On-the-Fly Adaptation of Regression Forests for Online Camera Relocalisation

Tommaso Cavallari[1] Stuart Golodetz[2*] Nicholas A. Lord[2*]
Julien Valentin[3] Luigi Di Stefano[1] Philip H. S. Torr[2]

[1] Department of Computer Science and Engineering, University of Bologna
[2] Department of Engineering Science, University of Oxford
[3] perceptive[io], Inc.

[1] {tommaso.cavallari,luigi.distefano}@unibo.it
[2] {smg,nicklord,phst}@robots.ox.ac.uk
[3] julien@perceptiveio.com

Abstract

Camera relocalisation is an important problem in computer vision, with applications in simultaneous localisation and mapping, virtual/augmented reality and navigation. Common techniques either match the current image against keyframes with known poses coming from a tracker, or establish 2D-to-3D correspondences between keypoints in the current image and points in the scene in order to estimate the camera pose. Recently, regression forests have become a popular alternative to establish such correspondences. They achieve accurate results, but must be trained offline on the target scene, preventing relocalisation in new environments. In this paper, we show how to circumvent this limitation by adapting a pre-trained forest to a new scene on the fly. Our adapted forests achieve relocalisation performance that is on par with that of offline forests, and our approach runs in under 150ms, making it desirable for real-time systems that require online relocalisation.

1. Introduction

Camera pose estimation is an important problem in computer vision, with applications in simultaneous localisation and mapping (SLAM) [29, 28, 19], virtual and augmented reality [1, 4, 14, 30, 31, 38] and navigation [21]. In SLAM, the camera pose is commonly initialised upon starting reconstruction and then tracked from one frame to the next, but tracking can easily be lost due to e.g. rapid movement or textureless regions in the scene; when this happens, it is important to be able to relocalise the camera with respect to the scene, rather than forcing the user to start the reconstruction again from scratch. Camera relocalisation is also crucial for loop closure when trying to build globally consistent maps [7, 18, 40]. Traditional approaches to camera relocalisation have been based around one of two main paradigms:

(i) Image matching methods match the current image from the camera against keyframes stored in an image database (potentially with some interpolation between keyframes where necessary). For example, Galvez-Lopez et al. [10] describe an approach that computes a bag of binary words based on BRIEF descriptors for the current image and compares it with bags of binary words for keyframes in the database using an L1 score. Gee et al. [12] estimate camera pose from a set of synthetic (i.e. rendered) views of the scene. Their approach is interesting because unlike many image matching methods, they are to some extent able to relocalise from novel poses; however, the complexity increases linearly with the number of synthetic views needed, which poses significant limits to practical use. Glocker et al. [13] encode frames using Randomised Ferns, which when evaluated on images yield binary codes that can be matched quickly by their Hamming distance: as noted in [23], this makes their approach much faster than [12] in practice.

(ii) Keypoint-based methods find 2D-to-3D correspondences between keypoints in the current image and 3D scene points, so as to deploy e.g. a Perspective-n-Point (PnP) algorithm [16] (on RGB data) or the Kabsch algorithm [17] (on RGB-D data) to generate a number of camera pose hypotheses that can be pruned to a single hypothesis using RANSAC [8]. For example, Williams et al. [41] recognise/match keypoints using an ensemble of randomised lists, and exclude unreliable or ambiguous matches when generating hypotheses. Their approach is fast, but needs significant memory to store the lists. Li et al. [23] use graph matching to help distinguish between visually-similar keypoints. Their method uses BRISK descriptors for the keypoints, and runs at around 12 FPS. Sattler et al. [32] describe a large-scale localisation approach that finds

*S. Golodetz and N. Lord assert joint second authorship.

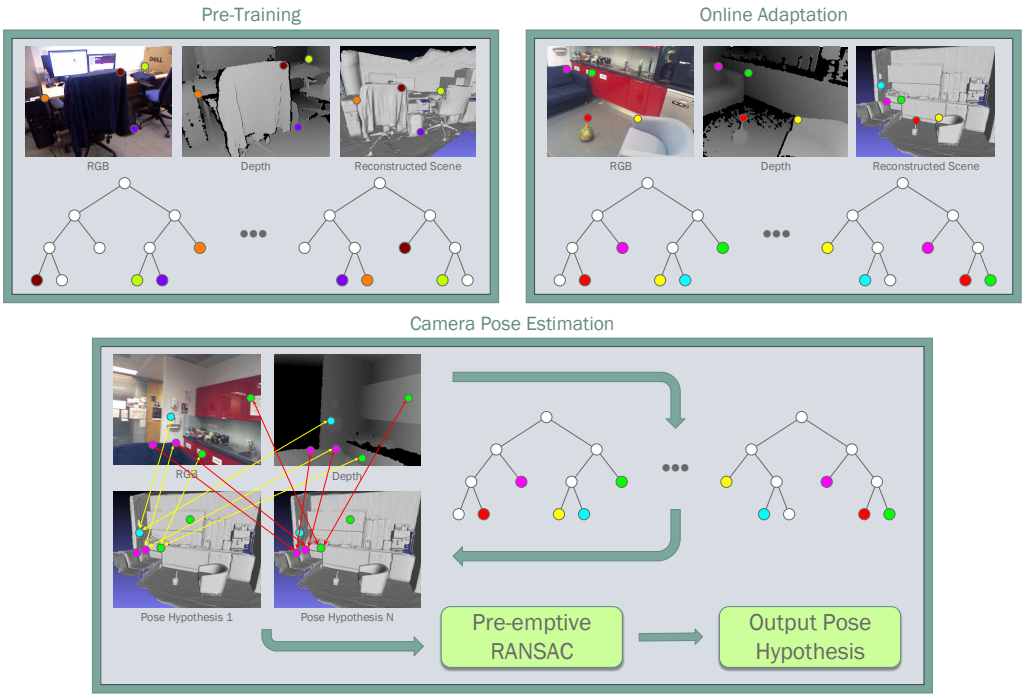


Figure 1: **Overview of our approach**. First, we train a regression forest *offline* to predict 2D-to-3D correspondences for a generic scene. To adapt this forest to a new scene, we remove the scene-specific information in the forest's leaves while retaining the branching structure (with learned split parameters) of the trees; we then refill the leaves *online* using training examples from the new scene. The adapted forest can be deployed to predict correspondences for the new scene that are fed to Kabsch [17] and RANSAC [8] for pose estimation.

correspondences in both the 2D-to-3D and 3D-to-2D directions before applying a 6-point DLT algorithm to compute pose hypotheses. They use a visual vocabulary to order potential matches by how costly they will be to establish.

