Mathematical Foundations of Computing
©American Institute of Mathematical Sciences
Volume **3**, Number **2**, May **2020**
doi:10.3934/mfc.2020014
pp. **125–140**

HOMOGRAPHY ESTIMATION ALONG SHORT VIDEOS BY RECURRENT CONVOLUTIONAL REGRESSION NETWORK

Yang Mi, Kang Zheng and Song Wang*
University of South Carolina, Columbia, 29208, USA

(Communicated by Dapeng Tao)

Abstract. Many moving-camera video processing and analysis tasks require accurate estimation of homography across frames. Estimating homography between non-adjacent frames can be very challenging when their camera view angles show large difference. In this paper, we propose a new deep-learning based method for homography estimation along videos by exploiting temporal dynamics across frames. More specifically, we develop a recurrent convolutional regression network consisting of convolutional neural network (CNN) and recurrent neural network (RNN) with long short-term memory (LSTM) cells,followed by a regression layer for estimating the parameters of homography. In the experiments, we evaluate the proposed method on both the synthesized and real-world short videos. The experimental results verify that the proposed method can estimate the homographies along short videos better than several existing methods.

1. **Introduction.** A homography is the invertible mapping between two images of the same planar surface [16]. Homography estimation is an important problem in computer vision and plays a key role in many video-based applications, such as video stitching [22], video stabilization [19], optical flow estimation [28], action recognition [33, 34, 35], simultaneous localization and mapping (SLAM) [9], visual odometry (VO) [14], and augmented reality [48]. Many of these applications, such as video stitching [22] and action recognition [33, 34, 35], require to estimate homographies between both adjacent and non-adjacent frames.

We can simply treat a pair of frames, either adjacent or non-adjacent, as two input images and directly compute their homography without considering the information of any other frames. For this purpose, classical methods first detect hand-crafted features [30, 11, 5, 42] on two images separately, followed by estimating homographies with Random sample consensus (RANSAC) [13]. Many template matching algorithms are also developed for estimating homographies between two images by identifying and matching relevant regions [2, 6]. Recently, many deep learning methods [12, 8, 40] were developed to learn the homography parameters from training image pairs with known homographies. However, non-adjacent video frames may show very different camera view angles, which bring in difficulties to identify reliable matched features, either hand-crafted or deeply learned, for estimating homographies.

2020 *Mathematics Subject Classification.* Primary: 68U10, 68T10.
Key words and phrases. Homography, short videos, recurrent convolutional network, regression.

* Corresponding author: Song Wang.

A video consists of a sequence of frames, which usually show good temporal relationship. Within a video, the homography transformation between a pair of non-adjacent frames is resulted from a temporal sequence of adjacent-frame homography transformations between these two frames. The homography transformation between non-adjacent frames are not only dependent on the spatial relationship between these frames, but also relied on their temporal relationship. Such temporal relationship can be used to help estimate homography between non-adjacent frames. For example, one can estimate homographies between adjacent frames and then compose them sequentially to get homographies between non-adjacent frames [15]. Template matching can be also applied to temporal sequences in an accumulative way for computing homographies between non-adjacent frames [23]. However, these approaches may accumulate estimation errors sequentially, and lead to large errors in computing non-adjacent frame homographies. We can also run a feature tracker, such as Lucas-Kanade tracker [31], along the video and then use the tracked features across non-adjacent frames to directly estimate their homographies. This approach requires high accuracy of feature tracking over a sequence of frames, which can be difficult in practice.

In this paper, we propose a Recurrent Convolutional Regression Network to estimate homographies along a short video by taking the whole video as the input. With this network, we exploit the temporal dynamics along the whole video in an end-to-end fashion to more accurately estimate the homographies between non-adjacent frames. The proposed network consists of convolutional neural network (CNN) and recurrent neural network (RNN) with long short-term memory (LSTM) cells, followed by a regression layer for estimating the parameters of homography. By employing recurrent architecture with LSTM, the proposed method does not need exhaustive feature/template matching or feature tracking, and can alleviate high accumulative errors in computing homographies between non-adjacent frames. We also introduce a simple but effective approach to synthesize large-scale videos with ground-truth homographies for network training and performance evaluation. In the experiment, we show that the proposed method can estimate more accurate homographies along videos than several existing methods, on both the synthesized and real-world videos.

2. **Related work.**

2.1. **Homography estimation between two images.**

Conventional methods for estimating the homography between two images first extract hand-crafted features, such as
SIFT [30], SURF [5], HOG [11] or ORB [42] from each image. Recently, Barath et al. [3] proposed two general constraints on the orientation- and scale-covariant features (e.g. SIFT). The extracted features are then matched between the two input images using various matching algorithms, such as Fast Library for Approximate Nearest Neighbors (FLANN) [37] or Brute-force (BF) matcher. Finally, Direct Linear Transform (DLT) with Random sample consensus (RANSAC) [13] is applied to estimate the homography between images based on the feature correspondence.

Template matching is also applied for estimating homography between two images [2, 6], where a quadrilateral template area is matched between two images in an iteratively way. In [2], the inverse compositional (IC) algorithm is used to exchange the role of the two images, and leads to the optimization which contains a constant pre-computed Hessian [36]. In [6], transformation is estimated by minimizing the

sum-of-squared-difference between the correct and estimated templates, using an efficient second-order minimization (ESM).

Recently, deep learning has been used for homography estimation between two images. In [12], CNN implemented on VGG architecture [44] is applied to compute the homography. In the network, two approaches were explored to compute homography: direct regression and distribution model via classification. Experiments show that direct regression can achieve more accurate homography estimation. In [8], a cascaded Lucas-Kanade network is developed to progressively refine the homography estimation, by combining the IC algorithm and a pyramid feature representation. In [40], a hierarchy of twin CNNs is developed to regress the homography, with visual warping between adjacent hierarchical levels. In [39], an unsupervised deep learning method which combines CNN based on VGG network and DLT, is developed to estimate homography.

As discussed earlier, the direct application of these methods for estimating homographies between non-adjacent video frames may produce large errors when the non-adjacent frames show very different camera view angles.

