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 environment. 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.

The problem of visual relocalization has been studied for decades, and recent advances [11, 28] have reached around 100% camera pose accuracy (5cm / 5°) on the popular in-
door scene benchmarks 7-scenes [47] and 12-scenes [50]. One type of successful approach in this regard is designed based on decision trees, which was firstly introduced into the camera relocalization field in [47], with many follow-ups [31, 32, 33, 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 [53] 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 outliers-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 outliers-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 reject most dynamic input samples during neural routing.

2. Related Work

2.1. Camera Relocalization

**Straight-to-pose.** This type of approaches aim for predicting the camera pose directly from the input image. One popular solution in this direction is image retrieval. They approximate the camera pose of the query image by matching the most similar images in the dataset with known pose using low-level image features [41, 30, 40, 42]. However, these methods may fail in textureless image regions due to extracted poor descriptors. Recently, deep learning features [1, 39, 38, 48] emerge for more robust and efficient relocalization. While these learning works also deal with scene changes, they concentrate on mostly large-scale urban environments. The focus of our work lies in dynamic indoor environments. Instead of matching features, PoseNet [25] and many follow-ups [24, 54, 9, 52, 43] learn a convolution neural network to directly regress camera pose and is more efficient for realistic settings.

**Correspondence-based approaches.** [47] proposes to regress the 3D points in the world space from a query image point by training a decision tree, followed by many variants [31, 32, 33, 5, 51]. Once 2D/3D-3D point correspondence is established by evaluating a decision tree, camera pose can be directly optimized with RANSAC [13]. [20] learns to generate “marginally relevant” predictions that are effective in dealing with uncertainty stemming from ambiguities and multi-modality in the data. [12, 11] demonstrate how to adapt a pre-trained decision tree to a new scene on-the-fly by only retraining the leaf nodes, and achieve the best performance on the public camera relocalization benchmark, 7-scenes, from an RGB-D input so far. The other related works [4, 6, 8, 7, 29, 55, 28] in this direction leverage the deep convolutional neural network to regress the world coordinates from a full-frame image, with a differentiable RANSAC for end-to-end pose estimation.
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. The deep neural decision trees [27] propose a principled, joint and global optimization of split and leaf node parameters, and hence it enable end-to-end differentiable training of the whole decision tree. [46] 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 [45] and monocular depth estimation [37]. 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. Approach

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 relocalization 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, 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, it 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 environ-
ment 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 [47, 31, 32, 33, 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 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 [35, 36] 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 $\{x_{o_i}\}_{i=1}^{N_{12}}$ 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 [22]. Then we encode the query point and its context points into pairs features,

$$\{(x^{(p)}_q, x^{(n)}_q, x_{o_1}^{(p)}, x_{o_1}^{(n)}), (x^{(p)}_q, x^{(n)}_q, x_{o_2}^{(p)}, x_{o_2}^{(n)}), \ldots, (x^{(p)}_q, x^{(n)}_q, x_{o_N}^{(p)}, x_{o_N}^{(n)})\} \in \mathbb{R}^{12 \times N}$$

(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 $(x^{(p)}_q, x^{(n)}_q, x_{o_i}^{(p)}, x_{o_i}^{(n)})$ is then transformed into the rotation-invariant geometric
representation [15] that consists of three angles and one pair distance,
\[
\left\{ \angle(x_q^{(n)}, x_q^{(p)}) - x_o^{(p)}), \angle(x_o^{(n)}, x_q^{(p)}) - x_o^{(p)}), \\
\angle(x_q^{(n)}, x_o^{(n)}), \|x_q^{(p)} - x_o^{(p)}\|_2 \right\} \in \mathbb{R}^4
\tag{2}
\]

