This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Unrolling the Shutter: CNN to Correct Motion Distortions

Vijay Rengarajan[1*], Yogesh Balaji[2†], A.N. Rajagopalan[3]
[1,3]Indian Institute of Technology Madras, [2]University of Maryland
[*]vijay.ap@ee.iitm.ac.in

Abstract

Row-wise exposure delay present in CMOS cameras is responsible for skew and curvature distortions known as the rolling shutter (RS) effect while imaging under camera motion. Existing RS correction methods resort to using multiple images or tailor scene-specific correction schemes. We propose a convolutional neural network (CNN) architecture that automatically learns essential scene features from a single RS image to estimate the row-wise camera motion and undo RS distortions back to the time of first-row exposure. We employ long rectangular kernels to specifically learn the effects produced by the row-wise exposure. Experiments reveal that our proposed architecture performs better than the conventional CNN employing square kernels. Our single-image correction method fares well even operating in a frame-by-frame manner against video-based methods and performs better than scene-specific correction schemes even under challenging situations.

1. Introduction

The delay in the start of the exposure between the first row and the last row of the sensor array (total line delay) in CMOS cameras causes rolling shutter (RS) distortions. In the presence of camera motion, each row experiences different camera poses unlike in a global shutter camera, which causes skew and curvature in the recorded image. Irrespective of whether an image or a video is captured, the camera motion inevitably causes distortions; though the type of distortion varies depending on the exposure duration as compared to the total line delay, and this leads to the need for different correction methods as shown in Fig. 1 (top). Short exposure time causes only the RS effect, while medium to long exposure causes motion blur too. In this work, we study the short exposure scenario.

While most existing works [8, 14, 3, 10] deal with video RS correction in short exposure setting, the task is highly challenging for the data-starved situation comprising of only a *single* image. This situation is very much plausible given the prevalent hand-held on-the-go imaging using mobile phones. Other sources of information in the camera can be tapped to facilitate such a data-deserted scenario. The motion information provided by the gyroscope can be used for RS correction post-capture, but it is heavily limited by the sparsity of gyro-samples within the short exposure especially for single-image correction. Electronic image stabilizer (EIS) such as the one present in a Google Pixel phone camera needs multiple frames making it inapt for single image capture. Optical image stabilizer (OIS) such as in an iPhone camera can tackle only small motions, and its main application is in videos and long exposure images.

To handle RS motion distortions, one needs to understand from where the uneasiness of human visualization comes about. In videos, it is due to the local structural changes along the temporal axis (through frames); in a blurred image, it arises due to the unsharpness of edges, while in a single RS distorted image, it is due to the structural changes compared to a human preconceived perception of the scene. Hence, as shown in Fig. 1 (bottom), different types of information can be utilized for these various

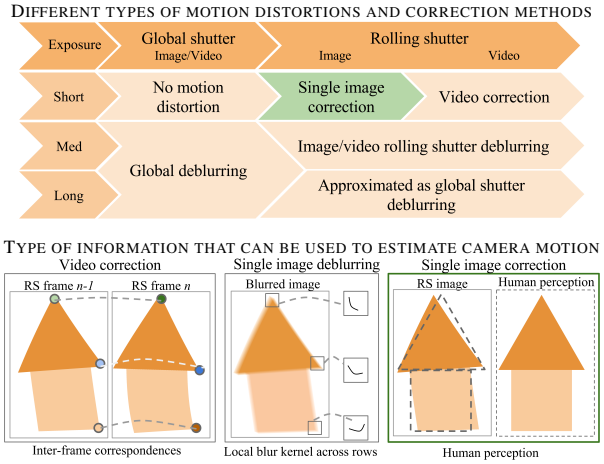


Figure 1. Overview of correction techniques for various image capture and exposure settings.

[†]This work was done when the second author was studying at IIT Madras.

Figure 2. *Top*: Rolling shutter distorted images captured using mobile phones. *Bottom*: Corrected images using our CNN-based method.

correction methods. Frame-to-frame correspondences are used for video correction, while local blur kernels are used in the blurred case. In our method, for single image RS correction, with no other extra information available, we have chosen human perception as our utility.

Video correction methods: In these works, the wobbly effect between frames are corrected and stabilized for better visual output. Important works include block-wise optical-flow-like model of [8], block-wise seven-parameter model of [6], motion interpolation-based model of [14], homography mixture based model of [3] and spline-based model of [10]. All these models utilize inter-frame correspondences (based on intensity in [8] and [6], and features in others) to estimate the camera trajectory and register frames. They also leverage the continuity and smoothness of camera motion between video frames.

Image correction methods: Few works indeed study RS from only a single image. The RS deblurring work of [19] uses local blur kernels to fit a continuous camera motion. In the absence of blur, the work of [4] corrects the RS distortions from face images using facial key points as features, and is restricted to skew-type RS distortion. Using a similar inspiration for urban scenes, the method of [13] corrects the RS effect from urban images using curves as features. These two algorithmic methods are tailored for specific image classes and are thus heavily dependent on the extraction of their respective scene-specific features.

