Exploiting Points and Lines in Regression Forests for RGB-D Camera Relocalization

Lili Meng[1], Frederick Tung[1], James J. Little[1], Julien Valentin[2], Clarence W. de Silva[1]

Abstract— **Camera relocalization plays a vital role in many robotics and computer vision applications, such as self-driving cars and virtual reality. Recent random forests based methods exploit randomly sampled pixel comparison features to predict 3D world locations for 2D image locations to guide the camera pose optimization. However, these point features are only sampled randomly in images, without considering geometric information such as lines, leading to large errors with the existence of poorly textured areas or in motion blur. Line segments are more robust in these environments. In this work, we propose to jointly exploit points and lines within the framework of uncertainty driven regression forests. The proposed approach is thoroughly evaluated on three publicly available datasets against several strong state-of-the-art baselines in terms of several different error metrics. Experimental results prove the efficacy of our method, showing superior or on-par state-of-the-art performance.**

I. INTRODUCTION

Camera relocalization plays a vital role in many computer vision, robotics, augmented reality (VR) and virtual reality (AR) applications. In the real world, camera relocalization has empowered the recent consumer robotics products such as Dyson 360 Eye and iRobot Roomba 980 to know where they have previously visited [1]. In AR/VR products such as Hololens and Oculus Rift, camera relocalization helps to correctly overlay visual objects in an image sequence or real world.

Scene Coordinate Regression Forests (SCRF) [2] is the pioneer in using machine learning for camera relocalization. In this method, a regression forest is trained to infer an estimate of each pixel's correspondence to 3D points in the world coordinate. Then these correspondences are used to infer the camera pose with a robust optimization scheme. Since then, various machine learning based methods, mainly random forests based [3], [4], [5], [6], [7], [8], [9] and deep learning based methods [10], [11], [12], [13], [14], [15] have been proposed to accelerate the progress of camera relocalization, in parallel with the traditional but still active feature-based methods [16], [17] and key-frame based methods [18], [19].

In these random forests based methods, either RGB-D/RGB pixel comparison features [2], [3], [5], [7], or the sparse features such as SIFT [8] are employed, without considering the spatial structure. In poorly textured areas or with the existence of motion blur, line segments provide important geometric information and are robust image features

[1] Lili Meng, Frederick Tung, James J. Little and Clarence W. de Silva are with Institute for Computing, Information and Cognitive Systems (ICICS), The University of British Columbia, Vancouver, Canada.
[2] Julien Valentin is with Perceptive^{IO} Inc, United States.

Fig. 1: Line segment example. (a) Input image (b) Line segments. In scenes with little texture and repetitive patterns which are typical in indoor environments, line segments are more robust.

[20] as shown in Fig. 1. Therefore, we propose to use both general points and line-segment points in regression forests to improve camera relocalization performance. The general points are randomly sampled from the image. The line-segment points are explicitly sampled from line segments in the image. The latter implicitly encodes the line structure of the scene. The main contributions of this work are:

- We exploit general points and line-segments points in scene coordinate regression forests, increasing the prediction accuracy.
- We use the uncertainty of general points predictions and line-segment points predictions to optimize the camera pose.
- We thoroughly evaluate our methods on three publicly available datasets against several strong state-of-the-art baselines, proving the efficacy of our method with superior or on-par accuracy.

II. RELATED WORK

Camera relocalization has been widely studied in large scale global localization [21], [22], [10], recovery from tracking failure [18], [19], loop closure detection in visual SLAM [23], [24] and global localization in mobile robotics [16], [25], and sports camera calibration [26], [27]. Various methods have been proposed to advance camera relocalization performance. We provide a review of random forests based methods here and refer to [8] for a review of other methods.

A. Random forests based method for camera relocalization

In approaches based on random forests for camera relocalization, random forests are used as regressors so we also refer to them as regression forests. These approaches first employ

a regression forest to learn each 2D image pixel's corresponding 3D points in the scene's world coordinates with training RGB-D images and their corresponding ground truth poses. Then camera pose optimization is conducted by an adapted version of preemptive RANSAC [2]. Random forests based methods do not need to compute ad hoc descriptors and search for nearest-neighbors, which are time-consuming steps in local feature based and key-frame based methods. Because the environment generally has repeated objects, such as similar chairs in an office room, the random forests have multi-outputs from an input, resulting in ambiguities. To solve this problem, [3] proposed a hybrid discriminative-generative learning architecture to choose optimal camera poses in SCRF. To improve camera relocation accuracy, [4] exploits uncertainty from regression forests by using a Gaussian mixture model in leaf nodes. [6], [7] extend the random forests based method to use a single RGB image in the test stage by employing the Perspective-n-Point method rather than the Kabsch [28] algorithm in the camera pose optimization stage. However, RGB-D images are still needed in the training stage. [8] integrates local features in the regression forests to enable the use of RGB-only images for both training and testing. However, none of these random forests based methods have been evaluated in dynamic scenes for the indoor environment, where the dynamic objects such as people or pets are common.
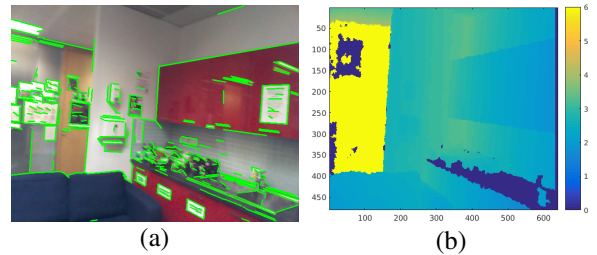
B. Line segments

