Homography Estimation from Image Pairs with Hierarchical Convolutional Networks

Farzan Erlik Nowruzi, Robert Laganiere
Electrical Engineering and Computer Science
University of Ottawa
fnowruzi@uottawa.ca, laganier@eecs.uottawa.ca

Nathalie Japkowicz
Computer Science
American University
japkowic@american.edu

Abstract

In this paper, we introduce a hierarchy of twin convolutional regression networks to estimate the homography between a pair of images. In this framework, networks are stacked sequentially in order to reduce error bounds of the estimate. At every convolutional network module, features from each image are extracted independently, given a shared set of kernels, also known as Siamese network model. Later on in the process, they are merged together to estimate the homography. Further, we evaluate and compare effects of various training parameters in this context. We show that given the iterative nature of the framework, highly complicated models are not necessarily required, and high performance is achieved via hierarchical arrangement of simple models.

Effectiveness of the proposed method is shown through experiments on MSCOCO dataset, in which it significantly outperforms the state-of-the-art.

1. Introduction

In the recent years, modern deep learning models [19] have achieved a great success in many fields including image classification [17][27], object detection [14], scene segmentation [8], natural language processing [23] and many others. In the field of the computer vision, they have encountered significant success over traditional methods. One of the main differences between deep models and traditional methods is the automation of previously hand-crafted feature extraction that optimizes itself to specific data and task. This novelty led to an ever-growing amount of interest in using these methods in many research fields. During recent years, various architectures have been proposed to deal with specific problems, such as Convolutional Neural Networks (CNN) to extract image features, Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) to extract order and temporal information from the inputs [20].

Besides the various models and architectures being proposed almost on a daily basis, there are novelties regarding the ways that a model could be trained. These modifications aim at improving the convergence and speed up the exhaustive learning process, which can be of crucial importance. Such methods include regularization methods as Batch Normalization [16], Weight Normalization [28], or optimization methods like momentum optimizer [31], Adam optimizer [18].

Another hot topic among researchers and industrials is autonomous driving. Using deep methods in autonomous robotics and vehicles is currently focused on sensor data processing [10][3], planning [7], or end-to-end learning [4]. However, there are still many open problems such as odometry extraction, localization, mapping, or 3D model generation that have not been investigated thoroughly. Improving the effectiveness of deep methods on these tasks is essential.

All of these tasks at some point require a form of matching and regression. We therefore, study here the fundamental task of extracting deformation parameters from two observations. This is the building block of almost all tasks dealing with multiple observations and trying to build models from them. For example, when a camera calibration task is performed by using a planar rectangular grid in front of the camera, one must extract features from them. Then, using a regression module the homography is estimated and subsequently, extrinsic parameters between stereo cameras are calculated [33]. Similar ideas are applied for panoramic image generation [5]. To generate 3D models of the environment, all models use a similar frameworks to extract, localize and register features from various images to get relative poses information [2]. These methods later are used as building blocks in various fields, such as in autonomous driving, augmented reality, indoor robotic tasks and so on.

In this paper, we introduce a method that significantly improves accuracy of planar homography estimation from image pairs using deep convolutional neural networks. We show that a hierarchical framework can learn to relate im-

ages to each other and extract accurate results. The proposed model also benefits from real-time computational complexity, that makes it suitable for many tasks.

2. Literature Review

The basic approach to tackle a homography estimation is to use two sets of corresponding points in Direct Linear Transform (DLT) method. However, finding the corresponding set of points from images is not always an easy task. In this regard, there have been significant amount of research. Features such as SIFT [22] and ORB [26] are used to find the interest points, and employing a matching framework, point correspondences are achieved. Commonly, a RANSAC approach is applied on the correspondence set in order to avoid incorrect associations. And, after an iterative optimization process, the best estimate is chosen [5][24].

