Learning Image-based Localization

by

Lili Meng

B.Sc., University of Science and Technology Beijing, 2008

M.Sc., University of Science and Technology Beijing, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Mechanical Engineering)

The University of British Columbia

(Vancouver)

September 2017

© Lili Meng, 2017

Abstract

Image-based localization plays a vital role in many tasks of robotics and computer vision, such as global localization, recovery from tracking failure, and loop closure detection. Recent methods based on regression forests for camera relocalization directly predict 3D world locations for 2D image locations to guide camera pose optimization. During training, each tree greedily splits the samples to minimize the spatial variance. This thesis develops techniques to improve the performance camera pose estimation based on regression forests method and extends its application domains. First, random features and sparse features are combined so that the new method only requires an RGB image in the testing. After that, a label-free sample-balanced objective is developed to encourage equal numbers of samples in the left and right sub-trees, and a novel backtracking scheme is developed to remedy the incorrect 2D-3D correspondence in the leaf nodes caused by greedy splitting. Furthermore, the methods based on regression forests are extended to use local features in both training and test stages for outdoor applications, eliminating their dependence on depth images. Finally, a new camera relocalization method is developed using both points and lines. Experimental results on publicly available indoor and outdoor datasets demonstrate the efficacy of the developed approaches, showing superior or on-par accuracy with several state-of-the-art baselines.

Moreover, an integrated software and hardware system is presented for mobile robot autonomous navigation in uneven and unstructured indoor environments. This modular and reusable software framework incorporates capabilities of perception and autonomous navigation. The system is evaluated are in both simulation and real-world experiments, demonstrating the efficacy and efficiency of the developed system.

Lay Summary

Image-based localization plays a vital role in many robotics and computer vision tasks. Deep learning research has attempted to overcome the challenges of using depth images for camera relocalization, making it applicable for outdoor scenes. The accuracy level, however, was low compared to other methods. The main contributions of this work resulted in increased prediction accuracy with reduced training time and eliminating the dependence on depth images which broadened the application to outdoor scenes.

Robots are operating in shared spaces indoors with people more and more. However, robots still face challenges operating in uneven and unstructured environments, such as those designed for wheelchair mobility. The other contribution of this thesis is an integrated software and hardware system for autonomous mobile robot navigation in uneven and unstructured environment. The thesis details the results of extensive experiments in both simulation and real-world, demonstrating the efficacy and efficiency of the system.

Preface

This dissertation is based on the research work conducted in collaboration with multiple researchers at The University of British Columbia under the guidance and supervision of Prof. Clarence W. de Silva.

- Chapter 3 addresses the problem of estimating camera pose relative to a known scene, given a single RGB image. This chapter is based the following published work: Lili Meng, Jianhui Chen, Frederick Tung, James J. Little, and Clarence W. de Silva. "Exploiting Random RGB and Sparse Features for Camera Pose Estimation." In 27th British Machine Vision Conference(BMVC), 2016.

- Chapter 4 addresses the problem of camera pose learning based on backtracking regression forests. This chapter is based on the accepted publication: Lili Meng, Jianhui Chen, Frederick Tung, James J. Little, Julien Valentin and Clarence W. de Silva. "Backtracking Regression Forests for Accurate Camera Relocalization." In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems(IROS), 2017.

- Chapter 5 addresses exploiting points and lines in regression forests for RGB-D camera relocalization. This chapter is based on the following under reviewed submission: 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." IEEE International Conference on Robotics and Automation (ICRA), 2018.

- Chapter 6 addresses the problem of mobile autonomous navigation in uneven

and unstructured world with localization learning and other methods. This chapter is based on the following accepted publication: Chaoqun Wang*, Lili Meng*, Sizhen She, Ian M. Mitchell, Teng Li, Frederick Tung, Weiwei Wan, Max. Q.-H. Meng, and Clarence W. de Silva. "Autonomous Mobile Robot Navigation in Uneven and Unstructured Indoor Environments." In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems(IROS), 2017. (*Indicates equal contribution)

For the BMVC, IROS on camera relocalization, and ICRA paper/submission, the author identified the problem, formulated the solution, designed and implemented most of the experiments. For the IROS paper on autonomous navigation, the author is equally contributed. For the BMVC paper, the author implemented the camera relocalization using a single RGB image while the implementation on the sports camera calibration was mostly done by Jianhui Chen, which is not included in this thesis. For the IROS backtracking regression forests and journal paper, Jianhui Chen contributed ideas in discussion and provided suggestions on the coding, while the author implemented all the experiments. For all the BMVC, IROS paper and ICRA submission on camera relocalization, Dr. Frederick Tung and Prof. James J. Little helped on ideas discussion and the manuscripts edition. For the autonomous navigation paper, Chaoqun Wang helped on the system experiment, and designed the global path planning algorithm, which is not included in the thesis. Sizhen She helps on the platform setup at the preliminary stage for the Pioneer robot which is not included in thesis. Teng Li helped part of the robot autonomous navigation environment setup in the Industrial Automation Lab, especially in the period lab relocation in the ICICS building.

Prof. Clarence De Silva contributed in at all stages of the projects including research proposal, problem identification, experimental equipment, infrastructure and environments developments, research funding, supervision, guidance, thesis writing and so on. Prof. James J. Little contributed ideas and provided feedback for the camera localization learning. Prof. Ian M. Mitchell contributed ideas, discussions and manuscript revision on the autonomous navigation IROS paper. Weiwei Wan contributed the discussion and manuscript revision on the IROS autonomous navigation paper. Julien Valentin helped on the discussion on running speed of

regression forests and suggested to add other experiments in IROS backtracking regression forests paper and ICRA submission.

Table of Contents

List of Tables

List of Figures

Nomenclature

(μ, Σ)	mean and covariance
$\hat{\mathbf{y}}_{\mathbf{p}}$	estimated world coordinate associated to image location $\mathbf{p}$
$\Lambda(\mathtt{S})$	covariance matrix of the labels in $\mathtt{S}$
$\mathbf{p}$	a 2D location in image coordinates
$\mathbf{P}^c$	a 3D point in camera coordinates
$\mathbf{P}^w$	a 3D point in world coordinates
$\mathbf{t}$	translation vector
$\mathbf{v}$	a feature vector
$\mathtt{D}(\mathbf{p})$	depth in image location $\mathbf{p}$
$\mathtt{I}(\mathbf{p}, c)$	pixel lookup at location $\mathbf{p}$, channel c
$\mathtt{K}$	camera matrix
$\mathtt{R}$	rotation matrix
Θ	space of split parameters
θ_j	split parameter in tree node j
$\{\mathbf{p}, \mathbf{m}_{\mathbf{p}}\}$	a 2D image location and its associated 3D world location
$f(.)$	a regression function

$h(\mathbf{p};\theta)$ decision tree binary feature indication function

$H(S)$ entropy of examples in set S

I_j Information in tree node j

j tree node index

S_j set of examples in tree node j

S_j^L set of examples in the left child of tree node j

S_j^R set of examples in the right child of tree node j

List of Abbreviations

AMCL adaptive Monte Carlo localization
AR augmented reality
ATE absolute trajectory error
BRF balanced regression forests
BTRF backtracking regression forests
CNN convolutional neural network
DOF degree of freedom
EPnP efficient perspective-n point (camera pose estimation)
GPS global positioning system
LSD Line Segment Detector
LSTM long short-term memory
NN nearest neighbor
ORB Oriented FAST and rotated BRIEF
PnP perspecative-n point (camera pose estimation)
PLForests point line regression forests
RANSAC ransom sample consensus
RFs random forests
RMSE root mean squared error
ROS robot operating system
SCRF scene coordinate regression forest
SfM structure from motion
SIFT scale invariant feature transformation
SLAM simultaneous localization and mapping
SURF speeded up robust features

VR virtual reality
WHT Walsh-Hadamard transform

Acknowledgments

It would be an impossible task to list everyone who has contributed to my research and study during my PhD, but I am sincerely grateful to everyone who has helped me in various of ways.

I am very fortunate to be in The Institute for Computing, Information and Cognitive Systems (ICICS) which is a multidisciplinary research institute that promotes collaborative research in advanced technologies systems. I am exposed both to mechatronics and computer science here, and most importantly, the great people in both fields.

I'd like to express my sincere gratitude to my supervisor Prof. Clarence de Silva for providing me the research freedom and supporting my research in various of ways all these years in UBC. I'd like to express my deep gratitude to Prof. Jim Little who provides many delightful encouragements, insightful views and inspirational discussions not only for my doctoral research but also my view on robotics. I'd like to thank Prof. Bob Woodham who introduced me to the field of Computer Vision in his vivid computer vision course with candies (students who ask and answer questions in his class could get candies!). I'd like to thank Prof. Ian Mitchell for introducing me to the computational robotics field, for his kind support and numerous of discussions on various of ideas in robotics.

I would like to express my sincere gratitude to Jianhui Chen and Frederich Tung on the numerous of inspirational and fruitful discussions, patient coding reviews, detailed paper writing suggestions and warm friendship. I would like to express my gratitude to Chaoqun Wang for a great collaborator on autonomous navigation and I learned a lot of path planning from him. Thanks Sizhen She a lot for helping setup the robot platform together. Thanks a lot Victor Gan, Neil Traft and Ankur

Gupta on the discussion of various questions on robotics with long distance running. Julieta Martinez and Alireza Shafaei, thank you so much for your insightful views and helpful discussion on many computer vision problems.

Thanks Ji Zhang and Weiwei Wan for contributing fruitful discussion and providing valuable suggestions on visual odometry and paper writing.

I would like to express my sincere gratitude to Yu Du, Yunfei Zhang, Teng Li, Min Xia, Yanjun Wang, Haoxiang Lang Muhammad Tufail, Jiahong Chen, Simon Gong and all the labmates for the great time and various of support.

Lastly, and most importantly, I wish to acknowledge my gratitude for my family and Jesus Christ. I deeply thank my family for their unconditional love, patience, and support. Lord Jesus, thank you so much for coming into my life and being in me, for lighting my life and career, for filling me with your love, peace and grace, and for bringing all these amazing people in my life!

Chapter 1

Introduction

We keep moving forward, opening new doors, and doing new things, because we're curious and curiosity keeps leading us down new paths. — Walt Disney

1.1 Research motivation

1.1.1 Image-based localization methods

With the advance of machine learning methods, the field of robotics is undergoing a transformation from sensing and executing fixed tasks in fixed environments to exhibiting increased adaptability for autonomous operation in dynamic, unknown, unstructured and flexible environments. In order to operate autonomously in unstructured environments, the robot must be capable of interacting with its environment in an intelligent way, which implies that an autonomous robot must be able to correctly and comprehensively sense the environment and then perform actions based on the sensed information. Sensing is the preliminary step for all tasks in a sense-plan-act scheme. Localization, which is the problem of determining the location and orientation of a robot relative to a known environment, is a basic perceptual problem for mobile robots [101].

Image-based localization (also referred to as camera relocalization in literature) is one of the fundamental problems in robotics, computer vision and virtual reality.

It is the problem of estimating the position and orientation of the camera relative a known scene, given a single RGB/RGB-D image. The global positioning system (GPS), which is widely used in outdoor environments, is not available in indoor environments and with the presence of skyscrapers and other interfering structures in outdoor environments. Moreover, the GPS may have approximately $2-10$ meters of error. As a result, it poses serious problem for autonomous robots, such as mobile robots, self-driving cars and unmanned aerial vehicles. Light-weight and affordable visual sensors are an appealing alternative to GPS and other expensive sensors. Recent consumer robotics products such as iRobot 980 and Dyson 360 eyes are already equipped with visual simultaneous localization and mapping (SLAM) techniques, enabling them to effectively navigate in complex environments. Meanwhile, in order to insert virtual objects in an image sequence (i.e., in augmented reality applications), camera poses have to be estimated in a visually acceptable way.

Camera relocalization with a single RGB-D image or RGB image belongs to the subject of global localization in robotics research and development. However, global localization may also use other sensors such as laser or sonar. Local and global localization are two types of localization [101]. Local localization concerns estimation of the relative robot pose from a previous state. It aims to compensate for robot motion noise from sensors (e.g., wheel odometry). In global localization, the initial pose and previous motion information are unknown. Global localization is more difficult than local localization as there is no previous pose/motion information available and the environment can be extensive and have repeatable objects. It plays a critical role as it can help solve the kidnapped robot problem and allow the robot to recover from severe pose errors, which is essential for truly autonomous robots.

The image-based camera localization with machine learning methods is an attractive field for advancing the state-of-the-art of intelligent robotics because of the low cost of vision sensors and scalability of machine learning methods. Random forests (RFs) are popular and successful machine learning method because of their speed, robustness and learning abilities [21]. RFs have been used for camera relocalization by Shotton *et al*.[93]. In Scene Coordinate Regression Forests (SCRF), a regression forest is employed, which is trained to have capabilities of predicting

any pixel's correspondence to its 3D point position in the scene's world coordinate. Then the camera pose is estimated using a robust optimization scheme with predicted world points. Methods based on deep learning also have been used in camera re-localization. For example, PoseNet [53] trains a convolutional neural network to regress the 6-DOF camera pose from a single RGB image in real-time. However, the accuracy is much worse than the methods based on random forests. Therefore, this thesis focuses on a machine learning method random forests for image-based camera relocalization.

1.1.2 Mobile robot autonomous navigation

With a rapidly aging population and a shortage of workforce, autonomous robots such as self-driving cars, autonomous delivery robots, smart wheelchairs, and home assistant robots play a more an more important role in our life. According to the UNs population projections, the older population (aged 65 and over) in 2050 will be about 16.7 percent of the total population in many parts of the world, which is more than double that of 2015 [42]. The "old-age dependency ratio", which is the ratio of old people to those of working age, will grow even faster. In 2015 the world had 15 persons aged 65 and over for every 100 adults between the ages of 25 and 64. By 2050 that number is projected to have risen to 30 [42]. A significant portion of the aging population is expected to need physical and cognitive assistance. Confidence in the ability to undertake various tasks is core to one's psychological functioning [69], and a greater sense of control of life can be positively correlated with a reduced mortality rate. Yet, the shortage of space and staff in nursing homes and other care facilities is already causing problems. Therefore, there exists an urgent need to develop robot-assisted systems. A great deal of attention and research is directed to assistive systems aimed at promoting aging-in-place, thereby facilitating living independently as long as possible.

Significant progress has been achieved in recent decades in advancing the state-of-the-art of mobile robot technologies. These robots are operating more and more in unknown and unstructured environments, which requires a high degree of flexibility, perception, motion and control. Companies such as Google and Uber are developing advanced self-driving cars and expecting to present them to market in

the next few years. Various mobile robots are roaming in factories and warehouses to automate the production lines and inventory, saving workers from walking daily marathons [25]. Robot vacuum cleaners such as Roomba and Dyson360 eyes are moving around in the house to help clean the floors. Personal robots such as PR2 [46, 67] and Care-O-bot [82] have demonstrated the ability to perform a variety of integrated tasks such as long-distance navigation and complex manipulation.

Mobile robots that work safely and autonomously in uneven and unstructured environments still pose great challenges, such as navigation through sloped areas instead of staircases. However, little work has focused on an integrated system of autonomous navigation in sloped and unstructured indoor areas, which are common in many modern buildings.

In this backdrop, the present thesis focuses on an integrated software and hardware system for autonomous mobile robot navigation in indoor environments that are designed for and shared with people. It is believed that a functioning robot system of this type will inspire and benefit technology development and the general public.

1.2 Contributions

The main contributions of this thesis can be summarized as follows:

- Three camera relocalization methods are developed in this thesis. The first one is to use just RGB images in the testing stage to estimate camera pose. The second one contains a sample balanced decision-tree objective function and a backtracking search scheme. The third one employs both point and line features to estimate the camera pose. The state-of-the-art performances of the developed methods are demonstrated on publicly available camera relocalization benchmarks against several strong baselines.

- For autonomous navigation, an integrated software and hardware architecture for autonomous mobile robot navigation in 3D uneven and unstructured indoor environments is presented. The integrated system is evaluated both in simulation and on real scenarios, demonstrating the efficacy of our methods, providing some insight for more autonomous mobile robots and wheelchairs working around us.

1.3 Thesis outline

The thesis is organized into seven chapters. It starts with an introduction to the importance of camera relocalization and motivation for image-based localization learning methods. Chapter 2 introduces related work and background on the regression forests and camera pose estimation that are used in our work. Chapter 3 presents the proposed camera relocalization method, which only uses a single RGB image in testing. Chapter 4 discusses the problem of accurate camera relocalization with the developed sample-balance scheme and backtracking technique. Chapter 5 presents the developed method that uses both point and line features in camera pose estimation. Chapter 6 presents an integrated system for autonomous robot navigation. Chapter 7 provides conclusions on the topic of image-based localization learning, and presents several promising avenues for possible future work.

Chapter 2

Background

If I have seen farther it is by standing on the shoulders of Giants.
— Isaac Newton

This chapter presents an overview of previous work on image-based camera re-localization and autonomous navigation of mobile robots. Moreover, some background is provided of the random forests method and camera pose estimation which represents the foundation for the localization method that is developed in the present work.

2.1 Related work

2.1.1 Image-based camera relocalization

Image-based camera relocalization has been widely studied in the context of large scale global localization [53, 75], recovery from tracking failure [33, 54], loop closure detection in visual SLAM [111], global localization in mobile robotics [23, 89], and sports camera calibration [17]. Moreover, place recognition[4, 30, 99, 100] could be seen as a special case of camera relocalization with the focus on recognizing whether the places have been visited or not, without providing an accurate 6D pose estimation. Approaches based on local features, keyframes, random forests and deep learning are four categories of camera pose estimation. Other successful variants and hybrid methods also [83] exist.

Local feature based approaches

Local feature based methods [75, 89] usually match the descriptors extracted from the incoming frame and the descriptors stored in the database. Then the combination of the perspective-three-point [31] and RANSAC [28] is usually employed to determine camera poses. The local features (e.g. commonly used SIFT [64] and SURF [5]), can represent the image local properties, so they are more robust to viewpoint changes as long as a sufficient number of keypoints can be recognized. However, these methods have to store a large database of descriptors and require efficient approximate nearest neighbor search methods for feature matching. Much recent work has focused on efficiency [87], scalability [88], and learning of feature detection, description, and matching [60].

Keyframe based approaches

Methods based on keyframes [27, 32] hypothesize an approximate camera pose by computing whole-image similarity between a query image and keyframes. For example, randomized ferns encode an RGB-D image as a string of binary codes. They have been used to recover from tracking failure [33] and detect loop closure in SLAM [111]. Place recognition [30, 100] uses keyframe based camera relocalization [72, 110]. First, the incoming frame is encoded by bag of visual words [96], and the keyframe database is queried for the most similar keyframe [72]. Second, the sparse features are extracted from the incoming frame and the most similar keyframe in the database, and the camera pose is estimated through the 2D-3D correspondence. However, these methods provide inaccurate matches when the query frame is significantly different from the stored keyframes.

Random forests based approaches

Approaches based on random forests for image-based camera relocalization have gained interest since the introduction of scene coordinate regression forests (SCRF) [93]. Random forests are used as regression for camera relocalization so we also refer to them as regression forests. These approaches first employ a regression forest to learn an estimation of 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. The features in random forests are based on fast random pixel comparisons. A similar technique has been developed by Lepetit and Fua [60] for keypoint recognition and image registration. These simple and fast pixel comparison features suffice to work for detection and tracking even under large perspective and scale variations.

Random forests based methods do not need to compute ad hoc descriptors and search for nearest-neighbors, which are time-consuming steps common in local feature based and key-frame based methods by shifting much computational burden to the training stage. They have been used in many applications. For example, Shotton *et al.*[94] use the random pixel comparison feature from the depth image in the random forests for per-pixel classification with the application of classifying which body parts the pixel belongs. Both [60] and [94] use synthetic images under different views for data augmentation in the training set. In contrast to [60, 94] using random features in random forests as a classification, scene coordinate regression forests (SCRF) formulate the 2D-3D correspondence search problem as a regression problem which is solved by random forests. 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, Guzman *et al.*[38] proposed a hybrid discriminative-generative learning architecture to choose the optimal camera poses in SCRF. To improve the camera relocation accuracy, Valentin *et al.*[102] exploit uncertainty from regression forests by using a Gaussian mixture model in leaf nodes.

However, these methods need RGB-D images for both training and test, constraining their applications. [6, 70, 103] extended the random forest based methods to use RGB images at test time, while depth images are still needed to get the ground truth labels. Valentin *et al.*[103] proposed multiple accurate initial solutions and defined a navigational structure over the solution space that can be used for efficient gradient-free local search. However, their method needs an explicit 3D model as the auxiliary of the RGB image at the testing stage. We have proposed a method that only require RGB pixel comparison features in [70], eliminating the dependency on the depth image at test time. Furthermore, the method integrates random RGB features and sparse feature matching in an efficient and accurate way,

allowing the method to be applied to highly dynamic scenes. Cavallari *et al*.[14] have extended SCRF for online camera relocalization by adapting a pre-trained forest to a new scene on the fly.

Deep learning based approaches

Deep learning [59] has led to rapid progress and impressive performance on a variety of computer vision tasks, such as visual recognition [41, 95] and object detection [80]. There is emerging work [51, 53, 108] for camera re-localization. For example, PoseNet [53] trains a convolutional neural network to regress the 6 degree-of-freedom (DOF) camera pose from a single RGB image in real time. Bayesian PoseNet [51] applies an uncertainty framework to the PoseNet by averaging Monte Carlo dropout samples from the posterior Bernoulli distribution of the Bayesian ConvNet's weights. Besides the camera pose, it also estimates the model's relocalization uncertainty. [116] uses CNNs to regress pixels world coordinates, and it has larger camera relocalization error but better performance on predicting world coordinates of pixels compared to SCRF, which indicates that better prediction on world coordinate does not necessarily lead to better final camera pose estimation due to the outlier removal capability of camera pose optimation using Preemptive RANSAC. [62] presents a dual-stream CNN with both color images and depth images as the network inputs, but still has almost an order of magnitude of error compared to methods based on regression forests. To improve the PoseNet accuracy, geometric loss functions is explored in [52] and LSTM units are used on the CNN output to capture the contextual information. However, these methods based on deep learning have much lower overall accuracy compared with methods based on random forests in indoor scenes and methods based on local features in outdoor scenes. Moreover, high-end GPU dependency and high power consumption make these deep learning based methods less favorable for energy sensitive applications such as mobile robots.

2.2 Random forests

Random forests (RFs) are one of the most popular machine learning tools because of their speed, robustness and generalization [9, 19]. RFs are broadly applied in

many domains, including natural language processing tasks such as language modeling [117] and financial tasks such as stock price prediction [77] and computer vision tasks such as human pose recognition [94]. A random forest is an ensemble of decision trees. In a random forest, a fraction of data and a particular number of features are selected at random to train a decision tree. Now a single decision tree is discussed, followed by the tree ensemble method.

2.2.1 Decision tree

A decision tree is a binary tree data structure made of a collection of nodes organized in a hierarchical way as shown in Fig. 2.1 (a). The decision tree solves a complex problem by running a sequence of simpler tests. For a given input object, a decision tree estimates an unknown property of the object by asking successive questions about its known properties. Which question to ask next depends on the answer of the previous question and this relationship is represented graphically as a path through the tree [19]. The final answer is stored in the leaf node which is the terminal node of the path that the object follows. Consider the simple example of deciding whether a photo represents an indoor scene or an outdoor scene as illustrated in Fig. 2.1 (b). Suppose that only the image pixel information is available. Then, the questions can start from whether the top of the image is blue or not. If the answer is yes, then that part might be the sky, and the whole image data is then sent to the right child. Based on this answer, another question can be asked, for instance if the bottom part is green. If it is, it might be the grass, which indicates that the photo has high probability of showing outdoor scene. Successive questions and answers are followed up to the leaf node. The more questions asked and answered, the higher the confidence on the final prediction.

Data Points and Features: In decision trees, a generic data point is denoted by a vector $\mathbf{v} = (x_1, x_2, ..., x_d) \in R^d$, where the components x_i represent an individual measurable attribute of a data point, called feature. The number of features depends on the property of the data point. In theory, the dimensionality of d can be very large, or even infinite. In practice, only a small important portion of d is used as needed.

