2019

Deep Convolutional Neural Networks for Estimating Lens Distortion Parameters

Sebastian Lutz
Trinity College Dublin, Ireland

Mark Davey
Foundry

Aljosa Smolic
Trinity College Dublin, Ireland

Recommended Citation

Lutz, S., Davey, M., & Smolic, A. (2019). Deep convolutional neural networks for estimating lens distortion parameters. *IMVIP 2019: Irish Machine Vision & Image Processing*, Technological University Dublin, Dublin, Ireland, August 28-30. doi: 10.21427/yg8t-6g48

Deep Convolutional Neural Networks for estimating lens distortion parameters

Sebastian Lutz†*, Mark Davey*, Aljosa Smolic†

†*Trinity College Dublin* **Foundry*

Abstract

In this paper we present a convolutional neural network (CNN) to predict multiple lens distortion parameters from a single input image. Unlike other methods, our network is suitable to create high resolution output as it directly estimates the parameters from the image which then can be used to rectify even very high resolution input images. As our method it is fully automatic, it is suitable for both casual creatives and professional artists. Our results show that our network accurately predicts the lens distortion parameters of high resolution images and corrects the distortions satisfactory.

Keywords: Camera calibration, lens distortion, CNN, deep learning

1 Introduction

In professional photography and cinematography, the choice of camera lens is very important, mostly for artistic reasons. Wide angle lenses, for example, are often used for landscape shots to capture as much of the scene as possible, while lenses with much higher focal length are typically used for portraits. All lenses introduce different amounts of lens distortion to the image and it can be important to change the amount of lens distortion in an image after it has been taken, e.g. in Computer-generated imagery (CGI) work. In professional production, lens distortion parameters corresponding to a lens distortion model are often estimated from imaging a known grid, e.g. a checkerboard, before the actual capture. Post-production software then allows straightening out the curved lines in the images. However, this necessary step is often forgotten or impractical and it becomes much more difficult to rectify a distorted image without this information. Still professional software provides tools to do that, but it remains a cumbersome interactive procedure. An automatic solution would help reducing costs. Additionally, many computer vision algorithms, e.g. pose estimation and 3D reconstruction, depend on the assumption of the ideal pinhole camera model [Hartley and Zisserman, 2003]. If applied in automatic systems these often require automatic lens distortion correction.
Our motivation in this paper is therefore to devise a convolutional neural network to automatically estimate the distortion parameters of an image. These can then be used to easily rectify any image or shot with the same camera and lens combination.
Our contribution is the first convolutional neural network to estimate multiple lens distortion parameters corresponding to Fitzgibbon's division model [Fitzgibbon, 2001], which corresponds to the default model that is used in NUKE [Foundry, 2007], the premier compositing tool for professional post-production. We deliberately chose to design the network to estimate the parameters directly instead of outputting a rectified image of the input, to seamlessly integrate this method into NUKE and allowing it to work even for the ultra-high-resolution images that are used in professional movie production. Our results show the suitability of our CNN method on a range of images.

2 Related work

We formulate the estimation of lens distortion parameters as a regression problem. Regression is very common in computer vision and there are many similar problems in the field [Fanelli et al., 2011], [Burgos-Artizzu et al., 2013], [Sun et al., 2012].

There has also been done a lot of research on lens distortion itself. Important for this paper is especially the work of Fitzgibbon et. al. [Fitzgibbon, 2001], who defines the lens distortion model that we are using in this paper and which will be explained in more detail in section 3. To rectify distorted images, there are several works that use lines for the estimation [Melo et al., 2013], [Mei et al., 2015], [Bukhari and Dailey, 2013]. They essentially aim to estimate the distortion that caused straight lines to appear curved in the image. Similar in aim to our work, Bukhari et. al. [Bukhari and Dailey, 2013] use an extended Hough transform of image lines to estimate one radial lens distortion parameter. Rong et. al. [Rong et al., 2016] aim to do the same, but use a CNN to estimate one lens distortion parameter from an input image. However, they only estimate the first lens distortion coefficient, whereas our model includes more coefficients and also estimates the center of the distortion. There has also been some research done on estimating the parameters for fish-eye lenses. Yin et. al. [Yin et al., 2018] developed a network to rectify fish-eye images. Instead of estimating the distortion parameters, they output the final rectified image directly. We find this approach not suitable for high-quality images, since the images would have to be scaled down to fit into the network and the final network output would have to be upscaled afterwards, leading to a loss in quality.

