Exploiting Random RGB and Sparse Features for Camera Pose Estimation

Lili Meng[1]
lilimeng@mech.ubc.ca
Jianhui Chen[2]
jhchen14@cs.ubc.ca
Frederick Tung[2]
ftung@cs.ubc.ca
James J. Little[2]
little@cs.ubc.ca
Clarence W. de Silva[1]
desilva@mech.ubc.ca

[1] Department of Mechanical Engineering
University of British Columbia
Vancouver, Canada

[2] Department of Computer Science
University of British Columbia
Vancouver, Canada

Abstract

We address the problem of estimating camera pose relative to a known scene, given a single RGB image. We extend recent advances in scene coordinate regression forests for camera relocalization in RGB-D images to use RGB features, enabling camera relocalization from a single RGB image. Furthermore, we integrate random RGB features and sparse feature matching in an efficient and accurate way, broadening the method for fast sports camera calibration in highly dynamic scenes. We evaluate our method on both static, small scale and dynamic, large scale datasets with challenging camera poses. The proposed method is compared with several strong baselines. Experiment results demonstrate the efficacy of our approach, showing superior or on-par performance with the state of the art.

1 Introduction

Camera pose estimation plays a vital role in many computer vision, robotics, and augmented reality (AR) applications. Recent consumer robotics products, such as iRobot Roomba 980 and Dyson 360 eyes, have been equipped with the visual SLAM (simultaneous localization and mapping) technology and know where they have visited before. In AR products such as HoloLens, Project Tango and Oculus Rift, accurate camera poses are required to correctly overlay virtual objects onto the real world. Our work is mainly inspired by recent advances in scene coordinate regression forests (SCRF) based methods [19, 33, 36] for camera relocalization. Our method also benefits from the privilege learning [9, 32, 38] in which additional information only exists at the training stage. The SCRF-based methods use an efficient regression forest to guide the camera pose optimization using RGB-D images, achieving high accuracy. Our method 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.

Pages 59.1-59.12
DOI: https://dx.doi.org/10.5244/C.30.59

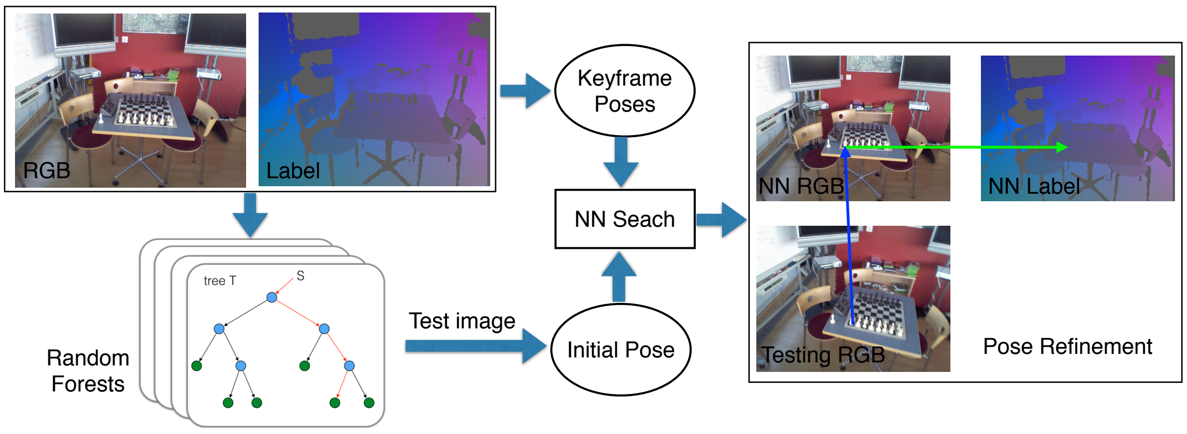


Figure 1: **Camera pose estimation pipeline.** During training, the scene information is encoded in a random forest. At test time, an initial camera pose is estimated using the predictions from the random forest with real-time response. Then, the initial camera pose is used to query a nearest neighbor (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 be any information (e.g. scene coordinate positions or pan-tilt-zoom configuration in PTZ camera) associated with pixel locations.