Proposed method: In this work, we automatically learn features of scene classes using convolutional neural networks (CNN) to estimate the underlying camera motion from a single RS-affected image and finally correct the distortions. In contrast to manual selection and extraction of features in scene-specific methods which can be an unfruitful exercise, CNNs can learn desired features essential to correct distortions for a particular class by themselves. Camera motion distortions causing motion blur have been studied using CNNs so far. Apart from the works that perform non-blind deblurring [16, 15, 22], the work of [20] takes a classification approach for blur kernel estimation, and recently, the work of [2] regresses on the blur kernel directly. All these methods learn blur information from local image patches, while to learn the motion trajectory that causes the RS effect in the absence of any blur, one needs to extract and combine information from different parts of the image. The effect of RS either with or without motion blur has not been studied using neural networks.

In our work, the interplay between the scene structure and the row-wise camera motion is learned using a neural network that employs long kernel features. Our design first extracts basic image features using square-convolutional layers, followed by two banks of feature-interactive layers employing row-kernel and column-kernel convolutions to extract properties along horizontal and vertical directions, specifically addressing the nature of the RS effect. Finally, these directional features are combined using fully-connected layers to arrive at the RS motion.

We train the network using synthetic RS images regressing for the intra-image camera motion. We do not regress directly on the image, for example using generative networks [12, 9], since we do not seek better or new information but only geometric unwarping. Once our trained network predicts the motion, we correct the RS image using local warping. Fig. 2 shows our corrected outputs of distorted images captured using mobile phones. Skew and curvature distortions are corrected in all these examples.

Main contributions

- A method that rectifies the rolling shutter effect from a single image without tailored extraction of specific scene features and the knowledge of camera parameters.
- A new CNN architecture designed specific to the exposure mechanism in rolling shutter cameras that learns row-oriented and column-oriented features.

We neglect depth-dependent motion and lens distortions.

2. Rolling Shutter Model

A static CMOS rolling shutter camera captures the same image as that of captured using a global shutter camera. This is referred to as the global shutter image, $\mathbf{I}_{GS}$. When the camera moves during exposure, each row of sensors experiences different camera pose due to the row-wise acquisition resulting in local image warping. The observed distorted image is referred to as the rolling shutter image $\mathbf{I}_{RS}$.

Let $[t_x(y), t_y(y), t_z(y), r_x(y), r_y(y), r_z(y)]$ denote the camera trajectory vector observed by the row y of the RS image, where t denotes translations, and r denotes rotations. To correct the distortion from an RS image having M rows, we need to estimate $6M$ parameters corresponding to six camera poses for each row. The questions that we ask are what type of motion information can a single image provide, and what further restrictions does the assumption of unknown camera intrinsics lead to. To answer these questions, we observe the RS effects produced by different types of camera motion on a single image.

RS effects produced by camera motion Fig. 3 illustrates the local distortions produced by different types of RS motion. Although each motion type creates its own kind of distortion, there are similarities between some of them. Some prominent effects are noted as follows:

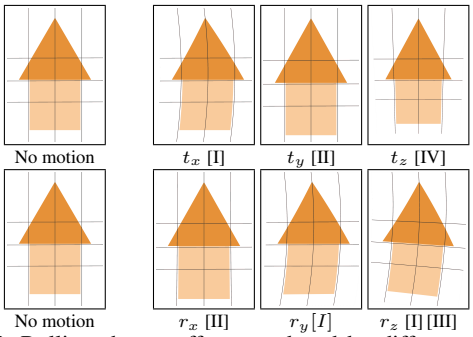


Figure 3. Rolling shutter effects produced by different types of camera motion.

[I] Vertical curvature: The translation t_x (along the horizontal axis) and the rotation r_y (around the vertical axis) produce similar RS effect, translating points on a row horizontally. This manifests as curvature in vertical lines. For t_x, all points on a row move by the same amount, whereas for r_y, it varies slightly depending on the camera focal length. For high focal lengths, all points on a row move by almost the same amount.

[II] Vertical stretch/shrinking: Along the same lines, t_y and r_x produce similar kind of RS effect, displacing points vertically which results in stretching or shrinking of vertical structures. Note the elongation of the house in the vertical direction for t_y and r_x.

[III] Horizontal curvature: The in-plane rotation r_z bends both vertical and horizontal lines. It is important to note that the same r_z affect the points on the same row differently based on their distances from the rotation center. Hence, different vertical lines produce different curvatures due to r_z, which is unlike t_x that affects all vertical lines in the same manner irrespective of their column locations.

[IV] Vertical scale change: The optical axis translation t_z slants the vertical lines located to either side of the vertical axis only a little, since the camera motion away from/towards the scene has to be very high within a single exposure to create pronounced curvatures.

