Image-based Localization using Hourglass Networks

Iaroslav Melekhov[1], Juha Ylioinas[1], Juho Kannala[1], and Esa Rahtu[2]

[1]Aalto University, Finland
firstname.lastname@aalto.fi
[2]Tampere University of Technology, Finland
esa.rahtu@tut.fi

Abstract

In this paper, we propose an encoder-decoder convolutional neural network (CNN) architecture for estimating camera pose (orientation and location) from a single RGB-image. The architecture has a hourglass shape consisting of a chain of convolution and up-convolution layers followed by a regression part. The up-convolution layers are introduced to preserve the fine-grained information of the input image. Following the common practice, we train our model in end-to-end manner utilizing transfer learning from large scale classification data. The experiments demonstrate the performance of the approach on data exhibiting different lighting conditions, reflections, and motion blur. The results indicate a clear improvement over the previous state-of-the-art even when compared to methods that utilize sequence of test frames instead of a single frame.

1. Introduction

Image-based localization, or camera relocalization refers to the problem of estimating camera pose (orientation and position) from visual data. It plays a key role in many computer vision applications, such as simultaneous localization and mapping (SLAM), structure from motion (SfM), autonomous robot navigation, and augmented and mixed reality. Currently, there are plenty of relocalization methods proposed in the literature. However, many of these approaches are based on finding matches between local features extracted from an input image (by usually applying local image descriptor methods such as SIFT, ORB, or SURF [18, 23, 2]) and features corresponding to 3D points in a model of the scene. In spite of their popularity, feature-based methods are not able to find matching points accurately in all scenarios. In particular, extremely large viewpoint changes, occlusions, repetitive structures and textureless scenes often produce simply too many outliers in the matching process. In order to cope with many outliers, the typical first aid is to apply RANSAC which unfortunately increases time and computational costs.

The increased computational power of graphic processing units (GPUs) and the availability of large-scale training datasets have made Convolutional Neural Networks (CNNs) the dominant paradigm in various computer vision problems, such as image retrieval [1, 8], object recognition, semantic segmentation, and image classification [17, 10]. For image-based localization, CNNs were considered for the first time by Kendall *et al*. [15]. Their method, named PoseNet, casts camera relocalization as a regression problem, where 6-DoF camera pose is directly predicted from a monocular image by leveraging transfer learning from a large scale classification data. Although PoseNet overcomes many limitations of the feature-based approaches, its localization performance still lacks behind traditional approaches in typical cases where local features perform well.

Looking for possible ways to further improve the accuracy of image-based localization using CNN-based architectures, we adopt some recent advances discovered in efforts solving the problems of image restoration [19], semantic segmentation [22] and human pose estimation [20]. Inspired by these ideas, we propose to add more context to the regression process to better collect the overall information, from coarse structures to fine-grained object details, available in the input image. We argue that this kind of a mechanism is suitable for getting an accurate camera pose estimate using CNNs. In detail, we propose a network architecture which consists of a bottom part (the encoder) that is used to encode the overall context and a latter part (the decoder) that recovers the fine-grained visual information by up-convolving the output feature map of the encoder by gradually increasing its size towards the original resolution of the input image. Such a symmetric ”encoder-decoder” network structure is also known as an hourglass architec-

ture [20].

The contributions of this paper can be summarized as follows:

- We complement a deep convolutional network by adding a chain of up-convolutional layers with shortcut connections and apply it to the image-based localization problem.
- The proposed network significantly outperforms the current state-of-the-art methods proposed in the literature for estimating camera pose.

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3 we provide the details of the proposed CNN architecture. Section 4 presents the experimental methodology and results on a standard evaluation dataset. We conclude with a summary and ideas for future work.

The source code and trained models will be publicly available upon publication.

2. Related Work

Image-based localization can be solved by casting it as a place recognition problem. In this approach, image retrieval techniques are often applied to find similar views of the scene in a database of images for which camera position is known. The method then estimates an approximate camera pose using the information in retrieved images. As noted in [29], these methods suffer in situations where there are no strong constraints on the camera motion. This is due to the number of the key-frames that is often very sparse.

