SQUEEZEPOSENET: IMAGE BASED POSE REGRESSION WITH SMALL CONVOLUTIONAL NEURAL NETWORKS FOR REAL TIME UAS NAVIGATION

M. S. Müller*, S. Urban, B. Jutzi

Institute of Photogrammetry and Remote Sensing, Karlsruhe Institute of Technology (KIT) - Karlsruhe, Germany
{markus.mueller5, steffen.urban, boris.jutzi}@kit.edu

Commission I/II, ICWG I/II

KEY WORDS: Pose Estimation, Navigation, Convolutional Neural Networks, Image-Based, UAV, UAS

ABSTRACT:

The number of unmanned aerial vehicles (UAVs) is increasing since low-cost airborne systems are available for a wide range of users. The outdoor navigation of such vehicles is mostly based on global navigation satellite system (GNSS) methods to gain the vehicles trajectory. The drawback of satellite-based navigation are failures caused by occlusions and multi-path interferences. Beside this, local image-based solutions like Simultaneous Localization and Mapping (SLAM) and Visual Odometry (VO) can e.g. be used to support the GNSS solution by closing trajectory gaps but are computationally expensive. However, if the trajectory estimation is interrupted or not available a re-localization is mandatory. In this paper we will provide a novel method for a GNSS-free and fast image-based pose regression in a known area by utilizing a small convolutional neural network (CNN). With on-board processing in mind, we employ a lightweight CNN called SqueezeNet and use transfer learning to adapt the network to pose regression. Our experiments show promising results for GNSS-free and fast localization.

1. INTRODUCTION

The last years show an increasing number of UAVs operating in industrial, military and private areas. Hence, research in the fields of navigation, data acquisition, path planning or obstacle avoidance for UAVs is increasing. Most of these systems need reliable navigation solutions which are mostly based on GNSS methods and often combined with alternative methods such as inertial navigation systems (INS). As failures due to gaps in signal coverage caused by occlusions or multipath effects weaken satellite-based navigation, alternative methods for a reliable navigation of unmanned aerial systems (UAS) are necessary.

Alternative methods for navigating in a global coordinate system, are usually based on digital surface models, digital terrain models or CAD models (Armenakis, 2015). However, the large memory consumption of such models renders this approach unfeasible for on-board processing on computational weak computers which are currently mounted on UAVs.

Further methods are aerial image matching using feature detectors, correlations or neural networks. Feature detection - like the well known Scale Invariant Feature Transform (SIFT, Lowe 1999) - or correlation based methods are computational intensive, especially for navigating in a huge area where potentially millions of descriptors have to be stored and matched. Thus, losing real time capability and making them inefficient for real time navigation. Convolutional neural networks (CNNs), on the other hand, can perform fast forward passes through the network even on small on-board GPUs and have a limited memory demand and power consumption (Nvidia, 2017).

The use of image-based local navigation methods like visual Simultaneous Localization and Mapping (SLAM) or Visual Odometry (VO) are potential solutions for reconstructing trajectories (Engel et al., 2014; Mur-Artal et al., 2015; Engel et al., 2016). However, they lack navigation in a global coordinate system without providing additional information, like relevant geo-referenced key points or initialization to a referenced model. Especially for short frame-to-frame distances these methods have a high accuracy but are subject to drift for longer distances. Nevertheless, these local solutions can support global navigation algorithms.

*Corresponding author.

Visual SLAM or VO fails once the pose estimate is lost which can be caused by fast motions or occlusions. Global re-localization methods could be used to recover the pose. However, global navigation methods are capable as standalone systems. With fast real time capability, global solutions recover a dense trajectory independent on any local navigation method. While most of the SLAM and VO solutions, as well as global navigation methods work fine on powerful computers, running them on weak or embedded devices may lead to limited real time capability and inadequately results. Considering running a navigation framework on small UAVs, respectively Micro Aerial Vehicles (MAVs), the computation power is limited due to maximum payload. Therefore all developed methods are subject to the limited computational power of the on-board processing units and need to be carefully designed with the target hardware in mind.

A solution may be provided by a CNN for local pose estimation. CNNs are also capable of ego-motion estimation (Konda and Memisevic, 2015). However, the results need to be improved to compete with conventional methods. Relative pose estimation of oblique images works successfully using CNN (Melekhov et al., 2017). They solve image matching tasks, where methods like Oriented FAST and rotated BRIEF (ORB) (Rublee et al., 2011) or Speeded Up Robust Features (SURF) (Bay et al., 2006) fail due to dissimilarity of descriptors. However a satisfying solution for VO using CNNs trained end-to-end is not yet available.

Convolutional neural networks offer a valuable solution to run on on-board computers, as the main computational effort of learning the network is performed offline on a powerful computer. Subsequently the on-board computer only has to process captured images by performing a forward pass through the network using the pre-trained weights. However, considering very deep CNNs like the VGG16-Net (Simonyan and Zisserman, 2014) which have a high number of parameters, the storage (some hundreds of megabytes) and processing requirements tend to exceed on-board capabilities. Therefore they are cumbersome to process on weak computing units. In contrast to this, a small CNN which is efficient in terms of processing while maintaining a satisfying accuracy and generalization is desirable. For this purpose we introduce a CNN-based solution for the navigation or localization of a UAV in a known area by using a variant of SqueezeNet (Iandola et al., 2016), which is originally a CNN for classification and achieves AlexNet Krizhevsky et al. (2012) accuracy with 50x less parameters. We modify the CNN to solve for pose regression. We call the modification SqueezePoseNet, since it is adapted from SqueezeNet but solves for poses.