Motions considered for RS correction The disturbance of straightness is visually unpleasant if it goes against human preconception. We rank the four distortions as follows based on their undesirability: [III],[I],[II],[IV]. Humans are more reactive to vertical and horizontal curvatures [I,III] than that of vertical stretching/shrinking [II]. Optical axis translation during a single image capture creates negligible distortions, and thus, [IV] can be safely ignored. With unknown camera intrinsics, we also approximate the effect due to r_y as that caused by t_x. Therefore, we consider only the motions t_x and r_z in our model and ignore the others. We write the mapping from a 2D point $\mathbf{x}_{GS}$ on $\mathbf{I}_{GS}$ to the point $\mathbf{x}_{RS}$ on row y of $\mathbf{I}_{RS}$ as a 2D point transformation:

$$\begin{aligned}\mathbf{x}_{RS} &= \begin{bmatrix}\cos r_z(y) & -\sin r_z(y)\\ \sin r_z(y) & \cos r_z(y)\end{bmatrix}\mathbf{x}_{GS} + \begin{bmatrix}t_x(y)\\ 0\end{bmatrix},\\ &\equiv \mathbf{R}(y)\mathbf{x}_{GS} + \mathbf{t}(y), \end{aligned} \quad (1)$$

where $\mathbf{R}(y)$ is the 2D z-axis rotation matrix for row y and $\mathbf{t}(y)$ is the 2D translation vector with zero y-axis motion. Note that the unit of $t_x(y)$ is pixels in (1), and there is no dependence on the camera intrinsic matrix. From this point onwards, by translation, we mean t_x, and by rotation, we mean r_z. It is important to note that the 2D motion model in (1) is indeed row-wise, and hence it can model even high visual distortions and not just a simple affine transformation as would be the case for global 2D motion.

Motion trajectory model To express the camera trajectory through the row exposures, we define the camera motion as two vectors, one each for translation and rotation, given by $\mathbf{p}_1 = [t_x(1), t_x(2), \ldots, t_x(M)]$ and $\mathbf{p}_2 = [r_z(1), r_z(2), \ldots, r_z(M)]$. For a row $y \in [1, M]$ of the distorted image, the camera pose is $[t_x(y), r_z(y)]$, and thus, each distorted image $\mathbf{I}_{RS}$ is associated with a camera trajectory tensor $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2]$ having $2M$ values. To correct the distortion by undoing the motion, one needs to estimate $2M$ unknowns, or equivalently M camera poses, from a single image.

Since estimating M poses from a single image is very ill-posed, we leverage the short exposure time setting that we operate on to model the camera motion by a polynomial trajectory. To verify this assumption, we use the human-recorded handshake trajectory dataset of [7]. We fit an n-th degree polynomial to t_x and r_z motions in [7]. Fig. 4(top) shows two sample trajectory plots with blue circles denoting the recorded camera poses during the exposure and red dotted lines representing the fitted polynomial trajectory. We fit polynomials of different degrees to the recorded pose samples and observed that the average fitting error almost converges after $n = 3$ as shown in Fig. 4(bottom).

Therefore, we model the translation and rotation trajectories as polynomials with respect to the row number. The

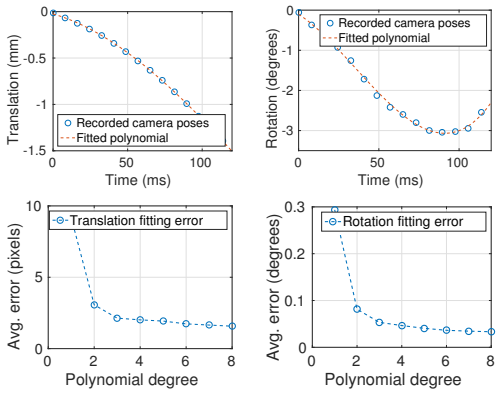


Figure 4. Polynomial trajectory fitting in real camerashake dataset.

camera poses at each row index $y \in [1, M]$, we have

$$p_i(y) = \alpha_{i0} + \sum_{j=1}^{n} \alpha_{ij} ((y-1)/M)^j, i = 1, 2, \quad (2)$$

where $p_1(y) = t_x(y)$, $p_2(y) = r_z(y)$, α_{ij} is the j^{th}-degree polynomial coefficient for the i^{th} motion, and $n = 3$.

3. Rolling Shutter Correction

Fig. 5 provides an overview of our method. There are three modules: a neural network for camera motion estimation, trajectory fitting to get row-wise motion, and image correction using the estimated camera motion. The input to our method is a single RGB RS-distorted image, and the output is the corresponding corrected image.

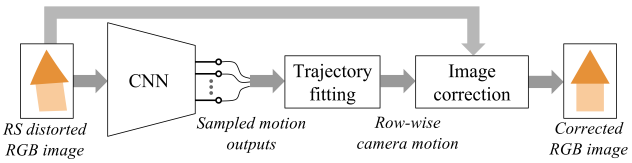


Figure 5. Overview of rolling shutter correction.

3.1. Camera Motion Estimation using CNN

In this work, we treat the problem of camera motion estimation as one of regression, where we train and use a function $\psi(\mathbf{I}_{RS}; \boldsymbol{\theta})$ to predict the camera trajectory tensor $\mathbf{P}^*$, where $\boldsymbol{\theta}$ represents the weights (parameters) of the system.

$$\mathbf{P}^* = \psi(\mathbf{I}_{RS}; \boldsymbol{\theta}) \quad (3)$$

The power and complexity of the proposed method is in ψ which is based on CNN that extracts information from images to output the camera motion. CNNs allow us to do away with the laborious manual design of algorithms to pick correct features of the scene that are distorted in the image to aid in the camera motion estimation. For face images, [4] carefully chooses facial keypoints required for RS correction, while for correction of urban scenes, [13] chooses curves and lines as features.