Overall, for each input context point, it consists of both color \(c\) and transformed rotation-invariant information \(r\), represented as \(\{x_o^{(c)}, x_o^{(r)}\} \in \mathbb{R}^7\). 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 [35], 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,
\[
h_q = f_{featQ}(x_q^{(c)}) \tag{3}
\]
\[
h_o = f_{featO}([x_o^{(c)}, x_o^{(r)}]_i=1^N) \tag{4}
\]
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,
\[
p = f_{class}(h_q, h_o) \tag{5}
\]
where \(f_{class}\) is also implemented as a three-layer MLP (512-64-\(k\)), 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,
\[
\mathcal{L} = - \sum_{i=1}^k \{ y_i = i \} \log \frac{\exp(p_i)}{\sum_{i=1}^k \exp(p_i)} \tag{6}
\]
where \(y_i\) is the label for the \(i\)th child node.

Outlier rejection. The aforementioned neural routing function is designed to route the input sample into one of the child nodes that are binded 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 and irregular, we simply consider any image pixel not covered by the current split node as outlier input query point, which simulate 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 \(m^{t-1}\) neural routing functions at level \(j\) except for the bottom level that contains leaf nodes, and totally at most \(\frac{m^{t-1}}{m-1}\) 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 [21, 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 $\theta_{scale}$ and shift $\theta_{shift}$ parameters in the normalization layer of the classification module in the neural routing function,

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

(7)

where $f_{\text{hyper}}$ refers to the HyperNetwork, and is implemented a three-layer MLP. The size of $\theta_{scale}$ and $\theta_{shift}$ 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,

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

(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 $\text{SE}(3)$, 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. To explore the multi-modality and uncertainty in scene coordinates, following [51], 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, which will be later used to score the camera pose hypotheses.

After mode fitting of the leaf nodes, we leverage the preemptive locally-optimized RANSAC [13] for camera pose optimization following [12]. We start by generating 1024 pose hypotheses, each of which is computed by applying the Kabsch algorithm [23] 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. For purpose of efficiency of the pose optimization process, we reduce the number of hypotheses from 1024 to 64 with the lowest energy. The score is calculated as the Mahalanobis distance between the transformed observations in the world space and its closest mode in GMM, as depicted in [11]. The final camera pose is selected by iteratively re-evaluate and re-rank the hypotheses using the above computed score, 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. 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 [26] 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 [53] and 7-scenes [47], which are developed for dynamic and static indoor scenes respectively. The RIO-10 dataset includes 10 real indoor environments,
### Table 1. Comparison on the test split of the RIO-10 benchmark w.r.t. the average score $(1 + \text{DCRE}(0.05) - \text{Outlier}(0.5))$, 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.

| Method                  | Score ↑ | DCRE(0.05) ↑ | DCRE(0.15) ↑ | Pose(0.05m, 5°) ↑ | Outlier(0.5) ↓ | N/A |
|-------------------------|---------|--------------|--------------|------------------|----------------|-----|
| HFNet [38]              | 0.373   | 0.064        | 0.103        | 0.018            | 0.360          | 0.000 |
| HF-Net Trained [38]     | 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 [49]          | 0.507   | 0.008        | 0.136        | 0.000            | 0.501          | 0.006 |
| Active Search [42]      | 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 |
| **Ours**                | 1.441   | 0.538        | 0.615        | 0.358            | 0.097          | 0.227 |

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(0.05m, 5°) ↑ | Chess  | Fire   | Heads | Office | Pumpkin | Kitchen | Stairs | Average  |
|-------------------|--------|--------|-------|--------|---------|---------|--------|----------|
| Shotton et al. [47] | 92.60% | 82.90% | 49.40% | 74.90% | 73.70%  | 71.80%  | 27.80% | 67.60%   |
| Guzman-Rivera et al. [20] | 96.00% | 90.00% | 56.00% | 92.00% | 80.00%  | 86.00%  | 55.00% | 79.30%   |
| Valentin et al. [51] | 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. [44]  | 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%   |
| **Ours**            | 99.85% | 100.00%| 100.00%| 99.80% | 88.80%  | 90.96%  | 84.20% | 94.80%   |

