Decision Forest For Efficient and Robust Camera Relocalization

Amine Kacete, Thomas Wentz, Jérôme Royan

[POSTER] Decision Forest For Efficient and Robust Camera Relocalization

Amine Kacete* Thomas Wentz † Jérôme Royan ‡

Institute of Research and Technlogy b-com

ABSTRACT

To robustly estimate the pose, classical methods assume some geometrical and temporal assumptions (SfM: Structure from Motion, SLAM: Simultaneous Localization and mapping). These approaches take a pair of images as input and establish correspondences based on global strategy (using the whole image information) or sparse strategy (using key-points features). These correspondences allow solving a set of linear equations related to the 3D information and camera pose in that environment. These past years, machine learning has been considered as an efficient way to tackle different problems in image processing and computer vision fields. To handle the task in hand, we propose to learn directly the mapping function between the acquired information from the camera and its pose using sparse decision forest. We achieved state-of the-art results on public and on our databases.

Index Terms: Camera relocalization, pose estimation, Random Forest, SLAM.

1 INTRODUCTION

Camera pose estimation is the process of determining the position and the orientation of the camera in a given scene using the acquired information (rgb-image, depth-image, rgb-d image, infrared-image,point cloud..). Knowing the pose information is crucial in various fields such Augmented Reality, indeed, to add additional synthetic objects on the environment in a realistic way, the camera pose is required to render optimally these objects.

Common approaches use global or sparse features matching between two images to estimate the camera continuous pose. With sparse strategy, a set of key-points are detected such as SIFT [17], SURF [1] or ORB [22]. Using the descriptors of these 2D key points, a matching can be established across key-frames descriptors corresponding to different poses to retrieve the optimal one as performed in [4].

In [15], the 2D-2D correspondences are robustly learned using randomized trees. [15] exploits directly the key-points to establish 2D-3D correspondences and to optimize the camera pose calculation. In [5, 18, 19, 27], key-points are used as a primordial component in a global relocalization system based on visual SLAM. One of the most important limitation of sparse camera pose estimation is related to the key-points detection and matching efficiency. Indeed, in unconstrained of textureless environments, matching key-points yields outliers which impacts considerably the accuracy and robustness of the pose estimation. To ensure sparsity without keypoint detection [6] samples directly pixel intensities to establish correspondences.

Unlike sparse strategy, global matching uses the whole image information to perform a geometrically dense mapping as in [7, 21, 26]. Some methods define the global matching as a learning task. [10] uses a global regression over a set of synthetic views based on Nadaraya-Watson estimator [20]. [13] traines a convolutional neural network to regress the camera pose from a single rgb image. These methods perform a robust camera pose estimation but present a considerable training and testing complexity.

The work from [23] and [3] presents the closest work to our approach. Indeed, to regress the camera pose, we train a decision forest on labeled training rgb-d patches. Unlike these approaches which use dense prediction from the image pixels, we extract a set patches based on a highly sparse key-points extraction. Our approach considerably reduces training complexity and enhances discrimination during trees learning. Figure 1 shows an example of an automatic camera pose estimation using our approach.

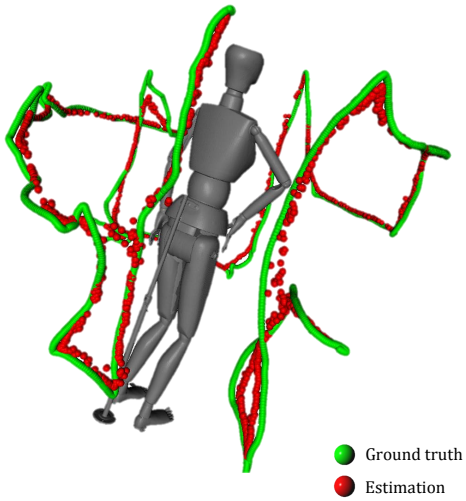


Figure 1: Automatic camera pose estimation performed by our method.

2 SPARSE CAMERA RELOCALIZATION LEARNING

In this section we explain in details our approach. In 2.1 and 2.2, we describe our camera pose estimation forest training and testing steps respectively.

2.1 Training

Random forest is a powerful technique to map highly dimensional input spaces into discrete or continuous output spaces. Introduced in [2], Random forest is a collection of randomly trained trees with high ability of generalization compared to the standard version. They have been successfully used for classification problems [16, 24] and regression [8, 9, 12]. Figure. 2 gives an illustrative example of a decision forest applied for an automatic camera pose estimation.