2.2. **Homography estimation along a video.** Garcia-Fidalgo et al propose to directly compute homographies between adjacent frames, and then compose them sequentially for estimating homographies between non-adjacent frames along the video [15]. In [23], homography is computed iteratively between each frame and a reference frame using Hyperplane Approximation (HA), which is a template matching algorithm that finds the relationship between the measured error and the transformation parameters variation by using difference decomposition in an off-line processing stage. Such composition or iteration scheme usually suffers from large accumulative errors. Various feature trackers can be used for identifying corresponding features across frames for homography estimation. For example, Lucas-Kanade (LK) tracker [31] first uses "Good Features To Track" [43] for tracking initialization, and then uses Lucas-Kanade algorithm to calculate optical flow for tracking features across frames, along with back-tracking for match verification between frames. The point correspondence of the tracked features can then be used to compute homographies. In practice, the accuracy of the tracked features usually get worse when tracking over many frames, and this affects the accuracy of homographies estimated between non-adjacent frames.

In some applications, structures with special geometry can be detected and tracked along a video for estimating homographies. In [20], homography estimation is accomplished by finding conic correspondences with visual features built on ellipse shapes detected and tracked along the video. In [29], homography for sports and traffic videos is estimated based on the correspondences of lines detected and tracked along the sport or traffic field. These methods require the presence of structures with pre-specified geometry in the video and can only be used in special applications. In this paper, we will develop a deep-learning based method for estimating the homography, which is applicable to general videos, without requiring the presence of such structures.

2.3. **Other relevant work.** Deep learning has also been used to find other kinds of mapping between frames. For example, in [47], CNN was used to learn a mapping from pixels to optical flow, by following a signal processing principles. In [24], CNN was used to regress camera pose from frame to frame, by utilizing transfer learning from large scale classification data. In [46], deep multi-view feature

learning (DMVFL) was proposed to exploit the collaboration between hand-crafted and deep-learning features for person re-identification. They are different from this work in that we aim to estimate homographies along a video. From the technical perspective, our method is also different from these methods by using recurrent architecture with LSTM to exploit the temporal dynamics across frames.

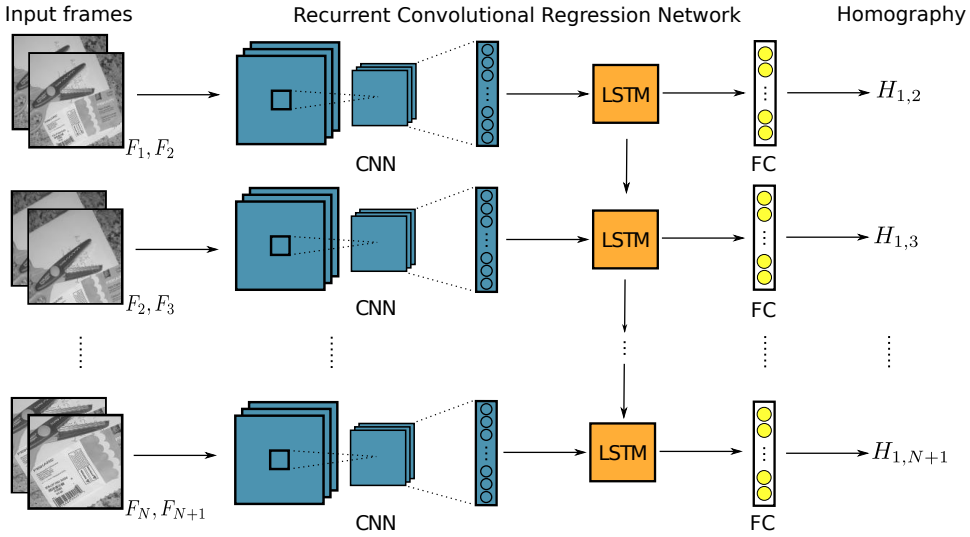

FIGURE 1. Architecture of the proposed recurrent convolutional regression network.

3. **Proposed method.** In this section, we elaborate on the proposed method. As mentioned earlier, homography transform is defined for planar surface, and like many existing work on homography estimation, the proposed method focus on videos in which all frames reflect the same planar surface. For more general videos containing multiple planes, planar surface detection [41] can be applied first and then estimate homography for each plane separately.

3.1. **Overview.** Two common parameterizations of the homography between two images are: the 4-point parameterization and the matrix parameterization [1]. The 4-point parameterization H^{4point} is defined as the 8 displacement values of 4 matched corners as follows:

$$H^{4point} = \begin{pmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \Delta x_3 & \Delta y_3 \\ \Delta x_4 & \Delta y_4 \end{pmatrix} \tag{1}$$

where $(\Delta x_i, \Delta y_i)$ denote the horizontal and vertical displacements for each matched corner on two images. The matrix parameterization H^{matrix} is a 3×3 matrix that contains both the rotational and translational terms. As shown in [12, 40], the 4-point parameterization is more suitable than the matrix parameterization to represent homography, because it is difficult to balance these terms as part of an optimization problem [12]. Besides, H^{4point} can be easily converted to H^{matrix} via

simple perspective transform. Therefore, we use H^{4point} to represent the homographies in this paper and abbreviate it as H for brevity.

Without loss of generality, given an input of video consisting of $N+1$ frames $\{F_1, F_2, \cdots, F_{N+1}\}$, we estimate the homographies between the first frame F_1 and its succeeding N frames. Specifically, we take the original video and re-organize it as a sequence of frame pairs $\{F_{1,2}, F_{2,3}, \cdots, F_{N,N+1}\}$, where $F_{t,t+1} = \{F_t, F_{t+1}\}$ is a pair of adjacent frames with $t = 1, \ldots, N$. We aim to find a sequence of homographies $\{H_{1,2}, H_{1,3}, \cdots, H_{1,N+1}\}$, where $H_{1,t+1}$ is the homography between frames F_1 and F_{t+1}, $t = 1, \cdots, N$. Inspired by the success of regressing homography for image pairs in deep learning methods [12, 40], we formulate the homography estimation along videos as a regression problem. Specifically, we propose a novel Recurrent Convolutional Regression Network, to address this problem. As shown in Fig. 1, the proposed network contains convolutional neural network (CNN) and recurrent neural network (RNN) followed by a regression layer. Since temporal dynamics of a video are transfered through consecutive adjacent-frame pairs, we first use CNN to extract features from each adjacent-frame pair. Then, the features of all adjacent-frame pairs in each video are fed to RNN such that the temporal dynamics are fully exploited. The regression layer performs the final estimation of 4-point homography between F_1 and F_{t+1} for $t = 1, \cdots, N$. We elaborate on the network in the following sections.