One major problem with such methods is their requirements for the hand-crafted features and exhaustive matching step. Deep models automate feature extraction and provide much stronger features than conventional approaches. Their superiority has been shown many times in various tasks [3][14][17][27]. Recently, there have been attempts to address matching problem with similar models. Flownet [13] targets optical flow estimation by employing a parallel convolutional network model to extract features from each image independently. A correlation layer is used to locally match extracted features against each other and aggregate them with responses. The expanded feature set is then used in further convolutional layers. Finally a refinement stage consisting of de-convolutions is used to map optical flow estimates back to the original image coordinates. Most recently, Flownet 2.0 [15] was introduced that is using Flownet models as building blocks to create a hierarchical framework to solve the same problem.

Similarly, [25] proposes an optical flow estimation method that employs a deep matching process with convolutions. From each image, patches of size 4×4 are extracted and described based on SIFT descriptors. Then, they are fed into a convolutional layer to produce correlation maps. This process is repeated and a pyramid of responses are created, which are used later to find local maximas and track them through the pyramids to get pixel correspondences.

Our paper is vastly inspired by the work of DeTone, Malisiewicz, and Rabinovich [11], where a deep neural network is devised to tackle the homography estimation task. To be compatible and comparable, we strictly followed their guidelines when developing the model and for the benchmark of it. As they proposed, a homography between two images are defined by relocation of a set of 4 points, also known as 4-point homography. Their model is based on the VGG's architecture [17] with 8 convolutional layers, a pooling layer after each 2 convolutions, and 2 fully connected layers with an L_2 loss function that results from the difference between predicted and true homography values. Their model starts with stacking up images in two channels and processing them together through the network. In contrast, the core of our model targets each input independently from the other. Motivation behind this approach is to have corresponding feature sets that will be merged in later stages, which has been shown in [9] to be more suitable for the tasks requiring feature matching.

Most of the existing deep models aim to solve the problem through a single, and in many cases, large model. Independent from how deep or wide the model is, an error margin will always be found, especially in tasks such as regression. This is due to the trade-off imposed on the model to fit the training data and also to be able to generalize. A well learned model should be able to minimize both errors in training and testing sets.

Based on concepts from iterative optimization approaches, boosting, or genetic models, we propose to stack the same model in a hierarchical manner such that first model takes an image pair and produces an estimation, along with a new image pair. The newly generated image pair has a smaller homography residual as the first model already estimated part of it. Next, a copy of the same model takes this data in and then provides a better approximation with an even smaller error bounds. Continuing this process will sequentially reduce error bounds until error margin is so low that there is nothing else to correct. As we will discuss later, in this situation, the train and test errors start to diverge, which indicates the model has entered the overfitting region. A similar idea was introduced by Carreira *et al.* in [6] for human pose estimation. In their work, the body part point coordinates are estimated using [32] and are iteratively updated to fit the human pose in a single image. In contrast to that work, we are introducing a visual warping step between each iteration to drive the model towards the target transformation, while using a smaller network. To the best of our knowledge, this framework has never been applied to the homography estimation task, which is a natural fit for the problem.

This framework produces mid-level results that are important in many real-world tasks. It provides a way to observe how system evolves thus preventing false behavior of the components. In addition, coarse-to-fine results could be used by other modules, thus reducing system latency. Modularity of this framework is another benefit. On a case-by-case basis, each model could be retrained, or replaced with a larger or smaller model, to fix the problem specific tasks. Further, training a small network is much faster and less cumbersome than a large one. Based on the required error bounds various combinations of networks can be coupled in the framework. We believe this framework could be applied to any problem that includes regression and/or optimization. In the Section 7, we will discuss how it can be adapted to