To train our decision forest $\mathcal{T} = \{\chi_k\}$ to automatically estimate camera pose from input rgb-d images, each tree χ_k is learned in a

*e-mail:amine.kacete@b-com.com
†e-mail:thomas.wentz@b-com.com
‡e-mail:jerome.royan@b-com.com

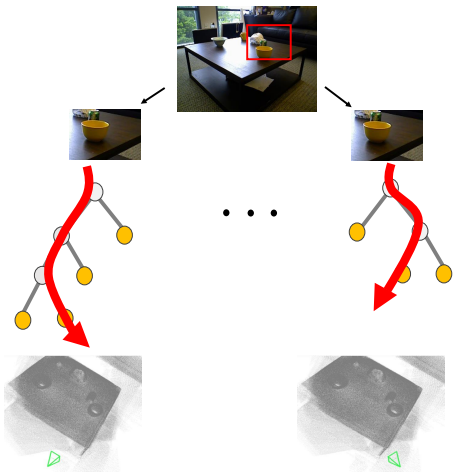


Figure 2: A visual example of a decision forest used in this work. Each tree processes a testing sample. By performing a simple learned binary test, each splitting node (white nodes) directs the sample until reaching the leaves (yellow nodes) which store a multivariate distribution of the camera pose (the green frustum represents the camera pose mean). The forest returns a combination of all the outputs.

supervised way on a set of annotated training samples.

Concretely, in every rgb-d sample, we extract a set of fixed size patches $\{\mathcal{P}_i = (\mathcal{I}_i^c, y_i)\}$ centred on the detected SURF key-points. $\mathcal{I}_i^c$ defines the appearance channel ($ie.$, $c = 0$ corresponds to the grayscale intensities and $c = 1$ to the depth). y_i represents the label which is the camera pose ground truth encapsulated as quaternion and translation as $(Q_i, T_i) = (q_w, q_x, q_y, q_z, t_x, t_y, t_z)$.

We provide, for each tree χ_k, 50% of the total training set. We train the trees in a greedy way, at each node, a simple binary test t is performed. According to the result of the test, a data sample is directed towards the left or the right child node. The tests are selected to achieve an optimal clustering. The terminal nodes of the tree called leaves, store the estimation models which approximate the best the desired output.

To achieve robustness, we formulate a differential binary test as performed in [24] as follows:

$$f_\theta(\mathcal{P}) = \mathcal{P}_i^c(x, y) - \mathcal{P}_i^c(x', y') \tag{1}$$

Where $\theta = ((x, y), (x', y'), c)$ represents the parameters encapsulating the two 2D pixels locations and the appearance channel in the processed patch $\mathcal{P}_i$ respectively. Supervising the training consists in finding at each non-leaf node the optimal separation which maximizes the purity of the data clustering. We performed the training following the standard Random forest pipeline:

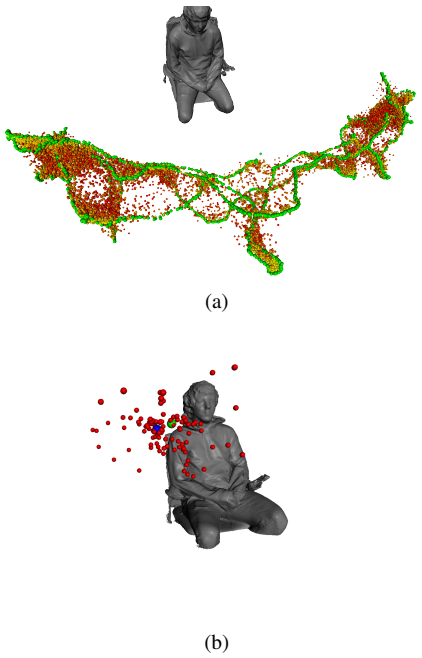
1. Generate a set of binary test parameters $\Theta = (\theta, \tau)$

2. For each Θ, split the parent set of patches $\mathcal{P}$ into two child subsets $\mathcal{P}_L$ and $\mathcal{P}_R$ as follows:

$$\mathcal{P}_L(\Theta) = \{\mathcal{P} | f_\theta \leq \tau\} \tag{2}$$

$$\mathcal{P}_R(\Theta) = \{\mathcal{P} \backslash \mathcal{P}_L(\Theta)\} \tag{3}$$

3. Evaluate the discrimination quality E for each splitting as follows:



(a)

