Towards CNN map representation and compression for camera relocalisation

Luis Contreras[1] and Walterio Mayol-Cuevas[2]

***Abstract*— This paper presents a study on the use of Convolutional Neural Networks for camera relocalisation and its application to map compression. We follow state of the art visual relocalisation results and evaluate the response to different data inputs. We use a CNN map representation and introduce the notion of map compression under this paradigm by using smaller CNN architectures without sacrificing relocalisation performance. We evaluate this approach in a series of publicly available datasets over a number of CNN architectures with different sizes, both in complexity and number of layers. This formulation allows us to improve relocalisation accuracy by increasing the number of training trajectories while maintaining a constant-size CNN.**

I. Introduction

Following our recent work on point cloud compression mapping via feature filtering in [1] and [2], we aim to generate compact map representations useful for camera relocalisation through compact Convolutional Neural Networks (CNNs). This effort is motivated by the end-to-end approach of CNNs and in order to extend such to map compression. Overall, having a minimal map representation that enables later use is a meaningful question that underpins many applications for moving agents. In this work, we specifically explore a neural network architecture tested for the relocalisation task; we study the response of such architecture to different inputs – e.g. color and depth images –, and the relocalisation performance of pre-trained neural networks in different tasks.

Biologically inspired visual models have been proposed for a while [3], [4]. How humans improve learning after multiple training of the same view and how they filter useful information have also been an active field of study. One widely accepted theory of the human visual system suggests that a number of brain layers sequentially interact from the signal stimulus to the abstract concept [5]. Under this paradigm, the first layers – connected directly to the input signal – are a series of specialized filters that extract very specific features, while deeper layers infer more complex information by combining these features.

Finally, overfitting a neural network by excessive training with the same dataset is a well known issue; rather, here we study how the accuracy improves by revisiting the same area several times introducing new views to the dataset.

*This work was partially supported by CONACYT and the Secretaria de Educacion Publica, Mexico

Department of Computer Science, University of Bristol, United Kingdom
[1]cslact@my.bristol.ac.uk
[2]wmayol@cs.bris.ac.uk

This paper is organized as follows. In Section II we discuss work related to convolutional neural networks and camera pose. Then, Section III introduces the notion of CNN map representation and compression. The CNN architectures used in the relocalisation task are then introduced in Section IV, where we describe their architecture. Experimental results are presented in Section V. Finally, we outline our discussion and conclusions.

II. Related Work

Even though neural networks are not a novel concept, due to the increase in computational power, their popularity has grown in recent years [6] [7]. Related to map compression, dimensionality reduction through neural networks was first discussed in [8]. In [9] an evaluation to up-to-date data encoding algorithms for object recognition was presented, and it was extended in [10] to introduce the use of Convolutional Neural Networks for the same task.

[11] introduced the idea of egomotion in CNN training by concatenating the output of two parallel neural networks with two different views of the same image; at the end, this architecture learns valuable features independent of the point of view.

In [12], the authors concluded that sophisticated architectures compensate for lack of training. [13] explore this idea for single view depth estimation where they present a stereopsis based auto-encoder that uses few instances on the KITTI dataset. Then, [14], [15], and [16] continued studying the use of elaborated CNN architectures for depth estimation.

Moving from depth to pose estimation was the next logical step. One of the first 6D camera pose regressors was presented in [17] via a general regression NN (GRNN) with synthetic poses. More recently, PoseNet is presented in [18], where they regress the camera pose using a CNN model. In the same sense, [19] presented VidLoc, where they improve PoseNet results in offline video sequences by adding a biderctional RNN that takes advantage of the temporal information in the camera pose problem. This idea is also explored in [20] for image matching via training a CNN for frame interpolation through video sequences.

III. Map Representation as a Regression Function

From a human observer point of view, it is common to think of spatial relationships among elements in space to build maps; for this reason, metric, symbolic, and topological are widely used map representations (such as probabilistic [21], topological [22], and metric and topological [23] map

representations). However, other less intuitive map representation have been proposed – e.g. [24] defines a map as a collection of images and uses image batch matching to find the current position in the map.

Overall, it can be argued that the map representation needs not conform to a *single* representation type, and that the task and other constraints can lead to different manners in which a map can be represented. Ultimately, for a robotic agent, maps are likely built to be explored or, more generally, re-explored. Thus, it is highlighted once more that relocalisation is a good measure of map effectiveness. In this context, the actual map representation used is less relevant as long as it allows relocalisation in a specific scene or task; therefore, we propose a mapping process based on Convolutional Neural Network, or CNN, to address the camera relocalisation problem.

