Rethinking Planar Homography Estimation Using Perspective Fields

Rui Zeng[0000−0003−0155−1288], Simon Denman[0000−0002−0983−5480], Sridha Sridharan[0000−0003−4316−9001], and Clinton Fookes[0000−0002−8515−6324]

Queensland University of Technology, Brisbane, Australia
{r5, s.denman, s.sridharan, c.fookes}@qut.edu.au

Abstract. Planar homography estimation refers to the problem of computing a bijective linear mapping of pixels between two images. While this problem has been studied with convolutional neural networks (CNNs), existing methods simply regress the location of the four corners using a dense layer preceded by a fully-connected layer. This vector representation damages the spatial structure of the corners since they have a clear spatial order. Moreover, four points are the minimum required to compute the homography, and so such an approach is susceptible to perturbation. In this paper, we propose a conceptually simple, reliable, and general framework for homography estimation. In contrast to previous works, we formulate this problem as a perspective field (PF), which models the essence of the homography - pixel-to-pixel bijection. The PF is naturally learned by the proposed fully convolutional residual network, PFNet, to keep the spatial order of each pixel. Moreover, since every pixels' displacement can be obtained from the PF, it enables robust homography estimation by utilizing dense correspondences. Our experiments demonstrate the proposed method outperforms traditional correspondence-based approaches and state-of-the-art CNN approaches in terms of accuracy while also having a smaller network size. In addition, the new parameterization of this task is general and can be implemented by any fully convolutional network (FCN) architecture.

Keywords: Homography · Autoencoder · Perspective Field · PFNet.

1 Introduction

Planar homography estimation, which is a fundamental task in computer vision, aims to provide a non-singular linear relationship between points in two images. It has gained much attention since efficiently computing homographies is invaluable for many practical applications, such as image rectification [28,20,4], image registration [7,18], and camera calibration [24,8].

A well-known framework for this task is to use a hand-crafted key point detector to determine the transformation between two images. This typically consists of three steps: key point detection, correspondence matching, and a direct linear transformation (DLT). First, a hand-crafted detector, such as SIFT [17], ORB

[22], KLT [9], SURF [2] etc., is used to extract key points using local invariant features from each image. Correspondence matching aims to match the two detected key point sets by a heuristic criterion (e.g. euclidean distance, correlation, etc.). Then outliers in the initial correspondence set are further eliminated using RANSAC [9]. In the last step, the transformation between two images can be established by making use of a direct linear transform (DLT), which takes as input the correspondence set. While such frameworks have achieved success to some extent, their performance is largely dependent on the accuracy of the key point detector and the correspondence matching. When the correspondences are misaligned, this framework may fail. This situation often happens when hand-crafted features perform poorly in the presence of large variations in perspective, illumination, etc.

To alleviate the dependence on key-point detection and matching performance, estimating homographies in a end-to-end fashion using convolutional neural networks (CNNs) has been widely studied in the last two years. Most existing works [21,12,5] aim to regress the 4-point homography directly, i.e., the offsets of the four corners are embedded into a vector and then predicted by a CNN through a dense layer preceded by a fully-connected layer. Unfortunately, the strategy of formulating this task has three drawbacks:

1. The vector representation of the four corners via a dense layer breaks their spatial structure and hence results in a loss of spatial information.
2. Four points are the minimum to estimate the homography. Perturbation of any one predicted corner may significantly affect the accuracy of the estimate. In other words, a 4-point homography is not robust.
3. Fully-connected layers are frequently used in these architectures, which increases the computation requirement and decreases the representation power of the network. For example, the fully-connected layers in networks such as VGGNet [23], AlexNet [13], etc., occupy nearly 20% (i.e., less representation capacity) for the weights but require 80% of the computation.

Aiming to address the aforementioned issues, we propose a conceptually simple, reliable, and general framework for homography estimation. Our approach parametrizes the bijective pixel-to-pixel linear mapping described by a homography using the perspective field, which encodes every pixels' offsets while retaining their spatial information. Subsequently, this new formulation is naturally learned by the perspective field network (PFNet), which contains a novel deconvolution block designed in a residual fashion.

The contribution in this paper is two-fold:

1. We propose a new parameterization for homography estimation which can be easily applied to any existing FCN.
2. We introduce the PFNet for homography estimation, endowed with the proposed novel deconvolution blocks, that allows for upsampling and predicting homographies efficiently. Thanks to the proposed fully convolutional architecture, the network has fewer hyper-parameters while greatly boosting the performance compared with existing state-of-the-art methods.

2 Related Works

Approaches for homography estimation can be divided into two groups: feature-based and end-to-end deep-learning-based approaches.

Feature-based homography estimation Prior to the advent of deep learning, the standard approach for homography estimation consisted of three stages: (1) detecting a set of distinctive points using hand-crafted features (e.g. SIFT [17], ORB [22], SURF [2]), (2) finding matching detections using the distance between points according to a descriptor and a random sampling method (such as RANSAC [9]), and (3) estimating the homography via a direct linear transformation (DLT) [9]. Despite the prevalence of this style of approach, its performance is largely dependent on the accuracy of feature point detection and matching. It may fail when correspondences cannot be detected accurately due to the large variances in illumination, perspective, and clarity. In addition, designing effective features for new data or tasks typically requires new domain knowledge, which is often unavailable for new scenes. Regarding the properties of existing local invariant features, we refer readers to [19].