(b)

Figure 3: (a) shows the estimation space $\mathcal{H}$ learned by the tree, which represents all the possible votes than can be cast by the forest (The more the vote is red, the less is confident). (b) illustrates, in red, the votes for a griven testing sample, in green the camera pose ground truth and in blue the final estimation calculated using mean-shift clustering.

$$E(\Theta) = \sum_{i \in \mathcal{P}_L} ||y_i - \bar{y}_L||_2^2 + \sum_{i \in \mathcal{P}_R} ||y_i - \bar{y}_R||_2^2 \tag{4}$$

$$\bar{y} = \frac{\sum_{i \in \mathcal{P}} y_i}{|\mathcal{P}|} \tag{5}$$

E defines the global euclidean distance between all the elements y_i to the centrod $\bar{y}$ in each cluster.

4. Select the optimal splitting binary test which minimizes E:

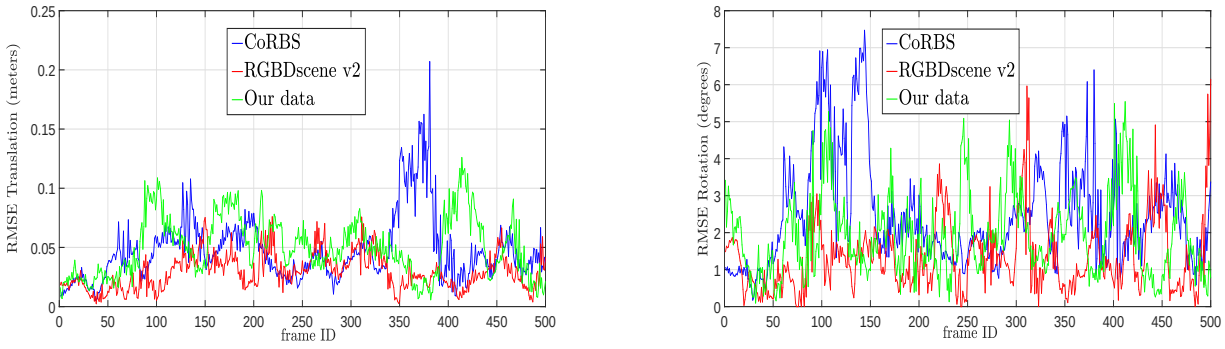
$$\Theta^* = \arg\min_{\Theta} E(\Theta). \tag{6}$$

The learning process finishes when the data reach a predefined maximum depth value of the tree or the number of patches let down a threshold value yielding the creation of the leaves. Each leaf l stores multivariate Gaussian distribution of all the reached poses as $p(y) = \mathcal{N}(y, \bar{y}_l, \Sigma_l^y)$.

2.2 Testing

To estimate camera pose from an unseen rgb-d sample, we extract, following the same strategy as in training, a set of patches. Each patch is passed through all the learned trees in the forest. Using the optimal stored binary test, each tree processes the patch until reaching a leaf. The estimation according to a single tree is given by the reached leaf l in terms of the stored distribution $p(y)$. The pose estimation for a given patch $\mathcal{P}_j$ over all the trees is calculated as follows:

$$p(y|\mathcal{P}_i^c) = \frac{1}{N_T} \sum_j p(y|l_j(\mathcal{P}_i^c)) \tag{7}$$



(a) (b)

Figure 4: (a) reports the translation errors of our approach on CoRBS, RGBDscene.v2 and our database respectively. (b) indicates the rotation errors on the same databases.

.

All the estimations corresponding to the extracted patches are regrouped as votes. Before performing the clustering of these votes, we discard the estimations from the leaves with high variance considered as non-informative. To locate the centroid of the cluster of the votes, we perform five mean-shift iterations using a Gaussian kernel. Figure. 3a illustrates the knowledge $\mathcal{H}$ of the learned forest. It represents all the estimations stored by the leaves. Visualizing $\mathcal{H}$ can help in predicting ahead some bad estimation. Figure. 3b shows the non-parametric clustering by mean-shift. For a given sample, the forest casts a set of votes (in red), we can notice the gap between the estimates which is directly linked to the decorrelation (bagging) introduced during learning. Performing mean-shift with some iterations allows to determine more accurate estimation than a simple statistical mean (the green sphere represents the ground truth and the blue one is the final estimation).