A CNN can be considered as a filter bank where the filters' weights are such that they minimize the error between an expected output and the system response to a given input. Figure 1 shows the elements from one layer to the next in a typical CNN architecture – a more detailed CNN implementation can be found in specialized works such as [25] and [26]. From the figure, for a given input I and a series of k filters f_k, it is generated an output $\hat{I}_k = I * f_k$, where $*$ represents the convolution operator (hence, this layer is also called *convolutional* layer), where the filters f_k can be initialized randomly or with pre-trained weights in a different task. It is important to notice the direct relationship between the input channels and the filters' depth among consecutive layers; it makes possible to work with different n-dimensional inputs just by adjusting the first convolutional layer depth.

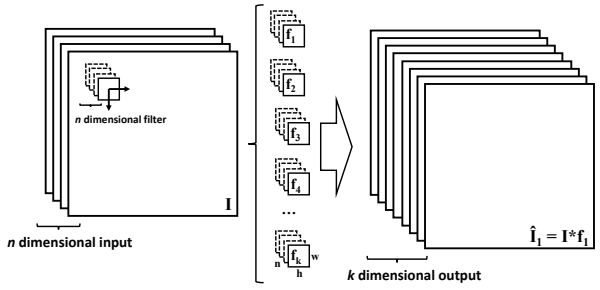


Fig. 1: Convolutional Neural Network (CNN) elements. It consist of an input I, a series of filters f_k, and its associated output $\hat{I}_k$ as a result of the convolution $I * f_k$. The filters' depth n depends on the number of input channels.

As a result, in this work we represent a map as a regression function $\hat{p} = cnn(I)$, where each element of the population is formed by an input I and its associated output p (e.g. an RGB image and the 6-DoF camera pose, respectively). The parameters in the regressor cnn are optimised from a population sample; the more representative the sample, the more accurate the model [27].

A. CON-POCO: CNN Map Compression

The notion of compression using a regression function as a map representation is introduced as follows. First, a population sample is defined as a collection of elements (I, p) that represents a sensor's travelled trajectory and, further, this collection can be divided in training and testing sets. From the training set, a regressor $\hat{p} = cnn(I)$ is proposed such that it minimises the error $|p - \hat{p}|$ over the test set.

This regressor, once defined, will have constant size, and should improve its performance while increasing the training set size (e.g. by generating more training trajectories) without increasing the regressor size itself. The compact map representation under the CNN paradigm is then stated as the problem of finding an optimal model cnn(I) that keep minimum relocalisation error values given a population sample.

IV. The Relocalisation CNN

To evaluate the CNN map representation in the relocalisation task, we test several CNN architectures of the form $\hat{p} = cnn(I)$, where I is an input image and the expected output is a 6-DoF pose $p = [x, q]$, with x as the spatial position and q as the orientation in quaternion form. We use PoseNet loss function, as described in [18], that has the form:

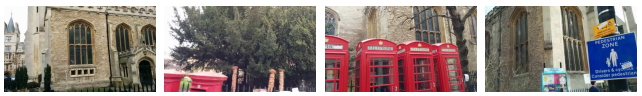
$$loss(I) = \|\hat{x} - x\|_2 + \beta \left\| \hat{q} - \frac{q}{\|q\|} \right\|_2$$

where PoseNet is based on the GoogLeNet arquitecture [28], and β is a scale factor. As a reference, GoogLeNet has 7 million parameters but with a more elaborated architecture [29]; in contrast, in this work to evaluate the relocalisation performance with respect to the CNN size, we only vary the number of convolutional layers and no interaction among them is introduced. Learning the external parameters in a CNN is time consuming because there is not really an optimal approach to this task; for this reason, the use of generic representations has been proposed such as in [30], where the models trained in one task can be used in another, a process known as *transfer learning*. Thus, we tested several architectures using a series of pre-trained models on the ImageNet dataset [31] and implemented in the MatConvNet platform [32], as detailed bellow.

First, we use a relatively small CNN architectures with different complexities, as in [10], with eight layers: five convolutional and three fully-connected. We use three implementations: a fast architecture (VGG-F) with 61 million parameters (considering an RGB input), where the first convolutional layer has a four pixel stride; a medium architecture (VGG-M) and 100 million parameters, where a smaller stride and a smaller filter size with a bigger depth are used in the first convolutional layer, and bigger filters' depths are used in the remaining convolutional layers. Finally, we study a slow architecture (VGG-S), with a similar architecture and number of parameters as in the VGG-M, but with a smaller stride in the second convolutional layer.

