Exploiting Uncertainty in Regression Forests for Accurate Camera Relocalization

Julien Valentin
University of Oxford

Matthias Nießner
Stanford University

Jamie Shotton
Microsoft Research

Andrew Fitzgibbon
Microsoft Research

Shahram Izadi
Microsoft Research

Philip Torr
University of Oxford

Abstract

Recent advances in camera relocalization use predictions from a regression forest to guide the camera pose optimization procedure. In these methods, each tree associates one pixel with a point in the scene's 3D world coordinate frame. In previous work, these predictions were point estimates and the subsequent camera pose optimization implicitly assumed an isotropic distribution of these estimates. In this paper, we train a regression forest to predict mixtures of anisotropic 3D Gaussians and show how the predicted uncertainties can be taken into account for continuous pose optimization. Experiments show that our proposed method is able to relocalize up to 40% *more frames than the state of the art.*

1. Introduction and Related Work

Simultaneous Localization and Mapping (SLAM) systems such as [14, 15, 17, 7] and commercial systems such as HoloLens or Project Tango demonstrate that visual SLAM is a maturing technology. Given new measurements, visual SLAM systems build and update a map of an unknown environment while keeping track of the position of the camera within it. There are two predominant scenarios in which it is usually not possible for the system to estimate a correctly updated camera pose. In the first scenario, the camera moves significantly between consecutive frames. In such cases, generic pose optimization routines such as ICP or the minimization of reprojection error can get stuck in local minima due to the non-convex nature of their energy landscapes. In the second scenario, the user acquires a map of the environment and switches the reconstruction system off. When the SLAM system is turned on again, it is unable to recover the camera pose with respect to the previously acquired environment again for reasons of convergence. The task of camera relocalization is to tackle both scenarios by providing an estimate of the camera pose relative to an existing model, ideally from a single frame.

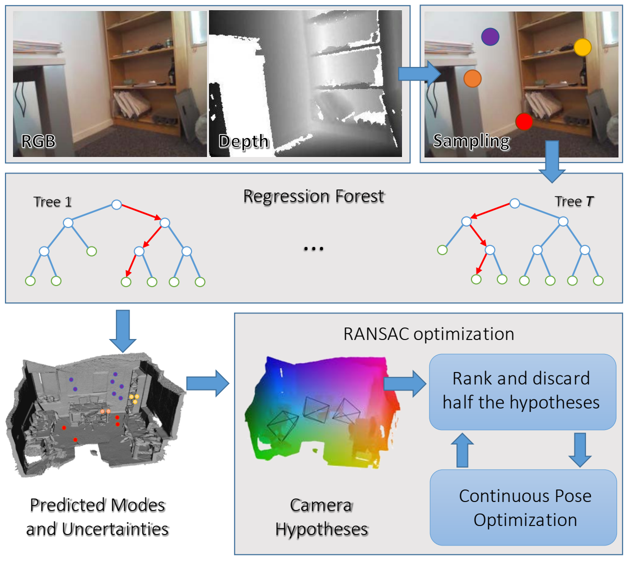
RGB
Depth
Sampling
Tree 1
Regression Forest
...
Tree *T*
RANSAC optimization
Rank and discard half the hypotheses
Continuous Pose Optimization
Predicted Modes and Uncertainties
Camera Hypotheses

Figure 1. **Camera Relocalization Pipeline.** As a new frame arrives (top left), pixels are sparsely sampled (top right) and passed down a regression forest (middle). The forest predicts a series of candidate locations in the scene as well as the uncertainty associated with each prediction (bottom left). Given these predictions, camera pose hypotheses are robustly sampled (bottom middle) and continuously optimized over iterations of RANSAC (bottom right).

Camera relocalization has mainly been studied and developed for SLAM applications; however, this technology could be applicable in a wide range of use cases. For instance, one can imagine scenarios where the user is in a large environment (shopping mall, exhibition, etc.) and queries the location of a destination such as 'Exit', 'Mona Lisa', or the name of a specific shop, and takes a snapshot of her immediate environment. Given this information, an on-site service could precisely localize the user and guide her to the desired destination.

Relocalization techniques largely belong to two classes. The first class is composed of image-based (also known as key-frame) approaches and the second of keypoint-based approaches. A few successful approaches also exist that do not belong to either of those two classes [5, 16].

Image-based approaches estimate an approximate camera pose by computing whole-image similarity measures against a set of images for which the camera position is known. These images will be referred to as key-frames. The challenges with image-based approaches are the on-line selection of key-frames for good spatial coverage and the selection of metrics to rank the similarities between images. Successful attempts at solving these challenges have been proposed in [9, 8, 12]. A key limitation of these methods is that the number of key-frames increases as the camera moves through space, which directly impacts the time required for identifying key-frames that are similar to the current image. Another major limitation is that they are bad at recovering when the query frame is too different from those in the database of stored key-frames. This is due to the sparsity of the sampling of the camera space. A recent work [8] approached the problem of sampling views for a denser coverage of the space of the camera poses by generating new views. This is an interesting way of tackling the problem, but is quite costly. Due to the aforementioned limitations, this first family of methods is better suited to systems which have strong constraints on the camera motion, and will not work well when the camera can move freely in large environments.

