A SIMPLE YET EFFECTIVE PIPELINE FOR RADIAL DISTORTION CORRECTION

He Zhao, Yongjie Shi, Xin Tong, Xianghua Ying, Hongbin Zha*

Key Lab of Machine Perception (MOE)
Peking University

ABSTRACT

Eliminating the radial lens distortion of an image is a crucial preprocessing step for many computer vision applications. This paper explores a simple yet effective pipeline for radial distortion correction. Different from existing state-of-the-art methods that design complex network structure and concatenate multi-branch features. Our model uses a single network without any additional supervision. We design two differentiable layers to synthesize and rectify distorted images efficiently. Based on these layers, an online data synthesis strategy, a sampling grid loss, and an image reprojection loss are proposed to improve the distortion correction accuracy. Compared with the state-of-the-art methods, our model achieves the best rectification quality on both the synthetic and real distorted images with dozens of times faster inference speed. The training data and codes will be released.[1]

***Index Terms*—** Radial distortion correction, Camera calibration, Spatial transform, Deep learning

1. INTRODUCTION

Wide-angle lenses have been widely used in varieties of computer vision tasks, including video surveillance [1], intelligent vehicles [2, 3] and depth estimation [4, 5], due to their wide field of view. However, images taken by wide-angle lenses usually suffer from severe geometric distortions, and the radial distortion is the most significant one. Therefore, it is vital to perform rectification as a fundamental pre-processing step for subsequent tasks. To automatically rectify distortion from a single image, Wang [6] and Alemen-Flores [7] calibrate images with the principle that a straight line should be projected into a straight line. However, their methods have reported promising correction performance only when direct lines can be accurately detected.

To achieve robust and efficient performance, Rong [8] first employ convolutional neural networks (CNNs) on radial distortion correction. Recently, the FishEyeRecNet [9] introduce scene parsing semantics into the rectification process and achieves better accuracy. Shi [10] propose an inverted foveal model to improve correction accuracy. Xue [11] propose a line-guided parameters estimation module and a distorted line segments perception module to utilize the line information better. Li [12] rectify distortion image with a learnable pixel flow and achieve state-of-the-art performance. Unfortunately, they usually apply networks with complex structures, and their training processes often rely on additional supervision. Besides, their data synthesis processes is time-consuming, which limit the sample diversity.

In this paper, we propose an end-to-end trainable pipeline for radial distortion correction. Given a single distorted image, our network first estimate the distortion parameter from its geometric structures. Then, the corrected images are generated using the obtained parameters. With the help of our Radial Distortion Synthesis and Correction layers (RDS & RDC), the data synthesis and correction speed of our system is dozens of times faster than the previous method. Our system achieves the state-of-the-art performance with ResNet-18 [13], a simple network without any other supervision except for the distortion parameter. Therefore, we call it a simple yet effective pipeline. We will release all training data and codes to help reproduce the results. The contributions of our work can be summarized as follows:

- We propose an end-to-end pipeline for radial distortion correction, which achieves state-of-the-art performance.
- Two differentiable layers (RDS & RDC) are designed to synthesize and rectify distorted images efficiently. An online data synthesis strategy, a sampling grid loss, and an image reprojection loss are proposed to improve the distortion correction accuracy.
- We will make the training data and codes publicly available to help reproduce our results.

2. PROPOSED METHOD

2.1. System Overview

We present an overview of our system in Figure 1. During the training process, the Radial Distortion Synthesis (RDS) layer first take original images as input to synthesize various distorted data with randomly sampled distortion coefficients. Then, the Estimation Network extracts features of unnatural structures which appear in the distorted images and estimates radial distortion coefficient. Finally, our Radial Distortion Correction (RDC) layer generate a correction grid of each image with the estimated coefficient and rectify the distorted image by bilinear sampling.