Fig.1 illustrates our pipeline. A regression forest is trained using RGB images and pixel-wise labels. At test time, 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 work are:

- Extend the SCRF based methods to use RGB features (without depth), and test with only a single RGB image.
- Integrate random features and sparse features, ensuring both efficiency and accuracy.
- Broaden random feature methods for fast sports camera pose estimation in large scale and highly dynamic scenarios.

2 Related Work

Camera pose estimation with respect to known scenes has been widely studied in large scale global localization [21, 29, 35], recovery from tracking failure [18, 22], and loop closure detection in visual SLAM [11, 28, 31]. Feature-based and keyframe-based approaches are two general categories for camera pose estimation, and other successful variants exist [20, 21, 34].

Feature-based methods usually match the descriptors extracted from the incoming frame and the descriptors in the database [29, 31]. These methods first use image patch descriptors (e.g., SIFT [25] and SURF [2]) to obtain 2D-3D correspondences. Then the perspective-three-point [16] method and RANSAC [13] are employed to determine camera pose. However, these methods require a large database of descriptors and efficient retrieval methods.

One important extension of traditional feature-based methods is SCRF based methods [19, 33, 36] which use random features in regression forests. In SCRF based methods, every pixel can directly provide a 2D-3D correspondence to the scene, replacing the traditional pipeline of feature detection, description and matching. However, these methods [19, 33, 36] need RGB-D images for both training and test, constraining their applications. Contemporaneous work [3] and [37] have extended the SCRF method to RGB camera relocalization.

Keyframe-based methods [12, 17, 22] 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 [18] and detect loop closure in SLAM [40]. However, these keyframe-based methods provide inaccurate matches when the query frame is significantly different from the stored keyframes.

The PoseNet [20, 21] based methods only use RGB images and achieve real-time response. However, they sacrifice accuracy compared with the SCRF method in indoor scenes as PoseNet models the camera pose estimation as a regression problem without post-processing. Rubio *et al.* [30] proposed an efficient hybrid method based on deep neural networks and 2D-3D feature matching in constrained viewpoints.

Our work aims to take advantage of the random features [33] and sparse features [27]. Compared with the sparse feature-based methods [15, 28, 31], our method is faster as we employ the random features by regression forests. Instead of searching in a high-dimensional image descriptor space, we search in the low dimension camera pose space. Moreover, our initial camera poses are accurate enough for some applications as will be shown in experiments in Sec. 4.

3 Method

In the initial camera pose estimation, we model the problem as a structural regression problem:

$$\hat{\mathbf{y}}_{\mathbf{p}} = f(\mathtt{I}, \mathbf{p}|\theta) \tag{1}$$

where $\mathtt{I}$ is an RGB image, $\mathbf{p}$ is a 2D pixel location and θ is model parameters. In the training, $\{\mathbf{p}, \mathbf{m}_{\mathbf{p}}\}$ are paired training data. The label $\mathbf{m}_{\mathbf{p}}$ can be any information associated with that pixel. For example, it is the world coordinate in camera relocalization, and the PTZ configuration in sports camera pose estimation. At test time, $\hat{\mathbf{y}}_{\mathbf{p}}$ is the prediction associated with pixel $\mathbf{p}$.

We choose randomized regression forests [10] to implement the regression function as it naturally handles structure regression, while being fast and remaining reasonably easy to train.

3.1 The random RGB features

We use features based on pairwise pixel comparison as in [23, 33]:

$$f_{\phi}(\mathbf{p}) = \mathtt{I}(\mathbf{p}, c_1) - \mathtt{I}(\mathbf{p} + \delta, c_2) \tag{2}$$

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 $\{\delta, c_1, c_2\}$. A significant difference with the depth-adaptive feature used by Shotton *et al.* [33] is that our feature does not require depth information. It makes our method more flexible and allows for the application to large scale scenes.

3.2 Training a random regression forest

A regression forest is an ensemble of T independently trained decision trees. Each decision tree is a tree-structured regressor consisting of decision (or split) nodes and prediction (or leaf) nodes. We grow the regression forest using greedy forest training [10].

