This CVPR Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

RGB-D Indoor Mapping Using Deep Features

Oguzhan Guclu
Ahi Evran University
Kirsehir, Turkey
guclu.oguzhan@outlook.com

Ali Caglayan
Bingol University
Bingol, Turkey
caglayan.ali@outlook.com

Ahmet Burak Can
Hacettepe University
Ankara, Turkey
abc@cs.hacettepe.edu.tr

Abstract

RGB-D indoor mapping has been an active research topic in the last decade with the advance of depth sensors. However, despite the great success of deep learning techniques on various problems, similar approaches for SLAM have not been much addressed yet. In this work, an RGB-D SLAM system using a deep learning approach for mapping indoor environments is proposed. A pre-trained CNN model with multiple random recursive structures is utilized to acquire deep features in an efficient way with no need for training. Deep features present strong representations from color frames and enable better data association. To increase computational efficiency, deep feature vectors are considered as points in a high dimensional space and indexed in a priority search k-means tree. The search precision is improved by employing an adaptive mechanism. For motion estimation, a sparse feature based approach is adopted by employing a robust keypoint detector and descriptor combination. The system is assessed on TUM RGB-D benchmark using the sequences recorded in medium and large sized environments. The experimental results demonstrate the accuracy and robustness of the proposed system over the state-of-the-art, especially in large sequences.

1. Introduction

Simultaneous Localization and Mapping (SLAM) plays a key role in many robotic applications such as autonomous navigation, obstacle avoidance, and manipulation. In the last decade, visual SLAM research has received a great attention with the availability of commodity RGB-D cameras. It seems that SLAM research will continue to preserve its importance in the near future, especially with the widespread use of autonomous robots and vehicles in daily life.

In the recent years, deep learning techniques has led to important advances for various machine vision tasks including object recognition [21, 17, 45], object detection [24, 32, 23], and semantic segmentation [25, 9, 10]. However, despite different components of SLAM have been handled by supervised learning approaches such as keypoint extraction [41, 34], camera localization [19, 5], and place recognition [11, 26], it has made a slower progress for more advanced tasks, such as geometric tasks in SLAM. Considering the advancement in the last years, deep learning based approaches do not have the same level of maturity in RGB-D data based SLAM systems yet.

Figure 1. Point cloud reconstruction of the industrial hall (*fr2* environment) in TUM RGB-D Benchmark [37]

In this paper, we present an RGB-D SLAM system using deep features to map indoor environments. In the proposed system, deep features of color frames are employed to obtain frame associations. Specifically, we employ the recent work in [7] for data association as timing is critical for the handled task. We adapt this work, which originally integrates a pre-trained CNN model with recursive neural networks for RGB-D object recognition, to our problem. Deep features are extracted from the pre-trained CNN model without any training or fine-tuning. In order to take advantage of the extracted CNN features optimally, multiple random recursive neural networks (RNN) [35] are applied to encode these features into higher level representations without a back-propagation algorithm. In this way, we efficiently produce optimum feature representations without training. Then, the deep features are indexed in a priority search k-means tree [28] to handle high dimensional data and perform search effectively. A dynamic thresholding approach is used for outlier rejection. The system performs frame-to-frame motion estimation by exploiting keypoint correspondences between frames. To this end, a robust keypoint detector and a binary descriptor are used. Figure 1

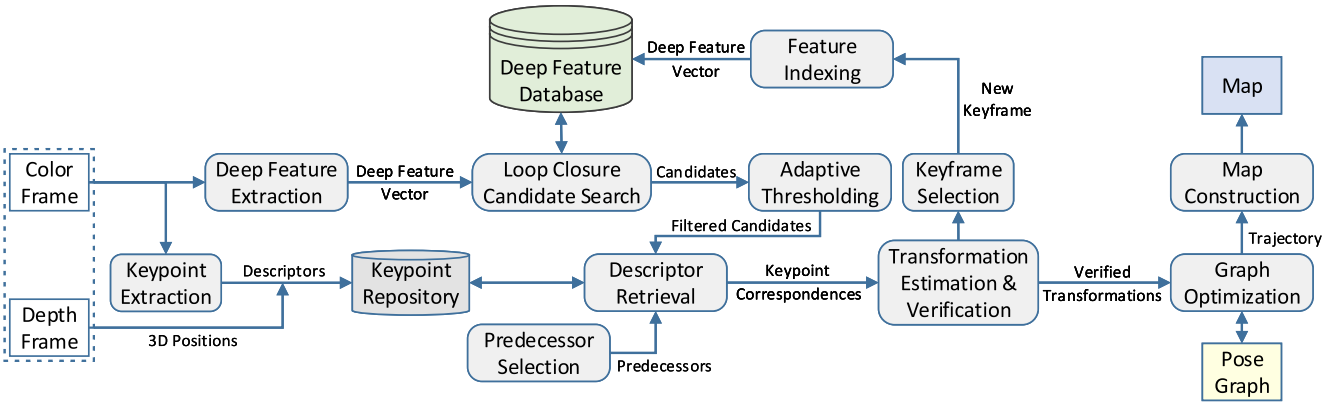

Figure 2. General overview of the proposed SLAM system.

shows a point cloud reconstruction of a test environment obtained with our system.