Homography prediction with CNNs Due to the remarkable advances of deep learning in the field of computer vision, research into homography estimation has been driven towards the use of CNNs. Detone *et al.* [5] was the first to use a CNN for regressing four corners offsets from a pair of image in a VGG-like network with 14 layers. Their technique was shown to outperform traditional feature-based methods, such as SITF, ORB, SURF, etc., when sufficient features could not be detected. Inspired by this work, [12] proposed a twin CNN module, which processes two images in parallel in the first 4 convolutional layers and concatenates feature maps to generate estimates using another 4 convolutional layers and 2 fully-connected layers. Subsequently, they arrange the proposed module in a stacked manner and trained them in a hierarchical fashion to reduce estimation error bounds. While this iterative architecture significantly outperforms [5], it increases both the model and computational complexity significantly.

Alternatively, to improve the quality of the predicted homography, [21] develops a hybrid approach that combines the strengths of both a deep learning and a conventional feature-based method. Specifically, it relies on features to compute the homography estimates using a 4-point parameterization in the first half of the network and then optimizes the pixel-wise photometric loss of the two images in the second half.

These three approaches have two key commonalities: (1) they design a dedicated network which lacks flexibility and so cannot benefit from existing the state-of-the-art CNN architectures, and (2) the 4-point parameterization does not model the bijective pixel-wise linear mapping and breaks the spatial information among four corners' offsets.

Unlike these existing methods, we formulate this task as predicting a perspective field, and then adapt the proposed fully convolutional network (FCN) to learn it simply and efficiently from a pair of images. Moreover, this new parameterization does not rely on any specific network and as such it has great generalization. It can be easily learned by any existing FCN to estimate homographies.

3 Homography Parameterization

The standard way to parameterize a homography is to find a 3×3 non-singular matrix which can establish a bijective projection between every pixel in two images. Suppose that we have a point $\mathbf{p} = (x_\mathbf{p}, y_\mathbf{p})$ on image I_A and its corresponding point on image I_B is denoted as $\mathbf{q} = (x_\mathbf{q}, y_\mathbf{q})$. Thus, the homography $\mathbf{H}$ that maps $\mathbf{p}$ to $\mathbf{q}$ can be expressed as,

$$\begin{bmatrix} x_\mathbf{q} \\ y_\mathbf{q} \\ 1 \end{bmatrix} \sim \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_{33} \end{bmatrix} \begin{bmatrix} x_\mathbf{p} \\ x_\mathbf{p} \\ 1 \end{bmatrix} . \tag{1}$$

Since the transformation is defined up to a scaling factor, it can be normalized by scaling $\mathbf{H}_{33} = 1$, and as such $\mathbf{H}$ can be parameterized by 8 parameters.

While this straightforward parameterization has been widely used in computer vision, it may be problematic when we employ it as the learning objective of a CNN. This is because $\mathbf{H}$ implicitly encodes the rotation, translation, focal length, skew, and optical center parameters; and the value range of each parameter has high variance. Typically, the magnitude of translation, focal length, and optical center are much greater than that of rotation, as such it is very hard to balance (normalize) all parameters' contribution to the loss function.

To address this issue, we parameterize $\mathbf{H}$ between I_A and I_B using a PF, F_AB. F_AB has two channels F_{AB_x}, F_{AB_y}, where each channel represents the pixel offset caused by the homography in the x and y directions. Let W and H represent the width and height of I_A and I_B respectively. The displacement of $\mathbf{p}$ in terms of the x-axis is denoted $\Delta x_\mathbf{p} = x_\mathbf{p} - x_\mathbf{q}$. Similarly, $\Delta y_\mathbf{p} = y_\mathbf{p} - y_\mathbf{q}$ is the displacement of $y_\mathbf{p}$. Thus F_{AB_x}, F_{AB_y} can be expressed as follows,

$$F_{\mathrm{AB}_x} = \begin{bmatrix} \Delta x_{\mathbf{p}_{11}} & \Delta x_{\mathbf{p}_{12}} & \cdots & \Delta x_{\mathbf{p}_{1W}} \\ \Delta x_{\mathbf{p}_{21}} & \Delta x_{\mathbf{p}_{22}} & \cdots & \Delta x_{\mathbf{p}_{2W}} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta x_{\mathbf{p}_{H1}} & \Delta x_{\mathbf{p}_{H2}} & \cdots & \Delta x_{\mathbf{p}_{HW}} \end{bmatrix} , \tag{2}$$

and

$$F_{\mathrm{AB}_y} = \begin{bmatrix} \Delta y_{\mathbf{p}_{11}} & \Delta y_{\mathbf{p}_{12}} & \cdots & \Delta y_{\mathbf{p}_{1W}} \\ \Delta y_{\mathbf{p}_{21}} & \Delta y_{\mathbf{p}_{22}} & \cdots & \Delta y_{\mathbf{p}_{2W}} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta y_{\mathbf{p}_{H1}} & \Delta y_{\mathbf{p}_{H2}} & \cdots & \Delta y_{\mathbf{p}_{HW}} \end{bmatrix} . \tag{3}$$

To convert the PF back to the homography, we first simply restore the correspondences between two images by,

$$\{\mathbf{p}_i, \mathbf{q}_i\}_{i=1}^{HW} = \{\mathbf{p}_i, \mathbf{p}_i - \Delta \mathbf{p}_i\}_{i=1}^{HW} , \tag{4}$$