RS correction is essentially local image warping undoing the geometric distortion; no new or better image information (in the sense of applications such as denoising and super-resolution) is sought. Hence, we regress on the camera motion instead of directly on the image. Further, generative models such as general adversarial networks [12, 9], which could directly learn to output the undistorted image, are usually limited by the visual quality of the image output. In our method, we correct the distorted image geometrically using the motion estimate from the CNN.

Instead of learning to estimate the camera pose for every row, we tap only the motion of K equally spaced rows as CNN outputs since the motion lies in a lower dimensional space as shown earlier. The size of each sampled $\mathbf{p}_1^s$ and $\mathbf{p}_2^s$ is K, making the length of the output camera trajectory tensor $\mathbf{P}^*$ of the CNN in Fig. 5 as $N = 2K$. In our CNNs, the input is a $256 \times 256 \times 3$ RGB image and the output is a 30-length motion vector (corresponding to $K = 15$).

VanillaCNN We propose two CNN architectures in this work: the first one is *VanillaCNN* as shown in Fig. 6(top). It uses standard convolutional and pooling layers, in which square kernels extract and combine local information from the RS image to deduce the camera motion. Out of the seven layers, the first four convolutional layers consist of square filters, the outputs of which are passed on to ReLU units followed by max-pooling over 2×2 non-overlapping cells. The last three are fully connected layers; the first of the three uses Tanh, the second uses HardTanh, and the final, none.

RowColCNN Obtaining and combining local information from different parts of the image is a crucial aspect to learn the RS motion. Our motivation for a new architecture stems from the following observations:

(i) temporal motion information is present along image columns,

(ii) information from image rows helps to reinforce row-wise motion constancy, and

(iii) rotation can be better estimated if information from image rows are extracted earlier since it affects left and right areas of an image row differently.

Hence, we branch out VanillaCNN after feature extraction from the initial square-convolutional layers into two banks. The column kernel bank employs filters whose effective support spans longer along the column, while the row kernel bank employs row-oriented filters. Both these banks extract locally oriented information and combine them in their own fully connected layers, before propagating them to the final fully connected layers which have the same nonlinearities as those of VanillaCNN. The two 4096-vectors from the banks are first added, and then passed on to the nonlin-

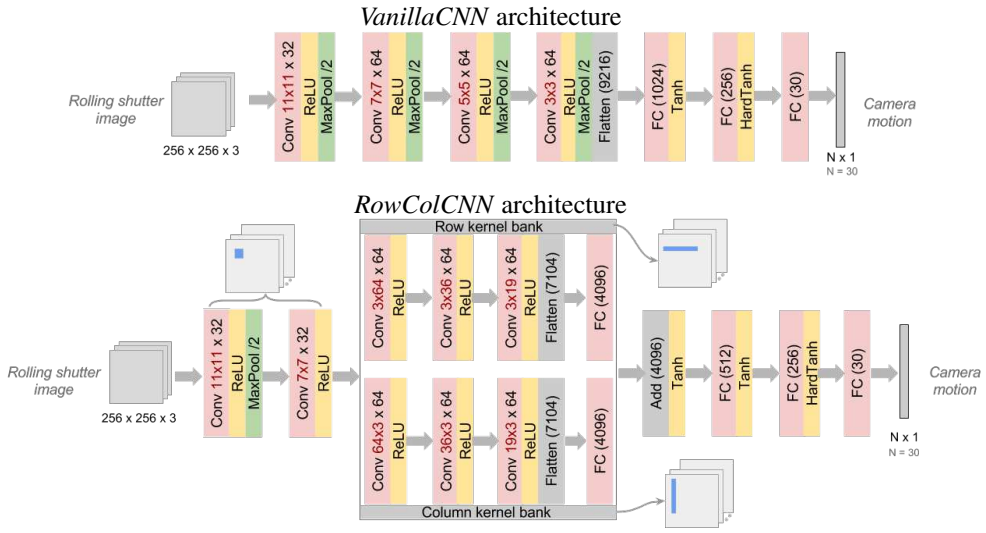


Figure 6. Proposed architectures for rolling shutter motion estimation. All convolution layers use valid pixel convolution and all maxpooling layers use 2x2 window with a stride of 2.

earity leading to the single 4096-vector at the start of the final fully connected layers. We call this architecture *RowColCNN* which is shown in Fig. 6(bottom).

Training Given a set of S labeled images $\{(\mathbf{I}_i, \mathbf{P}_i)\}$, the parameters $\boldsymbol{\theta}$ of the model in (3) are estimated by minimizing the loss between the predicted trajectory vector $\mathbf{P}^*$ and the ground truth trajectory vector $\mathbf{P}$ as given in (4). We use mean square error as our loss function and sampling from a uniform distribution for initialization of weights during our training. We train using stochastic gradient descent with a learning rate of 0.05.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{S} \sum_{i=1}^{S} \|\psi(\mathbf{I}_i; \boldsymbol{\theta}) - \mathbf{P}_i\|_2^2 \qquad (4)$$

3.2. Prediction and Correction

During the motion prediction phase, the input RS image is forwarded through the trained network to output K-length translation and rotation vectors (N values). These two vectors are then fit using a third-degree polynomial to obtain the estimated motion values for each row $\mathbf{p}_1^*$ and $\mathbf{p}_2^*$ as in (2). The estimated trajectory tensor is then given by $\mathbf{P}^* = [\mathbf{p}_1^*, \mathbf{p}_2^*]$.