To the best of our knowledge, we are the first to exploit deep features with such an indexing structure in an RGB-D based SLAM system. The main contributions of this work are summarized as follows:

1. We develop an RGB-D SLAM system which maps large indoor environments consistently on CPU by utilizing deep features and extends the state-of-the-art performance for challenging large sequences.

2. Our approach for deep feature extraction has no training or fine-tuning and presents highly distinctive information together with significant computation savings with small feature set sizes. Consequently, this yields to a robust SLAM system in terms of both mapping accuracy and processing time in accordance with the objectives of the handled task.

3. We present experimental evaluation on a large set of sequences of TUM RGB-D benchmark [37] and compare the proposed SLAM system with the state-of-the-art systems in challenging scenes. We demonstrate that our proposed system outperforms conventional alternatives especially in large-scale mapping.

2. Related Work

The major approaches in RGB-D based SLAM systems can be divided into two groups: feature based and direct.

In feature based approaches, motion estimation relies on extracting and matching distinctive keypoints between frames. One of the state-of-the-art feature based SLAM systems is ORB-SLAM2 [29], which performs all operations using ORB keypoints extracted from color frames. Motion estimation is performed through minimizing reprojection error between keypoint correspondences by bundle adjustment. Loop closure detection is performed using the bag-of-words approach. Endres *et al.* [13] presents RGB-D SLAM, which uses SIFT, SURF, and ORB keypoints. RANSAC [14] is applied on keypoint matches to estimate motion. Computed transformations are validated using an environment measurement model. Loop closure candidates are chosen from keyframes randomly. In [16], a loop closure detection approach based on visual place recognition is proposed to extend the performance of the RGB-D SLAM system [13]. Image histogram similarity is used to obtain an initial group of loop closure candidates and a filtering approach called adaptive thresholding is applied to discard unrelated candidates. The extended system works more effectively than RGB-D SLAM [13] in larger environments.

The direct approaches utilize all frame data without applying a feature extraction step. An impressive direct approach is KinectFusion of Newcombe *et al.* [30], which represents the scene using a truncated signed distance function (TSDF). The volumetric model is progressively updated by integrating depth frames through applying ICP alignment [4] with a predicted surface. The system is limited to small environments and works on GPUs. Kintinuous [39] extends KinectFusion by modifying the TSDF structure in order to move dynamically. As the camera moves, the TSDF is shifted virtually along the trajectory to include a new area. A point cloud that represents the previous region leaving the TSDF is added to a triangular mesh. A bag-of-words descriptor database containing SURF features is used for loop closure detection. DVO SLAM [20] minimizes both photometric and geometric errors for motion estimation. Loop closure detection is performed by searching the candidates metrically in a sphere of keyframes. An entropy based similarity metric is used to validate the estimated transformations. MRSMap [36] estimates motion by registering multi-resolution surfel maps. The surfels are associated and the transformation that maximizes the matching likelihood is estimated. Loop closures are searched randomly and matching likelihood between key views are used to determine similarity. Whelan *et al.* proposes ElasticFusion [40], which