Moreover, we evaluated two long CNN architectures, as in [33], one with 16 layers (13 convolutional and three fully connected layers with 138 million parameters) or VGG-16, and the other with 19 layers (16 convolutional and three fully connected layers with a total of 144 million parameters), referred as VGG-19. We introduce a couple of changes to these networks as follows: the dimension in the first layer depends on the input n; in addition, the final fully-connected layer size changes to the pose vector length (i.e. from 4096 to 7).

To demonstrate the impact of smaller architectures in the relocalisation problem, we evaluate their performance in the St Marys Church sequence, a large scale outdoor scene [34], with 1487 training and 530 testing frames, as shown in Figure 2 and we only use RGB information as input, and pre-processing the original input by cropping the central area and resizing it, generating arrays of size 224x224. We compare their performance against PoseNet, as reported in [18].



(a) (b) (c) (d)

Fig. 2: Typical views from the St Marys Church sequence dataset [34].

The training process is described next. First, we use the fast implementation (VGG-F) to find a valid range for the hyper-parameters involved in the CNN response, namely the batch size, the learning rate, and the weight decay. Then, with this valid hyper-parameters range, we perform a similar experiment using the proposed CNN architectures for the relocalisation task in the same dataset (the St Marys Church sequence) to find a valid hyper-parameters range in all architectures. We evaluate twenty hyper-parameters combination per architecture for 250 epochs.

Given the best hyper-parameters range from the process described above (batch size of 30, weight decays of 5E-01, and learning rates from 1E-06 to 1E-10), we perform an evaluation of the relocalisation performance using smaller CNN architectures than that in the original PoseNet. In general, larger architectures present better performance; however, none of them outperforms the baseline, as described bellow.

Figure 3 shows the error in position, defined as the Euclidean distance between the expected position x and the CNN response $\hat{x}$, $e_p = \|x - \hat{x}\|$; here, the minimum error obtained was 4.76 meters using the VGG-19 CNN architecture (for comparison, PoseNet's error is 2.65 meters). Then, the smallest error in orientation – in this work, it is given by the angle between the two quaternion vectors $e_a = cos^{-1}(< q, \hat{q} >)$ – is 10.40°, also given by the VGG-19 CNN architecture, and again PoseNet (whose error is 4.24°) outperforms in this scenario.

Given that a regression model performance increases with the number of training samples, we perform a similar evaluation in a larger dataset to assess their effect in the CNN map

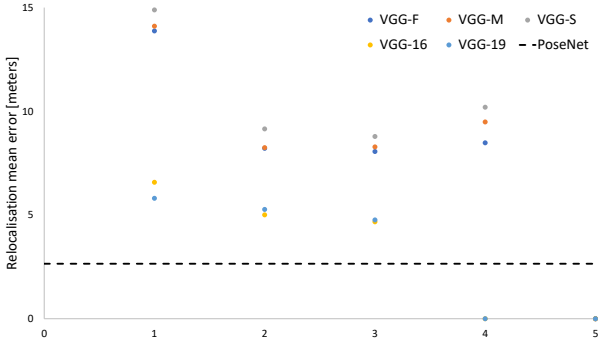


Fig. 3: Relocalisation performance to different CNN architectures in the St Marys Church dataset [34], where five parameter combinations were used per architecture during 500 training epochs: batch size of 30, weight decays of 5E-01, and learning rates from 1E-06 to 1E-10, where the position error is defined as the Euclidean distance between camera positions $\hat{x}$, $e_p = \|x - \hat{x}\|$. In dotted lines are the results for PoseNet as in [18].

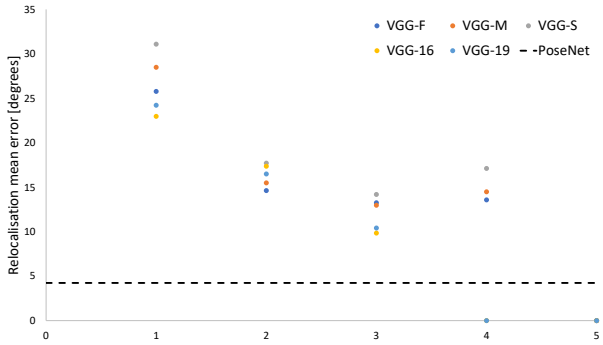
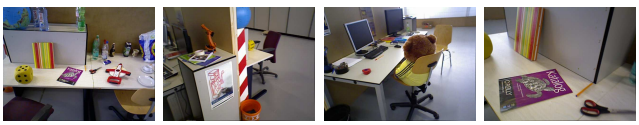


Fig. 4: Orientation error, as the angle between quaternion orientations $e_a = cos^{-1}(< q, \hat{q} >)$. Parameters are as in to Figure 3.

representation. In particular, we tested these implementations in the TUM's long household and office sequence [35] – a texture and structure rich scene, with 21.5m in 87.09s (2585 frames), and a validation dataset with 2676 extra frames, as can be seen in Figure 5.



