Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images

Jamie Shotton Ben Glocker Christopher Zach Shahram Izadi Antonio Criminisi Andrew Fitzgibbon
Microsoft Research, Cambridge, UK

Abstract

We address the problem of inferring the pose of an RGB-D camera relative to a known 3D scene, given only a single acquired image. Our approach employs a regression forest that is capable of inferring an estimate of each pixel's correspondence to 3D points in the scene's world coordinate frame. The forest uses only simple depth and RGB pixel comparison features, and does not require the computation of feature descriptors. The forest is trained to be capable of predicting correspondences at any *pixel, so no interest point detectors are required. The camera pose is inferred using a robust optimization scheme. This starts with an initial set of hypothesized camera poses, constructed by applying the forest at a small fraction of image pixels. Preemptive RANSAC then iterates sampling more pixels at which to evaluate the forest, counting inliers, and refining the hypothesized poses. We evaluate on several varied scenes captured with an RGB-D camera and observe that the proposed technique achieves highly accurate relocalization and substantially out-performs two state of the art baselines.*

1. Introduction

This paper presents a new, efficient algorithm for estimating the camera pose from a single RGB-D image, relative to a known scene (or environment). This has important applications in robotics (the 'lost robot' problem), SLAM, augmented reality, and navigation. A standard approach for solving the problem is first to find a set of putative correspondences between image pixels and 3D points in the scene, and second to optimize the camera pose to minimize some energy function defined over these correspondences. In this work, we first demonstrate how regression forests can be used to predict the correspondences, and further show how to optimize the camera pose efficiently.

Our main contribution is the *scene coordinate regression forest* (SCoRe Forest). As illustrated in Fig. 1, the forest is trained to directly predict correspondences from any image pixel to points in the scene's 3D world coordinate frame. The aim is that, in one go, the forest can remove the need for the traditional pipeline of feature detection, description, and matching. A SCoRe Forest is trained on a particular scene, using RGB-D images with known camera poses. The depth maps and camera poses are sufficient to compute scene coordinate training labels at every pixel. These labels are used in a standard regression objective to learn the forest. SCoRe Forests employ only simple RGB and depth image pixel comparison features which are fast to compute.

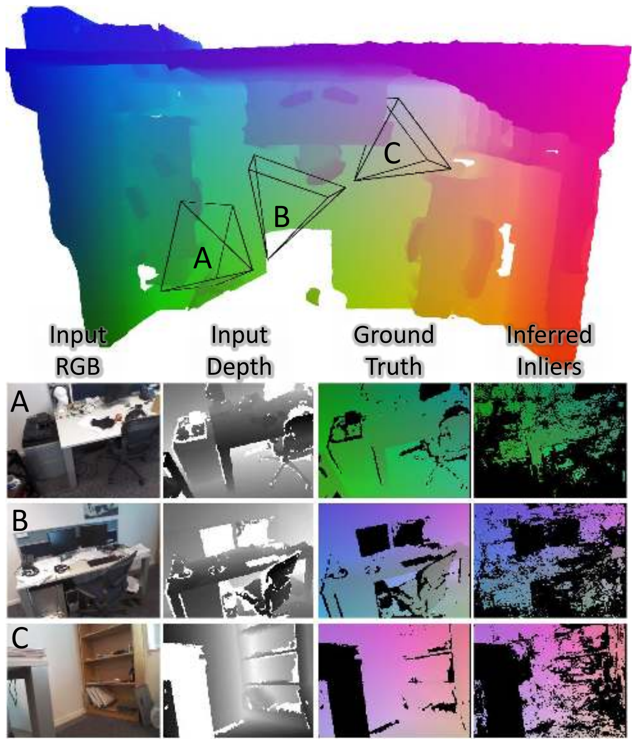

Figure 1. **Scene coordinate regression labels.** (Top) A 3D representation of a scene's shared world coordinate frame, with overlaid ground truth camera frusta for the images below. The color visualization maps scene coordinates to the RGB cube. A scene coordinate regression forest (SCoRe Forest) is trained to infer the scene coordinates at *any* image pixel. (Bottom) Three test frames: the input RGB and depth images; the ground truth scene coordinate pixel labels; and the inliers inferred by the SCoRe Forest after camera pose optimization. For this visualization we show all inlier pixels, but note that the optimization algorithm only actually evaluates the forest at a much sparser set of pixels. Fig. 7 shows example inferred camera poses.

Our second contribution is an efficient test-time camera pose optimization algorithm based on RANSAC; see Fig. 2.

The algorithm assumes only a single RGB-D image as input. We optimize an energy function that measures the number of pixels for which the SCoRe Forest predictions agree with a given pose hypothesis. Because we trained the forest *densely*, we are free to test the forest at whichever (valid) image pixels we please. The RANSAC algorithm can thus efficiently evaluate the SCoRe Forest at only a sparse, randomly sampled set of pixels. Note that the optimization does not require an explicit 3D model of the scene; instead, the forest implicitly encodes the scene structure. This could be useful when a 3D model is not available, or in low power (*e.g.* cellphone) scenarios. Our approach is also completely stateless: we are able to localize the camera from a single frame without tracking.

We validate the proposed camera pose estimation technique on a new dataset of seven varied scenes which will be made available for research purposes. Our experiments demonstrate superior localization accuracy compared to two state of the art baselines, and high accuracy at recognizing to which scene a particular image belongs.