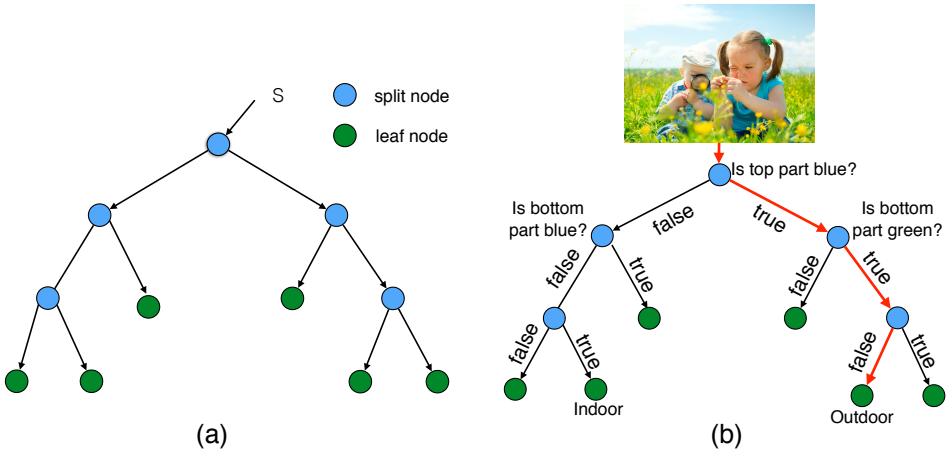


Figure 2.1: Decision tree example. Best viewed in color. (a) A tree is a set of split nodes and leaf nodes organized in a hierarchical way. Split nodes are denoted with blue circles and leaf nodes with green circles. (b) Each split node of a decision tree is a split (or test) function to be applied to the incoming data. Each leaf node stores the final answer (predictor). An example is presented here on using a decision tree to classify whether an image represents an indoor scene or an outdoor scene.

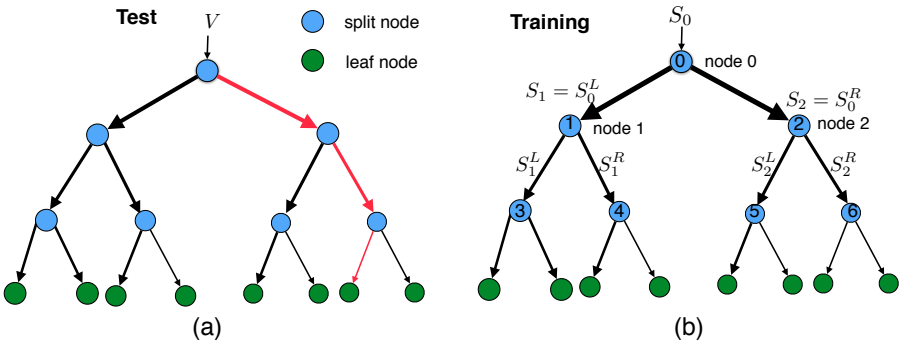


Figure 2.2: Decision tree test and training process Best viewed in color. (a) In the test, a split node applies a test to the input data $\mathbf{v}$ and sends it to the appropriate child. The process is proceeded until a leaf node is reached (red path). (b) Training a decision tree is sending the entire training set S_0 into the tree and optimizing the parameters of the split nodes so as to optimize a chosen energy function.

Training Points and Training Sets: A training point is a data point with a known ground truth label, and these feature-label correspondence can be used to compute the tree parameters. A training set $\mathbf{S}_0$ is a collection of different training data points as shown in Fig. 2.2 (b).

Training a Decision Tree: The training entails finding the parameters of split functions stored in the split nodes by optimizing a specified objective function given the feature-label correspondences. The process proceeds in a greedy manner. At each node j, depending on the subset of the incoming training set S, the function that best splits S_j into S_j^L and S_j^R is learned. Fig. 2.1 (b) shows the training process. S_j represents the subset of incoming training data before split, while S_j^L and S_j^R represent the subset after the split at the node j. $S_j^L = S_{2j+1}$ and S_j=$S_j^L \cup S_j^R$. The selection of the "best" split parameters can be formulated as the maximization of an objective function I_j at the split node:

$$\theta_j^* = \arg\max_{\theta \in \Theta} I_j(S_j, \theta_j) \tag{2.1}$$

where Θ represents the space of all split parameters.

Given the set S_j and the split parameters θ^j, the corresponding left and right sets are uniquely determined as:

$$\begin{cases} S_j^L = ((v,\cdot) \in S_j | h(v,\theta_j) = 0) \\ S_j^R = ((v,\cdot) \in S_j | h(v,\theta_j) = 1) \end{cases} \tag{2.2}$$

where $(v,)$ denotes the feature-label pair, and $\cdot$ can represent either continuous labels for the regression problem or the discrete labels for the classification problem.

The objective function I_j is essential in constructing a decision tree that will perform the desired task. In fact, the result of the optimization problem in Eq. 2.1 determines the parameters of the split functions, which, in turn, determine the path followed by a data point. The split functions determine the prediction behavior of a decision tree. Here a generic information gain associated with a tree split node is used as the objective function, which is defined as the reduction in uncertainty achieved by splitting the training data S_j that arrives at the node j into multiple child subsets.

$$I_j = H(S_j) - \sum_{i \in \{L,R\}} \frac{|S_j^i|}{S_j} H(S_j^i) \tag{2.3}$$

where H is the entropy.

$$H(S) = -\sum_{c \in C} p(c) log(p(c) \tag{2.4}$$

where $p(c)$ is calculated as the normalized empirical histogram of labels corresponding to the training points in S.

The split procedure described above proceeds recursively to all the newly constructed nodes and the training phase continues until a stopping criterion is met. There are various stopping criteria such as stopping the tree when a maximum number of levels has reached or a node contains too few training points. In the leaf node, the statistic of labels are stored to predict the unknown data points. When the training phase is done, we can get the split functions associated with each internal node and label distributions in each leaf node.

Testing of a Decision Tree In the testing stage, the new data point v travels the decision tree from the root node to a particular leaf node using the splitting values in the tree (see Fig 2.2 (a)). The leaf node predicts a label for the input data point v.

2.2.2 The randomness model

Randomness is injected into the decision trees at the training stage, while testing is almost always considered to be deterministic [19]. Random training set sampling [8, 9] and randomized node optimization [3, 44] are two popular methods for injecting randomness. A short introduction is provided here, and detailed explanations are found in [3, 8, 9, 22, 44, 63] for more detailed explanation. Random training set sampling (e.g. bagging) concerns training each tree in a forest on different training subset, sampled at random from the same labeled database. It can help improve generalization. Randomized node optimization concerns selecting a small random set of split node parameters rather than the entire set. Random training set splitting

and randomized node optimization could also be used together.

2.2.3 Random forest ensemble

Because a single tree has weak prediction capabilities, a combination of trees (i.e., a forest) is used in practice. A random forest $F = \{F_1, F_2, ..., T_N\}$ is an ensemble of N randomly and independently trained decision trees. The independence between these trees lead to de-correlation between the individual tree predictions, resulting in better generalization and robustness compared with a single tree. In the test phase, each test point v is simultaneously pushed through all trees until it reaches the leaf node. Combining all tree predictions into a single forest prediction can be done by a simple averaging scheme.

$$p(c|v) = \frac{1}{N}\sum_{k=1}^{N} p_k(c|v) \tag{2.5}$$

where $p_k(c|v)$ denotes the posterior distribution of the kth tree.

Random forests used as regression is referred as regression forests.

2.3 Camera pose estimation

2.3.1 RGB camera pose estimation

An RGB camera projects a 3D point $\mathbf{P}^w = [X, Y, Z, 1]^T$ in world coordinate to a 2D point $\mathbf{p} = [x, y, w]^T$ in image coordinate. Here homogeneous coordinates are used, which are suitable for projective geometry. The perspective projection is:

$$\mathbf{p} = \mathtt{K}[\mathtt{R}|\mathbf{t}]\mathbf{P}^w, \tag{2.6}$$

where $\mathtt{K}$ is a 3×3 camera matrix, $\mathtt{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector. In the camera relocalization problem, the camera matrix is known from camera calibration and its value is fixed. The goal is to estimate the rigid transformation $[\mathtt{R}|\mathbf{t}]$ from 2D-3D correspondences $\{\mathbf{p} \leftrightarrow \mathbf{P}\}$. Because the freedom of rigid transformation is 6 and each correspondence provides 2 constraints (one in the image X coordinate and the other in the image Y coordinate), at least three

correspondences are required to solve this problem. To robustly solve this problem, Lepetit [61] *et al*.have developed the EPnP method which requires at least 4 correspondences. This method has been implemented in OpenCV [7] library.

2.3.2 RGB-D camera pose estimation

In RGB-D cameras, an image location $\mathbf{p} = [x, y, w]$ has a depth d in the camera coordinate. So the back-projection is

$$\mathbf{P}^c = \mathtt{K}^{-1}\mathbf{p}, \text{subject to } \mathbf{P}^c_z = d \tag{2.7}$$

where $\mathbf{P}^c$ is a 3D point in the camera coordinate. As a result, the camera relocalization is to estimate the rigid transform from world coordinates to camera coordinates:

$$\mathbf{P}^c = [\mathtt{R}|\mathbf{t}]\mathbf{P}^w. \tag{2.8}$$

The constraint is 3D-3D correspondences $\{\mathbf{P}^c \leftrightarrow \mathbf{P}\}$. Given at least 3 correspondences, the Karbsh algorithm [49] can robustly solve this problem with a closed form solution.

In practice, the correspondences have outliers because of the errors of measurement of sensors and predictions from machine learning method. The RANSAC method is used to remove these outliers and estimate the camera pose in an iterative way.

Chapter 3

Image-based localization using regression forests and keyframe pose refinement

This chapter addresses the problem of estimating the camera pose relative to a known scene, given a single incoming RGB image. The recent advances in scene coordinate regression forests for camera relocalization in RGB-D images are extended to use RGB features, enabling camera relocalization from a single RGB image. Furthermore, regression forests and sparse feature matching are integrated in an efficient and accurate way. The developed method is evaluated using both small scale and large scale publicly available datasets with challenging camera poses against several strong baselines. Experimental results demonstrate the efficacy of the developed approach, showing superior or on-par performance with several strong baselines.

3.1 Introduction

The present work is mainly inspired by recent advances in the methods based on scene coordinate regression forests (SCRF) [38, 93, 102] for camera relocalization. The present method also benefits from privilege learning [18, 90, 104] in which additional information only exists at the training stage. The SCRF-based methods

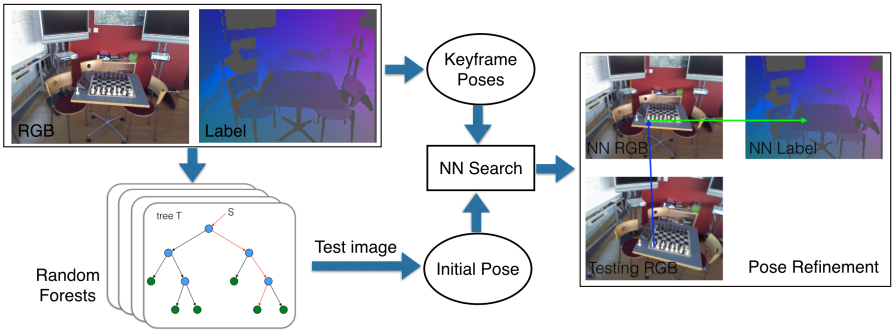


Figure 3.1: Camera pose estimation pipeline. During training, the scene information is encoded in a random forest. In the testing stage, an initial camera pose is estimated using the predictions from the random forest with real-time response. Then, the initial camera pose is used to query a nearest neighbor (NN) image from keyframes. Finally, the camera pose is refined by sparse feature matching between the test image and the NN image. In our method, the labels can constitute any information (e.g., scene coordinate positions) associated with pixel locations.

use an efficient regression forest to guide the camera pose optimization using RGB-D images, achieving high accuracy. In these SCRF-based methods, much of the computational burden is shifted to a training phase, while the test phase is very efficient. The method developed in the present thesis learns from RGB-D images during training but does not require depth images at test time, enabling it to be more accessible for end users.

Fig.3.1 illustrates the present pipeline. A regression forest is trained using RGB images and pixel-wise labels. In the testing stage, an initial camera pose is estimated using predicted labels from the random forest. The accuracy of the camera pose is refined by sparse feature matching.

The main contributions of this chapter are:

- Extend the SCRF-based methods to only use RGB features (without depth) in the testing stage.
- Integrate random features and sparse features, ensuring both efficiency and accuracy.

- Evaluate using publicly available datasets against several strong baselines, showing superior or on-par performance.

3.2 Method

In the initial camera pose estimation, the problem is modeled as a structural regression problem:

$$\hat{\mathbf{y}}_{\mathbf{p}} = f(\mathtt{I}, \mathbf{p}|\theta) \tag{3.1}$$

where $\mathtt{I}$ is an RGB image, $\mathbf{p} \in \mathbb{R}^2$ is a 2D pixel location and θ contains model parameters. In the training stage, $\{\mathbf{p}, \mathbf{m}_{\mathbf{p}}\}$ represent the paired training data. The label $\mathbf{m}_{\mathbf{p}}$ can represent any information associated with that pixel. For example, it is the world coordinate in camera re-localization. In the testing stage, $\hat{\mathbf{y}}_{\mathbf{p}}$ denotes the prediction associated with pixel $\mathbf{p}$.

Random regression forests [22] are chosen here to implement the regression function as it naturally handles structure regression, while being robust and reasonably easy to train.

3.2.1 Random RGB features and labels

Random RGB Features: Here features based on pairwise pixel comparison are used as in [60, 93]. At a given 2D pixel location $\mathbf{p}$ of an image, the feature $f_\phi(\mathbf{p})$ computes the color intensity difference between the pixel $\mathbf{p}$ in a random color channel c_1 and the pixel with a 2D offset δ in a random color channel c_2:

$$f_\phi(\mathbf{p}) = \mathtt{I}(\mathbf{p}, c_1) - \mathtt{I}(\mathbf{p} + \delta, c_2) \tag{3.2}$$

where δ is a 2D offset and $\mathtt{I}(\mathbf{p}, c)$ represents an RGB pixel lookup in channel c. The ϕ contains feature response parameters $\{\delta, c_1, c_2\}$. Pixels outside the image boundary are marked and not used as samples for both training and testing. A significant difference here with the depth-adaptive feature used by Shotton *et al.*[93] is that the present feature does not require depth information.

Fig. 3.2 illustrates two random RGB features at different pixel locations p_1 and p_2. Feature f_{ϕ_1} looks rightwards and Eq.3.2 will give a large response for 3.2(a) and small response for 3.2(b) according to different color changes. Feature f_{ϕ_2} looks

Figure 3.2: Random RGB pixel comparison features. The red star represents the pixel p being computed. The distance between the red star and the blue circle represents the pixel offset as defined in Eq. 3.2. In (a), the two example features at image position p_1 and p_2 give a large color difference response. In (b), the two features at the same image locations in a different image give a much smaller response compared with (a).

upwards and Eq.3.2 will give a large response for 3.2(a) and a small response for 3.2(b). In this way, the pixel comparison feature distinguishes f_{ϕ_1} from f_{ϕ_2}.

Individually each feature only provides a weak description about which part of the scene the pixel belongs to. But with thousands of such features per image in combination in a random forest, these features are accurate enough to describe all the trained images. These features are very efficient. Each feature needs only 2 arithmetic operations without pre-processing.

Labels: For scenarios where RGB-D images are available in the training stage, the training set contains sequences of RGB-D frames with associated known camera poses $\mathtt{P}$ which includes 3×3 rotation matrix R and 3×1 translation vector T from the camera coordinate to world coordinate as shown in Eq. 3.3.

$$\mathtt{P} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \in SE(3) \tag{3.3}$$

The $3D$ point $\mathbf{x}$ in camera coordinate of the corresponding pixel $\mathbf{p}$ is computed by back-projecting the depth image pixels:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (u - c_x) \times d / f_x \\ (v - c_y) \times d / f_y \\ d \end{bmatrix} \tag{3.4}$$

where $[u, v]^T$ is the pixel $\mathbf{p}$ position in image plane, and $[x, y, z]^T$ is the point position in camera coordinate, $[c_x, c_y]^T$ and $[f_x, f_y]^T$ are the camera principal point and focal length respectively. $d_i = depth_{image[v,u]}/factor$, and $depth_{image[v,u]}$ is the measured depth value at image point $[v, u]$. The factor is RGB-D camera depth factor, usually $factor = 5000$ for the $16 - bit$ PNG files.

The scene's world coordinate position $\mathbf{m}$ of the corresponding pixel $\mathbf{p}$ could be computed by:

$$\mathbf{m} = \mathtt{P}\mathbf{x} \tag{3.5}$$

The associated camera pose P for each RGB-D image in the training data can be obtained through camera tracking method KinectFusion[73] or visual SLAM method [58].

3.2.2 Random forests for 2D-3D correspondence regression

A regression forest is an ensemble of T independently trained decision trees. Each decision tree is a binary-tree-structured regressor consisting of decision (or split) nodes and prediction (or leaf) nodes. We grow the regression forest using greedy forest training [22].

Weak Learner Model: Each split node i represents a 'weak learner' parameterized by $\theta_i = \{\phi_i, \tau_i\}$ that splits the data to the left or the right child. In the training phase as shown in Fig. 3.3, a set S'_0 of labeled sample pixels $(\mathbf{p}, \mathbf{m})$ is randomly chosen for each tree, where $\mathbf{m}$ is the 3D world coordinate label of pixel $\mathbf{p}$. 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

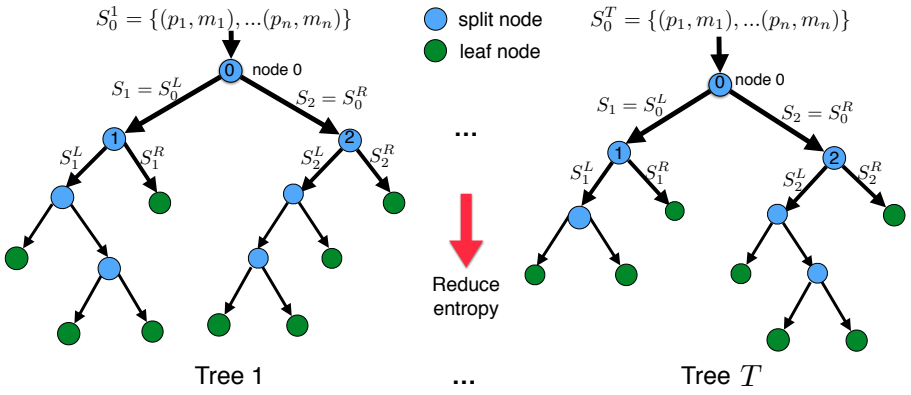


Figure 3.3: Training random forests for 2D-3D coordinate correspondence regression. A set S_0^t of labeled sample pixels $(\mathbf{p}, \mathbf{m})$ is randomly chosen from the entire set $S_0 = [S_0^1, \cdots, S_0^T]$ for each tree, where $\mathbf{m}$ is the 3D world coordinate label of pixel $\mathbf{p}$. The tree grows recursively from the root node to the leaf node. The goal is to optimize the parameters of the tree split nodes. Here spatial-variance is used to represent entropy.

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:

$$h(\mathbf{p}; \theta_i) = \begin{cases} 0, & \text{if } f_{\phi_i}(\mathbf{p}) \leq \tau_i, \quad \text{then go to the left subset } S_i^L. \\ 1, & \text{if } f_{\phi_i}(\mathbf{p}) > \tau_i, \quad \text{then go to the right subset } S_i^R. \end{cases} \tag{3.6}$$

Here, τ_i is a threshold on random feature $f_{\phi_i}(\mathbf{p})$.

At each split node i, for the incoming training set S_i, the training comprises learning of the split parameter θ_i that best splits the left child subset S_i^L and the right child subset S_i^R. The selection of the "best" split parameters can be formulated as the maximization of an information gain I_i:

$$\theta_i^* = \arg\max_{\theta_i \in \Theta_i} I_i(S_i, \theta_i) \tag{3.7}$$

$$I_i = V(S_i) - \sum_{j \in \{L,R\}} \frac{|S_i^j(\theta_i)|}{|S_i|} V(S_i^j(\theta_i)) \tag{3.8}$$

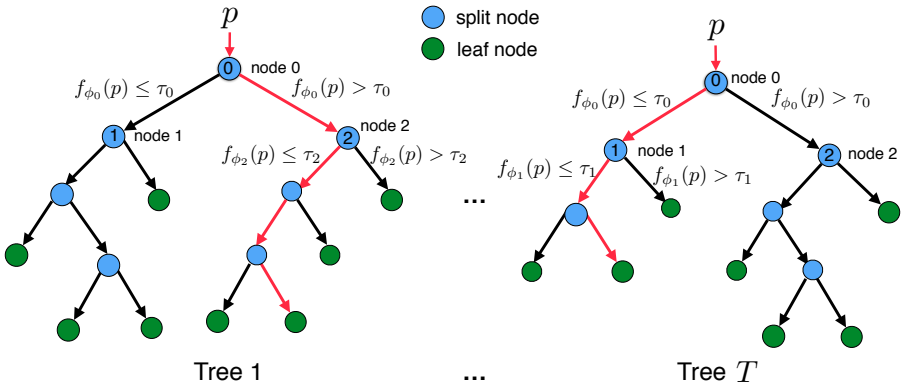


Figure 3.4: Test phase of random forests for 2D-3D correspondence regression. In the testing stage, at each split node i, the feature f_{ϕ_i} is compared with the feature response τ_i to determine whether to go to the left or the right child node. A particular input may go along different paths in different trees as indicated by the red arrows.

$$V(S_i) = \frac{1}{|S_i|} \sum_{(\mathbf{p},\mathbf{m}) \in S_i} ||\mathbf{m} - \bar{\mathbf{m}}||_2^2 \tag{3.9}$$

where $V(S_i)$ is the spatial variance of the labels in S_i, subset S_i^j is conditioned on the split parameter θ_i, and $\bar{\mathbf{m}}$ represents the mean of $\mathbf{m}$ in S_i.

Fig. 3.4 shows the test phrase. At each split node i, the feature f_{ϕ_i} is compared with the feature response τ_i to determine whether to go to the left or right child node. This process proceeds until it reaches the leaf node.

Leaf Prediction: Training terminates when a node reaches a maximum depth D or contains too few examples. In tree t, one leaf node contains a set of samples whose distribution is described by the leaf model parameter θ_f. During the testing stage, the leaf generates an estimated vector:

$$\mathbf{v}_t^* = \arg\max_{\mathbf{v}} p(\mathbf{v}|\theta_f) \tag{3.10}$$

with

$$p(\mathbf{v}|\theta_f) = N(\mathbf{v};\mu,\Sigma) \tag{3.11}$$

where μ,Σ are the mean and covariance of a Gaussian distribution of labels. Besides the pixel location in 3D world coordinate, the present method also stores the color distribution of samples in the leaf node to reduce color ambiguities.

Forest Ensemble: A forest is an ensemble of independently trained decision trees. Since the number of training samples and possible split node tests are large in the present work, building the optimal tree quickly can become intractable. Instead, multiple trees are grown so that each tree yields a different partition of the space. Furthermore, multiple trees can have better generalization in the test phase. Once the trees are built, the 3D world coordinate prediction in the leaf node that has the most similar color distribution with the test pixel is used as the final prediction because the present method stores RGB color distribution in leaf nodes.

3.2.3 Pose refinement

The downside of the RGB random feature is that it is not naturally scale-invariant. A naive way to solve this problem is to train the random forest using image pyramids. However, it was found that the pyramid method was too time-consuming and did not significantly improve the accuracy.

Based on this observation, a hybrid pipeline is designed. First, the 2D-3D correspondence from the above regression forests is used to get an initial camera pose $\mathtt{P}_0$. Then, accurate SIFT features [64] are used to refine the initial camera pose. The second step requires an image from the training set that shares similar camera parameters with the test image. Searching for the nearest neighbor (NN) image whose camera is closest to the location of $\mathtt{P}_0$ while having an orientation difference no greater than a predefined threshold τ_0 (1/4 of the field of view) is done.

The dimension of the camera pose space (which is ≤ 6) is much smaller than the image descriptor space (which can be up to several hundreds [16]). Therefore, the present method is much more efficient than the methods that use bag of words [30, 72]. Once the NN image is found, the camera pose is refined by minimizing the

reprojection error, which is a geometric error corresponding to the image distance between a projected point $\hat{\mathbf{x}}_k$ and a measured one $\mathbf{x}_k$:

$$\mathtt{P}^* = \arg\min_{\mathtt{P}} \sum_k d(\mathbf{x}_k, \hat{\mathbf{x}}_k)^2 = \arg\min_{\mathtt{P}} \sum_k d(\mathbf{x}_k, \mathtt{P}\mathbf{X}_k)^2 \tag{3.12}$$

where P is a 3×4 matrix including the camera intrinsic and extrinsic parameters, $\mathbf{x}_k$ and $\hat{\mathbf{x}}_k$ are the kth observed and projected feature point locations in the image, respectively. $\mathbf{X}_k = [X_k, Y_k, Z_k]$ is the corresponding 3D world coordinate from the NN image.

Here, a pinhole camera model is used in which a scene view is formed by projecting 3D points into the image plane using a perspective transformation [39]. Therefore, each projected feature point $\hat{\mathbf{x}}$ can be obtained by:

$$\hat{\mathbf{x}} \approx \mathtt{P}\mathbf{X} = K[R|T]\mathbf{X} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.13}$$

where X, Y, Z are the coordinates of a 3D point in the world coordinate space, K is a known camera intrinsic parameters, $[R|T]$ is a camera matrix of extrinsic parameters in which R is a 3×3 matrix which represents rotation and T is a 1×3 matrix which represents translation, (c_x, c_y) is a principal point that is usually at the image center, and f_x, f_y are the focal lengths expressed in pixel units. We use $\approx$ to indicate that the image location $\mathbf{x}$ is transformed from a 3×1 homogeneous vector.