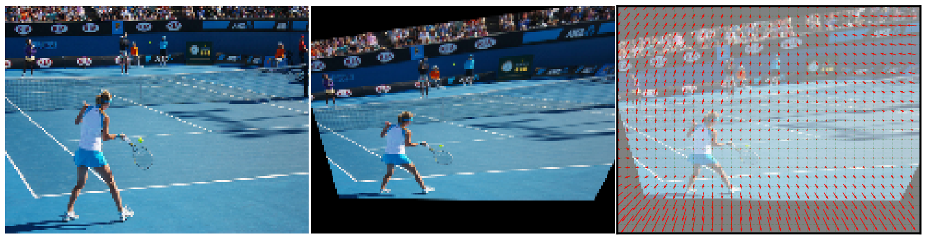

Fig. 1. Illustration of the PF between a pair of images. From left to right: the original image, the warped image (using a random homography), the PF. The PFs are plotted on a green gird at a 10 pixel interval. The red quiver on the pixel represents the direction and length of the pixel offset. By combining this PF with the warped image, we can easily recover the original image and obtain the homography.

where i is the index of the correspondence, and $\Delta\mathbf{p}_i = (\Delta x_{\mathbf{p}_i}, \Delta y_{\mathbf{p}_i})$. Once the correspondences have been established by Equation 4, RANSAC is applied to further filter out outliers and then we use a DLT to solve the homography from a set of equations,

$$\begin{bmatrix} \mathbf{0}^\top & -\mathbf{q}_i^\top & y_{\mathbf{p}_i}\mathbf{q}_i^\top \\ \mathbf{q}_i^\top & \mathbf{0}^\top & -x_{\mathbf{p}_i}\mathbf{q}_i^\top \end{bmatrix} \mathbf{h} = 0. \quad i = 1, \ldots, HW \ , \tag{5}$$

where $\mathbf{h} \in \mathbb{R}^{9\times 1}$ is the vectorized homography $\mathbf{H}$. Figure 1 visualizes the PF generated by a pair of images.

4 Methodology

In Section 4.1, we describe the architecture of the proposed fully convolutional residual network in detail. Then in Section 4.2 and 4.3, the strategies of building the loss function and optimization are introduced respectively. Figure 2 visualizes the proposed architecture.

4.1 FCN Architecture

Our objective is to design an end-to-end architecture that outputs the PF between a pair of images. Thus choosing an appropriate CNN backbone is crucially important. In this paper, ResNet-50 [10] is employed as our FCN backbone for its state-of-the-art performance in many computer vision tasks. In addition, ResNet-50 does not have high computational complexity when compared to many deep networks such as ResNet-101, DenseNet, VGG, etc., while having an acceptable level of performance.

The proposed fully convolutional residual network consists of two parts: the encoder and the decoder. In the encoder, we strictly use the first 4 stages of ResNet-50, where each stage is constructed by a specific number of residual

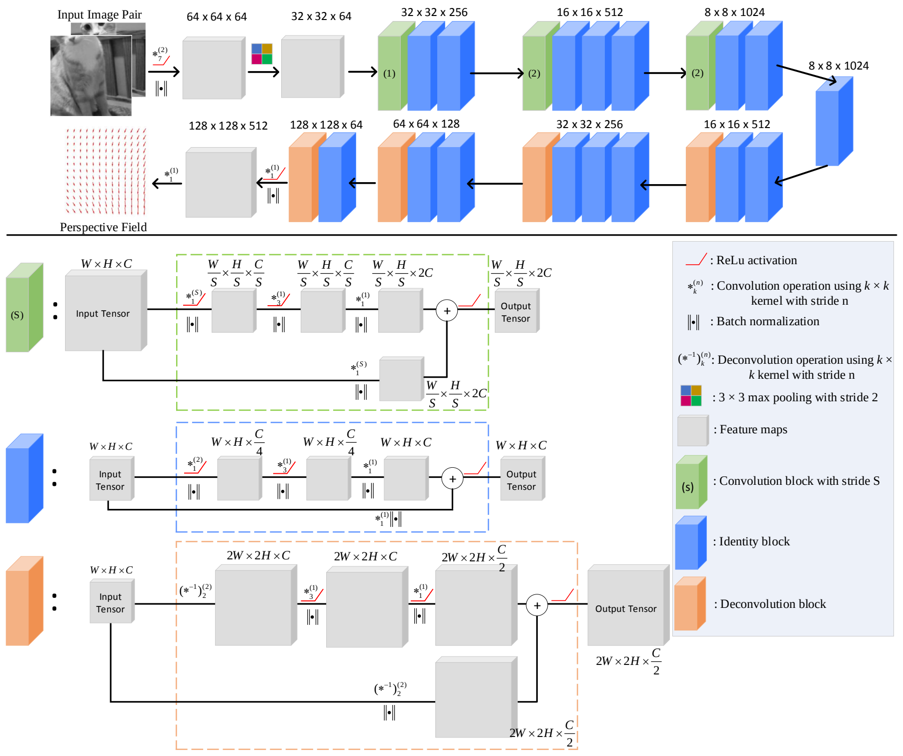

Fig. 2. The architecture of the proposed network. This network consists of two parts: encoder (the top row) and decoder (the bottom row). In the encoder, the network takes as input a pair of images and encodes its information into a set of feature maps through a series of convolutions. In the decoder, the PF information embedded in the feature maps is recovered progressively. In the second last layer, we use 512 filters to expand the width to enhance the representation ability of the network. The last layer outputs F_{AB_x} and F_{AB_y} respectively.

