Deep Dive into DeepCalib: Exploring Methods for Camera Calibration

Amrita Palaparthi
Stanford University
amritapv@stanford.edu

Yifei He
Stanford University
yifeihe@stanford.edu

Abstract

Many popular approaches to camera calibration leverage photographs of a calibration rig to map known coordinates in the world reference frame to coordinates in the image plane, enabling the accurate estimation of camera parameters and distortion. Recently, however, Deep Learning has been leveraged to perform effective camera calibration without the use of a calibration rig or known world coordinates. We explore the performance of DeepCalib [3], a CNN-based model designed to estimate camera parameters for images taken on wide-field-of-view cameras. In particular, we examine both its effectiveness in comparison to traditional checkerboard-based calibration as well as its generalizability across datasets real-world panoramas and synthetic images rendered in Blender. Qualitative and quantitative analysis of parameter prediction and undistortion indicate that DeepCalib performs comparably to checkerboard-based calibration methods on synthetic data, and that performance generalizes to datasets outside of those used for evaluation by the authors.

1. Introduction

A wide variety of methods have been proposed for the task of camera calibration. However, traditional checkerboard-based calibration techniques are often time consuming to do, as they require knowledge of ground truth 3D points and several images of a specific calibration rig (often a checkerboard pattern). Bogdan et al. introduce a method of camera calibration that circumvents these constraints, using a CNN model to predict focal length and a single distortion parameter from images taken with a wide field-of-view camera. This model was trained and evaluated on the Sun360 dataset [6].

In this paper, we explore the accuracy and generalizability of the DeepCalib model for camera parameter prediction and image undistortion. We begin by examining focal length and distortion accuracy when DeepCalib is applied to a new dataset - real-world images from Pano3D [1]. This dataset was selected because, like Sun360, it contains panoramic photographs that can be converted into distorted images using the unified spherical distortion model, but unlike Sun360, the images we use consist solely of views from building-scale indoor scenes.

We then compare the performance of DeepCalib to OpenCV's checkerboard camera calibration method [5]. We generate and distort images of checkerboards and scenes in the 3D rendering software Blender, and we leverage the known focal lengths and uniform distortion across sets of images to compare reprojection error between the two approaches and the results of undistortion using learned camera parameters.

Additionally, we have contributed:

1. A step-by-step interactive Jupyter notebook that presents a walkthrough of dataset generation from panoramas using the unified spherical model of distortion.
2. An updated implementation of the DeepCalib SingleNet classification model modified to use the latest language versions and libraries (Python 3, Tensorflow 2, and Keras 2.8).
3. A new set of synthetically generated checkerboard images and other scenes in Blender with known focal length and distortion on which to evaluate and compare camera calibration approaches.

2. Background/Related Work

2.1. OpenCV Camera Calibration

We leverage built-in checkerboard corner detection and camera calibration functions in the OpenCV library to estimate intrinsic camera parameters, extrinsic camera parameters, and camera distortion. In this implementation, intrinsic camera parameters are represented as a 3×3 camera matrix, extrinsic parameters are represented as a series of rotation matrices and translation vectors, and distortion is represented as the vector of radial and tangential distortion coefficients $(k_1, k_2, p_1, p_2, k_3)$. These coefficients obey the following model for a point (x, y) and distance to distortion center r:

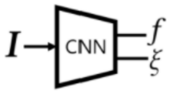


Figure 1: DeepCalib SingleNet architecture [3]

Radial distortion:

$$x_{\text{distorted}} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$y_{\text{distorted}} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

Tangential distortion:

$$x_{\text{distorted}} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{\text{distorted}} = [p_1(r^2 + 2y^2) + 2p_2xy]$$

OpenCV takes as input 3D points that represent the locations of each corner of the checkerboard in the world reference frame, and it finds the corresponding 2D points in each of the images using corner detection. This method of camera calibration and undistortion requires at least 10 test patterns (images) to perform well and assumes that all object points are coplanar.

2.2. DeepCalib Architecture

DeepCalib uses a deep-learning-based approach built upon the Inception-V3 CNN architecture. The authors frame the problem of camera parameter estimation in two ways: a **classification problem**, with softmax as the activation function for the output layers and cross entropy for the loss function, and a **regression problem** with logcosh loss, where the sigmoid activation function is used in the output layers.