Because correspondences contain outliers, Eq.3.12 is optimized using EPnP [61] and RANSAC[28]. The Perspective-n-Point (PnP) is to determine the position and orientation of a camera given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projections. The complexity of EPnP is linear with the number of 2D-3D correspondences .

Algorithm 1 briefly summarizes the proposed method step by step.

Algorithm 1 Camera pose estimation from a single RGB image

Require: A set $\mathscr{S}_{\mathtt{I}} = \{\mathtt{I}_1, \mathtt{I}_2, \cdots \mathtt{I}_n\}$ of images with pixel-wise labels
Require: An RGB image $\mathtt{I}$
Ensure: The camera pose $\mathtt{P}$ of image $\mathtt{I}$
1: $\mathscr{S}_i$ = a set of randomly sampled pixel-wise training samples;
2: **train** a regression forest using $\mathscr{S}_i$; //
3: $\mathtt{P}_0$ = initial camera pose from the regression forest predictions;
4: **search** nearest neighbor image $\mathtt{I}_{nn}$ in $S_{\mathtt{I}}$ using $\mathtt{P}_0$;
5: **match** $\mathtt{I}$ and $\mathtt{I}_{nn}$ to obtain 2D-3D correspondences;
6: **estimate** $\mathtt{P}$ using the 2D-3D correspondences and *solvePnPRansac*; // Sec. 3.2.3
7: **return** $\mathtt{P}$

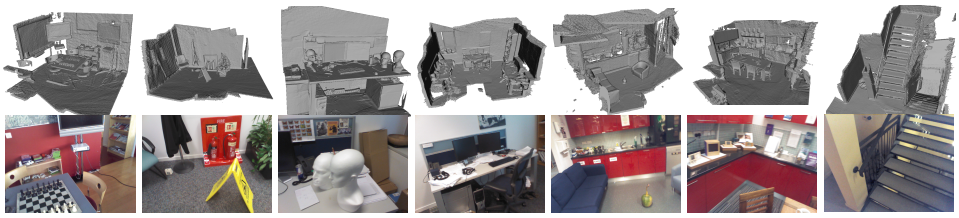
3.3 Experiments

The proposed method is evaluated using both a small scale (about $2-6m^3$) Microsoft 7 *Scenes* dataset and a considerably larger ($14-79m^3$) Stanford 4 *Scenes* dataset.

3.3.1 Camera relocalization on Microsoft 7 *Scenes* dataset

Dataset: The 7 *Scenes* dataset as shown in Fig. 3.5 is from [93] and 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. The ground truth camera pose is obtained from an implementation of the KinectFusion [73] system. The dataset exhibits shape/color ambiguities, specularities and motion blur, which present great challenges for the proposed RGB-only random features.

Baselines and Error Metric: Three strong methods are used as the baselines: SCRF [93], PoseNet [53] and Bayesian PoseNet [51]. SCRF employs a scene coordinate regression forest to guide camera pose optimization using RANSAC with RGB-D images. PoseNet trains a ConvNet as a pose regressor to estimate the 6-DOF pose from a single RGB image. Bayesian PoseNet improved the accuracy of PoseNet through an uncertainty framework by averaging Monte Carlo dropout



(a) (b) (c) (d) (e) (f) (g)

Figure 3.5: Example sequence in Microsoft 7 Scenes dataset. (a) Chess, (b) Fire, (c) Heads, (d) Office, (e) Pumpkin, (f) Red Kitchen, (g) Stairs. The upper row shows the 3D reconstructed scenes and the lower row shows the RGB image example.

samples. The same median translational error and rotational error are used as in [51, 53] for fair comparison.

Results and Analysis: The main results of the proposed work are given in Table 3.1. The proposed method considerably outperforms the PoseNet and Bayesian PoseNet on all scenes. It is noted, however, that the PoseNet based methods need only RGB images for both training and testing. The proposed method is not as accurate as the SCRF method. However, the proposed approach does not require the depth image in the testing stage, which greatly lightens the requirements of end users.

The best performance of the proposed method is found on the *Heads* Scene. The worst performance is for the *Stairs* in which the SCRF approach has the same problem due to the repetitive properties of the scene. The second worst scene is *Pumpkin* in which there are large uniformly colored planes, such as the cabinet or the ground. The lack of color distinction degrades the localization capability.

To separately evaluate the performance of random RGB features, the pixel-wise prediction error is visualized using heat maps as shown in Fig. 3.6. The error is the truncated distance between the predicted 3D locations and the ground truth. The typical large error areas are black screens and other texture-less regions, which are intractable for low-level visual features.

The pose refinement step by integrating sparse feature matching proved crucial to achieve good results. With this turned off, our RGB forest achieves only median

Table 3.1: Relocalization results for the 7 *Scenes* dataset. Median performance is shown for the proposed method on all scenes against three state-of-the-art methods: SCRF [93], PoseNet [53] and Bayesian PoseNet [51].

	# Frames		**Spatial**	**Baselines**			**Ours**
Scene	Train	Test	**Extent**	SCRF	PoseNet	Bayesian	
Training	—	—	—	RGB-D	RGB	RGB	RGB-D
Test	—	—	—	RGB-D	RGB	RGB	RGB
Chess	4k	2k	3x2x1m	0.03m, 0.66°	0.32m, 8.12°	0.37m, 7.24°	0.12m, 3.92°
Fire	2k	2k	2.5x1x1m	0.05m, 1.50°	0.47m, 14.4°	0.43m, 13.7°	0.14m, 4.64°
Heads	1k	1k	2x0.5x1m	0.06m, 5.50°	0.29m, 12.0°	0.31m, 12.0°	0.10m, 6.82°
Office	6k	4k	2.5x2x1.5m	0.04m, 0.78°	0.48m, 7.68°	0.48m, 8.04°	0.15m, 4.23°
Pumpkin	4k	2k	2.5x2x1m	0.04m, 0.68°	0.47m, 8.42°	0.61m, 7.08°	0.22m, 5.40°
Red Kitchen	7k	5k	4x3x1.5m	0.04m, 0.76°	0.59m, 8.64°	0.58m, 7.54°	0.14m, 3.71°
Stairs	2k	1k	2.5x2x1.5m	0.32m, 1.32°	0.47m, 13.8°	0.48m, 13.1°	0.30m, 8.08°
Average	—	—	—	0.08m, 1.60°	0.44m, 10.4°	0.47m, 9.81°	0.17m, 5.26°

Figure 3.6: Pixel-wise prediction error distribution from the regression forests. Heat maps show the prediction error distribution directly from the regression forests on the *Chess* and *Heads* scenes. Large errors occur on black screens and other texture-less regions.

localization result $0.21m, 6.09°$ for *Chess* and $0.25m, 8.53°$ on *Fire*, for instance.

3.3.2 Camera relocalization on the Stanford 4 *Scenes* dataset

Dataset: The 4 *Scenes* dataset was introduced in [103] to push the boundaries of RGB-D and RGB camera relocalization. It contains two apartment scenes and two office scenes which are very common indoor environments. The recorded environment is significantly larger than the 7 *Scenes* dataset [93] ($14-79m^3$ versus about $2-6m^3$). This large environment is more practical for the application of robot indoor localization. The scenes were captured by a Structure.io depth sensor with an iPad RGB camera. Both cameras have been calibrated and temporally

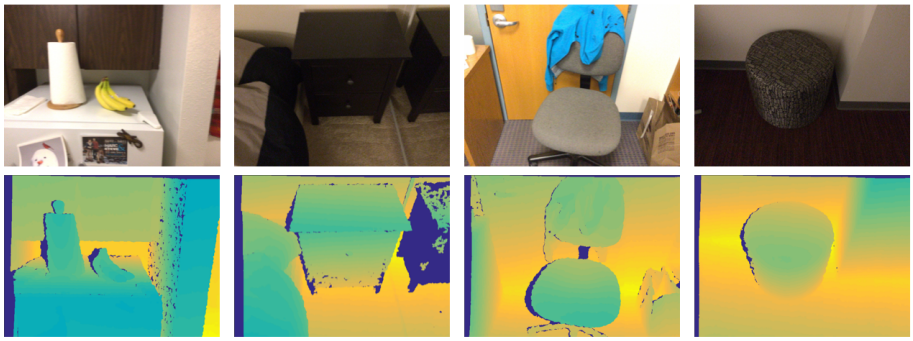


Figure 3.7: 4 *Scenes* **dataset example images[103].** Each RGB-D image is accompanied with a camera pose. Each sequence also provides a textured 3D model (not used in the present thesis).

synchronized. The RGB image sequences were recorded at a resolution of 1296×968 pixels, and the depth resolution is 640×480. The depth image is re-sampled to the RGB resolution to align the RGB and depth images. The ground truth camera poses are from BundleFusion [24], which is a real-time globally consistent 3D reconstruction system.

Baselines and Error Metric: Three state-of-the-art methods of camera relocalization with a single RGB image are used as the baselines: SIFT+PnP, ORB+PnP, Bayesian PoseNet[51]. The SIFT+PnP and ORB+PnP are based on matching local features, following mainstream feature-based relocalization approaches such as [72, 112]. SIFT+PnP uses SIFT[64] as features for feature description and matching while ORB+PnP uses ORB[84] as features.

These implementations are from [103]. As PoseNet [53] and Bayesian PoseNet [51] have similar performance in [51], here only the performance of Bayesian PoseNet is shown as the baseline. For error metric, the percentage of "correct frames" is used for SIFT+PnP, ORB+PnP and the proposed method, while the median performance is used for Bayesian PoseNet and our method. The "correct frames" is defined as the camera pose estimation error is within $5cm$ translational and 5^o rotational error compared with the ground truth. This accuracy is sufficient for augmented reality applications such as restarting any good model-based

tracking system [93].

Results and Analysis Table 3.2 shows the camera relocalization results for the 4 *Scenes* dataset. From this dataset, it can be seen that the proposed method is almost one order of magnitude more accurate than the Bayesian PoseNet. Compared with ORB+PnP and SIFT+PnP, our method has the best performance on some scenes, such as Apt1/living, Apt2/Kitchen, Office1/Floor5a, while ORB+PnP and SIFT+PnP also have the best performance on some scenes. It shows that the performance is highly dependent on the data properties. It is noticed that the proposed method has much better accuracy on the 4 *Scenes* dataset than on the 7 *Scenes* dataset, although the 4 *Scenes* dataset has a spatial extent that is an order of magnitude larger than those of 7 *Scenes*. This is probably due to the following reasons. First, the 4 *Scenes* dataset has much better RGB image quality, including higher resolution and less motion blur. The RGB image quality is critical for the proposed method as this method is using RGB pixel comparison feature. Second, the camera poses in 4 *Scenes* are not as challenging as 7 *Scenes*. For instance, the *Stairs* scene in 7 *Scenes* dataset contains too many similar and repetitive steps, which increases the inherent ambiguity.

Although SIFT+PnP and ORB+PnP have some best performance scenes compared with the proposed method, the proposed method has much better overall performance considering both speed and accuracy. The proposed method has similar efficiency with ORB+PnP but has 6% higher accuracy. Compared with SIFT+PnP, the proposed method is more efficient as it uses 32 dimension SIFT and searches only in the camera pose space of 6 dimensions rather than the SIFT feature space of 128 dimensions.

3.3.3 Implementation details

The approach developed in the present work is implemented with C++ using OpenCV [7] and VXL [1] on an Intel 2.3 GHz, 8GB memory Mac System. For the random forest, the parameter settings are: tree number $T = 10$ for 7 *Scenes* and $T = 5$ for 4 *Scenes*; 500 and 200 training images per tree for 7 *Scenes* and 4 *Scenes* datasets, respectively; 5,000 randomly sampled example pixels per training image. The

Table 3.2: Camera relocalization results for 4 *Scenes* **dataset.** The percentage of correct frames is given (within 5*cm* translational and 5° angular error), and median of the proposed method on 4 Scenes dataset against three state-of-the-art methods: ORB+PnP, SIFT+PnP, and Bayesian PoseNet[51]. The best performance is highlighted.

	Frame numbers		**Spatial**	**Baselines**			**Ours**
Sequence	training	test	**Extent**	ORB+PnP	SIFT+PnP	Bayesian PoseNet	Random+Sparse
Training	—	—		RGB-D	RGB-D	RGB	RGB-D
Test	—	—		RGB	RGB	RGB	RGB
Kitchen	744	357	$33m^3$	66.39%	**71.43**%	0.423m, 5.19°	0.030m, 1.73°, 70.3%
Living	1035	493	$30m^3$	41.99%	56.19%	0.611m, 3.70°	0.040m, 1.56°, **60.0**%
Bed	868	244	$14m^3$	71.72%	**72.95**%	0.626m, 7.68°	0.039m, 1.81°, 65.7%
Kitchen	768	230	$21m^3$	63.91%	71.74%	0.175m, 11.92°	0.030m, 1.45°, **76.7**%
Living	725	359	$42m^3$	45.40%	**56.19**%	0.294m, 10.10°	0.047m, 1.89°, 52.2%
Luke	1370	624	$53m^3$	54.65%	**70.99**%	0.649m, 7.71°	0.056m, 2.35°, 46.0%
Floor5a	1001	497	$38m^3$	28.97%	38.43%	0.437m, 8.43°	0.050m, 2.06°, **49.5**%
Floor5b	1391	415	$79m^3$	**56.87**%	45.78%	0.899m, 6.87°	0.042m, 1.42°, 56.4%
Gates362	2981	386	$29m^3$	49.48%	**67.88**%	0.223m, 6.77°	0.033m, 1.41°, 67.7%
Gates381	2949	1053	$44m^3$	43.87%	**62.77**%	0.410m, 10.35°	0.044m, 1.91°, 54.6%
Lounge	925	327	$38m^3$	**61.16**%	58.72%	0.602m, 7.75°	0.046m, 1.61°, 54.0%
Manolis	1613	807	$50m^3$	60.10%	**72.86**%	0.034m, 2.02°	0.034m, 1.56°, 65.1%
Average	—	—	—	53.7%	**62.2**%	0.483m, 8.08°	0.041m, 1.73°, 59.9%

maximum tree depth is 16. A modified 32-dimension SIFT feature is used from the VLFeat library [105] for sparse feature matching. The SIFT feature computation is still the bottleneck of the current implementation, the frame-rate response can be achieved with the GPU SIFT [114]. For the camera pose optimization, off-the-shelf *solvePnPRansac* is used in OpenCV.

In the present work, the PoseNet and Bayesian PoseNet baselines are run using the code by their original author on a Linux machine with an Nvidia GeForce GTX670 GPU. The pre-trained weights are used to initialize the network weights.

3.4 Conclusions

This chapter presented a hybrid method using random RGB features and sparse features for camera pose estimation. The method used RGB-only images in the test and achieves near real-time response. The comparisons on the challenging 7 *Scenes*

and 4 *Scenes* dataset with several strong baselines demonstrated the efficacy of the developed method, showing comparable results with state-of-the-art methods.

Future work may include improving the prediction accuracy using deep features. It is also planned to investigate the possibility of integrating accurate sparse features with the random features in the training phase so that the prediction accuracy could be further improved from random regression forests.

Chapter 4

Image-based localization using backtracking regression forests

This chapter presents a backtracking technique that improves the accuracy of camera relocalization. Methods based on random forests for camera relocalization directly predict 3D world locations for 2D image locations to guide the camera pose optimization. During training, each tree greedily splits the samples to minimize the spatial variance. However, these greedy splits often produce uneven sub-trees or incorrect 2D-3D correspondences predictions. To address these problems, a sample-balanced objective is proposed to encourage equal numbers of samples in the left and right sub-trees, and a novel backtracking scheme to remedy the incorrect 2D-3D correspondence in the leaf nodes caused by greedy splitting. Furthermore, the regression forests based methods are extended to use local features in both training and test stages for outdoor applications. Experimental results using publicly available indoor and outdoor datasets demonstrate the efficacy of the developed approach, showing superior or on-par accuracy with several state-of-the-art baselines.

4.1 Introduction

Methods based on local features and keyframes have been extensively studied and are still very active approaches for image-based camera relocalization [54, 87, 89].

Recent advances in machine learning methods, especially random forests and deep learning, have led to rapid progress on their application to camera relocalization.

Methods based on random forests are among the first machine learning methods for camera relocalization [38, 93, 102]. In these methods, the forest is trained to directly predict correspondences from any image pixel to points in the scene's 3D world coordinate, which removes the traditional pipeline of feature detection, description and matching. These correspondences are then used for camera pose estimation based on an efficient RANSAC algorithm without an explicit 3D model of the scene, which is very attractive when a 3D model is not available. Furthermore, these methods can localize the camera from a single RGB-D frame without tracking. The latest work [6, 70] extends these random forests based methods to test with just RGB images. However, depth images are still necessary to get 3D world coordinate labels in the training stage.

In the training stage of these methods based on random forests, each tree greedily splits the samples to minimize the spatial variance. However, these greedy splits usually produce uneven left and right sub-trees, or incorrect 2D-3D correspondences in the test. To address this problem, a label-free sample-balanced objective is proposed here to encourage equal numbers of samples in the left and right sub-trees, and a novel backtracking scheme to remedy the incorrect 2D-3D correspondence in the leaf nodes caused by greedy splitting. The efficacy of the developed methods is demonstrated through evaluations on publicly available indoor and outdoor datasets.

Some recent methods based on deep learning [51, 53, 108] overcome the challenges of using depth images for camera relocalization, extending their application to outdoor scenes. These methods train a convolutional neural network to regress the 6-DOF camera pose from a single RGB image in an end-to-end manner in real time. However, even integrating LSTM units on the CNN output [108], these methods still generate much lower accuracy compared with methods based on random forests [93] in indoor scenes and methods based on local features [87] in outdoor scenes in general. To eliminate the dependency on depth images while ensuring high accuracy, the present method integrates local features in the random forests, broadening the application of random forests methods to outdoor scenes for the first time while achieving the best accuracy against several strong state-of-the-art

baselines.

To summarize, the main contributions of the work in this chapter are as follows:

- The present work proposes a sample-balanced objective that encourages equal numbers of samples in the sub-trees, increasing prediction accuracy while reducing training time.
- The present work proposes a novel backtracking scheme to remedy incorrect 2D-3D correspondence in the leaf nodes caused by greedy splitting, further improving prediction accuracy.
- The present work integrates local features in the regression forests, enabling the use of just RGB images for both training and testing. The elimination of the dependence on depth images broadens the present scope of application to outdoor scenes.

4.2 Method

The present work models the world coordinate prediction problem as a regression problem:

$$\hat{\mathbf{m}}_{\mathbf{p}} = f_r(\mathtt{I}, \mathtt{D}, \mathbf{p}|\theta) \tag{4.1}$$

where $\mathtt{I}$ is an RGB image, $\mathtt{D}$ is an optional depth image, $\mathbf{p} \in \mathbb{R}^2$ is a 2D pixel location in the image $\mathtt{I}$, $\mathbf{m}_{\mathbf{p}} \in \mathbb{R}^3$ is the corresponding 3D scene coordinate of $\mathbf{p}$, and θ is the model parameter. In training, $\{\mathbf{p}, \mathbf{m}_{\mathbf{p}}\}$ are paired training data. In testing, the 3D world location $\hat{\mathbf{m}}_{\mathbf{p}}$ is predicted by the model θ. Then, the camera parameters are estimated using predicted image-world correspondences in a RANSAC framework.

The novelty of the present method lies in both the regression forest training and prediction. First, the present work uses a sample-balanced split objective in training. This objective function encourages equal numbers of samples in the left and right sub-trees. In practice, it acts as a regularization term in the training and thus improving prediction accuracy. Second, the present work presents a novel backtracking technique in prediction. The backtracking searches the tree using a priority queue and decreases the chance of falling in a local minimum.

Table 4.1: Image features. The present method uses random features [93], SIFT features [64] and Walsh-Hadamard transform (WHT) features [43] according to different scenarios. The choice of features considers robustness and computational efficiency.

	Feature Types		
Dataset	Split	Backtrack	Local descriptor
RGB-D	Random	Random	WHT
RGB	SIFT	SIFT	SIFT

4.2.1 Image features

The present method employs different image feature $f_{\phi_i}(\mathbf{p})$ that associates with pixel location $\mathbf{p}$ as shown in Table 4.1 according to different application scenarios.

Image Features on RGB-D Images

For indoor scenes, the present method uses random pixel comparison features[93] and Walsh-Hadamard transform (WHT) features [43]. Random features and WHT features do not require expensive feature detection so they can speed up the process. Also they provide a sufficient number of robust features in texture-less areas, which is especially important for indoor camera relocalization.

In the present method, random features are used for splitting decision trees in internal nodes, while WHT features are used to describe local patches $P_r(\mathbf{p})$ centered at the pixel $\mathbf{p}$. The random features as shown in Fig. 4.1 are based on pairwise pixel comparison as in [93, 102]:

$$f_\phi(\mathbf{p}) = \mathtt{I}(\mathbf{p}, c_1) - \mathtt{I}(\mathbf{p} + \frac{\delta}{\mathtt{D}(\mathbf{p})}, c_2) \tag{4.2}$$

where δ is a 2D offset and $\mathtt{I}(\mathbf{p}, c)$ indicates an RGB pixel lookup in channel c. The ϕ contains feature response parameters $\{\delta, c_1, c_2\}$. The $\mathtt{D}(\mathbf{p})$ is the depth at pixel $\mathbf{p}$ in image $\mathtt{D}$ of the corresponding RGB image $\mathtt{I}$. Pixels with undefined depth and those outside the image boundary are marked and not used as samples for both training and test.

Figure 4.1: Depth-adaptive random RGB pixel comparison features. The red star represents the pixel p being computed. The distance between red star and blue circle represents the offset pixels $\frac{\theta}{D(\mathbf{p})}$ as defined in Eq. 4.2. In (a), the two example features at image position p_1 and p_2 give a large color difference response. In (b), the two features at the same image locations in a different image give a much smaller response compared with (a).

For WHT features, the Walsh-Hadamard Transform [43] is calculated for all patches $P_r(\mathbf{p})$ centered at all pixel positions $\mathbf{p}$. We choose the first 20 Walsh-Hadamard projection vectors for each color channel, thus the total WHT feature dimension is 60 for all the three color channels in the present case. The reason of choosing the first 20 projection vectors that they correspond to the ones with largest average magnitude of responses, while the rest mainly capture very fine details and thus do not influence performance too much. The fixed patch size of 64×64 pixels is used for the WHT feature.

Image Features on RGB images

For scenarios where only RGB images are available in both training and test phase, the present method uses SIFT features [64] as local feature descriptors. The 64 dimension SIFT features are used to decrease the trained model size as our back-tracking regression forests method stores the mean vector of feature descriptors.

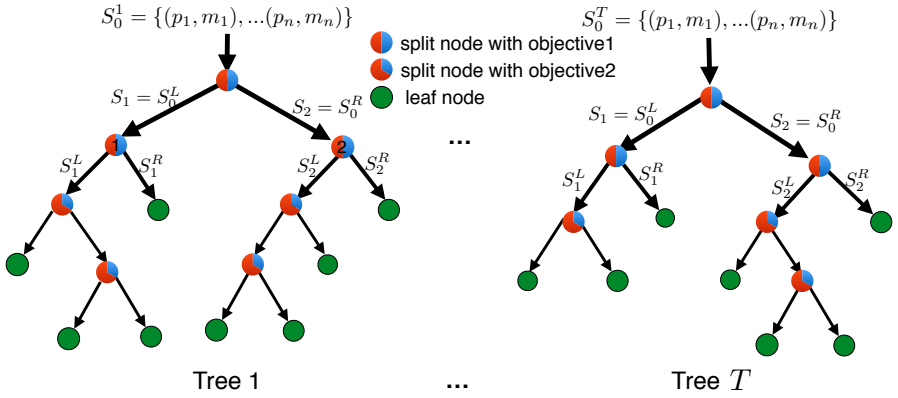


Figure 4.2: Training random forests for 2D-3D coordinate correspondence regression. A set S_0^t of labeled sample pixels $(\mathbf{p}, \mathbf{m})$ is randomly chosen from the entire set $S_0 = [S_0^1, \cdots, S_0^T]$ for each tree, where $\mathbf{m}$ is the 3D world coordinate label of pixel $\mathbf{p}$. The tree grows recursively from the root node to the leaf node. The goal is to optimize the parameters of the tree split nodes. The training objective 1 is the sample-balanced objective, and the training objective 2 is the spatial-variance objective.

4.2.2 Scene coordinate labels

The labels for image features are the pixel $\mathbf{p}$'s corresponding world coordinate point position $\mathbf{m}$.

Labels for RGB-D Images

For scenarios where RGB-D images are available in the training stage, please refer to Sec. 3.2.1 on the label details.

Labels for RGB Images

For scenarios where only RGB images are available in the test stage, for each feature point at pixel $\mathbf{p}$, the scene's world coordinate position $\mathbf{m}$ could be obtained from visual structure from motion [115].