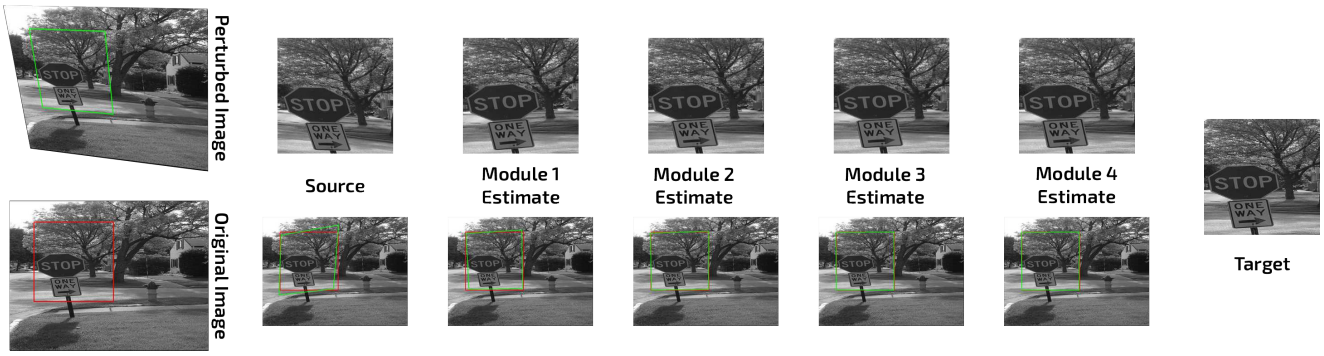

Figure 1. Sample output and the warping process. Source and target images are provided to the system. Due to shrinking error residual after each module, warped image is getting visually more similar to the target at each step.

the sensor odometry problem. Our model achieves real time performance on consumer available graphics cards that satisfies industrial requirements.

In essence, our proposed model takes the basics of the Siamese network model [9] that are shown to be more effective in extracting similarities between inputs using parallel layers with shared weights. We performed extensive testing and compared performance of multiple parameters for the model training.

3. Hierarchical Convolutional Network Model

In this section, we present our network architecture in detail and show ho to calculate the four point homography estimate from an image pair. This network is shown in Figure 2.

3.1. Network Architecture

Similar to [11], the main network in our method consists of 8 convolutional layers followed by two fully connected layers. In contrast to that model, the first 4 layers of our network are designed to process images in parallel. Input images are both normalized and each image is fed to one of the identical parallel layers. A filter with a kernel size of 3×3 is used to create features in 32 dimensions. At the end of parallel convolutional layers, two 32 dimensional feature vectors are concatenated along their third dimension, totaling in a 64 dimensional feature vector. Another 4 convolutional layers are applied on the merged feature map to conclude the convolutional layer. After each 2 convolutions a max pooling layer with strides of 2 is applied on feature output. To avoid over-fitting, commonly used drop-out [30] scheme with a drop probability of 0.5 is employed prior to passing the output to the fully connected layer. The two fully connected layers have a dimensionality of 1024 and 8 respectively in which the latter is the flattened output homography estimate. Instances are processed in batches of size 64.

Rectified Linear Units (ReLU) are used in each neurons fire module to add non-linearity to the feature outputs. Both the momentum and Adam optimizers have been tested for the proposed system. For momentum optimizer, as proposed in [11], a momentum value of 0.9 with piecewise constant learning rate of 0.005 with a decay factor of 0.1 applied in the iterations 30000 and 55000. For Adam optimizer, we kept the learning rate and decay factor as the same as momentum optimizer with the epsilon set to 0.1. Total number of epochs per model is set to 75000 which roughly equates to 10 rounds over the dataset.

To evaluate and train the network, an L_2-norm on the difference of the target H_i and estimate H_i^* is used as the loss function.

$$loss = \frac{1}{2} \sum ||H_i - H_i^*||^2 \tag{1}$$

3.2. Hierarchical Model

A hierarchical model arranges neural network modules in a stacked manner and successively reduces estimation error bounds. In each module, an approximate H_i^* of targeted homography H_i is generated. To prepare data for the next module, a new target homography H_i is calculated by simply subtracting estimate from the target of processed module, and a new image pair (I_i^1, I_i^2) is generated by warping one image in the pair using the new target.

$$\begin{aligned} H_i &= H_{i-1} - H_{i-1}^* \\ (I_i^1, I_i^2) &= (I_{i-1}^1 * H_i, I_{i-1}^2) \end{aligned} \tag{2}$$

