Fully Convolutional Networks for Homography Estimation

Darwin Bautista
Electrical and Electronics Engineering Institute
University of the Philippines Diliman
Email: darwin.bautista@eee.upd.edu.ph

Richard Guinto
Samsung Electronics Philippines Corporation
Email: richard.guinto@gmail.com

***Abstract*—In this work, we study the effectiveness of fully convolutional networks in the task of homography estimation. State-of-the-art models use too many parameters to be useful in resource-constrained environments, such as in embedded applications. Thus, we propose a smaller, deeper, and fully convolutional network based on MobileNet. We leverage a self-supervised training method and a composite loss function. Based on test results on our COCO-derived dataset, our proposed model significantly outperforms the state-of-the-art.**

I. Introduction

Homography is a transformation of coordinates from one image to another, normally expressed in a 3×3 matrix. Its many practical applications include image rectification, stitching, and estimation of camera motion, which can be used for robot navigation (e.g. monocular SLAM). Estimating this homography matrix is one of the fundamental problems in computer vision.

Classical feature-based approaches to homography estimation consists of a two-stage process: feature point detection and homography estimation. Feature points are extracted by using a feature detector such as Scale Invariant Feature Transform (SIFT), or Oriented FAST and Rotated BRIEF (ORB). Due to the limitations of current hand-engineered feature detectors (e.g. presence of outliers), an overcomplete set of feature points is extracted and used in the next stage. Robust homography estimation is then performed by employing the Random Sample Consensus (RANSAC) algorithm.

Recent breakthroughs in computer vision were mainly driven by rapid advances in deep learning. Image classification, object detection, and semantic segmentation are some of the tasks in which deep learning models achieve state-of-the-art performance [1, 2, 3, 4, 5]. More recently, deep learning models have also been developed for homography estimation. DeTone et al. [6] demonstrated and pioneered the use of the four-point homography parameterization [7] in a deep learning model. Nguyen and Chen et al. [8] extended the work of DeTone et al. by employing a Spatial Transformer Network (STN) [9] to make the training self-supervised, thereby allowing the model to be used in a wider context. Both prior work used VGGNet [10] as the backbone CNN for feature detection, and dense layers for the regression of the homography parameters.

In this work, we further improve the homography estimation model by using a lightweight backbone CNN (MobileNet [11]) and making the model fully convolutional, thereby allowing our model to be used in embedded applications. A fully convolutional network (FCN) [5] makes more sense in this case because convolutional layers preserve spatial context, which are lost when dense layers are used (because the feature maps need to be flattened).

To summarize, our key contributions are as follows:

1) Reduced model footprint in terms of number of parameters.
2) Demonstrate the effectiveness of FCNs in homography estimation

We implement our model in Keras [12] using its Tensorflow [13] backend.

II. Related Work

In this section, we focus only on the reent deep learning work on homography estimation.

DeTone et al. [6] proposed a supervised deep learning model for estimating the homography given a pair of 128×128 images. They demonstrated the feasibility of using the 4-point parameterization of the homography [7] in training both a regression-based and a classification-based estimator. Since there are no publicly available datasets for homography estimation, they also proposed a method for generating a synthetic dataset for homography estimation. Their results show that the deep learning model outperformed the classical approaches by a significant margin. However, their model, which is based on VGGNet, use 34 million parameters which can be too large for embedded applications.

Nguyen and Chen et al. [8] extended the work of DeTone et al. by making the training self-supervised, i.e. the ground truth homography is not used in training the model. They created a customized Spatial Transformer Network [9] which accepts the estimated homography to reproject the first image onto the second one which allowed them to use a pixel-wise photometric (L1) loss function in training the model. The authors focused on making the training self-supervised and did not improve upon the original model. The choice of loss function (L1) also seem inadequate, thus in this work, we propose to use a combination of L2 and Structural Dissimilarity (DSSIM) losses.

