Robust Neural Routing Through Space Partitions for Camera Relocalization in Dynamic Indoor Environments

Siyan Dong[1,2*] Qingnan Fan[3*] He Wang[3] Ji Shi[4] Li Yi[5]
Thomas Funkhouser[5] Baoquan Chen[4] Leonidas Guibas[3]
[1]Shandong University [2]AICFVE, Beijing Film Academy [3]Stanford University
[4]Peking University [5]Google Research
{siyandong.3,fqnchina,baoquan.chen}@gmail.com, hewang@stanford.edu, i@sjj118.com,
{ericyi,tfunkhouser}@google.com, guibas@cs.stanford.edu

Abstract

Localizing the camera in a known indoor environment is a key building block for scene mapping, robot navigation, AR, etc. Recent advances estimate the camera pose via optimization over the 2D/3D-3D correspondences established between the coordinates in 2D/3D camera space and 3D world space. Such a mapping is estimated with either a convolution neural network or a decision tree using only the static input image sequence, which makes these approaches vulnerable to dynamic indoor environments that are quite common yet challenging in the real world. To address the aforementioned issues, in this paper, we propose a novel outlier-aware neural tree which bridges the two worlds, deep learning and decision tree approaches. It builds on three important blocks: (a) a hierarchical space partition over the indoor scene to construct the decision tree; (b) a neural routing function, implemented as a deep classification network, employed for better 3D scene understanding; and (c) an outlier rejection module used to filter out dynamic points during the hierarchical routing process. Our proposed algorithm is evaluated on the RIO-10 benchmark developed for camera relocalization in dynamic indoor environments. It achieves robust neural routing through space partitions and outperforms the state-of-the-art approaches by around 30% on camera pose accuracy, while running comparably fast for evaluation.

1. Introduction

The task of camera relocalization is to estimate the 6-DoF (Degree of Freedom) camera pose from a test frame with respect to a known environment. It is of great importance for many computer vision and robotics applications, such as Simultaneously Localization and Mapping (SLAM), Augmented Reality (AR), and navigation, *etc*. One popular solution to camera relocalization is to make use of advanced hardware, *e.g.*, LIDAR sensors, WIFI, Bluetooth or GPS. However, these approaches may suffer from bad weather for outdoor environments, and instability or blocked signal for indoor environments. Another popular solution replaces the above hardware with a RGB/RGB-D sensor that feeds only visual observation for camera relocalization, also known as visual relocalization, which is the focus of this paper.

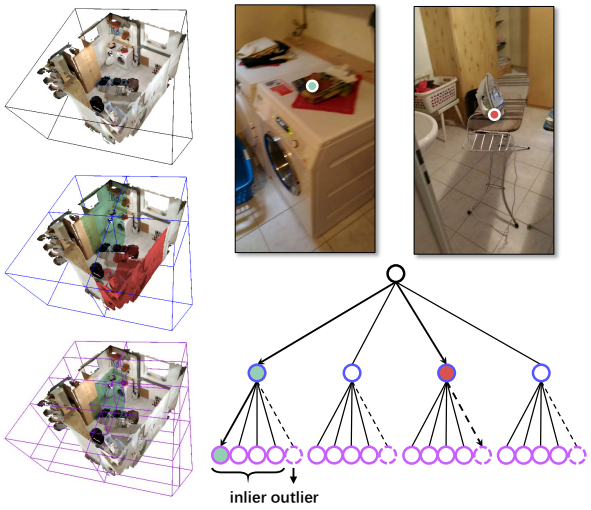

Figure 1. Demonstration of our algorithm. We build a hierarchical space partition over the entire scene environment to construct a 3-level 4-way neural tree. For the input static (green) or dynamic (red) points from a visual observation, our neural tree will route them into either inlier (solid line) or outlier (dashed line) categories. Only the points falling into the inlier category will be considered for camera pose estimation.

The problem of visual relocalization has been studied for decades, and recent advances [11, 32] have reached around 100% camera pose accuracy (5cm / 5°) on the popular in-

*Equal Contribution

door scene benchmarks 7-scenes [50] and 12-scenes [53]. One type of successful approach in this regard is designed based on decision trees, which was firstly introduced into the camera relocalization field in [50], with many follow-ups [35, 36, 37, 12, 11]. They build a binary regression forest that takes a query image point sampled from the visual observation as input, and routes it into one leaf node via a hierarchical splitting strategy, which is simply implemented as color/depth comparison within the neighbourhood of the query point. The leaf node fits a density distribution over the 3D world coordinates from the training scene. Hence, by evaluating the decision tree with a test image, a 2D/3D-3D correspondence can be easily established between the input sample and regressed world coordinate for camera pose optimization.

Although the aforementioned approaches are good at camera relocalization in static training environments, they tend to fail in dynamic test scenes, which are quite common yet challenging in real life. This is mainly due to the fact that the decision tree is constructed using only the static training image sequence so that, for any image point belonging to dynamic regions captured during evaluation, it is challenging to locate its correct correspondence in the leaf node. Recent studies [56] have demonstrated that the decision tree based approaches achieve around 28% camera pose accuracy (5cm 5°), which is also the best among all the competitors, in their proposed RIO-10 benchmark developed for dynamic indoor scenes. This performance is far from being comparable to the ones in static indoor scenes.