blocks [10] (including identity blocks and convolution blocks), and we remove the 5th stage and following fully-connected layer to enable the connection between the encoder and decoder. The encoder takes as input a $W \times H \times 2$ gray image pair and gives rise to a tensor of features of shape $W_m \times H_m \times C_m$ for stage $m = \{3, 4\}$, where $W_m = \frac{W}{2^m}$, $H_m = \frac{H}{2^m}$, and $C_m = 2^{6+m}$. The output of the encoder captures the spatial correlation of the image pair at a set of uniformly sampled pixel locations, and forms the input to the decoder.

The decoder aims to learn how to generate the PF given the spatial correlations learned by the encoder. To achieve this goal, it progressively upsamples the feature maps through a series of identity blocks and the proposed deconvolution block, to ensure the output has the same size as the input image. The decoder is symmetrical to the encoder. It also consists of 4 stages, where each stage (denoted as $n = \{1, \dots, 4\}$) has the same number of residual blocks but replaces the convolution block with the proposed deconvolution block. The output tensor generated from stage n is of shape $W_n \times H_n \times C_n$, where $W_n = \frac{W}{2^{4-n}}$, $H_n = \frac{H}{2^{4-n}}$, and $C_n = 2^{10-n}$. The deconvolution block embedded in each stage provides the ability to propagate low-resolution feature maps to high-resolution feature maps. More details are introduced in following discussion.

Once the encoder and decoder have been set up, we convolve the last layer of the decoder with a $1 \times 1 \times 512$, and a $1 \times 1 \times 2$ filters successively to enable the prediction of the PF, as such the output size is $W \times H \times 2$. It is worth noting that our model does not have any long skip connections between m and the corresponding n, which are heavily used in segmentation tasks [16]. Skip connections aim to propagate pixel-wise information from low stages to high stages to assist segmentation. However, homographies have significantly distorted pixel-to-pixel spatial relationships between two images and as such we omit long skip connections from our architecture. Moreover, we do not use drop out to further regularize our architecture because we want to enable fair comparison with other baselines.

Deconvolution Block The structure of the deconvolution block is depicted in the bottom half of Figure 2. The deconvolution block is the last residual-like block in each decoder stage. It receives an input tensor, and increases the spatial resolution using a deconvolution operation. The design of the deconvolution block is inspired by the philosophy of ResNet and as such we follow two simple rules: (1) the input tensor is added directly to the end of the block by being only convolved once to enable the residual operation, and (2) The block doubles the feature map size while halving the number of the filters. More specifically, we first adopt two 2×2 deconvolutional kernels with stride 2 to convolve the input tensor to obtain two data streams. In the first data stream, the number of filters is the same as that of the input tensor. The second data stream halves the number of filters of the input tensor to enable a direct add operation between the output of the first data stream and the second.

4.2 Loss Function

A standard loss for dense prediction problems is the l_2 loss, which minimizes the Euclidean distance between predictions and the ground truth. However, this commonly used loss is not suitable for PFNet because the l_2 loss places emphasis on outliers, i.e., the optimization objective focuses on minor outliers rather than major inliers in the PF. As mentioned in Section 3, we further use RANSAC to filter out outliers predicted by PFNet, this operation means that PFNet is robust to any outliers. Thus what really determines the performance of PFNet is the inliners in the estimated PF. To make use of this property, we choose a smooth-l_1 loss, which is robust to outliers, to optimize this network as follows,

$$\text{loss} = \begin{cases} 0.5 \times (F_{AB_*}^{\text{pred}} - F_{AB_*})^2, & \text{if } \left|F_{AB_*}^{\text{GT}} - F_{AB_*}\right| \leq 1 \\ \left|F_{AB_*}^{\text{pred}} - F_{AB_*}\right| - 0.5, & \text{otherwise} \end{cases} \tag{6}$$

where $*$ denotes x or y, and superscript *pred* represents the predictions in terms of F_{AB_*}. More details and a comparison of the smooth-l_1 and l_2 loss on PFNet are provided in supplementary materials.

4.3 Training and Inference

We have implemented our system using Keras [3] and TensorFlow [1]. The full network is trained in an end-to-end fashion. We train this network from scratch, where all weights are initialized using the method of [6]. We choose NAdam [25] as our optimizer because it generally acts well for many datasets and has a favorable convergence speed. The parameters used for NAdam are set to the defaults, with momentum 0.9, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. We use a learning rate of 1×10^{-3} in the first 40 epochs, and divide it by 10 after each 40 epochs. Each epoch contains 1000 steps in which the batch size is set to 64. The model has finished when training reaches 80 epochs. One batch takes approximately 2s on a NVIDIA Tesla M40 GPU and it takes about 44 hours of training for the model to converge. Once a PF has been obtained from a given image pair in the inference stage, we use Equation 4 to recover dense correspondences and then adopt RANSAC to further filter outliers within. It is important to note that RANSAC is not contained within the network, and is used as a post processing step to improve the quality of the homography extracted from PFNet. In the last step, a DLT is applied on the refined dense correspondences to obtain the homography.