For each of these approaches, the authors also consider three network architectures:

- **SingleNet** (Figure 1): Use a single network which takes the input image as the input and outputs both the focal length f and the distortion value ξ.
- **DualNet**: Use two networks where both take the input image as the input, and one outputs the focal length f, and the other outputs the distortion value ξ.
- **SeqNet**: Use two networks where one takes the input image as the input and outputs the focal length f, and the other takes the input image as well as the focal length f as the input and outputs the distortion value ξ.

2.3. Pano3D Dataset

We evaluate DeepCalib on images generated from panoramas in the Pano3D dataset; this dataset consists of two types of images: real-world indoor building scenes from Matterport3D and simulated environments from GibsonV2. We use photographs of panoramas within real-world building scenes originally derived from Matterport3D for evaluation in this report.

3. Approach

3.1. Generalizability to Pano3D Dataset

3.1.1 Data Generation with Unified Spherical Distortion Model

In our project, we reimplement the generation of images with arbitrary focal length and distortion from panoramas. This process uses the unified spherical model (Figure 2) where a 3D world point $\vec{P_w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ is projected onto a sphere at $\vec{P_s} = \frac{\vec{P_w}}{||\vec{P_w}||}$. This point on the sphere $\vec{P_s}$ is then projected onto the image plane at the location $\vec{p} = \begin{bmatrix} x \\ y \end{bmatrix}$. The projection of the point $\vec{P_s}$ on the sphere starts from the point located at $\begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix}$ above the sphere center (instead of at the sphere center) where ξ models the geometric distortion of the camera.

The projection process can be expressed as

$$\vec{p} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{Xf}{\xi\sqrt{X^2+Y^2+Z^2}+Z} + u_0 \\ \frac{Yf}{\xi\sqrt{X^2+Y^2+Z^2}+Z} + v_0 \end{bmatrix}$$

We observe that when $\xi = 0$ (i.e., no distortion), this is exactly the same as the standard pinhole perspective projection.

We then complete the following steps to generate a new image with an arbitrary focal length f and distortion value ξ from an input panorama:

1. Map the panorama linearly onto a unit sphere.

   To do so, define W and H to be the width and the height of the input panorama. For an arbitrary pixel $x \in (1, W)$ and $y \in (1, H)$ in the input panorama, it is linearly mapped to the point on the unit sphere with the azimuth angle $\theta \in (0, 2\pi)$ and the elevation angle $\phi \in (-\pi/2, \pi/2)$.

2. Rotate the unit sphere to the desired angle (e.g., a random angle). (This is equivalent to moving the virtual camera around.)

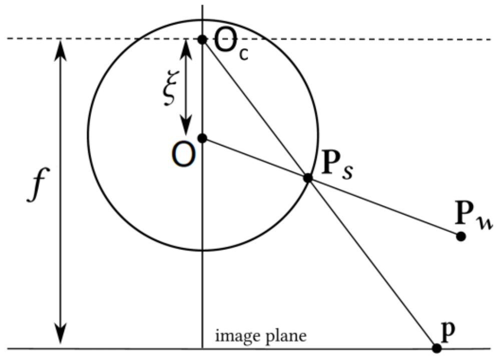


Figure 2: the unified spherical model [2, 4]

3. Back-project the positions on the unit sphere to form the new image with a specific focal length f and distortion value ξ.

   First, we construct an intrinsic matrix $\mathbf{K} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ with our desired focal length f (assuming the principal point is at the image center, the skew is negligible, and the pixel aspect ratio is 1).

   Then, for a 2D point $\vec{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ in homogeneous coordinates in our new image, we obtain the color of this pixel from the 3D position on the sphere

   $$\vec{P_s} = (\omega \hat{x}, \omega \hat{y}, \omega - \xi) \tag{1}$$

   where

   $$\begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} = \mathbf{K}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

   and

   $$\omega = \frac{\xi + \sqrt{1 + (1 - \xi^2)(\hat{x}^2 + \hat{y}^2)}}{\hat{x}^2 + \hat{y}^2 + 1}$$

   as shown in the paper.

By doing this for many panoramas and with different focal lengths, distortion values, and camera configurations (i.e., unit sphere rotation), we automatically generate a dataset of images with various configurations along with their ground truth focal lengths and distortion values. An example of the results of this procedure is included in Figure 3. Notice that rather than the original Sun360 dataset, we use 360° FOV panoramas from Pano3D.