Weak learner model Each decision node i represents a 'weak learner' parameterized by $\theta_i = \{\phi_i, \tau_i\}$. We pick the parameter θ_i from a set of randomly sampled candidates Θ_i. Samples are evaluated on weak learners and passed from the root node to the leaf nodes as follows:

$$h(\mathbf{p};\theta_i) = \begin{cases} 0, & \text{if } f_{\phi_i}(\mathbf{p}) \geq \tau_i, \quad \text{go to the left subset } \mathcal{S}_i^L. \\ 1, & \text{if } f_{\phi_i}(\mathbf{p}) < \tau_i, \quad \text{go to the right subset } \mathcal{S}_i^R. \end{cases} \tag{3}$$

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

The training is to optimize θ_i for each decision node:

$$\theta_i^* = \arg\max_{\theta_i \in \Theta_i} I_i(\mathcal{S}_i, \theta_i) \tag{4}$$

$$I_i = E(\mathcal{S}_i) - \sum_{j \in \{L,R\}} \frac{|\mathcal{S}_i^j(\theta_i)|}{|\mathcal{S}_i|} E(\mathcal{S}_i^j(\theta_i)) \tag{5}$$

where $E(\mathcal{S}_i)$ is the entropy of the labels in $\mathcal{S}_i$, and subset $\mathcal{S}_i^j$ is conditioned on the split parameter θ_i.

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, the leaf outputs an estimated vector:

$$\mathbf{v}_t^* = \arg\max_{\mathbf{v}} p(\mathbf{v}|\theta_f) \tag{6}$$

with

$$p(\mathbf{v}|\theta_f) = N(\mathbf{v};\mu,\Sigma) \tag{7}$$

where μ, Σ are the mean and covariance of a multi-variate Gaussian distribution. The more sophisticated Gaussian Mixture Model can better describe the observed multi-modal distribution in the leaf node [36]. To achieve higher efficiency, we keep the leaf node as simple as possible. Besides the pixel location in 3D world coordinate, our 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. Because our method stores RGB color distribution in leaf nodes, we use the prediction that has the most similar color distribution with the test pixel as the final prediction.

3.3 Pose refinement

In order to extend the random feature prediction from small scale to large scale, we drop the depth information as it is not available for large scale from current consumer depth cameras (e.g. Kinect maximum depth range 6m [14]). 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, we found that the pyramid method is too time-consuming and does not significantly improve the accuracy.

Based on this observation, we design a hybrid pipeline. First, we use the faster random features to get an initial camera pose $\mathtt{P}_0$. Then, we employ accurate SIFT features [25] to refine the initial camera pose. In the second step, the sparse feature matching based method requires an image that shares the similar camera parameters with the test image. We search for the nearest neighbor (NN) image whose camera is closest to the location of $\mathtt{P}_0$ while

Algorithm 1 Camera pose estimation from a single RGB image

```
Input: A set $\mathcal{S}_{\mathtt{I}} = \{\mathtt{I}_1, \mathtt{I}_2, \cdots \mathtt{I}_n\}$ of images with pixel-wise labels
Input: An RGB image $\mathtt{I}$
Output: The camera pose $\mathtt{P}$ of image $\mathtt{I}$
1: $\mathcal{S}_i$ = a set of randomly sampled pixel-wise training samples;
2: train a regression forest using $\mathcal{S}_i$; // Sec. 3.1 3.2
3: $\mathtt{P}_0$ = initial camera pose from the regression forest predictions;
4: search nearest neighbor image $\mathtt{I}_{nn}$ in $\mathcal{S}_{\mathtt{I}}$ 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. 3.3
7: return $\mathtt{P}$
```

having an orientation difference no greater than a predefined threshold τ_0 (1/4 of the field of view).

The dimension of the camera pose space (≤ 6) is much smaller than the image descriptor space (up to several hundreds [7]). Therefore, our method is much more efficient than the place recognition methods using bag of words [15, 28]. Once the NN image is found, we refine the camera pose by minimizing the reprojection error:

$$\mathtt{P}^* = \arg\min_{\mathtt{P}} \sum_k d(\mathbf{x}_k, \mathtt{P}\mathbf{X}_k)^2 \tag{8}$$

where $\mathtt{P}$ is the camera pose, $\mathbf{x}$ are the feature locations in the image, $\mathbf{X}$ are the correspondent 3D world coordinates in the NN image. Because correspondences contain outliers, we optimize Eq. 8 using EPnP [24] and RANSAC[13].