(a) (b) (c) (d)

Fig. 5: Some scenes in the TUM long office sequence [35].

Figure 6 shows the relocalisation performance for different CNN architectures – to discover the difference in performance between architectures, we use the Euclidean distance between pose vectors $e = \|p - \hat{p}\|$ as error metric, where $p = [x, q]$, and $\hat{p}$ is the estimated camera pose. It is observed that, for some parameter combinations, there is no significant difference in performance between short and long

architectures, and therefore short architectures (i.e. compact map representations) may be used in the relocalisation task in such cases.

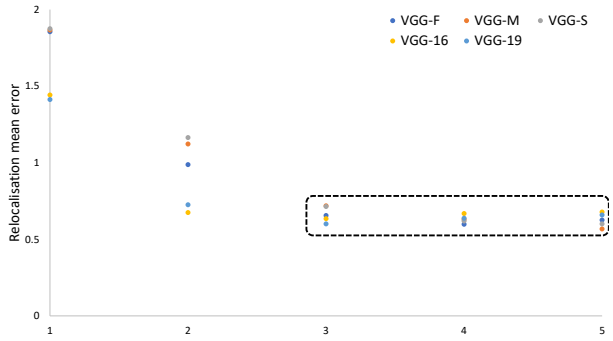


Fig. 6: Relocalisation response, as the distance between pose vectors $e = \|p - \hat{p}\|$, among different CNN architectures in the TUM long office dataset [35]; each point represents a hyper-parameter combination with a batch size of 30, weight decays of 5E-01, and learning rates from 1E-06 to 1E-10. For the highlighted combinations, there is no significant difference in the performance among short and long architectures in this dataset.

V. EXPERIMENTS AND RESULTS

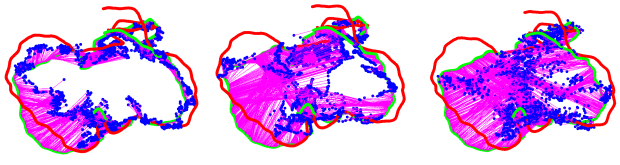
A. CNN for camera relocalisation

In this section we study the relocalisation performance depending on the input type where we use a fixed architecture while the nature of the input varies. For ease, we use a fast implementation; more specifically, the VGG-F architecture is tested with pre-trained weights optimized for object detection in the ImageNet dataset [31] as well as with randomly initialised weights, and we evaluated them in the TUM's long household and office sequence [35], as no significant difference between architectures has been observed in this scene and therefore the difference in performance is attributable to the difference in the input.

First, the response to an RGB input can be seen in Figure 7, where the red line indicates the training sequence and the green line the test one – results show a relocalisation mean error of 0.572 meters using pre-trained weights (Figure 7a), and 0.867 meters when random weights were used (Figure 7b), where the difference in performance is attributable to the filters initialisation.

Then, we evaluate other sensor inputs with a different nature than RGB data, e.g. Figure 7c shows the relocalisation performance for a dense point cloud of 3D points. The CNN weights were randomly initialized, as pre-trained CNNs are more common for RGB than depth or spatial data.

Table I shows the relocalisation mean error for all different inputs after 1000 epochs. Again, the pre-trained CNN on RGB information seems to outperform the others; however, with exception of the pre-trained cases, there is not a significant difference among the distinct input data, and this is attributable to the lack of training samples. We also can observe that, when combined information layers are used, the performance decreases, what might be due to the difference in the input nature (color, depth, and spatial position). One way to overcome it can be the use of parallel networks for each input and then average the output, as in [11].



(a) (b) (c)

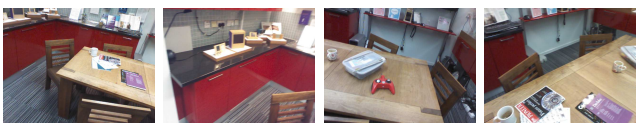
Fig. 7: Relocalisation performance in the TUM sequence [35] and a) an RGB input with a pre-trained CNN in the ImageNet dataset [31], b) an RGB input and a randomly initialized CNN, and c) a point cloud of 3D points input. Red line indicates the training trajectory while the green line is the testing one. Blue points are the neural network's output.

TABLE I: CNN-F relocalisation mean error [in meters] for different inputs after 1000 epochs in the long office sequence from the TUM dataset [35].