Keypoint-based approaches do not require a dense sampling of viewpoints and have the key advantage of being able to deal with novel viewpoints if enough keypoints can be matched. First, keypoints are detected in the images, and their corresponding descriptors and positions in world coordinates are stored. When the tracking of the camera is lost, keypoints and descriptors are computed in the current image and matched against the existing database. Given these matches, a robust pose can be inferred after optimization. The challenges with such approaches are to compute keypoints and corresponding descriptors on-line, then to choose which ones to store in order to control the growth of the database while ensuring good coverage of the scene, and finally to identify a robust matching mechanism. An inherent limitation of this family of methods lies in the sparsity of the keypoints, which strongly influences the inferred camera pose. The method presented in [19] solves the problems of sparsity and density using a different pipeline, but needs to train a model off-line. Publications presenting on-line keypoint-based relocalization methods include [6, 22, 18, 21].

We now describe the recent advances [19, 10] in solving this problem. The *scene coordinate regression forest* (SCoRe Forest) approach of [19] trains regression trees to predict the location of any pixel in the scene's world coordinate frame. Given these 2D-to-3D point estimates, camera hypotheses are sampled and refined using the Kabsch algorithm. This method has been shown to work well on a variety of scenes, but all the components of their pipeline assume isotropic and uni-modal distribution of the samples. For any given scene, it is very likely to find regions of the descriptor space that contain multiple clusters of points living in different parts of the scene. For instance, imagine that we use a descriptor that is composed of color gradient features, and that our samples are taken from close-up views of an untextured wall and an untextured table. In such cases, it is unsatisfying to only predict a point on the wall or a point on the table. Instead, we would like to predict both, together with the uncertainty associated with each prediction. Furthermore, clusters of points within the scene generally are extremely anisotropically distributed. The SCoRe Forest framework was recently extended in [10], which learned an ensemble of regression forests that are both relevant and diverse predictors. In contrast to [19], they used an explicit representation (based on a truncated signed distance function) of each scene to score hypotheses. Note that both methods neither model nor use uncertainty about their predictions, and hence use predictions as point estimates.

The main contributions of this work are (i) the extension of the state of the art on RGB-D camera relocalization by modeling and minimizing uncertainties for regression tree induction and predictions performed by the regression forest; and (ii) leveraging these uncertainties in order to provide for improved relocalization without using explicit models of the scenes.

2. Method Overview

The proposed camera relocalization approach consists of two major components: (i) a regression forest trained on RGB-D input data to predict anisotropic Gaussian mixtures of 3D scene coordinates; and (ii) a continuous pose optimization leveraging the anisotropic Gaussian mixtures predicted by the forest. In more detail:

- Given samples from the scene for which the relocalization task will be performed, a regression tree is trained using an objective function that minimizes the spatial variance of the scene coordinates. The distribution at the leaves is typically anisotropic and multimodal (see Fig. 2), and thus the leaves predict anisotropic Gaussian mixture models that have been fitted to those distributions.

- At test time, pixels are randomly sampled and passed down the regression forest. For each of these samples, the ensemble learner predicts a Gaussian mixture model that specifies a probability density function over that sample's location in the scene's 3D world coordinates. The predictions are then aggregated to generate robust camera hypotheses that are passed to a preemptive locally-optimized RANSAC. Each loop of the optimization ranks the camera hypotheses, discards the worst half, and optimizes the pose of all the remaining hypotheses by leveraging the predicted Gaussian mixtures. This process is repeated until only one hypothesis remains.

An overview of the complete relocalization pipeline is shown in Fig. 1.

3. Learning Uncertainties

We first describe the image features and the tree training objective, followed by our method for modeling uncertainty in the predictions made by the forest.

3.1. Image Features

Following [19], we use features based on pairwise pixel relationships [13, 20] as they have proven to be well-suited for the task of camera relocalization. Aside from their discriminative power, these simple features are also extremely fast to evaluate. The two types of features we use are the following 'Depth' and 'Depth-Adaptive RGB' ('DA-RGB') features:

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

$$f_{\Omega}^{\text{DA-RGB}} = G(p, c_1) - G\left(p + \frac{\delta}{D(p)}, c_2\right) \tag{2}$$

$D(p)$ is the depth at pixel p, $G(p, c)$ is the value of the c$^{\text{th}}$ color channel of pixel p and Ω is a vector of randomly sampled feature parameters. For the 'Depth' feature, the only random parameter is δ, which is a 2D offset in image space. The 'DA-RGB' feature has two extra random variables that correspond to the channels in which the intensity lookups are performed. Both 'Depth' and 'Depth-Adaptive RGB' features are translation invariant and largely scale invariant. In this work, we use a combination of both of these features and will refer to this as 'DA-RGB + D'.

