for the 'All-in-One' representation with 2091 keyframes where harvesting takes about 7 ms.

4 Marker-Free Augmented Reality

Real-time 3D reconstruction provided by RGB-D systems such as KinectFusion enables exciting augmented reality applications. Here, we present a prototype of an AR system with potential use in medical and industrial environments where digital 3D models or scans of physical objects are available. In medical settings, anatomical scans of patients are frequently acquired with computed tomography (CT) or magnetic resonance imaging (MRI) systems for the task of diagnosis, interventional planning and surgical navigation. For example, in keyhole surgery doctors use the anatomical scan to plan the port placement of instruments to have optimal access to the region of interest [47]. Industrial AR systems for support and training of complex maintenance tasks [19] benefit from overlay of 3D geometries and other information extracted from available CAD or mesh models [7], [48]. In contrast to systems based on optical tracking and RGB only cameras, an additional depth sensor can overcome challenges such as occlusion handling and fusion of real and virtual objects.

Let us assume we are given a surface mesh of a real world object consisting of a set of n 3D vertices $V = \{\mathbf{v}_i\}_{i=1}^{n}$. For example, the mesh could have been extracted from an available 3D scan and could represent the skin surface of a patient's head. In order to be able to correctly overlay object-specific visual information on top of the camera image, we need to find a transformation $T : \mathbb{R}^3 \mapsto \mathbb{R}^3$ which registers the mesh and the online 3D reconstruction. We assume the latter is represented as a truncated signed distance function (TSDF) [6] denoted as $D : \mathbb{R}^3 \mapsto \mathbb{R}$ where zero-crossings correspond to object surfaces. Finding the optimal transformation $\hat{T}$ can be formulated as an optimization problem as follows

$$\hat{T} = \arg\min_{T} \sum_{i}^{n} \min[|D(T(\mathbf{v}_i))|\,,\ \lambda] \quad . \tag{5}$$

Here, the value λ truncates the cost function which makes it robust to outliers and missing data in partially reconstructed objects. The minimum of the cost function corresponds to the transformation where a large number of mesh vertices is located at object surfaces, *i.e.* at zero-crossings in the TSDF volume. The advantage of this direct mesh-to-volume registration is that no explicit correspondences are required between the 3D model and the reconstruction. However, a sufficiently good initialization is important.

As mentioned earlier, we assume a user controlled camera which allows us to employ a simple manual initialization mechanism which works well in practice. Although, an automatic approach can be envisioned [49]. When a reasonable part of the object to be augmented has been reconstructed, we display the 3D mesh model with a fixed offset in front of the camera. The user's task is then to navigate the mesh model by moving the camera to the proximity of the reconstructed object. Enabling z-buffer based rendering gives additional guidance to the user when the mesh and the reconstruction are sufficiently close. The actual registration can then be performed automatically. We employ the downhill simplex algorithm as an iterative optimization method which requires less than 5 seconds for the registration. Once the transformation according to Eq. (5) is determined, any information contained in the object's 3D model or scan can be correctly overlaid on top of the tracked camera images. This procedure enables compelling, marker-free AR which for example allows to peek inside the human body and visualize anatomical details of clinically relevant structures as illustrated for a head phantom in Fig. 14.

5 Conclusion

In this extended version of our work on keyframe-based relocalization using randomized ferns we explored in particular the performance under varying set of parameters. We hope this exploration provides valuable insights about our method. The comparison with (offline) landmark-based methods shows that there is still space for improvement in terms of relocalization performance. An additional comparison with online LBAs would be interesting but is beyond the scope of this work. Setting up an entirely fair and convincing comparison with existing implementations (*e.g.* FAB-MAP [30]) is a challenge on its own due to the complexity of finding optimal parameters for methods developed by someone else. The main limitation of keyframe-based approaches, that the camera view should not be substantially different from views covered by keyframes, could be potentially overcome using synthetic view sampling [36]. Here, our compact frame encoding is a key advantage. Instead of synthesizing whole frames, we only need to synthesize the compact codes for novel views which could dramatically reduce computational costs.

Another possibility to improve the relocalization performance could be to consider a multi-frame retrieval process similar to [50]. Instead of only looking at one incoming frame and try to relocalize its particular pose, one could consider a set of consecutive frames for which a

Fig. 14. Marker-free augmented reality: Illustrated is a potential medical AR application. The top left image shows a skin surface model of a head phantom fused with a real MRI scan of a patient with a brain tumor. Relevant clinical structures such as the tumor (red) and important blood vessels (blue) are highlighted. Such a segmentation is commonly done during interventional planning. The second image shows our polystyrene head phantom. Reconstructing the same outer surface with a real-time system such as KinectFusion (third image, top row) allows a mesh-to-volume registration between the surface model and the TSDF of the reconstruction (top right image). The bottom row shows in-situ AR visualizations of anatomical structures overlaid on the RGB camera view. The live depth measurements also enable realistic occlusion handling.

frame-to-frame tracking could give additional information about the relative pose trajectory. This information could then be matched to a set of retrieved pose proposals which would ideally improve the robustness for selecting the correct poses.