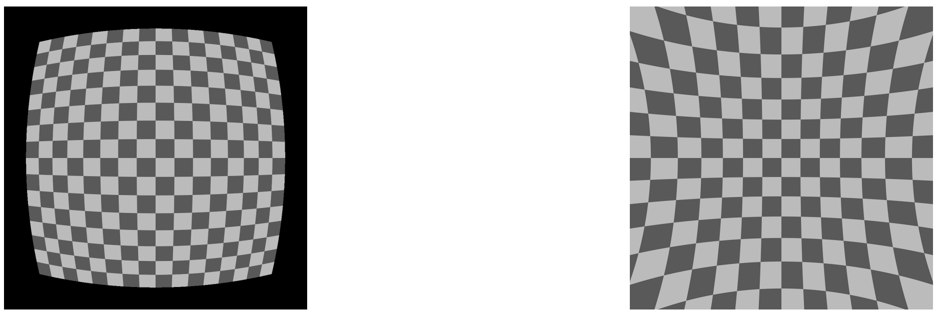

Figure 1: Two types of common lens distortion types. To the left: Barrel distortion. This type folds the image inwards and introduces black regions on the outside where image information is missing. To the right: Pincushion distortion. This type expands the image outwards and the outer edges of the image disappear.

3 Lens distortion model

The lens distortion model that is used in this work and also as the standard model in NUKE [Foundry, 2007] is the division model introduced by Fitzgibbon [Fitzgibbon, 2001]. The division model is written as:

$$\begin{aligned} x_{new} &= c_x + \frac{\hat{x}}{1 + k_1 * r^2 + k_2 * r^4 + \ldots} \\ y_{new} &= c_y + \frac{\hat{y}}{1 + k_1 * r^2 + k_2 * r^4 + \ldots} \end{aligned} \tag{1}$$

where c_x and c_y are the center coordinates for the distortion, $k_1, k_2, \ldots$ are the distortion denominators, $\hat{x} = x_{old} - c_x$ and $\hat{y} = y_{old} - c_y$ are the pixel coordinates corrected for the center of the distortion, and $r^2 = \hat{x}^2 + \hat{y}^2$ is the radius of the coordinates from the center of the distortion. All coordinates are normalized such that the top-right pixel in a square image has the coordinate $(1,1)$ and the bottom-left pixel the coordinate $(-1,-1)$. The above equation is used to apply a distortion on an undistorted image.

Depending on the distortion coefficients, this can lead to a number of distortion effects. Generally, one distinguishes between barrel distortion and pincushion distortion, as can be seen in figure 1. The reverse transformation, i.e. correcting a lens distortion effect on an image to straighten it out is unfortunately less straightforward and Newton's method has to be used to iteratively undistort the image.

To date correction of lens distortions is an interactive process in professional production. Automatic computer vision systems typically apply pre-defined lens distortion parameters which may be inaccurate. Interactive methods for instance require an image of a checkerboard grid that fills at least the biggest part of the image. The algorithm automatically finds the corners in the checkerboard and finds the lines connecting them. In the distorted image, these lines will appear curved and the algorithm will estimate the correct lens distortion parameters by straightening out those curved lines. Similarly, interactive post-production software such as NUKE, provides tools to indicate curved lines in images, which can then be used to estimate distortion parameters causing them.

4 Method

Our network takes a RGB image of arbitrary aspect ratio and a maximum size of 512×512 as input and outputs the estimated coordinates of the distortion center, as well as the distortion denominators of the division model. It was a conscious design decision not to let the network undistort the image such as in [Yin et al., 2018], but to estimate the parameters directly so that they could be used in NUKE. This allows for seamless integration of the network into a NUKE node graph and allows for high-quality undistortions of even very large images that would not have fit into the network.

4.1 Network

The network architecture of our model is similar to the architecture of other models aimed to solve computer vision regression tasks [Lathuilière et al., 2018]. The network uses Xception [Chollet, 2017] as a convolutional backbone to extract dense feature maps from the input RGB image. To add more spatial information, these feature maps are then multiplied by an equal sized feature map that contains the squared radial distance of each element from the center of the feature map. This is equivalent to a weighting method that prefers elements on the borders of the feature maps to those in the center. Therefore, elements on the border, where the lens distortion effects are strongest, have more impact on the following layers.
To reduce the number of elements in the dense feature maps, two convolutions, followed by batch normalization respectively, with kernel size of 1 and stride of 2 are used to reduce the spatial size of the feature maps by a factor of 4 and the number of channels from 2048 to 512.
These final feature maps are then flattened and followed by two branches of fully-connected layers. While one branch estimates the the center coordinates of the distortion, the other one estimates the distortion denominators. Structurally both branches are the same and consist of one fully-connected layer with 256 neurons followed by one last fully-connected regression layer to estimate the final parameters. All convolution layers and all fully-connected layers except the last one are followed by ReLU activation functions. A diagram of our network architecture can be seen in figure 2.