3.2. Training a Regression Forest

A regression forest is a collection of regression trees. We use the standard greedy tree training algorithm to grow the trees. Each tree is trained with some randomness that comes from the random generation of pairs of feature indices ϕ and thresholds τ, and optionally also from bagging. We denote each pair of (ϕ,τ) as θ, and the set of candidate random parameters θ in node n as Θ_n. Each node n uses the randomly generated Θ_n to greedily optimize

$$\theta_n^* = \operatorname*{argmax}_{\theta \in \Theta_n} I_n \tag{3}$$

which is the set of parameters that will be used as the weak learner at node n. We use the classical information gain as the objective function:

$$I_n = E(\mathcal{S}_n) - \sum_{i \in \{\text{L},\text{R}\}} \frac{|\mathcal{S}_n^i|}{|\mathcal{S}_n|} E(\mathcal{S}_n^i) \tag{4}$$

where $E(\mathcal{S})$ is a measure of the entropy of the labels of the examples in set $\mathcal{S}$. Note that the left and right subsets $\mathcal{S}_n^i$ are implicitly conditioned on the candidate parameters θ.

3.3. Measuring Entropy

Given the above definition of information gain, we now need a definition of the entropy $E(\mathcal{S})$. We do this by fitting a model that approximates the distribution of labels (the scene coordinates) in set $\mathcal{S}$ and computing the entropy of that distribution. As motivation, the top row of Fig. 2 shows some empirical distributions of scene coordinates that reach particular tree nodes. Note how these distributions appear to be highly multi-modal, and that each model is often highly anisotropic.

There are several approaches to modeling the observed distributions, with varying levels of accuracy and efficiency. Perhaps the simplest approach, used in [19], is to fit an isotropic Gaussian to the labels in $\mathcal{S}$. While very fast, this approach provides a poor approximation and thus we believe a sub-optimal entropy measure for training the trees. The next approach we explored was to build a non-parametric model using mean shift, and scoring candidate splits based on the weighted sum of the determinant of the generated modes. The need to run mean shift for each candidate weak learner made the training extremely slow to the point where we could not feasibly train a deep enough tree to obtain reasonable results. Furthermore, mean shift typically assumes isotropic kernels which are unlikely to explain our anisotropic distributions. We thus investigated an approach that fits a Gaussian mixture model to the data. This proved somewhat faster than mean shift, and allowed us to fit anisotropic Gaussians. However, ultimately, we settled on an approach that uses a single full-covariance Gaussian model, with the entropy defined as

$$E(\mathcal{S}) = \frac{1}{2} \log((2\pi e)^d |\Lambda(\mathcal{S})|) \;, \tag{5}$$

with dimensionality $d = 3$, and Λ representing the full covariance of the labels in $\mathcal{S}$. This gave a good trade-off between training speed and the accuracy of the final predictor. A uni-modal Gaussian clearly cannot represent the multi-modal empirical distributions we observe, and so the next section describes how, after training the tree structure, we fit a multi-modal distribution at the leaf nodes.