Once the full camera trajectory is estimated, the RS image must be corrected back in time to the first-row exposure (i.e. the global shutter image). We first subtract the pose of the first row from those of all rows, leading to an identity transformation for the first row and all the remaining rows having warps with respect to it. The dewarping or distortion correction is done using a forward mapping where for each pixel coordinate of the corrected (GS) image, we pick an intensity from the distorted (RS) image. For every pixel $\mathbf{x}_{GS}$, we find a y^* for which warping $\mathbf{x}_{GS}$ using $\mathbf{P}^*(y^*)$ takes it to an $\mathbf{x}_{RS}$ having a row coordinate closest to y^* [13]:

$$y^* = \arg\min_{y} \| [\mathbf{x}_{RS}]_{row} - [\mathbf{R}^*(y)\mathbf{x}_{GS} + \mathbf{t}^*(y)]_{row} \|_2^2 \quad (5)$$

where $\mathbf{R}^*(y)$ is the rotation matrix corresponding to $r_z^*(y)$ and $\mathbf{t}^*(y) = [t_x^*(y), 0]^T$. Finally, the intensity at pixel $\mathbf{x}_{GS}$ on the GS image is copied from the location $\mathbf{R}^{*T}(y^*)(\mathbf{x}_{RS} - \mathbf{t}^*(y^*))$ of the RS image.

4. Experiments

The experiments section is arranged as follows: (i) description of comparison methods, (ii) creation of training and testing datasets, (iii) quantitative results and comparisons, and (iv) visual results and comparisons.

4.1. Comparison Methods

CNN models We perform comparisons between the two proposed network architectures – VanillaCNN and RowColCNN. We evaluate the effectiveness of both the architectures on different datasets using various metrics.

Video models In non-learning based methods, we first compare with two contemporary RS video correction methods of [14] and [3]. Since our method is single-image based, we cannot directly use these video methods for comparison. Therefore, for quantitative comparisons, we feed both the reference undistorted GS and distorted RS images as inputs to them. We then use our own two-image implementation of their frameworks to correct the RS image. These two reference-based correction schemes are used as the baseline in our experiments. For visual video comparisons, we employ our method frame-by-frame on the videos used in their works and compare with their outputs.

Single-image models We also compare with [13] and [4], which address RS correction of urban scenes and faces, respectively. We sent our images to the authors of [13] and got back their results. We used our own implementation for [4]. We gauge the outputs of these two methods and our method against the reference-based baseline outputs as described in the previous paragraph.

4.2. Dataset Creation

We use 256×256 as the size of both the input and output images. We describe below the generation of synthetic RS camera motion and the creation of training and testing datasets for three classes of images. For each of the classes, our CNNs are trained separately.

Camera motion We use random third-degree polynomials as synthetic camera trajectories to generate training and testing data. Each trajectory is a set of two 256-length vectors (for row-wise motion), one each for translation and rotation. We limit the translation to the range $[-40, 40]$ pixels, and rotation to the range $[-\frac{\pi}{8}, \frac{\pi}{8}]$ radians.

Chessboard class We take different horizontally and/or vertically translated versions of a 16-square chessboard image as our basic data. We then apply synthetic RS motions over these images to populate our full training set. To aid in quantitative analysis, only for the chessboard class, we train for three motion models: translation-only (T-only), rotation-only (R-only), and combined translation and rotation (T+R). For each of these three models, we generate a training set of size 7014 (14 basic images $\times$ 500 random motions + 14 basic images without motion). The CNN output vector length is N=15 for the translation and rotation-only models, and it is N=30 for the combined model (15 each for translations and rotations). We train both VanillaCNN and RowColCNN for each of these three models.

Urban scene class We build the clean urban scene data by combining the building images in Sun [21], Oxford [11] and Zurich [18, 17] datasets. Each clean image is distorted with 150 random camera trajectories giving us approximately 300,000 labeled images. We also randomly flip the original images left-right before applying motion distortion. For testing, we pick new images from the combined dataset of [21], [11] and [18, 17] not used for training, and synthetically apply random motion to generate 200 test images. We also create a separate test set from Caltech building dataset [1] (from which none of the images are used in training at all) generating RS effect in a similar manner.

Face class We use face images from the Labeled Faces in the Wild (LFW) face dataset [5] for both training and testing. The training set consists of faces of 5000 persons at different poses with 50 motions applied on each face, thereby making the size of training data as 250,000. We choose 200 faces (different from that of training) for testing having different camera motions applied on each of them.

4.3. Quantitative Analysis

We first describe the metrics that we use for quantitative analysis and then show our results.

Metrics We use the following three metrics: [P1] PSNR (dB) between the ground-truth and corrected images, [E2] root mean squared error (RMSE) between the ground-truth and predicted motion in pixels for translation and degrees for rotation, and [E3] curvature residual in pixels. A high P1, and low E2 and E3 indicate better performance.