In order to tackle the challenges of camera relocalization in dynamic indoor environments, in this paper, we propose to learn an *outlier-aware neural tree* to help establish point correspondences for accurate camera pose estimation focusing only on the confident static regions of the environment. Our algorithm inherits the general framework of decision trees, but mainly differs in the following aspects in order to obtain better generalization ability in dynamic test scenes. (a) **Hierarchical space partition.** We perform an explicit hierarchical spatial partition of the 3D scene in the world space to construct the decision tree. Then each split node in the decision tree not only performs a hard data partition selection, but in fact one which also corresponds to a physically-meaningful 3D geometric region. (b) **Neural routing function.** Given an input point sampled from the 2D visual observation, the split node needs to determine which divided sub-region in the world space to go. Such a classification task needs more contextual understanding of the 3D environment. Therefore, we propose a neural routing function, implemented as a deep classification network, for learning the splitting strategy. (c) **Outlier rejection.** In order to deal with potential dynamic input points, we propose to consider these points as outliers and reject them during the hierarchical routing process in the decision tree. Specifically, the neural routing function learns to classify any input point from the dynamic region into the outlier category, stopping any further routing for that point. Once our proposed neural tree is fully trained, we follow the optimization and refinement steps in existing works [12, 11] to calculate the final pose.

We further train and test our proposed outlier-aware neural tree on the recent camera relocalization benchmark, RIO-10, which aims for dynamic indoor scenes. Experimental results demonstrate that our proposed algorithm outperforms the state-of-the-art localization approaches by at least 30% on camera pose accuracy. More analysis shows that our algorithm is robust to various types of scene changes and successfully rejects most dynamic input samples during neural routing.

2. Related Work

2.1. Camera Relocalization

Direct pose estimation. Approaches of this type aim for predicting the camera pose directly from the input image. One popular solution in this direction is image retrieval [21, 20, 22, 1, 26]. They approximate the camera pose of the query image by matching the most similar images in the dataset with known poses using low-level image features. Instead of matching features, PoseNet [29] and many follow-ups [28, 57, 9, 55, 46] propose to use a convolution neural network to directly regress the 6D camera pose from an input image. However, as mentioned in [46], the performance of direct pose regression using CNNs is more similar to the one using image retrieval, and still lags behind the 3D structure-based approaches detailed below.

Indirect pose estimation. Approaches of this type find correspondences between camera and world coordinate points, and calculate the camera pose through optimization with RANSAC [13]. One common direction is to leverage the 2D-3D correspondences between traditional keypoints in the observed image and 3D scene map [44, 34, 43, 45], followed by some recent works that deploy deep learning features [42, 41, 51, 16] to replace the extracted poor descriptors. Another common direction to seek correspondences is scene coordinate regression. Shotton *et al.* [50] proposes to regress the 3D points in the world space from a query image point by training a decision tree, followed by many variants [36, 37, 5, 54]. The other related works [4, 6, 8, 7, 33, 58, 32, 35] in this direction leverage the deep convolutional neural network to regress the world coordinates from an input image, with a following pose optimization step.

2.2. Decision Tree and Deep Learning.

Some recent efforts have been devoted to combining the two families of decision tree and deep learning techniques.