Some hybrid methods use both paradigms. For example, Mur-Artal et al. [27] describe a relocalisation approach that initially finds pose candidates using bag of words recognition [11], which they incorporate into their larger ORB-SLAM system (unlike [10], they use ORB rather than BRIEF features, which they found to improve performance). They then refine these candidate poses using PnP and RANSAC. Valentin et al. [36] present an approach that finds initial pose candidates using the combination of a retrieval forest and a multiscale navigation graph, before refining them using continuous pose optimisation.

Several less traditional approaches have also been tried. Kendall et al. [20] train a convolutional neural network to directly regress the 6D camera pose from the current image. Deng et al. [6] match a 3D point cloud representing the scene to a local 3D point cloud constructed from a set of query images that can be incrementally extended by the user to achieve a successful match. Lu et al. [24] perform 3D-to-3D localisation that reconstructs a 3D model from a short video using structure-from-motion and matches that against the scene within a multi-task point retrieval framework.

Recently, Shotton et al. [34] proposed the use of a regression forest to directly predict 3D correspondences in the scene for all pixels in the current image. This has two key advantages over traditional keypoint-based approaches: (i) no explicit detection, description or matching of keypoints is required, making the approach both simpler and faster, and (ii) a significantly larger number of points can be deployed to verify or reject camera pose hypotheses. However, it suffers from the key limitation of needing to train a regression forest on the scene *offline* (in advance), which prevents on-the-fly camera relocalisation. Subsequent work has significantly improved upon the relocalisation performance of [34]. For example, Guzman-Rivera et al. [15] rely on multiple regression forests to generate a number of camera pose hypotheses, then cluster them and use the mean pose of the cluster whose poses minimise the reconstruction error as the result. Valentin et al. [37] replace the modes used in the leaves of the forests in [34] with mixtures of anisotropic 3D Gaussians in order to better model uncertainties in the 3D point predictions, and show that by combining this with continuous pose optimisation they can relocalise 40% more frames than [34]. Brachmann et al. [3] deploy a stacked classification-regression forest to achieve results of a quality similar to [37] for RGB-D relocalisation. Massiceti et al. [26] map between regression forests

and neural networks to try to leverage the performance benefits of neural networks for dense regression while retaining the efficiency of random forests for evaluation. They use robust geometric median averaging to achieve improvements of around 7% over [3] for RGB localisation. However, despite all of these advances, none of these papers remove the need to train on the scene of interest in advance.

In this paper, we show that this need for *offline* training on the scene of interest can be overcome through *online* adaptation to a new scene of a regression forest that has been pre-trained on a generic scene. We achieve genuine on-the-fly relocalisation similar to that which can be obtained using keyframe-based approaches [13], but with both significantly higher relocalisation performance in general, and the specific advantage that we can relocalise from novel poses. Indeed, our adapted forests achieve relocalisation performance that is competitive with offline-trained forests, whilst requiring no pre-training on the scene of interest and relocalising in close to real time. This makes our approach a practical and high-quality alternative to keyframe-based methods for online relocalisation in novel scenes.

2. Method

2.1. Overview

Figure 1 shows an overview of our approach. Initially, we train a regression forest *offline* to predict 2D-to-3D correspondences for a *generic* scene, as per [37]. To adapt this forest to a new scene, we remove the contents of the leaf nodes in the forest (i.e. GMM modes and associated covariance matrices) whilst retaining the branching structure of the trees (including learned split parameters). We then adapt the forest *online* to the new scene by feeding training examples down the forest to refill the empty leaves, dynamically learning a set of leaf distributions specific to that scene. Thus adapted, the forest can then be used to predict correspondences for the new scene that can be used for camera pose estimation. Reusing the tree structures spares us from expensive offline learning on deployment in a novel scene, allowing for relocalisation on the fly.

2.2. Details

2.2.1 Offline Forest Training

Training is done as in [37], greedily optimising a standard reduction-in-spatial-variance objective over the randomised parameters of simple threshold functions. Like [37], we make use of 'Depth' and 'Depth-Adaptive RGB' ('DA-RGB') features, centred at a pixel $\mathbf{p}$, as follows:

$$f_{\Omega}^{\text{Depth}} = D(\mathbf{p}) - D\left(\mathbf{p} + \frac{\boldsymbol{\delta}}{D(\mathbf{p})}\right) \tag{1}$$

$$f_{\Omega}^{\text{DA-RGB}} = C(\mathbf{p}, c) - C\left(\mathbf{p} + \frac{\boldsymbol{\delta}}{D(\mathbf{p})}, c\right) \tag{2}$$

In this, $D(\mathbf{p})$ is the depth at $\mathbf{p}$, $C(\mathbf{p}, c)$ is the value of the c^{th} colour channel at $\mathbf{p}$, and Ω is a vector of randomly sampled feature parameters. For 'Depth', the only parameter is the 2D image-space offset $\boldsymbol{\delta}$, whereas 'DA-RGB' adds the colour channel selection parameter $c \in \{R, G, B\}$. We randomly generate 128 values of Ω for 'Depth' and 128 for 'DA-RGB'. We concatenate the evaluations of these functions at each pixel of interest to yield 256D feature vectors.