It is worth noting that the dynamic range of H_i at each iteration is smaller than the previous iteration. In other words, the new target is actually the error residual of estimates calculated in the module.

$$max(H_i) - min(H_i) < max(H_{i-1}) - min(H_{i-1}) \tag{3}$$

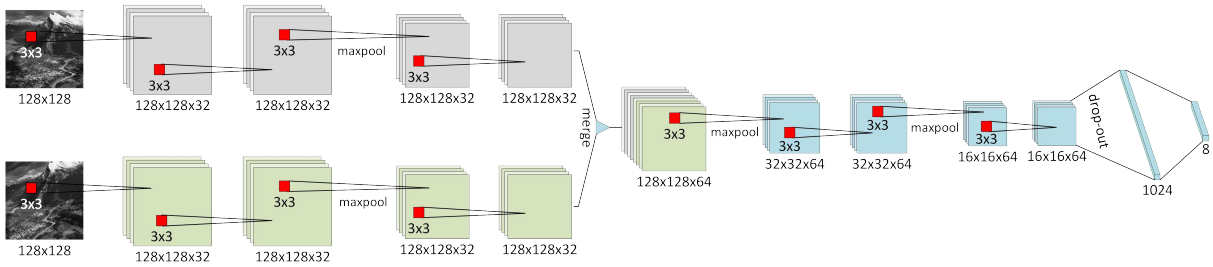

Figure 2. Twin convolutional neural network architecture used as the core module.

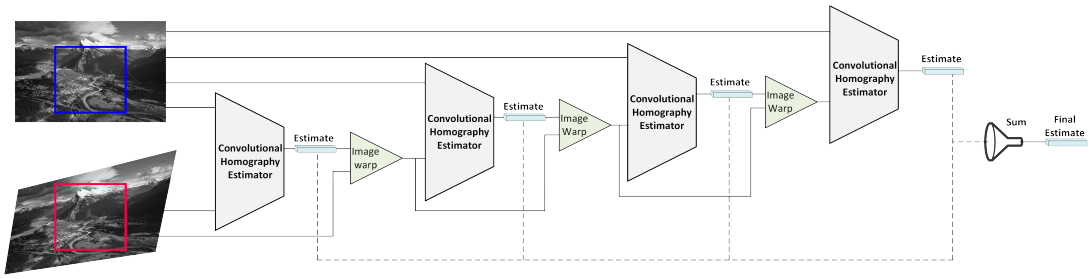

Figure 3. Hierarchical Model Framework. Each of the convolutional homography estimator modules consists of twin convolutional neural networks that perform homography estimation on an input image pair.

As explained, each module estimates the residual value of the overall homography. To calculate the final result that can directly transform one image to another, all 4-point estimates of the successive modules are added up together. Early modules tend to have larger values than later modules in the framework, since the residuals have lower dynamic ranges.

$$H^* = \sum_{i=0..n} H^*_{i-1} \tag{4}$$

Overall, this approach is similar to an iterative optimization scheme. Each module in this hierarchy is trained on the outputs of the previous module in order to produce a smaller error residual. This is in essence similar to Boosting methods [29] that divide feature space into chunks and assign them to various weak learners. After a learner is done, another weak learner starts learning based on the features assigned to it, and this process iteratively continues until all weak learners create a strong learner eventually. In contrast, our proposed framework is not dividing the feature space. Instead, it trains the next model only on the residual of the previous module. Boosting also uses a weighting mechanism. In our model, weighting scheme can be considered as an implicit function, as each module provides a smaller valued estimate. Figure 3 depicts the proposed hierarchical framework.

Warping with predicted homography values for each module results in a visually more similar patch pair. This can be visualized as a geometric morphing process that takes one image and successively makes it to look alike the other, shown in Figure 1.

One relatively important benefit of modular design of the hierarchy is that it is not restricted to the use of one specific network architecture. At each level, depending on the error boundaries, data statistics, accuracy and speed requirements, different networks could be used.