Nowruzi et al. proposed a hierarchical siamese network for estimating the homography [14]. The two input images each indepedently flow through a siamese network, then are merged together at some point before going through another series of convolutions. The homography estimate is iteratively refined at each level of the hierarchy. While their approach is sound, we argue that we can come up with a faster, more efficient model if we process the two images together, like in the original approach of DeTone et al. Thus, in this work, we leverage the MobileNet architecture which makes extensive use of Depthwise Separable Convolutions, making the convolutions more parameter-efficient.

III. Methodology

A. Dataset

We follow the method of DeTone et al. for dataset generation and use the same parameters as well. That is, all images are scaled to 320×240 pixels and are converted to grayscale. 128×128 pixel patches are used, and the corner offset is sampled from a uniform distribution of [-32, 32] pixels. We use images from the COCO [15] 2017 *test* and *val* datasets to generate 12 samples per image, resulting in 550,000 samples total. We use 500,000 samples for training and set aside the remaining 50,000 samples for both validation and testing.

B. Homography Estimation Model

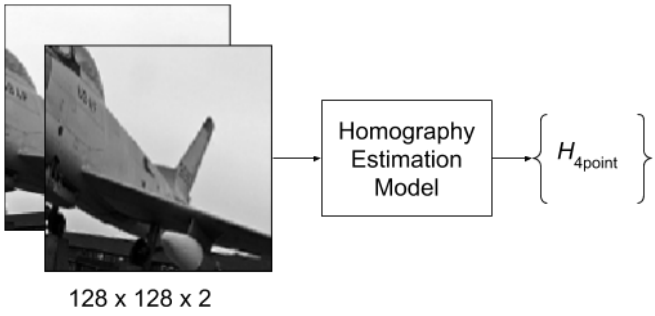


Fig. 1. Two 128×128 pixel grayscale images are stacked together channel-wise to produce the 128×128×2 input to the model. The output is the 4-point parameterized homography.

Our proposed homography estimation model is fully convolutional and is based on the MobileNet architecture. It uses the same inputs and produces the same outputs as the original model proposed by DeTone et al., as seen in Fig. 1. Given an image pair $I^A(x)$ and $I^B(x)$, the goal of our network model is to output the $\hat{H}_{4pt}$ that defines the homography transformation $\mathscr{H}(x_i)$. We do this by cropping two patches P^A and P^B of size 128×128 pixels from the image pairs I^A and I^B respectively, and using these as inputs to our model. The listing in Table I shows the layer descriptions of the model.

In order to train our model in a self-supervised fashion, we need a way of applying the estimated homography onto the original image, I^A, to transform it to $\hat{I}^B$, which we can then compare with the original I^B. We would need two auxiliary layers: the Direct Linear Transform (DLT) layer and a Spatial Transformer layer. The DLT layer is responsible for transforming $\hat{H}_{4pt}$, the 4-point homography output of the model, into $\mathscr{H}(x_i)$, the 3×3 matrix parameterization, which is what we need to be able to compute point-to-point correspondences between images. On the other hand, the Spatial Transformer layer is for applying the estimated homography, $\mathscr{H}(x_i)$, on the original image, I^A, to obtain $\hat{I}^B$. The details on these layers will be tackled in the following subsections.

With the two auxiliary layers in place, we can now reframe the homography regression task as an image reconstruction task, thereby allowing us to use the input images themselves as the training targets, as shown in Fig. 2.

TABLE I
Layer description of the model. Conv are vanilla convolutions while Conv dw are depthwise separable convolutions. We use batch normalization between layers and ReLU activation for all but the last Conv layer.