*Corresponding Author

[1] https://github.com/MccreeZhao/RDC_Pipeline

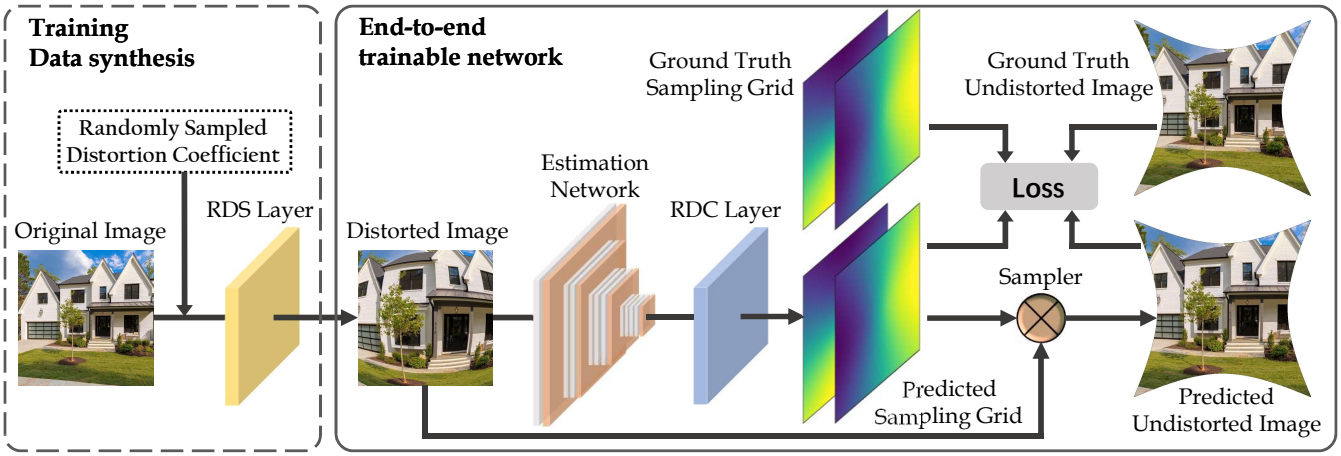

Fig. 1. The overview of our pipeline. The RDS and RDC layer are used to synthesize and rectify distorted images. The Estimation Network is used to predict the distortion coefficient. The whole pipeline is end-to-end trainable and supervised by the ground truth sampling grid and rectified image. We omit the gird generation and sampling process of RDS layer for simplicity.

2.2. Division Model for Radial Distortion

The general equation of a lens radial distortion model can be represented as:

$$\begin{pmatrix} x_u - x_c \\ y_u - y_c \end{pmatrix} = L(r) \begin{pmatrix} x_d - x_c \\ y_d - y_c \end{pmatrix}, \tag{1}$$

where (x_c, y_c) represents the distortion center which is usually same as the image center. (x_d, y_d) and (x_u, y_u) represent the point in the distorted image and undistorted image separately. We define $r = \sqrt{(x - x_c)^2 - (y - y_c)^2}$ as the radius of a point and $L(r)$ as the shape of the distortion model.

There are many radial distortion models such as the polynomial model [14], the division model [15], and the field of view model [16]. The division model proposed by Fitzgibbon [15] approximate the radial distortion curve with higher accuracy and fewer parameters. Following the previous works [10, 8, 12], we also apply this model which can be formulated as:

$$r_u = \frac{r_d}{1 + k r_d^2} \tag{2}$$

to synthesize and rectify distorted images. Here, k represents the distortion coefficient. The distortion degree increases as the absolute value of k rises. Lenses in the real world usually suffer from barrel distortion. Therefore we mainly focus on the negative term for the coefficient k.

2.3. Radial Distortion Synthesis and Correction Layers

To synthesize and rectify distorted images efficiently and allow the gradient flow back from the estimated grids and images to the parameters of Estimation Network, we design two differentiable RDS and RDC layers. Similar to previous work STN [17], each layer contains a grid generator and a bilinear sampler. However, our RDS and RDC layers perform more complex nonlinear radial distortion and correction transforms. Taking the RDC layer as an example, we describe the details of its forward and backward computations as following.