5 Experiments

In this section, we introduce the dataset used in our experiments and the strategy for generating samples (see Section 5.1). The baseline models including CNN-based and standard feature-based methods are outlined in Section 5.2. Section 5.3 evaluates the proposed network thoroughly with respect to CNN-based approaches to demonstrates the capacity of the proposed architecture and the contribution of the novel deconvolution block. Furthermore, a robustness analysis of our network and traditional feature-based methods, such as SIFT, and SURF, is examined in an extreme image contamination environment (see Section 5.4).

5.1 Synthetic Dataset Generation

The availability of large scale datasets is crucially important for the performance of CNNs. Unfortunately, there is no public dataset for this task and as such

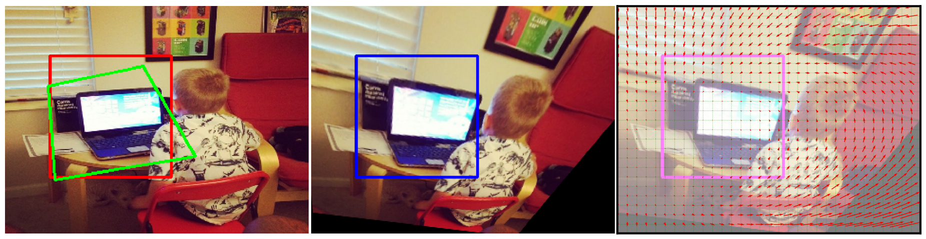

Fig. 3. Visualization of training dataset generation. **Left**: we randomly generate the red square in the original image (I_A) and then perturb it's four corners in $(-\rho, \rho)$ to get the green quadrilateral. The ground truth $\mathbf{H}$ can be computed from the red and green box using 4-point DLT. **Middle**: The warped image (I_B) obtained by applying the inverse of $\mathbf{H}$ on I_A. The blue square has the exact same position as that of the red box. **Right**: the F_AB generated from I_A and I_B. The magenta box has the same position as the blue box. Therefore, we stack the gray-scaled red patch and blue patch to form the input tensor with shape $128 \times 128 \times 2$. To obtain the $128 \times 128 \times 2$ output tensor, we stack F_{AB_x} and F_{AB_y} together in the magenta square.

previous works use synthetic datasets to train and evaluate the approaches. By strictly following previous works' dataset generation rules [5], we trained our network using the Microsoft Common Objects in Context (MSCOCO) 2014 dataset [15], which contains 82,081 training images and 40,137 testing images. All images in the dataset have been transformed to gray scale and resized to 320×240 prior to further processing.

In the training phase, we generate the pair of input and output tensors by applying a random homography to an image. More specifically, we first randomly generate a 128×128 square of I_A with a random left top corner $\mathbf{p}_\text{ltc} = (x_{\mathbf{p}_\text{ltc}}, y_{\mathbf{p}_\text{ltc}})$. Secondly, we perturb each corner of this square (four corners in total) using a random value in the range of $(-\rho, \rho)$. Then the ground truth homography can be obtained from the original corners and the perturbed corners. Please note that we set ρ to 32 and limit $32 < x < 224$ and $32 < y < 80$ to ensure that the four perturbed corners do not fall outside the image. Therefore, the warped image I_B can be computed using the randomly generated homography. To create the input of the proposed network, we crop two 128×128 patches from I_A and I_B at $\mathbf{p}_\text{ltc}$ and then stack them together to form the input tensor whose shape is $128 \times 128 \times 2$. With respect to output, we crop two patches of the same size at $\mathbf{p}_\text{ltc}$ from F_{AB_x} and F_{AB_y}, such that the shape of the output tensor is $128 \times 128 \times 2$. Figure 3 visualizes our dataset generation strategy.

5.2 Baseline Models

To thoroughly evaluate the performance of the proposed network on this task, we select previous state-of-the-art CNN-based works [5,12,14] and the feature-based methods SIFT and SURF as the baseline models for comparison.

HomographyNet [5] uses a VGG-like network for this task which heavily uses fully-connected layers. HCN-x [12] stacks the proposed twin convolutional regression networks in a hierarchical way to estimate the homography between a pair of images, where x stands for the number of stacked modules. HomographyNet and HCN-x both make use of the traditional homography parameterization. While FCRN [14] is proposed for depth estimation, which is another dense prediction task, its architecture is similar to ours. Both PFNet and FCRN are built upon ResNet-50. The difference between FCRN and PFNet is that FCRN uses the upsampling block [14] to do deconvolution and does not use any identity block in the deconvolution stage. To demonstrate the effect of the identity block in the architecture, we add identity blocks to FCRN in the same way as ours and refer this method to as FCRN-IB. Furthermore, the performance of the proposed deconvolution block is validated by comparing PFNet with FCRN-IB. To maximize the performance of PFNet, we train PFNet again using 300 epochs, where the learning rate is divided by 10 after each 100 epochs, and other parameters remain the same. We denote this method as PFNet-300. Besides applying the PF to FCRN and FCRN-IB, to testify the generalization of this parameterization, we also apply the PF on the state-of-the-art FCN-DenseNet [11,27,26], which has been successfully applied to many dense prediction problems.