4. Dataset

We are using the Microsoft Common Objects in Context (*MSCOCO*) 2014 dataset [21]. First, all the images are converted to gray-scale and are down-sampled to a resolution of 320×240. To prepare training and test samples, they are divided in two groups. A training set with 77870 and a test set of 5000 base images. Later, five samples from each base image is generated in order to increase the dataset size. To achieve this, five random rectangles of size 128×128, exclude a boundary region of 32 pixels, is chosen from each base image. A random perturbation in the range of 32 pixels is added to each corner point of the rectangles. This provides us with the target 4-point homography values. Target homography is used with the OpenCV library to warp original images. Finally, original corner point coordinates are used within the warped images to extract the warped patches. The pair of chosen patches along with the values of random perturbations (4-point homography) are fed as inputs to the system.

Model	Parallel layer dimensions	Merged layer dimensions
64_dims	32×4	64×4
128_dims	64×4	128×4
256_dims deep	64×4	$128 \times 4 + 256 \times 4$

Table 1. Model names and parameters.

5. Implementation

We have realized a Tensorflow[1] [1] implementation of the proposed module. In addition, we have introduced a slightly different implementation of the weight normalization paradigm. To train we have used local machines and Google Cloud Machine Learning Engine [2], but all the tests were performed on a local machine equipped with an Intel Core-i7 CPU at 4.0Ghz, 32GB of memory, and an Nvidia Titan Xp graphics card with 12GB of available memory.

During the test time, we achieved an average processing speeds of 3 milli-seconds per network. When the same neural network model is used in each module, the overall computational complexity is given by,

$$d_e = (l_m + l_w) \times n \tag{5}$$

where d_e is the end-to-end delay, l_m average latency at each module, l_w over-head of warping to generate a new pair, and n number of modules used in the framework. The real-time processing speed and flexibility of the proposed model satisfies the requirements most of potential applications.

6. Experiments

In this section, we report training and test performance of our method under varying conditions. First, we show how our framework ranks up against state-of-the-art approaches. Then, the results of single deep modules are compared. For this purpose, a series of models with different layer widths as described in table 1 are implemented. The performance of the proposed model against occlusions is demonstrated in the third subsection. Then, effects of various parameters are evaluated. And finally, we discuss how the optimal number of iterative modules is chosen.

6.1. Accuracy Results

First, we experimentally compared the corner error of our hierarchical convolutional network with two other approaches. The corner error is achieved by calculating L_2 distance between target and estimate corner locations and averaging them over 4 corners. The approaches used for comparison consist of a traditional one and a convolutional one. The selected traditional approach is based on ORB key-point matching followed by a robust RANSAC homography estimation scheme. The reference deep convolutional approach is the HomographyNet by [11].

[1] https://www.tensorflow.org/

[2] https://cloud.google.com/ml-engine/

Method	Pixel Error	Error reduction
ORB+RANSAC	11.7	—
HomographyNet[11],	9.2	21.37%
Proposed	**3.91**	**66.58%**

Table 2. Test Results. Comparison of our method with a traditional feature-based (ORB key-points) robust estimation scheme and homographyNet, both reported in [11], against our proposed approach.

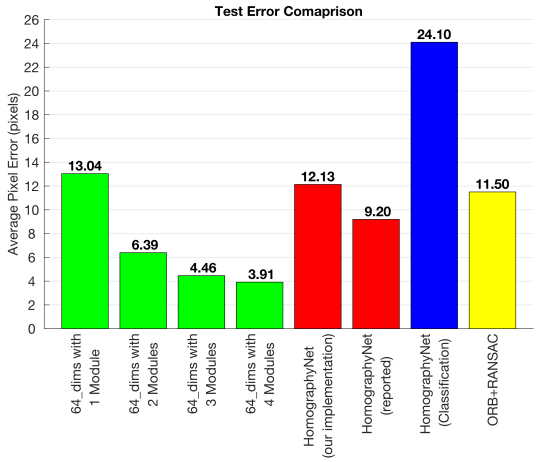

Figure 4. Test Error. Corner pixel error comparison of various methods.

