Geometric Correspondence Network for Camera Motion Estimation

Jiexiong Tang, John Folkesson, and Patric Jensfelt

***Abstract*—In this paper, we propose a new learning scheme for generating geometric correspondences to be used for visual odometry. A convolutional neural network (CNN) combined with a recurrent neural network (RNN) are trained together to detect the location of keypoints as well as to generate corresponding descriptors in one unified structure. The network is optimized by warping points from source frame to reference frame, with a rigid body transform. Essentially, learning from warping. The overall training is focused on movements of the camera rather than movements within the image, which leads to better consistency in the matching and ultimately better motion estimation. Experimental results show that the proposed method achieves better results than both related deep learning and hand crafted methods. Furthermore, as a demonstration of the promise of our method we use a naive SLAM implementation based on these keypoints and get a performance on par with ORB-SLAM.**

***Index Terms*—Visual-based navigation, SLAM, deep learning in robotics and automation.**

I. Introduction

Motion estimation is a fundamental part of mobile robotic systems. The advantages of being able to estimate motion solely using a RGB-D sensor are significant. The sensors are relatively low weight, low power and low cost compared to lidar for example. At the same time, they can give accurate estimates under the right conditions. Estimates based on a single sensor are easier to integrate into a system than ones based on a suite of sensors, as testing and validation can be done with the sensor detached from the system.

In this paper, we focus on visual odometry (VO), i.e. frame to frame motion estimation using information from a vision sensor. We investigate the use of a deep neural network to learn, in an end-to-end fashion, how to both detect keypoints and how to match them using a RGB-D sensor. We optimize only for geometric correspondence rather than semantic correspondence. A convolutional neural network (CNN) and a recurrent neural network (RNN) are trained together to perform both keypoint detection and feature descriptor generation in one unified structure which we call the Geometric Correspondence Network (GCN). To train the network we extract high gradient points in one image and warp them using the camera motion to the next image. It is important to note that we do not try to learn to reproduce the high gradient points or any standard descriptors based on them, but rather the network learns to find features and descriptors that can be matched well. In fact, the points in the warped image may not even be as detectable based on the gradient as they were in the unwarped image. We make use of a benchmark dataset for SLAM [1] for which ground truth positions are available to perform the warping to generate the training data. We compare GCN against a number of other visual features for visual odometry using the same pipeline.[1] Fig. 1 shows an example of 3D reconstruction using GCN. To further evaluate the method we implement a simple SLAM backend[2] to allow for loop closure and compare our results to other SLAM methods.

Decades of work has gone into engineering features for recognition and classification of objects, place recognition, loop closure detection, stereo image matching, etc. Recently, deep neural networks have been shown to outperform hand crafted features for all these tasks. The problem of finding correspondences for frame to frame keypoint matching for a visual odometry is a little investigated area in comparison. With consideration to that, achieving results that match popular methods is encouraging.

In summary, the main contribution of our work is a framework (GCN) for simultaneously learning both feature locations (for motion estimating) and descriptors (for matching) optimized for the task of camera motion estimation.

II. Related Work

This paper is related to many different well researched domains. We will focus on the relation to existing VO/SLAM solutions and deep learning used in related applications.

Existing methods to VO and SLAM can be divided into sparse, or feature based, methods and dense, or direct, methods. In the sparse methods, keypoints are first extracted and then they are matched based on geometric constraints and local descriptors. The motion is estimated based on the correspondences between points in two frames. ORB-SLAM [2], [3] represents the state-of-the-art for SLAM today in this category. ORB [4] features are extracted in new frames and are then matched to recent frames to perform camera motion estimation and to distant frames using a binary bag of words representation [5] to

Manuscript received September 10, 2017; accepted January 3, 2018. Date of publication January 17, 2018; date of current version February 1, 2018. This paper was recommended for publication by Associate Editor U. Frese and Editor C. Stachniss upon evaluation of the reviewers' comments. This work was supported in part by the Wallenberg Autonomous Systems and Software Program and in part by the European Unions Horizon 2020 research and innovation program under Grant 644839 (CENTAURO). *(Corresponding author: Jiexiong Tang.)*

The authors are with the Centre for Autonomous Systems at KTH Royal Institute of Technology, Stockholm SE-10044, Sweden (e-mail: jiexiong@kth.se; johnf@kth.se; patric@kth.se).

Digital Object Identifier 10.1109/LRA.2018.2794624

[1]OpenCV's PnP with RANSAC.
[2]Pose graph plus optimization.

detect loop closures. ORB-SLAM is available in implementations for monocular, stereo and RGB-D sensor setups. In dense methods, information from all the pixels in the image are used, thus eliminating the need for feature detection. Optimizing is instead done directly using the intensities in the images. In [6], Comport *et al.* present a visual odometry system able to estimate the motion of the sensor with a vertical error of only 20 cm over a distance of 200 m in a urban driving scenario. Kerl *et al.* present the Direct Visual Odometry (DVO) method in [7]. They build a pose graph and use g2o [8] for optimization to reduce the error. As an alternative to matching frame to frame, KinectFusion [9], Kintinous [10] and ElasticFusion [11] build a volumetric representation to which new frames are matched. RGBDTAM [12] uses a combination of both semi-dense photometric and dense geometric errors for pose estimation. The methods SVO [13] and LSD-SLAM [14] fall somewhere in between dense and sparse, combining the best traits from the feature based methods with those from the dense methods. The work in our paper is most closely related to feature based methods such as ORB-SLAM. However, whereas in ORB-SLAM they use an off-the-shelf feature (ORB) and design a complete pipeline around it, we focus on the design of the feature itself and thus complement rather than compete.