Type / Stride	Filter Shape	Input Size
Conv / s2	3×3×3×32	128×128×2
Conv dw / s1	3×3×32 dw	64×64×32
Conv / s1	1×1×32×64	64×64×32
Conv dw / s2	3×3×64 dw	64×64×64
Conv / s1	1×1×64×128	32×32×64
Conv dw / s1	3×3×128 dw	32×32×128
Conv / s1	1×1×128×128	32×32×128
Conv dw / s2	3×3×128 dw	32×32×128
Conv / s1	1×1×128×256	16×16×128
Conv dw / s1	3×3×256 dw	16×16×256
Conv / s1	1×1×256×256	16×16×256
Conv dw / s2	3×3×256 dw	16×16×256
Conv / s1	1×1×256×512	8×8×256
5 × Conv dw / s1	3×3×512 dw	8×8×512
5 × Conv / s1	1×1×512×512	8×8×512
Conv dw / s2	3×3×512 dw	8×8×512
Conv / s1	1×1×512×1024	4×4×512
Conv dw / s2	3×3×1024 dw	4×4×1024
Conv / s1	1×1×1024×1024	4×4×1024
Conv / s1	4×4×1024×8	4×4×1024

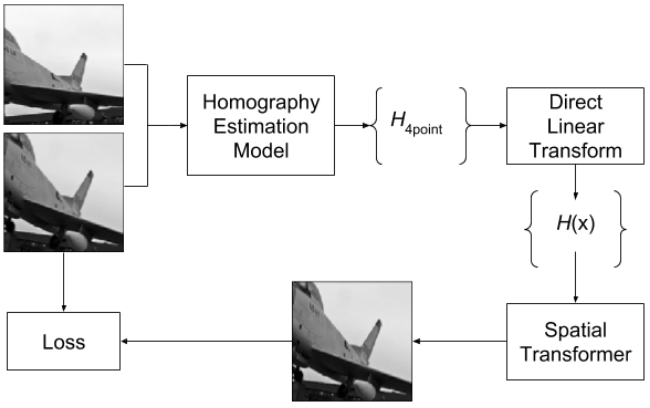


Fig. 2. Overview of training pipeline. We reframe the problem as an image reconstruction problem, similar to how autoencoders are trained. In addition to the image patches, we also use the full 320×240 source image as well as the patch corners.

For the loss, we use an equally weighted combination of pixel-wise Mean Squared Error (MSE) and Structural Dissimilarity (DSSIM), as shown in Eq. 4.

$$L_{MSE} = \frac{1}{n}\sum_{i}^{n}(P^A(\mathscr{H}(x_i)) - P^B)^2 \tag{1}$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2}$$

$$L_{DSSIM} = \frac{1 - SSIM(P^A, P^B)}{2} \tag{3}$$

$$L_{TOTAL} = L_{MSE} + L_{DSSIM} \tag{4}$$

Where μ_x is the average of x, σ_x^2 is the variance of x, σ_{xy} is the covariance of x and y, $c_1 = (k1L)^2$ and $c_2 = (k2L)^2$, L is the dynamic range of the pixel values (typically 255), $k1 = 0.01$, and $k2 = 0.03$ by default.

C. Direct Linear Transform Layer

Given a set of four 2D point correspondences $x_i \longleftrightarrow x_i'$ we can determine the transformation matrix H that satisfies the equation $x_i' = Hx_i$. However, for this equation involving homogeneous vectors, the 3-vector x_i' and Hx_i may differ in magnitude by a non-zero scale factor. The vector cross product form of this equation can be expressed with the equation $x_i' Hx_i = 0$ which allows us to derived H with a simple linear solution. Setting the j-th row of the matrix H to be denoted as h^{jT}, then:

$$Hx_i = \begin{pmatrix} h^{1T}x_i \\ h^{2T}x_i \\ h^{3T}x_i \end{pmatrix}$$

With $\boldsymbol{x}'_i = (x_i', y_i', w_i')^T$, the cross product may then be written as

$$\begin{bmatrix} \mathbf{0}^T & -w_i'\boldsymbol{x}_i^T & y_i'\boldsymbol{x}_i^T \\ w_i'\boldsymbol{x}_i^T & \mathbf{0}^T & -x_i'\boldsymbol{x}_i^T \\ -y_i'\boldsymbol{x}_i^T & x_i'\boldsymbol{x}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0$$