To measure the curvature residual, which is specific to the chessboard class, we first extract horizontal and vertical curves from the corrected output, and then calculate the distance between the curves and the ground-truth lines (at all row and column locations). The RMSE of this distance value for all curves gives E3.

Synthetic motion We now show the performance of RS correction for the testing data built up using synthetic camera motion.

Chessboard class: Table 1 shows the performance of different methods on the chessboard dataset based on P1, E2, and E3. It is very clear that both our CNNs perform on par with the baseline video correction methods (which use a reference frame for motion estimation). The performance of our single image method matches to that of these baseline methods. Our use of long kernels in RowColCNN results in better performance as against the traditional square kernel-based VanillaCNN. In all the cases, the PSNR (P1)[1] of RowColCNN is higher than that of VanillaCNN in all cases, and E2 and E3 are lower in most cases.

The urban-specific method [13] puts a hard constraint of forcing curves into horizontal and vertical lines, and it provides an advantage for the chessboard class. For the T-only motion, [13] performs better than our method; but in the presence of rotations, it is not able to correct curvatures properly. It fares poorly even compared to VanillaCNN in R-only and T+R cases.

Urban scene class: The correction performance of our networks in comparison to other methods on both the combined building and testing-exclusive Caltech urban datasets are shown in Table 2. We observe that RowColCNN performs better than VanillaCNN in both the datasets and on par with the baseline methods. It performs better than the urban-specific RS correction method of [13] in which proper curve detection to aid in motion estimation is a crucial step, and false curve detection might lead to wrong solutions. And hence, it performs worse on urban scenes compared to the chessboard class.

Face class: Table 2 also summarizes the correction performance for the face class. We observe that the CNNs

[1]The range of PSNR (36–40dB) is generally higher than the typical values observed in correction tasks. This is due to the nature of chessboard having only two values – black and white, which results in most of the entries attaining 0 value in the MSE image, thus leading to higher PSNR.

Table 1. Quantitative comparisons on the synthetic chessboard dataset.

Method	Type	Uses reference?	T-only				R-only				T+R				
			P1	E2t	E3h	E3v	P1	E2r	E3h	E3v	P1	E2t	E2r	E3h	E3v
Ringaby [14]	Baseline	Yes	40.84	0.65	0.68	0.89	36.84	1.31	0.74	1.85	36.91	1.86	1.32	0.69	2.05
Grundmann [3]	Baseline	Yes	38.76	0.92	0.76	1.12	36.99	1.18	1.12	2.19	37.03	1.63	1.41	0.72	1.93
RowColCNN	Testing	No	38.01	1.78	0.81	**1.97**	**37.83**	**0.42**	**0.88**	1.16	**37.41**	1.52	**0.43**	**0.88**	**1.60**
VanillaCNN	Testing	No	37.52	1.85	1.32	2.04	37.77	0.53	0.99	**1.12**	37.30	**1.48**	0.92	1.33	1.86
Rengarajan [13]	Testing	No	**39.75**	**1.75**	**0.72**	2.02	35.58	2.90	3.81	2.05	33.64	12.40	2.68	3.32	3.18

T: Translation, R: Rotation, P1: PSNR (dB), E2t: Motion RMSE (pixels), E2r: Motion RMSE (degrees), E3h: Horizontal Curve Residual, E3v: Vertical Curve Residual

Table 2. Quantitative comparisons on urban scene and face datasets.

Method	Type	Uses reference?	Combined building dataset			Caltech building dataset			LFW face dataset		
			P1	E2t	E2r	P1	E2t	E2r	P1	E2t	E2r
Ringaby [14]	Baseline	Yes	32.86	3.03	1.07	32.67	4.32	1.39	34.75	1.87	1.43
Grundmann [3]	Baseline	Yes	32.57	3.34	1.17	32.07	4.76	1.82	34.57	1.92	1.53
RowColCNN	Testing	No	**32.25**	**3.76**	**1.15**	**32.50**	**5.07**	**1.41**	**34.14**	**2.17**	**1.67**
VanillaCNN	Testing	No	32.19	3.84	1.29	32.36	6.22	1.68	34.01	2.96	1.82
Rengarajan [13]	Testing	No	29.82	11.89	3.58	29.15	15.76	3.26	–	–	–
Heflin [4]	Testing	No	–	–	–	–	–	–	29.32	18.03	–

P1: PSNR (dB), E2t: Motion RMSE (pixels), E2r: Motion RMSE (degrees)

achieve good performance here too. Even though the baseline methods perform better than ours, they outperform us only by a small margin. The performance of RowCNN is better than the face-specific correction method of [4] due to its limitations of working only on almost-frontal face poses. Also, [4] estimates only a skew parameter, and hence we have calculated only the PSNR (P1) and translation error (E2t) values in the last row of Table 2.

Human camerashake dataset To test the capabilities of our trained network on real camera motion, we employ the dataset of [7], which contains 40 trajectories of real camera shake by humans who were asked to take photographs with relatively long exposure times. Since long exposure leads to motion blur which is not within the scope of this work, we use a short segment of the recorded trajectory to introduce the RS effect (with 38.4ms top to bottom row delay). We generated 200 such RS images for both chessboard and urban scene classes by randomly clipping motion segments from the 40 trajectories. We corrected them using the predicted motion from RowColCNN.