Line segment detection and exploitation have been studied for more than three decades and are still very active [29], [30], [31], [32], [20]. Robust gradient orientations of the line segment rather than robust endpoints or gradient magnitudes play a crucial role in the line segment literature [30], [32], [20]. Besides line segment detection, this work is also related to pose estimation using line and/or point features [33], [20], [26], [34]. Unlike methods using line features for direct matching [34], we integrate the line features in the random forests for supervised learning.
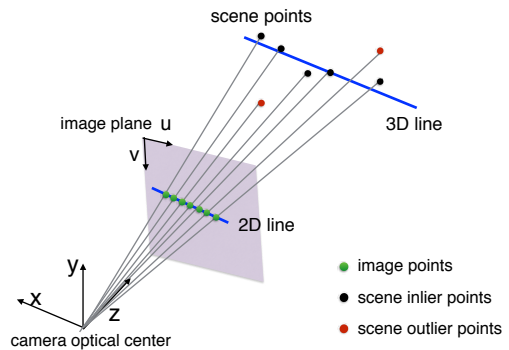
III. PROBLEM SETUP AND METHOD OVERVIEW

The following three assumptions of the RGB-D camera and input data are made: (i) the camera intrinsics are known; (ii) the RGB and depth frames are synchronized; (iii) the training set contains both RGB-D frames and their corresponding 6 DOF camera pose.

The problem is formulated as: given only a single acquired RGB-D image $\{I, D\}$, infer the pose H of an RGB-D camera relative to a known scene.

To solve the problem, we propose to exploit both points and line segments in uncertainty-driven regression forests. Our method consists of two major components. The first component is two regression forests trained using general points and line-segment points respectively. These two forests predict general points and line-segment points in testing. The second component is a camera pose optimization scheme using both point-to-point constraints and point-on-line constraints.

(a)　　　　　　(b)

Fig. 2: Depth corruption and discontinuity on line segments. (a) LSD line segments overlaid on original RGB image (b) truncated depth map. Effective depth information is not always available for 2D line segments in the corresponding RGB image, such as the wrong depth values shown on the desk and the glass corridor areas.

Fig. 3: 3D line estimation based on sampling points. Within a pinhole camera model, the 2D image points are evenly sampled on a 2D image line and then back-projected on the scene coordinate to be 3D scene points. These 3D scene points contain outliers which could be removed by RANSAC by fitting a 3D line in scene coordinates.

IV. REGRESSION FORESTS WITH GENERAL POINTS AND LINE-SEGMENT POINTS

A. Points sampling and scene coordinate labels

To take advantage of the complementary properties of lines and points, our method differentiates the points that are on line segments and general points.

Line segment sampling: Directly back-projecting the two endpoints to a 3D line using the depth information will cause large errors [34] due to discontinuous depth on object boundaries or lack of depth information as shown in Fig. 2. To avoid this problem, the Line Segment Detector (LSD) [32] method is employed to extract a set of 2D line segments $L = \{l_1, l_2, \cdots\}$ from image I as shown in Fig. 1, and then uniformly sample points from the line as shown in Fig. 3. Using this method, one could discard the sample points whose depths are unavailable, and only back-project the remaining points to the camera coordinates. The outliers of the back-projected points are further removed by fitting a 3D line using RANSAC [35].

Random points sampling: Besides the line-segment points, we also randomly sample points in the image. These

two types will be used separately in the training/testing process.

Image features: We use the pixel comparison feature [2] that associates with each pixel location $\mathbf{p}$:

$$f_\phi(\mathbf{p}) = \text{I}(\mathbf{p}, c_1) - \text{I}\left(\mathbf{p} + \frac{\delta}{\text{D}(\mathbf{p})}, c_2\right) \tag{1}$$

where δ is a 2D offset and $\text{I}(\mathbf{p}, c)$ indicates an RGB pixel lookup in channel c. The ϕ contains feature response parameters $\{\delta, c_1, c_2\}$. The $\text{D}(\mathbf{p})$ is the depth at pixel $\mathbf{p}$ in image D of the corresponding RGB image I. Pixels with undefined depth and those outside the image boundary are marked and not used as samples for both training and test. In leaf nodes, we also use Walsh-Hadamard Transform (WHT) features [36] to describe the local patches [8].

Scene coordinate labels: For both general points and line-segment points, the 3D points $\mathbf{x}$ in camera coordinate of the corresponding pixel $\mathbf{p}$ are computed by back-projecting the depth image pixels. The scene's world coordinate position $\mathbf{m}$ of the corresponding pixel $\mathbf{p}$ is computed through $\mathbf{m} = \text{H}\mathbf{x}$.

With the present sampling method, one can train both the general point prediction model and line-segment point prediction model in the same way. An alternative method is to use two different models. One model predicts point-to-point correspondences and another model directly predicts line-to-line correspondences. The challenge of the alternative method is that there are few robust and efficient representations of lines in the feature space. Therefore, the proposed method predicts point correspondences and employs point-on-line constraint in the camera optimization process, which greatly simplifies the model learning and prediction process.