Algorithm 2 Building the random forests

Input: Feature-3D dataset D ▷ from Visual Structure from Motion
Input: Number of trees N
Output: The random forest data structure

procedure BUILDRANDOMFOREST(D,N)
  $trees \leftarrow \{\}$
  for $i \leftarrow \{1 \cdots N\}$ **do**
    $D_i \leftarrow random\ subset\ of\ D$ ▷ e.g., data from 200 frames
    $trees_i \leftarrow$ BUILDTREE(D)
  end for
  return trees
end procedure
procedure BUILDTREE(D)
  if $|D| == 1$ **then** ▷ or satisfying other criterion
    CREATELEAFNODE(D) ▷ store mean values of labels and features
  else
    $\Phi \leftarrow$ a set of random split dimension and value
    $\{o^*, \phi^*\} \leftarrow \{\infty, \emptyset\}$ ▷ ϕ has splitDim and splitValue
    for each ϕ in Φ **do**
      $o \leftarrow$ objective value by splitting D using ϕ
      if $o < o^*$ **then**
        $\phi^* = \phi$ ▷ update the optimal splitting
      end if
    end for
    $\{D_l, D_r\} \leftarrow$ splitted dataset using ϕ^*
    $T_l \leftarrow$ BUILDTREE(D_l) ▷ left subtree
    $T_r \leftarrow$ BUILDTREE(D_r) ▷ right subtree
    CREATEINNERNODE(ϕ^*, T_l, T_r) ▷ Store splitting parameters
  end if
end procedure

4.2.3 Backtracking regression forest training

Weak learner model

A regression forest is an ensemble of T independently trained decision trees. Each tree is a binary tree structure consisting of split nodes and leaf nodes. We use the same weak learner as in Chapter 3.2.2.

Training objectives

At each split node i, for the incoming training set S_i, the training is 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. The selecting "best" split parameters can be formulated as the minimize the training objective Q_i:

$$\theta_i^* = \arg\min_{\theta_i \in \Theta_i} Q_i(S_i, \theta_i) \tag{4.3}$$

The present regression forest uses two objectives depending on the levels of the decision tree. At upper levels of a decision tree (tree depth is smaller than a threshold L_{max}), the present work uses a sample-balanced objective:

$$Q_b(S_i, \theta) = \frac{abs(|S_i^L| - |S_i^R|)}{|S_i^L| + |S_i^R|} \tag{4.4}$$

where $abs(.)$ is the absolute value operator and $|S|$ represents the size of set S. This objective penalizes uneven splitting. The sample-balanced objective has two advantages. First, it makes the training faster as it only counts the number of samples in sub-trees. Second, it produces more accurate predictions in practice, which will be shown in the experiments Sec. 4.3. When the tree depth is larger than a threshold L_{max}, the present work uses the spatial-variance objective Eqn. 3.8.

Fig.4.3 illustrates the idea of using two objectives in a five-level tree. The first two levels use the sample-balanced objective while the next two levels use the spatial-variance objective.

At the end of training, all samples reach to the leaf nodes. In a leaf node, the present method not only stores a mean vector of 3D positions but also a mean vector of local patch descriptors. The local patch descriptor will be used to choose the optimal predictions in a backtracking process which will be described in Sec.4.2.4.

4.2.4 Backtracking in regression forests prediction

In the testing phase, a regression tree predicts 3D locations by comparing the feature value and the feature response at the split nodes. Because the comparison is conducted on a single dimension at a time and all these parameters are optimized greedily during the training, making mistakes is inevitable.

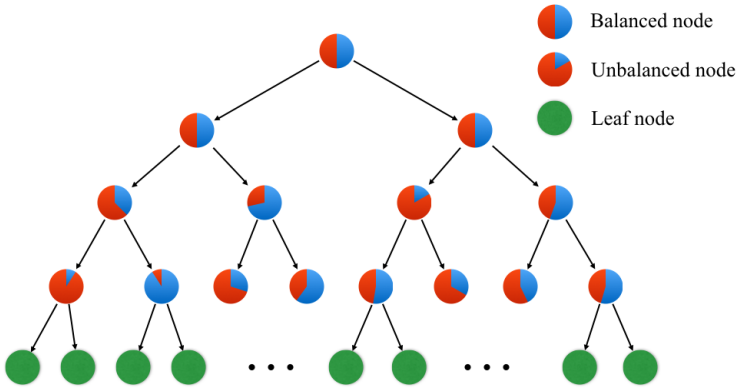


Figure 4.3: Decision tree using two split objectives. Best viewed in color. The split nodes are illustrated as the pie charts, which show the percentage of samples in the left and right sub-trees. In this five-level tree, the split nodes of the first two levels are split using the sample-balanced objective. While the rest of levels are split using the unbalanced objectives (e.g., the spatial-variance objective). Details are in Sect. 4.2.3.

To mitigate this drawback, the present work has developed a backtracking scheme in the testing phase as shown in Fig. 4.4 to find the optimal prediction within time budgets using a priority queue. The priority queue stores the sibling nodes that are not visited along the path when a testing sample travels from the root node to the leaf node. The priority queue is ordered by increasing distance to the split value of each split node. The backtracking continues until a predefined number N_{max} of leaf nodes are visited. Algorithm 3 illustrates the detailed procedure of the present backtracking scheme.

4.2.5 Camera pose optimization

The backtracking regression forest described above is capable of predicting the 3D world coordinate for any $2D$ image pixel. The present work uses this $2D-3D$ correspondence to estimate the camera pose. The problem of estimating the camera pose is formulated as the energy minimization:

$$\mathtt{P}^* = \arg\min_{\mathtt{P}} E(\mathtt{P}) \tag{4.5}$$

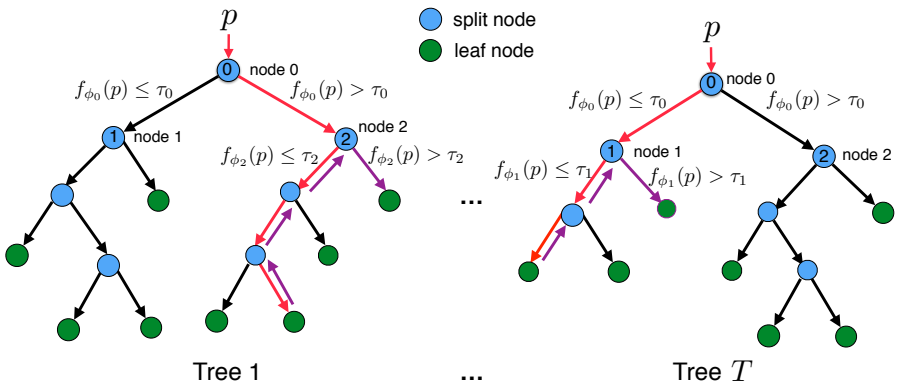


Figure 4.4: Test phase of backtracking regression forests In the test, at each split node i, the feature f_{ϕ_i} is compared with feature response τ_i to determine whether to go to the left or to the right child node. The red arrows represent the prediction process without backtracking while the purple arrows represent the backtracking process. A particular input may go along different paths in different trees as indicated by the red and purple arrows.

where P is a camera pose matrix. The energy function can be solved in different ways depending on whether depth information is available in the test phase.

When depth images are available (i.e., indoor application), we use the Kabsch algorithm [49] to estimate camera poses. Otherwise, the *solvePnPRansac* method from OpenCV [7] is used. Because predictions from the regression forests may still have large errors, the preemptive RANSAC method [74] is used to remove outliers.

4.3 Experiments

This section evaluates the present camera relocalization methods using publicly available indoor and outdoor datasets against several strong state-of-the-art methods.

Table 4.2: Camera relocalization results for the indoor dataset. The percentage of correct frames (within 5*cm* translational and 5° angular error) of the developed method is shown on 4 Scenes dataset against four state-of-the-art methods: ORB+PnP, SIFT+PnP, Random+SIFT [70], MNG [103]. The best performance is highlighted.

	Spatial	**Baselines**				**Our Results**			
Sequence	**Extent**	ORB+PnP	SIFT+PnP	Random+SIFT[70]	MNG[103]	BTBRF	UBRF	BRF	BTBRF
Training		RGB-D	RGB-D	RGB-D	RGB-D	RGB-D	RGB-D	RGB-D	RGB-D
Test		RGB	RGB	RGB	RGB+3D model	RGB	RGB-D	RGB-D	RGB-D
Kitchen	$33m^3$	66.39%	71.43%	70.3%	85.7%	77.1%	82.6%	88.2%	**92.7**%
Living	$30m^3$	41.99%	56.19%	60.0%	71.6%	69.6%	81.7%	90.5%	**95.1**%
Bed	$14m^3$	71.72%	72.95%	65.7%	66.4%	60.8%	71.6%	81.3%	**82.8**%
Kitchen	$21m^3$	63.91%	71.74%	76.7%	76.7%	82.9%	80.0%	85.7%	**86.2**%
Living	$42m^3$	45.40%	56.19%	52.2%	66.6%	62.5%	65.3%	92.3%	**99.7**%
Luke	$53m^3$	54.65%	70.99%	46.0%	83.3%	39.3%	50.5%	71.5%	**84.6**%
Floor5a	$38m^3$	28.97%	38.43%	49.5%	66.2%	50.3%	68.0%	85.7%	**89.9**%
Floor5b	$79m^3$	56.87%	45.78%	56.4%	71.1%	58.5%	83.2%	92.3%	**98.9**%
Gates362	$29m^3$	49.48%	67.88%	67.7%	51.8%	82.1%	88.9%	90.4%	**96.7**%
Gates381	$44m^3$	43.87%	62.77%	54.6%	52.3%	51.4%	73.9%	82.3%	**92.9**%
Lounge	$38m^3$	61.16%	58.72%	54.0%	64.2%	59.6%	74.6%	91.4%	**94.8**%
Manolis	$50m^3$	60.10%	72.86%	65.1%	76.0%	68.5%	87.2%	93.9%	**98.0**%
Average	—	53.7%	62.2%	59.9%	69.3%	63.6%	75.6%	87.1%	**92.7**%

4.3.1 Indoor camera relocalization

Indoor camera relocalization on Stanford 4 Scenes Dataset

Baselines: The present method uses four state-of-the-art methods as the baselines: SIFT+PnP, ORB+PnP, Random+SIFT [70], Multiscale Navigation Graph (MNG) [103]. The SIFT+PnP and ORB+PnP are based on matching local features, following mainstream feature-based relocalization approaches such as [72, 112]. These methods have to store a large database of descriptors. Random+SIFT [70] uses both random features and sparse features but in two separate steps. The sparse feature is used as post-processing in this method. The present method is different from this method by simultaneously using both random feature and sparse features within the framework of regression forests. The multiscale navigation graph (MNG) method [103] estimates the camera pose by maximizing the photo-consistency between the query frame and the synthesized image which is conditioned on the camera pose. In the testing stage, the MNG method requires the

trained model (retrieval forests) and the 3D model of the scene.

Error Metric: The percentage of test frames for which the estimated camera pose is essentially 'correct' is reported. A pose is correct when it is within $5cm$ translational error and 5^o angular error of the ground truth.

Main Results and Analysis: For the indoor dataset, the present work uses random features and Walsh-Hadamard transform (WHT) features [43]. The obtained main camera relocalization results are presented in Table 4.2. The results of ORB+PnP, SIFT+PnP and MNG are from Valentin *et al.*[103]. Besides the developed final method BTBRF, presented here are the results of unbalanced regression forest (UBRF) which does not use the sample-balanced objective, and balanced regression forest (BRF) which uses both sample-balanced objective and spatial-variance objective in training but does not use backtracking in testing.

In this dataset, sparse baselines do a reasonable job and the SIFT+PnP method is better than the ORB+PnP method. The present method using RGB-only images in the testing stage achieves higher accuracy than SIFT+PnP, ORB+PnP, and Random+Sparse, and is less accurate than the MNG method. However, the MNG method needs a explicit 3D model to render synthetic images to refine the pose while the present method does not need. Moreover, the MNG method needs a large number of synthetic images (9 times of the original training images) for data augmentation while the present method does not need them. The present method using RGB-D images at test time considerably outperforms all the baselines in accuracy for camera relocalization.

To demonstrate that the improvement is indeed from the sample-balanced objective and backtracking, the present work compares the world coordinate prediction accuracy from the present UBRF, BRF and BTBRF. Fig. 4.5 shows the accumulated percentage of predictions within different error thresholds for four sequences. For a particular threshold, the higher the percentage, the more accurate the prediction is. The figure clearly shows that the present method with the sample-balanced objective and the backtracking technique is consistently better than the method without sample-balanced objective and without backtracking.

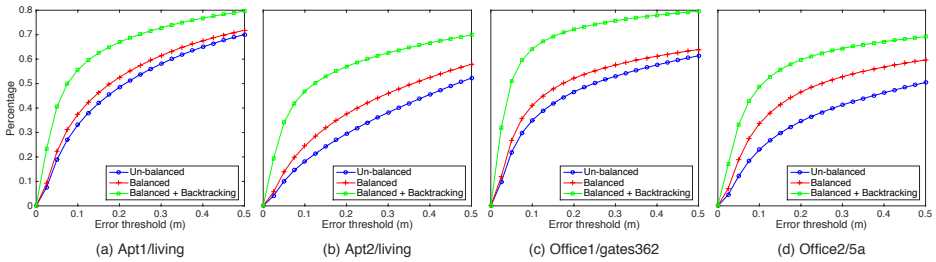


Figure 4.5: Impact of the sample-balanced objective and backtracking on prediction accuracy. These figures show the accumulated percentage of predictions within a sequence of inlier thresholds. The proposed method with the sample-balanced objective (red lines) consistently has a higher percentage of inliers compared with the unbalanced objective (blue lines). Backtracking (green lines) further improves prediction accuracy. Max number of backtracking leaves is 16 here.

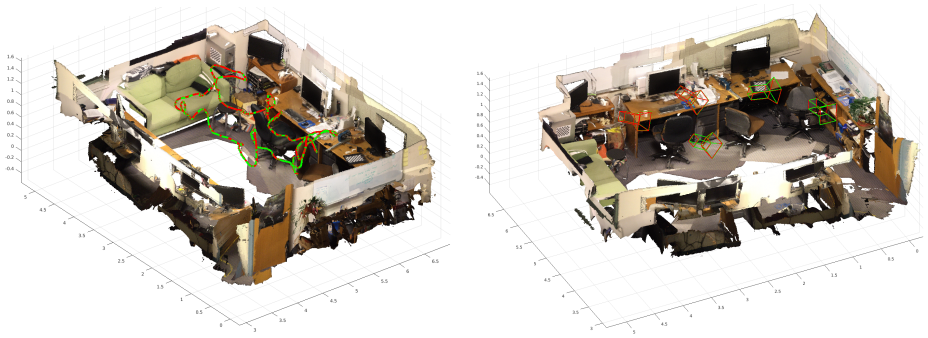


Figure 4.6: Qualitative results for indoor dataset (from office2/gates381). Best viewed in color. The ground truth is in red and the present estimated camera pose is in green. (a) camera trajectories. (b) several evenly sampled camera frusta are shown for visualization. the present method produces accurate camera locations and orientations. Note: the 3D model is only for visualization purposes and it is not used for the present camera relocalization.

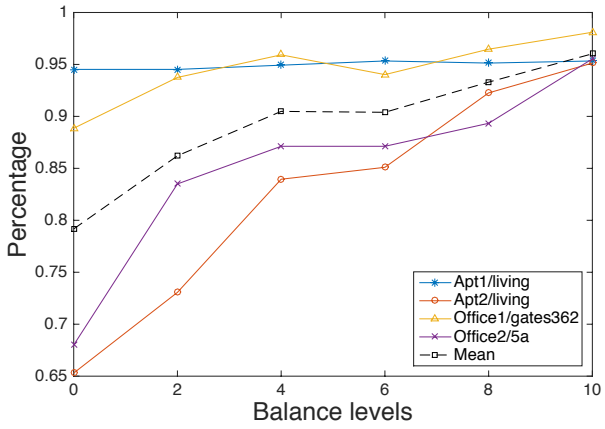


Figure 4.7: Camera relocaliztion accuracy vs. number of tree levels using the sample-balanced objective. When the number of levels using sample-balanced objective increases, the average performance (dashed line) increases.

Fig. 4.6 shows examples of qualitative results of the present method. In Fig. 4.6 (a), the estimated camera trajectory and the ground truth trajectory are highly overlapped in the scene, illustrating the high accuracy of predicted camera locations. In Fig. 4.6 (b), several camera frusta show the accuracy of estimated camera orientation. These results show that the estimated camera poses are accurate in both location and orientation.

Tree Structure Analysis

The structure of the developed decision trees is analyzed to explore the influence of several important parameters.

Maximum tree depth of using the sample-balanced objective L_{max} : Fig. 4.7 shows relocalization accuracy against tree levels with the sample-balanced objectives. The sample-balanced level is increased from 0 (no sample-balanced level) to 10. The mean accuracy gradually increases from around 80% to around 95%. Analyzing the four sequences separately, we find that the performance on the *apt2/living* and *offce2/5a* sequences improved significantly while the performance on the other two sequences improve moderately. Please note the accuracy in this figure is measured without using backtracking.

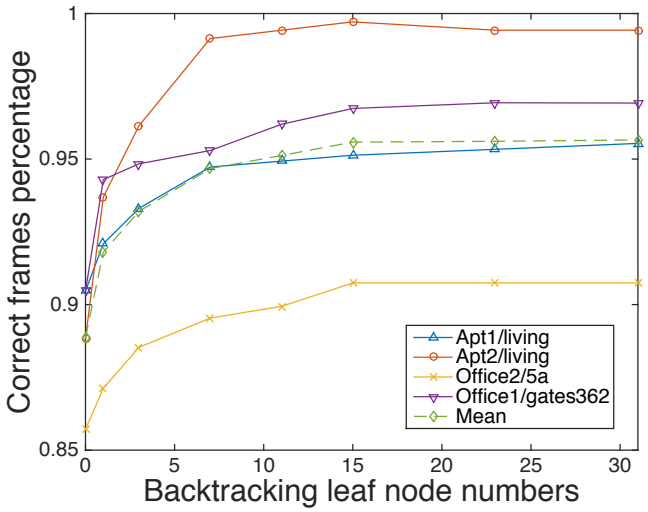


Figure 4.8: Camera relocalization accuracy vs. backtracking leaf node numbers. The camera relocalization performance increases with more backtracking leaf nodes though eventually levels out.

Backtracking number N_{max}: Fig. 4.8 shows the camera relocalization accuracy against different N_{max}. The accuracy is significantly improved within the first (about) 10 leaf nodes and saturates after that. Because the processing time linearly increases with the backtracking number, a small number of N_{max} is preferred.

Split Ratio Distribution: In internal nodes, the split ratio measures the balance of the number of samples. Fig. 4.9 shows an example of split ratio distribution. The tree uses the sample-balanced objectives in the first 8 levels. After level 8, the distribution is close to a uniform distribution. The present method successfully guides the tree to have a sample-balanced structure at designated levels. Compared with the unbalanced tree structure, the present method has relatively shallower trees. As it has been pointed out, deeper trees lead to over-fitting [20].

Tree Numbers: Fig. 4.10 plots regression forests method on apt1/kitchen scene. It illustrates that the camera relocalization performance increases with the number of trees while eventually leveling out. Other random forest variants for different tasks [20, 92] also have similar findings. It has been proved that the random forests have less danger of overfitting when the number of trees increases [9]. However, the memory overhead and training time of using multiple trees increase

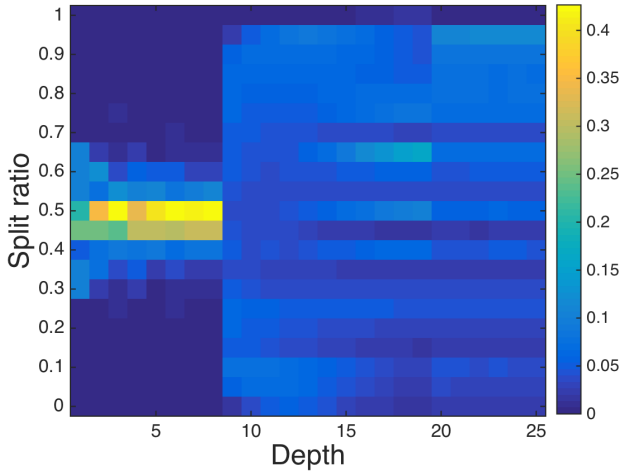


Figure 4.9: Split ratio distribution. The heat map shows the split ratio distribution as a function of tree depth (level). The data is from all split nodes in a sequence. In the first 8 levels, the distribution is concentrated around 0.5 as a result of the sample-balanced objective. From level 8 to level 25, the distribution is almost uniform.

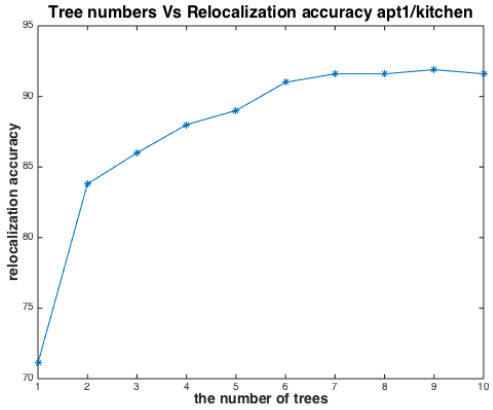


Figure 4.10: Camera relocalization accuracy VS number of trees

Table 4.3: Camera relocalization results for the 7 *Scenes* **dataset** .Correct percentage performance is shown for the developed method on all scenes against three state-of-the-art methods: Sparse baseline [93], SCRF [93], MutliOutput [38]. The correct percentage show the test frames within 5*cm* translational and 5° angular error.

	Baselines			**Ours**	
Scene	Sparse [93]	SCRF [93]	MultiOutput[38]	BRF	BTBRF
Training	RGB-D	RGB-D	RGB-D	RGB-D	RGB-D
Test	RGB	RGB	RGB-D	RGB-D	RGB-D
Chess	70.7%	92.6%	96%	97.8%	**99.6**%
Fire	49.9%	82.9%	90%	92.1%	**95.2**%
Heads	67.6%	49.4%	56%	77.3%	**90.4**%
Office	36.6%	74.9%	92%	91.3%	**95.9**%
Pumpkin	21.3%	73.7%	**80**%	76.5%	75.7%
Kitchen	29.8%	71.8%	86%	85.4%	**89.4**%
Stairs	9.2%	27.8%	55%	51.3%	**60.8**%
Average	40.7%	67.6%	79.3%	81.7%	**86.7**%

linearly with the number of trees, so at some point the accuracy increase may not justify the additional memory used. Fig. 4.10 shows the relocalization accuracy increases steeply at first several trees but slowly after around 5 trees. Similar trends are observed on other sequences. As a result, the present work uses 5 trees to strike a trade off between speed and accuracy.

Indoor camera relocalization on 7 Scenes Dataset

The developed method is also evaluated using the widely used Microsoft 7 *Scenes* dataset. Also the present method is compared against a sparse feature based method reported from [93], SCRF [93] and a multi-output version of SCRF [38] in terms of correct frame percentage. The present method is compared with PoseNet and Bayesian PoseNet in terms of median translation error and rotation error.

Main results and analysis: The main results and comparisons are presented in Table 4.3. The developed method BTBRF outperforms all the baselines on all

Table 4.4: Median camera relocalization performance for the 7 *Scenes* **dataset** . Median performance for the present method on all scenes is shown against three state-of-the-art methods: PoseNet [53], Bayesian [51], SCRF [93].

	Baselines			**Ours**	
Scene	PoseNet [53]	Bayesian [51]	SCRF [93]	BRF	BTBRF
Training	RGB	RGB	RGB-D	RGB-D	RGB-D
Test	RGB	RGB	RGB-D	RGB-D	RGB-D
Chess	0.32m, 8.12°	0.37m, 7.24°	0.03m, 0.66°	0.017m, 0.69°	**0.015**m, **0.59**°
Fire	0.47m, 14.4°	0.43m, 13.7°	0.05m, 1.50°	0.018m, 0.99°	**0.016**m, **0.89**°
Heads	0.29m, 12.0°	0.31m, 12.0°	0.06m, 5.50°	0.028m, 2.66°	**0.020**m, **1.84**°
Office	0.48m, 7.68°	0.48m, 8.04°	0.04m, 0.78°	0.022m, 0.90°	**0.018**m, **0.75**°
Pumpkin	0.47m, 8.42°	0.61m, 7.08°	0.04m, **0.68**°	0.024m, 0.90°	**0.023**m, 0.84°
Kitchen	0.59m, 8.64°	0.58m, 7.54°	0.04m, **0.76**°	0.027m, 1.22°	**0.025**m, 1.02°
Stairs	0.47m, 13.8°	0.48m, 13.1°	0.32m, 1.32°	0.047m, 1.40°	**0.040**m, **1.22**°
Average	0.44m, 10.4°	0.47m, 9.81°	0.08m, 1.60°	0.261m, 1.25°	**0.022**m, 1.02°

scenes except the *Pumpkin* scene in terms of correct percentage. The sparse baseline tends to perform poorly when there exists severe motion blur, or textureless areas in which situations few features are detected. The SCRF performs better than the present BRF but worse than BTBRF. The main reason is that mean shift is not used, which may improve the leaf prediction. Only the balanced objective cannot deal with the ambiguities on the leaf prediction but the backtracking scheme helps more. All these three methods perform poorly on the *Stairs* due to the inherently repetitive and ambiguous property of the scene. However, the present method achieves 35% more in correct percentage accuracy compared with the SCRF, 86% more with AutoContext and 308% more with Spare baseline. The performance boost may be attributed to the balanced objective and backtracking could help to correct the severely wrong prediction.