Table 3 shows the RMSE for both translation and rotation, and the PSNR between the clean and corrected images. The error is low and PSNR is high for both the classes signifying good RS correction by RowColCNN due to human handshake. This confirms the validity of our third degree polynomial assumption for camera motion during the exposure of interest.

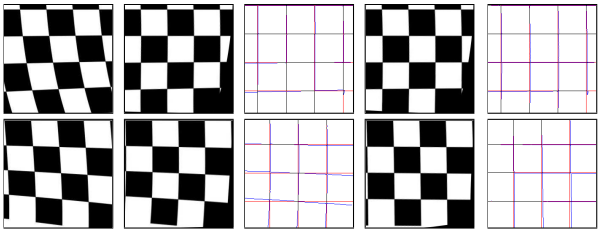
4.4. Visual Comparisons

We first compare the outputs of the two proposed CNNs for a distorted chessboard image, and then show other visual comparisons with existing methods.

CNN models Fig. 7(a) shows an RS image with heavy rotations and translations. Both vertical and horizontal lines are curved. The corrections by VanillaCNN and RowColCNN are shown in Figs. 7(b) and (d), respectively. VanillaCNN corrects distortions to a good extent; however, it has more residual errors compared to RowColCNN. The deviation of the edges in the corrected image from the original grid is shown in Figs. 7(c) and (e).

Table 3. RowColCNN performance on camerashake dataset [7].

Class	PSNR (dB) P1	Translation RMSE E2t (pixels)	Rotation RMSE E2r (degrees)
Chessboard	37.23	2.8074	0.35
Urban scene	32.19	3.9677	0.76



(a) RS (b,c) VanillaCNN and residual (d,e) RowColCNN and residual

Figure 7. Distortion correction of a chessboard image.

In RowColCNN, the initial square-convolutional layers provide low-level information to the subsequent directional banks which extract the required bidirectional information. Fig. 8 shows the square filters of the first layer trained for urban scenes which are mainly directed gradients at different angles (not only horizontal and vertical). Row and column kernel banks combine these different directional feature maps in their respective directions to extract motion.

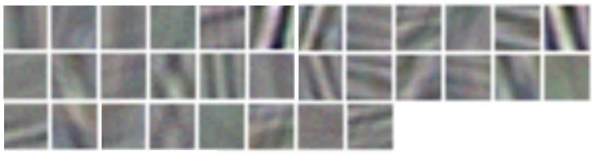
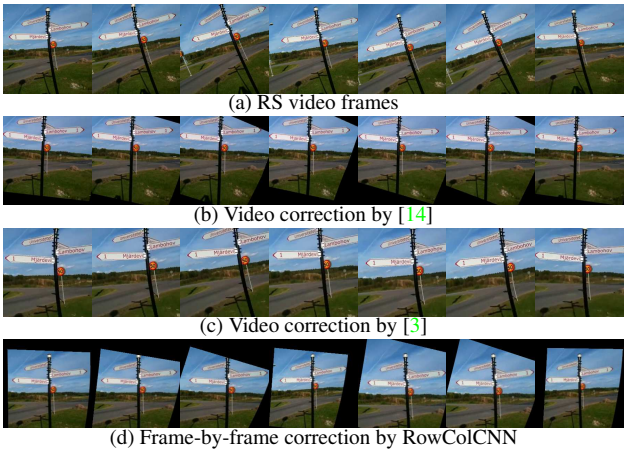


Figure 8. Trained filters for the first layer of RowColCNN.



(a) RS video frames

(b) Video correction by [14]

(c) Video correction by [3]

(d) Frame-by-frame correction by RowColCNN

Figure 9. Visual comparisons of our RowColCNN with existing video correction algorithms.

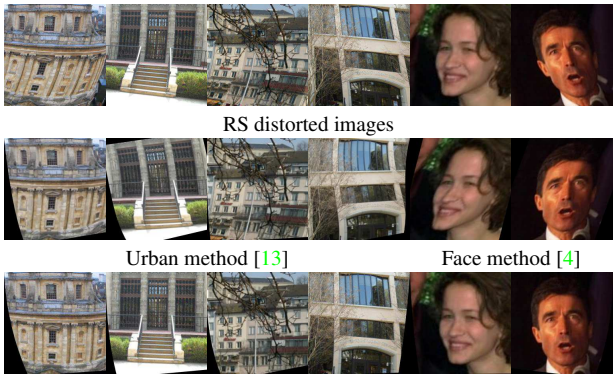


Figure 10. Visual comparisons with existing scene-specific single-image methods of [13] (urban scenes) and [4] (faces).

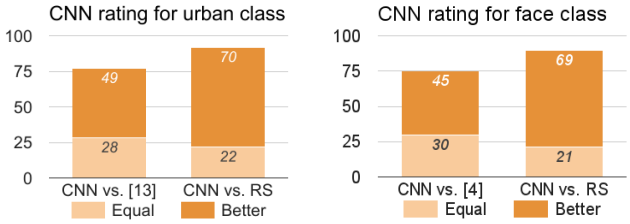


Figure 11. Human rating for CNN outputs against [13] and [4].