B. Regression forest training

A regression forest is an ensemble of T independently trained decision trees. At this stage, we train a general point regression forest and a line-segment point regression forest. Each tree is a binary tree consisting of split nodes and leaf nodes.

Weak learner model: Each split node i represents a weak learner parameterized by $\theta_i = \{\phi_i, \tau_i\}$ where ϕ_i is one feature dimension and τ_i is a threshold. The tree grows recursively from the root node to the leaf node. At each split node, the parameter θ_i is sampled from a set of randomly sampled candidates Θ_i. At each split node i, for the incoming training set S_i, samples are evaluated on split nodes to learn the split parameter θ_i that best splits the left child subset S_i^L and the right child subset S_i^R as follows:

$$h(\mathbf{p}; \theta_i) = \begin{cases} 0, & \text{if } f_{\phi_i}(\mathbf{p}) \leq \tau_i, \quad \text{go to the left subset } S_i^L. \\ 1, & \text{if } f_{\phi_i}(\mathbf{p}) > \tau_i, \quad \text{go to the right subset } S_i^R. \end{cases} \tag{2}$$

Here, τ_i is a threshold on feature $f_{\phi_i}(\mathbf{p})$. Although here we use random pixel comparison features as in Eq. 1, the weak learner model can use other general features to adapt to application scenarios such as SIFT feature for outdoor environment as [8].

Training objective: In the training, each split node i uses the randomly generated Θ_i to greedily optimize the parameters θ_i^* that will be used as the weak learner in the test phase by maximizing the information gain I_i:

$$\theta_i^* = \arg\max_{\theta_i \in \Theta_i} I_i(S_i, \theta_i) \tag{3}$$

with

$$I_i = E(S_i) - \sum_{j \in \{L, R\}} \frac{|S_i^j(\theta_i)|}{|S_i|} E(S_i^j(\theta_i)) \tag{4}$$

where $E(S_i)$ is the entropy of the labels in S_i, and subset S_i^j is conditioned on the split parameter θ_i. The present work employs a single full-covariance Gaussian model, with the entropy defined as:

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

with dimensionality $d = 3$ and Λ is the full covariance of the labels in S.

At the end of training, all samples reach leaf nodes. In a leaf node, we use the mean shift method [37] to estimate a set of modes. Each mode has a mean vector μ and a covariance matrix Λ to described the clustered 3D points. We also store a mean vector of local patch descriptors (i.e. WHT features) for each mode. The local patch descriptor will be used to choose the optimal predictions.

C. Regression forest prediction

In testing, we use the backtracking technique [8] to find the optimal prediction within time budgets using a priority queue. In backtracking, the optimal mode has the minimum feature distance from the patch descriptor. To speed up, we use the backtracking number of 8 instead of 16 as in [8].

V. CAMERA POSE OPTIMIZATION

Our method optimizes the camera pose using two types of constraints. The first constraint is point-to-point correspondence. For each sampled camera coordinate point x_i^c, the mode is found in $\mathcal{M}_i$ that concurrently best explains the transformed observation $\text{H}x_i^c$:

$$(\mu_i^*, \Sigma_i^*) = \arg\max_{(\mu, \Sigma) \in \mathcal{M}_i} \mathcal{N}(\text{H}x_i^c; \mu, \Sigma). \tag{6}$$

This can be optimized by minimizing the sum of Mahalanobis distances in world coordinates for image points set $\mathcal{I}_p$:

$$E_p = \sum_{i \in \mathcal{I}_p} \|(\text{H}x_i^c - \mu_i^*)^T \Sigma_{x_i}^{*-1} (\text{H}x_i^c - \mu_i^*)\|. \tag{7}$$

The second constraint is point-on-line constraint. For each predicted edge point x_i^w, the present work transforms its location to the camera coordinate $\text{H}^{-1}x_i^w$ and measures the Mahalanobis distance for line-segment points set $\mathcal{I}_l$ to the associated line L_i. This can be optimized by minimizing:

$$E_l = \sum_{i \in \mathcal{I}_l} \|(\text{H}^{-1}x_i^w - Q)^T \Sigma_i^{-1} (\text{H}^{-1}x_i^w - Q)\| \tag{8}$$

where Q_i is the closest point on L_i to the transformed point $H^{-1}x_i^w$. The covariance matrix Σ_i is rotated from the world coordinate to the camera coordinate by

$$\Sigma_c = R\Sigma R^T \tag{9}$$

where R is the rotation matrix in H^{-1}.

Our method jointly optimizes these two constraints by using the sum over the Mahalanobis distances as the performance index:

$$H^* = \arg\min_H(E_p + E_l) \tag{10}$$

This energy is optimized by a Levenberg-Marquardt optimizer [38] in a preemptive RANSAC framework as in [2].

VI. Experiments

This section evaluates the developed method on three publicly available datasets for camera relocalization against several strong baselines.

A. Evaluations on Stanford 4 Scenes dataset

Dataset: The 4 *Scenes* dataset [5] is with spatial extent of 14 to $79m^3$. This large environment includes the offices and apartments which are practical for the application of indoor robot localization. The RGB image sequences were recorded at a resolution of 1296×968 pixels, and the depth resolution is 640×480. We re-sampled the RGB images to the depth image resolution to align these images. The ground truth camera poses were from BundleFusion [39].