In this equation, only two of them are linearly independent, thus, the 3rd equation can be omitted to solve for H. In this case, the equation above can be reduced to the form:

$$\begin{bmatrix} \mathbf{0}^T & -w_i'\boldsymbol{x}_i^T & y_i'\boldsymbol{x}_i^T \\ w_i'\boldsymbol{x}_i^T & \mathbf{0}^T & -x_i'\boldsymbol{x}_i^T \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0$$

This can be written as $A_i h = 0$ where A_i is a 2×9 matrix. With the assumption that $x_i = (u_i, v_i, 1)$, then the equation above can be rewritten in the form of $\hat{A}_i\hat{h} = b_i$ for $i = 1, 2, 3, 4$ correspondence points. $\hat{A}_i$ is the 2×8 matrix of the matrix A_i

$$\hat{\boldsymbol{A}}_{\boldsymbol{i}} = \begin{bmatrix} 0 & 0 & 0 & -u_i & -v_i & -1 & v_i'u_i & v_i'v_i \\ u_i & v_i & 1 & 0 & 0 & 0 & -u_i'u_i & -u_i'v_i \end{bmatrix}$$

and b_i is the 2-dim vector representing the last column of $\boldsymbol{A_i}$ subtracted from both sides of the equation, $\boldsymbol{b_i} = [-v_i', u_i']^T$, and $\hat{\boldsymbol{h}}$ as the first 8 elements of vector $\boldsymbol{h}$. The equation $\boldsymbol{A_i}\hat{\boldsymbol{h}} = \boldsymbol{b_i}$ is differentiable and can solve $\hat{\boldsymbol{h}}$ by computing the pseudo inverse of $\hat{\boldsymbol{A}}_{\boldsymbol{i}}$.

D. Spatial Transformer Layer

This layer performs the homography transformation of an input image I^A using the estimated 3×3 homography matrix derived from the DLT layer.

Inverse warping is performed to avoid holes in the warped image using the following 3 steps:

1) Computing the normalized inverse of the homography matrix
2) Creating a Parameterized Sampling Grid Generator (PSGG)
3) Performing a Differentiable Sampling (DS)

The first step is simply computing the inverse of $\hat{H}$ and with the normalized height and width coordinates of the images I^A and I^B such that $-1 \leq u_i, v_i \leq 1$and $-1 \leq u_i', v_i' \leq 1$. The inverse $\hat{\boldsymbol{H}}_{inv}$ can be computed as follows:

$$\hat{\boldsymbol{H}}_{inv} = M^{-1}\hat{H}^{-1}M$$

where

$$M = \begin{bmatrix} W'/2 & 0 & W'/2 \\ 0 & H'/2 & H'/2 \\ 0 & 0 & 1 \end{bmatrix}$$

The second step is to perform the inverse homography of each pixel in the second image I_B to provide the pixel mapping to the first image I_A with the following equation:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathscr{H}_{inv}(G_i) = \hat{\boldsymbol{H}}_{inv} \begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix}$$

The 3rd step is to produce a sample warped image V of size H'×W' with C channels, where $V(x_i) = I^A(\mathscr{H}(x_i))$. Using bilinear interpolation, the equation will be:

$$V_i^c = \sum_{n}^{H}\sum_{m}^{W} I_{nm}^c max(0, 1 - |u_i - m|)max(0, 1 - |v_i - n|)$$

This equation is differentiable with respect to I and G, and thus, can be used for the backpropagation of the loss functions.

IV. Results

We train our model for about 180,000 iterations at a batch size of 64 using the Adam optimizer with Nesterov momentum. On a test set of 50,000 samples, we get the results in Fig. 3 which shows that our model improves upon the state-of-the-art by a significant margin.

In Fig. 4, we visualize the predicted homography using the square box we used to crop the image samples. We see that the green box for our model's output (Fig. 4b) is much closer to the ground truth box in blue.