Regarding SIFT and SURF, we directly employ them using their OpenCV implementations with default parameter settings to detect key points and extract local descriptors on the input patch. Subsequently, correspondences are established using a standard matching and RANSAC scheme to estimate the homography.

5.3 Evaluation

We evaluate the performance of our approach and CNN-based methods on test data generated using the strategy of [5,12]. Specifically, we generate test samples from 5000 randomly selected images from the test dataset of MS COCO 2014 by following the data generation process described in Section 5.1. To evaluate the baseline models fairly, we employ the mean average corner error (MACE) as the metric, which is widely used in previous research [12,5]. MACE averages the Euclidean distance between the ground truth and the estimated positions of the four patch corners. Table 1 reports the results of each method in terms of MACE and also reports the network size.

Focusing on the first two columns of Table 1, we find that PFNet consistently outperforms baseline models with respect to MACE. Specifically, compared with the previous state-of-the-art, HCN-4, we have reduced MACE from 3.91 pixels to 0.92. The performance of HomographyNet is the worst among learning-based methods because of the shallow architecture and 4-point parametrization. It is worth noting that FCRN performs worse than FCRN-IB, which adds the identity blocks in the decoder. The hypothesis is that the extra layers brought by identity block increases the representative capacity of the decoder in the network, and as such the decoder can recover the PF distribution from the output of the encoder. In particular, we obtain performance improvements when

Method	**MACE**	**Computation Complexity**	
		Architecture Depth	Parameters
HomographyNet [5]	9.20	14	34M
HCN-1 [12]	13.04	14	17M
HCN-2 [12]	6.39	28	34M
HCN-3 [12]	4.46	42	51M
HCN-4 [12]	3.91	56	68M
FCN-DenseNet [11]	4.20	121	20M
FRCN [14]	1.86	72	28M
FRCN-IB	1.72	96	31M
PFNet	**1.63**	96	31M
PFNet-300	**0.92**	96	31M

Table 1. Mean average corner error (MACE) comparison between our method and various baselines.

we replace the upsampling block [14] with the proposed deconvolution block in FCRN-IB to form PFNet, which suggests that the deconvolution block can avoid losing information while upsampling the input feature maps by a 2X ratio when compared to the upsampling block. This is because the input tensors in the upsampling block are convolved with an extra convolutional layer before it propagates information to the end of the block. The extra convolutional layer closes the shortcut between the input and output tensor, and as such it obstructs the information flow used to learn the residual function. The lower validation loss and MACE of PFNet in Figure 4 also demonstrates aforementioned analysis. For more details and comparisons between these three networks, readers are referred to the supplementary material. All four networks, i.e., FCN-DenseNet, FRCN, FRCN-IB, PFNet, have comparable performance with previous works and as such we claim the new homography parametrization for learning homographies has a good generalization ability.

Observing the last two columns of Table 1, one may see that HomographyNet and HCN-x have a significantly larger hyper parameter space. This is because when they formulate homography estimation as needing to estimate 8 parameters, they have to use at least two fully-connected layers to regress the homography. Therefore, the architectures have a huge numbers of parameters when compared to FCNs with the same depth. Please note that HCN-x progressively improves the performance by stacking the modules one by one. As a consequence of this hierarchical approach, a network with only 56 layers depth contains 68M parameters. However, the depth of a deep learning architecture is crucially important for computer vision tasks as demonstrated in many studies [10].

Qualitative Results We show example predictions of PFNet in Figure 5. The model generates the PF from given image pairs and the predicted homography is recovered from the generated PF. Please note that the predicted PF is very

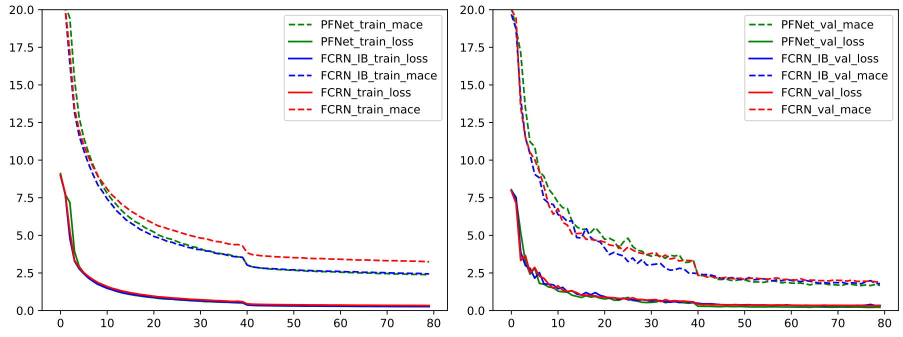

Fig. 4. The MACE and the loss value in the training and validation phases for FCRN, FCRN-IB, and PFNet.

similar to the ground truth. The same is true for the warped images. Additional qualitative experiments can be found in supplementary materials.

5.4 Evaluation on Noisy Data

In this section, we compare the proposed method with two feature-based methods, SIFT and SURF, in an extremely noise-contaminated environment to test the robustness of the algorithms. To simulate the noise contamination, we first create a image contaminator list which contains four contaminators: Gaussian blur (GB), dropout (DO), salt and pepper noise (SP), and Gaussian noise (GN); where their parameters are set in a range of (0, 4), (0, 0.4), (0, 0.4), and (0, 150) respectively according to the ImgAug toolbox [1].