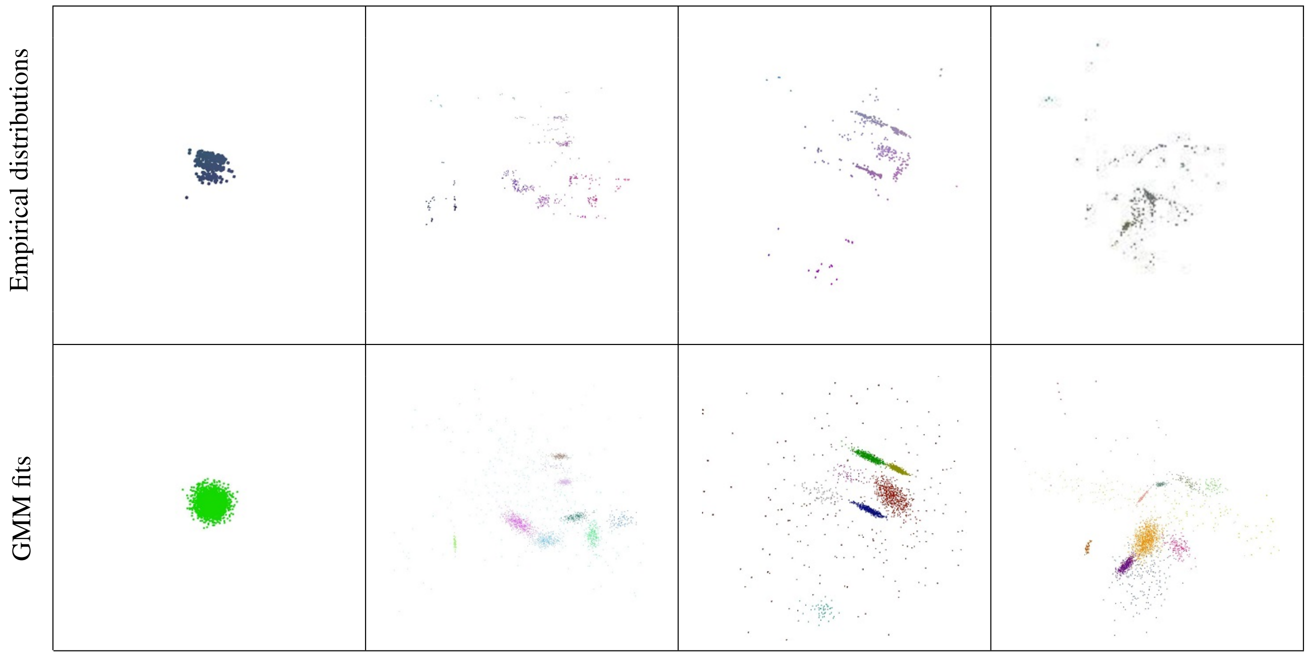

Figure 2. **Leaf models that represent uncertainty**. **Top row:** Examples of empirical distributions in different regression tree leaves. Colors indicate the 3D scene coordinates (i.e., RGB = XYZ). Note how the distribution is often multi-modal and anisotropic. **Bottom row:** Samples from the Gaussian mixtures fitted to the each of the above empirical distributions. Colors indicate mixture components. Note that all the images correspond to a 2D projection of 3D points.

3.4. Leaf Prediction Models

The previous section described the entropy criterion used to select good features θ when building the structure of the tree. This section now describes what we store at the leaf nodes, and thus the predictions that get made at test time when a particular leaf node is reached after descending a tree. Note that we do not need to (though of course could) use the model that we used in the previous section for computing entropies. Indeed in practice, for computational reasons, we ended up using a rather simplistic distribution when building the tree structure (a full covariance uni-modal Gaussian; see Sec. 3.3) but a more detailed model (see below) stored at leaf nodes.

Referring again to Fig. 2 (top row), we see examples of the empirical distribution of scene coordinates at a few randomly chosen (leaf) nodes. We aim to use prediction models that are compact but representative of these distributions. It is clear that these distributions are in general highly multi-modal and anisotropic. One might argue that if the tree was trained deeper, then these distributions would become increasingly more Gaussian. However, deeper trees are more expensive for both training and testing, and can be prone to over-fitting, which we confirmed in our experiments. Furthermore, regions of the appearance space are likely to have inherently multi-modal distributions in scene coordinate label space. For example, imagine a node which contains samples from a close-up view of both a white wall and a white floor. There are no appearance features that could reliably distinguish these two planar regions, and yet they can live in quite different regions of the scene. For such regions, a uni-modal distribution such as a Gaussian could never be a good representation. We must therefore be able to cope with multi-modality in our leaf models. One of our main contributions is to investigate how to represent and exploit the observed distributions, and whether doing so improves the quality of the final output.

In [19], mean shift was used to cluster the scene coordinate labels into a sparse set of point correspondences at each leaf. This approach does support multi-modality and is thus much better than fitting Gaussians in the leaves. However, the point correspondences are not able to accurately represent the uncertainty present in the scene coordinate labels. The energy function from [19] (Eq. 7) implicitly specifies a level of uncertainty in the predictions by the use of a robust error function ρ. However, this is not learned, and is isotropic.

Our models, in contrast, are learned and can be anisotropic. In particular, we explore the use of robustly-fit Gaussian mixtures as compact but representative models of leaf node uncertainty. Similar to [19], we run mean shift mode detection [3] to find the modes of the empirical distribution in each leaf. Given all the samples associated to each mode, a 3D Gaussian is estimated. Repeating this process for all the modes leads to a mixture of Gaussians as illustrated in Fig. 2 (bottom row). At test time, each tree in the regression forest predicts a mixture of Gaussians over the space of 3D scene coordinates y. Aggregating the predictions from multiple trees, we obtain a mixture of mixtures, itself a mixture:

$$l(y|\mathcal{M}) = \sum_{(m,\mu,\Sigma)\in\mathcal{M}} m\mathcal{N}(y;\mu,\Sigma). \tag{6}$$

For each mixture component, m is the (scalar) mixing coefficient, and μ and Σ are respectively the mean vector and full 3D covariance matrix. Because the decision tree evaluation directs each pixel i to a different set of leaf nodes, the forest thus predicts a different mixture $\mathcal{M}_i$ for each pixel. These predictions will be used in the optimization as described below.

4. Camera Pose Optimization

Similarly to [19], we employ a variant of preemptive locally-optimized RANSAC [2]. This variant starts by hypothesizing 1024 camera candidates *H* that are constrained to be rigid body transforms.We efficiently rank these initial hypotheses and take the top 64. The method then loops over the following steps until only one camera remains: (i) generate an additional batch of samples $\mathcal{I}$, (ii) use these samples to score the fit of each candidate, (iii) discard the worst half, and (iv) locally-optimize the remaining candidates using the current set of samples. For brevity, we describe the new aspects of our optimization algorithm below, and refer the reader to [2] for further details about the particular variant of RANSAC.