In Fig. 5, we visualize the predicted homography by using it to warp the first input image in Fig. 5a. We expect the warped version to look similar to the second input image in

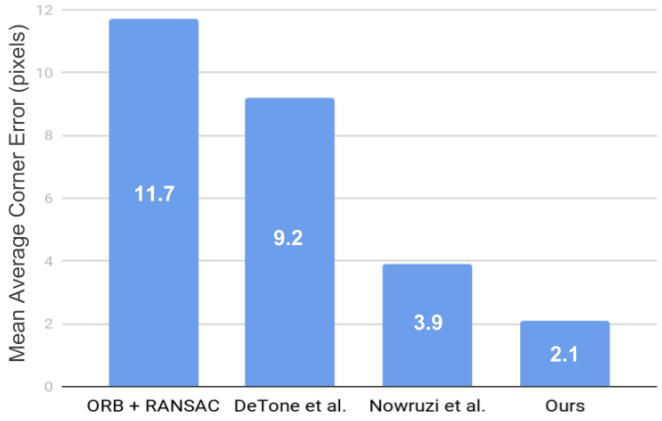
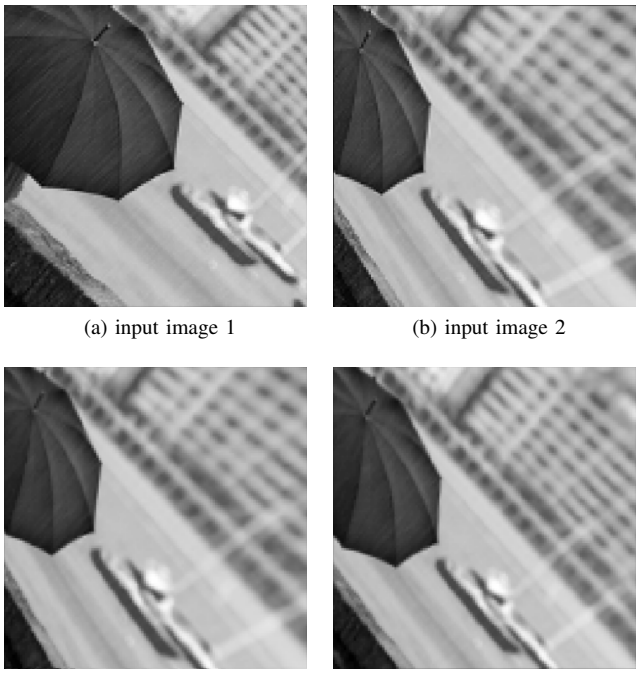


Fig. 3. Our model achieves a much lower Mean Average Corner Error than the state-of-the-art.

(a) DeTone et al. (b) ours

Fig. 4. Visualization of the predicted homography. The blue box corresponds to the ground truth while the green boxes correspond to the predictions.



(a) input image 1 (b) input image 2

(c) warped image 1 (ours) (d) warped image 1 (DeTone et al.)

Fig. 5. The estimated homography being applied on input image 1 to try to reproject it onto input image 2's plane.

Fig. 5b. In this case, both models estimate a homography near the ground truth, thus making the warped images look similar to the second input image. However, upon closer inspection, we see that Fig. 5c is actually closer to the ground truth than Fig. 5d.

V. Conclusions

In this work, we have shown the effectiveness of using a smaller but deeper fully convolutional network for homography estimation. We have also shown that a combination of MSE and Structural Dissimilarity objective functions are suitable for a self-supervised training setup. Our approach achieves higher accuracy while using much fewer parameters (3.4M vs 34M in the VGGNet-based models). We plan to extend our future work by benchmarking our results with real world datasets such as the aerial dataset performed by [8].

References

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

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[6] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.

[7] S. Baker, A. Datta, and T. Kanade, "Parameterizing homographies," *Technical Report CMU-RI-TR-06-11*, 2006.

[8] T. Nguyen, S. W. Chen, S. Skandan, C. J. Taylor, and V. Kumar, "Unsupervised deep homography: A fast and robust homography estimation model," *IEEE Robotics and Automation Letters*, 2018.

[9] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko,

W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[12] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning."

[14] F. E. Nowruzi, R. Laganiere, and N. Japkowicz, "Homography estimation from image pairs with hierarchical convolutional networks."

[15] 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*. Springer, 2014, pp. 740–755.