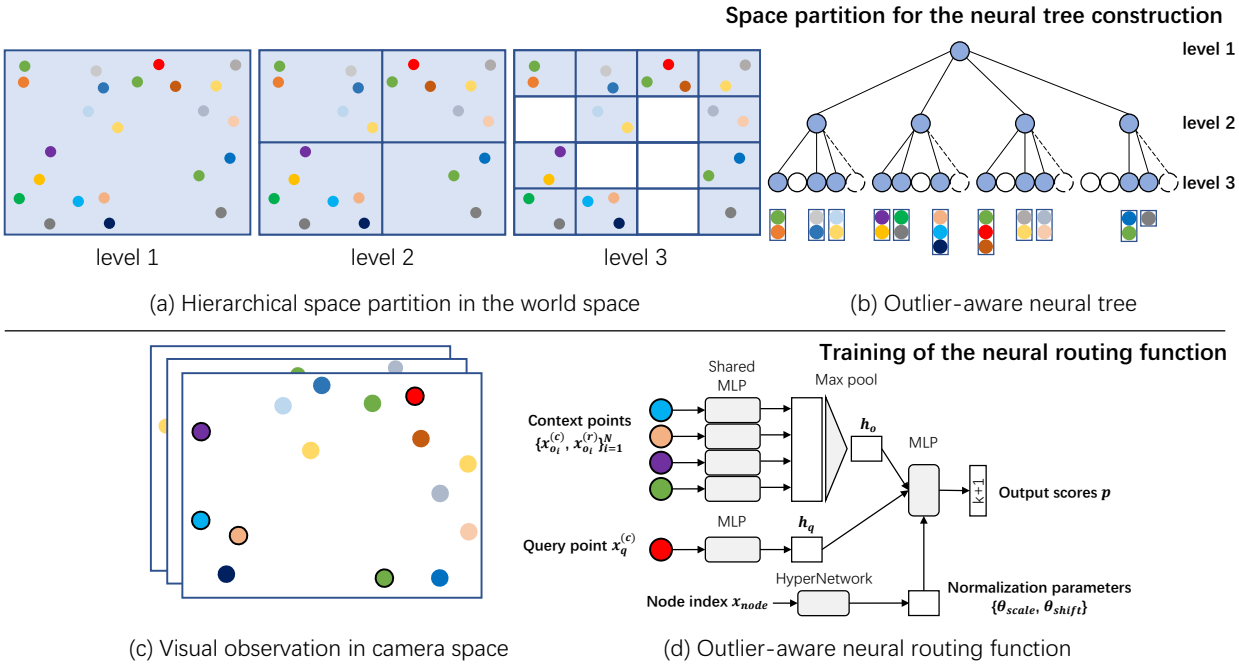

Figure 2. Illustration of our algorithm on the simple 2D case. Top: constructing a 3-level 4-way outlier-aware neural tree of a scene environment via hierarchical space partition. The dashed line and circle indicates the outlier category designed to reject the input dynamic points. Bottom: training an outlier-aware neural routing function for each split node in the neural tree.

The deep neural decision trees [31] propose a principled, joint and global optimization of split and leaf node parameters, and hence enable end-to-end differentiable training of the whole decision tree. Shen *et al.* [49] presents label distribution learning tree to enable all the decision trees in a forest to be learned jointly. The variants of deep neural decision trees have been successfully applied for the task of human age estimation [48] and monocular depth estimation [40]. Most of the aforementioned works formulate the last few fully connected layers in a classification neural network with the decision tree structure, and hence are significantly different from our algorithm.

3. NeuralRouting

3.1. Overview

The input to our algorithm is a training sequence of <RGB-D image, camera pose> and a test frame for camera relocalization. Our algorithm can be separated into two steps, *scene coordinate regression* and *camera pose estimation*. The former step is conducted by learning a neural tree that takes a query point along with its neighbor context as input, and regresses its scene coordinate in the world space to build a 3D-3D correspondence, based on which, the latter step estimates the camera pose via iterative optimization followed by an optional ICP refinement. The neural tree is constructed via performing an explicit space partition in the scene environment, and learns to reject the dynamic points as outliers during the hierarchical routing process. In this way, our algorithm learns to build the 3D-3D correspondence only within the confident static region for accurate camera pose optimization. We firstly revisit the decision tree and its adaptation for camera relocalizatoin in Sec. 3.2, and introduce our outlier-aware neural tree developed for relocalization in dynamic environments in Sec. 3.3. Finally, we describe the camera pose optimization and refinement details in Sec. 3.4.

3.2. Decision Tree for Camera Relocalization

Depending on whether the target is continuous or discrete, the decision tree can be used for either regression or classification tasks. A decision tree consists of a set of split nodes and leaf nodes. Each split node is assigned with a routing function, which learns the decision rules for the input sample partition, and each leaf node contains a probability density distribution fitted for the partitioned data. Given an input sample, inference starts from the root node and descends level-by-level until reaching the leaf node by evaluating the routing functions. A standard decision tree is binary, and employs greedy algorithms to learn the parameters at each split node to achieve locally-optimal hard data partition.

For the task of camera relocalization, the decision tree [11] is used to build the 3D knowledge of the known environment using the provided training sequence. Each split node takes a query point (RGB-D) sampled from the captured image as input and routes it into one child node. The leaf node fits a distribution over a set of 3D points in the world space

that are projected from the training images using the ground truth camera pose and calibration parameters. Therefore, when evaluating a test frame with a decision tree, by routing an input point from root node to leaf node, a 3D-3D point correspondence can be easily established and further used for camera pose optimization.

3.3. Outlier-aware Neural Tree

3.3.1 Hierarchical Space Partition for Decision Tree

For the existing decision trees [50, 35, 36, 37, 12, 11] developed for the camera relocalization problem, as there is no ground truth label for supervised training, the decision tree becomes ultimately a clustering strategy for the training data as observed in previous works [11, 10]. The decision rules at split nodes are learned via CLUS algorithm [3] that uses variance reduction as the split criterion and achieves local-optimal hard data partition. In this paper, we propose to perform a hierarchical space partition for the target scene environment to construct our decision tree. We represent the entire scene as the root node, and iteratively partition the scene until reaching predefined depth. Each split node is responsible for a geometric region in the scene, and partitions this region into sub-regions of equal size for its child nodes. Each leaf node contains a set of 3D world coordinates in its covered local geometric region. The space partition strategy is illustrated in Figure 2 and detailed below.

Given a 3D scene model constructed in world space using the training sequence of <RGB-D image, camera pose>, we build an m-way decision tree, where m is the zth power of 2. To perform a hierarchical space partition, we start from the root node which represents the entire scene environment. Then we compute the tight bounding box of the scene in the world coordinate system. We conduct iterative z cuts to divide the bounding box into m sub-boxes of equal volume size. In order to avoid the corner cases, such as long and narrow sub-boxes which may create challenges to learn the routing function, the decision rule is designed to encourage the divided bounding box to be similar to a cube. Specifically, to perform one cut on the bounding box of size (w, h, l), we find the longest edge over (w, h, l) and divide the box into two identical halves from the middle point of the edge. We iterate over such a process on the divided box until z cuts are achieved. We perform such a top-down data partition iteratively for the nodes among all the levels.