At training time, a set S of training examples, each consisting of such a feature vector $\mathbf{f} \in \mathbb{R}^{256}$, its corresponding 3D location in the scene and its colour, is assembled via sampling from a ground truth RGB-D video with known camera poses for each frame (obtained by tracking from depth camera input). A random subset of these training examples is selected to train each tree in the forest, and we then train all of the trees in parallel.

Starting from the root of each tree, we recursively partition the set of training examples in the current node into two using a binary threshold function. To decide how to split each node n, we randomly generate a set Θ_n of 512 candidate split parameter pairs, where each $\theta = (\phi, \tau) \in \Theta_n$ denotes the binary threshold function

$$\theta(\mathbf{f}) = \mathbf{f}[\phi] \geq \tau. \tag{3}$$

In this, $\phi \in [0, 256)$ is a randomly-chosen feature index, and $\tau \in \mathbb{R}$ is a threshold, chosen to be the value of feature ϕ in a randomly-chosen training example. Examples that pass the test are routed to the right subtree of n; the remainder are routed to the left. To pick a suitable split function for n, we use exhaustive search to find a $\theta^* \in \Theta_n$ whose corresponding split function maximises the information gain that can be achieved by splitting the training examples that reach n. Formally, the information gain corresponding to split parameters $\theta \in \Theta_n$ is

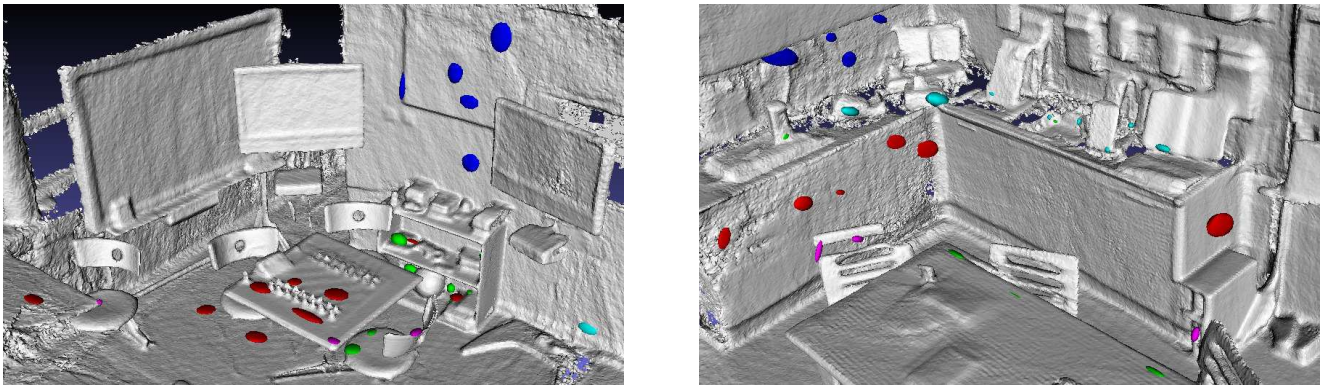
$$V(S_n) - \sum_{i \in \{\text{L,R}\}} \frac{|S_n^i(\theta)|}{|S_n|} V(S_n^i(\theta)), \tag{4}$$

in which $V(X)$ denotes the spatial variance of set X, and $S_n^L(\theta)$ and $S_n^R(\theta)$ denote the left and right subsets into which the set $S_n \subseteq S$ of training examples reaching n is partitioned by the split function denoted by θ. Spatial variance is defined in terms of the log of the determinant of the covariance of a fitted 3D Gaussian [37].

For a given tree, the above process is simply recursed to a maximum depth of 15. As in [37], we train 5 trees per forest. The (approximate, empirical) distributions in the leaves are discarded at the end of this process (we replace them during online forest adaptation, as discussed next).

2.2.2 Online Forest Adaptation

To adapt a forest to a new environment, we replace the distributions discarded from its leaves at the end of pre-training



(a) (b)

Figure 2: An illustrative example of the effect that online adaptation has on a pre-trained forest: (a) shows the modal clusters present in a small number of randomly-selected leaves of a forest pre-trained on the *Chess* scene from the 7-Scenes dataset [34] (the colour of each mode indicates its containing leaf); (b) shows the modal clusters that are added to the same leaves during the process of adapting the forest to the *Kitchen* scene.

with dynamically-updated ones drawn entirely from the new scene. Here, we detail how the new leaf distributions used by the relocaliser are computed and updated online.

We draw inspiration from the use of reservoir sampling [39] in SemanticPaint [38], which makes it possible to store an unbiased subset of an empirical distribution in a bounded amount of memory. On initialisation, we allocate (on the GPU) a fixed-size sample reservoir for each leaf of the existing forest. Our reservoirs contain up to 1024 entries, each storing a 3D (world coordinate) location and an associated colour. At runtime, we pass training examples (as per §2.2.1) down the forest and identify the leaves to which each example is mapped. We then add the 3D location and colour of each example to the reservoirs associated with its leaves. To obtain the 3D locations of the training examples, we need to know the transformation that maps points from camera space to world space. When testing on sequences from a dataset, this is trivially available as the ground truth camera pose, but in a live scenario, it will generally be obtained as the output of a fallible tracker. To avoid corrupting the reservoirs in our forest, we avoid passing new examples down the forest when the tracking is unreliable. We measure tracker reliability using the support vector machine (SVM) approach described in [18]. For frames for which a reliable camera pose *is* available, we proceed as follows:

1. First, we compute feature vectors for a subset of the pixels in the image, as detailed in §2.2.1. We empirically choose our subset by subsampling densely on a regular grid with 4-pixel spacing, i.e. we choose pixels $\{(4i, 4j) \in [0, w) \times [0, h) : i, j \in \mathbb{N}\}$, where w and h are respectively the width and height of the image.