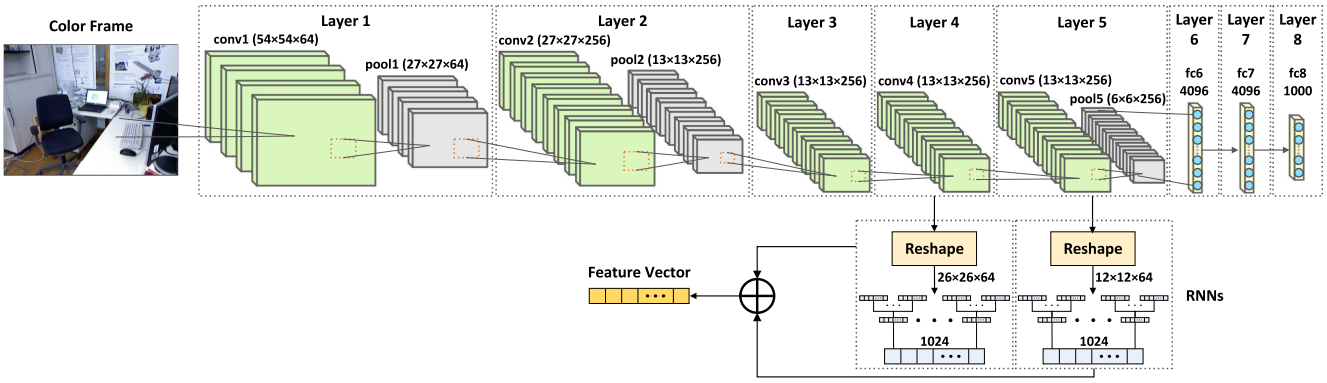

Figure 3. Structure of deep feature extraction mechanism.

uses a deformation graph to construct surfel based representations of room sized environments. In their system, recently observed parts of the model are labeled as active and tracking is performed by registering the current frame with the active area. Loop closure detection is performed using a fern encoding approach.

In comparison with conventional approaches that deal with handcrafted features and photometric constraints, deep learning approaches have been demonstrated as an effective solution in many visual mapping systems (e.g. [38, 31, 18, 44, 43]). CNN-SLAM [38] integrates predicted depth by convolutional neural networks into a monocular SLAM system. MapNet [18] builds a 2.5D spatial memory representation of the environment using recurrent neural networks. Parisotto *et al.* [31] estimates local poses with CNNs. Neural SLAM [43] employs an external memory and uses deep reinforcement learning to update pose beliefs. DeepTAM [44] performs camera tracking using an encoder-decoder architecture. In this respect, different from the abovementioned SLAM systems, we propose to use deep features of color frames in an indexing mechanism, in conjunction with sparse keypoint based motion estimation. The experimental evaluation reveals that our deep feature based RGB-D SLAM approach improves the mapping performance. The remaining sections explain the system components in detail.

3. System Structure

We use a pose graph that contains a node for each frame in the proposed system. Motion estimation is performed by extracting and matching keypoints between frames. A deep feature learning approach is employed for data association. Deep features extracted from color frames are indexed and searched through an efficient data structure. Outlier elimination from loop closure candidates are performed by a dynamic approach called adaptive thresholding [16]. The graph is optimized after processing all frames and the map is constructed using the estimated trajectory. Figure 2 shows the overall system structure.

3.1. Camera Tracking

In order to track the camera, pairwise transformations between successive frames are computed by utilizing sparse keypoints on color frames. CenSurE detector [1] and FREAK descriptor [2], which together is determined as the most successful combination in [15], are used for keypoint extraction. 3D keypoint coordinates are computed using the related depth frames. While keypoints are matched between frames according to the Hamming distance, the ratio approach of Lowe [27] is employed to filter ambiguous matches. After finding 3D keypoint correspondences between frames, RANSAC [14] is used to estimate the transformation through applying singular value decomposition in each iteration as in [3]. In this way, odometry estimation is performed by applying this procedure to each incoming frame with a number of most recent frames (predecessors). The camera is tracked by estimating the transformations between the incoming frame and the predecessors, and inserting the odometry edges into the graph between the sequential nodes.

3.2. Deep Feature Extraction

In the problem we deal with, it is important both acquiring effective features in a fast way and having a lower-dimensional final feature vector together. To this end, we employ a deep feature learning approach based on the recent work in [7] that presents discriminative features, to robustly obtain visual relations between observations. Our approach is based on a two-step deep feature extraction structure. In the first step, a sequential pre-trained CNN model [8] is employed as the underlying structure to extract the middle-layer features. In the second step, these features are reshaped and given to the multiple random recursive neural networks (RNNs) [35] to map higher level representations. Finally, the mid-level representations (4th and 5th layers) are combined to produce a strong global feature vector. The reason for particularly focusing on the mid-level representations is that these features provide a good tradeoff between

general features such as corners and edges and object semantics of the trained datasets [7, 6, 33, 42]. The activation maps obtained from the 4th and 5th layers of the CNN have $13 \times 13 \times 256$ and $6 \times 6 \times 256$ dimensions. We reshape these features into the forms of $26 \times 26 \times 64$ and $12 \times 12 \times 64$ to be given as inputs to the RNNs. The reason why the activation maps of the pre-trained CNN are reshaped is to reduce the dimension of the final feature vector obtained from the RNNs for the efficiency purpose. Then, we apply multiple RNNs to encode higher-level inference on a tree structure. In our approach, there is a single parent for each tree which merges r^2 number of K-dimensional leaf nodes with tied weights where the goal is to map inputs $X \in \mathbb{R}^{K \times r \times r}$ into a lower dimensional space $p \in \mathbb{R}^K$. The parent vector is computed by using a nonlinear squash function as follows:

$$p = f\left(W \begin{bmatrix} x_1 \\ \vdots \\ x_{b^2} \end{bmatrix}\right) \tag{1}$$