4.1. Hypothesis Generation

When per-pixel depth information is available, one can estimate the transformation parameters from the current view to the model with three local positions (current camera space) and their corresponding positions in the scene (world coordinates) using the Kabsch algorithm [11]. For each of the samples in the local world coordinates, [19] proposed to randomly sample one mode among the list of predicted modes. Although this is reasonable, such an approach ignores the fact that good camera samples are rigid body transformations between the points in camera space and their associated modes in scene coordinates. Given the proportion of 'bad' modes among the predictions, not checking for rigid body transforms leads to a significantly inferior proportion of 'good' camera hypotheses. We thus propose to check the properties of rigid body transformation, i.e. checking that the distance between modes and the angle between the lines going though them are preserved. Given that the covariance of the modes generally has a non-negligible determinant, those constraints are only loosely checked. Note that the proposed constraints can be incrementally verified during the sampling process, which reduces the number of samples required to form a plausible rigid body transformation matrix compared to simultaneously sampling three random correspondences at once.

4.2. Energy

The regression forest described above is able to predict a mixture model over the space of 3D scene coordinates for each pixel in the query image. We now define an energy function that evaluates the likelihood of the transformed observations (by the current hypothesis H) under these mixture distributions. Unlike [19], this allows exploitation of the learned uncertainty in the predictions of the forest, rather than assuming a fixed uncertainty model for all pixels. Let $\mathcal{I}$ be a batch of image pixels i at which the forest has been evaluated. Let x_i be an observed 3D depth pixel in camera space. Our energy for the hypothesis H is defined as

$$P(H) = \sum_{i\in\mathcal{I}} -\log l(Hx_i|\mathcal{M}_i) \tag{7}$$

i.e., the negative log of the relative likelihood of the transformed observations under the predicted mixture model $\mathcal{M}_i$. See Eq. 6 for the evaluation of l.

4.3. Early hypothesis filtering

The robustness of hypothesis sampling and the efficiency of the proposed optimization allows us to directly reduce the number of camera hypotheses from 1024 to 64 with only a small loss in relocalization accuracy[1]. The hypothesis pruning corresponds to reducing the number of subsequent operations by a factor of 8. This also means that the final estimate is the result of fewer steps of pose optimization, all of them performed with less samples. Aside from that initial rejection, the subsequent rejections consist of discarding the worst half of the candidates, as done in [19].

4.4. Optimization

Having computed the set of inliers, [2] suggests performing a local optimization step to refine the hypotheses. In [19] this step was performed by re-running the Kabsch algorithm using the updated set of inliers. Note that this method makes the assumption that all the inliers are subject to the same anisotropic uncertainty. Instead, we perform a continuous local optimization of an energy proxy (Eq. 9) that leverages the uncertainties captured during the training. One such problem is solved for each candidate camera pose H.

First, we randomly sample a batch of B pixels (B is set to 500 is our experiments). Then, we find for each pixel i

[1]That loss is of 1.2% on 'Heads', 0.5% on 'Chess' and 1.2% on 'Fire'.

the mixture component in $\mathcal{M}_i$ that currently best explains the transformed observation Hx_i:

$$(m_i^*, \mu_i^*, \Sigma_i^*) = \underset{(m,\mu,\Sigma)\in\mathcal{M}_i}{\operatorname{argmax}} \; m\mathcal{N}(Hx_i; \mu, \Sigma). \tag{8}$$

Note that this can be computed (almost) for free during the evaluation of the full energy (Eq. 7) in the RANSAC scoring stage described above. When the best mode is above some threshold on the distance between μ_i^* and Hx_i, it is not included in the optimization as it is unlikely to be a correct match for x_i. To be able to continuously optimize the proxy energy function over the 6 degrees of freedom of H, we project H from $\mathrm{SE}(3)$ to $\mathrm{se}(3)$ and obtain the corresponding 6D twist vector $\xi = \log(H)$ (see [1]). Now we can define the proxy energy function to be optimized:

$$P(\xi) = \sum_{i\in\mathcal{I}} ||\Sigma^{*-\frac{1}{2}}(\xi(x_i) - \mu_i^*)||, \tag{9}$$

where $\xi(x)$ represents x transformed by the twist ξ. This energy is the sum of the square root of the Mahalanobis distances between the transformed observations and the closest components chosen in Eq. 8. This energy is optimized by a Levenberg-Marquardt optimizer and the resulting $H = \exp(\xi)$ is used as the refined camera pose hypothesis in the next RANSAC iteration. Note that we also tried the sum of Mahalanobis distances, which provided inferior results in our experiments. Note also that, for simplicity, the mixture selected by Eq. 8 is held fixed in this optimization. However, this is effectively allowed to change across RANSAC iterations.