Input	Relocalisation mean error	
	Position [m]	Angle [°]
Depth	1.768 ± 0.568	44.93 ± 32.78
Gray	0.832 ± 0.675	31.91 ± 42.13
Point Cloud	0.986 ± 0.834	39.14 ± 46.85
RGB	0.778 ± 0.675	27.39 ± 41.43
Pre-trained RGB	**0.465 ± 0.447**	**22.16 ± 40.91**
RGB+Depth	0.863 ± 0.730	28.68 ± 41.67
RGB+Point Cloud	2.330 ± 0.494	79.63± 24.39

B. Multiple trajectories learning

To test the relocalisation performance with respect to the number of training sequences, and hence the neural network capability for map representation and compression, we use the Microsoft's 7-Scenes dataset ([38], [36]), as shown in Figure 8; this dataset consists of several trajectories taken by different persons moving around the same environment. Training, validation and testing sequences are indicated in the dataset itself.



(a) (b) (c) (d)

Fig. 8: Typical views from the Red Kitchen sequence in the 7-Scenes dataset [38].

Similar to the previous section, we evaluate the VGG-F architecture with all different input data and compared their response against PoseNet [18]; results are shown in Table IV. Although RGB error is the lowest again, in this case, where more training data per set is present, similar performances (within the variance error) are found among different data

TABLE II: Relocalisation mean error [in meters] for different architectures in several datasets given an RGB input.

	PoseNet	TUM	7Scenes		
	St Marys Church	Long Office	Pumpkin	Red Kitchen	Office
VGG-F	8.061 ± 6.193	0.541 ± 0.342	0.575 ± 0.278	0.616 ± 0.430	0.478 ± 0.258
VGG-M	8.282 ± 6.489	0.554 ± 0.384	0.606 ± 0.299	0.590 ± 0.453	0.489 ± 0.276
VGG-S	8.784 ± 6.694	0.544 ± 0.359	0.608 ± 0.315	0.628 ± 0.454	0.521 ± 0.302
VGG-16	4.671 ± 4.419	0.468 ± 0.367	0.448 ± 0.272	0.483 ± 0.352	0.345 ± 0.197
VGG-19	4.760 ± 4.620	0.470 ± 0.362	0.446 ± 0.264	0.471 ± 0.372	0.350 ± 0.217
PoseNet [18]	2.65	NA	0.47	0.59	0.48
SCoRe Forest [36]	NA	NA	0.04	0.04	0.04
ORB-SLAM2 [37]	NA	0.01	NA	NA	NA

TABLE III: Relocalisation mean error [in degrees] for different architectures in several datasets and an RGB input.

	PoseNet	TUM	7Scenes		
	St Marys Church	Long Office	Pumpkin	Red Kitchen	Office
VGG-F	13.25 ± 15.14	25.63 ± 44.68	9.67 ± 6.89	10.67 ± 9.24	10.66 ± 7.90
VGG-M	12.97 ± 16.57	24.72 ± 39.82	9.04 ± 6.79	10.82 ± 8.68	11.07 ± 7.71
VGG-S	14.18 ± 19.30	25.99 ± 41.96	9.72 ± 8.39	11.14 ± 11.82	11.76 ± 8.41
VGG-16	9.84 ± 16.59	28.96 ± 43.46	9.59 ± 7.08	8.45 ± 7.75	8.35 ± 7.10
VGG-19	10.41 ± 17.40	25.68 ± 42.19	8.88 ± 7.41	8.10 ± 7.57	9.10 ± 8.27
PoseNet [18]	4.24	NA	4.21	4.32	3.84
SCoRe Forest [36]	NA	NA	0.68	0.76	0.78
ORB-SLAM2 [37]	NA	NA	NA	NA	NA

types. Therefore, without lost of generality, for the rest of the section we will use only RGB-only data.

Additionally, from Table IV we observe that, in the case of RGB data, the VGG-F (8 layers) behaves as good as PoseNet, a 23 layers and more complex neural network, with a relocalisation mean error of 0.559 meters in the former and 0.59 meters in the latter, and therefore a compression in the number of layers is achieved. It remains an open problem the task of designing customized CNN map representation by systematically modifying the neural network architecture itself.

TABLE IV: Relocalisation mean error [in meters] using VGG-F for different inputs after 1000 epochs in the Red Kitchen sequence from the 7-Scenes dataset[38]. PoseNet mean error is indicated in italics, as reported in [18].