Baselines and error metric: We use the following state-of-the art methods as baselines: ORB+PnP, SIFT+PnP, Random+SIFT [7], MNG [5], BTBRF [8], Surface Regression (SR) [40]. The details on these baselines can be found in [5], [8], [40].

We adopt the commonly used error metric of the proportion of test frames within $5cm$ translational and $5°$ angular error.

Main results and analysis: Table I shows the main quantitative result on the 4 *Scenes* dataset. The proposed method PLForests considerably outperforms all the baselines, with the average correct frame percentage of 99.3%. Fig. 4 shows some qualitative results of the present camera pose estimation. The estimated camera poses including translations and orientations are very similar to the ground truth camera poses. However, for some scenes, such as Luke, there still exist a few large error camera pose estimates. To further investigate the large error, the RGB images and their large error poses are shown in Fig. 5. From the RGB images, it is seen that only a very little color information is available from the image. At the same time, the line segments shown in Fig. 5 (a) are not very apparent as well.

Here we do not compare speed mainly as there is no speed information on the same hardware benchmark. Our current implementation is not optimized for speed and no GPU is used, which makes it possible to speed up with more engineering effort and GPU implementation [41].

B. Evaluations on Microsoft 7 Scenes dataset

The developed method is also evaluated on the Microsoft 7 Scenes dataset.

Dataset: The Microsoft 7 Scenes dataset [2] consists of 7 scenes which were recorded with a handheld Kinect RGB-D camera at 640×480 resolution. Each scene includes several camera sequences that contain RGB-D frames together with the corresponding ground-truth camera poses which are obtained from the KinectFusion [42] system. The dataset exhibits shape and color ambiguities, specularities and motion blur, which present great challenges for camera relocalization.

Baselines and error metric: The developed method is compared against SCRF [2], a multi-output version of SCRF [3], an uncertainty version of SCRF [4] and an autocontext version of SCRF [43], Dense correspondence [14] and SR [40] in terms of correct frame percentage. We also provide results in terms of median translation error and rotation error against Bayesian PoseNet [11], PoseNet+Geometric [12], CNN+LSTM [13], Active Search [17], SCRF [2], BTBRF [8].

Main results and analysis: The main results are shown in comparison with several strong baselines in terms of correct frames in Table II, and in terms of median performance in Table. III. The proposed method achieves superior accuracy in terms of median performance, and on-par accuracy in terms of correct frames compared with various methods. Admittedly, Dense [14] achieves a little bit better average accuracy (1.7%), our method still can improve its accuracy with larger backtracking leaf node numbers but at the expense of reducing its speed. Compared with the 4 Scenes dataset, the average accuracy on the 7 Scenes dataset is relatively low. Several reasons may account for this: (i) the training and testing sequences in the 4 Scenes dataset are recorded at the same time as a whole sequence by the same person while the training and test sequences of the 7 Scenes dataset are recorded by different users as different sequences, and split into distinct training and testing sequence sets. (ii) the depth and RGB images have better quality and have better registration in the 4 Scenes. (iii) the scenes in the 7 Scenes are more challenging due to high ambiguity especially in *Stairs* scene and severe motion blur. Fig. 6 shows some qualitative results for the *Heads* scene of the 7 Scenes dataset. The estimated camera pose is similar to ground truth. Large errors occur in places where the test poses are very different from training poses. Similar findings are also seen in some other scenes.
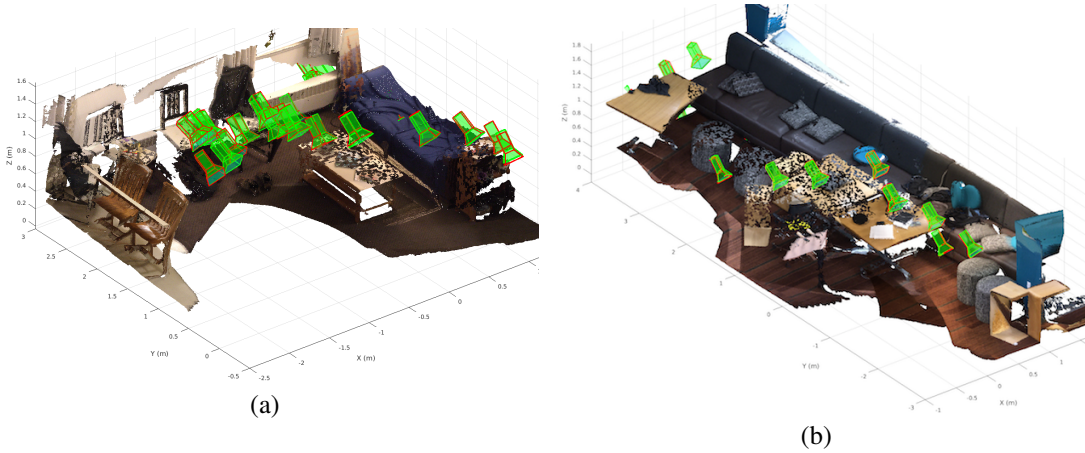
C. Evaluations on TUM dynamic dataset

To demonstrate the performance on datasets that have dynamic objects, the developed method is evaluated using the TUM dynamic dataset.