The advantages of such a method for aerial vehicle navigation is the GNSS-free localization allowing to maneuver in urban areas. Further, it gives the opportunity to resolve the trajectory loss of local navigation methods or the loss of GNSS signals, maintaining a global solution. With CNN methods we are able to build a navigation framework processable on simple computational devices. Additionally, small CNNs need less computations and thus have a lower power consumption. This leads to further energy savings, increasing the flight time of the UAS. As the proposed method will output a coarse pose, it is reasonable to expand the work flow with a pose refinement step using the initially estimated pose. It is an advantage to decrease the search space for fine registration by exploiting the spatial constraints given by the coarse pose estimation. Therefore, solutions for image matching are given by conventional matching algorithms based on SIFT, SURF, ORB or using CNN matching techniques like MatchNet (Han et al., 2015).

Providing reliable navigation solutions sustains widespread reconstruction methods utilized on UAVs. Equipped with mapping sensors like laser scanner, UAS are able to map environments and generate dense point clouds (Droeschel et al., 2013; Jutzi et al., 2014).Therefore, a relative calibration between a navigation camera and a mapping sensor is necessary (Hillemann and Jutzi, 2017). Analysis of point clouds are already established and still under development (Weinmann et al., 2017).

After reviewing the related work in Section 2 we focus on image-based navigation solutions and introduce our method for determining poses in Section 3 and the training process in Section 4. In Section 5 we present the data utilized for this work. Hardware as well as the system used for capturing data is described in Section 6. Experiments are described in Section 7 and the results are discussed subsequently in Section 8. The final section gives an outlook and provides ideas for future work and research.

2. RELATED WORK

The field of GNSS-free navigation is of high interest. There are several image-based approaches available to determine camera poses from images in a global coordinate frame.

Solutions are provided by finding correlations between aerial and UAV images for image matching and further localization of the vehicle (Conte and Doherty, 2009, 2011). Feature-based methods match remotely sensed data (Li et al., 2009) or oblique images (Huachao et al., 2012). Model-based approaches to determine camera poses from imagery are also available (Reitmayr and Drummond, 2006; Unger et al., 2016). Determining the camera pose in real time, in indoor environments, is practicable by CAD model matching (Ulrich et al., 2009; Zang and Hashimoto, 2011; Urban et al., 2013; Mueller and Voegtle, 2016). Convolutional neural networks are used to determine matches between aerial images and UAV images (Altwaijry et al., 2016) or terrestrial images and UAV images (Lin et al., 2015). However these methods are not operable on small drones in large environments, due to the on-board computers limited storage.

As mentioned, feasible methods to conduct local navigation are visual SLAM or VO approaches, reconstructing a trajectory based on image sequences. Efficient solutions are ORB-SLAM (Mur-Artal et al., 2015), Large-Scale Direct Monocular SLAM (LSD-SLAM) (Engel et al., 2014) or Direct Sparse Odometry (DSO) (Engel et al., 2016). These methods provide satisfying solutions according to accuracy and real time capability. However, their trajectory will only have a local relation. Navigating in a higher level coordinate frame can be achieved by fusing local solutions with for instance GNSS or other global localization methods. Even though SLAM or VO solutions show impressive results (e.g. DSO) and do drift only slightly for short distances, they will drift over long trajectories particularly if there are no loop closures. Additionally, SLAM methods or VO fail if the track is lost. Restoring the lost track is impossible without moving back to a known or mapped position. However local and global navigation build a complementary framework, merging the advantages of both systems. The satisfying accuracy of relative pose estimations derived from neighboring frames and the computational speed of local methods as well as the global referencing to counter drift and track losses build a well suited framework.

This work is mainly based on PoseNet (Kendall et al., 2015), a convolutional neural network which re-localizes an acquired image in a known area. For this purpose the CNN is trained with images and their corresponding poses in order to estimate the pose of an unknown, nearby position. The CNN is not only able to estimate poses between two trained images by regression, but also to extent the learned information to determine poses which slightly exceed the learned space. However, we will see that these results suffer a little concerning spatial accuracy. An enhancement is the Bayesian PoseNet (Kendall and Cipolla, 2016) which provides re-localization uncertainty by adding dropout layers after each convolutional layer and improves the PoseNet accuracy by averaging over multiple forward passes. For large-scale data sets the accuracy is approximately 2m and 6°. PoseNet is originally build on GoogLeNet (Szegedy et al., 2015), a CNN created for classification tasks, as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). However, a CNN designed for classification requires some changes to regress poses from images. The major adaptation to achieve this is to replace GoogLeNets classifiers with regressors by changing the loss function. In addition, the number of output neurons of the last fully connected layer is changed from 1000 to 7 to estimate camera poses. In fact GoogLeNet was designed to classify 1000 different classes whereas the camera pose is parametrized by position (3 parameters) and orientation represented as quaternions (4 parameters) and thus requires 7 parameters in total. However, Kendall et al. (2015) show the feasibility of their development. Enhanced accuracies in the task of estimating poses were derived by further improvement (Walch et al., 2016a) using Long Short-Term Memory layers (LSTM, Hochreiter and Schmidhuber 1997), a type of recurrent neural net which was combined with CNNs in the past. LSTM handle the problem of a dissolving gradient during the back-propagation using so called gates. However, they showed great success in handwriting or speech recognition and in this case, for image re-localization. Furthermore, to improve a potential navigation approach, one may support the CNNs input with additional data. Combining RGB data and depth data in a dual stream CNN showed further improvements of the localization results (Li et al., 2017).

One of our aims is to develop an image-based navigation framework operational on an on-board computer, while maintaining satisfying accuracies. Considering, that very deep CNNs with a large number of parameters need a higher computation power than smaller CNNs, the latter is preferred. For example, the VGG16-Net (Simonyan and Zisserman, 2014) has 528 MB of weights which will exceed the capacity of an on-board processing unit like DJI Manifold which we use for our development (Section 6.2). The model size of PoseNet (Kendall et al., 2015) or the LSTM-based CNN (Walch et al., 2016b) which are based on GoogLeNet have sizes of about 50 MB. However, these size varies slightly due to the modification of some layers, and therefore the number weighting parameters, of the basic net. We need a CNN that is small enough to run on an on-board processing unit while maintaining sufficient accuracy. Therefore, we introduce our approach on adapting SqueezeNet (Iandola et al., 2016), a smaller CNN with a model size of only 4,8 MB which is even 10 times smaller than GoogLeNet. In future work this might be reduced even more by quantizing (Wu et al., 2016) or binarization (Courbariaux et al., 2016).