To clearly illustrate this process, we also create an interactive, step-by-step Jupyter notebook at `https://hesyifei.github.io/CS231A-DeepCalib/` that replicates this dataset generation process. [1]

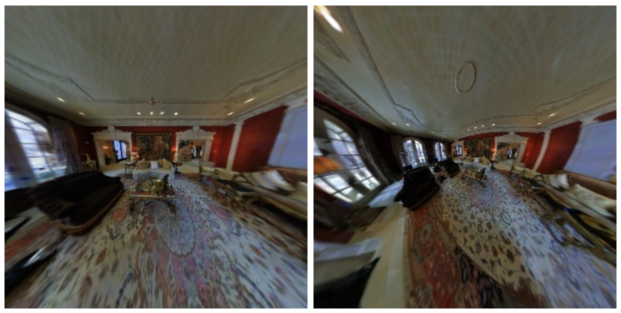


Figure 3: Two images with $f = 40$ pixels and levels of distortion $\xi = 0$ and $\xi = 0.52$, generated from the same panorama in Pano3D

3.1.2 Image Undistortion

During qualitative analysis, we also need to perform undistortion on the dataset. To undistort an arbitrary input distorted image, we first apply the model to estimate the focal length f and distortion parameter ξ. Then we back-project the input distorted image on a unit sphere with Equation 1. This unit sphere is then projected on the image plane with the desired intrinsic parameters to form a new image as described in Section 3.1.1.

3.1.3 Testing Procedure

We use weights from a pretrained model of DeepCalib's SingleNet Classification architecture to evaluate performance on Pano3D. This architecture was chosen because in DeepCalib's results, it yielded higher accuracy for parameter prediction relative to the DualNet and SeqNet architectures. While we initially attempted to additionally retrain DeepCalib on the Pano3D dataset, we shifted our approach to solely leverage existing weights because of low training efficiency and performance. After over 15 hours of training (approximately 20 epochs), validation accuracy remained nearly unchanged from the initial values of 2% and 4%, respectively. One cause for this discrepancy may be the difference in the size of datasets used for training; Pano3D contains 10,800 panoramic views from Matterport, while the currently unavailable Sun360 dataset contains over 67,000 panoramic views.

3.2. Comparison Against OpenCV Camera Calibration

In order to evaluate OpenCV camera calibration against DeepCalib, we first generated images of checkerboards as

[1]Note that the HTML file is around 70MB large, so please allow some time to load it.

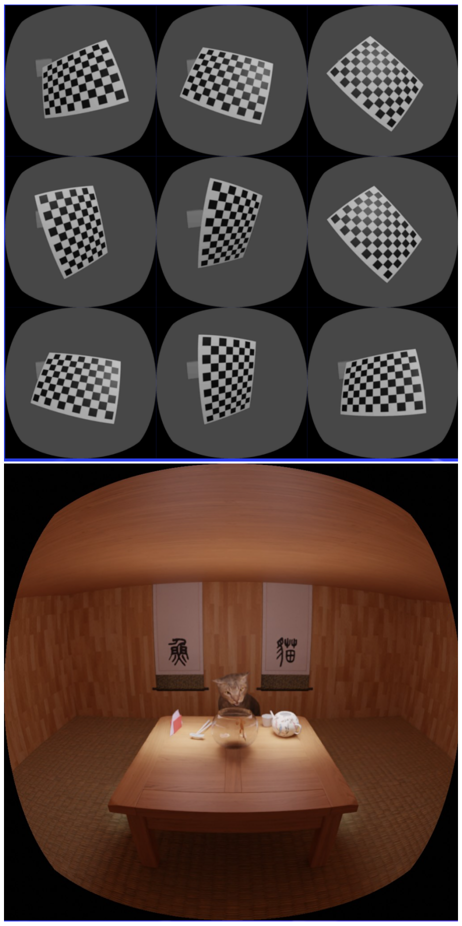


Figure 4: A set of distorted checkerboard images and a synthetic scene, taken on a virtual camera with a focal length of 10mm

well as other scenes with a known ground truth focal length and uniform distortion. We leveraged the 3D rendering software Blender to do so; for each camera calibration experiment, we captured a set of 24 images of checkerboards at various orientations and positions taken with the same virtual camera. We then used this camera to capture an image of a scene. Finally, we applied the same level of distortion to both the checkerboard and scene image using Blender tools for image postprocessing. See Figure 4 for an example scene and corresponding set of synthetic checkerboards. Though this example includes images with checkerboard close to the center of the scene, we have also generated sets with checkerboard located throughout the image plane.