TUM RGB-D dynamic dataset: TUM RGB-D Dataset [44] is mainly for the evaluation of RGB-D SLAM systems. A large set of image sequences of various characteristics (e.g. non-texture, dynamic objects, hand-held SLAM, robot SLAM) from a Microsoft Kinect RGB-D sensor. The ground

Sequence	Spatial Extent	Baselines						Ours
		ORB+PnP	SIFT+PnP	Hybrid [7]	MNG [5]	BTBRF [8]	SR [40]	PLForests
Kitchen	$33m^3$	66.39%	71.43%	70.3%	85.7%	92.7%	**100%**	98.9%
Living	$30m^3$	41.99%	56.19%	60.0%	71.6%	95.1%	**100 %**	**100%**
Bed	$14m^3$	71.72%	72.95%	65.7%	66.4%	82.8%	**99.5%**	99.0%
Kitchen	$21m^3$	63.91%	71.74%	76.7%	76.7%	86.2%	**99.5%**	99.0%
Living	$42m^3$	45.40%	56.19%	52.2%	66.6%	99.7%	**100%**	**100%**
Luke	$53m^3$	54.65%	70.99%	46.0%	83.3%	84.6%	95.5%	**98.9%**
Floor5a	$38m^3$	28.97%	38.43%	49.5%	66.2%	89.9%	83.7%	**98.8%**
Floor5b	$79m^3$	56.87%	45.78%	56.4%	71.1%	98.9%	95.0%	**99.0%**
Gates362	$29m^3$	49.48%	67.88%	67.7%	51.8%	96.7%	**100%**	**100%**
Gates381	$44m^3$	43.87%	62.77%	54.6%	52.3%	92.9%	96.8%	**98.8%**
Lounge	$38m^3$	61.16%	58.72%	54.0%	64.2%	94.8%	95.1%	**99.1%**
Manolis	$50m^3$	60.10%	72.86%	65.1%	76.0%	98.0%	96.4%	**100%**
Average	—	53.7%	62.2%	59.9%	69.3%	92.7%	96.4%	**99.3%**

TABLE I: Camera relocalization results for the 4 Scenes dataset. The percentage of correct frames (within $5cm$ translational and $5°$ angular error) of the developed method is shown against six state-of-the-art methods: ORB+PnP, SIFT+PnP, Random+SIFT [7], MNG [5], SR [40]. The best performance is highlighted.
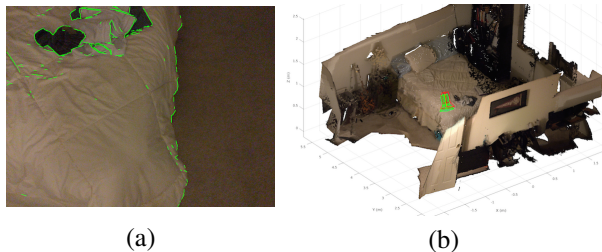

(a)

(b)

Fig. 4: Qualitative results on the 4 Scenes dataset. Best viewed in color. (a) apt1_living, (b) office2_5b. We evenly sample every 20 frames in the 3D reconstructed scenes for visualization. The ground truth (hollow red frusta) and the present estimated camera pose (green with light opacity) are very similar. Please note the 3D scenes are only used for visualization purposes and are not used in the present algorithm.

Scene	Space	Baselines						Ours
		SCRF[2]	Multi[3]	Uncertainty[4]	AutoContext [43]	Dense[14]	SR [40]	PLForests
Chess	$6m^3$	92.6%	96%	99.4%	**99.6%**	97.8%	97.6%	99.5%
Fire	$2.5m^3$	82.9%	90%	94.6%	94.0%	96.6%	91.9%	**97.6%**
Heads	$1m^3$	49.4%	56%	89.3%	95.9%	**99.8%**	93.7%	95.5%
Office	$7.5m^3$	74.9%	92%	**97.0%**	93.4%	**97.2%**	87.3%	96.2%
Pumpkin	$5m^3$	73.7%	80%	**85.1%**	77.6%	81.4%	61.6%	81.4%
Kitchen	$18m^3$	71.8%	86%	89.3%	91.1%	**93.4%**	65.7%	89.3%
Stairs	$7.5m^3$	27.8%	55%	63.4%	71.7%	**77.7%**	28.7%	72.7%
Average	—	67.6%	79.3%	89.5%	88.1%	**92.0%**	76.6%	90.3%

TABLE II: Relocalization results for the 7 Scenes dataset. Test frames satisfying the error metric (within 5cm translational and $5°$ angular error) are shown for the proposed method on all scenes against five strong state-of-the-art methods: SCRF [2], Multi[3], Uncertainty[4], AutoConext [43], Dense[14] and SR [40]. The best performance is highlighted.