4.2 Data

To train our network, we create a dataset of lens distorted images with known distortions by applying random parameters to a large set of images, which are assumed initially undistorted. As input we use the very large MSCOCO [Lin et al., 2014] dataset. This is not quite optimal, since we actually do not know any of the parameters of cameras and lenses that was used to take any of the images in the dataset. Nevertheless, we consider the images properly undistorted and use on-the-fly data augmentation to distort them according to the division model using a random selection of parameters. This is done through our tensorflow [Abadi et al., 2015] implementation of the division model directly on the GPU.

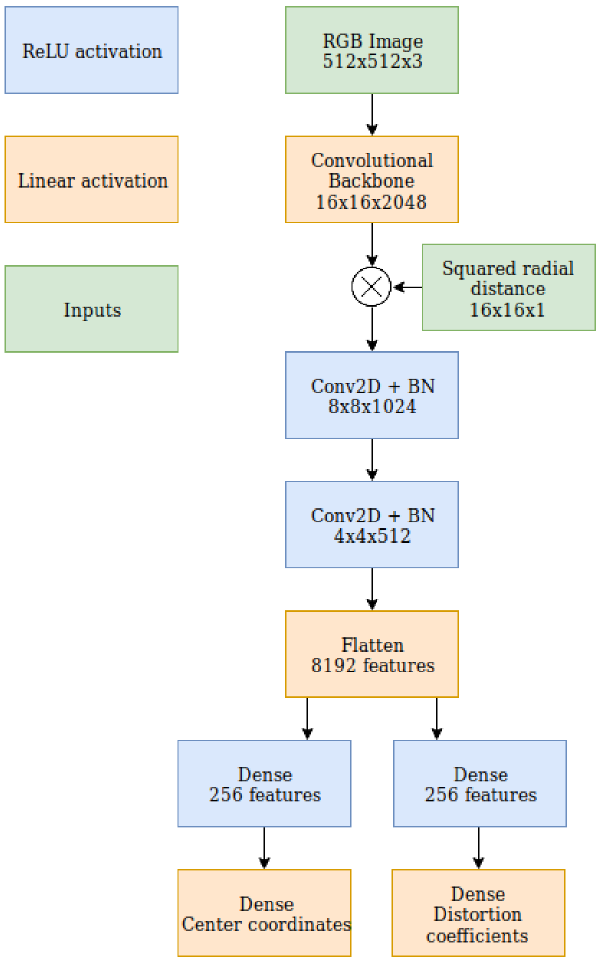

Figure 2: Network architecture.

Before the random distortion, all images are resized to 1024 on their longest size and the images are slightly zoomed in after the random distortion to remove any black borders that could have appeared due to a barrel distortion. This prevents the network to overfit on any curved image borders since those would not appear in real images. Finally, we add a small amount of random noise to the image and resize it so that it longest size is 512 pixels while padding the smaller side with black on both sides. This is done to prevent further overfitting on image artifacts that could appear due to the random distortion. The final batch of images sent to the network are square of size 512×512 and are normalized to have values within $[-1, 1]$.

4.3 Training details

We train our network using the dataset created as described above. For that, we initialize the network weights of the Xception convolutional backbone with pretrained Imagenet [Deng et al., 2009] weights. We also set the layers of the Xception entry flow to not trainable to reduce the amount of memory on the GPU the training takes. This allows for a bigger batch size of 16 and allows for faster training.
During the development of the network, we tried a variety of loss functions, but in the end, we found the standard mean squared error loss to work the best. We further used the Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.01.

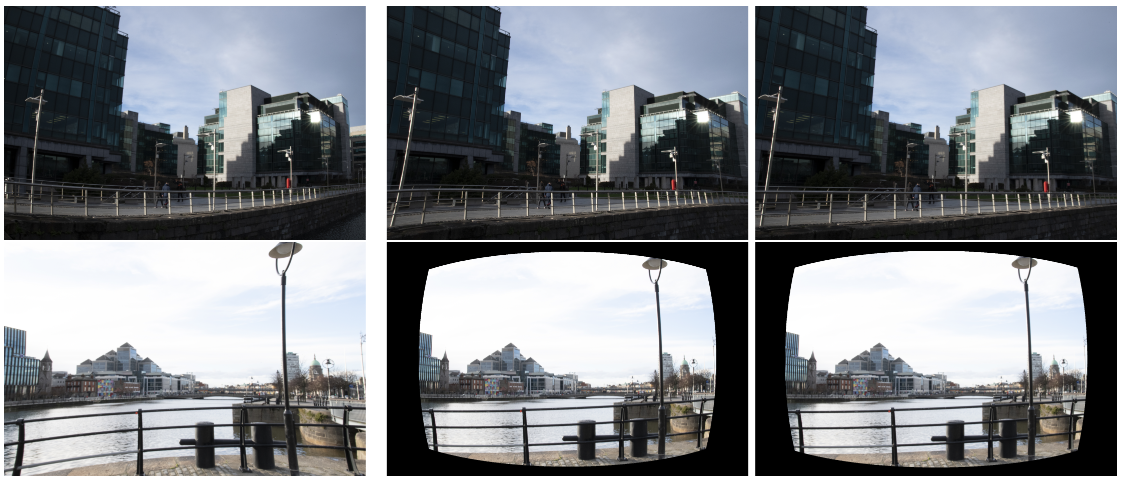

Figure 3: Our results on randomly distorted images.