2. Next, we pass each feature vector down the forest, adding the 3D position and colour of the corresponding scene point to the reservoir of the leaf reached in each tree. Our CUDA-based random forest implementation uses the node indexing described in [33].

3. Finally, for each leaf reservoir, we cluster the contained points using a CUDA implementation of Really Quick Shift (RQS) [9] to find a set of modal 3D locations. We sort the clusters in each leaf in decreasing size order, and keep at most 10 modal clusters per leaf. For each cluster we keep, we compute 3D and colour centroids, and a covariance matrix. The cluster distributions are used when estimating the likelihood of a camera pose, and also during continuous pose optimisation (see §2.2.3). Since running RQS over all the leaves in the forest would take too long if run in a single frame, we amortise the cost over multiple frames by updating 256 leaves in parallel each frame in round-robin fashion. A typical forest contains around $42{,}000$ leaves, so each leaf is updated roughly once every 6s.

The aforementioned reservoir size, number of modal clusters per leaf and number of leaves to update per frame were determined empirically to achieve online processing rates.

Figure 2 illustrates the effect that online adaptation has on a pre-trained forest: (a) shows the modal clusters present in a few randomly-selected leaves of a forest pre-trained on the *Chess* scene from the 7-Scenes dataset [34]; (b) shows the modal clusters that are added to the same leaves during the process of adapting the forest to the *Kitchen* scene. Note that whilst the positions of the predicted modes have (unsurprisingly) completely changed, the split functions in the forest's branch nodes (which we preserve) still do a good job of routing similar parts of the scene into the same leaves, enabling effective sampling of 2D-to-3D correspondences for camera pose estimation.

2.2.3 Camera Pose Estimation

As in [37], camera pose estimation is based on the pre-emptive, locally-optimised RANSAC of [5]. We begin by randomly generating an initial set of up to 1024 pose hypotheses. A pose hypothesis $H \in \mathbf{SE}(3)$ is a transform that maps points in camera space to world space. To generate each pose hypothesis, we apply the Kabsch algorithm [17] to 3 point pairs of the form $(\mathbf{x}_i^{\mathcal{C}}, \mathbf{x}_i^{\mathcal{W}})$, where $\mathbf{x}_i^{\mathcal{C}} = D(\mathbf{u}_i)K^{-1}(\mathbf{u}_i^\top, 1)$ is obtained by back-projecting a randomly-chosen point $\mathbf{u}_i$ in the live depth image D into camera space, and $\mathbf{x}_i^{\mathcal{W}}$ is a corresponding scene point in world space, randomly sampled from $M(\mathbf{u}_i)$, the modes of the leaves to which the forest maps $\mathbf{u}_i$. In this, K is the intrinsic calibration matrix for the depth camera. Before accepting a hypothesis, we subject it to a series of checks:

1. First, we randomly choose one of the three point pairs $(\mathbf{x}_i^{\mathcal{C}}, \mathbf{x}_i^{\mathcal{W}})$ and compare the RGB colour of the corresponding pixel $\mathbf{u}_i$ in the colour input image to the colour centroid of the mode (see §2.2.2) from which we sampled $\mathbf{x}_i^{\mathcal{W}}$. We reject the hypothesis iff the L0 distance between the two exceeds a threshold.

2. Next, we check that the three hypothesised scene points are sufficiently far from each other. We reject the hypothesis iff the minimum distance between any pair of points is less than 30cm.

3. Finally, we check that the distances between all scene point pairs and their corresponding back-projected depth point pairs are sufficiently similar, i.e. that the hypothesised transform is 'rigid enough'. We reject the hypothesis iff this is not the case.

If a hypothesis gets rejected by one of the checks, we try to generate an alternative hypothesis to replace it. In practice, we use 1024 dedicated threads, each of which attempts to generate a single hypothesis. Each thread continues generating hypotheses until either (a) it finds a hypothesis that passes all of the checks, or (b) a maximum number of iterations is reached. We proceed with however many hypotheses we obtain by the end of this process.

Having generated our large initial set of hypotheses, we next aggressively cut it down to a much smaller size by scoring each hypothesis and keeping the 64 lowest-energy transforms (if there are fewer than 64 hypotheses, we keep all of them). To score the hypotheses, we first select an initial set $I = \{i\}$ of 500 pixel indices in D, and back-project the denoted pixels $\mathbf{u}_i$ to corresponding points $\mathbf{x}_i^{\mathcal{C}}$ in camera space as described above. We then score each hypothesis H by summing the Mahalanobis distances between the transformations of each $\mathbf{x}_i^{\mathcal{C}}$ under H and their nearest modes:

$$E(H) = \sum_{i \in I} \left(\min_{(\boldsymbol{\mu}, \Sigma) \in M(\mathbf{u}_i)} \left\| \Sigma^{-\frac{1}{2}} (H\mathbf{x}_i^{\mathcal{C}} - \boldsymbol{\mu}) \right\| \right) \quad (5)$$

After this initial cull, we use pre-emptive RANSAC to prune the remaining ≤ 64 hypotheses to a single, final hypothesis. We iteratively (i) expand the sample set I (by adding 500 new pixels each time), (ii) refine the pose candidates via Levenberg-Marquardt optimisation [22, 25] of the energy function E, (iii) re-evaluate and re-score the hypotheses, and (iv) discard the worse half. In practice, the actual optimisation is performed not in $\mathbf{SE}(3)$, where it would be hard to do, but in the corresponding Lie algebra, $\mathfrak{se}(3)$. The details of this process can be found in [37], and a longer explanation of Lie algebras can be found in [35].

This process yields a single pose hypothesis, which we can then return if desired. In practice, however, further pose refinement is sometimes possible. For example, if our relocaliser is integrated into an open-source 3D reconstruction framework such as InfiniTAM [18], we can attempt to refine the pose further using ICP [2]. Since tasks such as 3D reconstruction are one of the key applications of our approach, we report results both with and without ICP in Table 1.