Deep learning based methods are already defining the state-of-the-art in many applications in computer vision. As a key component for 3D scene reconstruction, stereo matching using CNNs have been shown to perform very well [15]–[17]. In recent work, [18], the efficiency of deep stereo matching is improved by exploiting an inner product layer between two feature representations. Deep learning has also been applied to predict optical flow. DeepFlow [19] used DeepMatch [20] to compute optical flow using multi-layer architectures that match sub-patches on different scales. To reduce the computational cost, a faster and edge preserving method have been proposed in [21]. Combing DeepMatch with contour detection, EpicFlow [22] interpolates matches and obtain a dense estimation. In FlowNet [23], [24], a CNN is trained by a large synthetic dataset [25] to directly predict per-pixel flow. This line of work is closely related to our work as optical flow can be used to estimate the camera motion. However, we focus more on detecting and matching the points that are best suited for estimating the camera motion.

In another stream of work, a network is trained to estimate the depth from monocular images [26], [27], in essence creating a virtual RGB-D sensor for a standard monocular camera. In CNN-SLAM [28] this approach is used in a mapping framework, allowing them to recover the scale using only monocular information. Using a two-frame structure instead of one, DeMoN [29] proposed a novel architecture composed of multiple stacked iterative encoder-decoder networks.

Place recognition is another area for which deep nets have been applied with great success [30]–[33]. Similar to us [31] makes use of siamese networks. They are fine-tuned to match aerial and ground-level images. Addressing the same problem of matching aerial and ground level images is [34] where a recursive network outputs keypoint responses at different scales. Patches are extracted around these keypoints and descriptors are generated. Aiming for more general descriptor learning, methods [35]–[37] applied metric learning to image patches. Like in our work, the aim is to learn to both detect and describe keypoints. We differ by using a fully convolutional structure to generate the descriptors in a dense way. Dense feature structures only need $O(n)$ feed forward passes for testing which is faster than typical patch similarity based networks which require $O(n^2)$ [38].

In [39], the camera motion is estimated using features from a siamese network from color images in a classification manner. The translations and rotations are discretized within pre-defined ranges to a number of bins. The network is trained to classify the ego motion with classes defined as these bins. The discretization limits the accuracy of the motion estimation and the pre-defined range limits allowed motion of the camera.

In [38], the Universal Correspondence Network (UCN) is presented to extract dense features for applications where one needs geometric and semantic correspondences. In contrast to patch similarity based methods, UCN directly optimizes features for visual correspondence. This is the work that is closest to our work. But the aim is a universal approach rather than an approach targeting a specific application like ours. In an evaluation of camera motion estimation, they show that UCN performs better than [39] despite being more general. However, it is only on par with sparse features that have been designed for the task of motion estimation. Though general as a visual correspondence framework and outperforming patch similarity based methods in several settings, it does not beat traditional methods in the context of camera motion estimation. In contrast to UCN, we focus specifically on camera motion estimation and show that this allows us to outperform hand crafted features and as well as UCN for this task.

In summary, our work is closely related to VO and SLAM work. However, we focus on learning a way to produce keypoints to support the camera motion estimation rather than the estimation problem itself. We use a deep learning approach specifically for the task of camera motion estimation rather than the general problem of generating correspondences such as UCN [38].

III. Motion Estimation

In this section, we briefly review the feature based Bundle Adjustment (BA) algorithm which is used at the core of the motion estimation. Let points in the reference image be denoted as $\boldsymbol{x_i} = [u_i, v_i]^T, i \in \{1, \ldots, N\}$ and their correspondences in current image $\boldsymbol{y_i} = [u'_i, v'_i]^T, i \in \{1, \ldots, N\}$. To align the two images and estimate the relative transform between them, we first project $\boldsymbol{y_i}$ to 3D coordinates with the warping function $\pi(.)$:

$$\pi(\boldsymbol{x_i}, d_i) = \left[\frac{d_i(u_i - c_x)}{f_x}, \frac{d_i(v_i - c_y)}{f_y}, d_i\right]^T \tag{1}$$

where d_i is the depth measurement of the corresponding point and c_x, c_y, f_x, f_y are the intrinsic camera parameters. After the projection, we warp the 3D points with a rigid body transformation and project them back to the image plane. Geometrical errors caused by mis-alignment can be calculated as follows:

$$E_{geo}(\boldsymbol{\xi}) = \sum_i r_i(\boldsymbol{\xi}) = \sum_i (\boldsymbol{x_i} - \pi^{-1}(\mathbf{R}\cdot\pi(\boldsymbol{y_i}, d_i) + \boldsymbol{t}))^2 \tag{2}$$

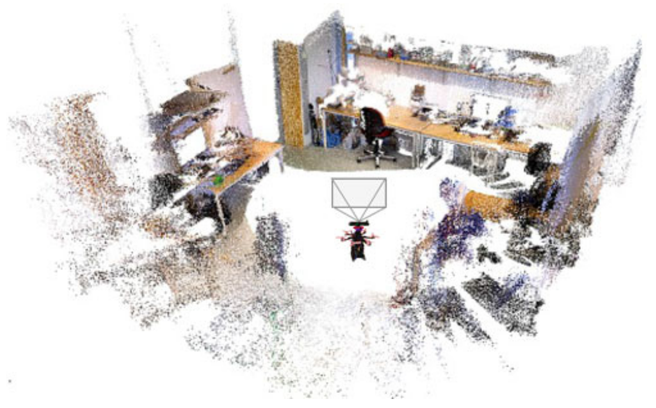

Fig. 1. RGB-D frames accumulated using frame-to-frame camera motion estimation (i.e., using no keyframes and loop closures) with our proposed method, GCN, for keypoint generation and matching. Data is collected by a RGB-D sensor on a hexacopter flying around the room.