The original OpenCV camera calibration method uses ground truth points in the world frame, and it subsequently predicts both intrinsic and extrinsic camera parameters. DeepCalib, however, does not predict the extrinsic rotation and translation matrices, and so we evaluated only the accuracy of OpenCV's intrinsic parameter and distortion prediction. In this vein, we compute the reprojection error for images that OpenCV has calibrated using known 3D corner locations in the camera frame, precomputing these locations in Blender and setting the extrinsic parameters to a rotation matrix of $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and a translation vector of $T = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$.

4. Experiment

4.1. Generalizability to Pano3D Dataset

We are interested in how DeepCalib performs when applied to a new dataset, namely the Pano3D dataset.

4.1.1 Quantitative Analysis

We first replicate the confusion matrix and cumulative error distribution presented in the original paper.

To obtain results related to the focal length, for each panorama, we generated 47 images of various orientations with distortion value of 0 and focal lengths on a range between 40 and 500px with a step size of 10px. In total, we generated 19975 images from 425 panoramas from the Pano3D dataset. Then we compare the ground truth focal length with the predicted focal length for each of these images. As we can see in Figure 5, the performance decreases when the focal length increases. This aligns with the original paper's observation and demonstrates the model's resilient to changes in dataset (e.g., distribution of indoor and outdoor panoramas).

To obtain results related to the distortion value, for each panorama, we generated 21 images of various orientations with focal length of 40px and distortion values on a range between 0 and 1.2 with a step size of 0.06. In total, we generated 19992 images from 952 panoramas from the Pano3D dataset. Then we compare the ground truth distortion value with the predicted distortion value for each of these images. As we can see in Figure 6, the performance increases when the distortion value increases. This also supports the original paper's observation.

4.1.2 Qualitative Analysis

We also perform a qualitative analysis on the undistortion process. As we see in Figure 7, DeepCalib performed well on undistorting the image, supporting our observation that images with low ground truth focal length and high ground truth distortion value perform more accurately. However, we can still see some distortion around the edges of the undistorted image, presumably because the predicted distortion value is smaller than the ground truth distortion value.

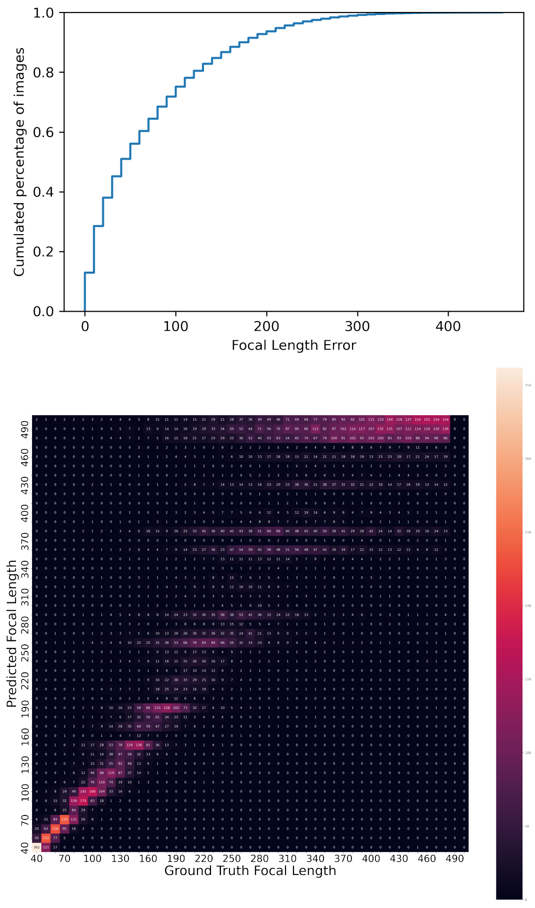


Figure 5: Cumulative error distribution (left) and confusion matrix (right) of DeepCalib's prediction vs. ground truth for various focal length on the generated images from the Pano3D dataset