3. Experiments

We perform both quantitative and qualitative experiments to evaluate our approach. In §3.1, we compare our *adaptive* approach to state-of-the-art *offline* relocalisers that have been trained directly on the scene of interest. We show that our adapted forests achieve competitive relocalisation performance despite being trained on very different scenes, enabling their use for *online* relocalisation. In §3.2, we show that we can perform this adaptation on-the-fly from live sequences, allowing us to support tracking loss recovery in interactive scenarios. In §3.3, we evaluate how well our approach generalises to novel poses in comparison to a keyframe-based random fern relocaliser based on [13]. This relocaliser is also practical for on-the-fly relocalisation (hence its use in InfiniTAM [18]), but its use of keyframes prevents it from generalising well to novel poses. By contrast, we are able to relocalise well even from poses that are quite far away from the training trajectory. Finally, in §3.4, we compare the speed of our approach with random ferns during both normal operation (i.e. when the scene is being successfully tracked) and relocalisation. Our approach is slower than random ferns, but remains close to real-time and achieves much higher relocalisation performance. Further analysis can be found in the supplementary material.

3.1. Adaptation Performance

In evaluating the extent to which we are able to adapt a regression forest that has been pre-trained on a different scene to the scene of interest, we seek to answer two questions. First, how does an adapted forest compare to one that has been pre-trained offline on the target scene? Second, to what extent does an adapted forest's performance depend on the scene on which it has been pre-trained? To an-

Training Scene		Relocalisation Performance on Test Scene							
		Chess	**Fire**	**Heads**	**Office**	**Pumpkin**	**Kitchen**	**Stairs**	**Average (all scenes)**
Chess	Reloc	99.8%	95.7%	95.5%	91.7%	82.8%	77.9%	25.8%	81.3%
	+ ICP	99.9%	97.8%	99.5%	94.1%	91.3%	83.3%	28.4%	84.9%
Fire	Reloc	98.4%	96.9%	98.2%	89.7%	80.5%	71.9%	28.6%	80.6%
	+ ICP	99.1%	99.2%	99.9%	92.1%	89.1%	81.7%	31.0%	84.6%
Heads	Reloc	98.0%	91.7%	100%	73.1%	77.5%	67.1%	21.8%	75.6%
	+ ICP	99.3%	92.3%	100%	81.1%	87.7%	82.0%	31.9%	82.0%
Office	Reloc	99.2%	96.5%	99.7%	97.6%	84.0%	81.7%	33.6%	84.6%
	+ ICP	99.4%	99.0%	100%	98.2%	91.2%	87.0%	35.0%	87.1%
Pumpkin	Reloc	97.5%	94.9%	96.9%	82.7%	83.5%	70.4%	30.7%	75.5%
	+ ICP	98.9%	97.6%	99.4%	86.9%	91.2%	82.3%	32.4%	84.1%
Kitchen	Reloc	99.9%	95.4%	98.0%	93.3%	83.2%	86.0%	28.2%	83.4%
	+ ICP	99.9%	98.2%	100%	94.5%	90.4%	88.1%	31.3%	86.1%
Stairs	Reloc	97.3%	95.4%	97.9%	90.8%	80.6%	74.5%	45.7%	83.2%
	+ ICP	98.0%	97.4%	99.8%	92.1%	89.5%	81.0%	46.6%	86.3%
Ours (Author's Desk)	Reloc	97.3%	95.7%	97.3%	83.7%	85.3%	71.8%	24.3%	79.3%
	+ ICP	99.2%	97.7%	100%	88.2%	90.6%	82.6%	31.0%	84.2%
Average	Reloc	98.4%	95.3%	97.9%	87.8%	82.2%	75.2%	29.8%	80.9%
	+ ICP	99.2%	97.4%	99.8%	90.9%	90.1%	83.5%	33.5%	84.9%

Table 1: The performance of our *adaptive* approach after pre-training on various scenes of the 7-Scenes dataset [34]. We show the scene used to pre-train the forest in each version of our approach in the left column. The pre-trained forests are adapted *online* for the test scene, as described in the main text. The percentages denote proportions of test frames with $\leq$ 5cm translational error and $\leq 5^\circ$ angular error.

swer both of these questions, we compare the performances of adapted forests pre-trained on a variety of scenes (each scene from the 7-Scenes dataset [34], plus a novel scene containing the first author's desk) to the performances of forests trained offline on the scene of interest using state-of-the-art approaches [34, 15, 37, 3].

The exact testing procedure we use for our approach is as follows. First, we pre-train a forest on a generic scene and remove the contents of its leaves, as described in §2: this process runs *offline* over a number of hours or even days (but we only need to do it once). Next, we adapt the forest by feeding it new examples from a training sequence captured on the scene of interest: this runs *online* at frame rates (in a real system, this allows us to start relocalising almost immediately whilst training carries on in the background, as we show in §3.2). Finally, we test the adapted forest by using it to relocalise from every frame of a separate testing sequence captured on the scene of interest.

As shown in Table 1, the results are very accurate. Whilst there are certainly some variations in the performance achieved by adapted forests pre-trained on different scenes (in particular, forests trained on the *Heads* and *Pumpkin* scenes from the dataset are slightly worse), the differences are not profound: in particular, relocalisation performance seems to be more tightly coupled to the difficulty of the scene of interest than to the scene on which the forest was pre-trained. Notably, all of our adapted forests achieve results that are within striking distance of the state-of-the-art *offline* methods (Table 2), and are considerably better than those that can be achieved by online competitors such as the keyframe-based random fern relocaliser implemented in InfiniTAM [13, 18] (see §3.3). Nevertheless, there is clearly a trade-off to be made here between performance and practicality: pre-training on the scene of interest is impractical for on-the-fly relocalisation, but achieves somewhat better results, probably due to the opportunity afforded to adapt the structure of the forest to the target scene.