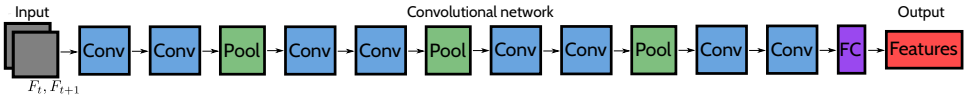

FIGURE 2. Configuration of the CNN used in the proposed method.

3.2. **CNN architecture.** Following [12], we use convolutional neural network (CNN) to first extract features from each pair of adjacent frames F_t and F_{t+1}, where $t = 1, \cdots, N$. Figure 2 shows the configuration of the CNN in our proposed network. Specifically, it contains 8 convolutional layers, every two of which are followed by a Batch Normalization (BN) [21] layer and a MaxPooling layer. There is no MaxPooling layer after the last two convolution layers. The input of the CNN is a pair of adjacent frames with size of 128×128, each of which is a gray-scale image. The number of filters in the convolution layers are 64, 64, 64, 64, 128, 128, 128, and 128, respectively. Rectified Linear Unit (ReLU) [38] is used as the activation function to add non-linearity. Each filter in the convolution layers is of size 3×3 and convolved with a stride of 1. The MaxPooling is performed over 3×3 window with a stride of 2. After the last convolutional layer, a fully-connected (FC) layer is used to output 1,024-dimensional features for each input pair of adjacent frames. During training process, we add a Dropout [45] layer with a drop rate of 0.5 after the FC layer to avoid over-fitting. For an $(N+1)$-frame video $\{F_1, F_2, \cdots, F_{N+1}\}$, we extract a sequence of features $\{x_1, x_2, \cdots, x_N\}$ from adjacent-frame pairs using the CNN network and feed these features to the RNN sequentially.

3.3. **RNN architecture and regression layer.** RNN with LSTM cells is good at sequence modeling, since there are specially designed units inside LSTM cells for remembering important information and forgetting unimportant information along the sequence. Figure 3 shows the architecture of an LSTM cell. Let σ and

ϕ be the logistic sigmoid function and hyperbolic tangent function, respectively. $\{i, f, o, c, h\}$ are the input gate, forget gate, output gate, memory cell and hidden state, respectively. The LSTM sequentially updates $\{i, f, o, c, h\}$ at time step t, given input x_t, hidden state h_{t-1}, and cell state c_{t-1}, as follows:

$$\begin{aligned} i_t &= \sigma\left(\mathcal{W}_{xi}x_t + \mathcal{W}_{hi}h_{t-1} + \mathcal{W}_{ci}c_{t-1} + b_i\right) \\ f_t &= \sigma\left(\mathcal{W}_{xf}x_t + \mathcal{W}_{hf}h_{t-1} + \mathcal{W}_{cf}c_{t-1} + b_f\right) \\ o_t &= \sigma\left(\mathcal{W}_{xo}x_t + \mathcal{W}_{ho}h_{t-1} + \mathcal{W}_{co}c_{t-1} + b_o\right) \\ c_t &= f_t c_{t-1} + i_t\, \phi\left(\mathcal{W}_{xc}x_t + \mathcal{W}_{hc}h_{t-1} + b_c\right) \\ h_t &= o_t\, \phi\left(c_t\right), \end{aligned} \tag{2}$$

where $\mathcal{W}$'s and b's are the network parameters to be learned and $t = 1, \cdots, N$. With input gate and forget gate, each LSTM cell can learn to selectively forget its old memories and refresh with new inputs. In addition, the output gate o_t controls how much of the stored memory to be passed to the hidden state h_t.

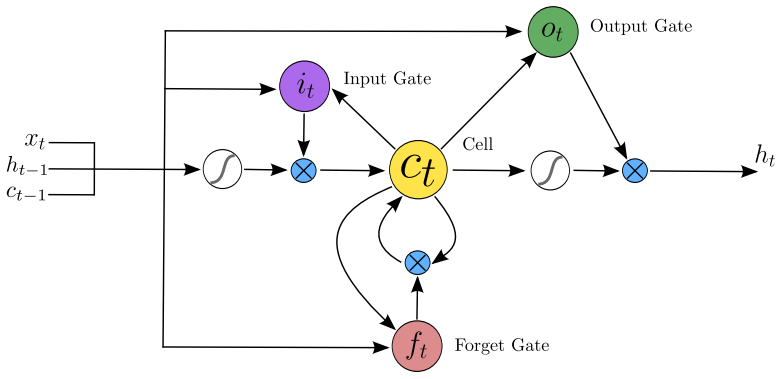

FIGURE 3. The architecture of an LSTM cell.

In this paper, we employ a single RNN layer with LSTM cells to process the features of each adjacent-frame pair extracted by the CNN. At each time step t, RNN will output features that contain information from the first frame until the t-th frame, which can exploit the temporal dynamics along the sequence. A Dropout layer with a drop rate of 0.5 is applied after the RNN layer during training process. The final regression layer is a fully-connected layer, which takes RNN features at time step t as input and estimates the homography $\hat{H}_{1,t+1}$ between the first frame and the $(t+1)$-th frame. The estimated homography $\hat{H}_{1,t+1}$ is an 8-dimensional vector, corresponding to the 8 values of 4-point homography parameterization.

3.4. **Loss function.** The proposed network is trained from scratch with Euclidean (L2) loss, which is shown to be effective for regressing homography in [12, 40]. Specifically, the network takes each input sequence of frame pairs $\{F_{1,2}, F_{2,3}, \cdots, F_{N,N+1}\}$ with the ground-truth homography sequence $\{H_{1,2}, H_{1,3}, \cdots, H_{1,N+1}\}$ as a training sample. For each sample, the loss is defined as

$$\mathcal{L} = \frac{1}{2N}\sum_{t=1}^{N} \|H_{1,t+1} - \hat{H}_{1,t+1}\|_2 \tag{3}$$

where $H_{1,t+1}$ and $\hat{H}_{1,t+1}$ are the ground-truth homography and estimated homography between the first frame and the $(t+1)$-th frame, respectively.