Taking a distorted image I^d as input, the Estimation Network E outputs the distortion coefficient as $k = E(I^d)$. To establish a reflection between the coordinates of distorted image I^d and corrected image I^c, we generate a sampling grid $G^c = \{G_i^c\}$ of pixel $G_i^c = (x_i^c, y_i^c)$. We assume that $\mathcal{T}$ is a transformation from I^d to I^c. Based on the Eq. 2, we have

$$\begin{pmatrix} x_i^d \\ y_i^d \end{pmatrix} = \mathcal{T}(G_i^c) = \frac{1 - \sqrt{1 - 4k(r_i^c)^2}}{2kr_i^c} \begin{pmatrix} x_i^c \\ y_i^c \end{pmatrix}. \tag{3}$$

Here we use a bilinear sampler which takes the information extracted by sampling grid $\mathcal{T}(G_i)$ from I^d to produce the output rectified image I^c. During back-propagation, the gradient should first flow back to the distortion coefficient k before back to the Estimation Network. That can be written as:

$$\frac{\partial I_i^c}{\partial k} = \frac{\partial I_i^c}{\partial x_i^d} \cdot \frac{\partial x_i^d}{\partial k} + \frac{\partial I_i^c}{\partial y_i^d} \cdot \frac{\partial y_i^d}{\partial k}. \tag{4}$$

Based on Eq. 3, the partial derivative can be calculated as:

$$\begin{aligned} \frac{\partial x_i^d}{\partial k} &= \frac{\partial x_i^d}{\partial r_i^d} \cdot \frac{\partial r_i^d}{\partial k} \\ &= \frac{\partial x_i^d}{\partial r_i^d} \cdot \frac{1 - 2k(r_i^c)^2 - \sqrt{1 - 4k(r_i^c)^2}}{2k(r_i^c)^2 \sqrt{1 - 4k(r_i^c)^2}} \end{aligned} \tag{5}$$

and the calculation of $\partial y_i^d / \partial k$ is in the same way.

We implement RDS and RDC layers with Pytorch [18] framework. Therefore, our data synthesis and correction can be processed efficiently with modern GPU cards.

2.4. Online Data Synthesis

Most of the previous deep learning-based methods construct a fixed training dataset. For example, Rong [8] and Shi [10] select 55,000 images from ImageNet [19], FishEyeRecNet [9] selects 19,011 images from ADE20K [20] dataset and they generate each image with a fixed distortion coefficient selected from a certain range. Their data synthesis strategies limit the variety of distortion and often lead to overfitting according to the training curves [10]. On the one hand, each image should be equally assigned with various coefficients. On the other hand, the dataset scale will increase linearly with the diversity of distortion, which demands a lot of memory storage. With the help of RDS layer, our data synthesis speed is accelerated several times. Therefore, we are able to dynamically synthesize distorted data online during every mini-batch of the training process with randomly sampled coefficients. It prevents our model from overfitting and improves distortion correction performance.

2.5. Sampling Grid Loss

The Euclidean distance between the estimated distortion coefficient and the ground truth can be represented as:

$$\mathcal{L}_k = \frac{1}{n}\sum_{i=1}^{n}\left\|k_i^{gt} - k_i^{pred}\right\|^2, \tag{6}$$

where n denotes the number of training samples. It is widely used to supervise the estimation network [10, 8, 11]. However, the distortion model only contains a few parameters. Optimizing them is not enough and prone to get stuck in the local minimum. On the contrary, the estimated sampling grid contains thousands of parameters which could provide a stronger constraint to boost the optimizing. Therefore, we design a sampling grid loss $\mathcal{L}_g$ to replace the coefficient loss, which is formulated as:

$$\mathcal{L}_g = \frac{1}{n}\sum_{i=1}^{n}\sum_{x}\sum_{y}\left\|G_i^{gt}(x,y) - G_i^{pred}(x,y)\right\|_1. \tag{7}$$

Here we apply Mean Absolute Error (MAE) since it is proved to perform better in convergence [21].

2.6. Image Reprojection Loss

High quality rectified image is the ultimate goal of our system. To this end, we design an image reprojection loss $\mathcal{L}_r$, which minimize the distance between the estimated rectified images and the corresponding ground truths. Here, we use Mean Squared Error (MSE) as the loss function:

$$\mathcal{L}_r = \frac{1}{n}\sum_{i=1}^{n}\sum_{x}\sum_{y}\left\|I_i^{gt}(x,y) - I_i^{pred}(x,y)\right\|^2. \tag{8}$$

While $\mathcal{L}_g$ use ℓ_1 loss to constraint the pixel location, the $\mathcal{L}_r$ focus on pixel value with ℓ_2 loss. The two terms are complementary with each other and jointly optimize the network to achieve better rectification accuracy.