Perhaps a more traditional approach to image-based localization is based on finding correspondences between a query image and a 3D scene model reconstructed using SfM. Given a query image and a 3D model, an essential part of this approach is matching points from 2D to 3D. The main limitation of this approach is the 3D model that may grow eventually too big in its size or just go too complex if the scene itself is somehow complicated, like large-scale urban environments. In such scenarios, the ratio of outliers in the matching process often grows too high. This in turn results in a growth in the run-time of RANSAC. There are methods to handle this situation, such as prioritizing matching regions in 2D to 3D and/or 3D to 2D and using co-visibility of the query and the model [24].

Applying machine learning techniques has proven very effective in image-based indoor localization. Shotton *et al*. [25] proposed a method to estimate scene coordinates from an RGB-D input using decision forests. Compared to traditional algorithms based on matching point correspondences, their method removes the need for the traditional pipeline of feature extraction, feature description, and matching. Valentin *et al*. [29] further improved the method by exploiting uncertainty in the model in order to move from sole point estimates to predict also their uncertainties for more robust continuous pose optimization. Both of these methods are designed for cameras that have an RGB-D sensor.

Very recently, applying deep learning techniques has resulted in remarkable performance improvements in many computer vision problems [1, 19, 22]. Partly motivated by studies applying CNNs and regression [26, 31, 27], Kendall *et al*. [15] proposed an architecture trying to directly regress camera relocalization from an input RGB image. More recent CNN-based approaches cover those of Clark *et al*. [4] and Walch *et al*. [30]. Both of these follow [15], and similarly adopt the same CNN architecture, by pre-training it first on large-scale image classification data, for extracting features from input images to be localized. In detail, Walch *et al*. [30] consider these features as an input sequence to a block of four LSTM units operating along four directions (up, down, left, and right) independently. On top of that, there is a regression part which encompasses fully-connected layers for predicting the camera pose. In turn, Clark *et al*. [4] applied LSTMs to predict camera translation only, but using short videos as an input. Their method is a bidirectional recurrent neural network (RNN), which captures dependencies between adjacent image frames yielding refined accuracy of the global pose. Both of the two architectures lead to improvement in the accuracy of 6-DoF camera pose outperforming PoseNet [15].

Compared to non-CNN based approaches, our method belongs to the very recent initiative of models that do not require any online 3D models in camera pose estimation. In contrast to [25, 29], our method is solely based on monocular RGB images and no depth information is required. Compared to PoseNet [15], our method aims at better utilization of context and provides improvement in pose estimation accuracy. In comparison to [30], our method is more accurate in indoor locations. Finally, our method does not rely on video inputs, but still outperforms the CNN-model presented in [4] for video-clip relocalization.

3. Method

Following [15, 30], our goal is to estimate camera pose directly from an RGB image. We propose a CNN architecture that predicts a 7-dimensional camera pose vector $\mathbf{p} = [\mathbf{q}, \mathbf{t}]$ consisting of an orientation component $\mathbf{q} = [q_1, q_2, q_3, q_4]$ represented by quaternions and a translation component $\mathbf{t} = [t_1, t_2, t_3]$.

Hiding the architectural details, the overall network structure is illustrated in Fig. 1. The network consists of three components, namely *encoder*, *decoder* and *regressor*. The encoder is fully convolutional acting as a feature extractor. The decoder consists of up-convolutional layers stacked to recover the fine-grained details of the input from the de-

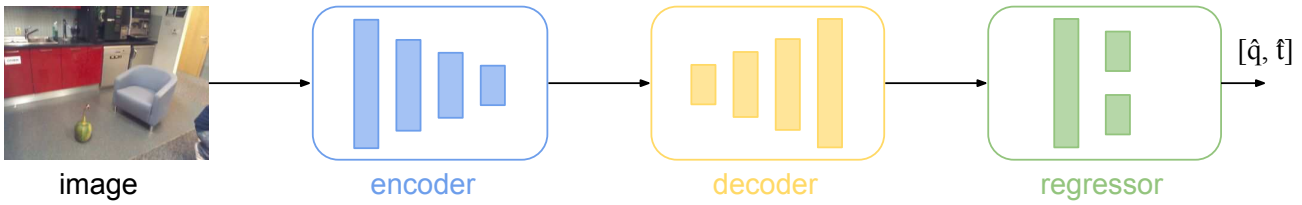


Figure 1: Overview of our proposed architecture. It takes an RGB-image as input and predicts the camera pose. The overall network consists of three components, namely encoder, decoder and regressor. The encoder is fully convolutional up until a certain spatial resolution. The decoder then gradually increases the resolution of the feature map which is eventually fed to the regressor that is composed of three fully connected layers.