The decision tree constructed in this way features several properties: (a) our constructed tree structure relies on the explicit space partition over the **3D scene environment** in the world space, not on the data partition of the visual observations (RGB+D) in the 2D camera space, then it requires the routing function to have more 3D understanding ability; (b) **each split node is physically meaningful**, and covers a specific geometric region, which is spatially related to other father or child nodes; (c) **the decision rules are predefined** by the z-cut space partition strategy introduced in the above paragraph and stay constant for all the nodes, instead of optimized via greedy algorithms to behave differently for different nodes; (d) the decision tree is more tolerant to **an m-way tree implementation**, not limited to a standard binary decision tree; (e) **the constructed tree structure is scene-dependent**, and may contain empty nodes that cover no geometric regions in the scene. Overall, the constructed decision tree via hierarchical space partition is more flexible in structure and physically meaningful compared to a standard decision tree in previous works.

3.3.2 Outlier-aware Neural Routing Function

Given an input sample, the purpose of the routing function at each split node is to send it to one of the child nodes. In our problem setting, the input sample is from the observed 2D RGB-D frame, and its ground truth label is determined by its corresponding location in the 3D world space. For purpose of accurate prediction, the routing function needs to understand the 3D scene context from 2D observations. Therefore, inspired by many previous works regarding point cloud classification [38, 39] and point generation from 2D images [17], we take advantage of the point cloud processing framework to implement a neural routing function. We introduce the formulation of the input and network structure in detail below.

Input representation and sampling. The input to the neural routing function is a query point x_q that needs to be localized in the 3D world space, along with a set of context points $\{x_{o_i}\}_{i=1}^{N}$ in the neighbourhood of the query point. The input point is associated with color and depth, which are however both highly viewpoint dependent. In order to obtain better generalization ability in different viewpoints, given an input RGB-D frame, We augment its depth channel via transforming it into the rotation-invariant geometric feature following PPF-FoldNet [15]. To be specific, we firstly compute the oriented point cloud by projecting the full-frame depth into 3D camera space using camera calibration parameters, and calculating the pointwise normals in a 17-point neighbourhood [25]. Then we encode the query point and its context points into pair features,

$$\{(x_q^{(p)}, x_q^{(n)}, x_{o_1}^{(p)}, x_{o_1}^{(n)}), (x_q^{(p)}, x_q^{(n)}, x_{o_2}^{(p)}, x_{o_2}^{(n)}), \cdots, (x_q^{(p)}, x_q^{(n)}, x_{o_N}^{(p)}, x_{o_N}^{(n)})\} \in \mathbb{R}^{12\times N} \quad (1)$$

where p and n denotes the camera coordinate and normal, which form a 12-channel vector for each pair of oriented points $(x_q,\ x_{o_i})$. Each pair feature $(x_q^{(p)}, x_q^{(n)}, x_{o_i}^{(p)}, x_{o_i}^{(n)})$ is then transformed into the rotation-invariant geometric representation [15] that consists of three angles and one pair

distance,

$$r = \{\angle(x_q^{(n)}, x_q^{(p)} - x_{o_i}^{(p)}), \angle(x_{o_i}^{(n)}, x_q^{(p)} - x_{o_i}^{(p)}), \angle(x_q^{(n)}, x_{o_i}^{(n)}), \|x_q^{(p)} - x_{o_i}^{(p)}\|_2\} \in \mathbb{R}^4 \quad (2)$$

Overall, for each input context point, it consists of both color c and transformed rotation-invariant information r, represented as $\{x_{o_i}^{(c)}, x_{o_i}^{(r)}\} \in \mathbb{R}^7$. Since the rotation-invariant feature for all context points is computed in the local reference frame with query point as origin, we omit the geometric feature and only take the color information as input for query point $x_q^{(c)} \in \mathbb{R}^3$.

Given an input image, the query point for a split node is randomly sampled among the 2D image pixels whose projected 3D world coordinates belong to the split node. The context points are randomly sampled within the 3D neighbourhood ball of the query point. Ball query defines a radius, which is adaptively changed from level to level due to the varying size of covered 3D geometric region in our problem setting. In the implementation, we calculate the radius as the length of the longest edge of the covered bounding box.

Routing function design. The routing function consists of two parts, the *feature extraction* module and *classification* module. The *feature extraction* module leverages the point-wise multi-layer perception (MLP) to learn the features from both query point and context points inspired by the recent popular point cloud processing network PointNet [38], while the *classification* module combines the deep features from query point and context points to learn which child node the query point should be routed to.

As the query point and context points are different in input channel, point number and impact for the classification task, we use different network parameters to encode their feature, specifically,

$$h_q = f_{featQ}(x_q^{(c)}) \quad (3)$$

$$h_o = f_{featO}(\{x_{o_i}^{(c)}, x_{o_i}^{(r)}\}_{i=1}^{N}) \quad (4)$$

where f_{featQ} and f_{featO} are implemented with a 3-layer pointwise MLP (64-128-32/512), and extract the internal deep features ($h_q \in \mathbb{R}^{32}$, $h_o \in \mathbb{R}^{512}$) for query and context points respectively. f_{featO} is followed with a max pooling layer to extract the global context feature.