To enable estimation of the PF in noise-contaminated images, we retrained our network by creating the dataset following the steps described in Section 5.1 but randomly select 0 to 4 contaminators to contaminate I_{A} and I_{B} separately. To analyze the robustness of the algorithms thoroughly, we split the testing experiments into 7 groups: 1. **N** (normal): the dataset is uncontaminated. 2-5. **GB** (Gaussian blur), **DO** (dropout), **SP** (salt and pepper noise), **GN** (Gaussian noise): only use one selected contaminator. 6. **M** (mix): mix 2 to 4 contaminators in a random order. 7. **HA** (heavy augmentation): mix 2 to 4 contaminators in a random order where each contaminator uses the maximum parameter. Testing samples in which MACE is less than 38.40 (30% $\times$ 128) are marked as successfully estimated samples. Using this, a success rate is employed to evaluate the robustness of each algorithm.

Table 2 reports the performance of the algorithms. We see that our method significantly outperforms feature-based methods in all cases. A 100% success rate in all cases means that our method has a high robustness when confronted with a noisy dataset. The significant drop in performance of SIFT and SURF when evaluating **DO** and **SP** contaminated images suggests that feature-based

[1] ImgAug toolbox: https://github.com/aleju/imgaug

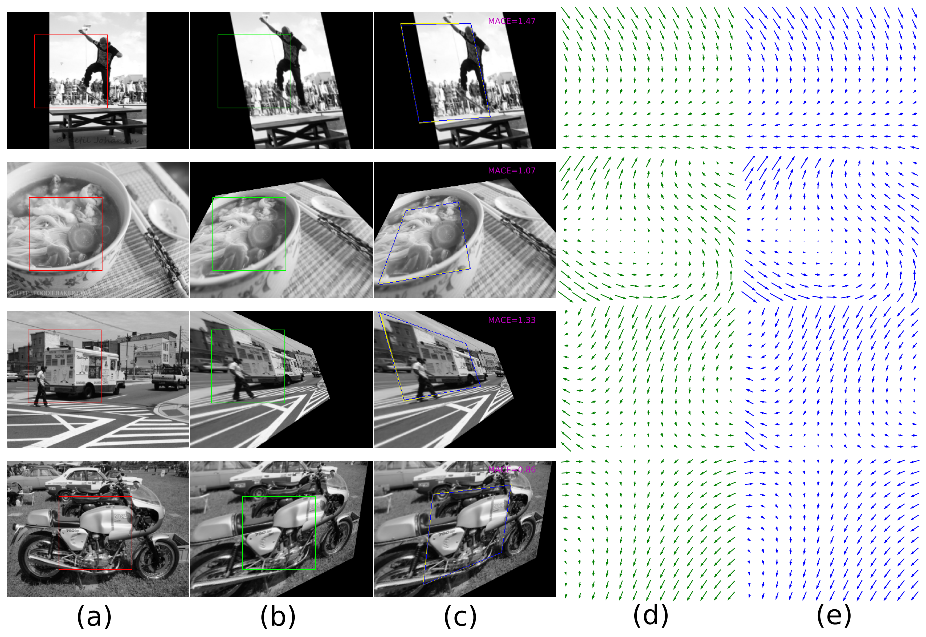

Fig. 5. Example PF generated by PFNet and the predicted homography. (a) The original 320×240 gray image. The red box is used to generate input. (b) The warped image transformed by the ground truth homography. A green square is placed in the same position as that of red box to form the input image patch pairs. (c) The warped image transformed by the predicted homography. MACE error is annotated in the top right corner of the image. The blue quadrilateral represents the ground truth homography, and yellow is the predicted homography. (d) The ground truth PF. (e) The predicted PF.

methods do not work when a large proportion of pixels are missing. It is worth noting that our method works acceptably for extremely contaminated image pairs, i.e., **HA**. However, SIFT, ORB, etc does not work at all in these conditions. Figure 6 shows example images and computed homography for the **HA** case.

5.5 Conclusions

In this work, we introduced a new parameterization for homography estimation, i.e., the perspective field, which models the nature of the homography - pixel-to-pixel bijection. Unlike previous CNN approaches that require fully-connected layers for regressing, the PF can be easily learned by FCN architectures to significantly reduce computational complexity. To address this parameterization, we developed the PFNet architecture, which can naturally learn the PF and enables end-to-end training. The proposed deconvolution block follows the residual function fashion, and as such input information can be passed to the output directly by a shortcut connection for the upsampling. Our experiments demonstrate the power and robustness of our model with respect to baselines and previous state-of-the art works. The qualitative experiments also show visually pleasing results.

	MACE							**Success rate (%)**						
Method	N	GB	DO	SP	GN	M	HA	N	GB	DO	SP	GN	M	HA
PFNet	**2.87**	**2.81**	**3.31**	**3.78**	**2.75**	**4.53**	**9.51**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
SIFT	25.35	25.54	25.80	26.18	25.28	25.78	fail	42.45	19.44	7.98	5.21	42.01	1.4	0
SURF	25.43	25.43	26.19	25.86	25.33	27.21	fail	40.13	21.22	4.49	2.5	38.67	0.4	0

Table 2. Comparison between the proposed network with traditional feature-based methods regarding robustness analysis.