Here, $W \in \mathbb{R}^{K \times r^2 K}$ is a randomly initialized weight matrix and f represents a nonlinearity function which is $tanh$ in this work. Each tree produces a single parent with K-dimensional feature vector in this way. For N randomly initialized RNNs, the final feature vector size is a total of $(N \times K)$-dimensional matrix. Unlike the setting in [7], we specifically use $N = 16$ number of RNNs in order to ensure the efficiency in our approach. Consequently, our deep feature extraction procedure requires no training and provides highly efficient features in a fast way in accordance with our purpose.

3.3. Feature Indexing

The extracted features are stored in a *deep feature database*, which is composed of a priority search k-means tree [28]. The system uses keyframes at this step because processing the whole set of previous frames for each incoming frame may increase computational cost excessively. Therefore, the database holds the deep features obtained from the keyframes.

The priority search k-means tree structure is used to find approximate nearest neighbors of any point by clustering them according to distance of all dimensions. The system holds deep feature vectors as individual points in the tree. For example, a feature vector of a keyframe containing 1024 elements is handled as a point in 1024-dimensional space.

The system builds the priority search k-means tree incrementally by decomposing the search space hierarchically according to branching factor k (See Algorithm 1) and using the Manhattan distance. The first frame from the camera is selected as the first keyframe, thus the tree is constructed with deep feature vector of this keyframe as the first point. After determining each new keyframe during mapping, the related feature vector is added to the tree incrementally. In order to prevent unbalancing, the tree is rebuilt when the number of current keyframes in the tree doubles in size.

Algorithm 1: Building the priority search k-means tree of deep feature vectors incrementally

Input : *priority search k-means tree T, deep feature vector F, branching factor k, maximum iteration number I_{max}*
Output: *tree T with a new deep feature vector*

if *tree T doubles in size from last rebuild* **then**
  → *A: = F + all feature vectors in tree T*
  → *call* `ClusterPoints`(A, k, I_{max})
else
  → *Traverse tree T and reach the leaf node having the closest cluster to F*
  → *Add F to the cluster*
  if *size of the cluster $\geq k$* **then**
    → *D: = all feature vectors in the cluster*
    → *call* `ClusterPoints`(D, k, I_{max})
  end
end
Procedure `ClusterPoints`(A, k, I_{max})
  if *size of $A < k$* **then**
    → *create leaf node with the points in A*
  else
    → *L: = select k points from A randomly as cluster centers*
    → *assign the vectors to the closest cluster and adjust the centers until convergence or I_{max}*
    → *C: = obtained clusters*
    for *each cluster $C_i \in C$* **do**
      → *create non-leaf node for C_i*
      → *call* `ClusterPoints`(C_i, k, I_{max})
    end
  end
end
return *priority search k-means tree*

3.4. Loop Closure Detection

The system detects loop closures by performing priority search firstly, and then eliminating outliers by applying a filtering approach.

3.4.1 Candidate Search

For each incoming frame, loop closure candidates are retrieved by searching the deep feature database. It is performed through traversing the priority search k-means tree recursively. Deep feature vector of the incoming frame is generated firstly. Then the candidates are searched in the tree according to the Manhattan distance between feature vectors of the current frame and the keyframes (See Algorithm 2). During the tree traversal, the closest cluster is selected in each recursion and the other nodes on the path are inserted into a priority queue according to the distance. The nearest neighbors in the cluster of the reached leaf node are picked. If the cluster does not have enough points, the

closest node in the priority queue is chosen and the traversal is continued from that node.

Algorithm 2: Loop closure candidate search in the deep feature database

Input : *priority search k-means tree T, deep feature vector F, number of candidates C*
Output: *C candidates as approximate nearest neighbors of the feature vector F*

→ Traverse T by selecting closest cluster in each recursion and reach the leaf node
→ Insert other nodes on the path into a priority queue according to distance
→ Pick the nearest neighbors of F in the cluster of the leaf

if *leaf does not have enough feature vectors* **then**
→ Select the closest node in the priority queue
→ Continue traversal from there
end
return *C nearest neighbors of F*