3. METHODOLOGY

For our demands on pose regression we adapt SqueezeNet, a small convolutional neural network, which is developed for solving classification tasks (Iandola et al., 2016). Figure 1 depicts the architecture of the adapted SqueezeNet. We call it SqueezePoseNet since it is modified to solve for pose regression. SqueezeNets originally design strategy lies in the Fire modules. Each Fire module first decreases the number of input channels from the previous layer by 1x1 convolutions in a so called squeeze operation. Subsequently an expand operation that is a combination of 1x1 and 3x3 filters increases the number of activation maps but keeps the number of parameters low (Figure 2). In addition the traditional SqueezeNet architecture lacks a final fully connected layer as these layers increase the number of parameters significantly. Instead, the final convolutional layer consists of as many 1x1 filters as classes. Subsequently average pooling is used to yield a vector whose length equals the number of classes. To modify SqueezeNets architecture for pose regression, we first introduce a fully connected layer with 500 neurons. This layer acts as a descriptor and is meant to help the network distinguish different poses from each other. Additionally, we change the layers activation functions from rectified linear units (ReLU) to leaky rectified linear units (Leaky ReLU) Maas et al. (2013). This is meant to help convergence. In addition, we add Batch Normalization Ioffe and Szegedy (2015) after each convolutional layer, making higher learning rates possible. Finally, we add two fully connected layers for actual pose estimation. A three neuron layer for position ('$\mathbf{x}$') and a four neuron layer for rotation ('$\mathbf{q}$'). The rotation is parametrized as a quaternion. The training is described in Section 4.

For evaluation purpose we also modify a deeper neural network, the VGG16-Net (Simonyan and Zisserman, 2014) to compare it with the smaller SqueezePoseNet. Deeper networks obviously tend to be more accurate than small ones (Iandola et al., 2016). As the modification on SqeezeNet, two dense layers ('x' and 'q') for pose regression were added and the layers activation functions were set to Leaky Rectified Linear Units.

CNNs are optimized by iteratively changing their weighting parameters using back-propagation. To optimize SqueezePoseNet for pose regression we minimize the following loss (Kendall et al., 2015):

$$\text{Loss}_i = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2 + \beta \, \|\mathbf{q}_i - \frac{\hat{\mathbf{q}_i}}{\|\hat{\mathbf{q}}_i\|}\|_2$$

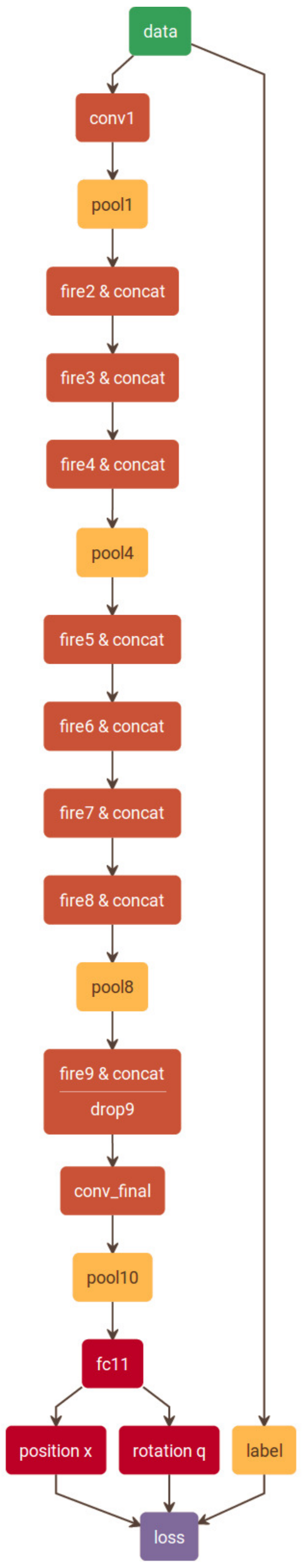

Figure 1. SqueezePoseNet. We add a fully connected layer ('fc11') with 500 neurons. For pose estimation we added two dense layers ('$\mathbf{x}$' and '$\mathbf{q}$') with 3 respectively 4 neurons. All weighting parameters up to the 'pool10'-layer are initialized from a pre-trained version of SqueezeNet that was trained on a subset of the *Places365* data set (Zhou et al., 2016). The additional layers are pre-trained using the *Shop Façade* data set and later on our test set of the *Atrium*.

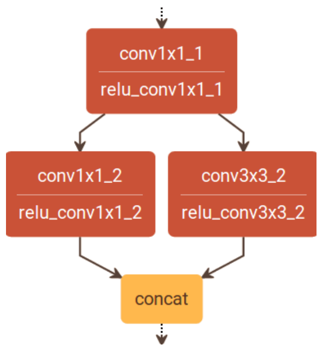

Figure 2. Architecture of a Fire module. A so called squeeze operation is performed by the 1x1 convolutional layer. Subsequently an expand operation that is a combination of 1x1 and 3x3 filters increases the number of activation maps while keeping the number of parameters low.

The loss is calculated as the sum of the position error and the weighted orientation error. $\hat{\mathbf{x}}_i$ and $\mathbf{x}_i$ are ground truth and estimated position in metric dimensions. $\hat{\mathbf{q}}_i$ and $\mathbf{q}_i$ are ground truth and estimated orientation in quaternions. Since position and orientation are not in the same unit space and therefore are not comparable, a weighting is done by using β to keep the errors in the same range. This prevents the CNN to minimize only one of the two error values. Typically, beta is set between 250 and 2000 for outdoor data sets. We set β to 500 for the experiments on our own data in Section 7.