To gain some insight into the translation and rotation error, the median performance is also presented in Table 4.4. It is can be seen that SCRF and the present methods are significantly more accurate than PoseNet and Bayesian PoseNet. The reason may be that both PoseNet and Bayesian PoseNet consider the holistic images which tends to fail when occlusions occurred. The other reason may be that

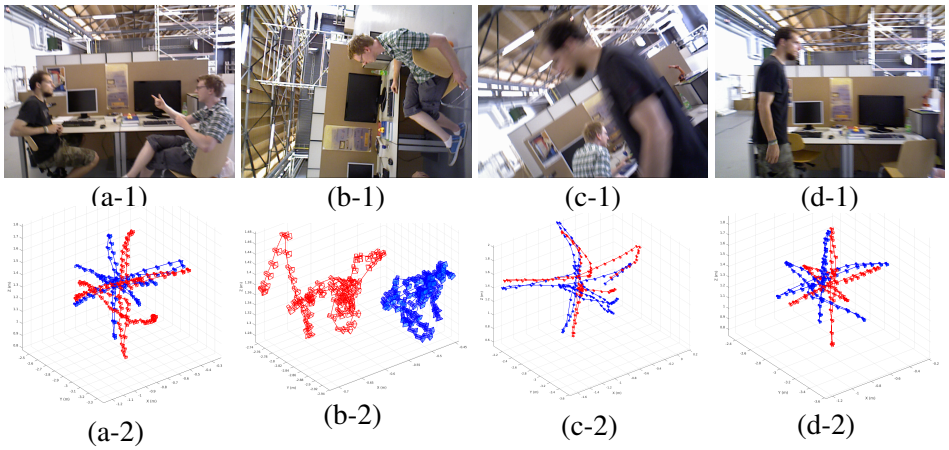


Figure 4.11: Example sequence in TUM dynamic dataset. (a) sitting_xyz, (b) sitting_rpy, (c) walking_halfsphere, (d) walking_xyz. The upper row shows the training image example. The training images also contain severe motion blur, occlusion and rotated images. The lower row shows the training (blue) and test (red) sequences. The camera frustums are uniformely sampled in every tenth image.

they do not use the depth image and lose more information. The present method BTBRF has much better performance on translational error but inferior performance on angular error.

Indoor camera relocalization on TUM Dynamic Dataset

TUM RGB-D dynamic dataset: TUM RGB-D Dataset [98] is mainly for the evaluation of RGB-D SLAM systems. A large set of image sequences of various of characteristics (e.g. non-texture, dynamic objects, hand-held SLAM, robot SLAM) from a Microsoft Kinect RGB-D sensor with highly accurate and time-synchronized ground truth camera poses from a motion capture system. The sequences contain both the color and depth images in image resolution of 640×480. Here, just the dynamic objects dataset is used to complement the previous static *Microsoft* 7 *Scenes* dataset and *Stanford* 4 *Scenes* dataset where no dynamic objects existed. This dynamic dataset is very challenging as there are severe occlusions and moving dynamic objects in the scene. The training set is from the scenes as listed in Table 4.5 while the test set is from the respective evaluation sequences.

Table 4.5: Camera relocalization results for the TUM dataset . The correct percentage performance, median performance, and RMSE of ATE are presented.

	# Frames		BTBRF		
Scene	Train	Test	Correct	Median	RMSE
sitting_static	676	715	64.6%	0.015m, 0.99°	0.018m
sitting_xyz	1216	829	70.2%	0.029m, 0.72°	0.039m
sitting_halfsphere	1069	944	44.4%	0.056m, 1.59°	0.046m
sitting_rpy	792	753	74.6%	0.029m, 0.98°	0.065m
walking_halfsphere	1018	1171	61.7%	0.042m, 1.03°	0.085m
walking_rpy	864	777	53.7%	0.047m, 1.14°	0.551m
walking_static	714	785	89.2%	0.019m, 0.49°	0.027m
walking_xyz	826	897	41.7%	0.048m, 1.24°	0.064m
Average	—	—	62.5%	0.036m, 1.02°	0.119m

Fig. 4.11 show several RGB example for training sequences.

Error metric Three error metrics are used for this evaluation: 'Correct frames' which is the percentage of test frames within $5cm$ and 5°, median translational and angular error, and root mean squared error (RMSE) for Absolute Trajectory Error (ATE) [98] which is commonly used in many SLAM systems [72, 98]. Unlike the $Microsoft$ 7 *Scenes* and $Stanford$ 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, TUM dataset benchmark provide tools using Horn's method [45] to align the estimated trajectory $\mathbf{P}_{1:n}$ and ground truth trajectory $\mathbf{Q}_{1:n}$. The ATE at time step i is computed as

$$\mathbf{F}_i = \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{4.6}$$

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

$$RMSE(\mathbf{F}_{1:n}) = (\frac{1}{n}\sum_{i=1}^{n}||trans(\mathbf{F}_i)||^2)^{\frac{1}{2}} \tag{4.7}$$

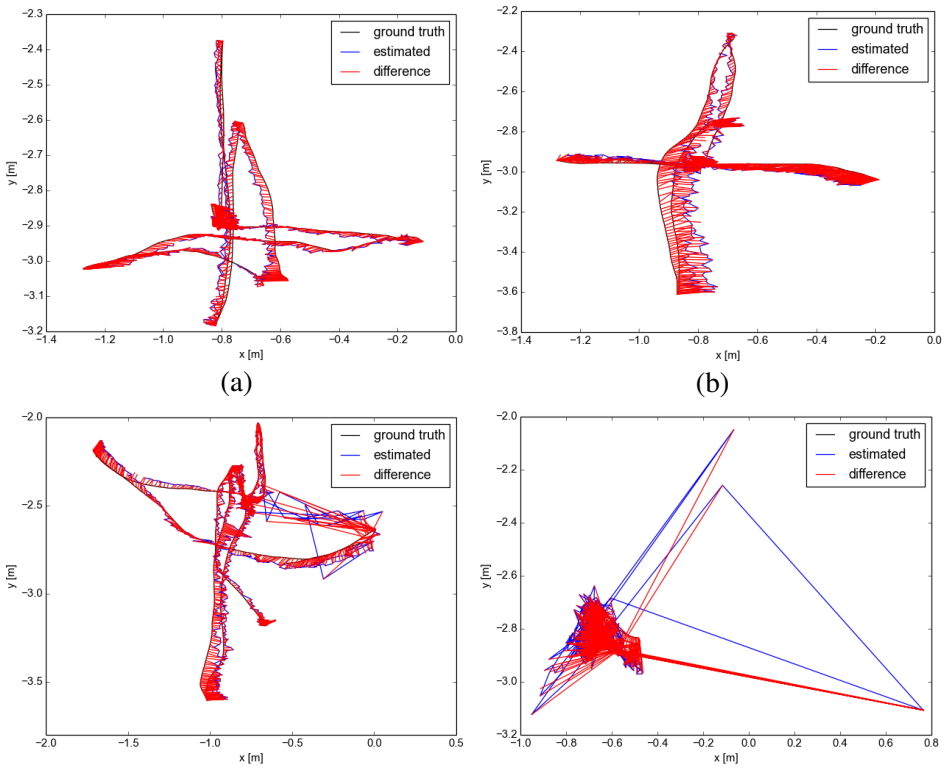
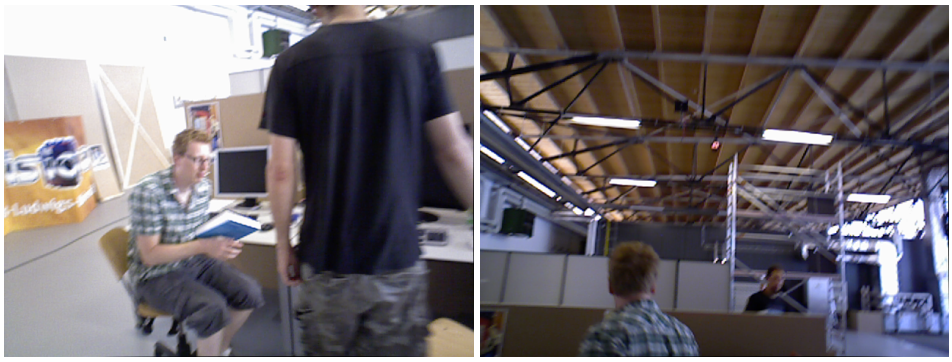


Figure 4.12: Quantitative results on TUM dynamic dataset. Here we show two good scenes and two bad scenes. (a) sitting_xyz, (b) walking_xyz, (c) walking_halfsphere, (d) walking_rpy.

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 The main results are presented in Table 4.5. It shows the developed method is somewhat robust in front of dynamic obstacles, with 62.5% average correct images. Fig. 4.12 (a) and (b) show that the present method stays close to ground truth trajectory. However, there are failure cases which cause the RMSE to be very large. For instance in *walking_halfsphere* as shown in Fig. 4.12 (c) and (d), it can be seen that several failure images caused large RMSE. Fig. 4.13 show the failure image examples. There are many possible reasons for the



(a) (b)

Figure 4.13: Failure cases on TUM dynamic dataset. (a) walking_halfsphere. Dynamic objects dominate the image and severe motion blur exists. (b) walking_rpy. No similar image in the training sequence and severe motion blur.

relocalization failure, such as severe motion blur, no similar image in the training sequence and dominating dynamic objects.

4.3.2 Outdoor camera relocalization

Fig.4.14 illustrates our framework of camera relocalization for outdoor scenes. Visual structure from motion is leveraged to autonomously generate training data (local features and their corresponding 3D point positions in the world coordinate). During training, regression forest is used to learn the correspondence between local features and 3D world coordinates. At test time, local features are extracted from query image, and then estimate their 3D correspondences from regression forests with backtracking. Finally the camera pose is estimated using PnP solver and RANSAC.

Dataset: The Cambridge Landmarks dataset [53] is used to evaluate the developed method. It consists of data in a large scale outdoor urban environment. Scenes are recorded by a smart phone RGB camera at 1920×1080 resolution. The dataset also contains the structure-from-motion (SfM) models reconstructed with all images to get the ground truth camera pose. The dataset exhibits motion blur, moving pedestrians and vehicles, and occlusion, which pose great challenges for camera

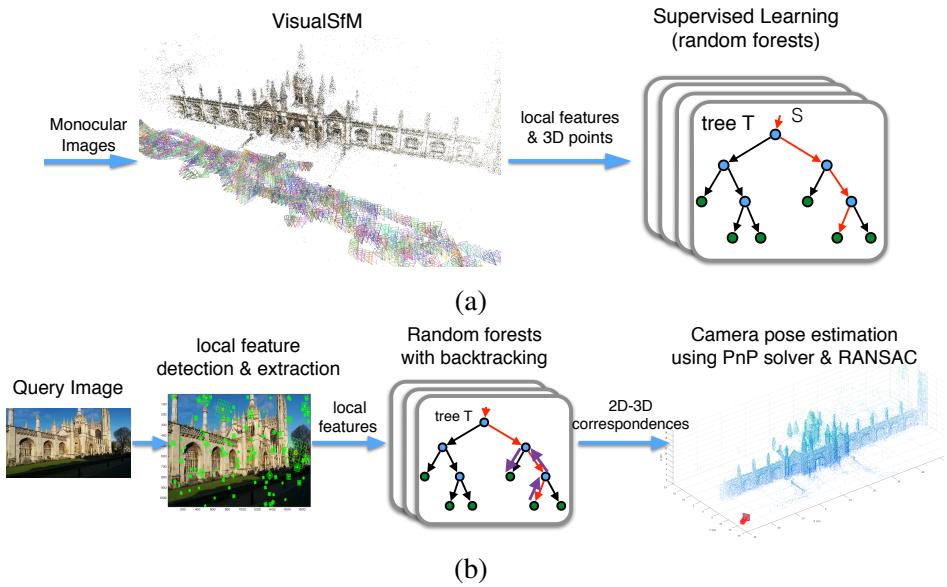


Figure 4.14: Overview of the present framework of SuperSIFT for camera relocalization (a) Visual structure from motion [115] is employed to autonomously generate training data (local features and their corresponding 3D point positions in the world coordinate). During training, regression forest is used to learn the correspondence between local features and 3D world coordinates. (b) At test time, local features are extracted from query image, and then their 3D correspondences are estimated from regression forests with backtracking. Finally the camera pose is estimated using PnP solver and RANSAC.

relocalization.

Baselines: Five state-of-the-art methods are used as our baselines: Active Search [87], PoseNet [53], Bayesian PoseNet [51], CNN+LSTM [108], and PoseNet+Geometric error[52]. Active Search employs a prioritized matching step that first considers features more likely to yield 2D-to-3D matches and then terminates the correspondence search once enough matches have been found. PoseNet trains a ConvNet as a pose regressor to estimate the 6-DOF camera pose from a single RGB image. Bayesian PoseNet improved the accuracy of PoseNet through an uncertainty framework by averaging Monte Carlo dropout samples from the posterior Bernoulli

Table 4.6: Camera relocalization results for the outdoor Cambridge Landmarks Dataset. The median performance for the developed method is shown against five state-of-the-art methods: Active Search without prioritization (w/o) and with prioritization (w/) [87], PoseNet [53], Bayesian PoseNet [51], CNN+LSTM [108], and PoseNet+Geometric loss[52].

Scene	Spatial Extent	Baselines						Ours
		Active Search (w/o) [87]	**Active Search (w/)**[87]	**PoseNet** [53]	**Bayesian PoseNet**[51]	**CNN+ LSTM**[108]	**PoseNet+ Geometric loss**[52]	**BTBRF**
King's	140x40m	0.59m, 0.48°	0.67m, 0.52°	1.92m, 5.40°	1.74m, 4.06°	0.99m, 3.65°	0.88m, 1.04°	**0.39**m, **0.36**°
Hospital	50x40m	1.25m, 0.71°	1.29m, 0.79°	2.31m, 5.38°	2.57m, 5.14°	1.51m, 4.29°	3.20m, 3.29°	**0.30**m, **0.41**°
Shop	35x25m	0.18m, 0.65°	0.17m, 0.53°	1.46m, 8.08°	1.25m, 7.54°	1.18m, 7.44°	0.88m, 3.78°	**0.15**m, **0.31**°
St Mary	80x60m	0.26m, 0.50°	0.29m, 0.55°	2.65m, 8.48°	2.11m, 8.38°	1.52m, 6.68°	1.57m, 3.32°	**0.20**m, **0.40**°
Average		0.57m, 0.59°	0.61m, 0.60°	2.08m, 6.83°	1.92m, 6.28°	1.30m, 5.52°	1.63m, 2.86°	**0.27**m, **0.39**°

distribution of the Bayesian ConvNet's weights. CNN+LSTM [108] uses a CNN to learn feature representations and LSTM units on the CNN output in spatial coordinates to capture the contextual information. PoseNet+Geometric error[52] which is based on PoseNet[53] uses geometric reprojection error instead of the manually tuned weighted loss as the regression loss to balance the rotational and positional quantities in a single scalar loss.

Error metric The median translational error and rotational error are used here as in previous work [51–53, 108] for fair comparison.

Main results and analysis Table 4.6 shows the median camera relocalization errors for the present method and the baseline methods. The results of PoseNet and Bayesian PoseNet are from the original papers. The results of active search and CNN+LSTM are from [108]. The present method considerably outperforms all these baselines for all scenes in terms of median translational and rotational errors. It is about an order of magnitude improvement in accuracy compared with PoseNet, five times as accurate as CNN+LSTM and twice as accurate as Active Search. Timings are not directly compared here as all these baselines are implemented on different high-end GPUs while the current implementation is on a single CPU core.

To gain insight on how backtracking helps to improve the accuracy, the camera

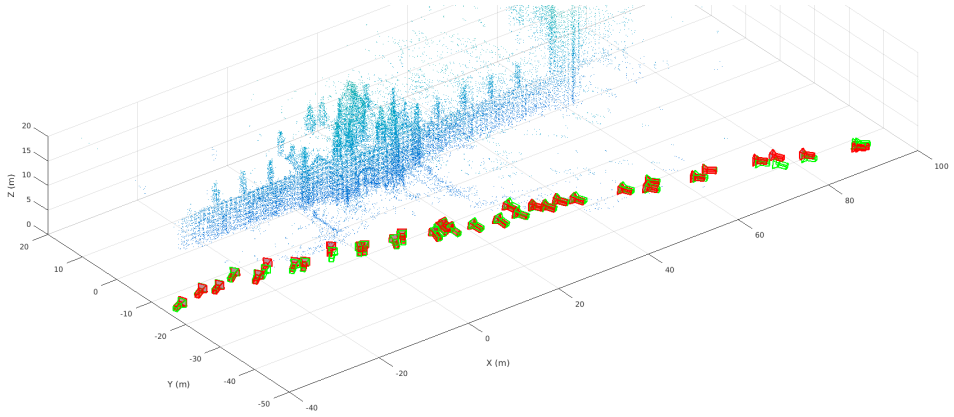


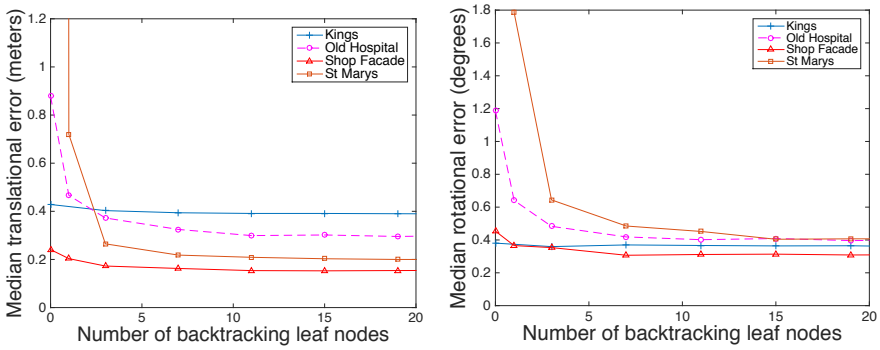
Figure 4.15: Qualitative results for the outdoor dataset Cambridge Landmarks, King's College. Best viewed in color. Camera frusta overlaid on the 3D scene. The camera poses are evenly sampled every ten frames for visualization. Camera frusta with hollow green frames show ground truth camera poses, while red ones with light opacity show the present estimated camera poses. The estimated camera poses of the developed method are very close to ground truth in spite of partial occlusion, moving objects, motion blur, large illumination and pose changes.

relocalization accuracy is plotted against backtracking leaf node numbers in Fig. 4.16. It shows that the camera relocalization errors significantly decrease within about 5 backtracking leaf nodes, which indicates that the relocalization accuracy can be improved by backtracking only a small number of leaf nodes.

Fig. 4.15 shows the qualitative results for the King's College Scene. We uniformly resampled the camera poses every ten frames to improve visualization.

4.3.3 Implementation details

The proposed method is implemented with C++ using OpenCV [7] on an Intel 3GHz i7 CPU, 16GB memory Mac system. The parameter settings for regression forest are: tree number $T = 5$; 500 (indoor) and 300 (outdoor) training images per tree; 5,000 randomly sampled pixels per training image (indoor); the number of SIFT features per image varies from around 500 to 1,500 (outdoor); the maximum depth of tree is 25; the maximum backtracking leaves is 16. For the test, the present



(a) (b)

Figure 4.16: Effect of backtracking leaf node numbers on camera relocalization accuracy in Cambridge Landmarks Dataset. (a) median translational error vs. backtracking leaf numbers (b) median rotational error vs. backtracking leaf numbers. Both median translational and rotational errors decrease with more backtracking leaf nodes though eventually level out.

unoptimized implementation takes approximately 1 second (indoor) and 2 seconds (outdoor) on a single CPU core. Most of the time is on the backtracking and the camera pose optimization part. It should be noted that the current implementation is not heavily optimized for speed and no GPU is used, which makes it possible to speed up with GPU implementation. There is empirical evidence supporting that a decent GPU implementation of the forest would not exceed 1ms [91], and even with backtracking it still can achieve fast response.

4.3.4 Limitations

The present BTBRF method has two limitations. First, it has to store mean local patch descriptors in leaf nodes, which doubles the trained model size. Second, the time complexity of the backtracking is proportional to the value of N_{max}, making the prediction process slower than the conventional regression forests method. It is important to find ways of adaptively setting the value of N_{max} to avoid unnecessary backtracking.

4.4 Conclusions

This chapter developed a sample-balanced objective and a backtracking scheme to improve the accuracy for camera relocalization. The proposed methods are applicable to regression forests both for RGB-D and RGB camera relocalization, showing state-of-the-art accuracy in publicly available datasets. Furthermore, we integrated local features in regression forests, advancing regression forests based methods to use RGB-only images for both training and test, and broadening the application to outdoor scenes while achieving the best accuracy against several strong state-of-the-art baselines. In the future, the present work may be implemented on a parallel computation architecture for higher efficiency.

Algorithm 3 Tree Prediction with Backtracking

Input: A decision tree T, testing sample S, maximum number of leaf to examine N_{max}

Output: Predicted label and minimum feature distance

procedure TREEPREDICTION(T, S, N_{max})
    $count \leftarrow 0$
    $PQ \leftarrow$ empty priority queue
    $R \leftarrow \emptyset$ ▷ the prediction result
    call TRAVERSETREE(S, T, PQ, R)
    while PQ not empty and $count < N_{max}$ **do**
        $T \leftarrow$ top of PQ
        call TRAVERSETREE(S, T, PQ, R)
    end while
    return R
end procedure
procedure TRAVERSETREE(S, T, PQ, R)
    if T is a leaf node **then**
        $dist \leftarrow distance(T.feature, S.feature)$ ▷ e.g., L2 norm
        if R is $\emptyset$ or $dist < R.dist$ **then**
            $\{R.label, R.dist\} \leftarrow \{T.label, dist\}$ ▷ update
        end if
        $count = count + 1$
    else
        $splitDim \leftarrow T.splitDim$
        $splitValue \leftarrow T.splitValue$
        if $S(splitDim) < splitValue$ **then**
            $closestNode \leftarrow T.left$
            $otherNode \leftarrow T.right$
        else
            $closestNode \leftarrow T.right$
            $otherNode \leftarrow T.left$
        end if
        add $otherNode$ to PQ
        call TRAVERSETREE($S, closestNode, PQ, R$)
    end if
end procedure

Chapter 5

Exploiting points and lines in regression forests for RGB-D camera relocalization

This chapter presents a camera pose relocalization method that uses both points and lines. Previous chapters used Perspective-n-Point (PnP) to estimate the pose of a calibrated camera from n 3D-to-2D or 3D-to-3D point correspondences. There are situations where point correspondences cannot be reliably estimated, for example in a texture-less office. In such scenarios, one can still exploit alternative geometric entities, such as lines, yielding the Perspective-n-Line (PnL) algorithms. This chapter presents a method that uses both point and line features to relocalize a RGB-D camera.

5.1 Introduction

Most existing camera relocalization approaches are designed with the assumption that many point features can be accurately tracked. However, it is not the case in the real world. Textureless areas are common in human environments and motion blur also often happens when moving the camera too fast. Recent camera relocalization methods based on deep learning [40, 51–53] are more robust when the sparse features such as SIFT [64] fail in the existence of motion blur or poorly

(a) (b)

Figure 5.1: Line segment example. (a) original RGB image (b) with LSD line features. In scenes with little texture and repetitive patterns which are typical in indoor environments, line features are more robust.

textured areas. However, the overall camera relocalization accuracy of these deep learning methods has almost an order of magnitude in error over the point-based methods or random forest based methods in publicly available datasets.

To solve this problem, this chapter proposes to use both line and point features in regression forests for camera relocalization. In motion blur and textureless areas where the points are struggled [85], the line segments provide important geometric information and are more robust features. Furthermore, 3D edges which are composed of many line segments are robust to viewpoint changes.

5.2 Related work

This section provides a literature review of the related work on the line and/or point feature for camera pose estimation. For more generated related work, please refer the reader to Sec. 2.1.1. Line segment detection and exploitation in many computer vision tasks such as camera pose estimation [81, 85], stereo analysis [15, 48] and crack detection [66] could date back to three decades [12, 50] and are still a very active area [36, 68, 85, 107]. Robust gradient orientations of the line segment rather than robust endpoints or gradient magnitudes play a crucial role in the line segment literature [11, 50, 85, 107]. Besides line segment detection, the present work is highly related to pose estimation using line and/or point features [26, 37, 81, 85].