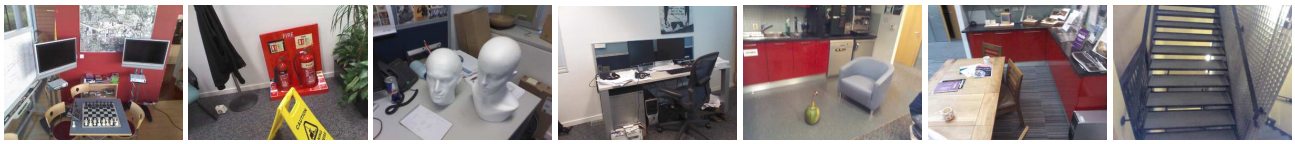


Figure 2: Visual representation of the categories of 7-Scenes dataset. From left to right: Chess, Fire, Heads, Office, Pumpkin, Red Kitchen and Stairs.

coder outputs. Finally, the decoder is followed by the regressor that estimates the camera pose $\mathbf{p}$.

To train our hourglass-shaped CNN model, we apply the following objective function [15]:

$$\mathscr{L} = \left\|\mathbf{t} - \hat{\mathbf{t}}\right\| + \beta \left\|\mathbf{q} - \frac{\hat{\mathbf{q}}}{\|\hat{\mathbf{q}}\|}\right\|, \tag{1}$$

where $(\mathbf{t}, \mathbf{q})$ and $(\hat{\mathbf{t}}, \hat{\mathbf{q}})$ are ground truth and estimated translation-orientation pairs, respectively. β is a scale factor, tunable by grid search, that keeps the estimated orientation and translation to be nearly equal. The quaternion based orientation vector $\mathbf{q}$ is normalized to unit length at test time. We provide the detailed information about the other hyperparameters used in training in Section 4.

3.1. CNN Architecture

Training convolutional neural networks from scratch for image-based localization task is impractical due to the lack of training data. Following [15], we leverage a pre-trained large-scale classification network. Specifically, to find a balance between the number of parameters of the network and accuracy, we adopt ResNet34 [10] architecture which has good performance among other classification approaches [3] as our base network. We remove the last fully-connected layer from the original ResNet34 model but keep the convolutional and pooling layers intact. The resulting architecture is considered as the encoder part of the whole pipeline.

Instead of connecting the encoder to the regression part directly, we propose to add some extra layers between them. In detail, we add three up-convolutional and one convolutional layer. The main idea of using up-convolutional layers is to restore essential fine-grained visual information of the input image lost in encoder part of the network. Up-convolutional layers have been widely applied in image restoration [19], structure from motion [28] and semantic segmentation [11, 21]. The proposed architecture is presented in Fig. 3.

Finally, there is a regressor module on top of the encoder. The regressor consists of three fully connected layers, namely localization layer, orientation layer and translation layer. In contrast to the regressor originally proposed in [15], we slightly modified its architecture by appending batch-normalization after each fully connected layer.

Inspired by the visualization of the steps of downsampling and upsampling of the feature maps flowing through encoder-decoder part and by [20]'s work, we call our CNN architecture `Hourglass-Pose`.

3.1.1 Hourglass-Pose

As explained, the encoder part of our architecture is the slightly modified ResNet34 model. It differs from the original one presented in [10] so that the final softmax layer and the last average pooling layer have been removed. As a result the spatial resolution of the encoder feature map is 7×7.

To better preserve finer details of the input image for the localization task, we added skip (shortcut) connections from

each of the four residual blocks of the encoder to the corresponding up-convolution and the final convolution layers of the decoder. The last part of the decoder, namely the final convolutional module (a chain of convolutional, batch-normalization [12] and ReLU layers) does not alter the spatial resolution of the feature map (56×56), but is used to decrease the number of channels. In our preliminary experiments, we also experimented with a Spatial Pyramid Pooling (SPP) layer [9] instead of the convolutional module. Particularly, SPP layer consists of a set of pooling layers (pyramid levels) producing a fix-sized feature map regardless the size of the input image. However, the camera pose estimations were not improved, and we omitted SPP in favor of simpler convolutional module. The encoder-decoder module is followed by a regressor which predicts the camera orientation $\mathbf{q}$ and translation $\mathbf{t}$. The detailed network configuration is shown in Table 1.