			Baselines				Ours
Scene	Geometric [10]	Bayesian [11]	CNN+LSTM [13]	Active Search[17]	SCRF [2]	BTBRF [8]	PLForests
Training	RGB	RGB	RGB	RGB	RGB-D	RGB-D	RGB-D
Test	RGB	RGB	RGB	RGB	RGB-D	RGB-D	RGB-D
Chess	0.13m, 4.48°	0.37m, 7.24°	0.24m, 5.77°	0.04m, 1.96°	0.03m, 0.66°	0.015m, 0.59°	**0.014m, 0.57°**
Fire	0.27m, 11.3°	0.43m, 13.7°	0.34 m, 11.9°	0.03m, 1.53°	0.05m, 1.50°	0.016m, 0.89°	**0.009m, 0.48°**
Heads	0.17m, 13.0°	0.31m, 12.0°	0.21m, 13.7°	0.02m, 1.45°	0.06m, 5.50°	0.020m, 1.84°	**0.008m, 0.68°**
Office	0.19m, 5.55°	0.48m, 8.04°	0.30 m, 8.08°	0.09m, 3.61°	0.04m, 0.78°	0.018m, 0.75°	**0.017m, 0.73°**
Pumpkin	0.26m, 4.75°	0.61m, 7.08°	0.33 m, 7.00°	0.08m, 3.10°	0.04m, 0.68°	0.023m, 0.84°	**0.019m, 0.65°**
Kitchen	0.23m, 5.35°	0.58m, 7.54°	0.37 m, 8.83°	0.07m, 3.37°	0.04m, 0.76°	0.025m, 1.02°	**0.025m, 1.02°**
Stairs	0.35m, 12.4°	0.48m, 13.1°	0.40 m, 13.7°	0.03m, 2.22°	0.32m, 1.32°	0.040m, 1.22°	**0.027m, 0.71°**
Average	0.23m, 8.11°	0.47m, 9.81°	0.31 m, 9.85°	0.05m, 2.46°	0.08m, 1.60°	0.022m, 1.02°	**0.017m, 0.70°**

TABLE III: Median performance for the 7 Scenes dataset. Results are shown for all scenes against six baselines: PoseNet+Geomeric [12], Bayesian PoseNet [11], Active Search without prioritization [17], SCRF [2], BTBRF [8]. The best performance is highlighted.
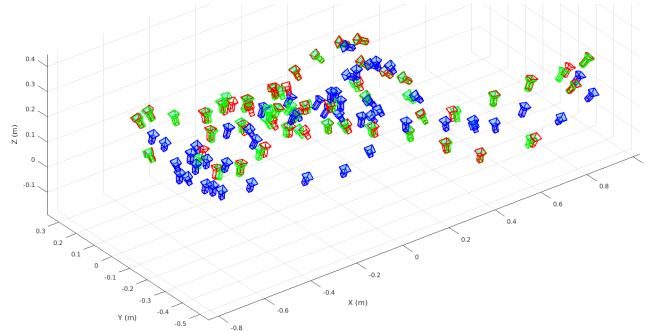

(a) (b)

Fig. 5: Large error examples on Stanford 4 Scenes dataset Apt2_Luke. (a) an RGB image overlaid with LSD line segments, (b) ground truth (red) and estimated camera pose (green) in the 3D scene. The white sheet and gray floor dominate the scene, little color information and few line segments cause large camera pose error.

Fig. 6: Qualitative results for the *Heads* scene in the 7 Scenes dataset. Best viewed in color and enlarge. Training (blue), test ground truth (red), and test estimated camera poses (green) are evenly sampled for every 20th images. The estimated camera pose is similar to ground truth in both translation and rotation. The large errors occur in places where training poses are very different from test poses.

truth is from a highly accurate and time-synchronized motion capture system. The sequences contain both the color and depth images at an image resolution of 640×480. Here, just the dynamic objects dataset is used to complement the previous static 7 Scenes dataset and 4 Scenes dataset where no dynamic objects exist. This dynamic dataset is very challenging as there are severe occlusions and moving objects in the scene. The training set is from the scenes as listed in Table IV while the test set is from the respective evaluation sequences.

Baselines and error metric: Three error metrics are used for this evaluation: 'Correct frames' which is the percentage of test frames within 5*cm* and 5°, median translational and angular error, and root mean squared error (RMSE) for Absolute Trajectory Error (ATE) [44] which is commonly used in many SLAM system [44], [45]. Unlike the 7 Scenes and 4 Scenes datasets, the training and evaluation data are in different world coordinate systems. The estimated trajectory is still in the training data world coordinate. For alignment, the TUM dataset benchmark provide tools using Horn's method [46] to align the estimated trajectory $\mathbf{P}_{1:n}$ and ground truth trajectory $\mathbf{Q}_{1:n}$. The ATE $\mathbf{F}$ at time step i is computed as

$$\mathbf{F}_i = \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \qquad (11)$$

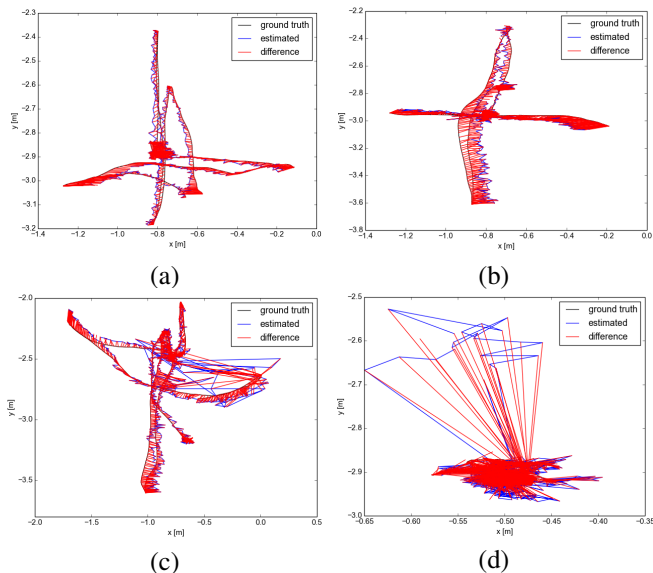The root mean square error (RMSE) over all time indices of the translational components is computed as:

$$RMSE(\mathbf{F}_{1:n}) = (\frac{1}{n}\sum_{i=1}^{n}||trans(\mathbf{F}_i)||^2)^{\frac{1}{2}} \qquad (12)$$

However, this ATE could only be used an auxilliary error metric, as it only considers the translational error while ignoring rotational errors. The translational and rotational error are simultaneously optimized.