5.3 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 encoding the 3D rotation and translation from camera coordinates to world coordinates.

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

To solve the problem, we propose to exploit both line and point features 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 points respectively. These two forests predict general points and line points in the testing. The second component is a camera pose optimization scheme using both point-to-point constraints and point-on-line constraints.

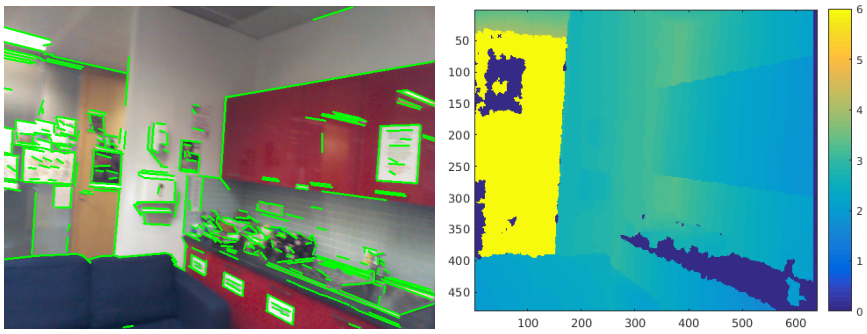
The novelty of our method lies in both the regression forests and camera pose estimation. First, point and line features are integrated in the training, modeling the uncertainty both in the line and point predictions. Second, the uncertainty of points and lines are simultaneously used to optimize the camera pose.

5.4 Regression forest with point and line features

5.4.1 Points sampling and scene coordinate labels

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

Line Segment Sampling: Directly back-projecting the two endpoints to 3D line using the depth information will cause large errors due to discontinuous depth on object boundaries or lack of depth information as shown in Fig. 5.2. To avoid this problem, the Line Segment Detector (LSD) is employed [107] to extract a set of 2D line segments $\mathtt{L} = \{\mathtt{l}_1, \mathtt{l}_2, \cdots\}$ from image $\mathtt{I}$ as shown in Fig. 5.1, and then uniformly sample points from the line as shown in Fig. 5.3. Using this sample scheme, one could discard the sample points whose depths are unavailable, and



(a) (b)

Figure 5.2: 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.

only back-project the remaining points to the camera coordinate. The 3D back-projected points could contain outliers which could be removed by RANSAC [28] and fit a 3D line.

Random points sampling: Besides the points sampled on the line segments, the present work also randomly samples general image points.

The same method is used to train both point and line point models.

Image features Here the proposed method uses the pixel comparison feature [93] that associates with each pixel location $\mathbf{p}$:

$$f_\phi(\mathbf{p}) = \mathtt{I}(\mathbf{p}, c_1) - \mathtt{I}(\mathbf{p} + \frac{\delta}{\mathtt{D}(\mathbf{p})}, c_2) \tag{5.1}$$

where δ is a 2D offset and $\mathtt{I}(\mathbf{p}, c)$ indicates an RGB pixel lookup in channel c. The ϕ contains feature response parameters $\{\delta, c_1, c_2\}$. The $\mathtt{D}(\mathbf{p})$ is the depth at pixel $\mathbf{p}$ in image $\mathtt{D}$ of the corresponding RGB image $\mathtt{I}$. Pixels with undefined depth and those outside the image boundary are marked and not used as samples for both training and test.

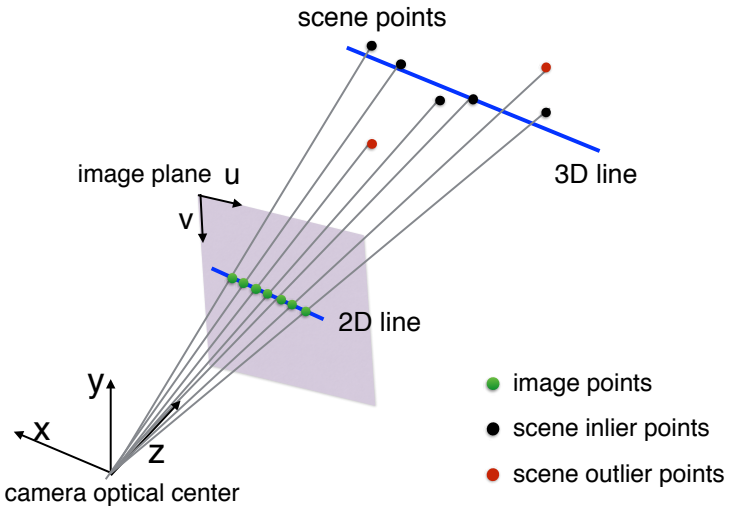


Figure 5.3: 3D line estimation based on sampling points. Within a pinhole camera model, the 2D image points are evenly sampled on a 2D image line and then back-projected on the scene coordinate to be 3D scene points. These 3D scene points contain outliers which could be removed by RANSAC to fit a 3D line in scene coordinate.

Besides the random pixel comparison feature, we also use Walsh-Hadamard Transform (WHT) features [43] to describe the local patch associated with the pixel $\mathbf{p}$.

Scene coordinate labels: For both random sampled points and points sampled on the line segment, 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 $\mathbf{m} = \mathtt{H}\mathbf{x}$.

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

process, which greatly simplifies the model learning and prediction process.

5.4.2 Regression forest training

A regression forest is an ensemble of T independently trained decision trees. Each tree is a binary tree structure consisting of split nodes and leaf nodes.

5.4.3 Weak learner model

Each split node i represents a weak learner parameterized by $\theta_i = \{\phi_i, \tau_i\}$ where ϕ_i is one dimension in features 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:

$$h(\mathbf{p};\theta_i) = \begin{cases} 0, & \text{if } f_{\phi_i}(\mathbf{p}) \leq \tau_i, \quad \text{then go to the left subset } S_i^L. \\ 1, & \text{if } f_{\phi_i}(\mathbf{p}) > \tau_i, \quad \text{then go to the right subset } S_i^R. \end{cases} \tag{5.2}$$

Here, τ_i is a threshold on feature $f_{\phi_i}(\mathbf{p})$. Although here we use random pixel comparison features as in Eq. 5.1, the weak learner model can use other general features to adapt application scenarios.

5.4.4 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:

$$\theta_i^* = \arg\max_{\theta_i \in \Theta_i} I_i(\mathtt{S}_i, \theta_i) \tag{5.3}$$

with

$$I_i = E(\mathtt{S}_i) - \sum_{j \in \{L,R\}} \frac{|\mathtt{S}_i^j(\theta_i)|}{|\mathtt{S}_i|} E(\mathtt{S}_i^j(\theta_i)) \tag{5.4}$$

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

$$E(\mathtt{S}) = \frac{1}{2} log((2\pi e)^d |\Lambda(\mathtt{S})|) \tag{5.5}$$

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

At the end of training, all samples reach leaf nodes. In a leaf node, the present work uses the mean shift method to estimate a set of modes. Each mode has a mean vector μ and a covariance matrix Λ to described the clustered 3D points. Meanwhile, the present method stores a mean vector of local patch descriptors for each mode. The local patch descriptor will be used to choose the optimal predictions.

5.4.5 Regression forest prediction

In testing, we use the backtracking technique in chapter 4 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. Fig. 4.4 shows the prediction process. To speed up, the maximum backtracking leaf number is set to be 8 instead of 16 as in the previous chapter. The point-on-line segment forests and the general point forests make point-on-line and general points predictions respectively.

5.5 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 $\mathscr{M}_i$ that concurrently best explains the transformed observation $\mathtt{H}x_i^c$:

$$(\mu_i^*, \Sigma_i^*) = \arg \max_{(\mu,\Sigma)\in\mathscr{M}_i} \mathscr{N}(\mathtt{H}x_i^c; \mu, \Sigma). \tag{5.6}$$

This can be optimized by minimizing the sum of Mahalanobis distances in world coordinates:

$$E_p = \sum_{i\in\mathscr{I}_p} \|(\mathtt{H}x_i^c - \mu_i^*)^T \Sigma_{x_i}^{*-1} (\mathtt{H}x_i^c - \mu_i^*)\|. \tag{5.7}$$

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 $\mathtt{H}^{-1}x_i^w$ and measures the Mahalanobis distance to the associated line L_i. This can be optimized by minimizing:

$$E_l = \sum_{i \in \mathscr{I}_l} \|(\mathtt{H}^{-1}x_i^w - Q)^T \Sigma_i^{-1} (\mathtt{H}^{-1}x_i^w - Q)\| \tag{5.8}$$

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

$$\Sigma_c = R\Sigma R^T \tag{5.9}$$

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

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

$$\mathtt{H}^* = \arg\min_H (E_p + E_l) \tag{5.10}$$

This energy is optimized by a Levenberg-Marquardt optimizer [71] and the result is used as the refined camera pose hypothesis in the next RANSAC iterations.

5.6 Experiments

This section evaluates the developed method on three publicly available datasets for camera relocalization against several state-of-the-art baselines.

5.6.1 Evaluations on Stanford 4 *Scenes* dataset

Dataset: See Secection 3.3.2 for the introduction of 4 *Scenes* dataset for a brief introduction or [103] for detailed description.

Baselines: Please refer Sec. 4.3.1 for details of ORB+PnP, SIFT+PnP, Random+SIFT [70], MNG [103]. The BTBRF is the present backtracking regression forest in Section. 4, but only random point features are used.

Table 5.1: Camera relocalization results for the indoor dataset. The percentage of correct frames (within 5*cm* translational and 5° angular error) of the developed method is shown using 4 Scenes dataset against four state-of-the-art methods: ORB+PnP, SIFT+PnP, Random+SIFT [70], MNG [103]. The best performance is highlighted.

	Spatial	**Baselines**					**Our Results**
Sequence	**Extent**	ORB+PnP	SIFT+PnP	Hybrid[70]	MNG[103]	BTBRF	PLForest
Kitchen	$33m^3$	66.39%	71.43%	70.3%	85.7%	92.7%	**98.9**%
Living	$30m^3$	41.99%	56.19%	60.0%	71.6%	95.1%	**100**%
Bed	$14m^3$	71.72%	72.95%	65.7%	66.4%	82.8%	**99.0**%
Kitchen	$21m^3$	63.91%	71.74%	76.7%	76.7%	86.2%	**99.0**%
Living	$42m^3$	45.40%	56.19%	52.2%	66.6%	99.7%	**100**%
Luke	$53m^3$	54.65%	70.99%	46.0%	83.3%	84.6%	**98.9**%
Floor5a	$38m^3$	28.97%	38.43%	49.5%	66.2%	89.9%	**98.8**%
Floor5b	$79m^3$	56.87%	45.78%	56.4%	71.1%	98.9%	**99.0**%
Gates362	$29m^3$	49.48%	67.88%	67.7%	51.8%	96.7%	**100**%
Gates381	$44m^3$	43.87%	62.77%	54.6%	52.3%	92.9%	**98.8**%
Lounge	$38m^3$	61.16%	58.72%	54.0%	64.2%	94.8%	**99.1**%
Manolis	$50m^3$	60.10%	72.86%	65.1%	76.0%	98.0%	**100**%
Average	—	53.7%	62.2%	59.9%	69.3%	92.7%	**99.3**%

Error metric: The correct percentage is used, which represents the proportion of test frames within 5*cm* translational and 5°angular error as the present error metric.

Main results and analysis Table 5.1 shows the main quantitative result on the 4 *Scenes* dataset. The present method PLForest considerably outperforms all the baselines and achieved the highest accuracy for all the sequences, with the average correct frame percentage of 99.3%. Fig. 5.4 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 exists still 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. 5.5. From the RGB images, it is seen that only a very little color information is available from which the present random pixel comparison features in Eq. 5.1 needed to differentiate each other.

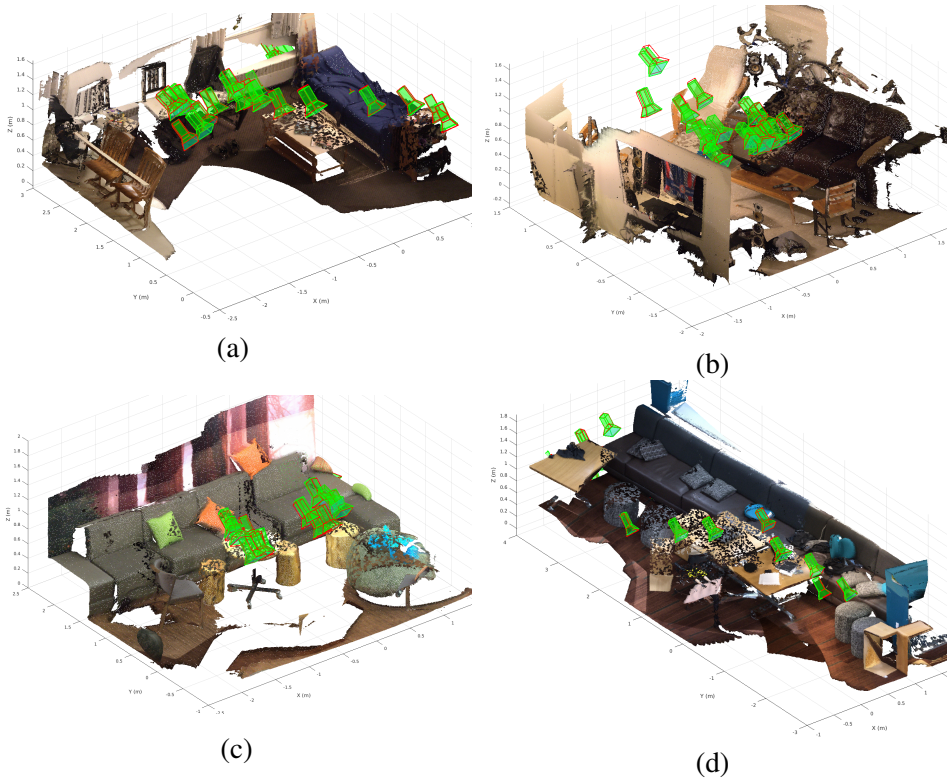
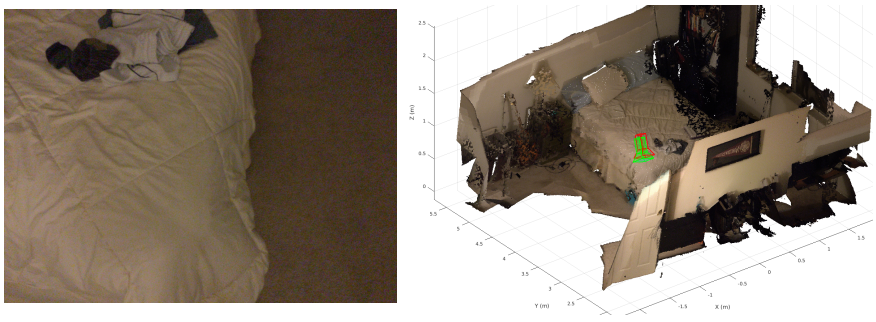


Figure 5.4: Qualitative results on *Stanford* 4 *Scenes* dataset. Best viewed in color. (a) apt1_living, (b) apt2_living, (c) office1_lounge, (d) office2_5b. The evenly sampled every 20th frames in 3D reconstructed scenes are shown for visualization. The ground truth (hollow red frusta) and the present 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.

5.6.2 Evaluations on Microsoft 7 *Scenes* dataset

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

Dataset Please refer to Sec. 3.3.1 for the introduction of Microsoft 7 *Scenes* dataset for a brief introduction or [93] for detailed description.

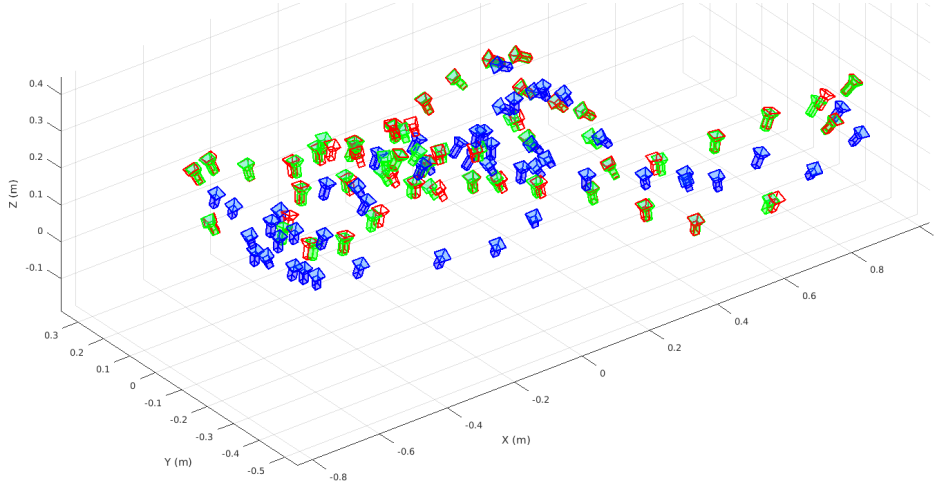


(a) (b)

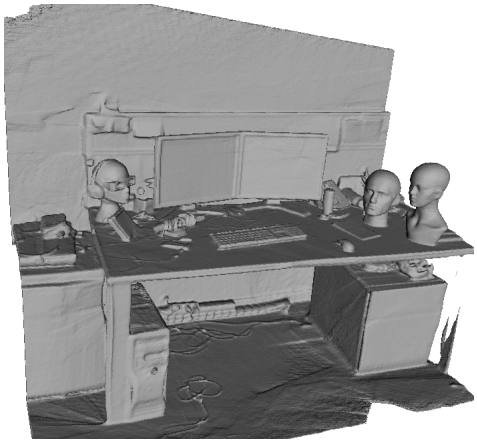
Figure 5.5: Large error examples on *Stanford* 4 *Scenes* dataset Apt2_Luke. (a) an RGB image, (b) ground truth (red) and estimated camera pose (blue) in the 3D scene. Dominating white color sheet and some color information caused large camera pose error.

Baselines and error metric: The developed method is compared against a sparse feature based method reported from [93], SCRF [93], a multi-output version of SCRF [38], an uncertainty version of SCRF [102] and an autocontext version of SCRF [6], the present backtracking regression forests in Chapter 4 in terms of correct frame percentage. The presented method is also compared with PoseNet and Bayesian PoseNet in terms of median translation error and rotation error.

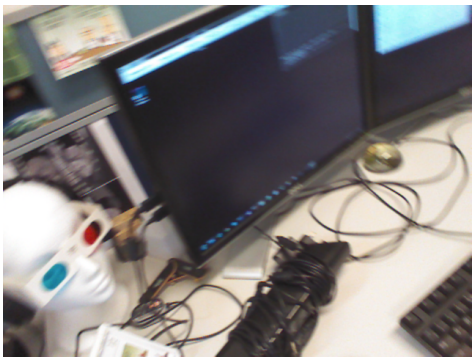
Main results and analysis: The main results are shown in comparison with strong state-of-the-art baselines in terms of correct frames in Table 5.2, and in terms of median performance in Table. 5.3. The present method achieves the best average accuracy in both criteria. Compared with *Stanford* 4 *Scenes* dataset, the average accuracy on *Microsoft* 7 *Scenes* dataset is relatively low. Several reasons may account for this: (i) the training and testing sequences in *Stanford* 4 *Scenes* are recorded at the same time as a whole sequence by the same person while the training and test sequences of *Microsoft* 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 image have better quality, and have better registration in *Stanford*4 *Scenes*. (iii) the scenes in *Microsoft* 7 *Scenes* are more challenging due to high ambiguity especially in *Stairs* scene and severe motion blur. Fig. 5.6 show some qualitative results for the *Heads* scene of the *Microsoft* 7 *Scenes*



(a)



(c)



(d)

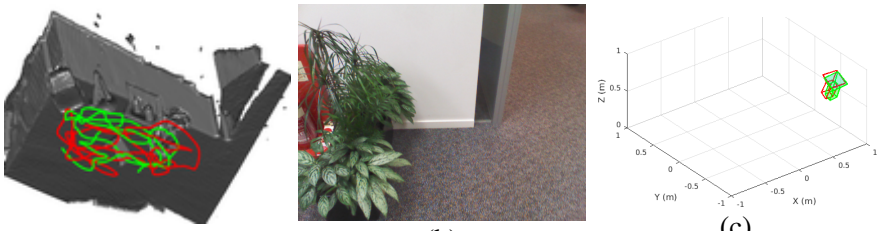
Figure 5.6: Qualitative results for the *Heads* scene in *Microsoft 7 Scenes* dataset. Best viewed in color. (a) Training (blue), test ground truth (red), and test estimated camera poses (green) are evenly sampled for every 20th images. The present estimated camera pose is similar to ground truth in both translation and rotation. The large errors occur in places where training poses are very different from test poses. (b) the 3D volume scene representation (c) large error image due to its pose is different from training poses, motion blur and texture-less areas.

Table 5.2: Relocalization results for the 7 *Scenes* **dataset**. Test frames satisfying the error metric (within 5cm translational and 5° angular error) are shown for the present method on all scenes against five strong state-of-the-art methods: SCRF [93], Multi[38], BTBRF, Uncertainty [102], AutoConext [6]. The best performance is highlighted.

		Baselines					**Ours**
Scene	**Space**	SCRF[93]	Multi[38]	BTBRF	Uncertainty[102]	AutoContext [6]	PLForest
Training		RGB-D	RGB-D	RGB-D	RGB-D	RGB-D	RGB-D
Test		RGB-D	RGB-D	RGB-D	RGB-D	RGB-D	RGB-D
Chess	$6m^3$	92.6%	96%	99.6%	99.4%	**99.6**%	99.5%
Fire	$2.5m^3$	82.9%	90%	**95.2**%	94.6%	94.0%	**97.6**%
Heads	$1m^3$	49.4%	56%	90.4%	**95.9**%	89.3%	95.5%
Office	$7.5m^3$	74.9%	92%	95.9%	**97.0**%	93.4%	96.2%
Pumpkin	$5m^3$	73.7%	80%	75.7%	**85.1**%	77.6%	81.4%
Kitchen	$18m^3$	71.8%	86%	89.4%	89.3%	**91.1**%	89.3%
Stairs	$7.5m^3$	27.8%	55%	60.8%	63.4%	71.7%	**72.7**%
Average	—	67.6%	79.3%	86.7%	89.5%	88.1%	**90.3**%

Table 5.3: Relocalization results for the 7 *Scenes* **dataset**. Median performance for the present method is shown on all scenes against five state-of-the-art methods: PoseNet [53], Bayesian Bayesian PoseNet [51], Active Search without prioritization [87], SCRF [93], BTBRF. The best performance is highlighted.

	Baselines					**Ours**
Scene	PoseNet[53]	Bayesian[51]	AC[87]	SCRF[93]	BTBRF	PLForest
Training	RGB	RGB	RGB	RGB-D	RGB-D	RGB-D
Test	RGB	RGB	RGB	RGB-D	RGB-D	RGB-D
Chess	0.32m, 8.12°	0.37m, 7.24°	0.04m, 1.96°	0.03m, 0.66°	0.015m, 0.59°	**0.014**m, **0.57**°
Fire	0.47m, 14.4°	0.43m, 13.7°	0.03m, 1.53°	0.05m, 1.50°	0.016m, 0.89°	**0.009**m, **0.48**°
Heads	0.29m, 12.0°	0.31m, 12.0°	0.02m, 1.45°	0.06m, 5.50°	0.020m, 1.84°	**0.008**m, **0.68**°
Office	0.48m, 7.68°	0.48m, 8.04°	0.09m, 3.61°	0.04m, 0.78°	0.018m, 0.75°	**0.017**m, **0.73**°
Pumpkin	0.47m, 8.42°	0.61m, 7.08°	0.08m, 3.10°	0.04m, 0.68°	0.023m, 0.84°	**0.019**m, **0.65**°
Kitchen	0.59m, 8.64°	0.58m, 7.54°	0.07m, 3.37°	0.04m, 0.76°	0.025m, 1.02°	**0.025**m, **1.02**°
Stairs	0.47m, 13.8°	0.48m, 13.1°	0.03m, 2.22°	0.32m, 1.32°	0.040m, 1.22°	**0.027**m, **0.71**°
Average	0.44m, 10.4°	0.47m, 9.81°	0.05m, 2.46°	0.08m, 1.60°	0.022m, 1.02°	**0.017**m, **0.70**°



(a) (b) (c)

Figure 5.7: Large error examples for $Office$ scene on $Microsoft$ 7 $Scenes$. (a) $Fire$ Scene. 3D Volume with training (green) and test (red) camera trajectory. It is seen that test sequence on the upper right part of the scene is very different from training sequence, which caused the large pose estimation errors for several frames. (b) an RGB image example corresponding to the upper right of the camera trajectory in (a). (c) qualitative comparison of ground truth (red) and estimated camera pose in similar spatial scale (2m in x axis, 2m in y axis and 1m of y axis) as (a).

dataset. The present estimated camera pose is similar to ground truth. The large error occur in places where the test poses are very different from training poses. Similar findings are also seen in some other scenes, such as the $Fire$ scene case as shown in Fig. 5.7.