5 Experiments

To evaluate our method, we created a small dataset of scenes. We took special care to capture some images that contain a lot of edges, but also some images that contain almost no straight edges. Afterwards we calculated the proper lens distortion parameters using images taken of a grid and made sure all images were rectified correctly. To create the final set of images for the experiment we then distorted all properly rectified images with random parameters to create a variety of lens distortion effects over the dataset. Since applying a barrel distortion to an image folds the image slightly inwards and introduces black areas around the edges where no image information is available (See figure 1), the images are slightly cropped around the center to remove these black areas.
The final images in the dataset all have a resolution of 1648×1065. This resolution is too big to feed directly into the network, so all images are resized to 512 pixels on the longest side and padded with zeros on the top and bottom to fill a 512 pixels square. This mirrors our data preparation during training. Afterwards all resized images are fed through the network to calculate the lens distortion parameters. Finally, these parameters are used on the original distorted images to rectify them.

6 Results

The results of our experiment can be seen in figure 3. As can be seen, our network works for both barrel and pincushion distortion and correctly rectifies the images. In direct comparison with the ground-truth, we can see that the network predictions are slightly better for pincushion distortions. In the first example image, some slight errors in the bottom left can be seen, where the lantern is not quite straight. The predicted parameters slightly overcompensate for the distortion.
Since our network predicts the lens distortion parameters directly, they can be applied directly to rectify very large images, even though the image needs to be downscaled to fit into the network. Therefore our undistorted results don't lose in quality as would be case if the network would have undistorted the images directly, since this output would have to be upsampled again.

7 Conclusion

We present a deep convolutional neural network to predict multiple lens distortion parameters for Fitzgibbon's division model. Our method works fully automatic and can be used in a variety of ways to help in a wide range of systems. It is suitable for casual creatives, e.g. on mobile platforms, who do not have the means or expertise to undistort their images using a checkerboard grid, but also for professional artists. Since we predict the lens distortion parameters directly, they can be used to undistort even ultra-high resolution images without a loss in quality. Additionally, our method is suitable as part of an automatic pipeline with algorithms that assume the idealized pinhole camera model, e.g. 3D reconstruction. Furthermore, our method is very fast and we can process multiple frames of the same camera/lens combination very quickly. As part of our future work, we would like to look into performance improvements by regressing the results over multiple frames or building a model that inherently takes multiple frames as input.

Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number 15/RP/2776.

References

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Bukhari and Dailey, 2013] Bukhari, F. and Dailey, M. N. (2013). Automatic radial distortion estimation from a single image. *Journal of Mathematical Imaging and Vision*, 45(1):31–45.

[Burgos-Artizzu et al., 2013] Burgos-Artizzu, X. P., Perona, P., and Dollár, P. (2013). Robust face landmark estimation under occlusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1513–1520.

[Chollet, 2017] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, pages 1610–02357.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee.

[Fanelli et al., 2011] Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 617–624. IEEE.

[Fitzgibbon, 2001] Fitzgibbon, A. W. (2001). Simultaneous linear estimation of multiple view geometry and lens distortion. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE.

[Foundry, 2007] Foundry (2007). Nuke. foundry.com.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Lathuilière et al., 2018] Lathuilière, S., Mesejo, P., Alameda-Pineda, X., and Horaud, R. (2018). A comprehensive analysis of deep regression. *arXiv preprint arXiv:1803.08450*.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[Mei et al., 2015] Mei, X., Yang, S., Rong, J., Ying, X., Huang, S., and Zha, H. (2015). Radial lens distortion correction using cascaded one-parameter division model. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 3615–3619. IEEE.

[Melo et al., 2013] Melo, R., Antunes, M., Barreto, J. P., Falcao, G., and Goncalves, N. (2013). Unsupervised intrinsic calibration from a single frame using a. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 537–544.

[Rong et al., 2016] Rong, J., Huang, S., Shang, Z., and Ying, X. (2016). Radial lens distortion correction using convolutional neural networks trained with synthesized images. In *Asian Conference on Computer Vision*, pages 35–49. Springer.

[Sun et al., 2012] Sun, M., Kohli, P., and Shotton, J. (2012). Conditional regression forests for human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3394–3401. IEEE.

[Yin et al., 2018] Yin, X., Wang, X., Yu, J., Zhang, M., Fua, P., and Tao, D. (2018). Fisheyerecnet: A multi-context collaborative deep network for fisheye image rectification. *arXiv preprint arXiv:1804.04784*.