4. TRAINING

Convolutional neural networks usually need a huge amount of training data to work well, which is often not available. Transfer-learning is a valuable process to overcome this issue. For that, a net is initially pre-trained with a huge amount of data, which is often publicly available, to determine weights for the nets layers. These weights are used as initial values for later training.

For our purpose we initialize SqueezeNet with weighting parameters trained on the *ImageNet* data set (Krizhevsky et al., 2012). It was shown that for pose regression, training on the *Places* data set (Zhou et al., 2014, 2016) leads to further improvement in terms of accuracy (Kendall et al., 2015), as it is a more suitable data set for pose regression. Since that, training was subsequent carried out on the *Places* data set.

Before starting training on our own re-localization data set *Atrium* (Section 5.2), we chose to pre-train our CNN with a data set for pose regression, the *Shop Façade* benchmark data set (Section 5.1). However, we only set our added layers ('fc11', 'position x' and 'rotation q') as trainable, since the preceding layers should already be well initialized by the pre-training steps. After training our network on the *Shop Façade* benchmark we finally fine-tune the CNN on our training data of the *Atrium* data set consisting of 864 images. The inputs for our CNNs are 757 training images and 95 evaluation images with their corresponding camera poses. The training on SqueezePoseNet ended with errors of 4.45 m for position and 14.35° for orientation.

For accuracy evaluation of SqueezePoseNet, we chose the deeper VGG16-Net, which should tend to be more accurate. Considering VGG16-Net is built to solve classification tasks, we adapt it in a similar manner as the SqueezeNet described above to estimate poses. We initialize the nets layers with weights obtained by training on the *Places365* data set. We also train our adaptation on *Shop Façade* to get initial weights for our added layers. With that we train our modified net on the data set *Atrium*. The training errors result in 2.35 m and 9.09° for position respectively rotation.

Summarizing, the work flow of our methodology can be described as follows. (i) Choose a suitable CNN for classification tasks. (ii) Initialize the CNN with pre-trained weighting parameters. (iii) Transfer learning on a suitable data set, e.g. *Places* (optional). (iv) Prepare the CNN to solve for pose regression by adding appropriate layers. (v) Produce training data for a test environment, e.g. by Agisoft PhotoScan, a Structure from Motion software (Agisoft, 2017). (vi) Train the network, whereas the trainable layers are restricted to the last fully connected layers. (vii) Train the CNN with own training data. (viii) Run the CNN on evaluation images.

The training procedure is carried out on a 64 GB RAM computer equipped with an Intel® Core™ i7-3820 3.6 GHz processor and an 8 GB GeForce GTX 980 graphics card.

5. DATA

In this section, we describe the data sets *Shop Façade* and *Atrium* that we used for training and evaluating the convolutional neural networks for pose regression. Besides, we also use the *ImageNet* and *Places* data sets for pre-training.

5.1 *Shop Façade* data set

The *Shop Façade* data set is part of the Cambridge Landmarks benchmark (Kendall et al., 2015). This outdoor data set for visual re-localization contains three video sequences recorded with a smartphone camera and their extracted image frames separated in 231 training and 103 evaluation images. A pose is provided for every image. We train our modified CNNs on *Shop Façade* to yield initial weighting parameters for further training on the *Atrium* data set. The data set containing image sequences and poses of *Shop Façade* is available online[1]. The dimension of the area in which the images were captured is denoted with 35 x 25 m^2.

5.2 *Atrium* data set

The *Atrium* data set consists of 864 images collected within the *LaFiDa*[2] benchmark collection by KIT (Urban and Jutzi, 2017). We use the high resolution images[3] to determine their poses with Agisofts Structure from Motion routine (Agisoft, 2017). The 3D model in top view is shown in Figure 3. The dimension of the captured area (39 x 36 x 18 m^3) is similar to the *Shop Façade* environment, at least for the ground area. However, we experienced higher training errors than on the *Shop Façade* data set, as the camera poses are sparser distributed and additionally extended in height (up to 18 m).

For evaluation of the CNNs, two sequences were captured in the Atrium with a X3 Zenmuse camera attached to the UAV. The *medium coverage sequence* (Section 7.1) captured on a low altitude is spatially close to the training data. The images of the *low coverage sequence* (Section 7.2) are spatially far away and have high discrepancies in perspectives compared to the training data. However, the *medium coverage sequence* includes 145 extracted images, which show a medium coverage to the training

[1]http://mi.eng.cam.ac.uk/ (last access 17th March 2017)
[2]https://www.ipf.kit.edu/lafida.php (last access 21st June 2017)
[3]https://www2.ipf.kit.edu/ pcv2016/downloads/photos_atrium_reconstruction.zip (last access 21st June 2017)

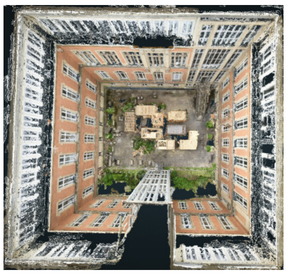
Figure 3. *Atrium* from above. Notice, the red bricked walls with the windows show very similar structures, which is a challenging environment for computer vision algorithms.

data shown in Figure 6.a. High coverage can be stated, if training and testing data show very similar poses. The *low coverage sequence* is captured with a higher altitude than the *medium coverage sequence* and shows a low coverage to the training data, as no images with high altitude or a downward facing field of view are present in our training images. Therefore, the positions as well as the orientations are 'far away' from our training poses. However, the *low coverage sequence* should be challenging for a convolutional neural network, as it has not learned similar images nor poses during the training process. Ground truth is built by adding the evaluation images to the SfM pipeline to determine their poses. For evaluation, the resulting poses are used to compute differences to the corresponding camera poses estimated by the CNN.