This drop in performance in exchange for practicality can be mitigated to some extent by refining our relocaliser's pose estimates using the ICP-based tracker [2] in InfiniTAM [19]. Valentin et al. [37] observe that the 5cm/5° error metric commonly used to evaluate relocalisers is 'fairly strict and should allow any robust model-based tracker to resume'. In practice, ICP-based tracking is in many cases able to resume from initial poses with even greater error: indeed, as Table 1 shows, with ICP refinement enabled, we are able to relocalise from a significantly higher proportion of test frames. Whilst ICP could clearly also be used to refine the results of offline methods, what is important in this case is that ICP is fast and does not add significantly to the overall runtime of our approach, which remains close to real time. As such, refining *our* pose estimates using ICP yields a high-quality relocaliser that is still practical for online use.

3.2. Tracking Loss Recovery

In §3.1, we investigated our ability to adapt a forest to a new scene by filling its leaves with data from a training sequence for that scene, before testing the adapted forest on a separate testing sequence shot on the same scene. Here, we quantify our ability to perform this adaptation *on the fly* by filling the leaves frame-by-frame from the testing sequence: this allows recovery from tracking loss in an interactive scenario without the need for prior training on anything other than the live sequence, making our approach extremely con-

Scene	[34]	[15]	[37]	[3]	Us	Us+ICP
Chess	92.6%	96%	99.4%	99.6%	99.2%	99.4%
Fire	82.9%	90%	94.6%	94.0%	96.5%	99.0%
Heads	49.4%	56%	95.9%	89.3%	99.7%	100%
Office	74.9%	92%	97.0%	93.4%	97.6%	98.2%
Pumpkin	73.7%	80%	85.1%	77.6%	84.0%	91.2%
Kitchen	71.8%	86%	89.3%	91.1%	81.7%	87.0%
Stairs	27.8%	55%	63.4%	71.7%	33.6%	35.0%
Average	67.6%	79.3%	89.5%	88.1%	84.6%	87.1%

Table 2: Comparing our *adaptive* approach to state-of-the-art *offline* methods on the 7-Scenes dataset [34] (the percentages denote proportions of test frames with $\leq$ 5cm translation error and $\leq 5^\circ$ angular error). For our method, we report the results obtained by adapting a forest pre-trained on the *Office* sequence (from Table 1). We are competitive with, and sometimes better than, the offline methods, without needing to pre-train on the test scene.

venient for tasks such as interactive 3D reconstruction.

Our testing procedure is as follows: at each new frame (except the first), we assume that tracking has failed, and try to relocalise using the forest we have available at that point; we record whether or not this succeeds. Regardless, we then restore the ground truth camera pose (or the tracked camera pose, in a live sequence) and, provided tracking hasn't actually failed, use examples from the current frame to continue training the forest. As Figure 3 shows, we are able to start relocalising almost immediately in a live sequence (in a matter of frames, typically 4–6 are enough). Subsequent performance then varies based on the difficulty of the sequence, but rarely drops below 80%, except for the challenging *Stairs* sequence. This makes our approach highly practical for interactive relocalisation, something we also show in our supplementary video.

3.3. Generalisation to Novel Poses

To evaluate how well our approach generalises to novel poses, we examine how the proportion of frames we can relocalise decreases as the distance of the (ground truth) test poses from the training trajectory increases. We compare our approach with the keyframe-based relocaliser in InfiniTAM [18], which is based on the random fern approach of Glocker et al. [13]. Relocalisation from novel poses is a well-known failure case of keyframe-based methods, so we would expect the random fern approach to perform poorly away from the training trajectory; by contrast, it is interesting to see the extent to which our approach can relocalise from a wide range of novel poses.

We perform the comparison separately for each 7-Scenes sequence, and then aggregate the results. For each sequence, we first group the test poses into bins by pose novelty. Each bin is specified in terms of a maximum translation and rotation difference of a test pose with respect to the training trajectory (for example, poses that are within 5cm and 5° of any training pose are assigned to the first bin, remaining poses that are within 10cm and 10° are assigned to the second bin, etc.). We then determine the proportion of the test poses in each bin for which it is possible to relocalise to within 5cm translational error and 5° angular error using (a) the random fern approach, (b) our approach without ICP and (c) our approach with ICP. As shown in Figure 4, the decay in performance experienced as the test poses get further from the training trajectory is much less severe with our approach than with random ferns.

A qualitative example of our ability to relocalise from novel poses is shown in Figure 5. In the main figure, we show a range of test poses from which we can relocalise in

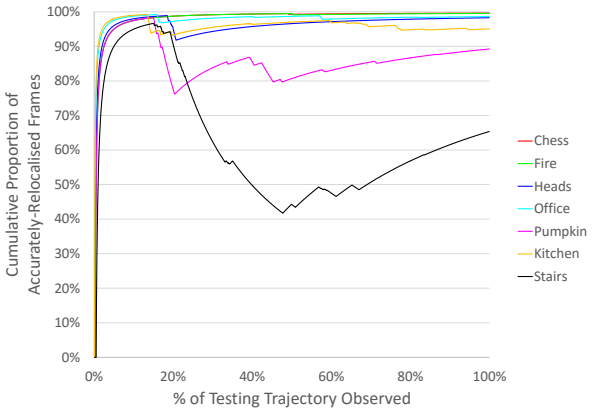


Figure 3: The performance of our approach for tracking loss recovery (§3.2). Filling the leaves of a forest pre-trained on *Office* frame-by-frame *directly* from the *testing* sequence, we are able to start relocalising almost immediately in new scenes. This makes our approach highly practical in interactive scenarios such as 3D reconstruction.