5.6.3 Evaluations on TUM dynamic dataset

To demonstrate the efficacy and robustness against dynamic objects, the developed method is evaluated using TUM dynamic dataset.

Dataset: Section. 4.3.1 gives a brief introduction. Also, [98] provides a detailed description.

Baselines and error metric: Correct percentage of test frames are used within $5cm$ and 5°, median translational and angular error, and root mean squared error (RMSE) for Absolute Trajectory Error (ATE).

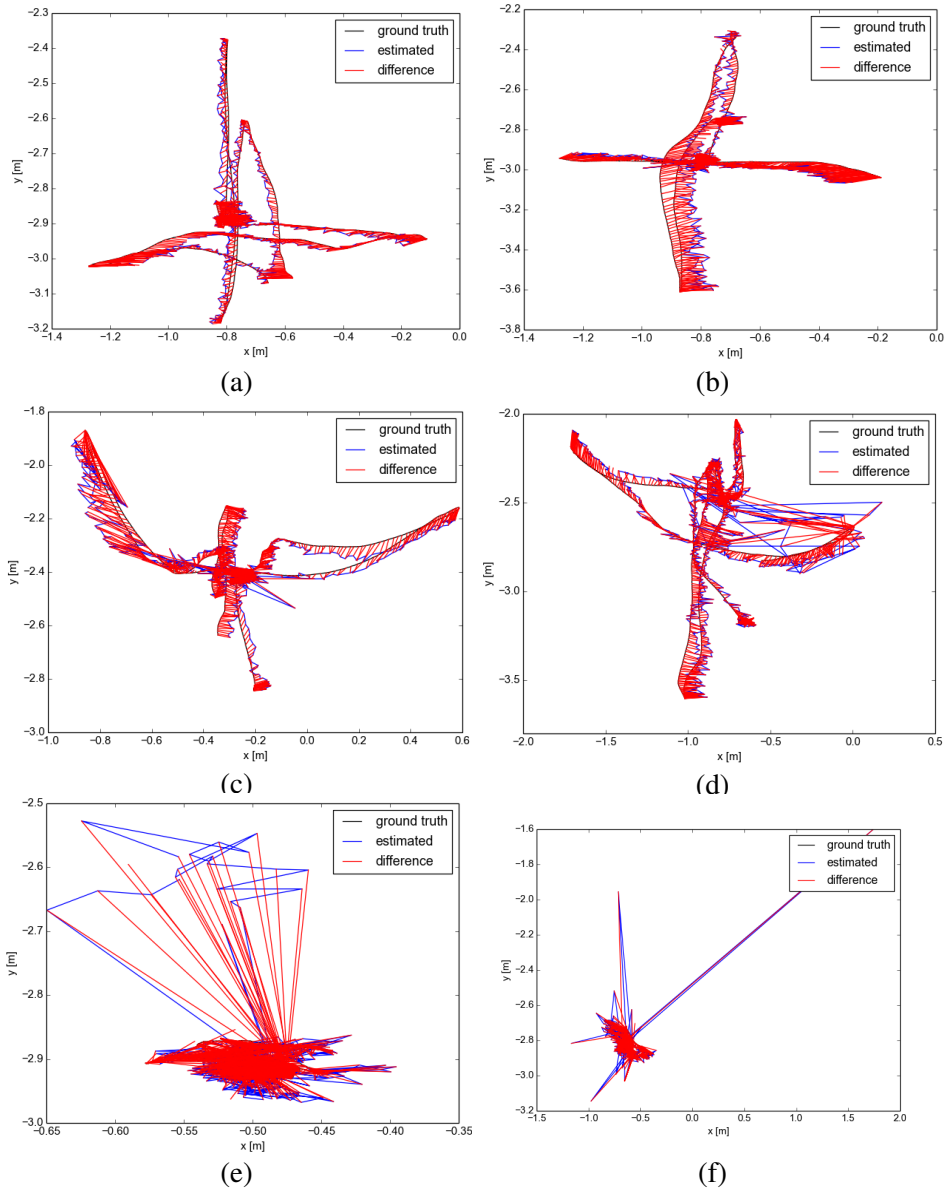


Figure 5.8: Quantitative results on TUM dynamic dataset. Two good scenes, two moderate scenes and two bad scenes are shown. The camera trajecotry is plotted on XY plane. Ground truth camera trajectory is shown in black line, estimated trajectory is shown in blue line and the difference is shown in red line. (a) sitting_xyz, (b) walking_xyz, (c) sitting_halfsphere, (d) walking_halfsphere, (e) sitting_rpy, (f) walking_rpy

Table 5.4: Camera relocalization results for the *TUM* dataset . Correct percentage performance, median performance, RMSE of ATE are shown.

	BTBRF			PLForest		
Scene	Correct	Median	RMSE	Correct	Median	RMSE
sitting_static	64.6%	0.015m, 0.99°	0.018m	**72.2**%	**0.012**m, **0.93**°	**0.016**m
sitting_xyz	70.2%	0.029m, 0.72°	**0.039**m	**74.1**%	**0.028**m, 0.73°	0.047m
sitting_halfsphere	**44.4**%	**0.056**m, **1.59**°	**0.046**m	39.8%	0.061m, 1.76°	0.072m
sitting_rpy	74.6%	0.029m, 0.98°	**0.065**m	**77.9**%	**0.026**m, **0.94**°	0.069m
walking_halfsphere	**61.7**%	**0.042**m, **1.03**°	**0.085**m	46.6%	0.052m, 1.26°	0.111m
walking_rpy	**53.7**%	0.047m, 1.14°	0.551m	53.4%	0.047m, **1.01**°	**0.169**m
walking_static	89.2%	0.019m, 0.49°	0.027m	**97.3**%	**0.017**m, **0.35**°	**0.021** m
walking_xyz	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

(a) (b)

Figure 5.9: Failure cases on TUM dynamic dataset. (a) walking_halfsphere. Dynamic objects dominates the image and severe motion blur exists. (b) walking_rpy. Large rotation angle changes.

Main results and analysis: Table 5.4 show the main results of the present method for TUM Dynamic dataset. Compared with the static *Stanford* 4 *Scenes* and *Microsoft* 7 *Scenes*, the performance is much lower due to high dynamics, but still could perform in most cases. The present method is more accurate than BT-BRF in terms of average correct frames and RMSE of ATE, and have the same average median performance. The present method could work satisfactorily with highly dynamic scenes, but still struggles in some extreme cases. Fig. 5.8 show the qualitative results on two good, two moderate and two bad sequences. Dynamic occlusions dominate the scene. They cause too few inliers and therefore lead to failure cases. Moreover, severe motion blur is also quite apparent in dynamic scenes due to the movement of both camera and objects such as in Fig. 5.9 (a). The other important failure case is large rotation angle changes, as shown in Fig. 5.9 (b). This is because the present random RGB comparison feature is not invariant to rotation.

5.7 Conclusions

This chapter presented 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 space 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.

Chapter 6

Mobile robot autonomous navigation in uneven and unstructured environments

This chapter presents an integrated software and hardware architecture for autonomous mobile robot navigation in 3D uneven and unstructured indoor environment. (Video link: https://www.youtube.com/watch?v=Bh-cHpbn3zEt=10s)

6.1 Introduction

Autonomous mobile robot navigation plays a vital role in self-driving cars, warehouse robots, personal assistant robots and smart wheelchairs, especially with in view of the shortage of a trained workforce and an ever-increasing aging population. Significant progress has been achieved in recent decades advancing the state-of-the-art of mobile robot technologies. These robots are operating more and more in unknown and unstructured environments, which requires a high degree of flexibility, perception, motion and control. Companies such as Google and Uber are developing advanced self-driving cars and expecting to move them into the market in the next few years. Various mobile robots are roaming in factories and warehouses to automate the production lines and inventory, saving workers from walking daily marathons [25]. Robot vacuums such as Roomba and Dyson360

eyes are moving around in the house to help clean the floors. Personal robots such as PR2 [46, 67] and Care-O-bot [82] have demonstrated the ability to perform a variety of integrated tasks such as long-distance navigation and complex manipulation.

Mobile robots navigating autonomously and safely in uneven and unstructured environments still face great challenges. Fortunately, more and more environments are designed and built for wheelchairs, providing sloped areas for wheeled robots to navigate through. However, little work focuses on an integrated system of autonomous navigation in sloped and unstructured indoor areas, especially narrow sloped areas and cluttered space in many modern buildings. The robots are required to safely navigate in narrow uneven areas such as those shown in Fig. 6.1 while avoiding static and dynamic obstacles such as people and pets.

This chapter presents an integrated software and hardware framework for autonomous mobile robot navigation in uneven and unstructured indoor environments that are designed for and shared with people. Fig. 6.3 shows a high-level system architecture of the present work. The robot first builds a 3D OctoMap representation for an uneven environment with the present 3D simultaneous localization and mapping (SLAM) using wheel odometry, 2D laser and RGB-D data. Then the present work projects multi-layer maps from OctoMap to generate the traversable map, which serves as the input for the present path planning and navigation. The robot employs a variable step size RRT approach for global planning, adaptive Monte Carlo localization method to localize itself, and elastic bands method as the local planner to close the gap between global path planning and real-time sensor-based robot control. The present focus is especially on efficient and robust environment representation and path planning. It is believed that reliable autonomous navigation in uneven and unstructured environments is not only useful for mobile robots but could also provide helpful insight on smart wheelchair design in the future.

6.2 Hardware and software platform

The development of a mobile robot system to work around us and assist people is the long-term goal. The main design principle for this system is that each hardware and software component could work as both a single module and a part of an inte-

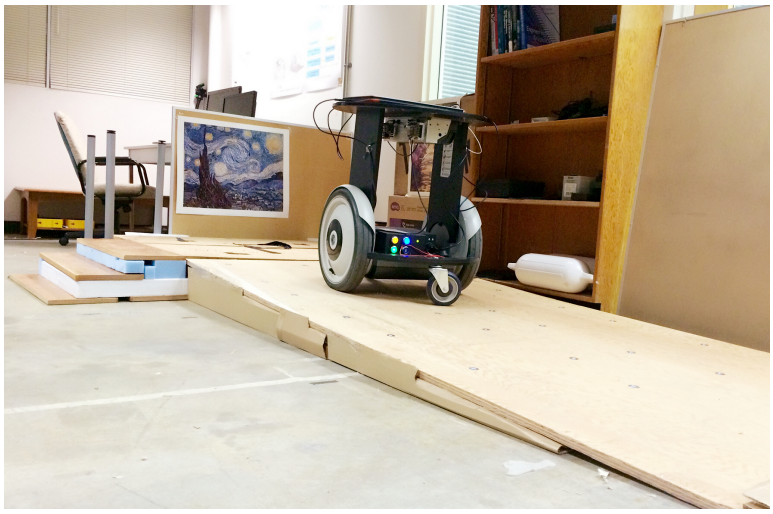


Figure 6.1: The robot is navigating up the slope to the goal at the higher platform. In the presence of staircases and slope, the robot builds a 3D representation of the environment for the traversable map, and then the robot can navigate through the slope and avoid the staircases to reach the goal efficiently and safely.

grated system. To realize this principle, these hardware components are assembled using screws and adjustable frames, while the software uses the Robot Operating System (ROS) [78].

Fig. 6.2 shows the hardware platform for the present robot. It includes a Segway RMP100 mobile base, an ASUS Xtion on the upper base, a Hokuyo UTM-30LX laser scanner mounted on the lower base, and a DELL laptop with Intel Core i7-4510U at 2GHz, 16GB memory (without GPU).

The software system is implemented in ROS Indigo release on top of an Ubuntu version 14.04LTS operating system. The 3D simulation experiments are performed on Gazebo [55]. Fig. 6.3 illustrates a high-level software architecture, and detailed key components of the software architecture will be described in Sec 6.3 and 6.5.

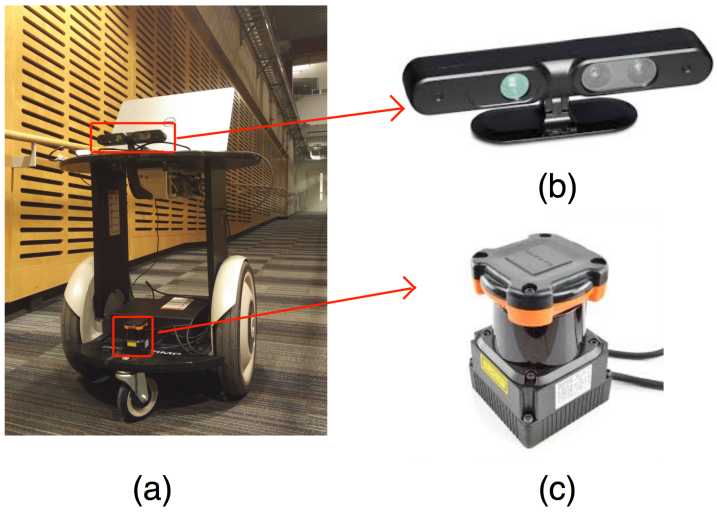


Figure 6.2: Robot hardware platform. (a) Segway robot in a sloped area. The robot base is segway RMP100 with custom installed casters for safety and onboard battery pack for providing power to sensors. (b) Xtion Pro Live RGB-D camera is capable of providing 30Hz RGB and depth images, with 640x480 resolution and 58 HFV. (c) Hokuyo UTM-30LX laser scanner with range 10m to 30m, and 270° area scanning range for localization.

6.3 Environment representation

A consistent and accurate representation of the environment is a crucial component of autonomous systems as it serves as input for motion planning to generate collision-free and optimal motions for the robot.

6.3.1 3D SLAM

3D SLAM pipelines commonly consist of localization and mapping components. Localization is the process to estimate robot pose, and mapping (or fusion) involves integrating new sensor observations into the 3D reconstruction of the environment. For an environment which lacks of feature points (such as SIFT, ORB) the present work uses a 2D laser to localize and an RGB-D camera to provide point clouds for building 3D maps. For environment which has rich feature points, the present work uses RGB-D camera to both localize the robot and build the map.

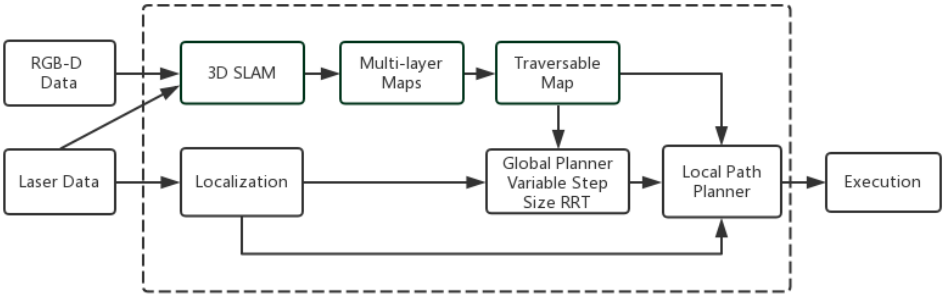


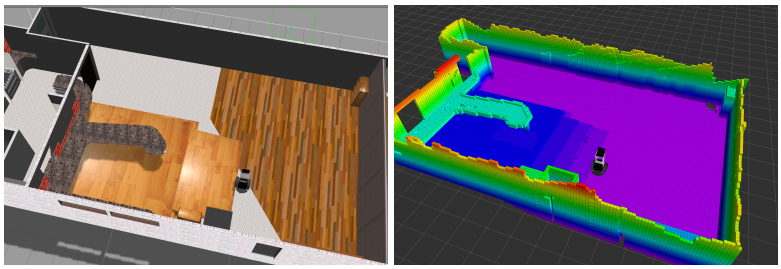
Figure 6.3: High-level system architecture. The robot first builds a 3D OctoMap representation for uneven environment with the present 3D SLAM using wheel odometry, 2D laser and RGB-D data. Multi-layer maps from OctoMap are used for generating the traversable map, which serves as the input for autonomous navigation. The robot employs a variable step size RRT approach for global planning, adaptive Monte Carlo localization method to localize itself, and elastic bands method as the local planner to gap the global planning and real-time sensor-based robot control.

3D SLAM using wheel odometry, 2D laser and RGB-D camera

To overcome the challenge that vision-based SLAM is not robust when the environment lacks local features, the present work employs wheel odometry, a 2D laser and an RGB-D camera concurrently to complement each other.

The present 3D SLAM framework builds on top of Karto SLAM [2], a robust method containing scan matching, loop detection, Sparse Pose Adjustment [57] as the solver for pose optimization and 2D occupancy grid construction. Karto SLAM takes in data from the laser range finder and wheel odometry. It is the best performing ROS-enabled SLAM technique in the real world, being less affected by noise [86, 106] compared with other 2D SLAM methods, such as gmapping [35], Hector SLAM [56], Lago SLAM [13], and GraphSLAM [101].

Instead of using Karto SLAM's default 2D occupancy map, which we found cannot represent an uneven environment reliably, the present work builds the environment based on OctoMap [47]. It is a probabilistic, flexible, and compact 3D mapping method which can represent a free, occupied and unknown environment. At each time step, the algorithm accepts point clouds of the environment from the



(a) (b)

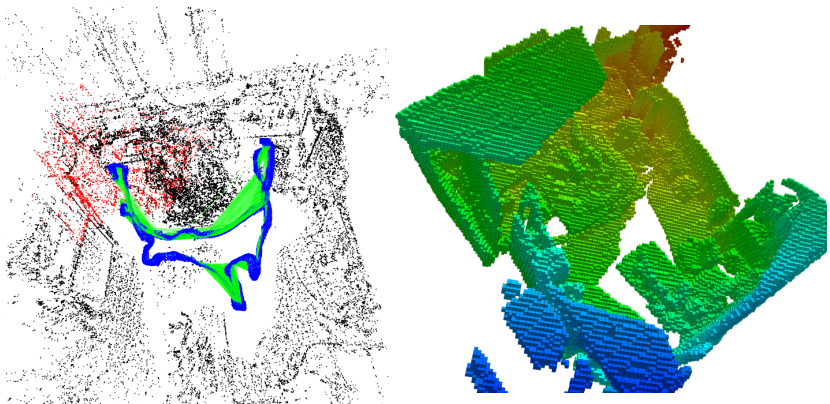
Figure 6.4: 3D environment representation of simulated world. (a) Simulated environment model in Gazebo (b) 3D OctoMap environment built by our 3D SLAM using wheel odometry, a 2D laser scanner and an RGB-D sensor.

RGB-D sensor and the localization information. Fig. 6.4(b) shows an OctoMap representation of the simulated world generated by our 3D SLAM. Note that free space is explicitly modeled in the experiment but is not shown in the figure for clarity.

3D SLAM using RGB-D camera

For environments that have rich feature points, the present work uses an architecture based on top of a variant of ORB-SLAM [72] and Octomap [47]. In order to satisfy our need in the subsequent global localization and path planning procedures, the present work has two main differences compared with the original ORB-SLAM: more keyframes and 3D probabilistic OctoMap representation.

Different from ORB-SLAM which uses the *survival of the fittest* strategy for selecting keyframes, the present work keeps as many keyframes as possible in order to obtain more camera poses for the training data in the camera relocalization in the autonomous navigation stage. As for the regression forest based global localization stage detailed in Sec. 6.4.1, the time spent on global pose optimization in SLAM may be sacrificed to obtain more keyframe poses for more training data and more accurate global localization. Therefore, the present work includes all the edges provided by the *covisibility graph* [97] rather than using the *essential graph* in ORB-SLAM.



(a) (b)

Figure 6.5: Octomap representation for $fr1/room$ of TUM RGB-D SLAM benchmark [98] with visual SLAM. (a) the sparse map from original ORB-SLAM [72], map points (black, red), keyframes (blue). (b) OctoMap representation

The output map from ORB-SLAM is the sparse feature map which is shown in Fig. 6.5 (a). It cannot directly used for path planning and navigation. To overcome this problem, the present work employs the keyframe camera poses described above and point cloud as the input for OctoMap to generate 3D environment representation. The raw point cloud generated from whether from RGB and depth image or directly from consumer RGB-D camera is very noisy due to varying point densities, measurement error, wrong registrations etc, posing great challenges for the subsequent surface/volume reconstruction stage. Therefore, the present work employs outlier removal based on the distribution of point to neighbors distances. For each raw point, the mean distance from a number of its neighbors is computed. It is assumed that the resulted distances are Gaussian distributed, and all points whose mean distances are larger than a threshold (One standard deviation is used here) are treated as outliers and removed. Then the inlier point cloud is sent to the OctoMap. The result is shown in 6.5 (b).

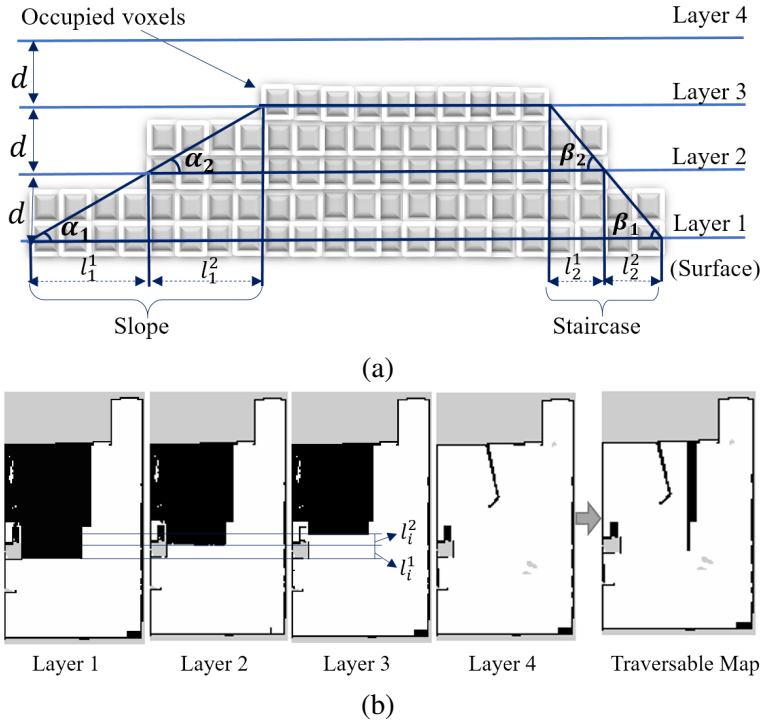


Figure 6.6: Generation of traversable map from multilayer maps for the Gazebo Caffe environment. (a) slope and staircase visualization with occupied voxels, (b) multi-layer maps and traversable map. In the traversable map, the staircases are occupied space, while the slope area except the edge is free space, which is safe for robot to navigate. For details please refer to Sec. 6.3.2.

6.3.2 Multilayer maps and traversable map

After the environment is represented by OctoMap as mentioned above, the OctoMap is cut from bottom to top with several layers depending on the environment and the required accuracy. Then these multilayers are integrated into a traversable map, which is safe and efficient for the robot to navigate through. Both the multilayer maps and the traversable map are represented as 2D grid occupancy maps, in which black represents occupied space, white as free space, and gray as unknown space. A flag F is used to mark the map traversablility and decide whether an area

is traversable:

$$F = \begin{cases} 0, & \text{if } \alpha_i \geq \theta, \quad \text{untraversable area} \\ 1, & \text{if } \alpha_i < \theta, \quad \text{slope, traversable area} \end{cases} \tag{6.1}$$

where θ represents the angle threshold which can be set according to the robot climbing capability, and α_i is the ith layer gradient:

$$\alpha_i = arctan\frac{d}{l_i^j} = arctan\frac{d}{rv_i^j} \tag{6.2}$$

where d represents the distance between projected layers as shown in Fig. 6.6 (a). It can be set to be as small as the resolution of the OctoMap for accurate representation. l_i^j represents the length of the edge difference between different layers, and it can be obtained through the number of voxels v_i^j in that area and OctoMap resolution r. Take the environment in Fig. 6.4 (a) as an example, the first four layers of the 2D map from the OctoMap is shown in Fig. 6.6 (a) and the left four maps in Fig. 6.6 (b). Then the gradient between layers are checked. For instance, in Fig. 6.6 (a), both α_1 and α_2 are less than θ, and that area is marked as a slope while β_1 and β_2 are larger than θ, And that area is marked as an untraversable area. At the same time, the left and right edges of the slope are marked as occupied for safety. The integration of all these multi-layer maps will generate the traversable map as shown in Fig. 6.6 (b).

6.4 Localization

Mobile robot localization comprises the problem of determining the pose of a robot relative to a known environment [101]. Local and global localization are two types of localization [101]. The former aims at compensating for robot motion noise which is usually small, and it requires the initial location of the robot. While the latter can localize without any information about the previous motion. In global localization, the initial pose and previous motion information are unknown [101]. Global localization is more difficult than local localization. However, it plays a critical role as it can help to solve the kidnapped robot problem and allow the robot to recover from severe pose errors, which is essential for truly autonomous robots.

6.4.1 Regression forests based global localization

The present work uses the backtracking regression forests method in Chapter 4 as it is a faster than the method in Chapter 5, and more appropriate for real robot navigation which needs fast response. To further increase the speed, the backtracking number 4 is used. As shown in Fig. 4.8, the accuracy increases quickly as the backtracking leaf node number increases before 4, and slowly thereafter, eventually saturating. In order to communicate with other modules, the present work uses ROS subscriber to accept RGB and depth images from RGB-D camera, and ROS publisher to send the camera pose to the path planning unit.