Figure 4 shows examples of training and testing images. Figure 4.a shows a test image where reflections could be detected. Figure 4.b shows a blurred image caused by angular motion of the UAV, both effects are known for challenging computer vision tasks. Figure 4.c and Figure 3 show the very ambiguous structures on the *Atriums* walls. Figure 4.d shows a test image of the *low coverage sequence*. A similar image is not contained in the training data. The position as well as the orientation are very far away from any training pose. All data for training and evaluation will be available online after the papers acceptance.

6. UNMANNED AERIAL SYSTEM

This Section presents an overview on our UAS, which mainly consist of a UAV (Section 6.1), an on-board processing unit (Section 6.2) and a camera (Section 6.3).

Figure 5. DJI Matrice 100 with on-board processing unit DJI Manifold and camera Zenmuse X3.

6.1 Unmanned Aerial Vehicle

A Matrice 100 from DJI (Figure 5) and the X3 camera (Section 6.3) serve to capture images for the experiments (Section 7). The on-board processing unit, a DJI Manifold (Section 6.2) is used for data storage. Besides, it is capable of running the proposed SqueezePoseNet. The quadrocopter is able to carry about 1 kg of payload with a maximum take off weight of 3600 g. The training of the CNNs is processed offline, merely the actual re-localization or navigation process has to be processed on-board.

6.2 On-board processing unit

The DJI Manifold is based on a NVIDIA® Tegra K1 with a Quad-core Arm® Cortex A-15 32-bit processor. The Kepler GPU consists of 192 CUDA® cores. CUDA® is NVIDIAs® software architecture to process computations on a graphics processing unit (GPU) which is necessary for most deep learning tasks. Furthermore, the Manifold is compatible with cuDNN®, NVIDIAs® GPU based deep neural network library, which includes standard routines for CNN-processing and CNMeM, a memory manager for CUDA. Besides, the DJI Manifold has 2 GB of RAM. With SqueezePoseNet, we are able to process our computations in real time. The weight of the device is less than 200 g.

6.3 Camera

The UAS is equipped with DJIs Zenmuse X3 camera. Its capable of taking single shot images with a resolution of 4000 x 2250 pixels or capturing video streams with a resolution of 3840 x 2160 pixels and 30 frames per second. The camera has a 1/2.3" CMOS sensor (6.2 x 4.6 mm) with 12.4 megapixel. Its field of view is 94°. The camera and its gimbal weigh 254 grams.

7. EXPERIMENTS

We evaluate SqueezePoseNet on the *medium* and *low coverage sequences* mentioned in Section 5.2. For comparison, we also evaluate the VGG16-Net adaptation and PoseNet (Kendall et al., 2015) on these two sequences. In Section 7.1 and 7.2 we show visual results as well as metric evaluations against ground truth.

7.1 *Medium coverage sequence*

This sequence shows a medium coverage of training and evaluation poses. A high coverage would be given by a more dense distribution of training data or a high similarity of training images and evaluation images. However, Figure 6 shows the training, ground truth and estimated camera poses for the *medium coverage sequence*. The resulting poses derived by the adaptation of VGG16-Net are visualized in Figure 6.b. The poses derived by SqueezePoseNet are plotted in Figure 6.c.

The histograms in Figure 7 show the derived errors to the ground truth poses. Figure 7.a and 7.b show the spatial errors of our VGG16-Net and SqueezePoseNet to ground truth. Figure 7.c and 7.d show the related angular errors in the same manner. The numerical results are represented by the difference between the spatial and angular error to the ground truth. The pose estimation accuracy is 4.91 m and 33.30° for the modified VGG16-Net, 5.19 m and 29.28° for SqueezePoseNet and 8.60 m and 50.83° for PoseNet (Table 1).

7.2 *Low coverage sequence*

This sequence shows a low coverage of training poses and evaluation poses. Figure 8 shows the training, ground truth and estimated camera poses for the *low coverage sequence*. The resulting poses derived by the modified VGG16-Net are visualized in Figure 8.b. The poses derived by SqueezePoseNet are plotted in Figure 8.c.

The histograms in Figure 9 show the derived errors to the ground truth poses. Figure 9.a and 9.b show the spatial errors of our VGG16-Net respectively SqueezePoseNet to ground truth. Figure 9.c and 9.d show the related angular errors in the same manner. The pose estimation accuracy is 11.34 m and 37.33° for our modified VGG16-Net, 15.18 m and 65.02° for SqueezePoseNet and 11.47 m and 46.40° for PoseNet (Table 1).

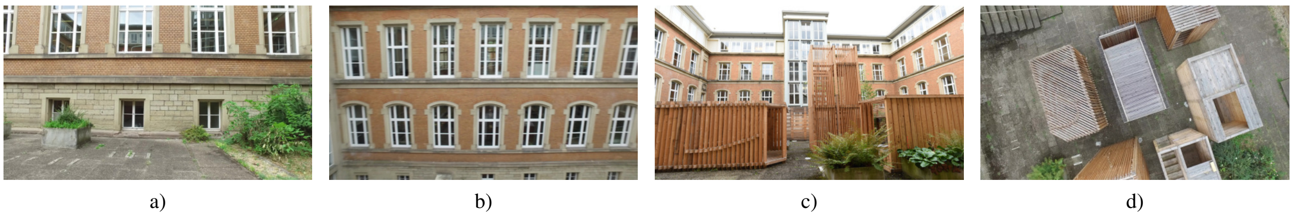

Figure 4. Image examples of the *Atrium*. a) Windows in the scene cause reflections, this is challenging for computer vision tasks, b) captured with high (angular) motion and therefore shows motion blur, c) shows ambiguous structures. d) is a test image of the *low coverage sequence* (Section 7.2) captured at high altitude. A similar perspective is not contained in the training data, the pose is 'far away' from any training pose (position as well as orientation).

Evalutaion Errors	PoesNet	VGG16-Net (modified)	SqueezePoseNet
Medium coverage sequence	8.60 m, 50.83°	**4.91 m**, 33.30°	5.19 m, **29.28°**
Low coverage sequence	11.47 m, 46.40°	**11.34 m**, **37.33°**	15.18 m, 65.02°

Table 1: Position and rotation errors on the *medium* respectively the *high coverage sequence*. Bold text marks best result on a sequence.