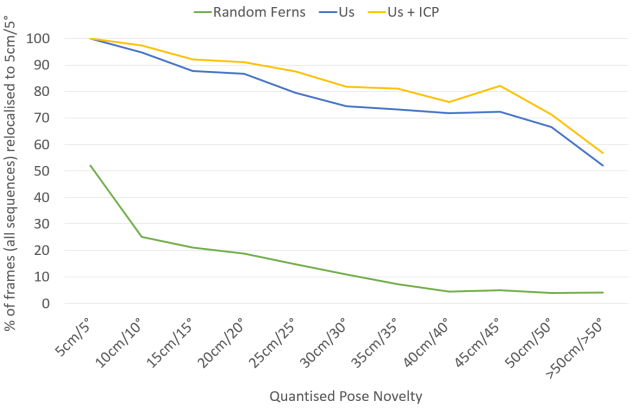


Figure 4: Evaluating how well our approach generalises to novel poses in comparison to a keyframe-based random fern relocaliser based on [13]. The performance decay experienced as test poses get further from the training trajectory is much less severe with our approach than with random ferns.

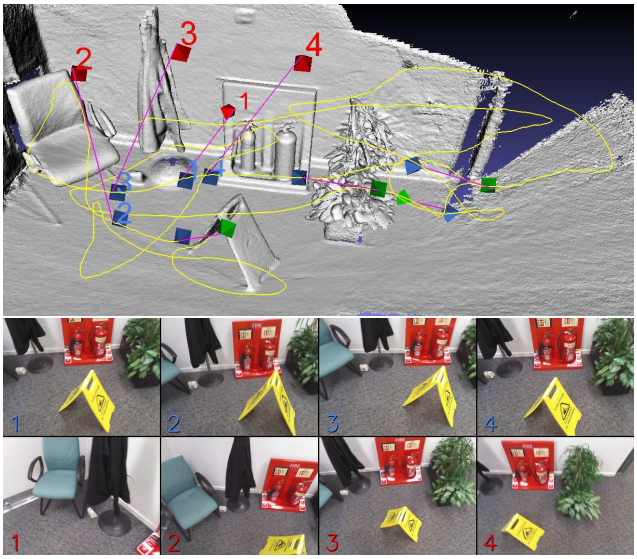


Figure 5: A qualitative example of novel poses from which we are able to relocalise to within 5cm/5° on the *Fire* sequence from 7-Scenes [34]. Pose novelty measures the distance of a test pose from a nearby pose (blue) on the training trajectory (yellow). We can relocalise from both easy poses (up to 35cm/35° from the training trajectory, green) and hard poses (> 35cm/35°, red). The images below the main figure show views of the scene from the training poses and testing poses indicated.

the *Fire* scene, linking them to nearby poses on the training trajectory so as to illustrate their novelty in comparison to poses on which we have trained. The most difficult of these test poses are also shown in the images below alongside their nearby training poses, visually illustrating the significant differences between the two.

As Figures 4 and 5 illustrate, we are already quite effective at relocalising from poses that are significantly different from those on which we have trained; nevertheless, further improvements seem possible. For example, one interesting extension of this work might be to explore the possibility of using rotation-invariant split functions in the regression forest to improve its generalisation capabilities.

3.4. Timings

To evaluate the usefulness of our approach for on-the-fly relocalisation in new scenes, we compare it to the keyframe-based random fern relocaliser implemented in InfiniTAM [13, 18]. To be practical in a real-time system, a relocaliser needs to perform in real time during normal operation (i.e. for online training whilst successfully tracking the scene), and ideally take no more than around 200ms for relocalisation itself (when the system has lost track). As a result, relocalisers such as [34, 15, 37, 3, 26], whilst achieving impressive results, are not practical in this context due to their need for offline training on the scene of interest.

	Random Ferns [13, 18]	**Us**
Per-Frame Training	0.9ms	9.8ms
Relocalisation	10ms	141ms

Table 3: Comparing the typical timings of our approach vs. random ferns during both normal operation and relocalisation. Our approach is slower than random ferns, but achieves significantly higher relocalisation performance, especially from novel poses. All of our experiments are run on a machine with an Intel Core i7-4960X CPU and an NVIDIA GeForce Titan Black GPU.

As shown in Table 3, the random fern relocaliser is fast both for online training and relocalisation, taking only 0.9ms per frame to update the keyframe database, and 10ms to relocalise when tracking is lost. However, speed aside, the range of poses from which it is able to relocalise is quite limited. By contrast, our approach, whilst taking 9.8ms for online training and 141ms for actual relocalisation, can relocalise from a much broader range of poses, whilst still running at acceptable speeds. Additionally, it should be noted that our current research-focused implementation is not heavily optimised, making it plausible that it could be sped up even further with additional engineering effort.

4. Conclusion

In recent years, offline approaches that use regression to predict 2D-to-3D correspondences [34, 15, 37, 3, 26] have achieved state-of-the-art camera relocalisation results, but their adoption for online relocalisation in practical systems such as InfiniTAM [19, 18] has been hindered by the need to train extensively on the target scene ahead of time.

We show how to circumvent this limitation by adapting offline-trained regression forests to novel scenes online. Our adapted forests achieve relocalisation performance on 7-Scenes [34] that is competitive with the offline-trained forests of existing methods, and our approach runs in under 150ms, making it competitive in practice with fast keyframe-based approaches such as random ferns [13, 18]. Compared to such approaches, we are also much better able to relocalise from novel poses, freeing the user from manually searching for known poses when relocalising.

Acknowledgements

We would like to thank Victor Prisacariu and Olaf Kähler for providing us with the InfiniTAM source code.
This work was supported by the EPSRC, ERC grant ERC-2012-AdG 321162-HELIOS, EPSRC grant Seebibyte EP/M013774/1 and EPSRC/MURI grant EP/N019474/1.