Then h_q and h_o are concatenated and inputted into the classification module,

$$p = f_{class}(h_q, h_o) \quad (5)$$

where f_{class} is also implemented as a three-layer MLP (2048-1024-k), and outputs the probability ($p \in \mathbb{R}^k$) for all the child nodes. Since the constructed tree structure is scene dependent as mentioned in Sec. 3.3.1, the number of child nodes k is adaptively changed from node to node and from scene to scene. As for supervision, we apply a cross entropy loss between the predicted probability p and the ground truth label y for supervision,

$$\mathcal{L} = -\sum_{i=1}^{k} \mathbb{1}\{y_i = i\} \log \frac{\exp(p_i)}{\sum_{i=1}^{k} \exp(p_i)} \quad (6)$$

where y_i is the label for the ith child node.

Outlier rejection. The aforementioned neural routing function is designed to route the input sample into one of the child nodes that are bound to 3D geometric regions. Given a query point belonging to dynamic regions in the test frame, the hierarchical routing functions will send it into one of the leaf nodes that contain the 3D world coordinates only from the static training scene, and it may establish an inaccurate 3D-3D correspondence for camera pose optimization. In order to solve this issue, we propose to reject the query points from dynamic regions as outlier, hence the established correspondence will be maintained in the confident static region.

In order to achieve this goal, we further improve the neural routing function to output the probability vector p of $k+1$ channels, where the additional channel refers to the outlier class. To generate the training samples for each split node from a given input image, the routing function considers any image pixel belonging to the current split node as *inlier input* query point, which should be routed into child nodes. As the dynamic points in test environments are highly unpredictable, irregular, and do not exist in the training data, we simply consider any image pixel not covered by the current split node as *outlier input* query point, which simulates the dynamic points and should be rejected without further routing. To train the routing function for each split node, the inlier and outlier input query points are sampled to be 3:1. Notice the outlier rejection strategy is incorporated into the neural routing function from the second level, since for the root node, all the image pixels belong to the inlier input.

3.3.3 HyperNetwork for the Routing Functions

In order to construct a t-level m-way tree, there are at most m^{j-1} neural routing functions at level j except for the bottom level that contains leaf nodes, and totally at most $\frac{m^{t-1}-1}{m-1}$ routing functions for the whole tree. It is both time-consuming and storage-inefficient to train so many deep networks. In order to decrease the training time and storage space for efficient deep learning, the previous works [24, 18] unify the learnable parameters among different convolution layers in a network, time steps in a RNN, or hyper-parameters in an image filter within a standalone network, mostly known as HyperNetwork. More recent work [19] further discovers that learning the normalization parameters

with the HyperNetwork has similar performance as learning the convolution parameters, while the former case is more storage and running time friendly due to much less learnable parameters in the normalization layer compared to convolution layer.

Inspired by these previous works, in this paper, we propose to leverage HyperNetwork to learn a single neural routing function for all the split nodes in the same level of a decision tree. Specifically, given the one-hot value x_{node} that indicates the split node index, we learn to predict the learnable scale θ_{scale} and shift θ_{shift} parameters in the normalization layer of the *classification* module in the neural routing function,

$$\theta_{scale}, \theta_{shift} = f_{hyper}(x_{node}) \tag{7}$$

where f_{hyper} refers to the HyperNetwork, and is implemented a three-layer MLP. The size of θ_{scale} and θ_{shift} depends on the channel number in the normalization layer. Then the normalization parameters in the *classification* module is replaced with the predicted ones from the HyperNetwork,

$$p = f_{class}(h_q, h_o; \theta_{scale}, \theta_{shift}) \tag{8}$$

Therefore, for a t-level tree, we only need to learn totally t neural routing functions.

3.4. Camera Pose Estimation

The core of our algorithm is a decision tree, which is the same as many previous camera relocalization works [12, 11]. Therefore, we inherit similar optimization and refinement steps following [11] for camera pose computation, which are introduced below. In order to generate the camera pose in $\mathbf{SE}(3)$, we firstly fit modes in the leaf nodes and then optimize the pose by leveraging the established 3D-3D correspondences. Each leaf node covers a set of 3D points (XYZ+RGB) in the world space projected from the 2D image pixels captured in the training sequence. Following [54], we detect the modes of the empirical distribution in each leaf node via mean shift [14], and then construct a Gaussian Mixture Model (GMM) via iteratively estimating a 3D Gaussian distribution for each mode. After mode fitting of the leaf nodes, we leverage the preemptive locally-optimized RANSAC [13] for camera pose optimization. We start by generating 1024 pose hypotheses, each of which is computed by applying the Kabsch algorithm [27] on three randomly sampled 3D-3D point correspondences that relate the camera and world space. Given an observed point in camera space, its corresponding world coordinate is sampled from one random mode in the fitted GMM of the routed leaf node. We filter out the hypotheses that do not conform to the rigid body transformations following [12], and regenerate the alternatives until they satisfy the above requirement. The final camera pose is selected by iteratively re-evaluate and re-rank the hypotheses using the Mahalanobis distance, and discard the worse half until only one pose hypothesis is left.

Multi-leaves. Given an input query point, the aforementioned pose optimization process fits modes only from the routed leaf node, which is common for the existing decision tree implementations as their routing function performs hard data partition and hence the input point can only be routed into a single leaf node. In contrast, the proposed neural routing function performs a "soft" data partition with predicted probability p, hence the input point can be "routed" to all the leaf nodes with different accumulated probabilities through probability multiplication of all routed split nodes. Motivated by the above observation, to achieve more robust pose optimization, we fit the mode by combining the world coordinates from multiple routed leaf nodes with highest accumulated probabilities, instead of a single leaf node. In the implementation, we use four leaf nodes, which works the best experimentally, for mode fitting.