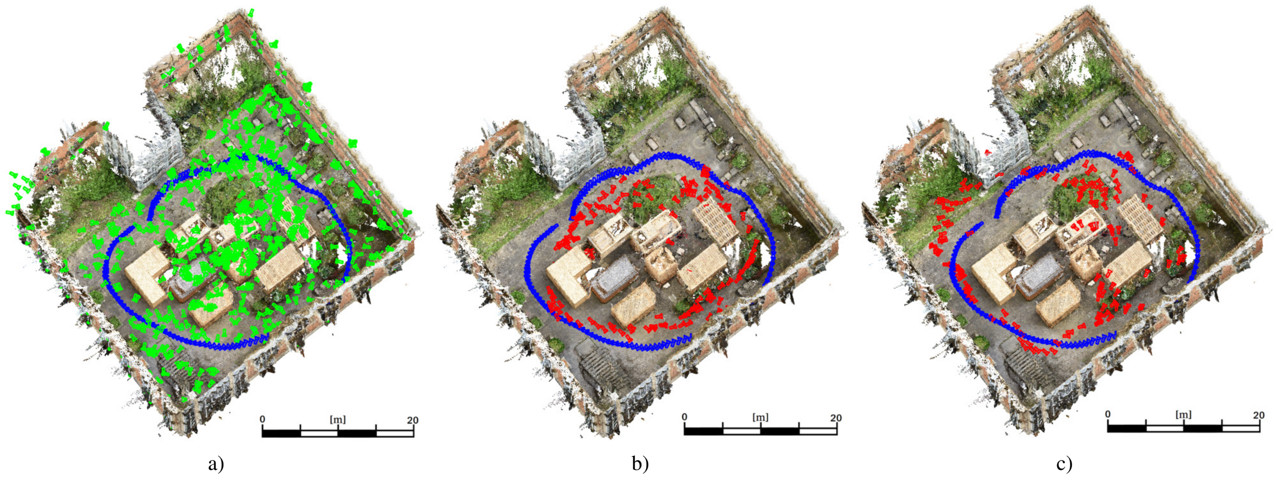

Figure 6. *Atrium* data set, *medium coverage sequence*. a) Visualization of ground truth, b) results derived by the CNN adaptation of VGG16-Net, c) results derived by SqueezePoseNet. Green cameras indicate training data, blue cameras depict the ground truth and red cameras are estimated poses derived by the modified CNNs.

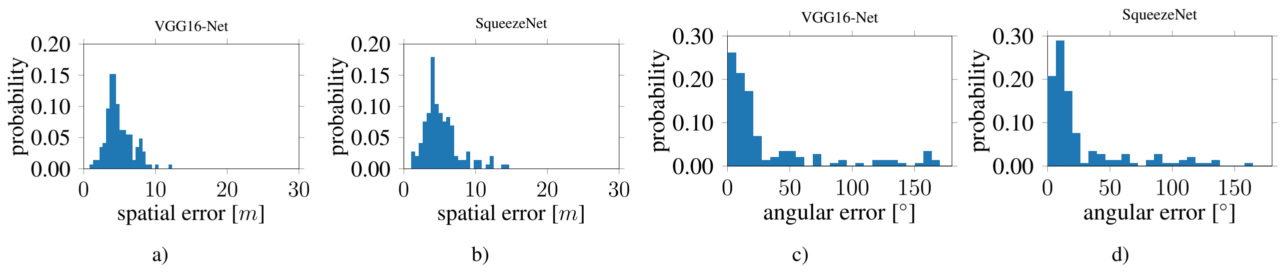

Figure 7. *Medium coverage sequence*. The histograms show the spatial and angular errors of the CNN derived poses to ground truth. Whereas a) and b) depict the spatial errors of our VGG16-Net respectively SqueezePoseNet and c) and d) show the angular errors.

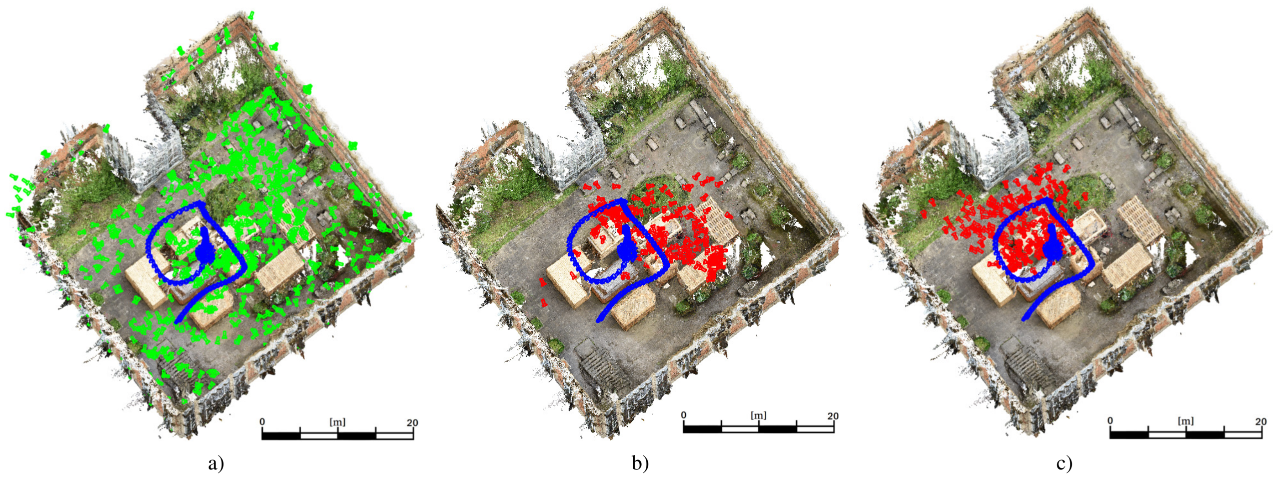

Figure 8. *Atrium* data set, *low coverage sequence*. a) Visualization of ground truth, b) results derived by the CNN adaptation of VGG16-Net, c) results derived by SqueezePoseNet. Green cameras indicate training data, blue cameras depict the ground truth and red cameras are estimated poses derived by the modified CNNs.