where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ is the translation vector. They together compose a rigid body transform $\exp(\hat{\boldsymbol{\xi}}) \in \mathbb{SE}(3)$, which is defined by $\boldsymbol{\xi} = [\boldsymbol{\omega}^T, \mathbf{t}^T]^T \in \mathfrak{se}(3)$. $\boldsymbol{\xi}$ is a member of the Lie algebra and it is mapped to the Lie group $\mathbb{SE}(3)$ through the matrix exponential $\exp(.)$:

$$\exp(\hat{\boldsymbol{\xi}}) = \begin{bmatrix} \mathbf{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3}$$

where $[\boldsymbol{\omega}]_\times$ is the skew-symmetric matrix of $\boldsymbol{\omega}$.

The estimated relative pose can be obtained by optimizing the residual error in (2). The Gaussian-Newton (GN) method is used to solve this non-linear least square problem. GN calculates $\boldsymbol{\xi}$ iteratively as follows:

$$\boldsymbol{\xi}^{(n+1)} = \boldsymbol{\xi}^{(n)} - \left(\mathbf{J_r}^\mathsf{T}\mathbf{J_r}\right)^{-1}\mathbf{J_r}^\mathsf{T}\mathbf{r}(\boldsymbol{\xi}^{(n)}) \tag{4}$$

where $\mathbf{J_r}$ is the Jacobian matrix with respect to the residual measurements.

IV. Proposed Method

The overall network structure of GCN can be seen in Fig. 2. The output of a deep CNN feeds into a shallow recurrent network to perform both dense feature extraction and keypoint detection. It is well known that good points to use for estimating rigid body transformations are those with high gradients in both horizontal and vertical directions in the image. In contrast to [38], we therefore train with input from keypoint detectors rather than random points. However, it is not guaranteed that the same keypoints will appear in two images, despite being consecutive. To tackle this problem during training, keypoints in first image are extracted and warped to the next frame, using the known frame to frame transformation. These then serve as the ground truth correspondences. This way, we optimize the network for the case where correspondences of points can be found by warping them with a rigid body transform, not by their appearance in the second frame. This method will not match

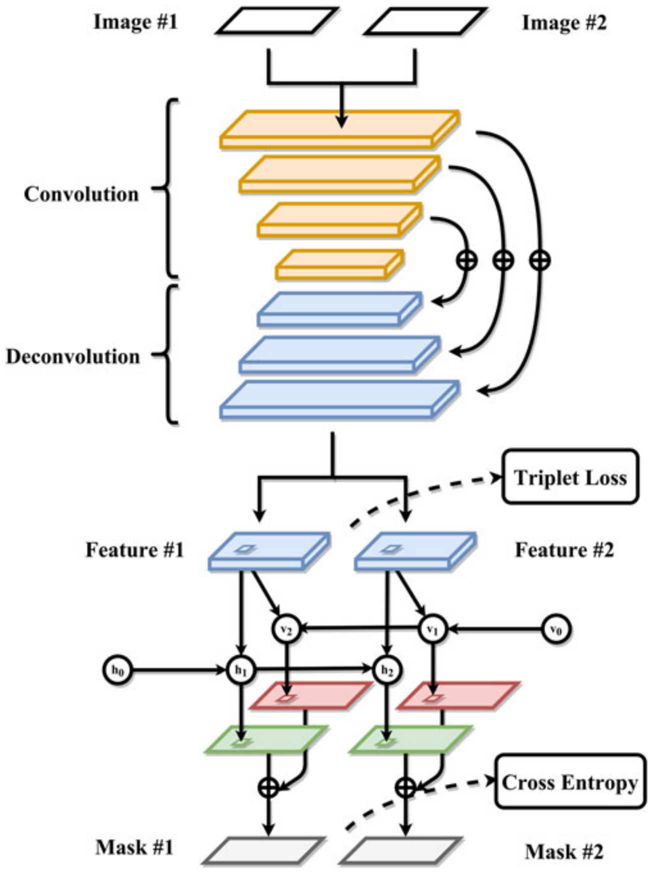

Fig. 2. The overall learning scheme and data flow of GCN is shown with colors represent different blocks. The upper part is the CNN that extracts features in a dense way. It consists of a fully convolutional part for multi-scale feature representation and a deconvolutional part for detail refinement. The lower part is a shallow bi-directional recurrent structure to predict the location of keypoints in both images. Note that the actual networks have more hidden layers than drawn. The objective functions for the training are the triplet loss and the cross entropy defined in (5) and (9) respectively.

points that have similar appearances in the two images but are not actually geometric points such as lines formed by crossing edges at different depths. That does not mean such poor features will not be found by the trained net but at least we are not using them for training.

In the remainder of this section, we first introduce deep metric learning for training dense feature descriptors. Then, we present the method for jointly training a detector using temporal information of resultant deep features. Finally, we summarize the overall multi-task learning scheme.