An interesting direction for future work would be to investigate how loop-closure detection could be realized within our method. Similar to [32], one could build a scene graph where stored keyframes correspond to nodes. As pointed out in [32], loop-closure and tracking recovery are similar events of edge creation between active and existing graph nodes.

In a similar context, one could explore the use of continuous pose retrieval for detecting tracking drift. When a previously scanned area of a scene is revisited one could compare the current pose estimation of the tracking algorithm with pose proposals obtained from the relocalization module. If the frame dissimilarity is very low, but the difference between the estimated and retrieved pose is large this could indicate a drift in the tracking. This information could then potentially be used for map correction.

In conclusion, with our current relocalization method we provided a solution to the main causes of tracking failure in systems such as KinectFusion. This has been acknowledged by the Kinect for Windows development team which has integrated our method into the Kinect for Windows SDK[5].

References

[1] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 1403–1410.

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Apr. 2007.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2007, pp. 225–234.

[4] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2320–2327.

[5] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche, "MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2013, pp. 83–88.

[6] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2011, pp. 127–136.

[7] S. Lieberknecht, A. Huber, S. Ilic, and S. Benhimane, "RGB-D camera-based parallel tracking and meshing," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2011, pp. 147–155.

[8] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1352–1359.

[9] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas, "Acquiring 3D indoor environments with variability and repetition," *ACM Trans. Graph.*, vol. 31, no. 6, p. 138, 2012.

[10] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially Extended KinectFusion," in *Proc. RGBD Workshop*, 2012.

[11] H. Roth and M. Vona, "Moving volume kinectfusion," in *Proc. Brit. Mach. Vis. Conf.*, 2012.

[12] J. Chen, D. Bautembach, and S. Izadi, "Scalable real-time volumetric surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 4, p. 113, 2013.

[13] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2013, pp. 209–218.

[14] M. Dou, H. Fuchs, and J.-M. Frahm, "Scanning and tracking dynamic objects with commodity depth cameras," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2013, pp. 99–106.

[15] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *Proc. IEEE Int. Conf. 3D Vis.*, 2013, pp. 1–8.

[16] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys, "Live Metric 3D Reconstruction on Mobile Phones," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 65–72.

[17] V. A. Prisacariu, O. Kahler, D. W. Murray, and I. D. Reid, "Simultaneous 3D tracking and reconstruction on a mobile phone," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2013, pp. 89–98.

[18] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg, "Global Localization from Monocular SLAM on a Mobile Phone," *IEEE Trans. Visualization Comput. Graph.*, vol. 20, no. 4, pp. 531–539, Apr. 2014.

5. http://www.microsoft.com/en-us/kinectforwindows/

[19] J. Platonov, H. Heibel, P. Meier, and B. Grollmann, "A mobile markerless AR system for maintenance and repair," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2006, pp. 105–108.
[20] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu ACM Symp User Interface Softw. Technol.*, 2011, pp. 559–568.
[21] R. Salas-Moreno, B. Glocker, P. Kelly, and A. Davison, "Dense Planar SLAM," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2014.
[22] M. Tatzgern, R. Grasset, D. Kalkofen, and D. Schmalstieg, "Transitional Augmented Reality Navigation for Live Captured Scenes," in *Proc. IEEE Virtual Reality*, 2014, pp. 21–26.
[23] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2011, pp. 719–722.
[24] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust Tracking for Real-Time Dense RGB-D Mapping with Kintinuous," MIT-CSAIL, Tech. Rep. 031, 2012.
[25] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 647–663, 2012.
[26] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-Time RGB-D Camera Relocalization," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2013.
[27] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi, "Multi-Output Learning for Camera Relocalization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1114–1121.
[28] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2930–2937.
[29] G. Reitmayr and T. W. Drummond, "Going out: robust model-based tracking for outdoor augmented reality," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2006, pp. 109–118.
[30] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *Int. J. Robot. Res.*, vol. 30, no. 9, pp. 1100–1123, 2011.
[31] K. Ni, A. Kannan, A. Criminisi, and J. Winn, "Epitomic location recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
[32] E. Eade and T. Drummond, "Unified loop closing and recovery for real time monocular SLAM," in *Proc. Brit. Mach. Vis. Conf.*, 2008.
[33] J. Martinez-Carranza, A. Calway, and W. Mayol-Cuevas, "Enhancing 6D visual relocalisation with depth cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2013, pp. 899–906.
[34] B. Williams, G. Klein, and I. Reid, "Automatic Relocalization and Loop Closing for Real-Time Monocular SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1699–1712, Sep. 2011.
[35] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[36] A. P. Gee and W. Mayol-Cuevas, "6D Relocalisation for RGBD Cameras Using Synthetic View Regression," in *Brit. Mach. Vis. Conf.*, 2012.
[37] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 802–815.
[38] C. Pirchheim, D. Schmalstieg, and G. Reitmayr, "Handling pure camera rotation in keyframe-based SLAM," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2013, pp. 229–238.
[39] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, Jan. 2010.
[40] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Eur. Conf. Comput. Vis.*, 2010, pp. 778–792.
[41] N. Sunderhauf and P. Protzel, "BRIEF-Gist – Closing the Loop by Simple Means," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2011, pp. 1234–1241.
[42] S. A. Winder and M. Brown, "Learning local image descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[43] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, 2006, pp. 2161–2168.
[44] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 448–461, Jan. 2010.
[45] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1465–1479, Jul. 2006.
[46] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.
[47] M. Feuerstein, T. Mussack, S. M. Heining, and N. Navab, "Intraoperative Laparoscope Augmentation for Port Placement and Resection Planning in Minimally Invasive Liver Resection," *IEEE Trans. Med. Imaging*, vol. 27, no. 3, pp. 355–369, Mar. 2008.
[48] B. Schwald and B. De Laval, "An augmented reality system for training and assistance to maintenance in the industrial context," *J. WSCG*, vol. 11, no. 1, 2003.
[49] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust Global Registration," in *Proc. Eur. Symp. Geometry Process.*, 2005.
[50] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 1643–1649.