Using our network, we report in Figure 4 the progressive improvement that results by iteratively stacking several networks. This is compared with the other two approaches. HomographyNet-classification provides worst results in this case. It's lower performance is attributed to the quantization performed on the range of values to extract bins for each value. Our net result is a drop in pixel error from 12 pixels to less than 4 pixels in the case of our hierarchical network. The comparable convolutional HomographyNet only achieves an error of 9 pixels.

For the sake of simplicity, we kept the same model at each module of training. However, multiple combinations are also possible, which will be discussed in Section 7. As more models are stacked, higher accuracy is achieved. This is due to the fact that each model is trained to reduce error bounds of the previous model. There are down sides for having more models in the stack. One is introduction of an end-to-end delay. However, the high speed of our network largely compensates for the introduced delay.

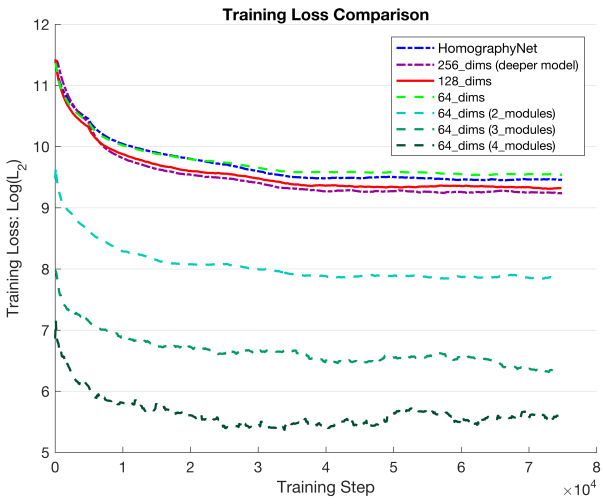

Figure 5. Training curves showing the learning process for implemented models in $log(L_2)$ scale.

6.2. Single Module Comparison

Another conducted experiment targets the width of the network. To address the learning process, we report the L_2 training loss curves in *logaritmic* scale at Figure 5. As expected, the narrower the network, the lower the performance. Using wider network configurations only slightly increases the performance with an additional cost on memory requirement. We have also observed that any network narrower than *64_dims* provides notably worse results, and anything wider than *128_dims* produces negligible increase in performance, while significantly increasing model size and training time. In fact, stacking a simple module such as *64_dims* twice in the hierarchy is capable of providing significantly better results than any of other methods.

6.3. Occlusion Analysis

In this section the performance of proposed system is benchmarked against occlusion. To simulate occlusion, the *Caltech-101* [12] dataset is used. Each image in this dataset is resized to patches of size 32×32 and 64×64. Then, we randomly augment *MSCOCO* dataset with generated occlusion patches from *Caltech-101* and create three datasets. First dataset is only augmented with 32×32 patches representing an occlusion ratio of 6.25%. Second one is created using 64×64 patches resulting in an occlusion ratio of 25%. And the final dataset is a fair mix of no-occlusion, 6.25%, and 25% occluded images.

First, we test the model that has never encountered any disturbances during training, against the noisy data. Second, we retrain the model with the occlusion dataset and test again for all the scenarios. The result of these tests are shown in Table 3.

Train \ Test	No noise	6.25% noise	25% noise	mix of all
No noise	3.91	7.5	12.56	8.21
Mix of all	4.58	6.41	9.04	6.74

Table 3. Effect of occlusion on the performance. The left column shows the datasets used for training. The pixel-wise corner accuracy of each test dataset is noted underneath the corresponding column header.