3.4.2 Adaptive Thresholding

False positive loop closures may reduce the trajectory accuracy, since incorrect constraints between unrelated poses can cause optimization failures. Retrieving constant number of candidates in each loop closure search may trigger false positives by involving some unrelated keyframes (outliers). Thus, the adaptive thresholding technique in [16] is employed to filter outliers dynamically. This technique checks the keyframes in the candidate group selected in the first stage and removes less relevant ones from the group according to the deep feature similarity. Using the same distance metric (Manhattan), a dynamic similarity threshold is applied with respect to the closest keyframe in the group. The lowest distance score (which belongs to the closest keyframe) is multiplied by a *thresholding factor* to compute the similarity threshold. The keyframes that are less similar to the current frame's than this threshold are eliminated. Since each incoming frame has different visual characteristics, computing the similarity threshold dynamically for each candidate search allows to change the threshold value adaptively. Instead of applying a fixed threshold value, this adaptive mechanism enables the system to discard a considerable number of unrelated candidates in each loop closure search, which allows to reduce computational cost and improves the accuracy.

3.5. Map Construction

For each incoming frame, a new node is added to the pose graph. The edges representing odometry constraints are inserted between the new node and the predecessor nodes by camera tracking. At the same time, motion estimation is performed between the incoming frame and each of the loop closure candidates. In motion estimation, the same procedure in tracking is applied through using the already extracted and stored keypoints. Each computed transformation is verified according to the number of inlier keypoint correspondences. The transformations having insufficient number of inliers are discarded. For the verified transformations, the edges holding loop closure constraints are added to the graph between the new node and the related loop closing nodes. If the new node could not be connected to any keyframe node with the newly established edges (considering the predecessors and the loop closing nodes), the direct predecessor of the incoming frame is chosen as the new keyframe and the feature vector of this keyframe is inserted into the deep feature database. In this way, the dense pose graph is expanded through adding new nodes and edges. The g2o framework [22] is employed for graph optimization after processing all frames. Then, the environment map is built using the obtained trajectory. A point cloud reconstruction is generated by projecting sensor measurements at each pose into a common coordinate system.

4. Evaluation

4.1. Experimental Setup

We assess the proposed system on the TUM RGB-D benchmark [37]. We use *fr1* and *fr2* datasets of the benchmark, which are recorded in medium and large environments respectively and contain different environmental conditions.

In our experiments, the priority search k-means tree is built using the branching factor of k as 32 and the maximum iteration number of I_{max} as 11 for the balance between precision and efficiency. The experiments are performed using up to 20 loop closure candidates obtained from the deep feature database. For the *fr1* sequences, due to the medium-sized environment and short trajectories, the *thresholding factor* is applied as 2.0 in adaptive thresholding and low number of predecessors such as 5 are used in tracking. For the *fr2* sequences, the predecessor count is increased up to 30 and the *thresholding factor* is set as 3.0, since the environment is large and the trajectories are long. The experiments are carried out on a desktop PC with an Intel Core i7-2600 CPU at 3.40GHz and 8GB RAM.

4.2. Experimental Results

Table 1 and Table 2 show accuracy performance of the system in terms of root mean squared absolute trajectory error (RMS-ATE) and make comparison with the state-of-the-art systems. The presented RMS-ATE results for the other systems have been taken from the publications except that of ORB-SLAM2 [29], in which only *fr1/desk*, *fr1/desk2*, and *fr1/room* results are presented in the original paper. Thus, the original implementation of ORB-SLAM2 provided by the authors has been run (using the proposed

Table 1. Comparison in terms of RMS-ATE (root mean square of absolute trajectory error in meters) on *fr1* sequences of the TUM RGB-D benchmark [37]

System	*fr1/360*	*fr1/desk*	*fr1/desk2*	*fr1/floor*	*fr1/plant*	*fr1/room*	*fr1/teddy*
KinectFusion [30]	0.913	0.057	0.420	-	0.598	0.313	0.154
Kintinuous [39]	-	0.037	0.071	-	0.047	0.075	-
ElasticFusion [40]	0.108	0.020	0.048	x	0.022	0.068	0.083
DVO SLAM [20]	0.083	0.021	0.046	-	0.028	0.053	**0.034**
ORB-SLAM2 [29]	0.180	**0.016**	**0.022**	x	**0.014**	**0.047**	0.047
MRSMap [36]	0.069	0.043	0.049	x	0.026	0.069	0.039
RGB-D SLAM [13]	0.079	0.026	0.043	0.035	0.091	0.087	0.076
Extended RGB-D SLAM [16]	0.075	0.022	0.034	0.032	0.068	0.054	0.060
This work - L4	0.052	0.020	0.031	0.028	0.036	0.050	0.040
This work - L5	0.056	0.020	0.028	0.028	0.037	0.049	0.039
This work - L4 + L5	**0.051**	0.019	0.029	**0.027**	0.034	0.049	0.036

L4 indicates the result of 4th layer as a feature vector, L4+L5 denotes the combination of 4th and 5th layers.

x indicates tracking failure of the cited method.