In order to investigate the benefits of using skip connections more thoroughly, we experimented with different aggregation strategies of the encoder and the decoder feature maps. In contrast to `Hourglass-Pose` where the outputs of corresponding layers are concatenated (See Fig. 3), we evaluated the whole pipeline by also calculating an element-wise sum of the feature maps connected via skip connections. We refer to the corresponding architecture as `HourglassSum-Pose`. Schematic illustration of a decoder-regressor part of this structure is presented in Fig. 4.

Module	Layers	Output Size	Hourglass-Pose
Encoder	Conv	112×112	7×7, 64 (3 / s2)
	Pool	56×56	3×3 max (s2)
	ResBlock1	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$ (1 / s1)
	ResBlock2	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$ (1 / s1)
	ResBlock3	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$ (1 / s1)
	ResBlock4	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$ (1 / s1)
Decoder	UpConv1	14×14	4×4 upconv (1 / s2)
	UpConv2	28×28	4×4 upconv (1 / s2)
	UpConv3	56×56	4×4 upconv (1 / s2)
	Conv	56×56	3×3, 32, (1 / s1)
Regressor	FC	2048	fully-connected
	$\text{FC}_\mathbf{q}$	4	
	$\text{FC}_\mathbf{t}$	3	

Table 1: Details of our Hourglass-Pose architecture for estimating camera pose. Note that each convolutional/upconvolutional layer in the encoder-decoder part corresponds 'Conv-ReLU-BatchNorm' sequence. In Res-blocks the resolution is downsampled with stride 2 convolutions. Upconvolution is implemented by first upsampling the signal by zero padding and then by applying normal convolution.

3.2. Evaluation Dataset

To evaluate our method and compare with the state-of-the-art approaches, we utilize Microsoft 7-Scenes Dataset containing RGB-D images of 7 different indoor locations [25]. The dataset has been widely used for camera relocalization [6, 15, 30, 4]. The images of the scenes were recorded with a camera of the Kinect device at 640×480 resolution and divided to train and evaluation parts accordingly. The ground truth camera poses were obtained by applying the KinectFusion algorithm [13] producing smooth camera trajectories. Sample images covering all scenes of the dataset are illustrated in Fig. 2. They represent indoor views of the 7 scenes exhibiting different lighting conditions, textureless (*e.g.* two statues in 'Heads') and repeated objects ('Stairs' scene), changes in viewpoint and motion blur. All of these factors make camera pose estimation an extremely challenging problem.

4. Experiments

In the following section we empirically demonstrate the effectiveness of the proposed approach on the 7-Scenes evaluation dataset and compare it to other state-of-the-art CNN-based methods. Like it was done in [15], we report the median error of camera orientation and translation in our evaluations.

4.1. Other state-of-the-art approaches

In this work we consider three recently proposed 6-DoF camera relocalization systems based on CNNs.

PoseNet is [15] is based on the GoogLeNet [26] architecture. It processes RGB-images and is modified so that all three softmax and fully connected layers are removed from the original model and replaced by regressors in the training phase. In the testing phase the other two regressors of the lower layers are removed and the prediction is done solely based on the regressor on the top of the whole network.

Bayesian PoseNet Kendall *et al.* [14] propose a Bayesian convolutional neural network to estimate uncertainty in the global camera pose which leads to improving localization accuracy. The Bayesian convolutional neural is based on PoseNet architecture by adding dropout after the fully connected layers in the pose regressor and after one of the inception layer (layer 9) of GoogLeNet architecture.