Algorithm 1 briefly shows our method step by step. Sports camera calibration shares the same process but the formulation is slightly different from the general camera relocalization. We put the detailed method of sports camera calibration in Sec 4.2 where more context information is available.

4 Evaluations

4.1 Camera relocalization from a single RGB image

Dataset We use the 7 *Scenes* dataset of [33] to evaluate our method. The dataset consists of 7 scenes which were recorded with a Kinect RGB-D camera at 640×480 resolution. Each scene includes several camera sequences that contain RGB-D frames together with corresponding ground-truth camera poses. The dataset exhibits shape/color ambiguities, specularities and motion blur, which expose great challenges for our RGB-only random features.

Baselines and error metric We use three strong methods as our baselines: SCRF [33], PoseNet [21] and Bayesian PoseNet [20]. 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 samples. We use the same median translational error and rotational error as in [20, 21] for fair comparison.

Scene	Space	SCRF	PoseNet	Bayesian	Ours
Training		RGB-D	RGB	RGB	RGB-D
Test		RGB-D	RGB	RGB	RGB
Chess	3x2x1m	0.03m, 0.66°	0.32m, 8.12°	0.37m, 7.24°	0.12m, 3.92°
Fire	2.5x1x1m	0.05m, 1.50°	0.47m, 14.4°	0.43m, 13.7°	0.14m, 4.64°
Heads	2x0.5x1m	0.06m, 5.50°	0.29m, 12.0°	0.31m, 12.0°	0.10m, 6.82°
Office	2.5x2x1.5m	0.04m, 0.78°	0.48m, 7.68°	0.48m, 8.04°	0.15m, 4.23°
Pumpkin	2.5x2x1m	0.04m, 0.68°	0.47m, 8.42°	0.61m, 7.08°	0.22m, 5.40°
Red Kitchen	4x3x1.5m	0.04m, 0.76°	0.59m, 8.64°	0.58m, 7.54°	0.14m, 3.71°
Stairs	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°

Table 1: **Relocalization results for the** 7 *Scenes* **dataset**. We show median performance for our method on all scenes against three state-of-the-art methods: SCRF [33], PoseNet [21] and Bayesian PoseNet [20].

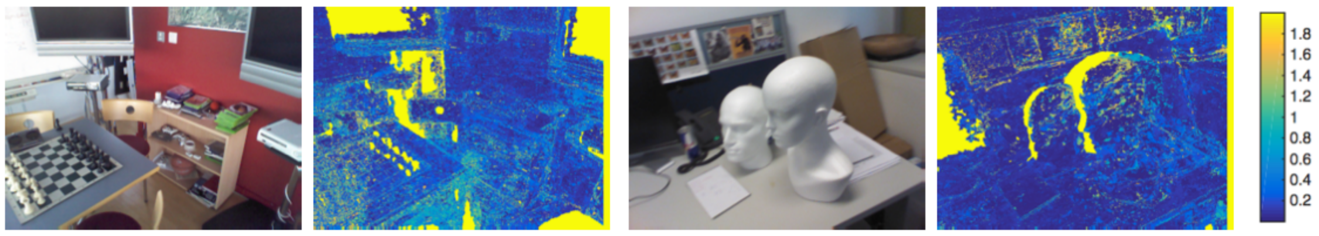


Figure 2: **Pixel-wise prediction error distribution from the regression forests.** Heat maps show the prediction error distribution directly from the regression forests on the *Chess* and *Heads* scenes. Large errors occur on black screens and other texture-less regions.

Results and analysis We present our main results in Table 1. Our method considerably outperforms the PoseNet and Bayesian PoseNet on all scenes. We note, however, that the PoseNet based methods need only RGB images for both training and test. Our method is not as accurate as the SCRF method. However, our approach does not require the depth image at test time which greatly lightens the requirements of end users.

The best performance of our method is on the *Heads* Scene. The worst performance is 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, we visualize the pixel-wise prediction error using heat maps as shown in Fig. 2. 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 localization result $0.21m, 6.09°$ for *Chess* and $0.25m, 8.53°$ on *Fire*, for instance.