3.5. **Data generation.** As a deep-learning model, the proposed network requires large-scale training and testing videos with known ground-truth homographies. Previously, random homography transformation is used in many existing deep-learning methods [12, 8, 40] to construct large amount of image pairs with known ground-truth homographies for training the networks. However, this approach is not directly applicable for video-based homography estimation, because the independent and random warping between different images could not reflect the temporal continuity and dynamics in videos. In this section, we introduce a simple approach to synthesize large-scale videos with realistic ground-truth homographies. As shown in Fig. 4, we draw four fixed points on a flat white board and then use a hand-held camera to freely record an $(N+1)$-frame video for these 4 points. With clear background (i.e., white board), we can easily detect and track these four points over the video. We monitor the tracking progress and rectify the incorrectly detected locations with manual annotations. The tracked 4-point correspondence is used to calculate the ground-truth homographies $\{H_{1,2}, H_{1,3}, \cdots, H_{1,N+1}\}$ along the recorded video, as shown in Fig. 4.

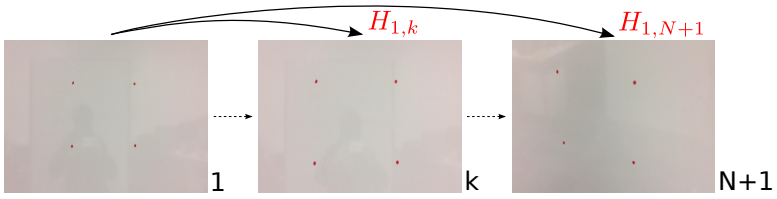

FIGURE 4. Sample frames of the 4 points in a recorded video, with computed homographies.

We then take a real image as the original image I_1, and apply transforms $\{H_{1,2}, H_{1,3}, \cdots, H_{1,N+1}\}$ to warp I_1 to construct warped images $I_2, \ldots, I_{N+1}$. This leads to a synthesized image sequence $I_1, \ldots, I_{N+1}$. More specifically, we make a uniform cropping, shown by blue boxes in Fig. 5, on both the original and the warped images to exclude blank areas as well as ensuring the identical size for all the generated frames along a sequence. Finally, the cropped regions are taken as the desired video consisting of frame sequence $F_1, \ldots, F_{N+1}$ with ground-truth homographies $\{H_{1,2}, H_{1,3}, \cdots, H_{1,N+1}\}$, which reflects temporal dynamics in real world. We can use different real images for I_1 and construct different ground-truth homographies by moving hand-held camera in different ways (i.e., with different tilt, pan and zoom) toward the four points. In the later experiments, we use images in MS-COCO [27] dataset for data generation. Some sample videos are shown in Fig. 6. We can interpret each sample video as observing a natural scene from one view to different views.

4. **Experiments.** In this section, we first describe the experimental setup, and then report the experimental results on the synthesized and real-world video datasets.

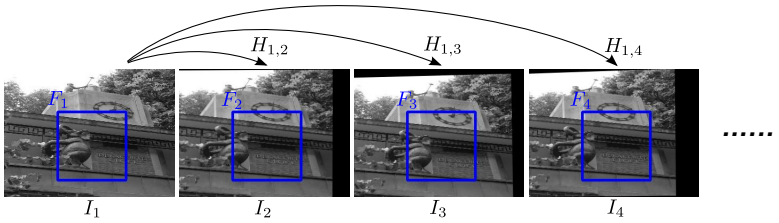

FIGURE 5. An illustration of constructing a video sequence with ground-truth homographies.

FIGURE 6. Sample videos generated using MS-COCO images with ground-truth homographies. Each row shows frames of a sample video.

4.1. **Experimental setup.** We implemented the proposed method via Keras [10], an open-source deep learning package. The parameters of the proposed recurrent convolutional regression network are initialized via He's method [17]. To train the network, we use the Adam optimizer [25] with the default parameters, except that ϵ is set to 0.1. The mini-batch size is set to 16. It takes approximately 20 hours to train our network for 200 epochs, on an NVIDIA Tesla P100 GPU. We monitor the validation loss to avoid over-fitting during the training process. The model that achieves the smallest validation loss is chosen for testing.

Following [12, 8, 40], we use the corner error as the metric for performance evaluation. The four corners of F_1 are transformed to $P_{kj}, j = 1, 2, 3, 4$ respectively using ground-truth homography $H_{1,k}$, and to $\hat{P}_{kj}, j = 1, 2, 3, 4$ respectively using estimated homography $\hat{H}_{1,k}$, where $k = 2, \ldots, N+1$. The corner error for the video is then

$$e = \frac{1}{N}\sum_{k=1}^{N} e_k = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{1}{4}\sum_{j=1}^{4}\|P_{kj} - \hat{P}_{kj}\|_2\right). \tag{4}$$

Lower corner error indicates better performance in homography estimation.

We compare the proposed method with nine existing homography estimation methods. SIFT+RANSAC, SURF+RANSAC and ORB+RANSAC are three conventional methods based on hand-crafted features SIFT, SURF and ORB respectively, all followed by DLT with RANSAC for homography estimation. We implemented them using the corresponding functions in the OpenCV library [7]. IC [2], ESM [6], and HA [23] are three template-matching based homography estimation methods, which are implemented based on the 2D template tracking code [32]. LK Tracker [31] is feature tracking based method, where tracked features are used for estimating homographies between adjacent or non-adjacent frames. It was implemented using the corresponding functions in the OpenCV library. HomographyNet is a deep-learning based method for homography estimation and we use the architecture and default setting as described in [12]. Graph-cut RANSAC [4] is a recent state-of-the-art homography-estimation approach, which runs a graph-cut algorithm in the local optimization step to separate inliers and outliers.

These comparison methods, except for HomographyNet, are unsupervised and we directly apply them on the testing data for evaluation. Furthermore, SIFT+RANSAC, SURF+ RANSAC, ORB+RANSAC, and HomographyNet are developed for directly estimating homographies between two images, e.g., any two frames in a video. We can also use them to only estimate homographies between adjacent frames and then compose them for homographies between non-adjacent frames [15]. We denote these methods with composing operations as SIFT+RANSAC*, SURF+RANSAC*, ORB+RANSAC* and HomographyNet*, respectively.