Pose refinement. Last but not least, we follow [11] to incorporate our camera relocalizer into a 3D reconstruction pipeline for further camera pose refinement, which mainly consists of ICP [2] and model-based hypothesis ranking.

4. Experiments

4.1. Implementation Details

Tree structure. For all the experiments in this paper, we implement the 5-level 16-way tree for scene partition, thus a perfect tree structure consists of 4369 nodes in this case. During our implementation, according to the specific scene geometry, such a tree contains about 2000 to 3000 valid nodes.

Training details. The neural routing functions are implemented in PyTorch. Benefited from the design of HyperNetwork, we only train 5 neural routing functions. Each routing function is trained for 60 epochs with a batch size of 256. The network weights are optimized with Adam [30] whose initial learning rate is 0.001 and betas are (0.9,0.999). The initial learning rate is halved every 20 epochs until the end. The number of context points is set as 600 all the time.

4.2. Dataset

We test our proposed algorithm on two camera relocalization benchmarks, RIO-10 [56] and 7-scenes [50], which are developed for dynamic and static indoor scenes respectively. The RIO-10 dataset includes 10 real indoor environments, each of which is scanned several times over different time periods, and demonstrates the common geometric and illumination changes in dynamic environments. This dataset is separated into training/validation/test split, while the test results should be obtained via submission to their online benchmark. The 7-scenes dataset contains only training and test set, and is the most popular camera relocalization

Method	**Score** ↑	**DCRE**(0.05) ↑	**DCRE**(0.15) ↑	**Pose**$(0.05\mathbf{m}, 5^\circ)$ ↑	**Outlier**(0.5) ↓	**N/A**
HFNet [41]	0.373	0.064	0.103	0.018	0.360	0.000
HF-Net Trained [41]	0.789	0.192	0.300	0.073	0.403	0.000
NetVLAD [1]	0.575	0.007	0.137	0.000	0.431	0.000
DenseVLAD [52]	0.507	0.008	0.136	0.000	0.501	0.006
Active Search [45]	1.166	0.185	0.250	0.070	0.019	0.690
Grove [12]	1.240	0.342	0.392	0.230	0.102	0.452
Grove V2 [11]	1.162	0.416	0.488	0.274	0.254	0.162
D2Net [16]	1.247	0.392	0.521	0.155	0.144	0.014
NeuralRouting (Ours)	1.441	0.538	0.615	0.358	0.097	0.227

Table 1. Comparison on the test split of the RIO-10 benchmark w.r.t. the average score (1 + DCRE (0.05) - Outlier (0.5)), DCRE errors, camera pose accuracy and outlier ratio. *N/A* denotes invalid/missing predictions. The red and blue numbers rank the first and second for each metric.

Pose$(0.05\mathbf{m}, 5^\circ)$ ↑	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Average
Shotton et al. [50]	92.60%	82.90%	49.40%	74.90%	73.70%	71.80%	27.80%	67.60%
Guzman-Rivera et al. [23]	96.00%	90.00%	56.00%	92.00%	80.00%	86.00%	55.00%	79.30%
Valentin et al. [54]	99.40%	94.60%	95.90%	97.00%	85.10%	89.30%	63.40%	89.50%
Brachmann et al. [5]	99.60%	94.00%	89.30%	93.40%	77.60%	91.10%	71.70%	88.10%
Schmidt et al. [47]	97.75%	96.55%	99.80%	97.20%	81.40%	93.40%	77.70%	92.00%
Grove [12]	99.40%	99.00%	100.00%	98.20%	91.20%	87.00%	35.00%	87.10%
Grove V2 [11]	99.95%	99.70%	100.00%	99.48%	90.85%	90.68%	94.20%	96.41%
NeuralRouting (Ours)	99.85%	100.00%	100.00%	99.80%	88.80%	90.96%	84.20%	94.80%

Table 2. Comparison on the 7-scenes dataset w.r.t. the camera pose accuracy. The red and blue numbers rank the first and second for each scene.

	Pose$(0.05\mathbf{m}, 5^\circ)$ ↑
Ours w/o outlier labels	25.14%
Ours w/o multi-leaves	25.80%
5-level 8-way Tree	24.60%
3-level 16-way Tree	16.75%
4-level 16-way Tree	25.31%
Ours (5-level 16-way Tree)	27.05%
Ours w. pose refinement	31.99%

Table 3. Ablation study on the validation set (10 scenes) of the RIO-10 benchmark. Ours is the full pipeline of our algorithm.

benchmark for the static indoor environments in the past.

4.3. Evaluation Metrics

In order to evaluate the quality of estimated camera pose, we adopt the common camera pose accuracy **Pose**$(0.05\mathbf{m}, 5^\circ)$, which is computed as the proportion of test frames whose translation error is within 5 centimeters and angular error is within 5 degrees. In the RIO-10 benchmark [56], we further adopt their proposed new metric **DCRE**, short for Dense Correspondence Re-Projection Error, which is computed as the average magnitude of the 2D correspondence displacement normalized by the image diagonal. The displacement is calculated between 2D projections of the underlying 3D model using the ground truth and predicted camera poses. DCRE depicts an error that correlates with the visual perception, not only with the absolute camera pose. Then **DCRE**(0.05) and **DCRE**(0.15) are the percentage of test frames whose DCRE is within 0.05 or 0.15, while **Outlier**(0.5) describes the opposite case, which is the percentage of test frames whose DCRE is above 0.5.