LSTM-Pose [30] is otherwise similar to PoseNet, but applies LSTM networks for output feature coming from the final fully connected layer. In detail, it is based on utilizing

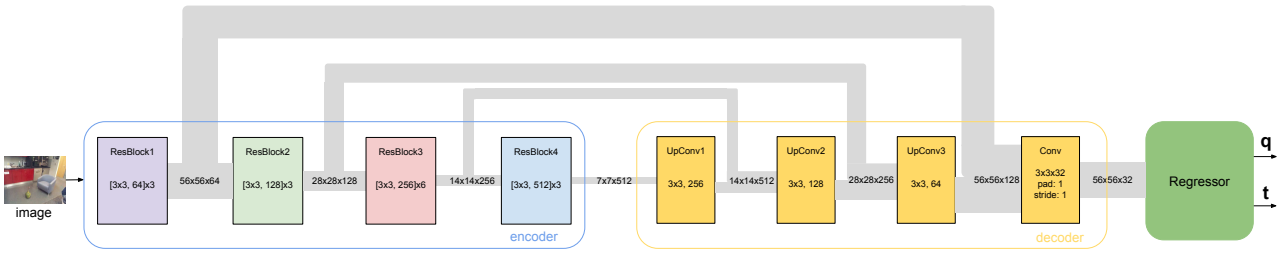


Figure 3: An illustration of the proposed architecture referred to as Hourglass-Pose for predicting camera pose. The encoder part is a modified version of ResNet34 [10], where We removed the last fully-connected and average pooling layers from the original ResNet34 arhitecture and kept only the convolutional layers. The decoder consists of a set of stacked up-convolutional layers gradually increasing the spatial resolution of the feature maps up to 56×56. We further added one convolutional layer for dimensionality reduction. Skip connections connect each block of the encoder to the corresponding parts of the decoder allowing the decoder to re-utilize features from the earlier layers of the network. Finally, camera pose is estimated by the regressor as explained in Section 3.

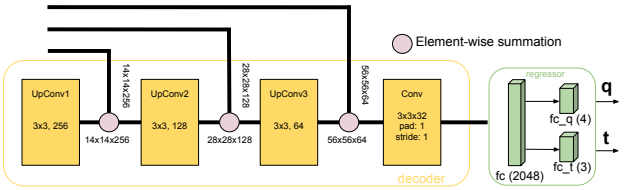


Figure 4: The structure of the decoder and the regressor of HourglassSum-Pose architecture for estimating camera pose $\mathbf{p} = [\mathbf{q}, \mathbf{t}]$. The output of the decoder is connected to the regressor consisting of a set of FC-layers to predict $\mathbf{q}$ and $\mathbf{t}$ respectively. The number of connections of each FC-layer is given in parenthesis.

the pre-trained GoogLeNet architecture as a feature extractor followed by four LSTM units applying in the up, down, left and right directions. The outputs of LSTM units are then concatenated and fed to a regression module consisting of two fully connected layers to predict camera pose.

VidLoc [4] is a CNN-based system based on short video clips. As in PoseNet and LSTM-Pose, VidLoc incorporates similarly modified pre-trained GoogLeNet model for feature extraction. The output of this module is passed to bidirectional LSTM units predicting the poses for each frame in the sequence by exploiting contextual information in past and future frames.

4.2. Training Setup

We trained our models for each scene of 7-Scenes dataset according to the data splits provided by [25].

For all of our methods, we take the weights of ResNet34 [10] pre-trained on ImageNet to initialize the encoder part with them. The weights of the decoder and the regressor are initialized according to [7]. Our initial learning rate is 10^{-3} and that is kept for the first 50 epochs. Then, we continue for 40 epochs with 10^{-4} and subsequently decrease it to 10^{-5} for the last 30 epochs.

As a preprocessing step, all images of the evaluation dataset are rescaled so that the smaller side of the image is always 256 pixels. We calculate mean and standard deviation of pixel intensities separately for each scene and use them to normalize intensity value of every pixel in the input image.

We trained our models using random crops (224×224) and performed the evaluation using central crops at the test time. All experiments were conducted on two NVIDIA Titan X GPUs with data parallelism using Torch7 [5]. We minimize the loss function (1) over a training part of each scene of the evaluation dataset using Adam [16] ($\beta_1 = 0.9$, $\beta_2 = 0.99$). The scale factor β (1) varies between 1 to 10. Training mini-batches are randomly shuffled in the beginning of each training epoch. We further used set the weight decay as 10^{-5}, used a mini-batch size of 40 and the dropout probability as 0.5. These parameter values were kept fixed during our experiments.