A. Dense Feature Extraction

Pyramid Network Backbone: As shown in Fig. 2 , the proposed network structure for extracting dense feature can be divided into two parts: one convolutional and one deconvolutional part. For the convolutional part we used ResNet-50 [40], which is a 50 layer CNN with a bottleneck structure and batch normalization. As shown in Fig. 4, ResNet-50 makes a good trade off between performance and network complexity. Using deeper version gives slightly better matching accuracy but at the cost of using more than twice the number of parameters. We didn't use shallower architecture either since the 50 layer version can generate features in real-time with our desktop setup. The existence of "shortcuts" makes features more fine-grained, making them well suited for visual tracking tasks that require pixel level precision. A shortcut is a direct concatenation of feature maps by summing over channels. The weights of ResNet-50 are pre-trained with the ImageNet [41] object classification dataset with 1.2 million labeled images. When the image is fed into ResNet-50, its size is gradually reduced by pooling operations and convolution with stride larger than one. To recover the features to its original size, the deconvolutional part is used for upsampling the features. Inspired by the structure in [42], multiple shortcut paths are made between intermediate layers of the convolutional and the deconvolutional parts to obtain more fine-grained feature.

Deep Metric Learning: Metric learning is applied to fine-tune the CNN as a dense feature extractor. Metric learning maps the input samples to a feature space where similar samples are closer and dissimilar samples are farther with the ℓ_2 distance as metric. Training with metric learning allows the features to be optimized for nearest neighbor matching.

We use triplet loss [43] to perform the metric learning. It has an objective function that penalizes three samples at a time: one anchor point and its paired positive/negative samples. In proposed method, we calculate the triplet loss on every point candidate as follows:

$$L_{\text{metric}} = \sum_i \max(0, s^2_{\boldsymbol{x_i},\boldsymbol{y_i}^*} - s^2_{\boldsymbol{x_i},\boldsymbol{z_i}^*} + m)$$
$$s_{\boldsymbol{x},\boldsymbol{y}} = ||\boldsymbol{f_1}(\boldsymbol{x}) - \boldsymbol{f_2}(\boldsymbol{y})||_2 \quad (5)$$

where $s_{\boldsymbol{x},\boldsymbol{y}}$ is the ℓ_2 distance between two feature vectors at location $\boldsymbol{x}$ and $\boldsymbol{y}$ from output feature $\boldsymbol{f_1}$ and feature $\boldsymbol{f_2}$ (shown in Fig. 2). m is a margin determining how far dissimilar points should be pushed away in feature space. $\boldsymbol{x_i}$ is called the anchor point which is the keypoint in frame #1. Points $\boldsymbol{y_i}^*$ and $\boldsymbol{z_i}^*$ are its positive and negative matched points in frame #2. The location of positive matched point $\boldsymbol{y_i}^*$ is obtained by warping $\boldsymbol{x_i}$ with ground truth $\mathbf{R}^*, \boldsymbol{t}^*$:

$$\boldsymbol{y_i}^* = \pi^{-1}(\mathbf{R}^* \cdot \pi(\boldsymbol{x_i}, d_i) + \boldsymbol{t}^*) \quad (6)$$

The location of the negative sample z_i^* is found by hardest negative sample mining:

$$\boldsymbol{z_i}^* = \underset{\boldsymbol{z_i}|\boldsymbol{z_i}\neq\boldsymbol{y_i}^*}{\operatorname{argmin}}\, s_{\boldsymbol{x_i},\boldsymbol{z_i}} \quad (7)$$

That is, pick the closest point in feature space that is not a positive sample. Any point other than the true match can be used as the negative pair for the anchor, but the hardest negative sample will contribute the most to the loss function, and thus the gradient, and thereby accelerates the metric learning.

B. Recurrent Mask Prediction

Recurrent Structure: We treat the keypoint detection as a binary classification problem which uses the dense features as input. To predict the keypoint locations simultaneously for the two input images, we want the network to exploit temporal information as well as spatial information. We use a shallow bi-directional recurrent convolutional network (RCNN) to achieve this goal. The proposed RCNN can be formulated as follows:

$$\begin{aligned}
\boldsymbol{h_1} &= \mathrm{ReLU}(\mathbf{W_{i2h}} \circ [\boldsymbol{f_1}, \boldsymbol{h_0}]), \boldsymbol{h_2} = \mathrm{ReLU}(\mathbf{W_{i2h}} \circ [\boldsymbol{f_2}, \boldsymbol{h_1}]) \\
\boldsymbol{v_1} &= \mathrm{ReLU}(\mathbf{W_{i2h}} \circ [\boldsymbol{f_1}, \boldsymbol{v_0}]), \boldsymbol{v_2} = \mathrm{ReLU}(\mathbf{W_{i2h}} \circ [\boldsymbol{f_2}, \boldsymbol{v_1}]) \\
\boldsymbol{o_1} &= \mathrm{softmax}(\mathbf{W_{h2o}} \circ \boldsymbol{h_1} + \mathbf{W_{h2o}} \circ \boldsymbol{v_2}), \\
\boldsymbol{o_2} &= \mathrm{softmax}(\mathbf{W_{h2o}} \circ \boldsymbol{h_2} + \mathbf{W_{h2o}} \circ \boldsymbol{v_1}), \quad (8)
\end{aligned}$$

where $\circ$ is the convolution operation, $\boldsymbol{f}$ is the feature map from the last layer before the recurrent network. $\boldsymbol{h}$ and $\boldsymbol{v}$ are two hidden states used to store information and $\boldsymbol{o}$ is the output at a given location where a sigmoid function will be applied to measure the probability of being a keypoint. $\mathbf{W_{i2h}}$ and $\mathbf{W_{h2o}}$