1.1. Related work

Two common approaches for image-based camera relocalization use global image matching and sparse feature matching, though other approaches exist *e.g.* [23].

In global image matching (*e.g.* [17]), an approximate camera pose is hypothesized by computing whole-image similarity measures between a query image and a set of keyframes with known associated keyposes. Given the set of best matching images, the final camera pose is computed as a weighted average of the keyposes. This approach was extended to use synthetic RGB-D views in [11].

A perhaps more prevalent alternative uses sparse keypoint matching. Here a set of interest points are detected in the image, assigned a descriptor based on appearance, and matched against a database of descriptors. Much work has focused on efficiency, scalability, and learning of feature detection [13, 26], description [4, 27, 34], and matching [18, 25, 29]. Closely related to our approach, Lepetit *et al.* [18] learn to match particular discrete features in images with random forests. Our approach goes beyond this by using a regression approach such that every pixel can provide correspondence to the scene, without feature detectors.

Given sparse keypoint matches, pose estimation is achieved either via intermediate image retrieval [2, 8, 14] or by directly establishing 2D-3D correspondences for a robust pose optimization [19, 28]. Efficient keypoint-based relocalization modules have been proposed for visual SLAM [9, 33] and robotics [30].

A limitation of keypoint-based approaches lies in their inherent sparse representation of the scene: the number of correctly detected and matched points directly impacts the accuracy of the pose estimation. Our SCoRe Forests can be applied as sparsely or densely as required, avoiding this problem.

Sharing a similar spirit to our work, the LayoutCRF [12, 35] defines a coarse grid of classification labels that roughly cover an object, and then exploits layout consistency terms in an MRF optimization framework to segment the objects. Our approach instead regresses more precise labels defined on 3D world space, and avoids expensive image-space optimization. Taylor *et al.* [32] use a regression forest to infer correspondences between each depth image pixels and points in a canonical articulated 3D human mesh. Our approach extends [32] by: introducing an efficient camera pose optimization that evaluates the forest sparsely; showing that the regression approach works well on arbitrary scene model topologies and on RGB-D input data; and demonstrating how to effectively exploit the several predictions given by multiple trees in a forest.

2. Scene Coordinate Regression Forests

In this section we provide some background on regression forests, detail our new scene coordinate pixel labeling, and discuss how this labeling is used to train a forest for each scene. Sec. 3 will then describe how the camera pose is optimized by evaluating the forest at a sparse set of pixels.

2.1. Regression forests

We give a brief introduction to regression forests; for more details, please see [7]. Decision forests have proven highly capable learning machines. They are most commonly used for classification tasks, *e.g.* [1, 18, 31], though increasingly for regression problems, *e.g.* [10, 20, 32]. We employ a fairly standard regression forest approach, optimizing a reduction-in-spatial-variance objective.

A regression forest is an ensemble of T decision trees, each consisting of split (internal) and leaf nodes. Each split node contains a 'weak learner' represented by its parameters $\boldsymbol{\theta} = (\boldsymbol{\phi}, \tau)$: feature parameters $\boldsymbol{\phi}$ (see below), and a scalar threshold τ. To evaluate a regression tree at a 2D pixel location $\mathbf{p}$ in an image, we start at the root node and descend to a leaf by repeatedly evaluating the weak learner:

$$h(\mathbf{p}; \boldsymbol{\theta}_n) = \left[f_{\boldsymbol{\phi}_n}(\mathbf{p}) \geq \tau_n \right] , \tag{1}$$

where n denotes the index of a node in the tree, $[\cdot]$ is the 0-1 indicator, and $f_{\boldsymbol{\phi}}$ is one of the feature response functions we define below in Sec. 2.2. If $h(\mathbf{p}; \boldsymbol{\theta}_n)$ evaluates to 0, the path branches to the left child of n, otherwise it branches to the right. This repeats until leaf node $l_t(\mathbf{p})$ is reached for tree t in the forest.

Each leaf node in a regression tree stores a distribution $\mathrm{P}_l(\mathbf{m})$ over a continuous variable $\mathbf{m}$. In our application these $\mathbf{m} \in \mathbb{R}^3$ are the coordinates in the scene's 3D world space, as described in Sec. 2.3. For efficiency, we represent the distribution $\mathrm{P}_l(\mathbf{m})$ as a set $\mathcal{M}_l$ of modes of the distribution, which are found using mean shift. The final prediction of the forest at pixel $\mathbf{p}$ is simply the union of these modes across all trees: $\mathcal{M}(\mathbf{p}) = \bigcup_t \mathcal{M}_{l_t(\mathbf{p})}$.

2.2. Image features

We investigate three variants of regression forests, each of which uses a different combination of RGB and depth features. We will refer to the variants as 'Depth', 'Depth-Adaptive RGB' (DA-RGB), and 'Depth-Adaptive RGB + Depth' (DA-RGB + D) forests.