4.2. **Result on synthesized data.** We use the technique developed in Section 3.5 to generate synthesized video data for performance evaluation. An iPhone 6s is used as the hand-held video camera to record videos of four points on a white board for constructing ground-truth homographies, with 30fps frame rate and 1920×1080 frame resolution. We collect 9 videos of four points and each video contains 2200 frames. We annotate every other frame in each video sequence. Starting from the every frame in one video, we take 16 consecutive annotated frames as an one-second segment, in which frame resolution is down-sampled to 320×240. This way, we can compute 15 sequential homographies $H_{1,2}, H_{1,3}, \ldots, H_{1,16}$ for each video segment, via the getPerspectiveTransform function in OpenCV library, as the ground truth. The images that randomly chosen from MS-COCO dataset, are down-sampled to 320×240, and then taken as the original images for video generation. The final frames, i.e., the ones cropped by blue boxes in Fig. 5, have a size of 128×128. In total, we construct a dataset of 9 videos, in which each video contains 1000 short video sequences that contains 16 frames with ground-truth homographies. We randomly select 5 videos for training, 2 videos for testing, and 2 videos for validation, i.e., the training, validation and testing sets have 5000, 2,000 and 2,000 short video sequences, respectively.

We first study the impact of RNN by varying the number of LSTM cells in the proposed network. Specifically, we test four cases, and show the results in Table 1. We can see that the proposed network is capable of learning more effective features and produce smaller corner error (averaged over all the testing data in the synthesized dataset), by increasing the number of LSTM cells until it reaches 1,024. After that, the corner error increases if we further increase the number of LSTM cells, which we believe is caused by over-fitting. We use 1,024 LSTM cells for all the remaining experiments.

Method	Number of LSTM Memory Cells	Corner Error	Error reduction
Proposed	256	2.34	—
	512	1.44	38.5%
	1024	1.36	41.9%
	2048	1.37	41.4%

TABLE 1. Average corner error of the proposed method by using different numbers of LSTM cells.

The average corner errors of the proposed method and comparison methods on the testing videos of the synthesized dataset are summarized in Fig. 7. We can see that all the conventional methods based on hand-crafted features show comparable performances. ORB+RANSAC shows higher corner error than SIFT+RANSAC and SURF+RANSAC. One possible reason is that ORB does not have scale-invariance properties as SIFT and SURF. Graph-cut RANSAC obtains lower corner error than other methods based on RANS -AC, due to the technical improvement by incorporating graph-cut into RANSAC. Among the template matching methods, ESM has lower corner error than IC and HA. This may be caused by the fact that ESM has a higher convergence rate and does not need to compute Hessian [36]. LK tracker obtains lower corner error than other methods based on feature/template matching, due to the utilization of temporal continuity via feature tracking across frames. HomographyNet performs better than other comparison methods, which verifies the effectiveness of the learned deep features.

Composing homographies between adjacent frames for homographies between non-adjacent frames (four methods with superscript "*") produce large errors because of the error accumulated frame by frame. The proposed method achieves the best performance. Compared to HomographyNet, the proposed method reduces the corner error by 64.2% to 1.36. It verifies that the proposed method can effectively exploit the temporal dynamics across frames and learn good features for homography estimation along short videos.

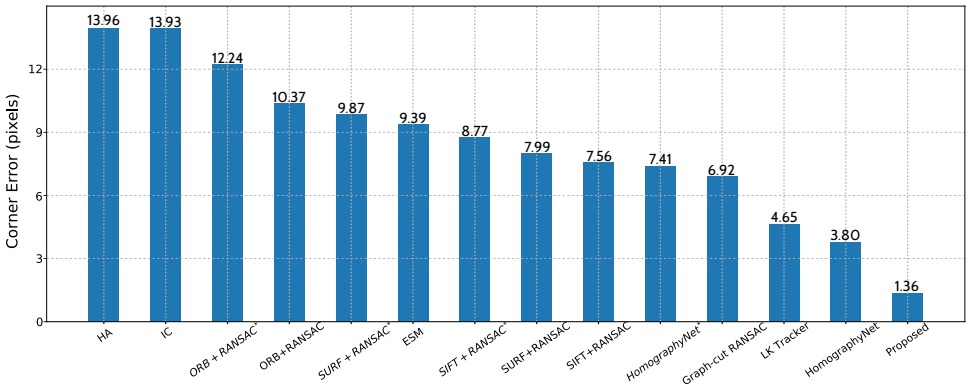

FIGURE 7. Comparison of the proposed method to the existing homography estimation methods on the synthesized video dataset.

We also present the complexity analysis of the proposed method in terms of the number of parameters and the number of floating-point operations (FLOPs) for processing one pair of frames, as shown in Table 2. We can see that the proposed method introduces 1.7M additional parameters and 16.8M more FLOPs, when

compared with the baseline method HomographyNet. These results show that the proposed method obtains much lower corner error without introducing too much computational cost.

Method	#Param.	FLOPs
HomographyNet	3.4M	68.4M
Proposed	5.1M	85.2M

TABLE 2. Complexity analysis, where #Param. denotes the number of parameters and FLOPs denotes the number of floating-point operations.

Figure 8 shows the corner error on the test videos over time – for each time k, we compute the average e_k over all the test data, where e_k is defined in Eq. 4. We can see that the corner error increases over time for all methods due to increased camera view angle changes over time. Four methods with superscript "*" produce large accumulation errors in composing adjacent-frame homographies. The corner error of the proposed method is lower than all the comparison methods at each time. Furthermore, the performance of the proposed method decreases much slower than those of all the comparison methods over time. This results from the usage of the RNN network with LSTM cells in the proposed method, which effectively exploits the temporal dynamics and contributes to better homography estimation.