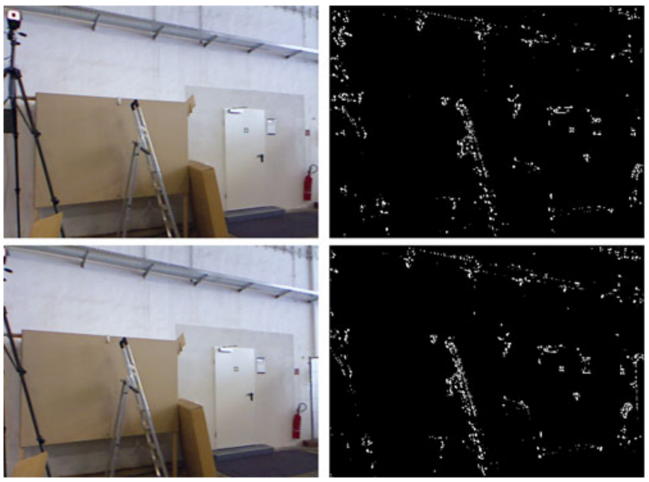

Fig. 3. An example of keypoint prediction using GCN for the TUM datasets [1]. The two color images are the input for the GCN. In each row, the two columns are raw RGB image and prediction, respectively. For the ground truth, points with no depth or warped points that are outside of image boundary are discarded.

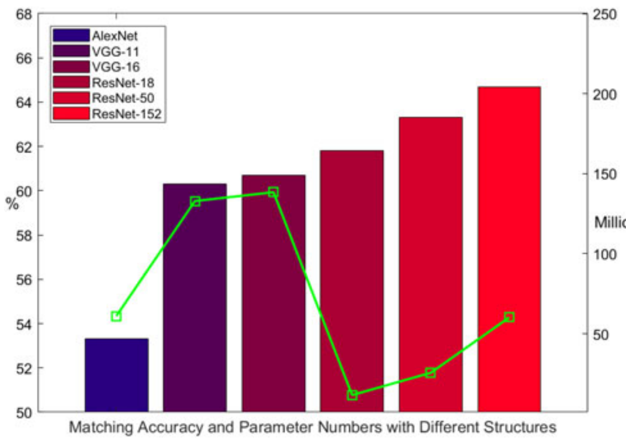

Fig. 4. Comparison between different structures using KITTI Raw [44] road scenes, we used sequences $[1, 2, 5, 9]$ for training and $[11, 13]$ for testing in this test. The bars represent overall matching accuracy which is shown on the left vertical axis and the green dots represent number of parameters, shown on the right vertical axis.

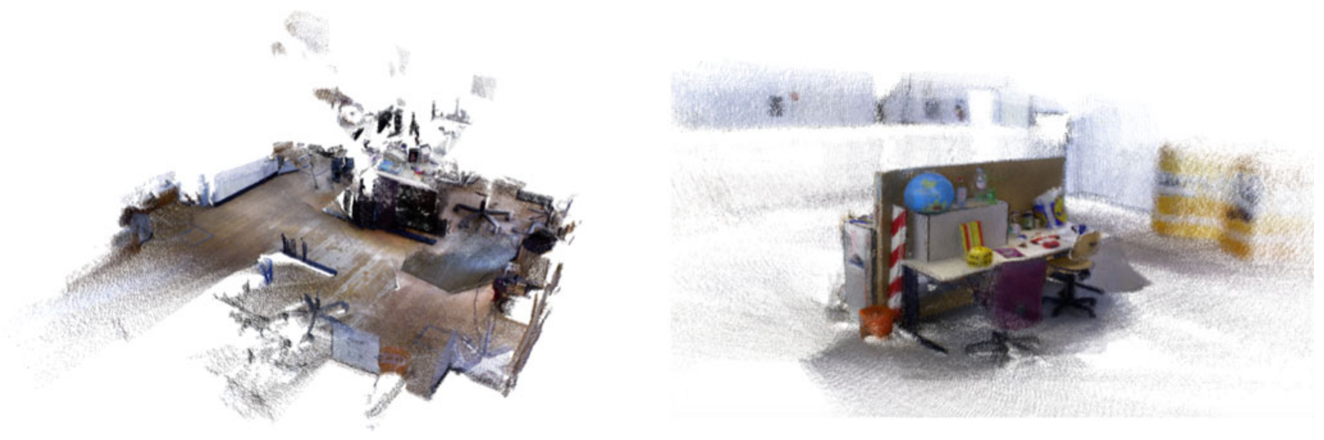

Fig. 5. Raw pointcloud reconstruction using fr1-floor and fr3-long. More can be found at: www.cas.kth.se/gcn.

are hidden weights to be learned and represent weights for the input-to-hidden path and the hidden-to-output path, respectively.

Equation (8) shows the data flow of the proposed recurrent structure. By using a bi-directional structure, we directly optimize the prediction of the RCNN by maximizing the consistency in both positive and negative time orders. This is similar to [3] where alignment of images is performed in both orders to utilize as much information as possible from them.

Mask Classification: The target for the keypoint classification is a mask with label 0 or 1 for each pixel, to indicate whether it is a keypoint or not (right column in Fig. 3). The masks are generated by giving keypoints in first images a value 1 and warping them to the next image. Then, a weighted cross entropy is calculated over every pixel as follows:

$$\begin{aligned} L_{mask} &= L_{ce}(\boldsymbol{o_1}, \boldsymbol{x}) + L_{ce}(\boldsymbol{o_2}, \boldsymbol{y}^*) \\ L_{ce}(\boldsymbol{o}, \boldsymbol{x}) &= -\sum_i (\alpha_1 c_{\boldsymbol{x_i}} \log(\text{sigmoid}(\boldsymbol{o}(\boldsymbol{x_i}))) \\ &\quad + \alpha_2 (1 - c_{\boldsymbol{x_i}}) \log(1 - \text{sigmoid}(\boldsymbol{o}(\boldsymbol{x_i})))\end{aligned} \tag{9}$$

where c is the label of a given 2D point and $\boldsymbol{o}$ is the predicted keypoint. $\boldsymbol{x_i}$ and $\boldsymbol{y_i}^*$ are keypoints and their correspondences obtained by warping, respectively, as in the previous subsection. α is a parameter to balance between different classes. This weighting is important for the training convergence as there is an obvious imbalance among these two classes: the number of keypoints (thousands) versus the total number of pixel (hundreds of thousands).