All features are based on simple pixel comparisons [18, 31] and so are extremely fast to evaluate. The two types of feature responses can be computed as follows:

$$f_{\boldsymbol{\phi}}^{\text{depth}}(\mathbf{p}) = D\left(\mathbf{p} + \frac{\boldsymbol{\delta}_1}{D(\mathbf{p})}\right) - D\left(\mathbf{p} + \frac{\boldsymbol{\delta}_2}{D(\mathbf{p})}\right) \quad (2)$$

$$f_{\boldsymbol{\phi}}^{\text{da-rgb}}(\mathbf{p}) = I\left(\mathbf{p} + \frac{\boldsymbol{\delta}_1}{D(\mathbf{p})}, c_1\right) - I\left(\mathbf{p} + \frac{\boldsymbol{\delta}_2}{D(\mathbf{p})}, c_2\right) \quad (3)$$

Here, $\boldsymbol{\delta}$ indicates a 2D offset, $D(\mathbf{p})$ indicates a depth pixel lookup, and $I(\mathbf{p}, c)$ indicates an RGB pixel lookup in channel c. Each split node in the forest stores a unique set of parameters $\boldsymbol{\phi}_n \subseteq \{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, c_1, c_2, z\}$, with $z \in \{\text{depth}, \text{da-rgb}\}$ indicating the type of feature to use. Pixels with undefined depth and those outside the image boundary are assigned $D = 6\text{m}$, and are not used as examples for training or test.

These features can implicitly encode contextual information, as the offsets can be fairly long range. The division by $D(\mathbf{p})$ makes the features largely depth invariant, and is similar in spirit to [36]. We assume a reasonable registration of depth and RGB images, such as is provided by standard RGB-D camera APIs. However, the registration need not be perfect as the forest will learn some degree of tolerance to misregistration.

2.3. Scene coordinate labels

One of the main contributions of this work is the use of scene coordinates to define the labels used to train the regression forest. By using scene coordinate labels, the forest will learn to directly predict the position in the scene's world space that corresponds to a test pixel.

Our training set consists of a set of RGB-D frames with known 6 d.o.f. camera pose matrices H that encode the 3D rotation and translation from camera space to world space. This data could be captured in several ways, for example by tracking from depth camera input [15, 21], or by using dense reconstruction and tracking from RGB input [22].

Our labels are defined as follows. At pixel $\mathbf{p}$, the calibrated depth $D(\mathbf{p})$ allows us to compute the 3D camera space coordinate $\mathbf{x}$. Using homogeneous coordinates, this camera position can be transformed into the scene's world coordinate frame as $\mathbf{m} = H\mathbf{x}$. Our labels are simply defined as these scene world positions, $\mathbf{m}$.

We train the forest using pixels drawn from all training images, so the forest can be applied at *any* test image pixel. In particular, one can evaluate the forest at any sparse set of test pixels. If the forest were a perfect predictor, only three pixel predictions would be required to infer the camera pose. In practice, the forest instead makes noisy predictions, and so we employ the efficient optimization described in Sec. 3 to accurately infer the camera pose.

2.4. Forest training

Given the scene coordinate pixel labeling defined above, we can now grow the regression forest using the standard greedy forest training algorithm [7], summarized next. For each tree, we randomly choose a set $\mathcal{S}$ of labeled example pixels $(\mathbf{p}, \mathbf{m})$.[1] The tree growing then proceeds recursively, starting at the root node. At each node n, a set of candidate weak learner parameters $\boldsymbol{\theta}$ is sampled at random. Each candidate $\boldsymbol{\theta}$ is evaluated in turn by (1) to partition the set $\mathcal{S}_n$ into left and right subsets $\mathcal{S}_n^{\mathrm{L}}$ and $\mathcal{S}_n^{\mathrm{R}}$ respectively. Given this partition, a tree training objective function $Q(\boldsymbol{\theta})$ is computed. We have found the following simple reduction-in-spatial-variance objective to work well:

$$Q(\mathcal{S}_n, \boldsymbol{\theta}) = V(\mathcal{S}_n) - \sum_{d \in \{\mathrm{L},\mathrm{R}\}} \frac{|\mathcal{S}_n^d(\boldsymbol{\theta})|}{|\mathcal{S}_n|} V(\mathcal{S}_n^d(\boldsymbol{\theta})) \,, \quad (4)$$

$$\text{with} \quad V(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{p},\mathbf{m}) \in \mathcal{S}} \|\mathbf{m} - \bar{\mathbf{m}}\|_2^2 \,, \quad (5)$$

and $\bar{\mathbf{m}}$ being the mean of $\mathbf{m}$ in $\mathcal{S}$. The candidate parameter set $\boldsymbol{\theta}$ which results in the largest reduction in variance is taken, and the training recurses on the resulting left and right children. Tree growing terminates when a node reaches a maximum depth $D_{\max}$, or the set $\mathcal{S}_n$ has only one element.

Mode fitting. Once the tree has been grown, the final stage of training is to summarize the distribution over $\mathbf{m}$ as a set of modes $\mathcal{M}_l$, for each leaf node l. To reduce training time, we sub-sample the set $\mathcal{S}_l$ of training pixels reaching leaf l to at most $N_{\text{ss}} = 500$ examples. We then run mean shift mode detection [6] with a Gaussian kernel of bandwidth $\kappa = 0.1\text{m}$. This clusters the points $\mathbf{m}$ in $\mathcal{S}_l$ into a small set of modes. In our current implementation, we keep only a single mode at a leaf node: the mode to which the largest number of examples was assigned.

3. Camera Pose Optimization

The regression forest described in the previous section is capable of associating scene coordinates with any 2D image pixel. We now discuss how to use this information to

[1]For notational clarity $\mathbf{p}$ is used to uniquely index a pixel within a particular image. Training pixels are sampled up to the image boundary.