Input	Relocalisation mean error	
	Position [m]	Angle [°]
Depth	1.261 ± 0.382	20.24 ± 12.21
Gray	0.747 ± 0.493	12.37 ± 11.12
Point Cloud	0.740 ± 0.666	14.11 ± 13.94
RGB	**0.559 ± 0.417**	**8.57 ± 7.86**
PoseNet (RGB)	*0.59*	*4.32*
RGB+Depth	0.704 ± 0.538	11.77 ± 11.22
RGB+Point Cloud	0.640 ± 0.661	12.12 ± 13.92

The different architectures evaluated in this work are presented in Table V. As a reference, it is presented SCoRe Forest [36], a regression forest trained for pixel to 3D point correspondence prediction – the authors used 5 trees with 500 images per tree and 5000 example pixels per image. It is also presented ORB-SLAM2 [37], where a map with 16k features was generated in the TUM's long office scene [35].

TABLE V: Relocalisation mean error [in degrees] for different architectures in several datasets. PoseNet has fewer parameters in a more complex architecture (parameter estimation based on the GoogLeNet architecture [29]).

Model	Parameters	Layers
VGG-F	61M	8
VGG-M	100M	8
VGG-S	100M	8
VGG-16	138M	16
VGG-19	140M	19
PoseNet [18]	7M	23
SCoRe Forest [36]	12.5M	NA
ORB-SLAM2 [37]	16k	NA

Table II and Table III present a summary of the relocalisation response in position and orientation, respectively, for the best combinations in each architecture using different datasets and an RGB input; again, from an expected pose $p = [x, q]$ and the CNN response $\hat{p} = [\hat{x}, \hat{q}]$, the error in position is given by $e_p = \|x - \hat{x}\|$, while the error in orientation is given by $e_a = cos^{-1}(< q, \hat{q} >)$. These results confirm that, when enough training data is available, there is not significant difference in performance among short and long CNN architectures and, therefore, they may be used alike. In addition, it is observed that, with a relatively small regression map (12M in the SCoRe Forest), it is possible to obtain a high accuracy – as a CNN can approximate any function, it remains an open question how to find the best

CNN architecture that approximate those results.

Furthermore, in traditional mapping techniques, the map usually increases when new views are added; instead, when using a CNN map representation, map information increases while maintaining a neural network of constant size by re-training it when new information is added.

We use the Red Kitchen, Office and Pumpkin sequences in the Microsoft's 7-Scenes dataset to test the CNN saturation point as follows. One of the trajectories is left out for testing, and the CNN is gradually trained by adding one remaining sequence at a time. Figure 9 shows that while increasing the number of trajectories, precision also increases but, by construction, the size of the CNN remains the same, as expected. Similarly, Figure 10, Figure 11, and Figure 12 show the performance in this incremental setup for short and long architectures; no significant difference in relocalisation performance between CNN architectures is observed, and therefore the use of compact CNN map representation in such scenarios is possible.

Nevertheless, an asymptotic behaviour has not been reached after using all the training sequences, indicating that the neural network is not saturated and suggesting that more trajectories can still be added, improving the performance. While compact, then, this map representation is also constant-size when new information is added. As stated before, compression comes in the sense of finding the smallest neural network architecture with that still performs well in the relocalisation problem. On the other hand, compression also comes in the sense that the map information increases when new information is received without increasing the map representation size given by a constant-size CNN architecture.

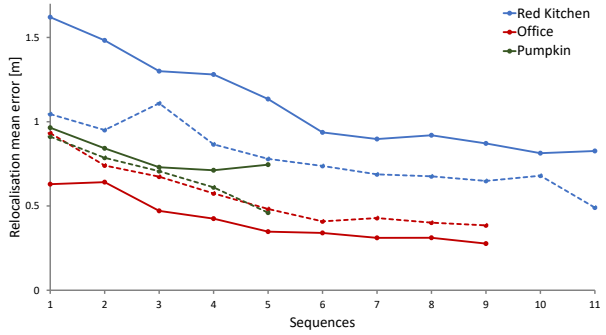


Fig. 9: Relocalisation mean error in several scenes from the 7-Scenes dataset [38] with respect to the number of training trajectories (one sequence is fixed for testing and the remaining are used for training; dotted lines indicates a different test sequence). The VGG-F is utilized for evaluation. While the number of training trajectories increases, the error decreases but the neural network size remains the same (the training only affects the weights).

Figure 13 shows some outputs for the Red Kitchen sequence where the relocalisation improves as more trajectories are used using the VGG-F fast architecture. There, it is also observed how the relocated cameras (blue points) are closer to the test sequence (green line) when more training sequences are present (red line).