3. EXPERIMENT

3.1. Implementation Details

Data Preparation. We use the same data as previous methods [8, 10] which are chosen from ImageNet [19]. The original dataset contains roughly 55,000 training images, 6,000 validation images and 6,000 test images. The difference lies in that, during the training process, we synthesize the distorted image online with randomly sampled distortion coefficient k. For the validation and test set, we fix the coefficient of each image either. The range of k is set to $[-4, 0] \times 10^{-6}$, where common radial lens distortion coefficient varies in. The resolution of all images is 256×256.

Training Details. We apply ResNet-18 as our Estimation Network in all experiments. Our system is trained with a combined loss of grid and reprojection loss: $\mathcal{L} = \lambda_g\mathcal{L}_g + \lambda_r\mathcal{L}_r$, where $\lambda_g = 1, \lambda_r = 0.5$. The optimization method we used is the Adaptive Moment Estimation (Adam) [22]. The learning rate starts from $1e-3$ and is divided by 10 at 20, 40, 60 epochs. We finish the training process at 80 epochs.

Evaluation Metrics. To evaluate the performance of our network on rectified images, we follow the evaluation metrics used in previous works [9, 11], including Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). In general, the larger the value of PSNR and SSIM are, the better the rectification quality will be. Since there is no ground truth rectified result for a real-world image, the performance of real data can only be evaluated visually.

3.2. Comparison with State-of-the-art Methods

Table 1. The PSNR and SSIM scores of different methods on our test dataset.

Methods	PSNR	SSIM
JMIV2013' Bukhari [23]	10.784	0.1847
IPOL2014' Alemán-Flores [7]	11.476	0.2743
CVPR2015' Zhang [24]	12.479	0.3129
ACCV2016' Rong [8]	13.883	0.3492
ICPR2018' Shi [10]	23.725	0.9746
CVPR2019' Li [12] (w/o fitting)	16.019	0.9353
Ours	**31.921**	**0.9954**

The dataset we constructed enables us to assess our method quantitatively. We run the proposed model and the SOTA ones on our test dataset and use PSNR and SSIM as evaluation metrics. The comparisons are reported in Table 1. From the evaluation results, we observe that no matter in image rectification or in structure maintenance, our method is obviously superior to other methods and has achieved the highest score on PSNR and SSIM. It is worth mentioning that the estimation network we use is a simple ResNet-18 and no additional supervision such as the line-segments [11], or semantic information [9] of images are needed. The superiority of our system can be attributed to our proposed online data synthesis strategy, sampling grid loss, and image reprojection loss.

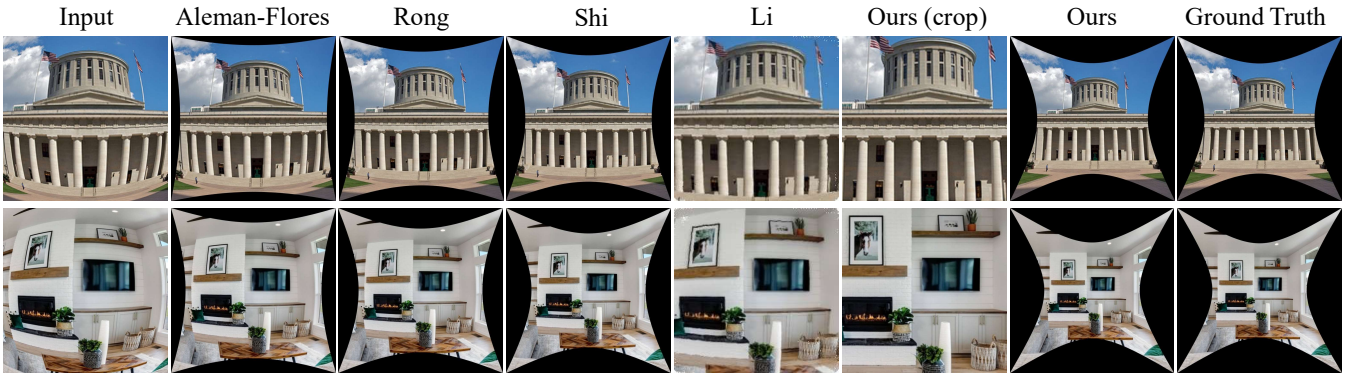

Fig. 2. Qualitative comparison between our system and Alemán-Flores [7], Rong [8], Shi [10], Li [12] on our test set.