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

| Pose(0.05m, 5°) ↑ | Chess  | Fire   | Heads | Office | Pumpkin | Kitchen | Stairs | Average  |
|-------------------|--------|--------|-------|--------|---------|---------|--------|----------|
| 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               | 27.05% | 25.80% | 5-level 8-way Tree | 24.60% | 3-level 16-way Tree | 16.75% | 4-level 16-way Tree | 25.31% |          |
| Ours w. pose refinement | 31.99% | 27.05% | 25.80% | 5-level 8-way Tree | 24.60% | 3-level 16-way Tree | 16.75% | 4-level 16-way Tree | 25.31% |

4.4. Numerical Results

We compare our algorithm with all the other approaches on the test split of the RIO-10 dataset, shown in Figure 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(0.05m, 5°), 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
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 [53].

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 multi-leaves in camera pose optimization by removing it 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.

Outlier rejection for semantic changes. A significant amount of outlier input points are generated due to the semantic changes in dynamic environments. We evaluate the robustness of our outlier rejection strategy by randomly sampling 1000 points per image from the semantically change regions annotated in the RIO-10 validation set, and then testing the success ratio of rejecting these points as 63.76%, which demonstrates the effectiveness of our proposed robust neural routing strategy. We further visualize rejected points in Figure 5.

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 learns a neural routing function to reject the dynamic input points during the level-wise routing process.

6. Appendix

The appendix provides the additional supplemental material that cannot be included into the main paper due to its page limit:

- More Space Partition Strategies.
- Extension to Neural Forest.
- Unified Neural Routing Function – PointNet++.
- Ablation for HyperNetwork.

A. More Space Partition Strategies

In the main paper, we evaluate different space partition strategies by varying the hyper-parameters in a \( t \)-level \( m \)-way tree. In this section, we conduct more experiments by constructing the covered bounding boxes in different manners, which is another important factor that influences the tree structure. To be specific, we firstly follow the axis of world coordinate system and compute the axis-aligned minimum bounding box (AABB) of the entire scene as the root node for space partition. This is our original implementation in the main paper. To explore more space partition strategies, we rotate the scene model along the x and y axis by 30°, and calculate the new AABB. Similarly, we also obtain a version by rotating 60°. The bounding boxes constructed above all follow the world coordinates, and may leave many blank 3D spaces in the box, which does not make full use of the neural routing functions. To resolve this issue, we obtain the compact box by recalculating the world coordinate system of the scene using PCA [34], and fit the tightest bounding box along the new axis. In order to alleviate the potential influence of coordinate axis to camera pose optimization as observed in [6, 4], we further rotate the compact box to align with the default world coordinate axis for a fair comparison with other boxes.

We illustrate the different bounding box constructions in Figure 6. We observe that regarding the compactness between the bounding box and 3D scene model, compact box > original box > rotation 60° > rotation 30°. The corresponding numerical results of the above space partition strategies are shown in Table 4. Consistent with the compactness, the camera pose accuracy also follows the same order: compact box > original box > rotation 60° > rotation 30°. It indicates an interesting observation that the more compact the box is, the higher the pose accuracy can be achieved by our algorithm. This is mainly because in a compact box, the geometric regions are more uniformly sampled among all the split nodes in the decision tree, which strengthens the utilization of the neural routing functions.

B. Extension to Neural Forest

In the existing camera relocalization works implemented with decision trees [12, 11], they usually train a number of trees on the same scene to obtain a forest for the pose optimization. In this way, the final prediction of the forest is simply the union of the fitted modes among all the trees. In these works, the decision rule for each split node is adaptively learned as either color or depth comparison from the training data. Hence by simply sampling different input samples in the same scene, they are able to learn different decision trees.

Motivated by the previous works, we further extend our
proposed neural tree to neural forest by training multiple trees. However, in our work, the decision rules are predetermined by the space partition strategy. In order to enable the diversity of multiple trees, we adopt the four different space partition strategies introduced in Section 6, and unify their predictions to form a neural forest. The numerical results are shown in Table 4. We are glad to observe that the performance can be further upgraded by the utilization of a forest.