3 Experiments

In this section, we report the results of our estimations according to different experimentations. Firstly, in 3.1 we describe the data used in our experiments, then we report our decision forest accuracy in 3.2. In 3.3, we compare our method to state-of-the-art methods and evaluate the robustness of the approach on unfavorable scenarios in 3.4.

3.1 Datasets

To provide our forest with sufficient annotated training rgb-d samples and evaluate our camera pose accuracy, we used two public datasets RGBDscene.v2 [14] and CoRBS [25]. The first dataset uses the method from [11], which is based on an ICP rigid alignment, to accurately annotate the rgb-d samples. The acquisition is performed using Kinect.v1. For the second dataset, the acquisition is performed with Kinect.v2, the annotation is achieved thanks to a motion capture system. In total, 14 and 4 scenes are reconstructed for the first and second dataset respectively. We also build our own dataset by using Kinect.v2 for the acquisition. To annotate our rgb-d samples, we used the method from [21] which is a real-time 3D dense reconstruction technique following a dense SLAM pipeline as in [23]. We build a total of 17 RGB-D scenes. We trained a specific forest for each scene on half of the global samples number.

3.2 Forest accuracy

Some empirical analysis, based on computational cost and accuracy criteria, allowed us to fix some parameters to their optimal value according to our task. The number of the extracted patches by sample is fixed to 20 with a size of (30×30). The forest size is fixed to 15, each tree is learned with a depth of 15. In Figure. 4a and 4b, we report the mean quadratic errors related to translation and rotation respectively on the datasets CoRBS, RGBDscene.v2 and ours. We reported a mean error of 0.047m and 2.46° for the rotation and translation respectively on CoRBS. On RGBDscene.v2, we achieved best results with errors of 0.029m in terms of translation and 1.34° for the rotation. This difference in terms of accuracy is directly linked to the nature of the camera trajectory, indeed, in RGBDscene.v2, the camera presents small motions in both translation and rotations (translation according to x and z, rotation only around y) which makes the splitting process inside the trees nodes easier, while CoRBS gives more complicated camera trajectory giving complex camera manifold. On our RGB-D scene, the forest performed a mean error of 0.0501m and 1.95° for translation and rotation respectively. Figure. 6 gives more visual illustrations of our estimations on different scenes.

3.3 Comparison to baseline

Table. 1 reports a comparison of our approach to two methods for camera relocalization, PoseNet [13] and ORB-SLAM [18] respectively on RGBDscene.v2. [13] performed the most important errors, 0.062cm and 4.66° for translation and rotation respectively. This method uses a convolutional neural network (CNN) learned on the whole image appearance producing a highly smoothy camera pose output. This result can be explained by the provided training set (400 images) producing a suboptimal network weights optimization. In contrast, by extracting a set of patches, our method explicitly extends the training set size which considerably enhance the ability of discrimination of our forest. In addition, by geometrically targeting relevant regions, our method achieves more robustness under noisy scenarios. [18] achieves better results compared to the first approach, 0.052cm and 1.94° respectively. By learning a set of key-frames (on the same training set provided previously), this method optimizes the camera pose trajectory based on sparse key-points matching. Instead of establishing correspondences between successive frame features matching which produces increasing output uncertainty, our approach implicitly reduces considerably temporal uncertainty by processing each patch independently giving more accurate and robust camera relocalization estimation. Our work achieves 0.029cm and 1.34° for translation and rotation errors respectively.