Ben Glocker received the PhD degree with high distinction from the Technical University of Munich, Munich, Germany, in 2011. He is currently a lecturer in Medical Image Computing at Imperial College London, London, United Kingdom. He is a member of the Biomedical Image Analysis Group in the section of Visual Information Processing at the Department of Computing. His research area includes image analysis and computer vision with a focus on machine learning. He is interested in real-time SLAM and 3D reconstruction and its use in medical applications. Before joining Imperial in 2013, he was a post-doctoral researcher at Microsoft Research Cambridge. He was awarded the Francois Erbsman Prize and won the Werner von Siemens Excellence Award in 2007. He received an honorary mention for the ERCIM Cor Baayen Award 2013 which recognizes promising young researchers in the field of Informatics and Applied Mathematics.

Jamie Shotton studied computer science at the University of Cambridge, Cambridge, United Kingdom, where he remained for his PhD in computer vision and visual object recognition. He joined Microsoft Research in 2008 where he is now a senior researcher in the Machine Learning and Perception group. His research focuses at the intersection of vision, graphics, and machine learning, with particular interests including human pose and shape estimation, object recognition, gesture and action recognition, and medical imaging. He has received multiple Best Paper and Best Demo awards at top academic conferences. His work on machine learning for body part recognition for Kinect was awarded the Royal Academy of Engineering's gold medal MacRobert Award 2011, and he shares Microsoft's Outstanding Technical Achievement Award for 2012 with the Kinect engineering team. In 2014, he received the PAMI Young Researcher Award.

Antonio Criminisi received the degree in electronics engineering at the University of Palermo, Palermo, Italy, in July 1996, and the PhD degree in computer vision at the University of Oxford, Oxford, United Kingdom, in December 1999. He joined Microsoft Research in Cambridge (Machine Learning and Perception Group), United Kingdom, as a visiting researcher in 2000. In February 2001, he moved to the Interactive Visual Media Group in Redmond, WA, as a postdoc, and back to Cambridge as a researcher in October 2002. In September 2011, he became a senior researcher and is now leading the medical image analysis team. His current research interests include medical image analysis, object category recognition, image and video analysis and editing, one-to-one teleconferencing, 3D reconstruction from single and multiple images with application to virtual reality, forensic science and history of art. In October 1990, he was appointed "Alfiere del Lavoro" by the Italian President F. Cossiga for his successful studies. His thesis "Accurate Visual Metrology from Single and Multiple Uncalibrated Images" won the British Computer Society Distinguished Dissertation Award for the year 2000. He was a Research Fellow at Clare Hall College, Cambridge from 2002 to 2005. He has won a number of best paper prizes in top computer vision conferences.

Shahram Izadi is currently a principal researcher at Microsoft Research, Cambridge, United Kingdom, where he leads the interactive 3D technologies (I3D) group. His group straddles many boundaries including human-computer interaction, augmented reality, applied computer vision and graphics, electronics, signal processing, and optics. He also holds a visiting professorship at UCL in the Virtual Environments and Computer Graphics (VECG) group. His research focuses on building new technologies and systems that push the boundaries of human-computer interaction (HCI). This typically involves building new types of cameras, sensors or displays; creating practical algorithms and techniques for these types of novel technologies; as well as designing new user experiences that are enabled through this technical research. He has been at Microsoft Research over nine years, spent time at Xerox PARC before and was awarded a PhD with Tom Rodden and Yvonne Rogers working on the EQUATOR project. He was awarded a TR35 in 2009 and a Microsoft Next award in 2012. He served on the organisational committee for ACM UIST, CHI, Ubicomp, TEI, ITS, ISMAR and CSCW. His work has led to over 80 research papers, and over 80 patents. These include multiple best paper awards at UIST, CHI, ISMAR, Pervasive, CSCW and Ubicomp. His work has led to products such as Microsoft Surface, Sensor-in-Pixel, Microsoft Touch Mouse, Kinect One, and Kinect SDK.