- sign means the result does not exist for the cited work.

instructions) on the other sequences for the sake of comparisons. Also we have reproduced the results of all the *fr1* sequences for the Extended RGB-D SLAM [16].

The experimental results for the *fr1* sequences in Table 1 show that the proposed system produces drift errors around 2 to 5 cm. For most sequences, our system competes with ORB-SLAM2 by showing very close performances where the differences are at the millimeter level (e.g. For *fr1/desk* our result of 1.9 cm is against their 1.6 cm, while for *fr1/room* our result of 4.9 cm is against their 4.7 cm). The *fr1* sequences contain scenes of a medium-sized office, which is highly textured and thus rich in keypoints. There are short trajectories and some sequences have small loop closures. Therefore, odometry estimation performance affects the accuracy more than loop closure detection. Consequently, for a medium sized and highly textured environment such as *fr1*, our system is highly competitive with the state-of-the-art systems by mostly producing the best or the second best result after ORB-SLAM2.

The *fr2* sequences have been recorded in a large industrial hall. There are long trajectories and large loop closures in most of these sequences. Moreover, they have highly challenging conditions including missing depth data (due to the range limit), sensor outage, textureless areas, less visual features, repetitive structures, motion blur, wrong depth data (due to thin structures), and illumination changes. Thus, the large *fr2* sequences are convenient to evaluate large-scale indoor mapping performance in challenging situations. In the experiments, we have used all the *fr2* sequences traversing the industrial hall with long trajectories. The *fr2* accuracy results in Table 2 demonstrate that the proposed system extends the state-of-the-art performance. The system works robustly on these sequences, where data association performance is crucial due to long trajectories and sensor outage. The observed drift errors change around 13 cm - 35 cm, which are acceptable and promising results for such a challenging environment. The trajectories estimated by the proposed system show its robustness (see Figure 5). For the ElasticFusion [40] and ORB-SLAM2 [29] systems, tracking failures are observed for considerable parts of the *fr2* sequences. ORB-SLAM2 shows a promising result for *fr2/large_no_loop* but fails tracking on other sequences. As stated in ElasticFusion [40], the reasons behind the *fr2* sequence failures might be sensor outage, missing depth data, and high angular velocity of the sensor. Besides these factors, data association difficulty might be another important reason due to repetitive structures and low texture. Likewise, these challenges may be the reasons why the *fr2* sequences with large trajectories are not used for evaluation in the most SLAM studies.

As for our deep learning approach, since we use a CNN model trained on the ImageNet [12] as the underlying feature extractor, our focus on the intermediate layers makes sense. Because these activation maps have been shown as the optimal representations as stated in [7]. To this end, we evaluate both individual middle layers and the combination of them as well. Using the features obtained from the individual layers separately, the estimation of our approach already gets at par with the state-of-the-art. The combination of layers 4 and 5 further enhances the performance and outperforms the other methods for many of the sequences. As for the *fr2/p_slam* sequences, our approach using the features from the 4th layer performs better than the combination of layers with a slight difference.

Table 2. Comparison in terms of RMS-ATE (root mean square of absolute trajectory error in meters) on *fr2* sequences of the TUM RGB-D benchmark [37]

System	*fr2/large_no_loop*	*fr2/large_with_loop*	*fr2/p_360*	*fr2/p_slam*	*fr2/p_slam2*	*fr2/p_slam3*
KinectFusion [30]	-	-	-	-	-	-
Kintinuous [39]	-	-	-	-	-	-
ElasticFusion [40]	x	x	x	x	x	x
DVO SLAM [20]	-	-	-	-	-	-
ORB-SLAM2 [29]	0.337	x	x	x	x	x
MRSMap [36]	-	-	-	-	-	-
RGB-D SLAM [13]	0.860	3.598	0.213	0.367	0.381	0.511
Extended RGB-D SLAM [16]	0.395	0.367	0.213	0.349	0.400	0.341
This work - L4	0.208	0.359	0.152	**0.310**	**0.157**	**0.267**
This work - L5	0.147	0.349	0.152	0.438	0.163	0.285
This work - L4 + L5	**0.135**	**0.344**	**0.148**	0.380	0.160	0.272

L4 indicates the result of 4th layer as a feature vector, L4+L5 denotes the combination of 4th and 5th layers.

x indicates tracking failure of the cited method.