4.3. Results

To compare Hourglass-Pose and HourglassSum-Pose architectures with other state-of-the-art methods, we follow the evaluation protocol presented in [15]. Specifically, we report the median error of camera pose estimations for all scenes of the 7-Scenes dataset. Like in [14, 30, 4], we also provide an average median orientation and translation error.

Table 2 shows the performance of our approaches along with the other state-of-the-art. The values for other methods are taken from [15], [14], [30], and [4]. According to the results, several conclusions can be drawn. First, our architectures clearly outperform the other state-of-the-art CNN-based approaches. In general, HourglassSum-Pose

Scene	Frames		Spatial	PoseNet	Bayesian	LSTM-Pose	VidLoc	Hourglass-Pose	HourglassSum-Pose
	Train	Test	Extent	ICCV'15 [15]	PoseNet [14]	[30]	[4]		
Chess	4000	2000	$3 \times 2 \times 1$m	0.32m, 8.12°	0.37m, 7.24°	0.24m, 5.77°	0.18m, N/A	0.15m, 6.53°	0.15m, 6.17°
Fire	2000	2000	$2.5 \times 1 \times 1$m	0.47m, 14.4°	0.43m, 13.7°	0.34m, 11.9°	0.26m, N/A	0.29m, 11.59°	0.27m, 10.84°
Heads	1000	1000	$2 \times 0.5 \times 1$m	0.29m, 12.0°	0.31m, 12.0°	0.21m, 13.7°	0.14m, N/A	0.21m, 14.52°	0.19m, 11.63°
Office	6000	4000	$2.5 \times 2 \times 1.5$m	0.48m, 7.68°	0.48m, 8.04°	0.30m, 8.08°	0.26m, N/A	0.21m, 9.25°	0.21m, 8.48°
Pumpkin	4000	2000	$2.5 \times 2 \times 1$m	0.47m, 8.42°	0.61m, 7.08°	0.33m, 7.00°	0.36m, N/A	0.27m, 6.93°	0.25m, 7.01°
Red Kitchen	7000	5000	$4 \times 3 \times 1.5$m	0.59m, 8.64°	0.58m, 7.54°	0.37m, 8.83°	0.31m, N/A	0.27m, 9.82°	0.27m, 10.15°
Stairs	2000	1000	$2.5 \times 2 \times 1.5$m	0.47m, 13.8°	0.48m, 13.1°	0.40m, 13.7°	0.26m, N/A	0.29m, 13.07°	0.29m, 12.46°
Average				0.44m, 10.4°	0.47m, 9.81°	0.31m, 9.85°	0.25m, N/A	0.24m, 10.24°	0.23m, 9.53°

Table 2: Performance comparison of two architectures (Hourglass-Pose and HourglassSum-Pose) and state-of-the-art methods on 7-Scenes evaluation dataset. Numbers are median translation and orientation errors for the entire test subset of each scene. Both models significantly outperform PoseNet [10] and LSTM-Pose [30] in terms of localization. It is a crucial observation emphasizing the importance of re-utilizing feature maps by using direct (skip) connections between encoder and decoder modules for image-based relocalization task. An Hourglass-Pose and HourglassSum-Pose architectures' comparison reveals that applying element-wise summation is more beneficial than features concatenation providing more accurate camera pose. Remarkably, the proposed models do perform even better than VidLoc [4] approach, which uses a sequence of test frames to estimate camera pose.

improves the accuracy of the camera position by 52.27% and orientation by 8.47% for average error with respect to PoseNet. Furthermore, HourglassSum-Pose manages to achieve better orientation accuracy than LSTM-Pose [30] in all scenes of the evaluation dataset. It can be seen that both of our architectures are even competitive with VidLoc [4] that is based on a sequence of frames. Our methods improve the average position error by 1 cm and 2 cm. The results in Table 2 confirm that it is beneficial to utilize an hourglass architecture for image-based localization.

For a more detailed comparison, we plot a family of cumulative histogram curves for all scenes of the evaluation dataset illustrated in Fig. 5. We note that both hourglass architectures outperforms PoseNet method on translation accuracy by a factor of 1.5 to 2.3 in all test scenes. Besides that, HourglassSum-Pose substantially improves orientation accuracy. The only exception is 'Office' and 'Red Kitchen' scenes where performance of HourglassSum-Pose is on par with PoseNet.