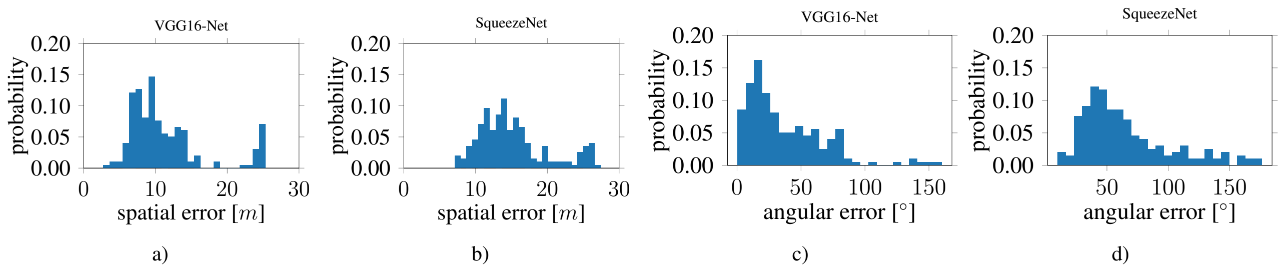

Figure 9. *Low coverage sequence*. The histograms show the spatial and angular errors of the CNN derived poses to ground truth. Whereas a) and b) depict the spatial errors of the VGG16-Net respectively SqueezePoseNet and c) and d) the angular errors.

8. DISCUSSION

The experiments show that convolutional neural networks are able to estimate camera poses. Furthermore SqueezePoseNet has a small net size and is therefore very well suited for UAS or MAV on-board applications.

We tested our approaches on the *medium* and the *low coverage sequence*. The *medium coverage sequence* show satisfying results for regressing a coarse pose. It works well for the VGG16-Net adaptation as well as for SqueezePoseNet. However, VGG16-Net shows comparable results, as its deeper than SqueezePoseNet. Both CNNs, with postition differences to ground truth of 4.91 m respectively 5.19 m, would provide good initial poses for a further and optional pose refinement step. Besides that, the calculation of these poses is processed within a few milliseconds.

As expected, the *low coverage sequence* where no similar training data was captured showed less accurate results. The spatial differences of 11.34 m and 15.18 m in an environment of about 39x36x18 m^3 should not be satisfying. We found, that the majority of the position error is contributed by a spurious height determination. The poses are systematically estimated too low. Its reasonable that the CNN can not extend its knowledge too far from the trained data, as it has never learned such extends. Even though interpolation works well, the extrapolation on the other hand does seem to perform less accurate. We are very interested to overcome this limitation in the future, seeing that it is unpractical to collect data of an environment with such a high density.

It is obvious that a high coverage of training data serves better to train a CNN than a low coverage. Obviously, one can achieve better results having a high similarity between training and testing images. Besides that, the pose error scales proportional to the environments scale. Therefore, indoor environments of small spatial dimensions, lead to better results than large areas. Beyond that, it has to be mentioned that different cameras are used to capture the evaluation images and training data of the *Atrium* data set, therefore the CNN does not know the camera with its intrinsic parameters, which may be a further reason of insufficient accuracies. However, this should only cause slight errors.

9. CONCLUSION AND OUTLOOK

We conclude that CNN based solutions for pose regression can satisfy the needs to improve UAV based navigation applications. It is possible to derive a coarse pose within milliseconds, e.g. to initialize a navigation framework or for a subsequent pose refinement. In combination with other navigation solutions like SLAM or VO, one may build a navigation framework suitable for navigating in urban areas and street canyons or even indoor environments.

A drawback of the proposed idea is the necessity of training data. Considering that a pose can only be determined if sufficient training data of the environment is available. This is often not given. However, image data is captured all over the globe commercial by companies or publicly available by private persons. A well known example is the reconstruction of parts of the city of rome by using solely online data (Agarwal et al., 2011). On the other hand, there is a lot of aerial imagery available covering urban and rural areas, which may serve as training data.

Of further interest is the amount of data content which can be learned by a CNN, especially by a small CNN. Even though the weighting parameters of the CNNs do not increase with a higher amount of training images, a CNN should only be able to recognize a finite part of the training data. Small CNNs like SqueezePoseNet may fail or at least lose accuracy with an increase of the spatial dimension of the training environment. We are looking forward to investigate that in future work.

References

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M. and Szeliski, R., 2011. Building rome in a day. 54(10), pp. 105–112.

Agisoft, 2017. Agisoft LLC, http://www.agisoft.com/, (last access 31th March 2017).

Altwaijry, H., Trulls, E., Hays, J., Fua, P. and Belongie, S., 2016. Learning to match aerial images with deep attentive architectures. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3539–3547.

Armenakis, C., 2015. (Editor). Proceedings of International Conference on Unmanned Aerial Vehicles in Geomatics (UAV-g). ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-1-W4, pp. 1–419.

Bay, H., Tuytelaars, T. and Van Gool, L., 2006. Surf: Speeded up robust features. In: European conference on computer vision, Springer, pp. 404–417.

Conte, G. and Doherty, P., 2009. Vision-based Unmanned Aerial Vehicle Navigation Using Geo-referenced Information. EURASIP J. Adv. Signal Process 2009, pp. 1–18.

Conte, G. and Doherty, P., 2011. A visual navigation system for uas based on geo-referenced imagery. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. and Bengio, Y., 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to + 1 or -1. arXiv:1602.02830.

Droeschel, D., Schreiber, M. and Behnke, S., 2013. Omnidirectional perception for lightweight UAVs using a continuously rotating 3d laser scanner. 40(1), pp. 107–112.