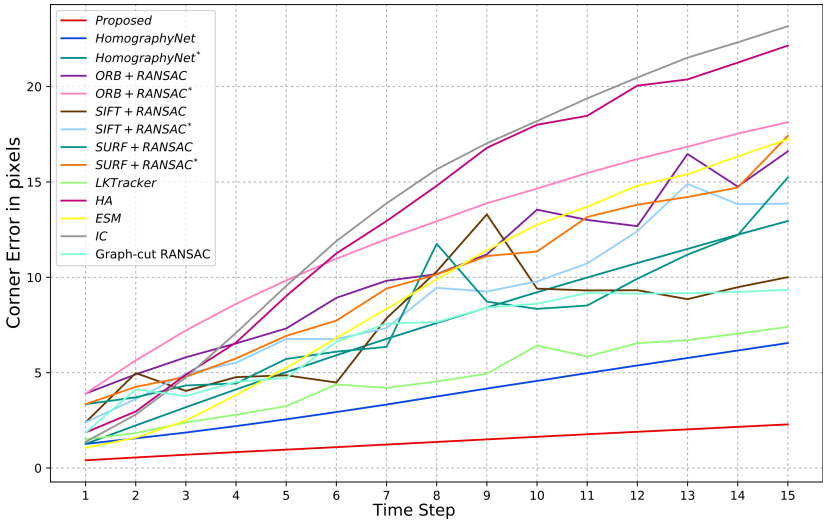

FIGURE 8. Corner errors of the proposed method and the comparison methods over time on the synthesized video dataset.

We also conduct an experiment to test the robustness of the proposed method against two kinds of challenges which are highly likely to be present in real world. Specifically, we add color variations and/or Gaussian noise on the original dataset, to generate three synthesized video datasets: with color variations, with Gaussian noise and with both. The original dataset and these three datasets are denoted as "Original", "Color variations", "Gaussian noise" and "Both" respectively. For color variation, we randomly enhance the contrast, brightness and color with an

enhancement amount between 0.5 and 1.5 in a random order for a whole sequence, following Howard [18]. For Gaussian noise, we apply it with zero mean and a standard variance of 0.02 on each sequence. We train a new model on the training data of the dataset "Both". The original model and the newly trained model are denoted as $model_{orig}$ and $model_{both}$ respectively.

Model \ Test Set	Original	Color variations	Gaussian noise	Both
$model_{orig}$	1.36	1.41	2.18	2.42
$model_{both}$	1.79	1.88	1.65	1.69

TABLE 3. Performance of the proposed method trained the "Original" dataset and the "Both" dataset, tested on the "Original", "Color variations", "Gaussian noise" and "Both" datasets.

The two models are tested on the test sets of all four datasets. The resulting corner errors are summarized in Table 3. $model_{orig}$ performs worse on the "Color variations", "Gaussian noise" and "Both" datasets than on the "Original" dataset, while $model_{both}$ performs worse on the "Color variations" and "Original" datasets than on the "Both" dataset. This is due to the domain shift brought by color variations and/or Gaussian noise. Obviously, $model_{orig}$ is more sensitive to Gaussian noise than color variations. $model_{both}$ performs better on the "Gaussian noise" dataset than on the "Color variations" dataset. Since the "Both" dataset differs from the "Color variations" dataset by Gaussian noise, and differs from the "Gaussian noise" dataset by color variations, we can conclude that $model_{both}$ is also more sensitive to Gaussian noise. We believe that batch normalization in the proposed network can address some domain shift from color variation, while Gaussian noise is more difficult to be handled. Note that, $model_{both}$ achieves lower corner error on the "Gaussian noise" dataset than on the "Both" dataset. This may be caused by the randomness of the network initialization.

4.3. **Results on real-world data.** We also evaluate the proposed method on a real-world video dataset [26]. This dataset is designed for planar object tracking, and captured in the wild scenario rather than the constrained laboratory environment. The videos are annotated for every other frame, and the ground-truth homographies are calculated by the labeled positions of a planar object in each frame. The resolution of each frame is 1280×720. Each video is recorded with 30 fps frame rate. Since the proposed method estimates homography based on the entire frames, we carefully choose all the frames in which the entire scenes are planar surfaces, i.e., the ground-truth homographies are suitable for the whole frames. As on synthesized data, starting from every frame, we take 16 consecutive frames as a segment, with a sequence of 15 ground-truth homographies for the segment. In total, we generate 335 non-overlapping video segments. This dataset involves several challenging factors, including lighting variation, scale change, perspective distortion, occlusion, and out-of-view, regarding the homography transformation of the planar object. Samples of the real-world videos are shown in Fig. 9.

In the experiment, the frames are first down-sampled to 128×128 and converted to gray-scale, in order to fit the input size of the proposed network. Then, we apply a three-fold cross validation scheme for evaluation. We randomly divide the generated (real-world) video dataset into three subsets of same size, and each time,

two subsets are used for training and the remaining subset is used for testing. For training, we initialize the parameters of the proposed network with the weights pre-trained on the synthesized video dataset. The same optimizer and mini-batch size as in Section 4.2 are used for fine-tuning. It takes about 150 epoch (20 minutes) to train the network until the training loss converges, on an NVIDIA Tesla P100 GPU.

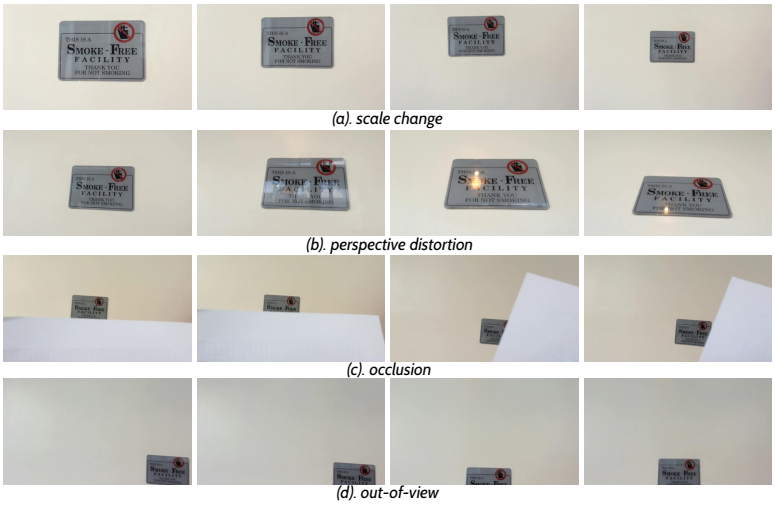

FIGURE 9. Sample real-world videos with ground-truth homographies. Each row shows a video with an observed challenge.