- sign means the result does not exist for the cited work.

fr2/p_360 : *fr2/pioneer_360*, *fr2/p_slam* : *fr2/pioneer_slam*

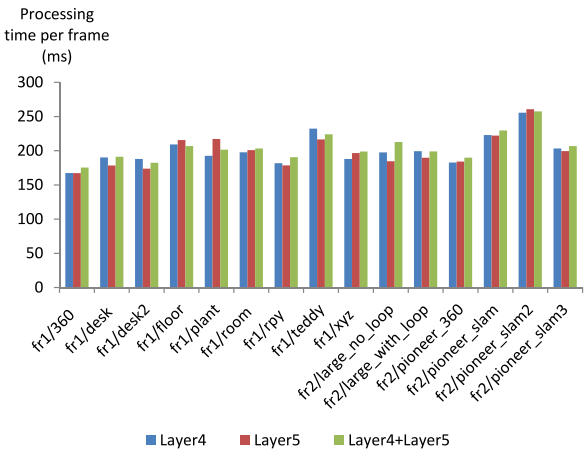

Figure 4. Average processing time per frame in milliseconds

4.3. Computational Performance

In Figure 4, average frame processing times are presented for each sequence using different layer features. The proposed system runs at 5 Hz generally on a CPU. Keypoint extraction time is ∼45 ms and motion estimation cost is ∼50 ms per frame. Deep feature extraction phase takes ∼70 ms for each frame and combining the feature layers has a negligible cost. It is possible to obtain deep features in a much shorter time using GPUs. However, we rather prefer using CPUs in order to fit the general CPU based structures of the proposed system. Loop closure search in the deep feature database takes less than 1 ms, which is not affected much by the size of the environment for the tested sequences. Normally, as the map expands, more observations needs to be searched and the cost is expected to increase. However, the indexing mechanism enables the system to find visual relations fast by hierarchically clustering deep feature vectors as points in high dimensional space. Therefore, keyframe feature vectors can be searched with low computational effort, even in large environments.

5. Conclusion

In this paper, we present an RGB-D SLAM system that employs deep features and sparse motion estimation to build 3D maps of indoor environments. A robust and efficient approach based on deep feature indexing has been developed to search observations and to find visual relations between frames. The system utilizes a unified deep feature learning approach, which incorporates a pre-trained CNN model with multiple random RNNs to acquire distinctive features by reducing the feature space dimensionality. The CNN-RNN network takes color frames as inputs and extracts highly efficient robust features. The deep feature extraction stage requires no training or fine-tuning. Moreover, the applied RNNs use non-overlapping receptive fields which make them computationally fast. The feature vectors are indexed in a priority search k-means tree as individual points, which allows to find potential visual associations fast by recursively traversing the tree. Outlier candidates are filtered using an efficient method called adaptive thresholding. The experimental results show that the system works in medium and large indoor environments consistently and outperforms other state-of-the-art systems especially in challenging large environments. As a future work, we plan to develop a GPU based implementation of the system for real-time performance.