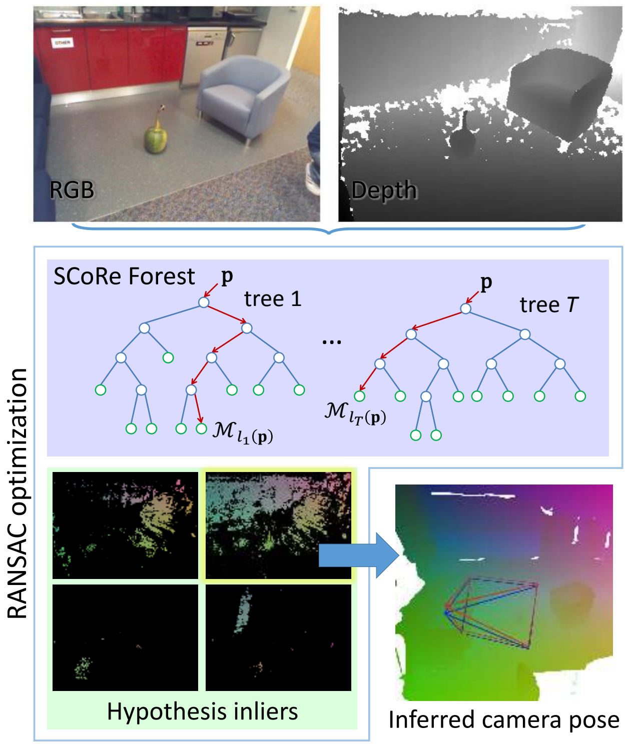

Figure 2. **Camera pose estimation.** Top: An example RGB-D test input. Middle: Our RANSAC optimization uses a scene coordinate regression forest (SCoRe Forest) to obtain image to scene correspondences. The algorithm maintains a set of inlier pixels for each of several camera pose hypotheses. Bottom right: The hypothesis with the lowest energy (highest number of inliers) is chosen as the final inferred pose (shown as the blue frustum; the ground truth is shown in red). For clarity of exposition, we show all image pixels that are inliers to each hypothesis; in fact our algorithm samples pixels *sparsely* and does not need to evaluate the SCoRe Forest at every pixel.

estimate the camera location and orientation. The problem is cast as the energy minimization

$$H^* = \underset{H}{\operatorname{argmin}}\, E(H) \tag{6}$$

over the camera pose matrix H. An overview of the method is given in Fig. 2.

3.1. Energy function

We define our energy function as follows

$$E(H) = \sum_{i\in\mathcal{I}} \rho\left(\min_{\mathbf{m}\in\mathcal{M}_i} \|\mathbf{m} - H\mathbf{x}_i\|_2\right) = \sum_{i\in\mathcal{I}} e_i(H)\ , \tag{7}$$

where: $i \in \mathcal{I}$ is a pixel index; ρ is a robust error function; $\mathcal{M}_i = \mathcal{M}(\mathbf{p}_i)$ represents the set of modes (3D locations in the scene's *world* space) predicted by the trees in the forest at pixel $\mathbf{p}_i$; and $\mathbf{x}_i$ are the 3D coordinates in *camera* space corresponding to pixel $\mathbf{p}_i$, obtained by back-projecting the depth image pixels. We use a top-hat error function ρ with a width of 0.1m. Pixels for which ρ evaluates to 0 are considered *inliers*, and pixels for which ρ evaluates to 1 are considered *outliers*. The energy function above thus counts the number of outliers for a given camera hypothesis H.

Note that computing this energy does not require an explicit 3D model of the scene: the model is implicitly encoded in the regression forest. Because the forest has been trained to work at *any* image pixel, we can randomly sample pixels at test time. This sampling avoids both the need to compute interest points and the expense of densely evaluating the forest. The summation in (7) is thus computed over a subset $\mathcal{I}$ of all possible image pixels; the larger the size of this subset, the more useful the energy E can be at ranking pose hypotheses. The minimization over predictions $\mathbf{m} \in \mathcal{M}_i$ means that at each pixel, the mode that is closest to the transformed observed pixel will be chosen. A consequence of this is that the minimization will infer at each pixel which tree in the forest gave the best prediction under a given hypothesis.

3.2. Optimization

To optimize this energy, we use an adapted version of preemptive RANSAC [24] (see Algorithm 1). The algorithm starts by sampling a set of K_{init} initial hypotheses. It then randomly samples a new batch of B pixels, evaluates the forest at these pixels, and updates the energies for each hypothesis based on the forest predictions. The hypotheses are re-ranked by energy, and the highest energy half of the hypotheses is discarded ('preempted'). Each remaining hypothesis is then refined [5] based on the set of inliers computed as a by-product of evaluating the energy at Line 9 of Algorithm 1. The **while** loop terminates when only a single hypothesis remains.

The iteration could instead be stopped earlier, either for efficiency, or if desired to obtain the top K hypotheses. The algorithm as presented evaluates the forest on-the-fly; alternatively the forest could be evaluated at the $B \log_2 K_{\text{init}}$ required pixels in advance, though this would require extra storage. Other RANSAC schedules (*e.g.* no preemption) are possible, though not investigated here.

Initial hypothesis sampling. Each initial pose hypothesis is sampled as follows. The forest is evaluated at three pixels (the minimal number required), and at each of those pixels a random mode $\mathbf{m} \in \mathcal{M}_i$ is sampled. These putative correspondences are passed to the Kabsch algorithm [16] (also known as orthogonal Procrustes alignment) which uses a singular value decomposition (SVD) to solve for the camera pose hypothesis with least squared error.