On real-world video data, the performance of the proposed method and comparison methods are shown in Fig. 10. Here the corner error is calculated by using the four corners of the planar object in each frame. Note that, all the methods, except for HomographyNet and the proposed method, compute homographies by using input videos with their original size. Therefore, for HomographyNet and the proposed method, we map their estimated corners to the original scale, and then calculate the corner error for fair comparison with other methods. From Fig. 10 we can see, the template matching methods achieve higher corner error than the hand-crafted feature methods. One possible reason is that, the template matching methods need to match the whole template, which could be highly affected by the cases of occlusion and out-of-view, while the hand-crafted feature methods do not rely on the whole template. Graph-cut RANSAC obtains lower corner error than the deep learning method HomographyNet, and shows better performance than other comparison methods. This may be due to its superiority over the original RANSAC by better separating the inliers and outliers among matched feature points via graph-cut algorithm. The proposed method outperforms all the comparison methods, and reduces the corner error by 40.7% to 7.49, when compared to HomographyNet. Again, on the real-world data, these results verify that the proposed method can learn visual features more effectively, via exploiting the temporal dynamics across frames.

We also evaluate the corner error over time for the proposed method and the comparison methods on the real-world data. As shown in Fig. 11, all methods have certain levels of performance degradations over time, due to the change of lighting

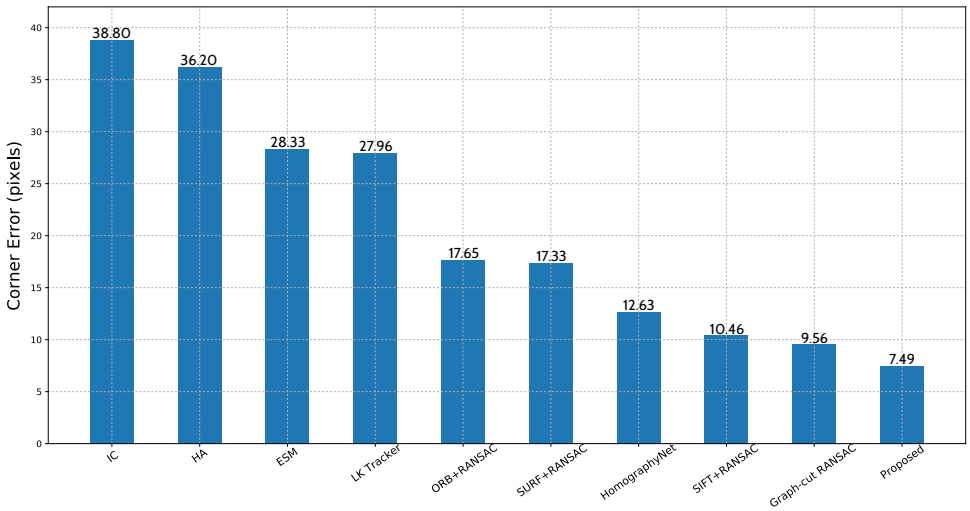

FIGURE 10. Performance of the proposed method and the comparison methods on the real-world video dataset.

and camera view angle resulted from the camera movement. The proposed method still outperforms all the comparison methods at every time step, and achieves much smaller performance degradation over time. The result proves the effectiveness of the usage of the RNN with LSTM cells in the proposed network, which can exploit the temporal dynamics and lead to more accurate homography estimation.

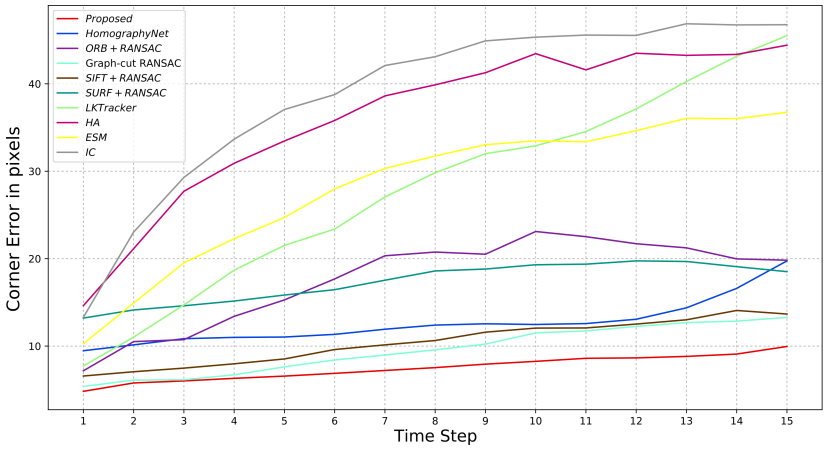

FIGURE 11. Corner errors of the proposed method and the comparison methods over time on the real-world video dataset.

5. **Conclusion.** In this paper, we propose a novel method, namely recurrent convolutional regression network, to estimate the homographies along a short video. The proposed network consists of CNN, RNN with LSTM cells and a regression layer to predict homographies sequentially. To train the network, we introduce a simple but effective approach to synthesize large video dataset with ground-truth

homographies. We evaluate the proposed method on both the synthesized and real-world video datasets. The experimental results on these datasets verify that, by exploiting the temporal dynamics across frames via RNN with LSTM cells, the proposed method can estimate homographies along short videos, better than several existing methods.

REFERENCES

[1] S. Baker, A. Datta and T. Kanade, Parameterizing homographies, in *Tech. Report, CMU-RI-TR-06-11*, Robotics Institute, Carnegie Mellon University, (2006).

[2] S. Baker and I. Matthews, Lucas-Kanade 20 years on: A unifying framework, *International Journal of Computer Vision*, **56** (2004), 221–255.

[3] D. Barath and Z. Kukelova, Homography from two orientation- and scale-covariant features, in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), (2019), 1091–1099.

[4] D. Barath and J. Matas, Graph-cut RANSAC, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, (2018), 6733–6741.

[5] H. Bay, T. Tuytelaars and L. Van Gool, SURF: Speeded up robust features, in *Computer Vision – ECCV 2006*, Lecture Notes in Computer Science, 3951, Springer, Berlin, Heidelberg, (2006), 404–417.

[6] S. Benhimane and E. Malis, Real-time image-based tracking of planes using efficient second-order minimization, *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, **1** (2004), 943–948.

[7] G. Bradski, The OpenCV Library, *Dr. Dobb's Journal of Software Tools*.