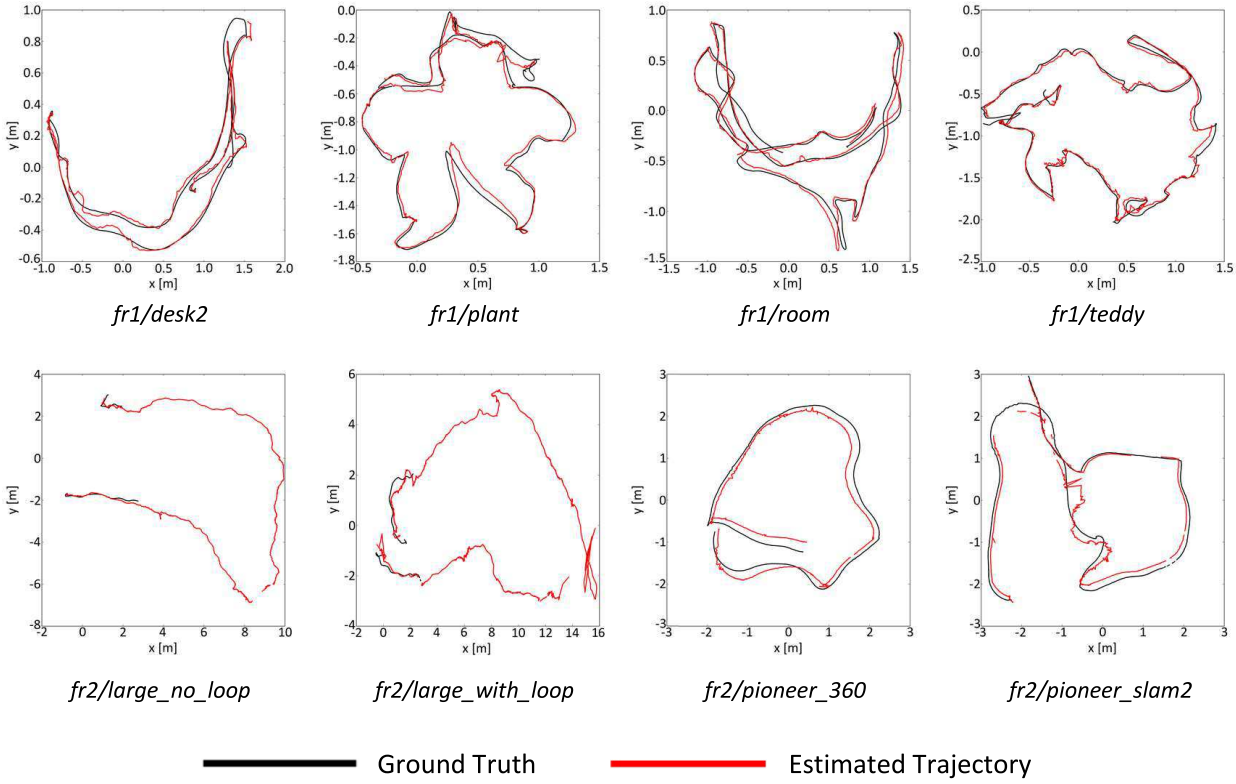

Figure 5. 2D plane projections of ground truth and estimated trajectories.

References

[1] M. Agrawal, K. Konolige, and M.R. Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*, pages 102–115. Springer, 2008. 3

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE, 2012. 3

[3] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987. 3

[4] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 2

[5] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proc. CVPR*, volume 8, 2018. 1

[6] Hieu Minh Bui, Margaret Lech, Eva Cheng, Katrina Neville, and Ian S Burnett. Object recognition using deep convolutional features transformed by a recursive network structure. *IEEE Access*, 4:10059–10066, 2016. 4

[7] Ali Caglayan and Ahmet Burak Can. Exploiting multi-layer features using a cnn-rnn approach for rgb-d object recognition. In *Computer Vision – ECCV 2018 Workshops*, pages 675–688, Cham, 2019. Springer International Publishing. 1, 3, 4, 6

[8] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014. 3

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. 1

[10] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[11] Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid, and Michael Milford. Deep learning features at scale for visual place recognition. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3223–3230. IEEE, 2017. 1

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 6

[13] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-d mapping with an rgb-d camera. *Robotics, IEEE Transactions on*, 30(1):177–187, 2014. 2, 6, 7

[14] 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*, 24(6):381–395, 1981. 2, 3

[15] O. Guclu and A.B. Can. A comparison of feature detectors and descriptors in rgb-d slam methods. In *Image Analysis and Recognition*, pages 297–305. Springer, 2015. 3

[16] Oguzhan Guclu and Ahmet Burak Can. Fast and effective loop closure detection to improve slam performance. *Journal of Intelligent & Robotic Systems*, pages 1–23, 2017. 2, 3, 5, 6, 7

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[18] Joo F. Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[19] Alex Kendall, Matthew Grimes, and Roberto 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. 1

[20] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2100–2106. IEEE, 2013. 2, 6, 7

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[22] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011. 5

[23] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[26] Manuel Lopez-Antequera, Ruben Gomez-Ojeda, Nicolai Petkov, and Javier Gonzalez-Jimenez. Appearance-invariant place recognition by discriminatively training a convolutional neural network. *Pattern Recognition Letters*, 92:89–95, 2017. 1

[27] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 3

[28] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014. 1, 4

[29] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 2, 5, 6, 7

[30] 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. 2, 6, 7

[31] Emilio Parisotto, Devendra Singh Chaplot, Jian Zhang, and Ruslan Salakhutdinov. Global pose estimation with an attention-based recurrent network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 3

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[33] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 4

[34] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126, 2015. 1

[35] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in neural information processing systems*, pages 656–664, 2012. 1, 3

[36] Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3d modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1):137–147, 2014. 2, 6, 7

[37] 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*, pages 573–580. IEEE, 2012. 1, 2, 5, 6, 7

[38] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3

[39] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015. 2, 6, 7

[40] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion:

Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016. 2, 6, 7

[41] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016. 1

[42] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 4

[43] Jingwei Zhang, Lei Tai, Joschka Boedecker, Wolfram Burgard, and Ming Liu. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017. 3

[44] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *The European Conference on Computer Vision (ECCV)*, September 2018. 3

[45] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1