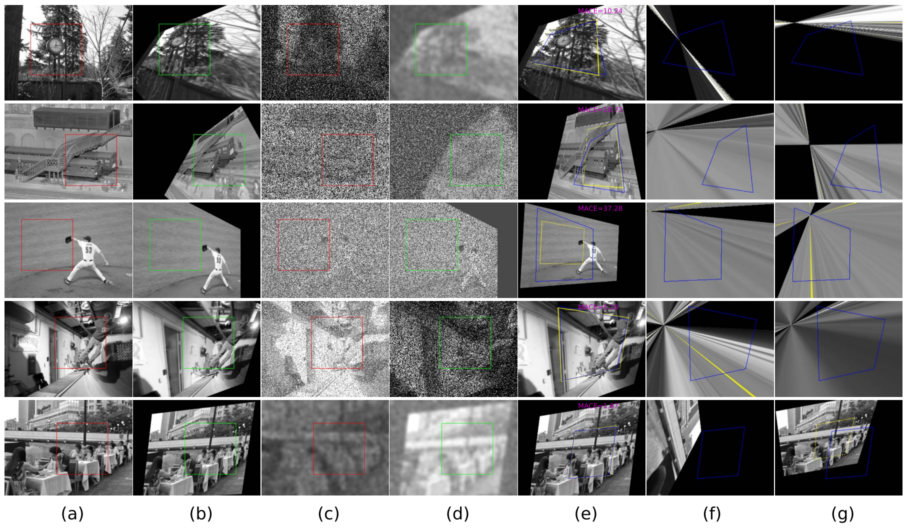

Fig. 6. Visualization of estimating homographies in heavily contaminated image pairs. (a): Input image I_{A}. The red square shows the patch we extracted for training (b): The warped image I_{B}. The green square has the same location as the red one. (c): We randomly select two contaminators from the list and use the highest value in the value range to corrupt I_{A}. (d) I_{B} is contaminated in the same way as I_{A} but separately. (e)-(g): The results obtained from our method, SIFT, and SURF respectively. To improve visualization, we draw the ground truth homography using a blue square. The yellow squares are the predicted homographies.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI. vol. 16, pp. 265–283 (2016)
2. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: Proceedings of European Conference on Computer Vision. pp. 404–417. Springer (2006)
3. Chollet, F., et al.: Keras (2015)
4. Chum, O., Matas, J.: Planar affine rectification from change of scale. In: Proceedings of Asian Conference on Computer Vision. pp. 347–360. Springer (2010)
5. DeTone, D., Malisiewicz, T., Rabinovich, A.: Deep image homography estimation. arXiv preprint arXiv:1606.03798 (2016)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010)
7. Gong, M., Zhao, S., Jiao, L., Tian, D., Wang, S.: A novel coarse-to-fine scheme for automatic image registration based on sift and mutual information. IEEE Transactions on Geoscience and Remote Sensing **52**(7), 4328–4338 (July 2014). https://doi.org/10.1109/TGRS.2013.2281391
8. Ha, H., Perdoch, M., Alismail, H., Kweon, I.S., Sheikh, Y.: Deltille grids for geometric camera calibration. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 5344–5352 (2017)
9. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2017)
12. Japkowicz, N., Nowruzi, F.E., Laganiere, R.: Homography estimation from image pairs with hierarchical convolutional networks. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW). pp. 904–911 (Oct 2017). https://doi.org/10.1109/ICCVW.2017.111
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
14. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: Proceedings of IEEE International Conference on 3D Vision. pp. 239–248. IEEE (2016)
15. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proceedings of European Conference on Computer Vision. pp. 740–755. Springer (2014)
16. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
17. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)

18. Ma, J., Zhou, H., Zhao, J., Gao, Y., Jiang, J., Tian, J.: Robust feature matching for remote sensing image registration via locally linear transforming. IEEE Transactions on Geoscience and Remote Sensing **53**(12), 6469–6481 (Dec 2015). https://doi.org/10.1109/TGRS.2015.2441954
19. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence **27**(10), 1615–1630 (2005)
20. Monasse, P., Morel, J.M., Tang, Z.: Three-step image rectification. In: Proceedings of British Machine Vision Conference. pp. 89–1. BMVA Press (2010)
21. Nguyen, T., Chen, S.W., Shivakumar, S.S., Taylor, C.J., Kumar, V.: Unsupervised deep homography: A fast and robust homography estimation model. IEEE Robotics and Automation Letters **3**(3), 2346–2353 (July 2018). https://doi.org/10.1109/LRA.2018.2809549
22. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: Proceedings of International Conference on Computer Vision. pp. 2564–2571. IEEE (2011)
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
24. Staranowicz, A.N., Brown, G.R., Morbidi, F., Mariottini, G.L.: Practical and accurate calibration of rgb-d cameras using spheres. Computer Vision and Image Understanding **137**, 102 – 114 (2015). https://doi.org/https://doi.org/10.1016/j.cviu.2015.03.013, `http://www.sciencedirect.com/science/article/pii/S1077314215000703`
25. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of International conference on Machine Learning. pp. 1139–1147 (2013)
26. Zhang, H., Patel, V.M.: Densely connected pyramid dehazing network. arXiv preprint arXiv:1803.08396 (2018)
27. Zhu, Y., Newsam, S.: Densenet for dense flow. arXiv preprint arXiv:1707.06316 (2017)
28. Zitova, B., Flusser, J.: Image registration methods: a survey. Image and vision computing **21**(11), 977–1000 (2003)