Pose refinement. Experimentally we found a pose refinement step [5] to be crucial to achieving accurate localization. We simply re-run the Kabsch algorithm on the enlarged set of inliers. For efficiency, we only store and update

Algorithm 1 Pseudocode for RANSAC optimization

```
 1: $K \leftarrow K_{\text{init}}$
 2: sample initial hypotheses $\{H_k\}_{k=1}^{K}$
 3: initialize energies $E_k \leftarrow 0$ for $k = 1, \ldots, K$
 4: while $K > 1$ do
 5:     sample random set of $B$ test pixels $\mathcal{I}$
 6:     for all $i \in \mathcal{I}$ do
 7:         evaluate forest to obtain $\mathcal{M}_i$
 8:         for all $k \in \{1, \ldots, K\}$ do
 9:             $E_k \leftarrow E_k + e_i(H_k)$
10:     sort hypotheses $(H_k, E_k)$ by $E_k$
11:     $K \leftarrow \lfloor \frac{K}{2} \rfloor$
12:     refine hypotheses $H_k$ for $k = 1, \ldots, K$
13: return best pose $H_1$ and energy $E_1$
```

the means and covariance matrices used by the SVD. Note that the refinement step means the final pose can be much more accurate than the individual forest pixel predictions of the inliers.

4. Evaluation

4.1. Dataset

We introduce a new RGB-D dataset, '7 Scenes', to evaluate our technique and compare with other approaches. We will release this dataset to facilitate future research and comparison with our approach. All scenes were recorded from a handheld Kinect RGB-D camera at 640×480 resolution. We use an implementation of the KinectFusion system [15, 21] to obtain the 'ground truth' camera tracks, and a dense 3D model (used only for visualization and the ICP experiments below). Several sequences were recorded per scene by different users, and split into distinct training and testing sequence sets. An overview of the scenes and camera tracks are shown in Fig. 3, and more details are given in Table 1. Both RGB and depth images exhibit ambiguities (*e.g.* repeated steps in Stairs), specularities (*e.g.* reflective cupboards in RedKitchen), motion-blur, lighting conditions, flat surfaces, and sensor noise. The varying difficulties of the scenes are reflected in the error metrics, consistently across all approaches.

4.2. Metrics

As our main test metric we report the percentage of test frames for which the inferred camera pose is essentially 'correct'. We employ a fairly strict definition of correct: the pose must be within 5cm translational error and $5°$ angular error of the ground truth (for comparison, our scenes have size up to 6m^3). The belief is that correctly localizing the pose under this metric might already be sufficient for some augmented reality applications, and should certainly be sufficient to restart any good model-based tracking system.

4.3. Baselines

We compare our approach against two complementary baseline methods: the 'sparse' baseline exploits matches between extracted features and a sparse 3D point cloud model, while the 'tiny-image' baseline matches down-sampled whole images to hypothesize putative camera poses.

Sparse baseline. This baseline uses state of the art feature-matching techniques. The approach employs the fast ORB descriptor [27] for feature detection and description. We build a (sparse) 3D point cloud with attached ORB descriptors using the RGB-D training images. At test time (using only RGB), features are matched to the 3D point cloud by hashing 14-bit descriptor substrings. We consider a 2D-3D correspondence a putative match if the Hamming distance between descriptor bit-strings is at most 50. The set of potential inliers is geometrically verified using RANSAC and the perspective 3-point method. The final camera pose is refined using a gold-standard method computed on all inlier 2D-3D correspondences.

Tiny-image baseline. This baseline represents state of the art whole-image matching approaches [11, 17]. All training images are stored with their camera poses, after downsampling to 40×30 pixels and applying a Gaussian blur of $\sigma = 2.5$ pixels [17]. At test time, to achieve the best possible accuracy for this baseline (at the expense of speed), we use brute-force matching against the entire training set, using the normalized Euclidean distance of [11]. The camera pose is finally computed as a weighted average of the poses of the 100 closest matches [11]. We compare below with RGB, depth, and RGB-D variants of this approach.

4.4. Main results

We present our main results and comparisons in Table 1. Our approach considerably outperforms both baselines on all but one of the scenes. The tiny-image baseline fails almost completely under our strict metric: the poses of the images in the training set are simply too far from the poses in the test set, and this approach cannot generalize well.[2] The sparse baseline does a reasonable job, though struggles when too few features are detected. This can happen under motion blur, when viewing textureless surfaces, and when noise levels are high. While our approach is also sparse, the SCoRe Forests can be evaluated at any pixel in the image, and so we are not dependent on feature detection. We present some qualitative results in Fig. 7, showing both successes and failures of our algorithm and the sparse baseline.

Of the three forest feature modalities we test, depth-only always performs worst. This is to be expected given the large amount of noise (especially holes) in the depth im-

[2]The tiny-image baseline does however provide a sensible *rough* localization, good enough for model-based pose refinement (Sec. 4.7).