6.4.2 Local localization

The present robot employs adaptive Monte Carlo localization (AMCL) [29] for local localization. AMCL is a method that uses a particle filter to track the pose of the robot with a known map. It takes laser scans, the wheel odometry information and the traversable map, and then outputs pose estimation.

6.5 Planning and execution

6.5.1 Global and local planning

The traversable map serves as the input for the present planning and navigation framework. The present work applies an efficient variable step size RRT planner [109] as the global planner to construct a plan in a configuration space $\mathscr{C}$ to reach the goal. The global planner generates a list of way-points for the overall optimal path from the starting position to the goal position through the map, while ignoring the kinematic and dynamic constraints of the robot. Then the local planner takes into account the robot kinematic and dynamic constraints, and generates a series of feasible local trajectories that can be executed from the current position, while avoiding obstacles and staying close to the global plan. the present work uses Elastic Bands method [79] as our local planner to close the gap between global path planning and real-time sensor-based robot control.

6.5.2 Plan execution

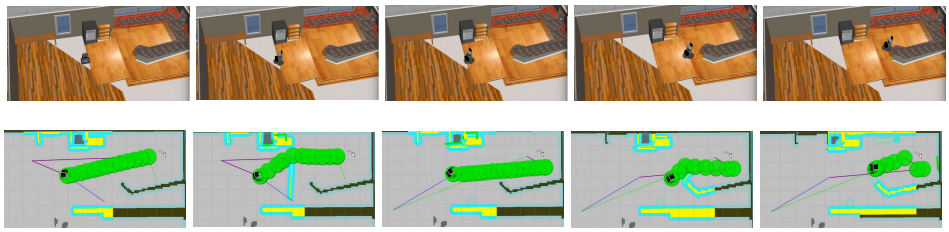
During execution, the short and smooth path from the local planner is converted into motion commands for the robot mobile base. The local planner computes the velocities required to reach the next pose along the path and checks for collisions in the updated local cost-map.

6.6 Experiments

Extensive experiments are conducted to evaluate the developed system both in simulation and real-world, demonstrating its efficacy for real-time mobile robot autonomous navigation in uneven and unstructured environments. Note that the present work also focuses on an integrated system besides the component methods.

6.6.1 Simulation experiments

Environment representation: The simulation is conducted in Caffe using an environment of size 13m x 10m x 3m, which was built in Gazebo. A visualization is shown in Fig. 4. The simulated robot model is equipped with wheel odometry, a 2D laser range finder and an RGB-D sensor. For this simulated environment, the present work only uses SLAM with a 2D laser and an RGB-D camera as the Gazebo environment has very few visual features. The robot is tele-operated to move around the environment. Fig. 6.4 (b) shows the OctoMap generated by the present 3D SLAM with 2D laser and RGB-D camera. The resolution of the OctoMap is set as 0.05m to strike a trade-off between speed and accuracy, and the threshold on the occupancy probability is 0.7. Note that free space is explicitly modeled in the experiment but is not shown in the figure for clarity. To evaluate the present 3D SLAM with only the RGB-D camera, the present work uses the standard TUM RGB-D SLAM benchmark [98]. Fig. 6.5 (b) shows our Octomap representation for the $fr1/room$ scene. For the Gazebo environment, the present work chooses four layers of projected maps as shown in the left part of Fig. 6.6 (b). It can be seen that the staircase area shows steeper changes than the sloped area. Therefore, in the generation of the present traversable map, the steeper changes are

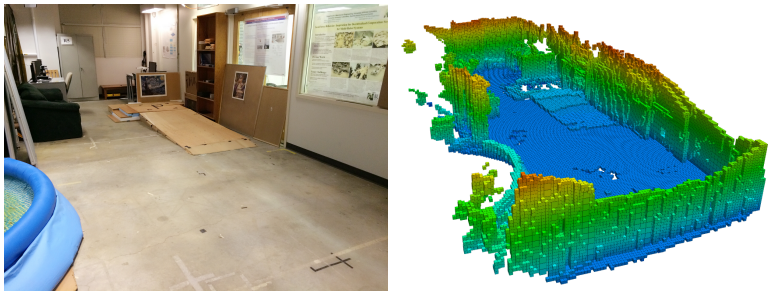


(a) (b) (c) (d) (e)

Figure 6.7: Autonomous navigation in the simulated environment. The first row shows images of autonomous navigation in Gazebo, and the second row shows the Rviz views of the process.

treated as untraversable area and the area with slower changes as the slope. The right part of Fig. 6.6 (b) shows the generated traversable map, which will be used for the present robot autonomous navigation.

Autonomous navigation For autonomous navigation, the present work uses the traversable map and sensors data as the input for path planning and localization. Various autonomous navigation experiments are conducted in which the robot starts and reaches goals at different positions. Fig. 6.7 illustrates an example of the present robot autonomous robot navigation process in which the robot starts at lower ground and reaches the goal in the higher platform. The blue and green lines show the exploration process of the global planner. This process ended after a global path connecting start point and end point is found, as shown by the red line. The local path planner can smooth the global path if it is not optimal, and the optimal path is marked as a concatenation of green bubbles. Fig.6.7 (b) – Fig.6.7 (c) show the re-planning process when an obstacle is found. If the height of a slope is higher than the laser on the robot, then the robot would expect that there is an obstacle ahead. The robot can replan its path when this occurs. Fig.6.7 (d) – Fig.6.7 (e) show the re-planning process of the local planner when a real obstacle appears. The robot can avoid the obstacle and reach the target position.

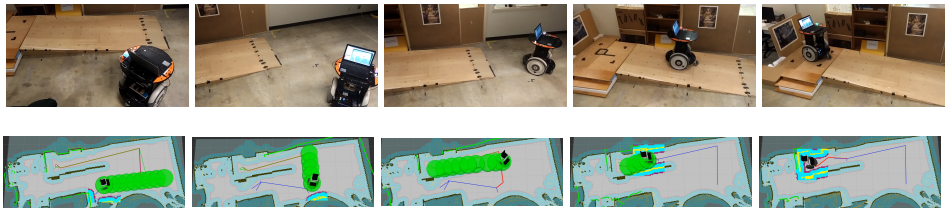


(a) (b)

Figure 6.8: Real environment. (a) a photo of the real environment. (b) 3D representation of the environment with OctoMap. Only occupied voxels are shown for visualization.

(a) (b) (c) (d) (e)

Figure 6.9: Multilayer maps and traversable map for real environment. (a)-(d) multiple projected layers from OctoMap, (e) the traversable map. The staircases and slope edge are occupied while the slope is free space.



(a) (b) (c) (d) (e)

Figure 6.10: Robot autonomous navigation example in real environment. The first row shows images of robot autonomous navigation, and the second row shows screenshots of Rviz views of the process.

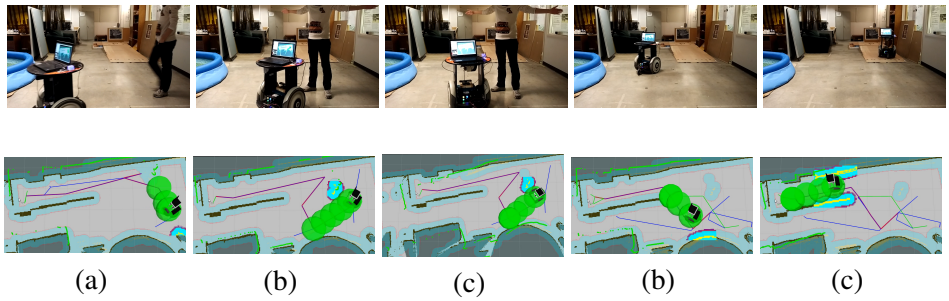


Figure 6.11: Dynamic obstacle avoidance. (a) a dynamic obstacle is approaching the robot. (b) The human suddenly blocks the way in front of the robot. (c) The robot changes direction to avoid the human.

6.6.2 Real-world experiments

Extensive experiments are conducted with a Segway Robot in a real-world environment of the Industrial Automation Laboratory. Fig. 6.8 shows the present real environment with dimension of $11m \times 5m \times 3m$, and the 3D OctoMap generated from the present 3D SLAM with wheel odometry, a 2D laser and an RGB-D data. At this 3D SLAM stage, we teleoperate the present Segway robot to move around the environment. Compared with the simulated environment, the real environment features challenges such as narrower slopes, different lighting conditions, and glass windows. The real robot is much more noisy due to long-term wear-and-tear, uncalibrated wheel odometry, and the disturbance of casters. The sensors are also more noisy. Therefore, the OctoMap of the real environment and projected multi-layer maps in Fig. 6.9 tend to have some distortions and inconsistent registrations compared with the simulated environment. The noisy environment, robot and sensors demand robustness from the present system. The resolution of the OctoMap is experimentally set to 0.05m to strike a trade-off between speed and accuracy. Four layers of projected maps are used to generate the traversable map in the real environment. The traversable map in Fig. 6.9 (e) will serve as the input for path planning and localization in autonomous navigation.

One important parameter for robot navigating up a slope is the size of the local costmap [65]. If the local costmap size is set too large, the far away ray sweep of the laser scan on the slope will intersect with the higher part of the slope, creating

an obstacle in the local costmap, which will block the robot's path. It is especially obvious in the more noisy real experiment in which the robot is prone to fail in localizing itself than in simulated environment. Facing localization failure, the robot will initiate the rotation recovery mode, in which the robot rotates and laser-sweeps in a circle. If the local costmap size is too small, the obstacles may not be considered in time, which may cause unsafe collision. The present work experimentally determined the costmap of 3 meters width and 4 meters length to work best for the use in this use case.

6.7 Conclusions

This chapter presented an integrated software and hardware architecture for autonomous mobile robot navigation in 3D uneven and unstructured indoor environments. This modular and reusable software framework incorporated capabilities of perception and navigation. The present work employed an efficient path planner which uses variable step size RRT in 3D environment with an octree-based representation. The present work generated a safe and feasible 2D map from multi-layer maps of 3D OctoMap for efficient planning and navigation. It is demonstrated and evaluated the developed integrated system in both simulation and real-world experiments. Both simulation and real-robot experiments demonstrated the efficacy and efficiency of the present methods, providing some insight for more autonomous mobile robots and wheelchairs working around us. Future work may integrate more accurate global localization and optimized path planning methods into the present system. Furthermore, it is desirable to explore the possibility of using 3D semantic scene parsing to understand the uneven areas.

Chapter 7

Conclusions and future work

It is good to have an end to journey toward; but it is the journey that matters, in the end. — Ursula K. Leguin

7.1 Conclusions

This thesis addressed the problem of image based RGB/RGB-D camera relocalization, one of the core problems in computer vision and robotics. The thesis presented a number of techniques based on regression forests that could improve the performance of camera relocalization. Experiments against many strong baselines using various of publicly available datasets demonstrated the efficacy of the developed approach, showing superior performance with respect to the state-of-the-art.

This thesis also presented an integrated software and hardware architecture for autonomous mobile robot navigation in 3D uneven and unstructured indoor environments. This modular and reusable software framework incorporated capabilities of perception and navigation. The present work employed an efficient path planner which used variable step size RRT in 3D environment with an octree-based representation. The presented work also generated a safe and feasible 2D map from multi-layer maps of 3D OctoMap for efficient planning and navigation. The work demonstrated and evaluated the developed integrated system in both simulation and real-world experiments. Both simulation and real-robot experiments demonstrated the efficacy and efficiency of the developed methods, providing some insight for

more autonomous mobile robots and wheelchairs working around us.

7.2 Future directions

This thesis is concluded by proposing several directions for possible future exploration.

7.2.1 Speed up image based localization

The speed of the developed method can be improved for real-time robotic applications. The bottleneck is in the tree predictions, which has to backtracking many leaf nodes. One solution is by using a GPU decision tree implementation [91], which greatly speeds up the prediction through parallel computing. An alternative method is to sample a few testing points but still correctly estimate the camera poses. It requires that the samples points have more precise predictions. Furthermore, adaptively selecting the backtracking number will reduce unnecessary backtracking times and will speed up the prediction process.

7.2.2 Transfer learning

For a new environment, the developed method requires an offline training process, which usually costs much time (up to hours). Transfer learning can speed up the training process so that the method would become useful. Transfer learning aims to extract knowledge from one or more source tasks and apply the knowledge to a target task. A major assumption in statistical machine learning is that training data and test data must be in the same feature space, independently and identically distributed. When the distribution changes, most statistical models need to be rebuilt using newly collected training data. In many real-world applications, it is expensive or impossible to recollect the required training data again and rebuild the models. Transfer learning can be very beneficial in such cases.

An early survey [76] provides a comprehensive overview of transfer learning for classification, regression and clustering. Recently, the transferability of deep convolutional neural networks has been successfully demonstrated for object recognition [118], place recognition [99] and camera relocalization [53] through pre-training on large source data and fine-tuning on small target data. However,

it is not adequately studied in the context of random forests, especially for regression. Lee *et al*.[113] build a prediction model incrementally from a partial decision tree model. Goussies *et al*.[34] present a transfer learning decision forests method to recognize gestures and characters, where random forests are used as classifiers. To the best of the present knowledge, random forests as a regressor for camera relocalization have not been studied.

It is believed that valuable knowledge can be transferred from the source tasks $S = \{S_1, S_2, ...S_N\}$ to the target task T in the camera relocalization, as it happens with localization in humans. For instance, it is easier to learn to localize in the new environment if a similar environment has already been localized. In other words, there exists latent information (or experience) that is common in similar environment that can be understood as common sense.

In the case of transfer learning of regression forests, it is believed that there are common tree structures and parameters among different scenes, which can be shared. This information can then be included in the process of growing each tree of forest when it encounters new information. The main idea, therefore, is to find the parameters θ of weak learners for each tree in order to obtain a partition of the feature space.

7.2.3 Active image-based localization learning for autonomous robot

Currently all the methods based on regression forests and deep learning only consider the image-based localization re-localization for the passive camera sensor, which exclusively addresses the localization estimation based on an incoming stream of sensor data rather than for the camera sensor mounted on an active robot. These methods assume that neither the camera motion, nor the pointing direction of the camera sensor can be controlled. Therefore they do not exploit the opportunity to control robot's active motion during localization. Active localization needs to set the robot's motion direction and determine the pointing direction of the sensors to more efficiently localize the robot [10]. It is a promising research direction for more efficient and more robust camera localization methods. Active localization for mobile robot navigation was first studied in [10] on top of Markov localization using sonar sensor. Active localization has been studied in the spare-feature-

based camera re-localization methods, but has never been studied in the machine learning-based camera re-localization methods. As in sparse-feature feature based methods, the number of sparse features is an important signature to indicate the localization quality. When the number of sparse features is small, the robot will change motion till enough sparse features appear. Here, in the machine learning-based method, it is difficult to have a criterion to indicate the localization quality while the robot is navigating through the environment. It will be important to find the criteria to decide ”where the robot should move” and ”where the camera should look” for these machine learning-based camera re-localization methods so as to best localize the robot. In the current study, severe motion blur and repetitive environments such as corridors or stairs (as shown in the 7 *Scenes* dataset in previous chapters) are very adverse situations for machine learning-based camera re-localization methods.

Bibliography

[1] VXL C++ libraries for computer vision research and implementation. http://vxl.sourceforge.net. Accessed: 2016-08-03.

[2] Karto SLAM, ROS package. accessed Nov, 2016. [online], *wiki.ros.org/slam_karto*. 2010.

[3] Y. Amit and D. Geman. Randomized inquiries about shape: An application to handwritten digit recognition. Technical report, CHICAGO UNIV IL DEPT OF STATISTICS, 1994.

[4] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. 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.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding (CVIU)*, 110(3): 346–359, 2008.

[6] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372, 2016.

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

[8] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[9] L. Breiman. Random forests. *Machine learning*, 2001.

[10] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *IJCAI*, pages 1346–1352, 1997.

[11] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *AAAI*, 1998.

[12] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (4):425–455, 1986.

[13] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona. A linear approximation for graph-based simultaneous localization and mapping. *Robotics: Science and Systems VII*, pages 41–48, 2012.

[14] 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. 2017.

[15] M. Chandraker, J. Lim, and D. Kriegman. Moving in stereo: Efficient structure and motion using lines. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, pages 1741–1748. IEEE, 2009.

[16] K. Chatfield, V. S. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference (BMVC)*, 2011.

[17] J. Chen and P. Carr. Mimicking human camera operators. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 215–222. IEEE, 2015.

[18] L. Chen, W. Li, and D. Xu. Recognizing RGB images by learning from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1418–1425, 2014.

[19] A. Criminisi and J. Shotton. *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

[20] A. Criminisi, J. Shotton, D. Robertson, and E. Konukoglu. Regression forests for efficient anatomy detection and localization in ct studies. In *International MICCAI Workshop on Medical Computer Vision*. Springer, 2010.

[21] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. Technical Report MSR-TR-2011-114, Microsoft Research, 2011.

[22] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2–3):81–227, 2012.

[23] M. Cummins and P. Newman. Appearance-only SLAM at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9): 1100–1123, 2011.

[24] 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 Transactions on Graphics*, 2017.

[25] R. D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 2012.

[26] E. Dubrofsky and R. Woodham. Combining line and point correspondences for homography estimation. *Advances in Visual Computing*, pages 202–213, 2008.

[27] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision (ECCV)*, pages 834–849. Springer, 2014.

[28] 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.

[29] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349): 2–2, 1999.

[30] D. Gálvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. on Robotics*, 2012.

[31] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):930–943, 2003.

[32] A. P. Gee and W. W. Mayol-Cuevas. 6D relocalisation for RGBD cameras using synthetic view regression. In *British Machine Vision Conference (BMVC)*, 2012.

[33] 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.

[34] N. A. Goussies, S. Ubalde, and M. Mejail. Transfer learning decision forests for gesture recognition. *Journal of Machine Learning Research*, 2014.

[35] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[36] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. On straight line segment detection. *Journal of Mathematical Imaging and Vision*, 32(3):313–347, 2008.

[37] A. Gupta, J. J. Little, and R. J. Woodham. Using line and ellipse features for rectification of broadcast hockey video. In *Computer and Robot Vision (CRV), Canadian Conf. on*, 2011.

[38] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi. Multi-output learning for camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[39] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[40] F. W. C. Hazirbas, L. L.-T. T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. 2017.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] W. He, D. Goodkind, and P. Kowal. An aging world: 2015. *US Census Bureau*, pages 1–165, 2016.

[43] Y. Hel-Or and H. Hel-Or. Real-time pattern matching using projection kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(9):1430–1445, 2005.

[44] T. K. Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 20 (8):832–844, 1998.

[45] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[46] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta. Navigation in three-dimensional cluttered environments for mobile manipulation. In *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, 2012.

[47] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[48] C.-X. Ji and Z.-P. Zhang. Stereo match based on linear feature. In *Pattern Recognition, 1988., 9th International Conference on*, pages 875–878. IEEE, 1988.

[49] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[50] P. Kahn, L. Kitchen, and E. M. Riseman. A fast line finder for vision-guided robot navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(11):1098–1102, 1990.

[51] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, pages 4762–4769. IEEE, 2016.

[52] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. *arXiv preprint arXiv:1704.00390*, 2017.

[53] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, pages 2938–2946, 2015.

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

[55] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems (IROS), 2004 IEEE/RSJ International Conference on*, 2004.

[56] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk. Hector open source modules for autonomous mapping and navigation with rescue robots. In *Robot Soccer World Cup*. Springer, 2013.

[57] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2D mapping. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RASJ International Conference on*, 2010.

[58] M. Labbe and F. Michaud. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745, 2013.

[59] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 2015.

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

[61] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[62] R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu. Indoor relocalization in challenging environments with dual-stream convolutional neural networks. *IEEE Transactions on Automation Science and Engineering*, 2017.

[63] Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

[64] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[65] D. V. Lu, D. Hershberger, and W. D. Smart. Layered costmaps for context-sensitive navigation. In *Intelligent Robots and Systems (IROS), 2014 IEEE/RASJ International Conference on*, 2014.

[66] S. Mahadevan and D. P. Casasent. Detection of triple junction parameters in microscope images. In *Proc. SPIE*, volume 4387, pages 204–214, 2001.

[67] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office Marathon. In *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, 2010.

[68] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding (CVIU)*, 78(1):119–137, 2000.

[69] K. M. Mathieson, J. J. Kronenfeld, and V. M. Keith. Maintaining functional independence in elderly adults the roles of health status and financial resources in predicting home modifications and use of mobility equipment. *The Gerontologist*, 42(1):24–31, 2002.

[70] L. Meng, J. Chen, F. Tung, J. J. Little, and C. W. de Silva. Exploiting random RGB and sparse features for camera pose estimation. In *27th British Machine Vision Conference (BMVC)*, 2016.

[71] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[72] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics (T-RO)*, 31(5):1147–1163, 2015.

[73] 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.

[74] D. Nistér. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications*, 2005.

[75] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168. IEEE, 2006.

[76] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. on Knowledge and Data Engineering*, 2010.

[77] J. Patel, S. Shah, P. Thakkar, and K. Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1): 259–268, 2015.

[78] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *Robotics and Automation (ICRA) workshop on open source software, Proceedings of IEEE International Conference on*, 2009.

[79] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, 1993.

[80] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[81] H. Rehbinder and B. K. Ghosh. Pose estimation using line-based dynamic vision and inertial sensors. *IEEE Transactions on Automatic Control*, 48 (2):186–199, 2003.

[82] U. Reiser, T. Jacobs, G. Arbeiter, C. Parlitz, and K. Dautenhahn. Care-o-bot® 3–vision of a robot butler. In *Your virtual butler*. 2013.

[83] A. Rubio, M. Villamizar, L. Ferraz, A. Penate-Sanchez, A. Ramisa, E. Simo-Serra, A. Sanfeliu, and F. Moreno-Noguer. Efficient monocular pose estimation for complex 3d models. In *Robotics and Automation (ICRA), Proceedings of IEEE International Conference on*, pages 1397–1402. IEEE, 2015.

[84] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, pages 2564–2571. IEEE, 2011.

[85] Y. Salaün, R. Marlet, and P. Monasse. Robust and accurate line-and/or point-based pose estimation without manhattan assumptions. In *European Conference on Computer Vision (ECCV)*, pages 801–818. Springer, 2016.

[86] J. M. Santos, D. Portugal, and R. P. Rocha. An evaluation of 2D slam techniques available in robot operating system. In *SSRR*, 2013.

[87] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2016.

[88] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[89] S. Se, D. G. Lowe, and J. J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics (T-RO)*, 21(3): 364–375, 2005.

[90] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Learning to rank using privileged information. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 825–832, 2013.

[91] T. Sharp. Implementing decision trees and forests on a gpu. In *European conference on computer vision*, pages 595–608. Springer, 2008.

[92] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2008.

[93] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2937, 2013.

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

[95] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[96] J. Sivic, A. Zisserman, et al. Video google: A text retrieval approach to object matching in videos. In *Computer Vision (ICCV), 2003 IEEE International Conference on*, volume 2, pages 1470–1477, 2003.

[97] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige. Double window optimisation for constant time visual slam. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359. IEEE, 2011.

[98] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

[99] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford. On the performance of convnet features for place recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4297–4304. IEEE, 2015.

[100] N. Sunderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. *Proceedings of Robotics: Science and Systems XII*, 2015.

[101] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[102] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. H. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4400–4408, 2015.

[103] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin. Learning to navigate the energy landscape. In *International Conf. on 3D Vision (3DV)*, 2016.

[104] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5):544–557, 2009.

[105] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *ACM international conf. on Multimedia*, 2010.

[106] R. Vincent, B. Limketkai, and M. Eriksen. Comparison of indoor robot localization techniques in the absence of GPS. In *SPIE Defense, Security, and Sensing*, 2010.

[107] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(4):722–732, 2010.

[108] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization with spatial LSTMs. *arXiv preprint arXiv:1611.07890*, 2016.

[109] 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 *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, 2017.

[110] 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. *The International Journal of Robotics Research (IJRR)*, 34(4-5): 598–626, 2015.

[111] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: science and systems (RSS)*, volume 11, 2015.

[112] B. Williams, G. Klein, and I. Reid. Automatic relocalization and loop closing for real-time monocular slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(9):1699–1712, 2011.

[113] J. won Lee and C. Giraud-Carrier. Transfer learning in decision trees. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 726–731. IEEE, 2007.

[114] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). 2007.

[115] C. Wu et al. VisualSFM: A visual structure from motion system. 2011.

[116] J. Wu, L. Ma, and X. Hu. Predicting world coordinates of pixels in rgb images using convolutional neural network for camera relocalization. In *Intelligent Control and Information Processing (ICICIP), 2016 Seventh International Conference on*, pages 161–166. IEEE, 2016.

[117] P. Xu and F. Jelinek. Random forests in language modeling. In *EMNLP*, pages 325–332, 2004.

[118] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems (NIPS)*, pages 3320–3328, 2014.