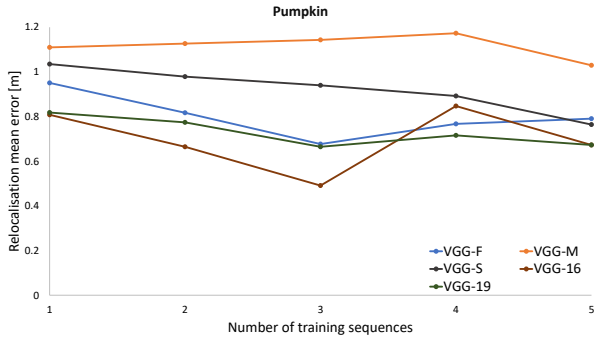


Fig. 10: Relocalisation mean error in the Pumpkin dataset [38] using several CNN architectures with respect to the number of sequences in the training process, except for one sequence selected for testing. No significant difference in the performance between the short (in particular VGG-F) and long architectures is observed and therefore short architectures can be used without sacrificing performance.

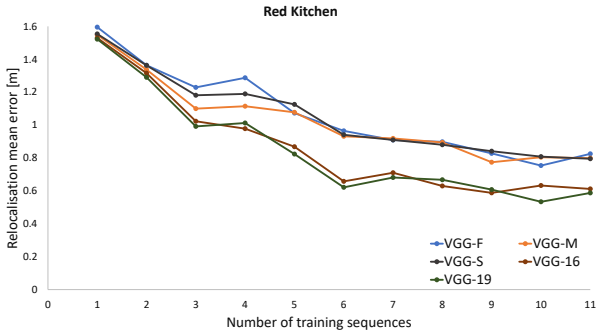


Fig. 11: Relocalisation mean error in the Red Kitchen dataset [38] as in Figure 10.

VI. Conclusions

We presented a first approach toward CNN map representation and compression for camera relocalisation. The response to different inputs and to different trajectories was studied. We first shown that for these kind of models, when few training data (some thousands samples) and training from scratch, the RGB images present the best performance compared with other types of data, as depth or 3D point clouds. Then, we presented the idea of map compression as the task of finding optimal CNN architectures where different architectures were evaluated: VGG-F, VGG-M, and VGG-S, all with 8 layers and different orders of complexity, VGG-16 (16 layers), and VGG-19 (19 layers). We observed that, when the amount of training data is reduced, the architecture plays a crucial role in the relocalisation performance; however, with a representative training set, the short and long architectures perform similarly. None o those architectures outperforms state-of-the-art techniques (e.g. ORB-SLAM); however, regression techniques like SCoRe Forests

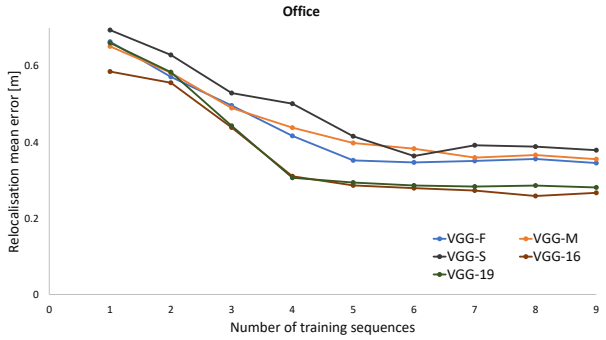
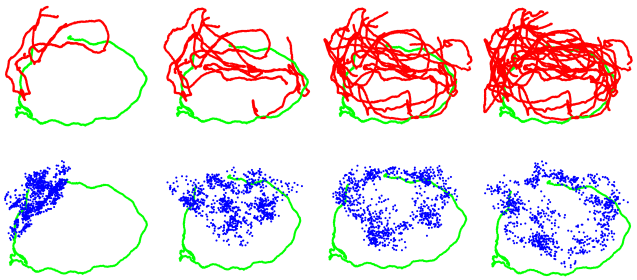


Fig. 12: Relocalisation mean error in the Office dataset [38] as in Figure 10 and Figure 11.



(a) (b) (c) (d)

Fig. 13: Relocalisation performance for the Red Kitchen sequence in the Microsoft-s 7-Scenes dataset [38] after training with a) two, b) five, c) eight, and d) eleven sequences using the VGG-F. Red lines indicate the training sequences and the green line is the test one; blue points are the output of the system.

show promising results for map representations as regression models.

On the other hand, we perform a study on the relocalisation performance with respect to the number of training sequences. We observed that the performance increases with the number of training sequences, as expected. However, in the context of map representation through neural networks, it means that the map accuracy can be improved without increasing the map size but only by adjusting its weights. This is important when a fixed amount of memory is available to store the map in the mapping and navigation process.