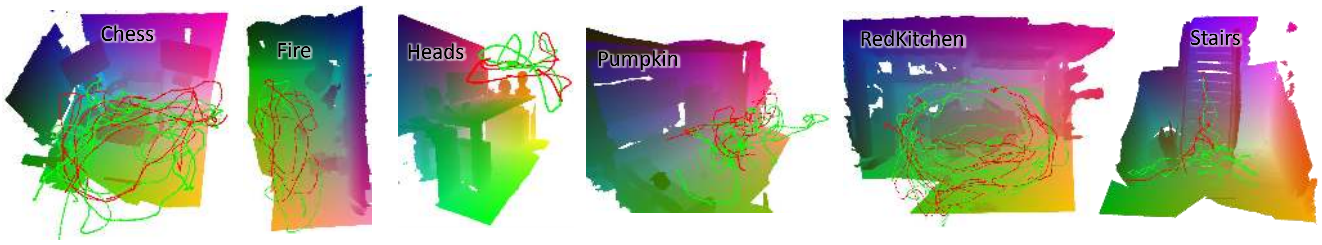

Figure 3. **The 7 Scenes dataset.** These renderings show the scene coordinate representations for each scene as RGB colors. The green and red camera tracks show the positions of the cameras in the training and test sequences. The Office scene is shown in Fig. 1.

	Spatial	**# Frames**		**Baselines**		**Our Results**			**Frame-to-Frame**
Scene	**Extent**	Train	Test	Tiny-image RGB-D	Sparse RGB	Depth	DA-RGB	DA-RGB + D	**Tracking**
Chess	3m^3	4k	2k	0.0%	70.7%	82.7%	**92.6%**	91.5%	95.5%
Fire	4m^3	2k	2k	0.5%	49.9%	44.7%	**82.9%**	74.7%	86.2%
Heads	2m^3	1k	1k	0.0%	**67.6%**	27.0%	49.4%	46.8%	50.7%
Office	5.5m^3	6k	4k	0.0%	36.6%	65.5%	74.9%	**79.1%**	86.8%
Pumpkin	6m^3	4k	2k	0.0%	21.3%	58.6%	**73.7%**	72.7%	76.1%
RedKitchen	6m^3	7k	5k	0.0%	29.8%	61.3%	71.8%	**72.9%**	82.4%
Stairs	5m^3	2k	1k	0.0%	9.2%	12.2%	**27.8%**	24.4%	39.2%

Table 1. **Main results.** We list: properties of our dataset; the results of the two baselines; the results of our approach using SCoRe Forests trained with three different feature types; and results from a simple frame-to-frame extension of our approach (Sec. 4.5), here using DA-RGB features. The percentages are the proportion of 'correct' test frames (within 5cm translational and 5° angular error). Our approach outperforms both baselines on all but one scene. The tiny-image baseline gives only imprecise localization and so fails almost completely under our tight metric (though see also Fig. 6). We tested multiple random restarts of RANSAC; all standard deviations were $< 1\%$.

ages. We also observe that DA-RGB performs better than DA-RGB + D in 5 out of 7 scenes. This slightly surprising result may be due to sub-optimal settings of the forest training parameters (see below), but in any case both modalities outperform the baseline in all but one scene.

In Fig. 4 we see that our inferred camera poses form a remarkably smooth track compared to the sparse baseline. This is likely because our algorithm is able to sample more putative matches than the sparse feature-based approach. Of course, if one were building a real tracking system, both approaches could be smoothed much further.

Failure modes. Our main failure modes are (i) imprecise localization, and (ii) failures due to ambiguities. As an example, if we widen our metric to allow 10cm translational errors, our DA-RGB forests achieve 100% accuracy on Chess, 58.8% on Heads, and 49.0% on Stairs. If we further use an oracle to pick the most accurate pose from the top 10 hypotheses, we achieve 68.4% on Stairs, highlighting the inherent ambiguity in this sequence. (See also Fig. 7 bottom right).

Pose refinement. The pose refinement step in Algorithm 1 proved crucial to achieving good results. With this turned off, our DA-RGB forests achieve only 31% accuracy on Chess and 9.5% on Office, for example.

Parameter settings. In all the experiments presented, the following parameters were used: initial number of hypotheses $K_{\text{init}} = 1024$; batch size $B = 500$; $T = 5$ trees in the forest, trained to depth $D_{\max} = 16$; 500 training images per tree and 5000 example pixels per training image, both randomly chosen; feature offsets $\boldsymbol{\delta}$ having a maximum of ± 130 pixel meters; and $|\mathcal{M}(l)| = 1$, *i.e.* one mode prediction $\mathbf{m}$ per tree. The sparse baseline uses almost identical RANSAC parameters. The tiny-image baseline uses the parameters from [11, 17].

We have not optimized the parameters of our approach: they were chosen by hand once and fixed. Preliminary investigation suggests considerable insensitivity of the pose optimization to changes in the training parameters, though it is reasonable to assume a more thorough optimization of these parameters would improve our accuracy.

Image resolution. Fig. 5(a) illustrates that our approach is robust to nearest-neighbor image downsampling.

Timings. Our unoptimized C++ test time implementation takes approximately 100ms per frame on a single modern CPU core. Of course, this depends heavily on the RANSAC parameters, and one could easily trade speed for accuracy. In comparison, the sparse baseline (also not optimized) takes approximately 250ms per frame, and the (deliberately inefficient brute-force) tiny-image baseline takes 500ms-3000ms according to the number of images in the training database. Training takes around 1-10 minutes per scene according to parameter settings.