6.4. Parametric Evaluations

6.4.1 Deeper model evaluation

As shown in Figure 5, a deeper network increases the performance but by only a small margin, while significantly increasing model complexity observed margin is not satisfactory. The reason for this is the fact that extracted features are already capturing variations to the full extent as they can. While in the hierarchical usage, warping function introduces a sort of geometric information that convolutional models are having hard time to capture from a single input set. The fact that more pooling layers are applied leads to the loss of spatial information.

To justify these claims, we have implemented a model with 4 extra convolutional layers. This results in a model with 4 twin layers, 8 convolutional layers with a pooling layer after each 2 convolutional layers, and 2 fully connected layers. For the newly added convolutional layers a feature dimension size of 256 is assigned. Results of this comparison are also shown in Figure 5, which confirms our hypothesis stating that more parameters and layers aren't assistive in this specific case. However, when using the second module in the hierarchy, our proposed framework clearly outperforms the competition.

6.4.2 Regularizers and Optimizers

Batch normalization aims at providing a faster learning convergence by reducing the effect of the covariate shift in the data using statistics from the batches. Weight norm has also a similar goal, however instead of normalizing the inputs of layer, normalization is applied on the layer weights. We have compared these two approaches and our results show that batch normalization has better performance for this task. Weight norm results in a slower convergence and under-fitting in the model. The comparative experiment is shown in Figure 6. Another important aspect to note is that having a regularization function is vital in training phase and failing to use any exhibited devastating effects on the learning performance.

We have concluded that both methods have a very similar performance in training and their differences are negligible. Results of this comparison are shown in Figure 6.

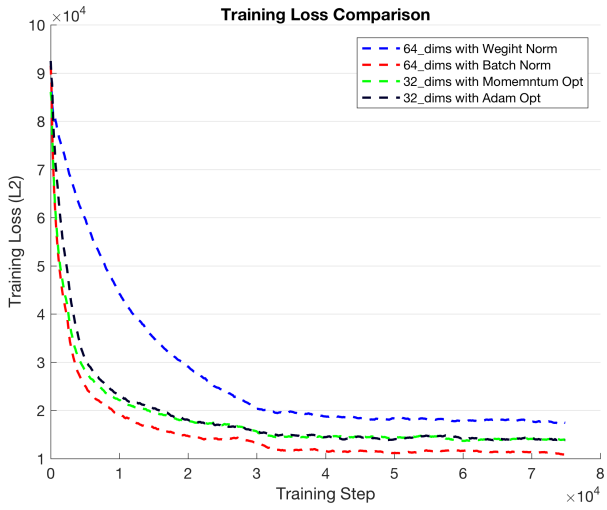

Figure 6. Parametric Evaluation. Batch normalization is benchmarked against Weight normalization. Effectiveness of batch normalization is shown for homography estimation. Performance of Momentum optimizer and Adam optimizer are very similar.

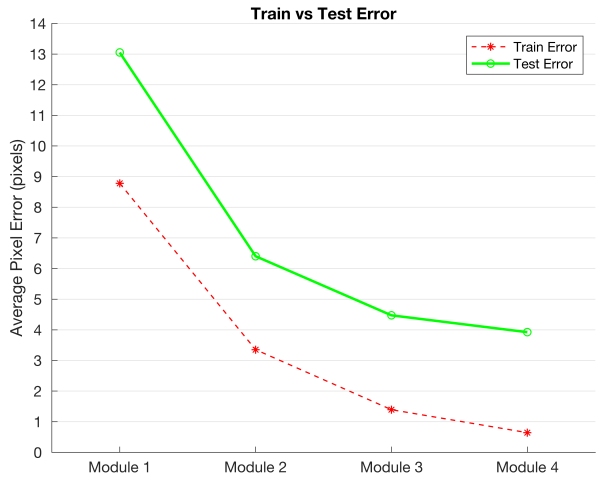

Figure 7. Hierarchy size evaluation: training error vs. test error.

6.5. Optimal Number of Modules

Finding optimal number of modules to use depends on complexity of the task and on the complexity of the module to be employed. As it is shown in Figure 7, for the 64_dims model, after the fourth module, error residual is too shallow and it is very difficult for the model to learn. This is also showed in 5 where the training loss for the forth module has almost plateaued.