Video methods We show a comparison of our *frame-by-frame* RS correction against the video rectification methods of [14] and [3]. The video methods employ a batch of neighboring frames to estimate the camera trajectory, while our method uses only one frame. We also do a global inplane motion registration between our frame-by-frame corrected outputs with respect to the corrected first frame (which in no way could correct skew and curves by itself). Fig. 9 shows some frames from the RS video of a sign pole taken from [14], and the corresponding output frames of our method and the two video correction methods. Our method restores the straightness of the pole in almost all frames similar to the correction by [14]. The correction by [3] is not as perfect for this heavy motion. The video is provided as a supplementary material.

Single-image methods In Fig. 10, we show visual comparisons against scene-specific correction methods. The motion estimation scheme for urban scenes in [13] heavily depends on the selection of suitable curvatures in the image, and hence, it fails to correct distortions when there are false detections or natural curvatures. In the second column, [13] tries to make the slanted step handles vertical, and it fails in the third and fourth examples due to the presence of tree branches. Similarly, [4] estimates only skew type distortion from eye and nostril feature points extracted from almost-frontal faces, and hence it fails on faces that are not frontal (fifth) and badly illuminated (sixth). In all these varied examples, RowColCNN corrects distortions properly.

Human perception rating We surveyed 50 users to provide preferences for 30 image sets, consisting of clean GS, RS, RowColCNN corrected, and [13] or [4] corrected images, based on their visual perception. The sets include a variety of RS images ranging from no to heavy motion. Fig. 11 shows the performance of RowColCNN against competing methods. The participants rate our outputs as equal or better than those of comparison methods at least 75% of the time for both urban and face corrections (left bar in both plots). Our method is equally or better preferred at least 90% of the time compared to the RS image. More information is provided in the supplementary material.

Captured real data In Fig. 2, we show the correction results of our captured images using RowColCNN. These are taken using a Motorola MotoG2 mobile phone camera either with a handshake or from a moving vehicle. The captured images are resized and cropped to 256×256 (without any image rotation that would affect the row-wise exposure property), and then are corrected by our method. Skew and curvature disortions in these varied scenes are properly removed by our method.

More examples are provided in the supplementary material.

5. Conclusions

We proposed a new CNN architecture based on long rectangular kernels to aid in correcting rolling shutter distortions from single-images. We modeled the camera motion as translation+rotation polynomials sans any camera calibration, and it was shown to work for real images captured with mobile phones. Our single image method performs on par with existing video correction algorithms which use multiple images. The learning power of CNNs removes the difficulty of manual feature choice and extraction as employed by existing nonlearning-based single-image works.

References

[1] M. Aly, P. Welinder, M. Munich, and P. Perona. Towards Automated Large Scale Discovery of Image Families. In *IEEE Second Workshop on Internet Vision, CVPR*, June 2009. 6

[2] A. Chakrabarti. A neural approach to blind motion deblurring. In *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, pages 221–235. Springer International Publishing, 2016. 2

[3] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Calibration-free rolling shutter removal. In *International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2012. 1, 2, 5, 7, 8

[4] B. Heflin, W. Scheirer, and T. E. Boult. Correcting rolling-shutter distortion of CMOS sensors using facial feature detection. In *International Conference on Biometrics: Theory Applications and Systems*, pages 1–6. IEEE, 2010. 2, 4, 6, 7, 8

[5] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. 6

[6] Y.-G. Kim, V. R. Jayanthi, and I.-S. Kweon. System-on-chip solution of video stabilization for cmos image sensors in hand-held devices. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(10):1401–1414, 2011. 2

[7] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII*, pages 27–40, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 3, 7

[8] C.-K. Liang, L.-W. Chang, and H. H. Chen. Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323–1330, 2008. 1, 2

[9] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations (ICLR)*, 2016. 2, 4

[10] A. Patron-Perez, S. Lovegrove, and G. Sibley. A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision (IJCV)*, pages 1–12, 2015. 1, 2

[11] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2007. 6

[12] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016. 2, 4

[13] V. Rengarajan, A. N. Rajagopalan, and R. Aravind. From bows to arrows: Rolling shutter rectification for urban scenes. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 4, 5, 6, 7, 8

[14] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal Computer Vision (IJCV)*, 96(3):335–352, 2012. 1, 2, 5, 7, 8

[15] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth. Discriminative non-blind deblurring. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 604–611, 2013. 2

[16] C. Schuler, H. Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1067–1074, 2013. 2

[17] H. Shao, T. Svoboda, T. Tuytelaars, and L. V. Gool. Hpat indexing for fast object/scene recognition based on local appearance. In *Proceedings of the second International Conference on Image and Video Retrieval*, CIVR'03, pages 71–80, Berlin, Heidelberg, 2003. Springer-Verlag. 6

[18] H. Shao, T. Svoboda, and L. Van Gool. Zubud – Zurich buildings database for image based recognition, 2003. 6

[19] S. Su and W. Heidrich. Rolling shutter motion deblurring. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1537, 2015. 2

[20] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 769–777, 2015. 2

[21] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 3485–3492. IEEE, 2010. 6

[22] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Conference on Neural Information Processing Systems (NIPS)*, pages 1790–1798, 2014. 2