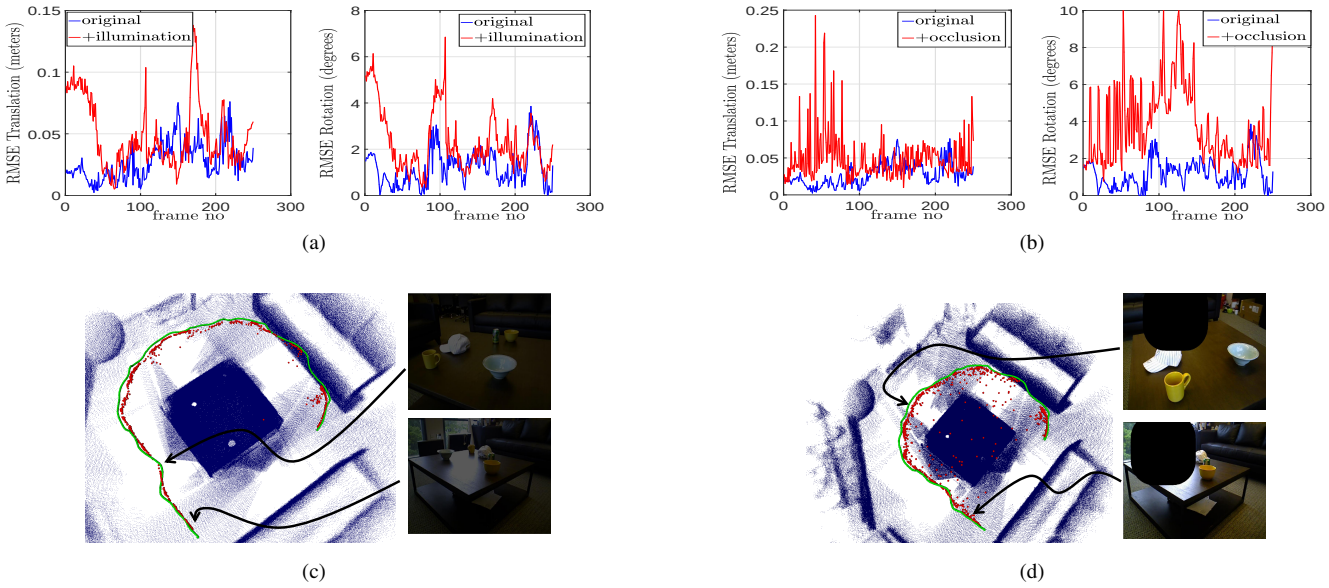


Figure 5: (a) reports translation and rotation errors of our method on original and unfavorably illuminated RGBDscene.v2. (c) gives a visual illustration of some under-illuminated frames and the correspondent estimation. (b) gives translation and rotation errors on original and occluded RGBDscene.v2. (d) gives an example of some occluded frames and the correspondent estimation.

Method	Translation error (cm)	Rotation error (°)
PoseNet [13]	0.062	4.66
ORB-SLAM [18]	0.052	1.94
Our method	**0.029**	**1.34**

Table 1: Comparison of our approach to the methods in [13] and [18].

3.4 Robustness to occlusions and illumination variation

One of the most challenging problem in camera relocalization consists in dealing with important image appearance variation. Unfortunately, this problem is frequently present in rgb-d scenarios. In this section, we evaluate the robustness of our approach with respect to illumination condition variation and strong occlusion in the rgb image. Concretely, we underexpose/overexpose the testing samples to synthetically produce unfavorable lightning conditions. To model occlusion, we randomly render a dark square with a fixed size of (120×120) in the rgb images. We report the results of these experiments in Figure. 5. Figure 5a illustrates the translation and rotation errors with and without illumination variation on RGBDscene.v2 database. The mean error goes from 0.0292m to 0.0570m and $1.34°$ to $3.38°$ for translation and rotation respectively. These results demonstrate the robustness and the high ability of generalization of our approach on such condition. Figure. 5c gives a visual illustration of the final estimation compared to the ground truth camera trajectory (red and green paths respectively) and typical lightning condition generated on testing samples. Figure. 5b describes the mean errors with respect to occlusions. We notice that errors go from 0.0292m to 0.0502m and $1.34°$ to $3.95°$ for translation and rotation respectively. Again, this result proves the efficiency of our approach under occlusion presence. Figure. 5d shows the results of our estimation and gives typical examples of the generated occlusions on the testing samples.

According to our forest parameters defined previously, our estimation runs in real-time on an Intel Core i7 @2.70GHZ with 8GB of RAM machine.

4 Conclusion

In this paper, we have proposed an approach based on regression decision forest to handle unconstrained camera pose estimation in rgb-d scenarios. Unlike similar approaches, we trained our forest on sparse data represented by a set of extracted patches. Based on SURF key-points detection, we demonstrated that targeting geometrically relevant regions in images increases forest accuracy and generalization. By establishing translation and rotation errors on challenging databases, we validate our approach by achieving state-of-the-art performance.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] T. Cavallari, S. Golodetz, N. A. Lord, J. P. C. Valentin, L. di Stefano, and P. H. S. Torr. On-the-fly adaptation of regression forests for online camera relocalisation. *CoRR*, abs/1702.02779, 2017.

[4] Z. Dong, G. Zhang, J. Jia, and H. Bao. Keyframe-based real-time camera tracking. In *ICCV*, 2009.

[5] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular slam. In *BMVC*, volume 13, page 136, 2008.

[6] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

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

[8] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 617–624. IEEE, 2011.