Figure 6 shows histograms of localization accuracy for both orientation (left) and position (right) for the two entire test scenes of the evaluation dataset. It is interesting to see that more than 60% of camera pose estimations produced by HourglassSum-Pose are within 20 cm in 'Chess' scene, while for PoseNet this quotient is equal to 5%. Remarkably, HourglassSum-Pose is able to improve accuracy even for such an ambiguous and challenging scene like 'Stairs' exhibiting many repetitive structures (See Fig. 6b). The presented results verify that an hourglass neural architecture is an efficient and promising approach for image-based localization.

5. Conclusion

In this paper, we have presented an end-to-end trainable CNN-based approach for image-based localization. One of the key aspect of this work is applying encoder-decoder (hourglass) architecture consisting of a chain of convolutional and up-convolutional layers for estimating 6-DoF camera pose. Furthermore, we propose to use direct connections forwarding feature maps from early residual layers of the model directly to the later up-convolutional layers improving the accuracy. We studied two hourglass models and showed that they significantly outperform other state-of-the-art CNN-based image-based localization approaches.

References

[1] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. In *Proc. ECCV*, 2014. 1, 2

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008. 1

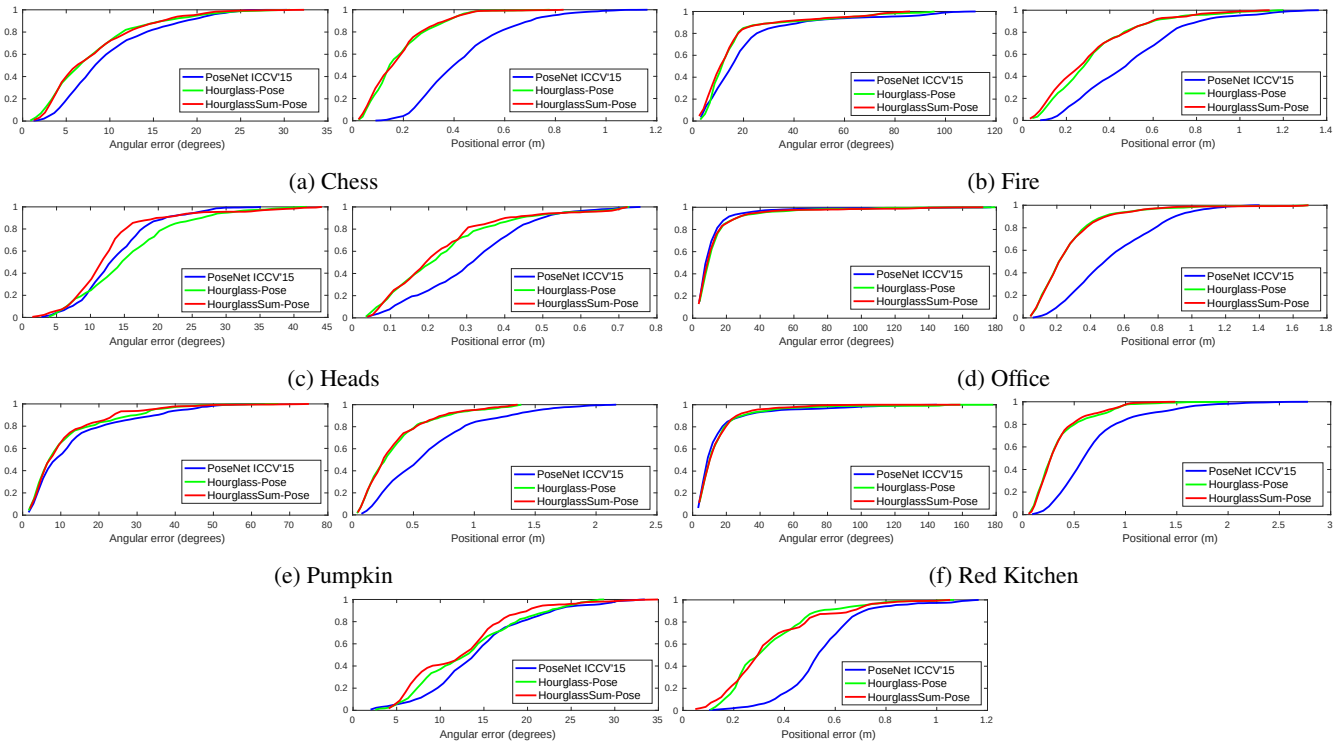
[3] A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *IEEE International Symposium on Circuits and Systems*, 2016. 3