To visually compare the rectification effects, we visualize the rectified results of our model and other state-of-the-art methods on both synthesized and real-world data. As shown in Figure 2 and Figure 3, our method achieves the best rectification performance, and the rectified images are closer to the ground truth, while other methods can not satisfy the needs of eliminating the distortion.

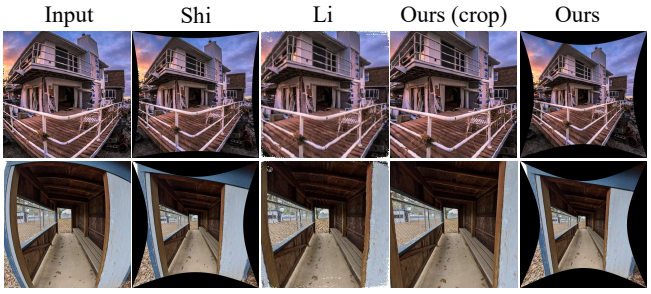

Fig. 3. Qualitative comparison between our system and Shi [10], Li [12] on real images.

We further compare the running speed of our methods with the state-of-the-arts. As presented in Table 2, the data synthesis process of Li [12] rely on loop operation, which is time-consuming and difficult to accelerate by hardware-based parallelization. On the contrary, our method can benefit from non-loop forward process implemented on GPU. The high-speed data synthesis and rectification allow us to synthesize data online and train the whole network in an end-to-end manner with grid and image loss functions.

Table 2. Runing speed (frames per second) of our system compared with Li [12]

Methods	Data Synthesis	Data Rectification
Li [12]	0.408 (Intel E5 2620)	33.07 (GTX 1080Ti)
Ours	**3774** (GTX 1080Ti)	**514.2** (GTX 1080Ti)

3.3. Ablation Study

In this section, we mainly analyse the validity of our online data synthesis strategy and loss functions. We evaluate the quality of rectified images on PSNR and SSIM with different settings and present the results in Table 3.

Table 3. Ablation study on loss function and data synthesis strategy. $\mathcal{L}_k$ represents replacing our combined loss with coefficient estimation error. Fix means the training data are synthesized with fixed distortion coefficient.

	Methods	PSNR	SSIM
Loss Function	w/o $\mathcal{L}_g$	29.326	0.9919
	w/o $\mathcal{L}_r$	28.980	0.9914
	$\mathcal{L}_k$ (w/o $\mathcal{L}_g$ & $\mathcal{L}_r$)	26.571	0.9862
Data Synthesis	Fix	24.052	0.9758
	Ours	**31.921**	**0.9954**

The results demonstrate that our grid loss and reprojection loss do play critical roles in our network and are complementary with each other. The $\mathcal{L}_k$ is proved to perform worth in convergence compared with $\mathcal{L}_g$ and $\mathcal{L}_r$. Besides, the rectification accuracy severely degrades with fixed training data. It indicates the importance of our RDS layer and the online data synthesis strategy which prevent our model from over-fitting.

3.4. Analysis and Future Work

As mentioned above, we apply a simple division model with only one parameter. Therefore, it is difficult for our system to deal with images under more complex distortion. The trade-off between model complexity and rectification capacity will be explored in our future work. Besides, although in most cases, the radiation center is the same as the image center, estimating the real distortion center may further improve the generalization ability of our system.

4. CONCLUSION AND FUTURE WORK

In this paper, we propose a pipeline for radial distortion correction. Our pipeline is simple, which only utilize classic ResNet-18 to estimate distortion coefficient without any additional supervision. It is also effective. With the help of the online data synthesis strategy and two carefully designed loss functions, our method achieves the best rectification performance compared with the state-of-the-art methods, both qualitatively and quantitatively. We will release all training data and codes to help reproduce our results.