4.4. Numerical Results

We compare our algorithm with all the other approaches on the test split of the RIO-10 dataset, shown in Table 1. Among all the metrics that evaluate the quality of camera pose estimations, our algorithm ranks the first except for **Outlier**(0.5), where our performance is the second best. Regarding the camera pose accuracy **Pose**$(0.05\mathbf{m}, 5^\circ)$, which is more common and directly measures the pose quality, our result (0.358) surpasses the state-of-the-art approaches (0.274) significantly by about 30%. It demonstrates the effectiveness and robustness of our proposed outlier-aware neural tree on the dynamic indoor environments.

We further test our algorithm on the popular camera relocalization benchmark 7-scenes for static indoor scenes, shown in Table 2. Our algorithm ranks the second place on the averaged camera pose accuracy among all the existing approaches, and lags behind the best performance within a

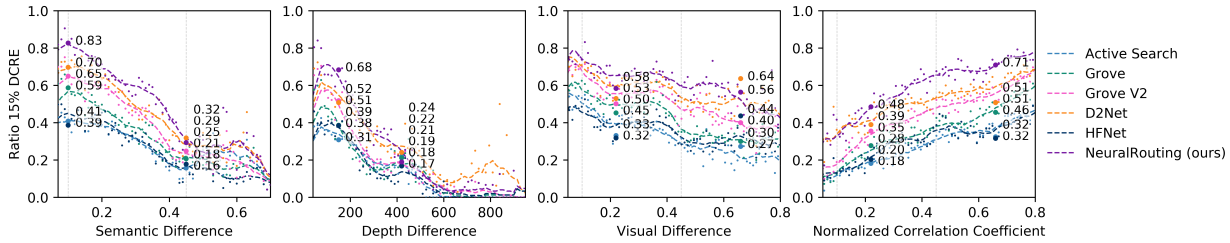

Figure 3. The charts show the performance (**DCRE**(**0.15**)) of compared approaches with respect to semantic (semantic difference), geometric (depth difference) and visual change (NSSD, Normalized Correlation Coefficient) as introduced in RIO-10 dataset [56].

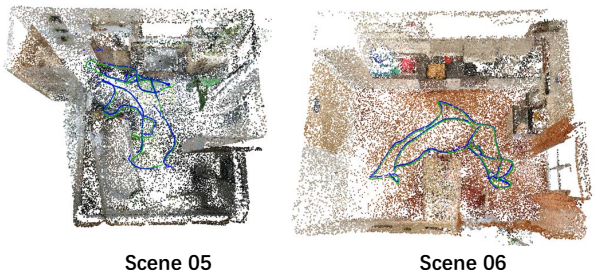

Figure 4. Ground truth (green) and predicted (blue) camera pose trajectory on the validation set of RIO-10 dataset.

very small gap. It further shows the excellent generalization ability of our algorithm on static scenes, though it is developed for dynamic environments. Note the baseline results in RIO-10 and 7-scenes datasets are from the official online benchmark and the recent SOTA relocalization paper [11], respectively.

We test the running time of our algorithm on GPU. For a single image, the camera pose optimization and refinement take around 100 ms and 150 ms separately, similar to the previous decision tree based approach [11]. The neural routing runs for 480 ms, while its light version without considering multiple leaves during routing takes only 100 ms yet achieves similar camera pose accuracy as verified in Table 3.

4.5. Ablation Study

To justify the effectiveness of our algorithm design, we conduct an ablation study as shown in Table 3, which is evaluated on the validation set of RIO-10 dataset. Space partition is important for our neural tree construction, hence we firstly test different strategies to partition the scene by varying the hyper-parameters t and m in the t-level m-way tree. We observe that as the number of levels t decreases, the camera pose accuracy also degrades, this is mainly due to the increased box size in leaf node, which creates difficulty in fitting a good distribution and sampling effective world coordinates. Notice the leaf nodes in 4-level 16-way tree and 5-level 8-way tree have the same box size, while 16-way tree is better in camera pose accuracy. This is mainly because the 4-level tree has fewer routing functions to be trained, and hence accumulates less error from the deep network during hierarchical routing. Finally we validate the design of outlier classification and multi-leaves in camera pose optimization by removing them from the entire framework, and observe worse pose accuracy as expected.

4.6. Analysis

Pose trajectory. We visualize the pose trajectory of both our estimations and ground truth on two scenes in the validation split of RIO-10 dataset in Figure 4. We observe a significant overlap between the two trajectories, which verifies the effectiveness of our algorithm in dynamic indoor environments.

Performance against various scene changes. To discover how our algorithm is robust to scene changes compared to other approaches, we visualize the overall performance of each method with images of increasing visual, geometric and semantic change as defined in RIO-10 dataset, in Figure 3. We are glad to see that our plotted curve is almost the best among all the different types of scene changes. It further verifies our algorithm for camera relocalization in dynamic indoor scenes.