[4] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. VidLoc: 6-DoF video-clip relocalization. In *Proc. CVPR*, 2017. 2, 4, 5, 6

[5] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *Proc. BigLearn, NIPS Workshop*, 2011. 5

[6] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-time RGB-D camera relocalization. In *Proc. ISMAR*, 2013. 4

[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, 2010. 5



(a) Chess

(b) Fire

(c) Heads

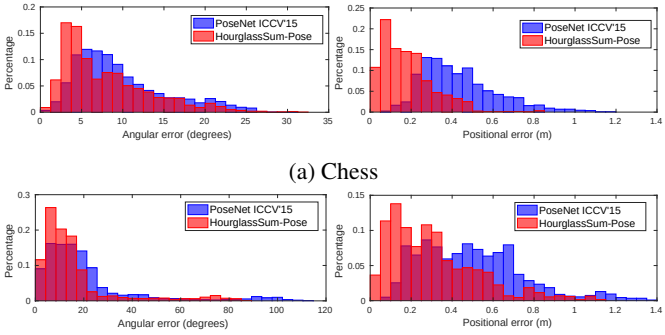
(d) Office

(e) Pumpkin

(f) Red Kitchen

(g) Stairs

Figure 5: Localization performance of the proposed hourglass-based network architectures (Hourglass-Pose and HourglassSum-Pose) presented as a cumulative histogram (normalized) of errors for all categories of 7-Scenes dataset. One of the important conclusion is that both architectures can significantly improve the accuracy of estimations camera location clearly outperforming state-of-the-art method (PoseNet). HourglassSum-Pose achieves better orientation performance in 5 cases to compare to Hourglass-Pose architecture.



(a) Chess

(b) Stairs

Figure 6: Histogram of orientation (left) and translation (right) errors of two approaches (PoseNet and HourglassSum-Pose) for the two entire scenes ('Chess' and 'Fire') of the evaluation dataset. It is clearly seen that an hourglass-architecture-based method performs consistently better than PoseNet.

[8] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *Proc. ECCV*, 2016. 1

[9] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015. 4

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 1, 3, 5, 6

[11] S. Hong, H. Noh, and B. Han. Decoupled deep neural network for semi-supervised semantic segmentation. In *Proc. NIPS*, 2015. 3

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 4

[13] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, 2011. 4

[14] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Proc. ICRA*, 2016. 4,

5, 6

[15] A. Kendall, M. Grimes, and R. Cipolla. Convolutional networks for real-time 6-DOF camera relocalization. In *Proc. ICCV*, 2015. 1, 2, 3, 4, 5, 6

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*. 5

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in NIPS*, 2012. 1

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 1

[19] X. Mao, C. Shen, and Y. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in NIPS*, 2016. 1, 2, 3

[20] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *Proc. ECCV*, 2016. 1, 2, 3

[21] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. ICCV*, 2015. 3

[22] P. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *Proc. ECCV*, 2016. 1, 2

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to SIFT or SURF. In *Proc. ICCV*, 2011. 1

[24] T. Sattler, B. Leibe, and L. Kobbelt. Efficient and effective prioritized matching for large-scale image-based localization. *IEEE TPAMI*, 2016. 2

[25] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *Proc. CVPR*, 2013. 2, 4, 5

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. 2, 4

[27] D. Tomè, C. Russell, and L. Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. *CoRR*, abs/1701.00295, 2017. 2

[28] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. DeMoN: Depth and motion network for learning monocular stereo. In *Proc. CVPR*, 2017. 3

[29] J. Valentin, M. Niebner, J. Shotton, and P. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proc. CVPR*, 2015. 2

[30] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization with spatial LSTMs. In *Proc. ICCV*, 2017. 2, 4, 5, 6

[31] X. Xi, Y. Luo, F. Li, P. Wang, and H. Qiao. A fast and compact saliency score regression network based on fully convolutional network. *CoRR*, abs/1702.00615, 2017. 2