As in all machine learning approaches, a zero error value can not be achieved by chaining many modules together. This is due to the fact that after a few modules error residual gets very small and it becomes extensively difficult for it to learn. This results in test error that starts to diverge from training error, indicating that an over-fitting phenomenon is happening in the module.

7. Conclusion

In this paper, we have proposed a hierarchical model to use convolutional neural networks in order to target homography estimation. We showed that with our simple hierarchical model, results are significantly better than the state-of-the-art. This paper proves that deep neural networks are capable of effectively learning to regress homography from image pairs. Reduced parameter space entails faster and more manageable implementations. In our architecture, twin modules are unaware of each other until the merge function happens. Adding correlation features between parallel layers could potentially lead to better solutions. In addition to stacking, parallel modules could also be used in this hierarchy. The significance of our results is convincing enough to adapt this approach to further tasks such as odometry estimation, which are currently dominated by conventional approaches. Adapting our model to odometry estimation is ultimately straightforward. However, in order to handle the complexity of odometry, multiple measures must be taken. Augmenting model to use semantic information could provide robustness against outliers such as dynamic objects. To capture a wider range of transformational cues, a better convolutional model needs to be designed. And finally, to handle the error propagation through consecutive frames, a sequential learning models must be utilized.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, and S. Ghemawat. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In *arXiv preprint arXiv:1603.04467*, 2016. 5

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Communications of the ACM*, pages 105–112, 2011. 1

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *arXiv preprint arXiv:1511.00561*, 2015. 1, 2

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, and X. Zhang. End to end learning for self-driving cars. In *arXiv preprint arXiv:1604.07316*, 2016. 1

[5] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. In *International journal of computer vision*, pages 59–73, 2007. 1, 2

[6] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *IEEE Con-*

ference on Computer Vision and Pattern Recognition, pages 4733–4742, 2016. 2

[7] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *IEEE International Conference on Computer Vision*, pages 2722–2730, 2015. 1

[8] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *arXiv preprint arXiv:1412.7062*, 2014. 1

[9] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 539–546, 2005. 2, 3

[10] M. Cordts, T. Rehfeld, L. Schneider, D. Pfeiffer, M. Enzweiler, S. Roth, M. Pollefeys, and U. Franke. The stixel world: A medium-level representation of traffic scenes. In *Image and Vision Computing*, 2017. 1

[11] D. DeTone, T. Malisiewicz, and A. Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016. 2, 3, 5

[12] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Generative-Model Based Vision*, 2004. 6

[13] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *arXiv preprint arXiv:1504.06852*, 2015. 2

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1, 2

[15] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *arXiv preprint arXiv:1612.01925*, 2016. 2

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1

[17] S. Karen and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv preprint arXiv:1409.1556*, 2014. 1, 2

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980.*, 2014. 1

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. In *Nature*, pages 436–444, 2015. 1

[21] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer. 4

[22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International journal of computer vision*, pages 91–110, 2004. 2

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *arXiv preprint arXiv:1301.3781*, 2013. 1

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. In *IEEE Transactions on Robotics*, pages 1147–1163, 2015. 2

[25] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Deepmatching: Hierarchical deformable dense matching. In *International Journal of Computer Vision*, pages 300–323, 2016. 2

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *IEEE International Conference In Computer Vision*, pages 2564–2571, 2011. 2

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and A. C. Berg. Imagenet large scale visual recognition challenge. In *International Journal of Computer Vision*, pages 211–252, 2015. 1, 2

[28] T. Salimans and D. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, number 901, 2016. 1

[29] R. E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer New York, 2003. 4

[30] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1929-1958), 2014. 3

[31] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013. 1

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition*, 2015. 2

[33] M. Warren, D. McKinnon, and B. Upcroft. Online calibration of stereo rigs for long-term autonomy. In *IEEE International Conference on Robotics and Automation*, pages 3692–3698, 2013. 1