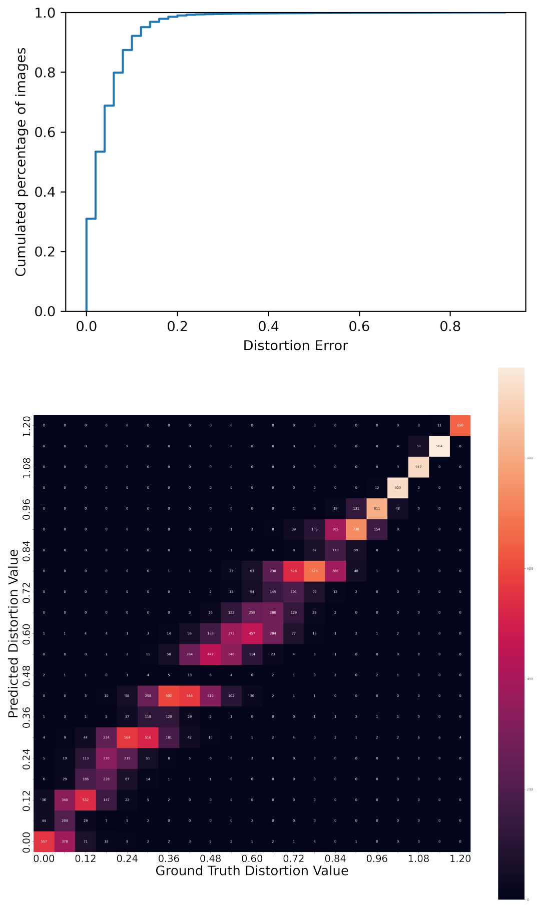


Figure 6: Cumulative error distribution (left) and confusion matrix (right) of DeepCalib's prediction vs. ground truth for various distortion values on the generated images from the Pano3D dataset

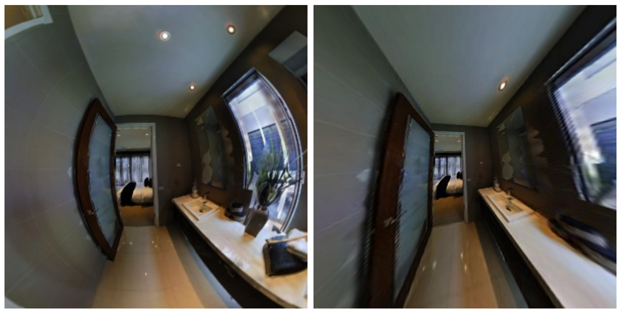


Figure 7: A distorted image generated from Pano3D with ground-truth $f = 140$px and $d = 1.14$ and its undistorted counterpart with predicted parameters $f = 140$px and $d = 1.06$

4.2. Comparison against OpenCV

4.2.1 Quantitative Analysis

We use reprojection error as the primary quantitative metric to compare OpenCV and DeepCalib's accuracy at intrinsic camera parameter prediction. For the j'th checkerboard image, we project the known 3D corner points in the camera frame into the image plane using the predicted intrinsic camera matrix and distortion, obtaining 2D points $\vec{p_j}_1, \ldots, \vec{p_j}_{70}$ corresponding to each of the 70 predicted corner locations. We compute the mean reprojection error e across all 24 checkerboard images (each image has size 299×299 pixels as in the original paper) with respect to the ground truth corner points $\vec{p'_j}_1, \ldots, \vec{p'_j}_{70}$ (found using corner detection) using the following formula:

$$e = \frac{1}{24} \sum_{j=1}^{24} \left(\frac{1}{70} \sum_{k=1}^{70} \|\vec{p_j}_k - \vec{p'_j}_k\|_{L2} \right)$$

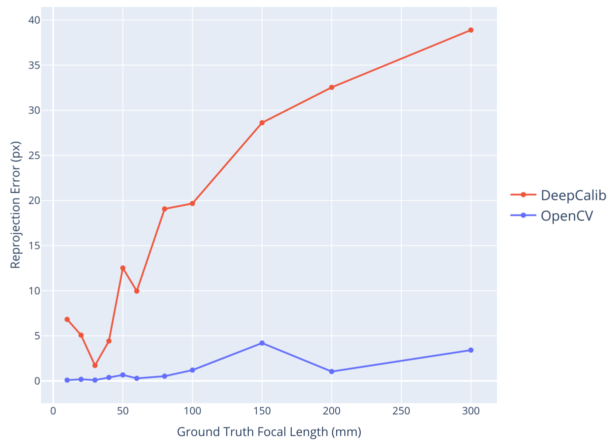


Figure 8: Relative performance of OpenCV and DeepCalib on various focal lengths with no distortion, measured in mean reprojection error (pixels)

To compute reprojection error as a metric of accuracy for DeepCalib, we run focal length and distortion prediction on the synthetic scene image and then apply these parameters using the unified spherical model to the checkerboard images (taken with the same virtual camera) to obtain the predicted corner locations p_j in the image plane. Like the DeepCalib authors, we chose not to predict camera parameters from checkerboard images directly and instead opted to use the scene image taken on the same camera because it more closely reflected the real-world images upon which the model was trained. The synthetic checkerboard images lack substantial background and context, while the scenes included context such as the floor, ceiling, and table in Figure 4, rendering them more representative both of training images and of realistic settings for camera calibration.