[9] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Decision forests for computer vision and medical image analysis*, pages 143–157. Springer, 2013.

[10] A. P. Gee and W. W. Mayol-Cuevas. 6d relocalisation for rgbd cameras using synthetic view regression. In *BMVC*, pages 1–11, 2012.

[11] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environ-

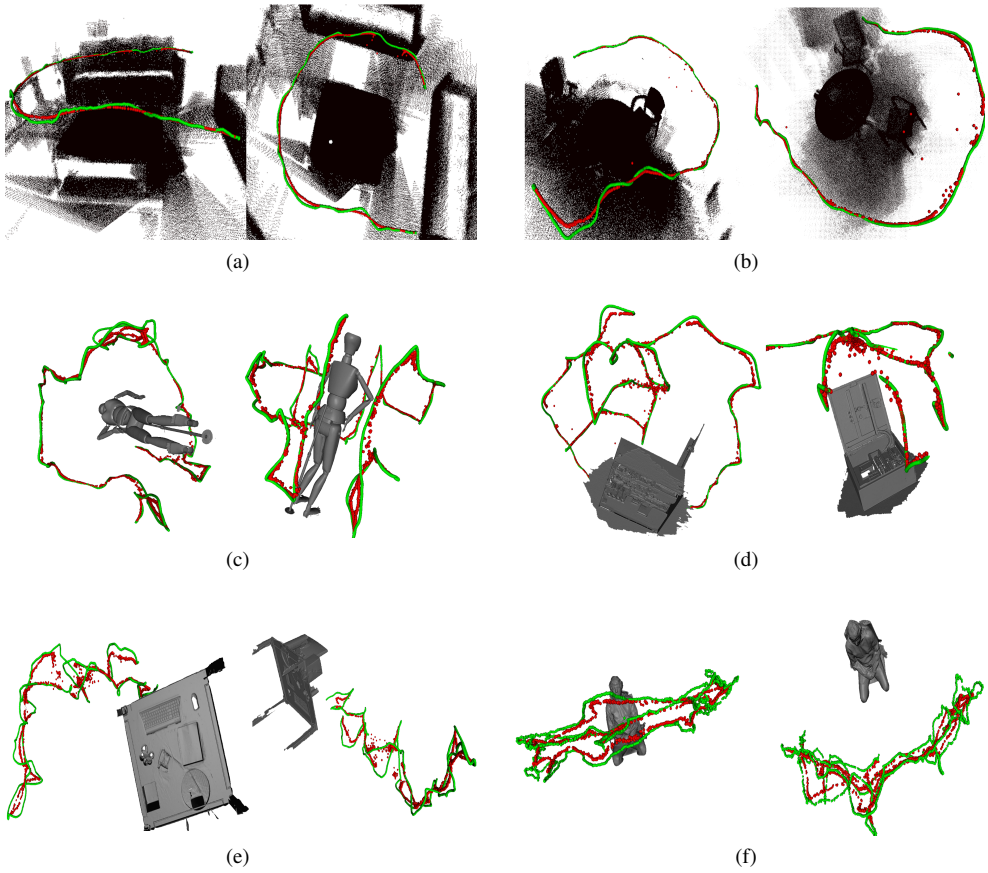


Figure 6: Camera pose estimation results of our decision forests on different databases, the green path represents the ground truth camera pose and the estimations in red. (a),(b) represents two different scenes in RGBDscene.v2 database.(c),(d) and (e) illustrates three different scenes in CoRBS database. The last scene represents one scene of our database described previously.

ments. In *the 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25. Citeseer, 2010.

[12] A. Kacete, R. Séguier, M. Collobert, and J. Royan. Unconstrained gaze estimation using random forest regression voting. In *Asian Conference on Computer Vision*, pages 419–432. Springer, 2016.

[13] 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*, pages 2938–2946, 2015.

[14] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3050–3057. IEEE, 2014.

[15] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE transactions on pattern analysis and machine intelligence*, 28(9):1465–1479, 2006.

[16] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 775–781. IEEE, 2005.

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

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

[19] R. Mur-Artal and J. D. Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[20] E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[21] 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), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[23] 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*, pages 2930–2937, 2013.

[24] 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.

[25] O. Wasenmüller, M. Meyer, and D. Stricker. Corbs: Comprehensive rgb-d benchmark for slam using kinect v2. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–7. IEEE, 2016.

[26] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[27] 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*, 33(9):1699–1712, 2011.