5. Results

Dataset. We use the 7-Scenes dataset from [19] to evaluate our contributions. This dataset consists of seven scenes recorded using a Kinect RGB-D camera at a resolution of 640x480. The creators of the dataset used an implementation of KinectFusion to generate the 'ground truth' camera pose for each frame.

Baselines. We compare the proposed method against the state of the art in RGB-D camera relocalization [19, 10] and a baseline that uses state-of-the-art feature matching techniques. For more details about the latter baseline, we refer the reader to Sec. 4.3 of [19].

Error metric. For comparison with [19, 10], we report accuracy as the proportion of test frames for which the translational error is below 5cm and the rotational error is below 5°. This metric is fairly strict and should allow any robust model-based tracker to resume. To further assess the robustness of our method, we also plot the probability of convergence of our method using camera hypotheses sampled to uniformly cover translational errors ranging between 0cm and 50cm and angular errors ranging from 0° to 50°.

Scene	**Baselines**			**Our**
	Sparse RGB	[19]	[10]	**method**
Chess	70.7%	92.6%	96%	**99.4%**
Fire	49.9%	82.9%	90%	**94.6%**
Heads	67.6%	49.4%	56%	**95.9%**
Office	36.6%	74.9%	92%	**97.0%**
Pumpkin	21.3%	73.7%	80%	**85.1%**
Kitchen	29.8%	71.8%	86%	**89.3%**
Stairs	9.2%	27.8%	55%	**63.4%**
Average	40.7%	67.6%	79.3%	**89.5%**

Table 1. **Main results.** Percentages denote the portion of frames satisfying the error metric. Our method very substantially improves upon the baselines on all the scenes. Note that our results were generated using the same features across all sequences, while we compare against the best results from [19, 10] that use different features for each sequence ('DA-RGB' or 'DA-RGB + D').

Main results. We present our main results and our comparison against the state of the art in Table 1. The proposed method yields major improvements over all baselines on all the test sequences. The decrease in relative error ranges from 18.7% on 'Stairs' to 90.7% on 'Heads', with a mean and median relative decrease of 50.3% and 46.0% respectively. As can be observed in Table 2, the source of the improvement is the use of the predicted anisotropic distributions in the proposed continuous pose optimization. Inferior results are obtained when the minimization of Eq. 9 is performed using isotropic distributions.

	'Chess'	**'Fire'**	**'Office'**
Isotropic Energy & Kabsch updates [19]	92.6%	82.9%	74.9%
Isotropic Energy & Continuous pose update	86.9%	77.1%	45.7%
Anisotropic Energy & Continuous pose update	**99.4%**	**94.6%**	**97.0%**

Table 2. **Source of the accuracy gain.** This table illustrates that the proposed continuous pose optimization has issues converging to good cameras when the trees predict isotropic Gaussians mixture models. Hence, the performance gains come from the pairing of anisotropic distributions with the proposed continuous pose optimization.

Constraints on hypothesis sampling. Table 3 shows the influence of checking the proposed constraints while sampling camera hypotheses. It can be observed that the proposed sampling of hypotheses provides for a non-negligible gain in accuracy over random sampling.

	Random Sampling	**Constrained sampling**
Isotropic Energy & Kabsch updates	75.0%	**76.5%**
Anisotropic Energy & Continuous pose update	91.8%	**95.9%**

Table 3. **Influence of the constraints added during the sampling of hypotheses on the 'Heads' sequence. First column:** Results obtained when no constraints are enforced during the sampling of hypotheses. **Second column:** Results obtained when constraints are enforced. **First row/Second row:** Observe that both constrained sampling and continuous optimization of anisotropic predictions are important for high accuracy.

Pose optimization. Tables 2 and 3 show a comparison of the results obtained when the camera hypotheses are optimized with the Kabsch algorithm or the proposed optimization using the same regression forest. It can be observed that in similar settings, our proposed optimization yields major improvements over running Kabsch, demonstrating the benefits of explicitly modeling and using the uncertainties in regression forests during the pose optimization process. Fig. 3 illustrates the probability of convergence for both the method described in [19] and our proposed approach. It can be observed that the method of [19] has a parameter space in which only a relatively small region has a high probability of convergence to cameras satisfying the error metric. For that method, the mean percentage of cameras satisfying the error metric after optimization is 4% and the median is 2%. By contrast, our proposed method is more robust to the initial position of the hypotheses with respect to the ground truth. Indeed, our method exhibits a mean and a median percentage of cameras satisfying the error metric of 48%. However, it is important to note that the method of [19] is more stable than the proposed method when the sampled camera directly satisfies the error metric (82.2% vs. 73.7%). In most cases, the method used by [19] robustifies hypotheses but does not allow them to evolve much. The top left image of Fig. 3 illustrates this behavior: in the region where the error metric is satisfied (5cm and $5°$), the percentage is high, but it greatly drops as soon as cameras are sampled outside that region.