C. Unified Neural Routing Function – PointNet++

The main paper employs the hierarchical node-wise neural routing functions to classify each input query point into one of the leaf nodes. This can be naturally viewed as the point-wise segmentation task, where each segmentation label refers to one leaf node. As the input is formulated as the form of point cloud, we can achieve a unified neural routing function by directly leveraging the popular state-of-the-art point cloud segmentation network PointNet++ [36]. In our problem setting, the PointNet++ takes the colored point cloud from a single frame as input, and outputs the point-wise segmentation mask. In this case, each input point is the query point and also serves as the context point for the other query points. In this unified neural routing function, the outlier rejection is not an option anymore and excluded from the segmentation label. We adopt the MSG as the PointNet++ backbone in the implementation.

Its numerical result is shown in Table 4, which performs much worse compared to our neural routing function implementation. It demonstrates the effectiveness of our unique neural tree design.

D. Ablation for HyperNetwork

In our algorithm, HyperNetwork unifies the network parameters of all the neural routing functions from the same level into a single network, and hence saves much storage space and training time. However, as observed in the previous work [19], HyperNetwork may potentially degrade the performance compared to the version that separately trains each network. To investigate the potential influence of HyperNetwork on the performance of neural routing functions, we select three split nodes for each level in our neural tree, and compare their classification accuracy between the implementations with and without HyperNetwork as shown in Table 5. Interestingly, we observe that the usage of HyperNetwork only degrades the performance within a reasonable range similar to the past experience [19].

|        | w. HyperNet | w/o HyperNet |
|--------|-------------|--------------|
| node 1 | 58.1% 75.9% 70.5% | 60.5% 79.5% 75.0% |
| node 2 | 57.9% 66.5% 69.5% | 56.8% 62.6% 69.3% |
| node 3 | 23.6% 48.5% 49.3% | 28.9% 66.7% 50.0% |
| average| 46.5% 63.6% 63.1% | 48.7% 69.9% 64.7% |

Table 5. Ablation study of HyperNetwork on the scene 01 in the validation set of RIO-10 dataset. For each level, we collect three split nodes to evaluate their classification accuracy on the validation set.

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, 7
[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. 6
[3] Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. arXiv preprint cs/0011032, 2000. 4
[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. 2, 9
[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. 2, 7
[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. 2, 9
[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. 2
[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. 2
[9] Samarath 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. 2
[10] Lauriane Castin and Benoit Frénay. clustering with decision trees: divisive and agglomerative approach. In ESANN, 2018. 4
[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. 1, 2, 3, 4, 6, 7, 8, 9
[12] Tommaso Cavalli, 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. 2, 4, 6, 7, 9

[13] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003. 2, 6

[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. 6

[15] 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. 2

[16] Karl Pearson. Liii. on lines and planes of closest fit to systems of points. In *Philosophical Magazine and Journal of Science*, 24(5):603–619, 1924. 6

[17] 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. 2

[18] 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. 2

[19] 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. 2

[20] 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. 2

[21] 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. 2

[22] 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. 2
suasal descriptors for hierarchical efficient localization. *arXiv preprint arXiv:1809.01019*, 2018. 2

[40] 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. 2

[41] 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. 2

[42] 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. 2, 7

[43] 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. 2

[44] 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. 7

[45] 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. 3

[46] 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. 3

[47] 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. 2, 4, 6, 7

[48] 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. 2

[49] 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. 7

[50] 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. 2

[51] 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. 2, 6, 7

[52] 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. 2

[53] Johanna Wald, Torsten Sattler, Stuart Golodetz, Tommaso Cavallari, and Federico Tombari. Beyond controlled environments: 3d camera re-localization in changing indoor scenes. *ECCV*, 2020. 2, 6, 7, 8

[54] Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. Atlloc: Attention guided camera localization. 2020. 2

[55] 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. 2