5. Conclusion

In this paper, we propose a novel outlier-aware neural tree to achieve accurate camera relocalization in dynamic indoor environments. Our core idea is to construct a decision tree via hierarchical space partition of the scene environment, and learn a neural routing function to reject the dynamic input points during the level-wise routing process. Extending our work to only the RGB input and generalization to novel environments are more realistic yet challenging settings, which are treated as valuable future directions to explore.

Acknowledgements

This work was supported in part by National Key R&D Program of China (2019YFF0302902), National Science Foundation of China General Program grant No. 61772317, NSF grant IIS-1763268, a grant from the Samsung GRO program, a Vannevar Bush Faculty Fellowship, and a gift from Amazon AWS ML program.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[2] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[3] Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[4] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017.

[5] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3364–3372, 2016.

[6] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662, 2018.

[7] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7525–7534, 2019.

[8] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4322–4331, 2019.

[9] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2018.

[10] Lauriane Castin and Benoit Frénay. clustering with decision trees: divisive and agglomerative approach. In *ESANN*, 2018.

[11] Tommaso Cavallari, Stuart Golodetz, Nicholas Lord, Julien Valentin, Victor Prisacariu, Luigi Di Stefano, and Philip HS Torr. Real-time rgb-d camera pose estimation in novel scenes using a relocalisation cascade. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[12] Tommaso Cavallari, Stuart Golodetz, Nicholas A Lord, Julien Valentin, Luigi Di Stefano, and Philip HS Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4457–4466, 2017.

[13] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[14] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[15] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018.

[16] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[17] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[18] Qingnan Fan, Dongdong Chen, Lu Yuan, Gang Hua, Nenghai Yu, and Baoquan Chen. Decouple learning for parameterized image operators. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 442–458, 2018.

[19] Qingnan Fan, Dongdong Chen, Lu Yuan, Gang Hua, Nenghai Yu, and Baoquan Chen. A general decoupled learning framework for parameterized image operators. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[20] Dorian Galvez-Lopez and Juan D Tardos. Real-time loop detection with bags of binary words. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 51–58. IEEE, 2011.

[21] Andrew P Gee and Walterio W Mayol-Cuevas. 6d relocalisation for rgbd cameras using synthetic view regression. In *BMVC*, volume 1, page 2, 2012.

[22] Ben Glocker, Jamie Shotton, Antonio Criminisi, and Shahram Izadi. Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding. *IEEE transactions on visualization and computer graphics*, 21(5):571–583, 2014.

[23] Abner Guzman-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew Fitzgibbon, and Shahram Izadi. Multi-output learning for camera relocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1114–1121, 2014.

[24] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *International Conference on Learning Representations*, 2017.

[25] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, 1992.

[26] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010.

[27] Wolfgang 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.

[28] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5974–5983, 2017.

[29] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[32] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[33] Xiaotian Li, Juha Ylioinas, and Juho Kannala. Full-frame scene coordinate regression for image-based localization. In *In Robotics: Science and Systems (RSS)*, 2018.

[34] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1043–1050. IEEE, 2012.

[35] Daniela Massiceti, Alexander Krull, Eric Brachmann, Carsten Rother, and Philip HS Torr. Random forests versus neural networks—what's best for camera localization? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5118–5125. IEEE, 2017.

[36] Lili Meng, Jianhui Chen, Frederick Tung, James J Little, Julien Valentin, and Clarence W de Silva. Backtracking regression forests for accurate camera relocalization. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6886–6893. IEEE, 2017.

[37] Lili Meng, Frederick Tung, James J Little, Julien Valentin, and Clarence W de Silva. Exploiting points and lines in regression forests for rgb-d camera relocalization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6827–6834. IEEE, 2018.

[38] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[39] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[40] Anirban Roy and Sinisa Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5506–5514, 2016.

[41] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019.

[42] Paul-Edouard Sarlin, Frédéric Debraine, Marcin Dymczyk, Roland Siegwart, and Cesar Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. In *Conference on Robot Learning*, pages 456–465. PMLR, 2018.

[43] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale location recognition and the geometric burstiness problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1582–1590, 2016.

[44] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *2011 International Conference on Computer Vision*, pages 667–674. IEEE, 2011.

[45] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1744–1756, 2016.

[46] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3302–3312, 2019.

[47] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, 2(2):420–427, 2016.

[48] Wei Shen, Yilu Guo, Yan Wang, Kai Zhao, Bo Wang, and Alan L Yuille. Deep regression forests for age estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2304–2313, 2018.

[49] Wei Shen, Kai Zhao, Yilu Guo, and Alan L Yuille. Label distribution learning forests. In *Advances in neural information processing systems*, pages 834–843, 2017.

[50] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013.

[51] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018.

[52] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1808–1817, 2015.

[53] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 323–332. IEEE, 2016.

[54] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. Exploiting

uncertainty in regression forests for accurate camera relocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4400–4408, 2015.

[55] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 627–637, 2017.

[56] Johanna Wald, Torsten Sattler, Stuart Golodetz, Tommaso Cavallari, and Federico Tombari. Beyond controlled environments: 3d camera re-localization in changing indoor scenes. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[57] Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. Atloc: Attention guided camera localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[58] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. Sanet: Scene agnostic network for camera localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 42–51, 2019.