C. Multi-Task Training

The final loss for the multi-task training is a combination of triplet loss in (5) and cross entropy in (9). These are weighted equally. The adaptive gradient decent method, ADAM [45], is used for optimizing this combined loss. The weights are randomly initialized except for ResNet-50. The length of the feature vector is set to 64. The improvements from using a higher dimensional feature vector was minor and a shorter feature vector caused training not to converge, in our tests. As commonly done in fine tuning, the learning rate for ResNet-50 is set smaller to the randomly initialized weights (10^{-4} and 10^{-3} in our case). The margin for the triplet loss is set to 1, and weights for $[\alpha_1, \alpha_2]$ handling imbalanced classes in the weighted cross entropy is set as $[0.1, 1.0]$.

TABLE I
TUM DATASETS USED FOR TRAINING

Dataset	# Frame Pairs
fr2_xyz	3611
fr2_rpy	3217
fr2_desk	2219
fr2_360_hemisphere	2643
fr2_360_kidnap	1409
fr2_large_with_loop	1223
fr2_pioneer_360	826
fr2_pioneer_slam	2168
Total	17316

V. IMPLEMENTATION

Training and Testing Settings: To properly validate the generalization ability, we used sequences from different categories in the TUM benchmark for training, $fr2$, and testing, $fr1/3$. This ensures that there is no overlap between the training and testing sequences. The list of the selected datasets are shown in Table I. They cover typical indoor scenes with semantic objects, e.g., desk, computer and chairs, etc. We subsample the sequences to use only every fourth image, to provide harder samples (images further apart) for training.

To generate the training data, we use high gradient points ranked by the Harris algorithm. To ensure a distribution of keypoints over the entire image and to allow highly textured regions to generate many keypoints we run the algorithm twice. First on the whole image and then on each sub-image when dividing the images into a 4×4 grid. Due to naturally occurring noise,[3] misalignments exist in the image planes after warping keypoints using the ground truth trajectory. To compensate for

[3]e.g., noise in depth measurements, errors in the ground truth and imperfect camera calibration.

this, the KNN matching criteria is relaxed to prevent divergence in the training. For negative sample mining, only hardest samples with coordinate distances to target correspondence more than 5 pixels are used.

SLAM system: After finding geometric correspondences with GCN, a lightweight, keyframe-based, mapping system is used to compare with state-of-the-art SLAM systems. This pre-integrated system includes basic BA, pose graph optimization and vocabulary based loop closure detection. The poses of keyframes are continuously optimized during tracking as follows:

$$E_{\text{graph}} = \sum_{<i,j>\in C} (\boldsymbol{\xi}_{i,j} \boxplus \boldsymbol{\xi}_j^{-1} \boxplus \boldsymbol{\xi}_i)^T \boldsymbol{\Omega}_{i,j} (\boldsymbol{\xi}_{i,j} \boxplus \boldsymbol{\xi}_j^{-1} \boxplus \boldsymbol{\xi}_i) \tag{10}$$

where C is the edge set of relative poses collected in the map. $\boxplus$ is the generalized sum operation on the Lie group manifold. $\boldsymbol{\Omega}$ is information matrix proportional to the uncertainty of parameters. The loop closure detection is performed by the bag-of-word (BoW) library [5] for querying image with similarity. It is an appearance based retrieval method that converts features to a BoW vector and match using a hierarchical tree structure.

Development Environment: The test platform is equipped with a GTX 1080 graphic card and i7-4790 CPU. The overall training is implemented with the deep learning framework Pytorch. The BA and pose graph optimization is implemented with g2o [8] and the OpenCV libraries, where a GPU version of KNN is used for feature matching.

VI. Experiments

In this section, we evaluate the effectiveness of our framework with the TUM RGB-D [1] and KITTI Raw benchmarks [44]. TUM dataset provides synchronized ground truth trajectory for an RGB-D camera in common office scenes. The KITTI Raw datasets include recorded data from calibrated stereo cameras, 3D GPS/IMU and aligned Velodyne point clouds in typical outdoor scenes. We first compared GCN with ORB, SIFT and SURF as keypoints for frame to frame motion estimation using the indoor TUM dataset. The Absolute Trajectory Error (ATE) is used as the metric. Then, we compare GCN with UCN [38] and a patch similarity based method [39] for the outdoor KITTI dataset. The average angular and translational deviations are used as metric as in [38]. Finally, we evaluate our lightweight SLAM system based on GCN against three state-of-the-art RGB-D SLAM systems: ORB-SLAM2, Elastic Fusion and RGBDTAM.

The first experiment will demonstrate the quality of the features for motion estimation compared to other features using the same frame to frame motion estimation pipeline. The second test will demonstrate the generalization ability of GCN in unseen scenarios and performance compared to other deep learning methods. The last test will provide a comparison against existing, and end-to-end solutions for RGB-D based motion estimation to allow for a quantitative assessment of the performance on this task.