[8] C. Chang, C. Chou and E. Y. Chang, CLKN: Cascaded Lucas-Kanade networks for image alignment, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, (2017), 3777–3785.

[9] F. Chhaya, D. Reddy, S. Upadhyay, V. Chari, M. Z. Zia and K. M. Krishna, Monocular reconstruction of vehicles: Combining SLAM with shape priors, in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, (2016), 5758–5765.

[10] F. Chollet et al., Keras, `https://keras.io`, 2015.

[11] N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, **1** (2005), 886–893.

[12] D. DeTone, T. Malisiewicz and A. Rabinovich, Deep image homography estimation, preprint, `arXiv:1606.03798`.

[13] M. A. Fischler and R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Comm. ACM*, **24** (1981), 381–395.

[14] C. Forster, M. Pizzoli and D. Scaramuzza, SVO: Fast semi-direct monocular visual odometry, in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, (2014), 15–22.

[15] E. Garcia-Fidalgo, A. Ortiz, F. Bonnin-Pascual and J. P. Company, A mosaicing approach for vessel visual inspection using a micro-aerial vehicle, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, (2015), 104–110.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2^{nd} edition, Cambridge University Press, Cambridge, 2003.

[17] K. He, X. Zhang, S. Ren and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, (2015), 1026–1034.

[18] A. G. Howard, Some improvements on deep convolutional neural network based image classification, preprint, `arXiv:1312.5402`.

[19] Y.-F. Hsu, C.-C. Chou and M.-Y. Shih, Moving camera video stabilization using homography consistency, in *2012 19th IEEE International Conference on Image Processing*, Orlando, FL, (2012), 2761–2764.

[20] M.-D. Hua, T. Hamel, R. Mahony and G. Allibert, Explicit complementary observer design on special linear group SL(3) for homography estimation using conic correspondences, in *2017*

IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, (2017), 2434–2441.

[21] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proceedings of the 32 nd International Conference on Machine Learning*, Lille, France, (2015), 448–456.

[22] W. Jiang and J. Gu, Video stitching with spatial-temporal content-preserving warping, in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Boston, MA, (2015), 42–48.

[23] F. Jurie and M. Dhome, Hyperplane approximation for template matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24** (2002), 996–1000.

[24] A. Kendall, M. Grimes and R. Cipolla, PoseNet: A convolutional network for real-time 6-DOF camera relocalization, in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, (2015), 2938–2946.

[25] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980.

[26] P. Liang, Y. Wu, H. Lu, L. Wang, C. Liao and H. Ling, Planar object tracking in the wild: A benchmark, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, (2018), 651–658.

[27] T.-Y. Lin, et al., Microsoft COCO: Common objects in context, in *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, 8693, Springer, Cham, (2014), 740–755.

[28] S. Liu, L. Yuan, P. Tan and J. Sun, SteadyFlow: Spatially smooth optical flow for video stabilization, in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, (2014), 4209–4216.

[29] S. Liu, J. Chen, C.-H. Chang and Y. Ai, A new accurate and fast homography computation algorithm for sports and traffic video analysis, *IEEE Transactions on Circuits and Systems for Video Technology* **28** (2018), 2993–3006.

[30] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, **60** (2004), 91–110.

[31] B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision, *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 2, Morgan Publishers Inc., San Francisco, CA, 1981, 674–679.

[32] C. Mei, S. Benhimane, E. Malis and P. Rives, Efficient homography-based tracking and 3-D reconstruction for single-viewpoint sensors, *IEEE Transactions on Robotics*, **24** (2008), 1352–1364.

[33] Y. Mi, K. Zheng and S. Wang, Recognizing actions in wearable-camera videos by training classifiers on fixed-camera videos, in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ICMR '18, Association for Computing Machinery, New York, NY, (2018), 169–177.

[34] Y. Mi and S. Wang, Recognizing micro actions in videos: learning motion details via segment-level temporal pyramid, in *2019 IEEE International Conference on Multimedia and Expo*, Shanghai, China, (2019), 1036-1041.

[35] Y. Mi, X. Zhang, Z. Li and S. Wang, Dual-branch network with a subtle motion detector for microaction recognition in videos, in *IEEE Transactions on Image Processing*, **29** (2020), 6194-6208.

[36] K. Mikolajczyk, et al. A comparison of affine region detectors, *Int. J. Comput. Vision*, **65** (2005), 43–72.

[37] M. Muja and D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in *VISAPP International Conference on Computer Vision Theory and Applications*, (2009), 331–340.

[38] V. Nair and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of ICML*, 27, Haifa, Isael, (2010), 807–814.

[39] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor and V. Kumar, Unsupervised deep homography: A fast and robust homography estimation model, *IEEE Robotics and Automation Letters*, **3** (2018), 2346–2353.

[40] F. E. Nowruzi, R. Laganiere and N. Japkowicz, Homography estimation from image pairs with hierarchical convolutional networks, in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Venice, (2017), 904–911.

[41] M. Ozuysal, M. Calonder, V. Lepetit and P. Fua, Fast keypoint recognition using random ferns, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32** (2010), 448–461.

[42] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, ORB: An efficient alternative to SIFT or SURF, in *2011 International Conference on Computer Vision*, Barcelona, (2011), 2564–2571.
[43] J. Shi and Tomasi, Good features to track, in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, (1994), 593–600.
[44] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, preprint, arXiv:1409.1556.
[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.*, **15** (2014), 1929–1958.
[46] D. Tao, Y. Guo, B. Yu, J. Pang and Z. Yu, Deep multi-view feature learning for person re-identification, *IEEE Transactions on Circuits and Systems for Video Technology*, **28** (2018), 2657–2666.
[47] D. Teney and M. Hebert, Learning to extract motion from videos in convolutional neural networks, in *Computer Vision – ACCV 2016*, Lecture Notes in Computer Science, 10115, Springer, Cham, (2016), 412–428.
[48] X. Yang, X. Si, T. Xue, L. Zhang and K.-T. T. Cheng, Vision-inertial hybrid tracking for robust and efficient augmented reality on smartphones, in *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, Association for Computing Machinery, New York, NY, (2015), 1039–1042.

Received December 2019; revised March 2020.

E-mail address: miy@email.sc.edu
E-mail address: zheng37@email.sc.edu
E-mail address: songwang@cec.sc.edu