Accurate relocalizations. Fig. 4 shows the accuracy of our predicted camera poses. When using the proposed optimization, 50% of the frames are relocalized within 0.45cm and $0.33°$ of their respective ground-truth. This is dramatically better than using an isotropic energy combined with Kabsch updates, for which the same relocalization rate is only attained at 2.3cm and $2.85°$.

Energy. At the end of each loop of the preemptive locally-optimized RANSAC, each candidate is assigned a score corresponding to its estimated degree of relevance. These candidates are then ranked and the bottom half is discarded. Given a set of cameras, it is then crucial to reliably distinguish between promising and less promising candidates. In a side-by-side comparison with the isotropic energy described in [19], the results in Table 4 show a major difference in favor of our proposed approach, confirming the benefits of leveraging the uncertainty of the predictions while computing the degree of relevance of each camera hypothesis.

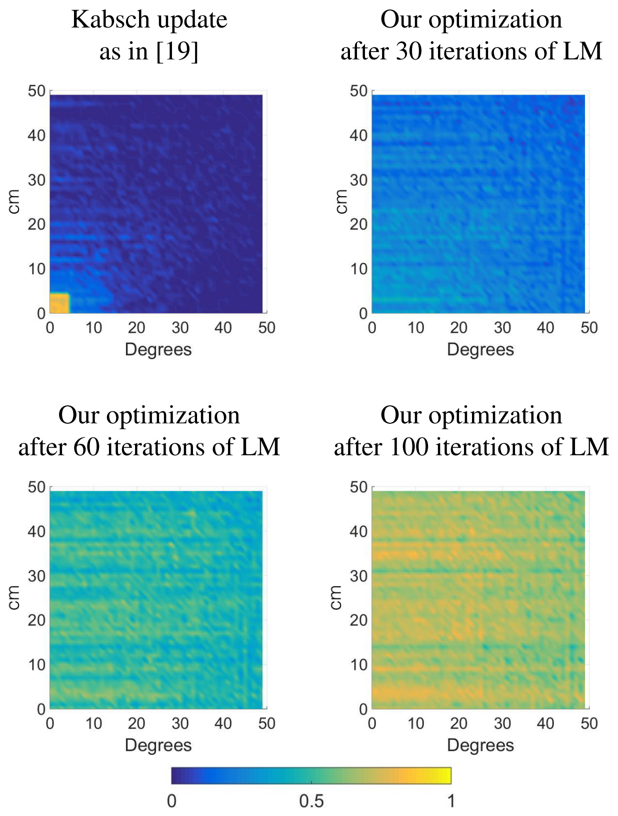

Figure 3. **Probability of convergence on 'Heads'. Top left:** Probability of convergence for the pose update based on Kabsch as described in [19]. **All but top left:** Probability of convergence for the proposed optimization at different iterations of LM. All results have been obtained by first randomly sampling 50 translation vectors per frame with errors against ground truth uniformly distributed between 0cm and 50cm; similarly for rotations with errors ranging from $0°$ to $50°$. These vectors are then combined into 2500 camera hypotheses which are *all* optimized over 10 rounds of RANSAC. For all the plots, the ordinate corresponds to the distance from the sampled cameras to the ground truth, the abscissa corresponds to their angular difference, and the color associated to each pair of angular and translational errors represents the percentage of the corresponding hypotheses that converged to a solution satisfying the error metric.

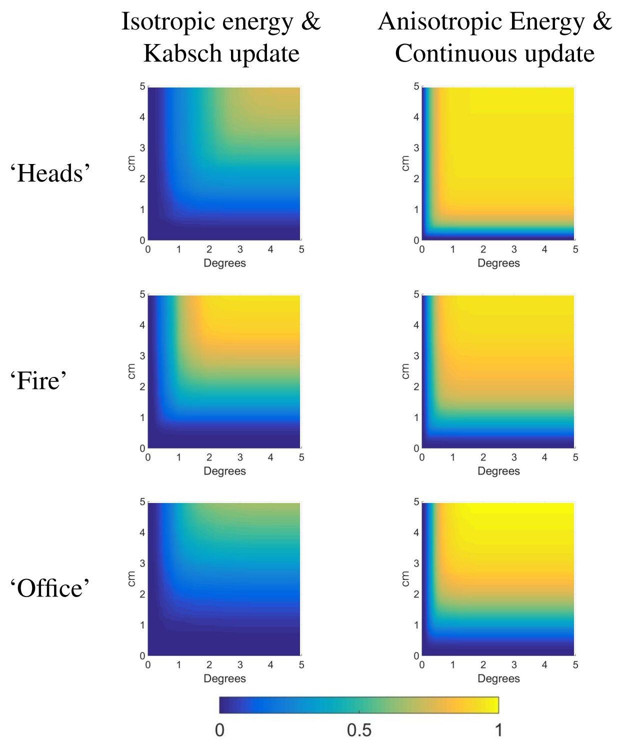

Figure 4. **Precision of the relocalization on 'Heads', 'Fire' and 'Office'.** For all the plots, each entry corresponds to the percentage of predicted cameras that have a translational and rotational error lower than the values specified by that entry. **Left column:** Results obtained when using an isotropic energy and the Kabsch algorithm to update camera hypotheses. **Right column:** Results obtained when using the proposed anisotropic energy and the proposed continuous optimization. Note that the same regression forest and the same camera hypotheses have been used to generate these results.