References

[1] H. Bae, M. Walker, J. White, Y. Pan, Y. Sun, and M. Golparvar-Fard. Fast and scalable structure-from-motion based localization for high-precision mobile augmented reality systems. *The Journal of Mobile User Experience*, 5(1):1–21, 2016. 1

[2] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *TPAMI*, 14(2):239–256, February 1992. 5, 6

[3] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In *CVPR*, 2016. 2, 3, 6, 7, 8

[4] R. Castle, G. Klein, and D. W. Murray. Video-rate Localization in Multiple Maps for Wearable Augmented Reality. In *IEEE International Symposium on Wearable Computers*, pages 15–22, 2008. 1

[5] O. Chum, J. Matas, and J. Kittler. Locally Optimized RANSAC. In *Joint Pattern Recognition Symposium*, pages 236–243, 2003. 5

[6] L. Deng, Z. Chen, B. Chen, Y. Duan, and J. Zhou. Incremental image set querying based localization. *Neurocomputing*, 208:315–324, 2016. 2

[7] N. Fioraio, J. Taylor, A. Fitzgibbon, L. D. Stefano, and S. Izadi. Large-Scale and Drift-Free Surface Reconstruction Using Online Subvolume Registration. In *CVPR*, pages 4475–4483, 2015. 1

[8] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *CACM*, 24(6):381–395, 1981. 1, 2

[9] B. Fulkerson and S. Soatto. Really quick shift: Image segmentation on a GPU. In *ECCV*, pages 350–358, 2010. 4

[10] D. Gálvez-López and J. D. Tardós. Real-Time Loop Detection with Bags of Binary Words. In *IROS*, pages 51–58, 2011. 1, 2

[11] D. Gálvez-López and J. D. Tardós. Bags of Binary Words for Fast Place Recognition in Image Sequences. *RO*, 28(5):1188–1197, 2012. 2

[12] A. P. Gee and W. Mayol-Cuevas. 6D Relocalisation for RGBD Cameras Using Synthetic View Regression. In *BMVC*, pages 1–11, 2012. 1

[13] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi. Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding. *TVCG*, 21(5):571–583, May 2015. 1, 3, 5, 6, 7, 8

[14] S. Golodetz*, M. Sapienza*, J. P. C. Valentin, V. Vineet, M.-M. Cheng, V. A. Prisacariu, O. Kähler, C. Y. Ren, A. Arnab, S. L. Hicks, D. W. Murray, S. Izadi, and P. H. S. Torr. SemanticPaint: Interactive Segmentation and Learning of 3D Worlds. In *ACM SIGGRAPH Emerging Technologies*, page 22, 2015. 1

[15] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi. Multi-Output Learning for Camera Relocalization. In *CVPR*, pages 1114–1121, 2014. 2, 6, 7, 8

[16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004. 1

[17] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 1, 2, 5

[18] O. Kähler, V. A. Prisacariu, and D. W. Murray. Real-Time Large-Scale Dense 3D Reconstruction with Loop Closure. In *ECCV*, pages 500–516, 2016. 1, 4, 5, 6, 7, 8

[19] O. Kähler*, V. A. Prisacariu*, C. Y. Ren, X. Sun, P. Torr, and D. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices. *TVCG*, 21(11):1241–1250, 2015. 1, 6, 8

[20] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, pages 2938–2946, 2015. 2

[21] Y. H. Lee and G. Medioni. RGB-D camera based wearable navigation system for the visually impaired. *CVIU*, 149:3–20, 2016. 1

[22] K. Levenberg. A Method for the Solution of Certain Problems in Least Squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944. 5

[23] S. Li and A. Calway. RGBD Relocalisation Using Pairwise Geometry and Concise Key Point Sets. In *ICRA*, pages 6374–6379, 2015. 1

[24] G. Lu, Y. Yan, L. Ren, J. Song, N. Sebe, and C. Kambhamettu. Localize Me Anywhere, Anytime: A Multi-task Point-Retrieval Approach. In *ICCV*, pages 2434–2442, 2015. 2

[25] D. W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2), 1963. 5

[26] D. Massiceti, A. Krull, E. Brachmann, C. Rother, and P. H. S. Torr. Random Forests versus Neural Networks – What's Best for Camera Localization? *arXiv preprint arXiv:1609.05797*, 2016. 2, 8

[27] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *RO*, 31(5):1147–1163, October 2015. 2

[28] R. Mur-Artal and J. D. Tardós. Fast Relocalisation and Loop Closing in Keyframe-Based SLAM. In *ICRA*, pages 846–853, 2014. 1

[29] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *ISMAR*, pages 127–136, 2011. 1

[30] R. Paucher and M. Turk. Location-based augmented reality on mobile phones. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition – Workshops*, pages 9–16, 2010. 1

[31] N. L. Rodas, F. Barrera, and N. Padoy. Marker-less AR in the Hybrid Room using Equipment Detection for Camera Relocalization. In *MICCAI*, pages 463–470, 2015. 1

[32] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization. *TPAMI*, PP(99), 2016. 1

[33] T. Sharp. Implementing Decision Trees and Forests on a GPU. In *ECCV*, pages 595–608, 2008. 4

[34] 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 *CVPR*, pages 2930–2937, 2013. 2, 4, 6, 7, 8

[35] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Imperial College London, 2012. 5

[36] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin. Learning to Navigate the Energy Landscape. In *3DV*, pages 323–332, 2016. 2

[37] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. Torr. Exploiting Uncertainty in Regression Forests for Accurate Camera Relocalization. In *CVPR*, pages 4400–4408, 2015. 2, 3, 5, 6, 7, 8

[38] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. Torr. SemanticPaint: Interactive 3D Labeling and Learning at your Fingertips. *TOG*, 34(5):154, 2015. 1, 4

[39] J. S. Vitter. Random Sampling with a Reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985. 4

[40] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *RSS*, 2015. 1

[41] B. Williams, G. Klein, and I. Reid. Automatic Relocalization and Loop Closing for Real-Time Monocular SLAM. *TPAMI*, 33(9):1699–1712, September 2011. 1