4.2 Sports camera calibration

In sports such as basketball and soccer, the main broadcasting cameras are pan tilt zoom (PTZ) cameras with fixed locations. Different from the conventional camera relocalization in which the focal length is fixed, a PTZ camera has large variation in focal length. Another issue is highly dynamic movements of players, which cause more noise in the training and test data. All these factors make sports camera calibration very challenging under near real-time response constraints.

Labels The camera pose is decomposed to pan, tilt and focal length (θ, ϕ, f) [8]. For an image location $\mathbf{p}$, the label is $\mathbf{m} = (\theta_\mathbf{p}, \phi_\mathbf{p}, f_\mathbf{p})$:

$$\begin{cases} \theta_\mathbf{p} = \theta + \arctan \dfrac{x-u}{f} \\ \phi_\mathbf{p} = \phi + \arctan \dfrac{y-v}{f} \\ f_\mathbf{p} = \sqrt{f^2 + (x-u)^2 + (y-v)^2} \end{cases} \tag{9}$$

where (u, v) is the principal point and $(\theta_\mathbf{p}, \phi_\mathbf{p}, f_\mathbf{p})$ approximate the pan, tilt angles and focal length of the location $\mathbf{p}$ [42].

Objective function We use the “error or fit” energy function [5]:

$$E(S) = \frac{1}{|S|} \sum \|\mathbf{w}(\mathbf{m} - \bar{\mathbf{m}})\|_2^2 \tag{10}$$

where the weight $\mathbf{w}$ jointly optimizes the angles and the focal lengths. The typical focal length is around $2,000$ pixels and camera angle range is around 80^o. We experimentally set $\mathbf{w} = [1, 1, 1e^{-3}]^T$ to mostly eliminate angular errors.

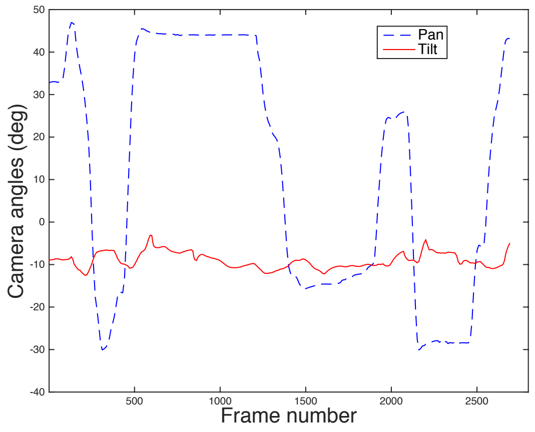
At test time, each sampled pixel gives a closed form solution for (θ, ϕ, f) using Eq. 9. As a result, our method can simply vote for the best result without using RANSAC.

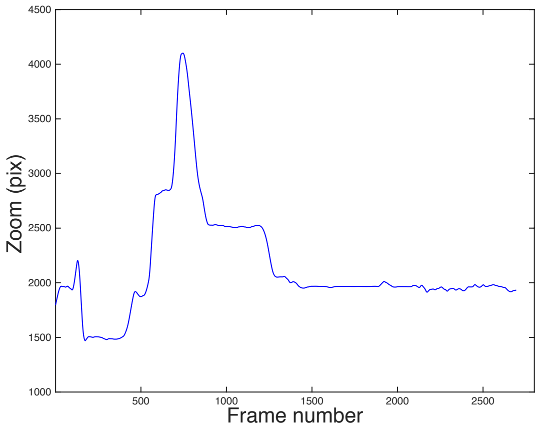
Dataset The basketball dataset records an indoor high-school basketball match using a PTZ camera at 1280×720 resolution and 30 FPS. ‘Timeouts’ were manually removed, resulting in ‘in-play’ data divided into four sequences. The ground truth labels were annotated manually and refined by the point-less calibration method [6]. These sequences were uniformly sampled at 3 FPS to remove frame content similarities. The downsampling is a practical strategy to speed up calibration by linearly interpolating the remaining frames during the test.