Engel, J., Koltun, V. and Cremers, D., 2016. Direct sparse odometry. arXiv:1607.02565.

Engel, J., Schöps, T. and Cremers, D., 2014. LSD-SLAM: Large-Scale Direct Monocular SLAM. In: D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars (eds), Computer Vision – ECCV 2014, Lecture Notes in Computer Science, Springer International Publishing, pp. 834–849. DOI: 10.1007/978-3-319-10605-2_54.

Han, X., Leung, T., Jia, Y., Sukthankar, R. and Berg, A. C., 2015. MatchNet: Unifying feature and metric learning for patch-based matching. pp. 3279–3286.

Hillemann, M. and Jutzi, B., 2017. UCalMiCeL - Unified intrinsic and extrinsic calibration of a multi-camera-system and a laserscanner. Proceedings of International Conference on Unmanned Aerial Vehicles in Geomatics (UAV-g). ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation 9(8), pp. 1735–1780.

Huachao, Y., Shubi, Z. and Yongbo, W., 2012. Robust and precise registration of oblique images based on scale-invariant feature transformation algorithm. IEEE Geoscience and Remote Sensing Letters 9(4), pp. 783–787.

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

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of The 32nd International Conference on Machine Learning, pp. 448–456.

Jutzi, B., Weinmann, M. and Meidow, J., 2014. Weighted data fusion for UAV-borne 3d mapping with camera and line laser scanner. International Journal of Image and Data Fusion 5(3), pp. 226–243.

Kendall, A. and Cipolla, R., 2016. Modelling uncertainty in deep learning for camera relocalization. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE, pp. 4762–4769.

Kendall, A., Grimes, M. and Cipolla, R., 2015. Posenet: A convolutional network for real-time 6-dof camera relocalization. In: Proceedings of the IEEE international conference on computer vision, pp. 2938–2946.

Konda, K. R. and Memisevic, R., 2015. Learning Visual Odometry with a Convolutional Network. In: VISAPP (1), pp. 486–490.

Krizhevsky, A., Sutskever, I. and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105.

Li, Q., Wang, G., Liu, J. and Chen, S., 2009. Robust scale-invariant feature matching for remote sensing image registration. 6(2), pp. 287–291.

Li, R., Liu, Q., Gui, J., Gu, D. and Hu, H., 2017. Indoor relocalization in challenging environments with dual-stream convolutional neural networks. PP(99), pp. 1–12.

Lin, T.-Y., Yin Cui, Belongie, S. and Hays, J., 2015. Learning deep representations for ground-to-aerial geolocalization. IEEE, pp. 5007–5015.

Lowe, D. G., 1999. Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on, Vol. 2, Ieee, pp. 1150–1157.

Maas, A. L., Hannun, A. Y. and Ng, A. Y., 2013. Rectifier nonlinearities improve neural network acoustic models. In: in ICML Workshop on Deep Learning for Audio, Speech and Language Processing, Citeseer.

Melekhov, I., Kannala, J. and Rahtu, E., 2017. Relative Camera Pose Estimation Using Convolutional Neural Networks. arXiv:1702.01381.

Mueller, M. and Voegtle, T., 2016. Determination of steering wheel angles during car alignment by image analysis methods. XLI-B5, pp. 77–83.

Mur-Artal, R., Montiel, J. M. M. and Tardós, J. D., 2015. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Transactions on Robotics 31(5), pp. 1147–1163.

Nvidia, 2017. http://www.nvidia.com/docs/io/90715/tegra_multi-processor_architecture_white_paper_final_v1.1.pdf, (last access 31th March 2017).

Reitmayr, G. and Drummond, T., 2006. Going out: Robust model-based tracking for outdoor augmented reality. In: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '06, IEEE Computer Society, pp. 109–118.

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G., 2011. ORB: An efficient alternative to SIFT or SURF. In: 2011 International Conference on Computer Vision, pp. 2564–2571.

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

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9.

Ulrich, M., Wiedemann, C. and Steger, C., 2009. Cad-based recognition of 3d objects in monocular images. In: ICRA, Vol. 9, pp. 1191–1198.

Unger, J., Rottensteiner, F. and Heipke, C., 2016. Integration of a generalised building model into the pose estimation of uas images. In: 23rd International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences Congress, ISPRS 2016, 12–19 July 2016, Prague, Czech Republic.

Urban, S. and Jutzi, B., 2017. LaFiDa - A Laserscanner Multi-Fisheye Camera Dataset. Journal of Imaging 3(1), pp. 5.

Urban, S., Leitloff, J., Wursthorn, S. and Hinz, S., 2013. Self-Localization of a Multi-Fisheye Camera Based Augmented Reality System in Textureless 3D Building Models. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science 2, pp. 43–48.

Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S. and Cremers, D., 2016a. Image-based localization with spatial lstms. arXiv:1611.07890.

Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S. and Cremers, D., 2016b. Image-based Localization with Spatial LSTMs. arXiv:1611.07890.

Weinmann, M., Mueller, M. S., Hillemann, M., Reydel, N., Hinz, S. and Jutzi, B., 2017. Point cloud analysis for UAV-borne laser scanning with horizontally and vertically oriented line scanners - concept and first results. Proceedings of International Conference on Unmanned Aerial Vehicles in Geomatics (UAV-g). ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.

Wu, J., Leng, C., Wang, Y., Hu, Q. and Cheng, J., 2016. Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4820–4828.

Zang, C. and Hashimoto, K., 2011. Camera localization by CAD model matching. In: 2011 IEEE/SICE International Symposium on System Integration (SII), pp. 30–35.

Zhou, B., Khosla, A., Lapedriza, A., Torralba, A. and Oliva, A., 2016. Places: An image database for deep scene understanding. arXiv:1610.02055.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A. and Oliva, A., 2014. Learning deep features for scene recognition using places database. In: Advances in neural information processing systems, pp. 487–495.