TABLE II
ATE Using Frame to Frame Tracking

Dataset (200 Frames)	GCN	ORB	SIFT	SURF
fr1_floor	**0.015 m**	0.080 m	0.073 m	0.074 m
fr1_desk	**0.037 m**	0.151 m	0.144 m	0.148 m
fr1_360	**0.059 m**	0.278 m	0.305 m	0.279 m
fr3_long_office	**0.061 m**	0.090 m	0.076 m	0.070 m
fr3_large_cabinet	**0.073 m**	0.097 m	0.091 m	0.143 m
fr3_nst	**0.020 m**	0.061 m	0.036 m	0.030 m
fr3_nnf	**0.221 m**	-	-	-

TABLE III
KITTI Raw Datasets for Training

Scene category	City	Road	Residential
Training sequences	**1, 2, 5, 9, 11, 13, 14,** 27, 28, 29, 48, 51, 56, 57, 59, 84	**15, 32**	**19, 20, 22, 23, 35,** 36, 39, 46, 61, 64, 79
Testing sequences	84, 91	52, 70	79, 86, 87

A. Evaluation of Motion Estimation Without Loop Closing

TUM Datasets: We compare the performance of GCN, ORB [4], SIFT [46] and SURF [47] on frame-by-frame tracking. During the test, features of different methods are extracted separately and fed into the same BA frontend for motion estimation. This frontend performs KNN matching for the resultant features and PnP with RANSAC to estimate camera motion. Reciprocal verification is performed during KNN matching. The number of features and the parameters used in PnP with RANSAC are set the same for fair comparison. Thus, the only variable in this sequential motion estimate setting is the feature used.

Table II shows the ATE for frame-to-frame tracking using the first 200 frames from each sequence. We use 200 frames from each sequence to show how ATE accumulated in different scenes. For all four methods, the number of keypoints to extract are set to 2000. Compared with other methods, GCN achieves the best performance on all sequences. Especially in scene fr3_nnf, a wall with weak texture, where GCN is still able to produce trackable features while others lose track.

KITTI Raw Datasets: We evaluated GCN on outdoor environment scenes to directly compare with related deep learning methods: dense feature based UCN and patch similarity based [39]. In addition, we also listed results of SIFT, SURF, DAISY [48] and KAZE [49] as baseline hand crafted features in this test. Two versions of GCN are prepared. The first is the model from before, trained on TUM sequences as in Table I and a second fine-tuned by KITTI. We denote them as GCN-TUM and GCN-KITTI to distinguish these two versions, where only GCN refers to GCN-TUM in other parts of the paper. The training/testing sets used in UCN are shown in Table III. The bold numbers are the sequence we used for training GCN-KITTI (UCN used all). The number of training sequences is less than half of what UCN used, to further demonstrate our generalization ability. The overall testing results are shown in Table IV. Both GCN-TUM and GCN-KITTI achieves better results compared to both hand crafted features and deep learning

TABLE IV
AVERAGE ANGULAR AND TRANSLATIONAL DEVIATIONS ON KITTI RAW DATASETS

Features	GCN-TUM	GCN-KITTI	UCN-HN	UCN-HN-ST	[39]	SIFT	DAISY	SURF	KAZE
Angular Deviation (degree)	**0.274**	**0.273**	0.317	0.325	0.394	0.307	0.309	0.344	0.312
Translational Deviation (degree)	**3.471**	**3.694**	4.147	4.728	9.293	4.749	4.516	5.790	4.584

TABLE V
ATE USING CLOSE LOOP SYSTEM

Dataset	Frames#	GCN	ORB-SLAM2	Elastic Fusion	RGBDTAM
fr1_floor	1227	0.038 m	**0.036 m**	-	-
fr1_desk	573	0.029 m	**0.016 m**	0.020 m	0.027 m
fr1_360	744	**0.069 m**	0.213 m	0.108 m	0.101 m
fr3_long_office	2488	0.040 m	**0.010 m**	0.017 m	0.027 m
fr3_large_cabinet	984	0.097 m	-	0.099 m	**0.070 m**
fr3_nst	1639	0.020 m	0.019 m	0.016 m	**0.010 m**
fr3_nnf	455	**0.064 m**	-	-	-

based methods: UCN and [39]. Note that GCN-TUM achieves the best results among all evaluated methods even if the testing data is visually significantly different from what it was trained on. That GCN-KITTI achieves slightly worse results than GCN-TUM might be the result of being trained with a smaller subset than what was deemed needed for UCN.

B. Evaluation of Closed Loop System

We compare our lightweight SLAM system using GCN features with open source SLAM frameworks; ORB-SLAM2, Elastic Fusion and RGBDTAM. These three represent state-of-the-art methods in sparse and dense geometric reconstruction. Note that these SLAM systems are much more comprehensive than our naive SLAM system.

ATE using all frames from the datasets are shown in Table V and two examples of raw point cloud reconstructions are shown in Fig. 5. Our method achieves competitive results compared with the other methods. We see that even with the longer sequences our method does not lose track. In sequence fr1_floor, Elastic Fusion lost track at places where there are missing frames, causing large displacements. This scene is also challenging for an ICP based method, such as Elastic Fusion, because most of the existing structures are flat. Since there are rich textures on the floor, both our method and ORB-SLAM2 works well and obtained similar results. Then, in fr1_360, the ORB feature didn't work as well as GCN, it results in an observable deterioration in the ATE of ORB-SLAM2. Similarly, in fr3_large_cabinet, ORB-SLAM2 lost track when the camera comes to the back of the cabinet with a rotation. The number of keypoints dropped sharply when the camera looked at the white back of the cabinet. In fr3_nnf, as shown in Section VI-A, the classical features cannot cope with this weak textured wall. Thus, ORB-SLAM2, Elastic Fusion and RGBDTAM failed to start the tracking at beginning.