The dataset is much larger than the 7 *Scenes* dataset in the space scale with a spatial extent of $20 \times 30 \times 6m$. Fig. 3 shows the camera trajectory of one sequence and three example images. The video sequences exhibit various difficulties, especially significant player dynamics and severe motion blur. As currently there are few sports camera calibration datasets, we will make this dataset publicly available.

Error metric We report the percentage of test frames for which the estimated camera pose is essentially ‘correct’. We define the correctness as: the camera parameters are within 2° pan-tilt angular error and 10% of focal length error compared with the ground truth. To better make use of the relative small amount of data and reflect the varying challenges in different sequences for different methods, we employ leave-one-out cross validation (e.g. train with sequences 1, 2, 3 and test on sequence 4), and report the corresponding results.

Main results Table 2 shows the PTZ camera estimation results with two strong baselines. The ‘sparse’ baseline uses accurate sparse feature matching techniques [26], while the PoseNet [21] baseline is originally designed to estimate 6D camera pose in large-scale scenes. Our method achieved best performance in accuracy and nearly real-time response.





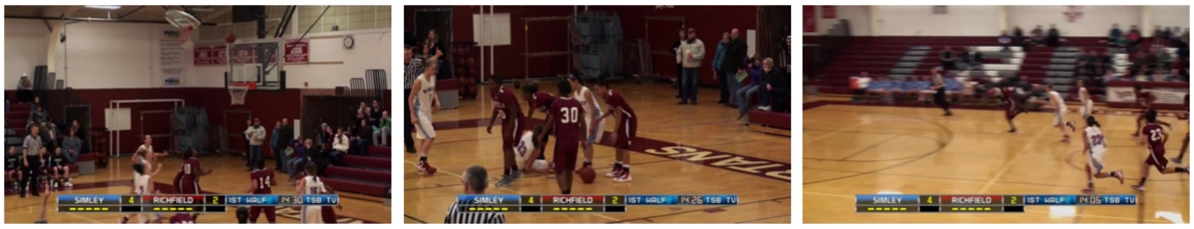


Figure 3: **Highly dynamic basketball dataset.** Top row: pan, tilt and zoom parameters of an example sequence. The camera trajectories exhibit varying changes. Bottom row: examples of images which have typical challenges for camera pose estimation such as motion blur, dynamic objects and narrow field of view.

Quick response is crucial for online sports broadcasting which only tolerates several seconds delay. At test time, our method achieves 5 FPS response running on a single CPU core with 2.3 GHz, and can achieve real-time response with GPU acceleration. At 5 FPS, our method is practically useful as the system can skip some frames and linearly interpolate the result from neighboring frames. The experiment results demonstrate that our method performs significantly better than the deep neural network based method in accuracy. Compared with the offline sparse feature based method, our method achieves superior or on-par performance in accuracy while significantly improving efficiency. Considering the practical camera calibration application which needs both fast speed and accurate performance, our method provides a better choice.

Test Sequence	Baselines			Our Results	
(# Frames)	Sparse RGB	PoseNet	PoseNet+Sparse	RF + voting	RF+Sparse
Hardware	CPU	GPU	GPU+CPU	CPU	CPU
Seq 1 (360)	**88.9%**	25.6%	79.2%	59.7%	86.1%
Seq 2 (270)	93.0%	50.4%	92.9%	67.8%	**94.4%**
Seq 3 (180)	95.6%	52.8%	98.3%	**98.9%**	97.8%
Seq 4 (180)	88.3%	49.5%	97.8%	98.9%	**100%**
Timings/frame	26s	**0.005s**	—-	0.08s	0.2s

Table 2: **PTZ camera calibration results.** We compare our method with the sparse feature based method [26], PoseNet [21], and PoseNet+Sparse. The percentages are of 'correct' test frames (within 2° in pan and tilt angular error and 10% of focal length error). The best performance is highlighted. Our method achieves the best performance considering accuracy and speed at the same time for sports camera calibration.