Main results and analysis: Table IV shows the main results of the proposed method on the TUM Dynamic dataset. Compared with the static datasets, our method works with a lower accuracy due to high dynamics. Our method is more accurate than BTBRF in terms of average correct frames and RMSE of ATE, and has the same average median performance. The proposed method could work satisfactorily under highly dynamic scenes and other challenges but still struggles in some extreme cases. Fig. 7 shows the qualitative results on two good and two bad sequences. In these bad sequences, dynamic occlusions dominate the scene, which causes too few inliers and therefore lead to failure cases, such as shown in Fig. 8 (a). The other important failure case

Scene	Trajectory	Baseline			Ours		
		BTBRF [8]			PLForests		
		Correct	Median	RMSE	Correct	Median	RMSE
sitting_static	0.26m	64.6%	0.015m, 0.99°	0.018m	**72.2%**	**0.012**m, **0.93°**	**0.016**m
sitting_xyz	5.50m	70.2%	0.029m, 0.72°	**0.039**m	**74.1%**	**0.028**m, 0.73°	0.047m
sitting_halfsphere	6.50m	**44.4%**	**0.056**m, **1.59°**	**0.046**m	39.8%	0.061m, 1.76°	0.072m
sitting_rpy	1.11m	74.6%	0.029m, 0.98°	**0.065**m	**77.9%**	**0.026**m, **0.94°**	0.069m
walking_halfsphere	7.68m	**61.7%**	**0.042**m, **1.03°**	**0.085**m	46.6%	0.052m, 1.26°	0.111m
walking_rpy	2.70m	**53.7%**	0.047m, 1.14°	0.551m	53.4%	0.047m, **1.01°**	**0.169**m
walking_static	0.28m	89.2%	0.019m, 0.49°	0.027m	**97.3%**	**0.017**m, **0.35°**	**0.021**m
walking_xyz	5.79m	41.7%	0.048m, 1.24°	0.064m	**46.6 %**	**0.047**m, **1.22°**	**0.063**m
Average		62.5%	0.036m, 1.02°	0.119m	**63.5%**	0.036m, 1.02°	**0.071**m

TABLE IV: Camera relocalization results for the TUM dynamic dataset. Performances are shown using three different error metrics: correct percentage, median, RMSE of ATE.
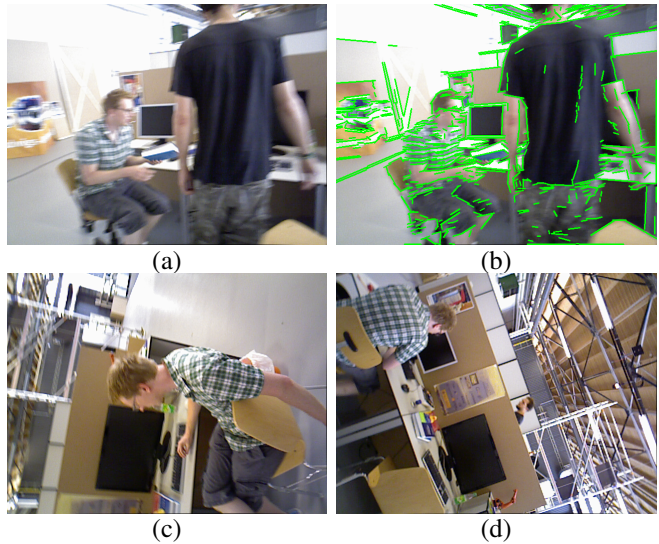

Fig. 7: Quantitative results on *TUM* dynamic dataset. Two good scenes and two bad scenes are shown. (a) sitting_xyz, (b) walking_xyz, (c) walking_halfsphere, (d) sitting_rpy.

Fig. 8: Failure cases on TUM dynamic dataset. (a) walking_halfsphere. Dynamic objects dominates the image and severe motion blur exists. (b) walking_halfsphere, overlaid with LSD line segments, there are too many line segments on dynamic objects. (c) sitting_halfsphere, (d) walking_rpy. Large rotation angle changes.

is large rotation angle changes, as shown in Fig. 8 (c) (d). This is because the random RGB comparison feature is not rotation invariant. Although our PLForests performs much better in static scenes such as 7 Scenes and 4 Scenes than BTBRF, there is no significant difference between BTBRF and PLForests in the TUM dynamic scenes in terms of correct frames and median performance. It shows that a different error metric matters. One possible reason is that there are too many line segments on dynamic objects such as shown in Fig.8 (b), but they are not stable line segments in correspondence matching and can not be filtered as outliers by RANSAC in this situation.