We found that results for reprojection error on DeepCalib were more consistent with zero distortion; this is likely because of the ambiguity in intrinsic parameters introduced by the unified spherical distortion model, where multiple combinations of f and ξ can lead to the same distorted image.

We summarize the relative reprojection errors for images with varying focal lengths and no distortion in Figure 8 (see Figure 9 for an example reprojection). As expected, we see that across all focal lengths, reprojection error is substantially lower for the OpenCV checkerboard camera calibration method than for DeepCalib. We also see that DeepCalib's reprojection error gets increasingly worse as the focal length increases, supporting the result depicted in Figure 5.

Nevertheless, despite using only a single image input for prediction rather than a set of images with known 3D points, reprojection error from DeepCalib remains quite low, even for the synthetic images.

4.2.2 Qualitative Analysis

We also perform a qualitative comparison between images where distortion removal has been performed using DeepCalib and OpenCV. Examples of undistorted checkerboard and scene images are depicted in Figures 10, 11, and 12. We see that DeepCalib performs similarly to OpenCV when applied to the checkerboard images, with only slight distortion in the corners of the board. In the scene images, OpenCV tends to display increased distortion in the outer corners of images relative to DeepCalib, or generate an `roi` (region of interest) that crops out large portions of the image. On the other hand, DeepCalib performed well on nearly all of the example synthetic scenes we performed this qualitative analysis on.

5. Conclusion

In this report, we present an analysis of the effectiveness and generalizability of the DeepCalib approach for camera calibration. When compared to OpenCV's checkerboard-based calibration method, we observe that DeepCalib performs well on combating distortion on synthetic scenes but is not as effective on images where ground-truth world points are known. We also note that DeepCalib generalizes effectively to the Pano3D dataset and similarly has decreased performance for high focal lengths and low distortion values.

Future work includes computing the relative performance of DeepCalib and traditional calibration approaches in the presence of distortion. This would require decoupling the relationship between focal length and distortion in the unified spherical model. In addition, a promising direction of research is evaluating the relative performance of DeepCalib and other deep learning-based approaches on synthetic images, both in Blender and in larger datasets including the simulated GibsonV2 data present in Pano3D.

References

[1] G. Albanis, N. Zioulis, P. Drakoulis, V. Gkitsas, V. Sterzentsenko, F. Alvarez, D. Zarpalas, and P. Daras. Pano3d: A holistic benchmark and a solid baseline for 360° depth estimation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun 2021.

[2] J. P. Barreto. A unifying geometric representation for central projection systems. *Computer Vision and Image Understanding*, 103(3):208–217, 2006.

[3] O. Bogdan*, V. Eckstein*, F. Rameau, and J.-C. Bazin. Deepcalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view cameras. *ACM SIGGRAPH European Conference on Visual Media Production*, (16):1–10, 2018.

[4] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings 2007 IEEE*

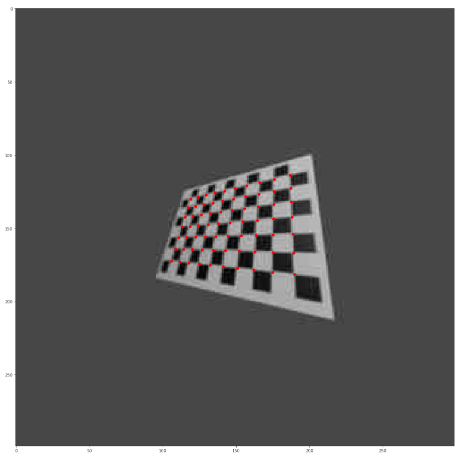
International Conference on Robotics and Automation, pages 3945–3950. IEEE, 2007.