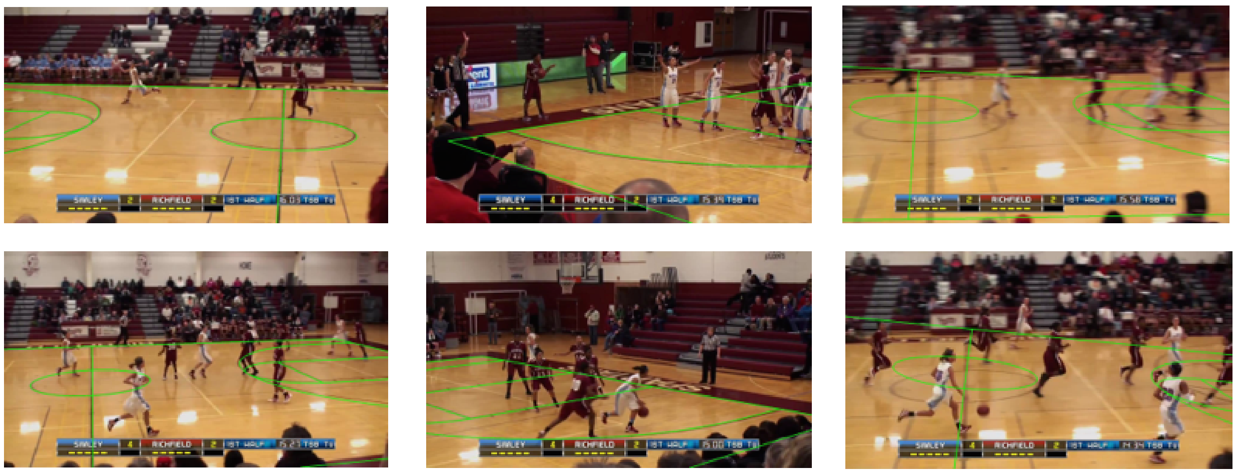


Figure 4: **Qualitative result.** (a) Moderate motion blur; (b) Narrow field of view; (c) Severe motion blur (best viewed in color). The standard basketball court lines are overlaid on the original images to indicate the accuracy of the calibration. Our method achieves satisfactory results on (a) and (b). It does not perform very well with severely motion-blurred images (c).

Analysis and discussion An interesting finding is that random forests with simple voting achieves unexpected high accuracy (98.9%) in sequence 3 and 4. We further explored the two sequences by manually examining the video sequences. We found that the two sequences have typical challenges such as motion blur. Both of them have relatively small variations in focal length compared with the other two sequences. So we believe the variations in focal length present the major difficulty as we do not explicitly model the scale-invariant property in the random forest. The SIFT feature in the camera refinement step significantly improves the accuracy (about 27%) in sequence 1 and 2. This phenomenon demonstrates the complementary property of the random features and sparse features. We also added our pose refinement step to the PoseNet. By doing so obtains similar result with our method but needs powerful GPU.

We present qualitative results in Fig. 4, showing both success and failures of our method. The standard basketball court lines are overlaid on the original images to indicate the accuracy of the calibration. We found that our method can accurately estimate camera pose from moderate motion blur (Fig. 4 (a)) and narrow field of view (Fig. 4 (b)). However, it does not perform very well when there is severe motion blur in the images (Fig. 4 (c)).

4.3 Implementation details

Our proposed approach is implemented with C++ using OpenCV [4] and VXL [1] on an Intel 2.3 GHz, 8GB memory Mac System. For the random forest, the parameter settings are: tree number $T = 10$; 500 and 200 training images per tree for 7 Scenes and basketball dataset, respectively; 5,000 randomly sampled example pixels per training image. We use a modified 32-dimension SIFT feature from the VLFeat library [39] 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 [41]. For the camera pose optimization, we use off-the-shelf *solvePnPRansac* in OpenCV.

We run the PoseNet baseline using the code by its original author on a Linux machine with an Nvidia GeForce GTX670 GPU. We use the pre-trained weights to initialize the network weights.

5 Conclusion and future work

In this paper, we proposed a hybrid method using RGB random features and sparse features for camera pose estimation. The method uses a single RGB-only image in the test and achieves near real-time response. The comparisons on the challenging 7 *Scenes* dataset with three strong baselines demonstrate the efficacy of our method, showing comparable results with state-of-the-art methods. The experimental results on the new basketball dataset show that our method achieves superior or on-par performance for PTZ sports camera calibration, extending the regression forest based methods to large-scale dynamic scenes.