D. Implementation details

The proposed method is implemented with C++ using OpenCV on an Intel 3GHz i7 CPU, 16GB memory Mac system. The parameter settings for our PLForests are: point tree number $T_p = 5$ and line-segment point tree number $T_l = 5$; 500 training images per tree; 5,000 randomly sampled pixels per training image; the maximum depth of trees is 25; the maximum backtracking leaves is 8.

VII. Conclusions

In this work, we propose to exploit both line and point features within the framework of uncertainty-driven regression forests. We simultaneously consider the point and line predictions in a unified camera pose optimization framework. We extensively evaluate the proposed approach in three datasets with different spatial scale and dynamics. Experimental results demonstrate the efficacy of the developed method, showing superior state-of-the-art or on-par performance. Furthermore, different failure cases are thoroughly demonstrated, throwing some light into possible future work. For future work, it will be promising to improve our current method to be more robust to dynamic environments such as using weighted edge points [47]. Moreover, implementing

our current method on a GPU will be more efficient [41] and it will also be interesting to integrate our method for a robot autonomous navigation system [48].

REFERENCES

[1] R. Lukierski, S. Leutenegger, and A. J. Davison, "Room layout estimation from rapid omnidirectional exploration," in *ICRA*, 2017.

[2] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *CVPR*, 2013.

[3] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi, "Multi-output learning for camera relocalization," in *CVPR*, 2014.

[4] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. H. Torr, "Exploiting uncertainty in regression forests for accurate camera relocalization," in *CVPR*, 2015.

[5] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," in *3DV*, 2016.

[6] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, and c. Rother, "Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image," in *CVPR*, 2016.

[7] L. Meng, J. Chen, F. Tung, J. J. Little, and C. de Silva, "Exploiting random RGB and sparse features for camera pose estimation," in *BMVC*, 2016.

[8] L. Meng, J. Chen, F. Tung, J. J. Little, J. Valentin, and C. de Silva, "Backtracking regression forests for accurate camera relocalization," in *IROS*, 2017.

[9] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, L. Di Stefano, and P. H. Torr, "On-the-fly adaptation of regression forests for online camera relocalisation," in *CVPR*, 2017.

[10] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *ICCV*, 2015.

[11] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *ICRA*, 2016.

[12] ——, "Geometric loss functions for camera pose regression with deep learning," in *CVPR*, 2017.

[13] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization using LSTMs for structured feature correlation," in *ICCV*, 2017.

[14] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE RA-Letters*, 2017.

[15] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[16] S. Se, D. G. Lowe, and J. J. Little, "Vision-based global localization and mapping for mobile robots," *Robotics, IEEE Trans. on*, 2005.

[17] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE Trans on PAMI*, 2016.

[18] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality.*, 2007.

[19] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-time RGB-D camera relocalization via randomized ferns for keyframe encoding," *Visualization and Computer Graphics, IEEE Trans. on*, 2015.

[20] Y. Salaün, R. Marlet, and P. Monasse, "Robust and accurate line-and/or point-based pose estimation without manhattan assumptions," in *ECCV*, 2016.

[21] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006.

[22] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *CVPR*, 2015.

[23] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *IJRR*, 2015.

[24] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense SLAM without a pose graph," *RSS*, 2015.

[25] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *IJRR*, 2011.

[26] A. Gupta, J. J. Little, and R. J. Woodham, "Using line and ellipse features for rectification of broadcast hockey video," in *CRV*, 2011.

[27] J. Chen, F. Zhu, and J. J. Little, "A two-point method for ptz camera calibration in sports," *WACV*, 2018.

[28] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 1976.

[29] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *PAMI*, 1986.

[30] P. Kahn, L. Kitchen, and E. M. Riseman, "A fast line finder for vision-guided robot navigation," *PAMI*, 1990.

[31] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *CVIU*, 2000.

[32] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *PAMI*, 2010.

[33] H. Rehbinder and B. K. Ghosh, "Pose estimation using line-based dynamic vision and inertial sensors," *IEEE Trans on Automatic Control*, 2003.

[34] Y. Lu and D. Song, "Robust RGB-D odometry using point and line features," in *ICCV*, 2015.

[35] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

[36] Y. Hel-Or and H. Hel-Or, "Real-time pattern matching using projection kernels," *PAMI, IEEE Trans. on*, 2005.

[37] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *PAMI*, 2002.

[38] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, 1978.

[39] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration," *ACM Trans. on Graphics*, 2017.

[40] E. Brachmann and C. Rother, "Learning less is more-6D camera localization via 3D surface regression," in *CVPR*, 2018.

[41] T. Sharp, "Implementing decision trees and forests on a gpu," in *ECCV*, 2008.

[42] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), IEEE international symp. on*, 2011.

[43] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *CVPR*, 2016.

[44] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *IROS*, 2012.

[45] R. Mur-Artal, J. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Trans. on Robotics*, 2015.

[46] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSA A*, 1987.

[47] S. Li and D. Lee, "RGB-D SLAM in dynamic environments using static point weighting," *IEEE Robotics and Automation Letters*, 2017.

[48] C. Wang, L. Meng, S. She, I. M. Mitchell, T. Li, F. Tung, W. Wan, M. Q. H. Meng, and C. de Silva, "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *IROS*, 2017.