4.5. Frame-to-frame tracking

A simple adaptation to our approach allows a basic form of frame-to-frame tracking. Here, we initialize one hypothesis (out of the K_{init}) with the camera pose inferred at the previous frame. This can bias the optimization to finding a good solution when the camera motion is not too great. We achieved the improvements in pose accuracy shown in the final column of Table 1. For all scenes there is an improve-

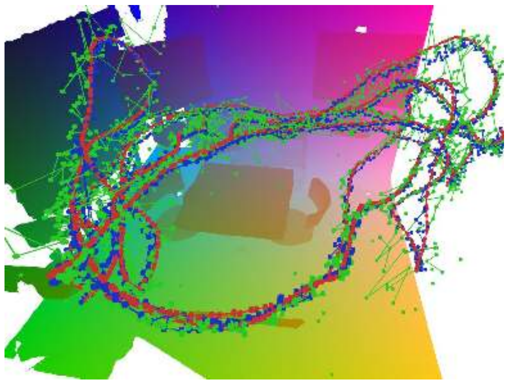
Figure 4. **Camera tracks for the Chess scene.** Red: ground truth. Blue: our result using a DA-RGB forest. Green: sparse baseline. Our algorithm gives remarkably smooth camera tracks from single frames at a time.

ment over our standard algorithm, greater than 10 percentage points for the Office, RedKitchen, and Stairs results. We expect an even greater improvement could be obtained by initializing (possibly multiple) hypotheses from the previous frame using a motion model.

4.6. Scene recognition

Beyond relocalizing within a particular scene, one might also want to recognize to which scene a particular camera view belongs. We built the following simple scene recognition system to investigate this. For a given test frame, our pose optimization algorithm is run, separately for each scene's SCoRe Forest. The scene with the lowest energy (largest number of inliers) is then taken as the result.

Evaluated on every 20th frame in our entire test set using the DA-RGB modality, the confusion matrix is given in Fig. 5(b). We observe very high scene recognition accuracy: the mean of the diagonal is 96.6%, with several scenes achieving perfect recognition. The energy (7) returned by the RANSAC optimization appears to be an accurate measure for disambiguating scenes.

An alternative, not investigated here, would be to embed all the scenes in a shared 3D world so that the coordinate pixel labels uniquely identify a point in a particular scene. A single forest could thus encapsulate all scenes, and scene recognition would simply be a by-product of pose estimation. A downside of this approach would be the need to retrain everything in order to add new scenes.

4.7. Model-based pose refinement

The results above demonstrate that accurate relocalization is achievable without an explicit 3D model. However, if such a model *is* available (*e.g.* from [15, 21]), then it is possible to refine our inferred camera pose against that model. To investigate, we ran an experiment which performs an ICP-based pose refinement [3] starting at our inferred pose. In Fig. 6 we compare the pose accuracy after ICP for our algorithm and the two baselines. We further

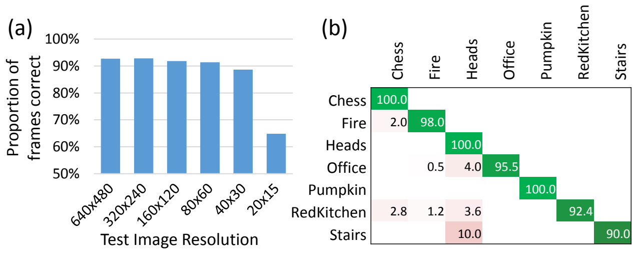

Figure 5. **(a)** Effect of image resolution on accuracy for DA-RGB Chess. **(b)** Scene recognition confusion matrix. For Chess, Heads, and Pumpkin we achieve correct recognition for all test frames.

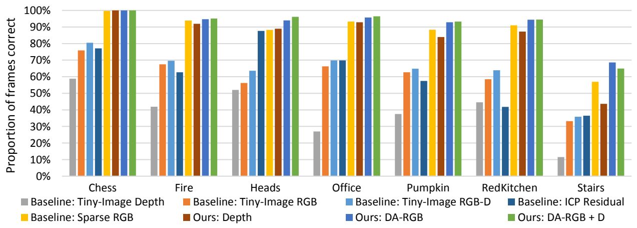

Figure 6. **Accuracy after ICP to a 3D model.** As expected, all results improve compared to Table 1.

add a brute-force 'ICP Residual' baseline which, for each test frame, runs ICP starting from each training keyposes (for tractability, a frame becomes a keypose if it is $>$ 5cm away from the previous selected keypose), and takes the resulting pose with lowest ICP residual error. Our DA-RGB and DA-RGB + D forests outperform all baselines on all scenes, and even give a perfect result for Chess. Note that the complete failure of the tiny-image baseline in Table 1 is much improved, suggesting that this baseline approach is only useful when paired with a refinement step.

5. Conclusions

We have demonstrated how scene coordinate regression forests (SCoRe Forests) can be trained to directly predict correspondences between image pixels and a scene's world space. For known environments, SCoRe Forests thus offer an alternative to the traditional pipeline of sparse feature detection, description, and matching. Our efficient RANSAC optimization is able to sparsely sample image pixels at which to evaluate the forest. The comparison on the challenging new 7 Scenes dataset with two baselines indicates that our algorithm achieves state of the art camera relocalization.

As future work we believe that our approach will extend very naturally to RGB-only test images. We also plan to investigate sampling additional training data by rendering new views from 3D scene reconstructions. Finally, we hope that an optimized implementation could be extended to support on-line model learning and relocalization.

Acknowledgements. We thank Pushmeet Kohli and the anonymous reviewers for their valuable input.