For future work we note that more complex relocalisation such as semantic or topological relocalisation were not explored here. One potential direction encouraged by these results is to train simpler networks for object recognition with labeled datasets, and use a second network that accepts semantic information as input for relocalisation. This kind of multi-network systems, where two complex systems interact to perform a single task are of interest to expand on the current work.

REFERENCES

[1] L. Contreras and W. Mayol-Cuevas, "Trajectory-driven point cloud compression techniques for visual SLAM," in *International Conference on Intelligent Robots and Systems*, IEEE, 2015.

[2] L. Contreras and W. Mayol-Cuevas, "O-POCO: Online POint cloud COmpression mapping for visual odometry and SLAM," in *International Conference on Robotics and Automation* , IEEE, 2017.

[3] R. Harvey, P. DiCaprio, and K. Heinemann, "A neural network architecture for general image recognition," *Lincoln Laboratory Journal*, vol. 4(2), p. 189207, 1991.

[4] M. J. Milford, G. F. Wyeth, and D. Prasser, "RatSLAM: a hippocampal model for simultaneous localization and mapping," IEEE, 2004.

[5] J. J. DiCarlo, D. Zoccolan, and N. C. Rust., "How Does the Brain Solve Visual Object Recognition?," *Neuron*, vol. 73(3), p. 415434, 2012.

[6] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR, abs/1206.5538*, vol. 1, 2012.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[9] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *British Machine Vision Conference*, 2011.

[10] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," in *British Machine Vision Conference*, 2014.

[11] P. Agrawal, J. Carreira, and J. Malik, "Learning to See by Moving," in *International Conference on Computer Vision*, 2015.

[12] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the Best Multi-Stage Architecture for Object Recognition?," in *International Conference on Computer Vision*.

[13] R. Garg, V. K. B. G, and I. D. Reid, "Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue," *CoRR*, vol. abs/1603.04992, 2016.

[14] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," *CoRR*, vol. abs/1406.2283, 2014.

[15] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2015.

[16] F. Liu, C. Shen, G. Lin, and I. D. Reid, "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields," *CoRR*, vol. abs/1502.07411, 2015.

[17] A. P. Gee and W. Mayol-Cuevas, "6D Relocalisation for RGBD Cameras Using Synthetic View Regression," in *British Machine Vision Conference*, September 2012.

[18] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," 2015.

[19] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen, "Vidloc: 6-dof video-clip relocalization," *CoRR*, vol. abs/1702.06521, 2017.

[20] G. Long, L. Kneip, J. M. Alvarez, and H. Li, "Learning Image Matching by Simply Watching Video," *CoRR*, vol. abs/1603.06041, 2016.

[21] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[22] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Incremental vision-based topological SLAM," in *International Conference on Intelligent Robots and Systems*, 2008.

[23] J. Lim, J. Frahm, and M. Pollefeys, "Online environment mapping using metric-topological maps," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1394–1408, 2012.

[24] M. Milford and G. Wyeth, "SeqSLAM : visual route-based navigation for sunny summer days and stormy winter nights," in *International Conference on Robotics and Automation* , pp. 1643–1649, IEEE, 2012.

[25] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. S. Kruthiventi, and R. V. Babu, "A Taxonomy of Deep Convolutional Neural Nets for Computer Vision," *Frontiers in Robotics and AI*, vol. 2, p. 36, 2016.

[26] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[27] G. James, T. Hastie, R. Tibshirani, and D. Witten, *An Introduction to Statistical Learning: With Applications in R*. Springer, second ed., 2014.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[29] J. Emer, V. Sze, and Y.-H. Chen, "Tutorial on Hardware Architectures for Deep Neural Networks," 2017. Available at `http://eyeriss.mit.edu/tutorial.html`. Accessed 01 Sep 2017.

[30] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," in *Conference on Computer Vision and Pattern Recognition*, pp. 512–519, IEEE Computer Society, 2014.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Conference on Computer Vision and Pattern Recognition*, 2009.

[32] A. Vedaldi and K. Lenc, "MatConvNet – Convolutional Neural Networks for MATLAB," in *Proceeding of the ACM International Conference on Multimedia*, 2015.

[33] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014.

[34] A. Kendall, M. Grimes, and R. Cipolla, "Research data supporting 'PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization': St Marys Church [dataset]," 2015. https://www.repository.cam.ac.uk/handle/1810/251294 (accessed February 2016).

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *International Conference on Intelligent Robots and Systems*, pp. 573–580, IEEE, 2012.

[36] 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 *Conference on Computer Vision and Pattern Recognition*, IEEE, 2013.

[37] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[38] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-Time RGB-D Camera Relocalization," in *International Symposium on Mixed and Augmented Reality*, IEEE, 2013.