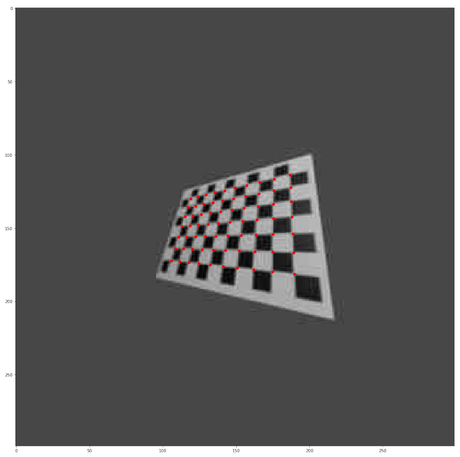
[5] OpenCV. Camera calibration tutorial.

[6] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Supplementary Materials

- Jupyter notebook explaining data generation process: `https://hesyifei.github.io/CS231A-DeepCalib/`
- All source code is available at `https://github.com/hesyifei/CS231A-FinalProject` and `https://github.com/hesyifei/CS231A-DeepCalib/` upon request


(a) Ground truth corner points


(b) Corner points reprojection using the intrinsic matrix and distortion predicted by OpenCV

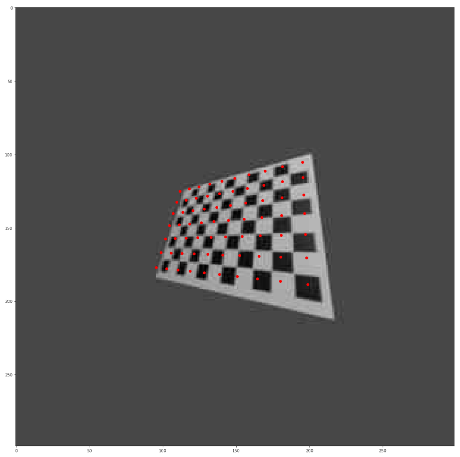

(c) Corner points reprojection using the focal length and distortion value predicted by DeepCalib

Figure 9: Example corner points reprojections using ground truth 3D corner points in camera frame

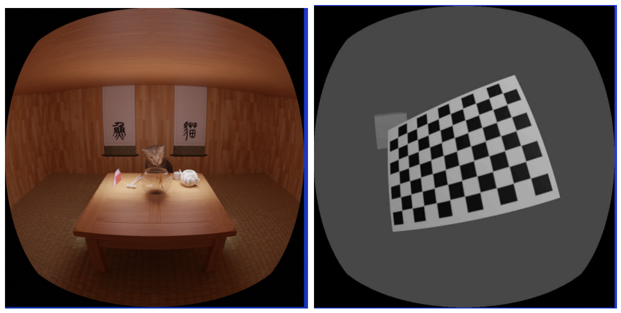


Figure 10: Original distorted scene and checkerboard images

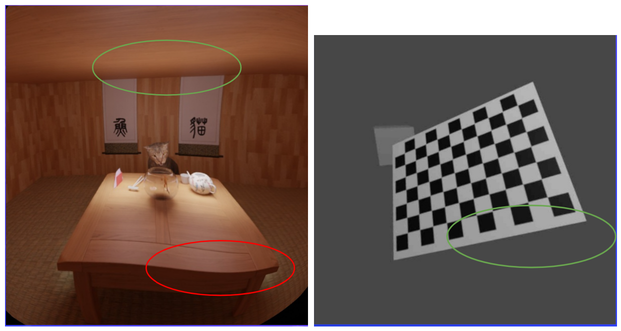


Figure 11: Undistorted images using OpenCV. Areas with more effective undistortion are labeled in green, and areas with relatively ineffective undistortion are labeled in red

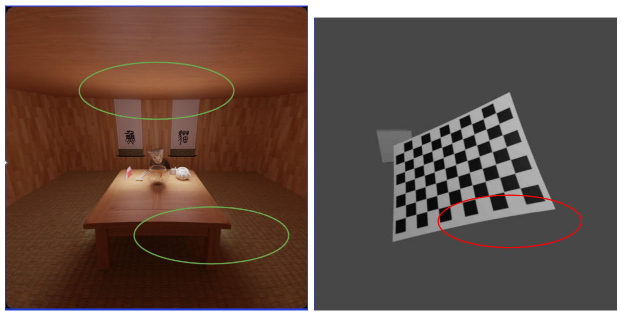


Figure 12: Undistorted images using DeepCalib