Isotropic energy [19]	Proposed energy
57.4%	**77.2%**

Table 4. **Energies for camera ranking on 'Heads'.** The isotropic energy refers to the energy of [19] and described in Eq. 7, and the proposed energy corresponds to Eq. 7. For each frame of the sequence, 1024 cameras are sampled, scored and ranked using both energies. The results correspond to the percentage of cameras that received the best score and passed the error metric (5cm and 5°).

6. Comparisons with the state of the art

Comparison with [19]. The proposed approach directly extends multiple aspects of [19]. A first notable difference lies in the fact that the training objective from [19] assumes an isotropic distribution of the samples where we propose to compute the entropy of an anisotropic estimator. A second major difference comes from the leaf models: in practice, the leaf model from [19] predicts a single point estimate where ours predicts multi-modal Gaussians. A third difference comes from the energy that is computed to rank the camera hypothesis. In [19], the authors propose to count inliers, where we propose to measure the relative likelihood of each predicted mode. A fourth difference is that we exploit some constraints about the problem to sample a higher ratio of relevant camera hypotheses. Finally, the most important difference comes from the fact that [19] optimizes the camera candidates with a least squares objective minimizing the distance between sampled points in camera coordinates and their corresponding inliers in world coordinates. Instead, we propose to use the multi-modal Gaussian predictions in the leaves and to leverage the predicted uncertainties during the optimization of the hypotheses.

Comparison with [10]. The main differences between the proposed work and the orthogonal contributions from [10] are twofold. First, they present a training method that iteratively trains an ensemble of forests by evaluating the error it makes on each sample, and uses these errors to reweight samples in a boosted fashion in order to train the next forest. The results are generated using 10 forests of 5 trees each, which is substantially more expensive to train and test than the approach proposed in this paper. Second, the camera hypotheses are scored using a truncated signed distance field which comes from an explicit reconstruction of the scene. In our case, we only use the implicit representation of the scene captured by the regression forest. Despite our simpler approach, Table 1 shows our considerably better accuracy.

7. Conclusion

We have presented a method to train and exploit uncertainty from regression forests for accurate camera relocalization. We were able to improve upon state of the art methods in multiple aspects. First, we have demonstrated that the proposed approach is better than the state of the art at relocalizing cameras. Second, we reported results illustrating that the proposed method has a *robust* convergence. Finally, we have shown that our predictions are considerably more *precise* than methods assuming isotropic distribution of the samples.

A very exciting opportunity and an interesting extension of this work would be RGB-only relocalization, as we believe that it would allow for a wide range of new applications, especially on mobile devices.

Acknowledgements

This research was partially supported by EPSRC, the Leverhulme Trust and the ERC grant ERC-2012-AdG 321162-HELIOS. We would like to thank Angela Dai and the anonymous reviewers for their valuable feedback.

References

[1] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Computer Vision and Pattern Recognition*. IEEE, 1998.

[2] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *Pattern Recognition*. Springer, 2003.

[3] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[4] A. Criminisi and J. Shotton. *Decision forests for computer vision and medical image analysis*. Springer, 2013.

[5] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[6] E. Eade and T. Drummond. Unified Loop Closing and Recovery for Real Time Monocular SLAM. In *British Machine Vision Conference*, 2008.

[7] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*. Springer, 2014.

[8] A. P. Gee and W. W. Mayol-Cuevas. 6D Relocalisation for RGBD Cameras Using Synthetic View Regression. In *British Machine Vision Conference*, 2012.

[9] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-time RGB-D camera relocalization. In *Mixed and Augmented Reality*. IEEE, 2013.

[10] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi. Multi-output learning for camera relocalization. In *Computer Vision and Pattern Recognition*. IEEE, 2014.

[11] 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.

[12] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision*. Springer, 2008.

[13] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006.

[14] R. A. Newcombe, A. J. 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 *Mixed and Augmented Reality*. IEEE, 2011.

[15] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *International Conference on Computer Vision*. IEEE, 2011.

[16] K. Ni, A. Kannan, A. Criminisi, and J. Winn. Epitomic location recognition. In *Computer Vision and Pattern Recognition*. IEEE, 2008.

[17] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *Transactions on Graphics*, 32(6):169, 2013.

[18] S. Se, D. G. Lowe, and J. J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21:364–375, 2005.

[19] 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 *Computer Vision and Pattern Recognition*. IEEE, 2013.

[20] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[21] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[22] B. Williams, G. Klein, and I. Reid. Automatic relocalization and loop closing for real-time monocular SLAM. *Pattern Analysis and Machine Intelligence*, 33(9), 2011.