In the future, we would like to improve prediction accuracy using deep features. We also plan to investigate the possibility of integrating accurate sparse features with the random features in the training phase so that we can further improve the prediction accuracy from random regression forests.

Acknowledgment The authors would like to thank Alex Kendall for PoseNet training suggestions. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, Canada Foundation for Innovation (CFI), British Columbia Knowledge Development Fund.

References

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

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *CVIU*, 110(3):346–359, 2008.

[3] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016.

[4] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

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

[6] Peter Carr, Yaser Sheikh, and Iain Matthews. Point-less calibration: Camera parameters from gradient-based alignment to edge images. In *WACV*, 2012.

[7] Ken Chatfield, Victor S Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[8] Jianhui Chen and Peter Carr. Mimicking human camera operators. In *WACV*, 2015.

[9] Lin Chen, Wen Li, and Dong Xu. Recognizing RGB images by learning from RGB-D data. In *CVPR*, 2014.

[10] Antonio Criminisi, Jamie Shotton, and Ender 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.

[11] Mark Cummins and Paul Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *IJRR*, 30(9):1100–1123, 2011.

[12] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*. 2014.

[13] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[14] Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren, and Kurt Konolige. *Consumer depth cameras for computer vision: research topics and applications*. Springer Science & Business Media, 2012.

[15] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *Robotics, IEEE Trans. on*, 28(5):1188–1197.

[16] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *PAMI, IEEE Trans. on*, 25(8):930–943, 2003.

[17] Andrew P Gee and Walterio W Mayol-Cuevas. 6D relocalisation for RGBD cameras using synthetic view regression. In *BMVC*, 2012.

[18] Ben Glocker, Jamie Shotton, Antonio Criminisi, and Shahram Izadi. Real-time RGB-D camera relocalization via randomized ferns for keyframe encoding. *Visualization and Computer Graphics, IEEE Trans. on*, 21(5):571–583, 2015.

[19] Abner Guzman-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew Fitzgibbon, and Shahram Izadi. Multi-output learning for camera relocalization. In *CVPR*, 2014.

[20] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. *ICRA*, 2016.

[21] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *ICCV*, 2015.

[22] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality. IEEE and ACM International Symp. on*, pages 225–234, 2007.

[23] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *PAMI, IEEE Trans. on*, 28(9):1465–1479.

[24] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate o (n) solution to the PnP problem. *IJCV*, 81(2):155–166, 2009.

[25] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60 (2):91–110, 2004.

[26] Wei-Lwun Lu, J-A Ting, James J Little, and Kevin P Murphy. Learning to track and identify players from broadcast sports videos. *PAMI, IEEE Trans. on*, 35(7):1704–1716, 2013.

[27] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *PAMI, IEEE Trans. on*, 27(10):1615–1630, 2005.

[28] Raul Mur-Artal, JMM Montiel, and Juan D Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[29] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[30] A Rubio, Michael Villamizar, Luis Ferraz, Adrián Penate-Sanchez, Arnau Ramisa, Edgar Simo-Serra, Alberto Sanfeliu, and Francesc Moreno-Noguer. Efficient monocular pose estimation for complex 3d models. In *ICRA*, 2015.

[31] Stephen Se, David G Lowe, and James J Little. Vision-based global localization and mapping for mobile robots. *Robotics, IEEE Trans. on*, 21(3):364–375, 2005.

[32] Viktoriia Sharmanska, Novi Quadrianto, and Christoph Lampert. Learning to rank using privileged information. In *ICCV*, 2013.

[33] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, 2013.

[34] Niko Sunderhauf, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft, and Michael Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. *Proceedings of Robotics: Science and Systems XII*, 2015.

[35] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *CVPR*, 2015.

[36] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015.

[37] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. *arXiv preprint arXiv:1603.05772*, 2016.

[38] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5):544–557, 2009.

[39] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *ACM international conf. on Multimedia*, pages 1469–1472, 2010.

[40] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. *Proc. Robotics: Science and Systems, Rome, Italy*, 2015.

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

[42] Ziyan Wu and Richard J Radke. Keeping a pan-tilt-zoom camera calibrated. *PAMI, IEEE Trans. on*, 35(8):1994–2007, 2013.