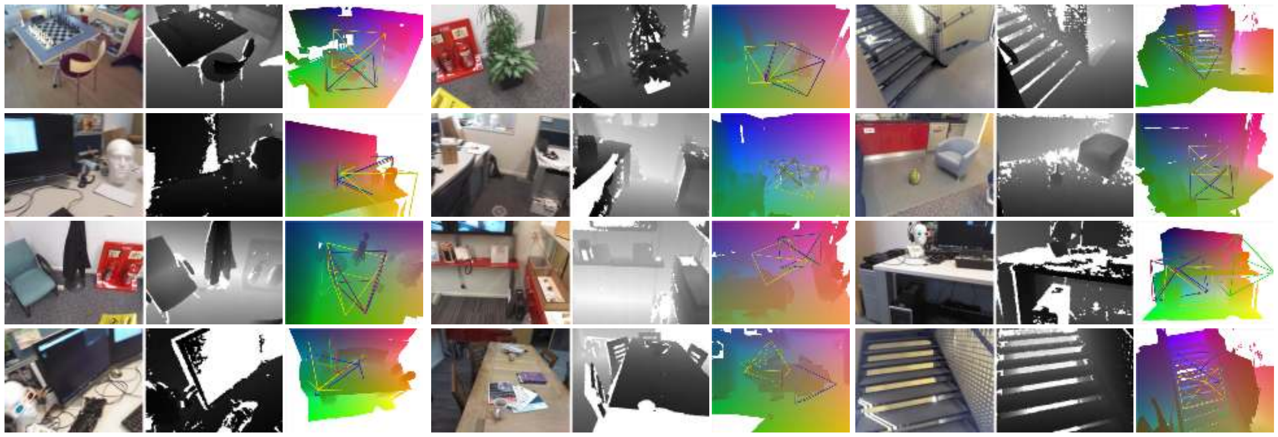

Figure 7. **Example results.** In each test example, we show the RGB and depth images, and a 3D visualization of the underlying scene with camera frusta overlaid: the ground truth pose (red), our inferred pose using a DA-RGB SCoRe Forest (blue), the sparse baseline result (green), and the closest pose in the training set (yellow). Observe good generalization across a variety of scenes to views not present during training. The bottom row shows failure cases, *e.g*. due to repeated structures in the scene. Best viewed digitally at high zoom.

References

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7), 1997. 2

[2] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, and M. Pollefeys. Leveraging 3D city models for rotation invariant place-of-interest recognition. *IJCV*, 2011. 2

[3] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Trans. PAMI*, 14(2), 1992. 7

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Proc. ECCV*, 2010. 2

[5] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *In Proc. DAGM*, 2003. 4

[6] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. PAMI*, 24(5), 2002. 3

[7] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013. 2, 3

[8] Z. Dong, G. Zhang, J. Jia, and H. Bao. Keyframe-based real-time camera tracking. In *Proc. ICCV*, 2009. 2

[9] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular SLAM. In *Proc. BMVC*, 2008. 2

[10] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough Forests for object detection, tracking, and action recognition. *IEEE Trans. PAMI*, 33(11), 2011. 2

[11] A. Gee and W. Mayol-Cuevas. 6D relocalisation for RGBD cameras using synthetic view regression. In *Proc. BMVC*, 2012. 2, 5, 6

[12] D. Hoiem, C. Rother, and J. Winn. 3D LayoutCRF for multi-view object class recognition and segmentation. In *Proc. CVPR*, 2007. 2

[13] S. Holzer, J. Shotton, and P. Kohli. Learning to efficiently detect repeatable interest points in depth data. In *Proc. ECCV*, 2012. 2

[14] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From SfM point clouds to fast location recognition. In *Proc. CVPR*, 2009. 2

[15] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. UIST*, 2011. 3, 5, 7

[16] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica*, 1976. 4

[17] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *Proc. ECCV*, 2008. 2, 5, 6

[18] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Trans. PAMI*, 28(9), 2006. 2, 3

[19] Y. Li, N. Snavely, and D. Huttenlocher. Location recognition using prioritized feature matching. In *Proc. ECCV*, 2010. 2

[20] A. Montillo and H. Ling. Age regression from faces using random forests. In *ICIP*, 2009. 2

[21] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. ISMAR*, 2011. 3, 5, 7

[22] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. CVPR*, 2011. 3

[23] K. Ni, A. Kannan, A. Criminisi, and J. Winn. Epitomic location recognition. *IEEE Trans. PAMI*, 2009. 2

[24] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Proc. ICCV*, 2003. 4

[25] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, 2006. 2

[26] E. Rosten, R. Porter, and T. Drummond. FASTER and better: A machine learning approach to corner detection. *IEEE Trans. PAMI*, 32, 2010. 2

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *Proc. ICCV*, 2011. 2, 5

[28] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *Proc. ICCV*, 2011. 2

[29] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proc. CVPR*, 2007. 2

[30] S. Se, D. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Trans. on Robotics*, 21(3), 2005. 2

[31] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *Proc. CVPR*, 2011. 2, 3

[32] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proc. CVPR*, 2012. 2

[33] B. Williams, G. Klein, and I. Reid. Automatic relocalization and loop closing for real-time monocular SLAM. *IEEE Trans. PAMI*, 33(9), 2011. 2

[34] S. Winder and M. Brown. Learning local image descriptors. In *Proc. CVPR*, 2007. 2

[35] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proc. CVPR*, 2006. 2

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