Acknowledgement. This work was supported in part by State Key Development Program Grand No. 2016YFB1001001, and NNSFC Grant No. 61971008.

5. REFERENCES

[1] Philip DeCamp, George Shaw, Rony Kubat, and Deb Roy, "An immersive system for browsing and visualizing surveillance video," in *ACM MM*. ACM, 2010, pp. 371–380.

[2] Yeqiang Qian, Ming Yang, Chunxiang Wang, and Bing Wang, "Pedestrian feature generation in fish-eye images via adversary," in *ICRA*. IEEE, 2018, pp. 2007–2012.

[3] Massimo Bertozzi, Luca Castangia, Stefano Cattani, et al., "360 detection and tracking algorithm of both pedestrian and vehicle using fisheye images," in *IV*. IEEE, 2015, pp. 132–137.

[4] Shishir Shah and JK Aggarwal, "Depth estimation using stereo fish-eye lenses," in *ICIP*. IEEE, 1994, vol. 2, pp. 740–744.

[5] Varun Ravi Kumar, Stefan Milz, Christian Witt, et al., "Near-field depth estimation using monocular fisheye camera: A semi-supervised learning approach using sparse lidar data," in *CVPRW*, 2018.

[6] Aiqi Wang, Tianshuang Qiu, and Longtan Shao, "A simple method of radial distortion correction with centre of distortion estimation," *JMIV*, vol. 35, no. 3, pp. 165–172, 2009.

[7] Miguel Alemán-Flores, Luis Alvarez, Luis Gomez, and Daniel Santana-Cedrés, "Automatic lens distortion correction using one-parameter division models," *IPOL*, vol. 4, pp. 327–343, 2014.

[8] Jiangpeng Rong, Shiyao Huang, Zeyu Shang, and Xianghua Ying, "Radial lens distortion correction using convolutional neural networks trained with synthesized images," in *ACCV*. Springer, 2016, pp. 35–49.

[9] Xiaoqing Yin, Xinchao Wang, Jun Yu, Maojun Zhang, Pascal Fua, and Dacheng Tao, "Fisheyerecnet: A multi-context collaborative deep network for fisheye image rectification," in *ECCV*, 2018, pp. 469–484.

[10] Yongjie Shi, Danfeng Zhang, Jingsi Wen, Xin Tong, Xianghua Ying, and Hongbin Zha, "Radial lens distortion correction by adding a weight layer with inverted foveal models to convolutional neural networks," in *ICPR*. IEEE, 2018, pp. 1–6.

[11] Zhucun Xue, Nan Xue, Gui-Song Xia, and Weiming Shen, "Learning to calibrate straight lines for fisheye image rectification," in *CVPR*, 2019, pp. 1643–1651.

[12] Xiaoyu Li, Bo Zhang, Pedro V Sander, and Jing Liao, "Blind geometric distortion correction on images through deep learning," in *CVPR*, 2019, pp. 4855–4864.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[14] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[15] Andrew W Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion," in *CVPR*. IEEE, 2001, vol. 1, pp. I–I.

[16] Frédéric Devernay and Olivier Faugeras, "Straight lines have to be straight," *Machine Vision & Applications*, vol. 13, no. 1, pp. 14–24, 2001.

[17] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al., "Spatial transformer networks," in *NIPS*, 2015, pp. 2017–2025.

[18] Adam Paszke, Sam Gross, Francisco Massa, et al., "Pytorch: An imperative style, high-performance deep learning library," in *NIPS*, 2019, pp. 8024–8035.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. Ieee, 2009, pp. 248–255.

[20] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba, "Scene parsing through ade20k dataset," in *CVPR*, 2017, pp. 633–641.

[21] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz, "Loss functions for image restoration with neural networks," *ITCI*, vol. 3, no. 1, pp. 47–57, 2016.

[22] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] Faisal Bukhari and Matthew N Dailey, "Automatic radial distortion estimation from a single image," *JMIV*, vol. 45, no. 1, pp. 31–45, 2013.

[24] Mi Zhang, Jian Yao, Menghan Xia, Kai Li, Yi Zhang, and Yaping Liu, "Line-based multi-label energy optimization for fisheye image rectification and calibration," in *CVPR*, 2015, pp. 4137–4145.