VII. SUMMARY AND CONCLUSIONS

In this paper, we presented a unified framework for learning both detection of keypoints and descriptors for these. The framework is optimized for the task of camera motion estimation. We showed how we can generate training data by using data from SLAM benchmark datasets for which the ground truth of the camera is provided. We demonstrated how the resulting features outperform currently used features for motion estimation without loop closure and that we, even with a minimalistic implementation of loop closure optimization, achieve results that are on par with state-of-the-art methods such as ORB-SLAM2.

Our target application is a fully autonomous flying robot. In the future, we plan to improve the efficiency of GCN in terms of inference and matching. For the inference, network depth and storage consumption can be greatly reduced as shown in Deep Compression [50] and SqueezeNet [51]. Alternatively, a shallower backbone network can be used at the cost of a loss in accuracy. For matching, feature like ORB can be matched very efficiently since the descriptor is binarized. Work in DeepBit [52] shows that a CNN is also capable of producing representative binary features. With these possible improvements, GCN has potential to be well exploited in our future real-time application.

REFERENCES

[1] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D slam systems," in *Proc. 2012 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 573–580.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "ORB-slam: A versatile and accurate monocular slam system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[3] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[4] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[5] D. Galvez-Lpez and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

[6] A. I. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3d visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 40–45.

[7] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for RGB-D cameras," in *Proc. 2013 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 2100–2106.

[8] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.

[9] R. A. Newcombe *et al.*, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proc. 2011 10th IEEE Int. Symp. Mixed Augmented Reality*, 2011, pp. 127–136.

[10] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D slam with volumetric fusion," *Int. J. Robot. Res.*, vol. 34, no. 4–5, pp. 598–626, 2015.

[11] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, “Elasticfusion: Real-time dense slam and light source estimation,” *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016.

[12] A. Concha and J. Civera, “RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system,” *IEEE Int. Conf. Intel. Robot. Systems*, pp. 6756–6763, 2017.

[13] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “SVO: Semidirect visual odometry for monocular and multicamera systems,” *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[14] J. Engel, T. Schöps, and D. Cremers, “LSD-slam: Large-scale direct monocular slam,” in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.

[15] S. Zagoruyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 4353–4361.

[16] J. Žbontar and Y. LeCun, “Stereo matching by training a convolutional neural network to compare image patches,” *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, 2016.

[17] J. Zbontar and Y. LeCun, “Computing the stereo matching cost with a convolutional neural network,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1592–1599.

[18] W. Luo, A. G. Schwing, and R. Urtasun, “Efficient deep learning for stereo matching,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 5695–5703.

[19] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, “Deepflow: Large displacement optical flow with deep matching,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1385–1392.

[20] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, “Deepmatching: Hierarchical deformable dense matching,” *Int. J. Comput. Vis.*, 2016.

[21] L. Bao, Q. Yang, and H. Jin, “Fast edge-preserving patchmatch for large displacement optical flow,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 3534–3541.

[22] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, “Epicflow: Edge-preserving interpolation of correspondences for optical flow,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1164–1172.

[23] A. Dosovitskiy *et al.*, “Flownet: Learning optical flow with convolutional networks,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.

[24] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 1647–1655.

[25] N. Mayer *et al.*, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4040–4048.

[26] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in *Proc. Int. Conf. 3D Vis.*, 2016, pp. 239–248.

[27] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2650–2658.

[28] K. Tateno, F. Tombari, and I. Laina, and N. Navab, “Real-time dense monocular slam with learned depth prediction,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 6565–6574.

[29] B. Ummenhofer *et al.*, “Demon: Depth and motion network for learning monocular stereo,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 5622–5631.

[30] N. Sünderhauf *et al.*, “Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free,” in *Proc. Robot., Sci. Syst.*, 2015.

[31] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays, “Learning deep representations for ground-to-aerial geolocalization,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 5007–5015.

[32] D.-K. Kim and M. R. Walter, “Satellite image-based localization via learned embeddings,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Singapore, 2017, pp. 2073–2080.

[33] M. Lopez-Antequera, R. Gomez-Ojeda, N. Petkov, and J. Gonzalez-Jimenez, “Appearance-invariant place recognition by discriminatively training a convolutional neural network,” *Pattern Recog. Lett.*, vol. 92, pp. 89–95, 2017.

[34] H. Altwaijry, A. Veit, and S. Belongie, “Learning to detect and match keypoints with deep architectures,” in *Proc. Brit. Mach. Vis.*, 2016, pp. 49.1–49.12.

[35] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, “Lift: Learned invariant feature transform,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 467–483.

[36] V. Kumar B G, G. Carneiro, and I. Reid, “Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2016, pp. 5385–5394.

[37] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, “Learning local feature descriptors with triplets and shallow convolutional neural networks,” in *Proc. Brit. Mach. Vis.*, 2016, pp. 119.1–119.11.

[38] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, “Universal correspondence network,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2414–2422.

[39] P. Agrawal, J. Carreira, and J. Malik, “Learning to see by moving,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 37–45.

[40] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[41] O. Russakovsky *et al.*, “Imagenet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, pp. 211–252, 2015.

[42] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proc. IEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3431–3440.

[43] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 815–823.

[44] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, pp. 1229–1235, 2013.

[45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv:1412.6980, 2014.

[46] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

[47] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 404–417.

[48] E. Tola, V. Lepetit, and P. Fua, “DAISY: An efficient dense descriptor applied to wide baseline stereo,” *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 5, pp. 815–830, May 2010.

[49] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, “Kaze features,” in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 214–227.

[50] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” in *Proc. Int. Conf. Learn. Represent.*, 2016.

[51] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size,” arXiv:1602.07360, 2016.

[52] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, “Learning compact binary descriptors with unsupervised deep neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 1183–1192.