# Deep Learning Self-Calibration from Planes

Hauke Brunken and Clemens Gühmann

Chair of Electronic Measurement and Diagnostic Technology
Technische Universität Berlin

## ABSTRACT

Many applications in computer vision require calibrated cameras, but identifying camera calibration parameters is a tedious task. Common methods require custom-built calibration patterns from which many images from different perspectives have to be taken. This research introduces a novel auto calibration method to reduce the work to a minimum.

The method utilizes a neural network framework and learns the parameters through backpropagation and gradient descent. Three views of the same arbitrarily textured flat surface are used as an input. Two of the views are transformed to match the third reference view by plane homographies. Feature maps are extracted and the views are compared with their help. In- and extrinsic, as well as distortion parameters can then be learned by maximizing the similarity between the transformed views and the reference view.

The results show that the method is able to find the calibration parameters of artificially distorted images. Results with real camera images are comparable to common methods that require planar calibration patterns, which makes the proposed method a quick alternative.

**Keywords:** Self-calibration, camera calibration, lens distortion, neural network, deep learning.

## 1. INTRODUCTION

If metric properties of images are to be reconstructed, the intrinsic camera parameters need to be known. This is the case for metric reconstruction from single views, depth reconstruction from stereo cameras and for augmented reality applications. Other applications rely on the pinhole camera model and assume undistorted camera images, an example being image stitching to create panoramas. Here, at least the distortion parameters are required.

The process of finding these parameters is called camera calibration. For this purpose, a calibration target is observed from one or multiple perspectives. Depending on the shape of and the knowledge about the calibration target, camera calibration can be divided into different categories. Following [11] these are:

- Camera calibration by a 3D target with control points. The 3D coordinates of which have to be known to a high precision.
- Camera calibration by a planar target with control points. The 2D coordinates within the plane have to be known to a high precision.
- Camera self-calibration by a 3D or planar target without any control points.

Combinations of these categories are also possible. In [11] it is stated that the precision of extracted parameters from the aforementioned methods decrease in the given order. However, the manufacture and high precision measurement of a 3D calibration target is often unfeasible. A 2D target is easier to create and is the standard calibration method using the popular OpenCV and MATLAB libraries. Nevertheless, it would be more practical not to have to use a custom-made pattern.

Self-calibration is the extraction of calibration parameters from unstructured scenes. It is widely used in structure from motion methods, where a metric reconstruction is performed from an unordered collection of images from uncalibrated cameras. For that purpose, the in- and extrinsic, as well as the distortion parameters, are reconstructed together with 3D point cloud coordinates of feature point matches in a bundle adjustment step. Feature points have to be found and matched across multiple views in a preliminary step. The bundle adjustment then minimizes the reprojection error of feature points between different views. The focus of such methods, however, is on 3D reconstruction and not on camera calibration.

Planar scenes on the other hand have been used for the purpose of self-calibration and in [10] promising results are shown. The general procedure remains the same for 3D and planar calibration targets and point correspondences must be found first. However, one degree of freedom is removed.

Every self-calibration method that is based on the concurrent localization of feature points together with the extraction of camera parameters has three important limitations:

1. Feature point matches between different images must be exact and actually belong to the same object point. Wrong matches would have a negative effect on the calibration result.
2. Common distortion models are based on polynomials. Since polynomials are bad at extrapolation, feature points have to be found uniformly across the entire image space. Not covering the entire image space has been shown to greatly decrease performance of self-calibration systems [7]. This is also true when most feature points lie in the center and less are covering the edges, because the number of points has a weighting effect. While images are being captured, it is difficult to predict where an algorithm for feature detection will find feature points.
3. Only image coordinates containing a feature point are used for calibration. This discards information present in other coordinates.

The proposed self-calibration method also uses planar scenes, but takes a fundamentally different path. Instead of relying on feature point matches, feature maps of entire images are calculated and used for matching between the views. Instead of optimizing a distance measure between reprojected features, the overall similarity between views is optimized by a neural network training approach. That makes it possible to do a full calibration from the absolute minimum of three views of a planar scene.

## 2. RELATED WORK

An overview of calibration techniques can be found in [9]. The most basic one utilizes 3D control points, which can be found in one or multiple images. If the pinhole camera model is adopted, a system of linear equations can be formulated, which can be solved by the direct linear transform. Results are then refined by non-linear optimization. If distortion is taken into account, the corresponding parameters can also be included in the non-linear optimization step.

Self-calibration methods that utilize 3D scenes are mainly a by-product of structure from motion from uncalibrated cameras. The fundamentals are also given in [9]. The basic idea is to compute a projective reconstruction from matched feature point correspondences across multiple images. That means projection matrices as well as 3D point coordinates need to be estimated. If a pinhole camera model is assumed, the calibration matrices can be reconstructed from projection matrices. In a following bundle adjustment step, point coordinates, linear camera parameters and distortion parameters are optimized by minimizing reprojection errors. Such an approach was shown in [16]. Self-calibration can be used for a single camera with fixed intrinsic parameters, but also for a collection of different cameras that picture the same scene. A critical part in self-calibration is the automatic matching of corresponding feature points between images. It has been addressed in [8], where an iterative search algorithm for feature matches is shown. It incorporates only a single radial distortion parameter, although the method itself is not limited to one parameter. A more recent work [5] follows a similar approach and incorporates more distortion parameters.

If planar targets are used, linear calibration parameters can be reconstructed from plane homographies [20] which in turn can be estimated from feature point matches between pairs of images. It is therefore a self-calibration technique. If the planar target contains a known 2D pattern that can be found in the images, a metric reconstruction can be performed. Distortion parameters can then be incorporated and results can be refined by a non-linear optimization of the reprojection error. This method is commonly referred to as Zhang's method [21] and is probably the most used method for camera calibration [11]. Variants of it are implemented in the OpenCV [2] and MATLAB [19] libraries.

A plane-based self-calibration method that considers lens distortion is shown in [14]. Just like in the 3D self-calibration methods, feature matching is a critical point and has to be handled. On real images in [14] the problem is solved by matching the points manually. In [10] another algorithm that considers lens distortion is described. It differs in the optimization approach, but also struggles with feature mismatches, as their results show.

## 3. METHOD

Under the assumption of the pinhole camera model, images of a plane are related by plane homographies. An image from one view can be warped to a reference view, where both images should match. The homography depends on the relative orientation between the cameras, the location of the plane in space and the calibration matrices of the cameras. The intrinsic camera parameters are assumed to be constant for all cameras. The idea is to transform two views by a plane homography to match a reference view. The views are compared with the help of feature maps and the orientation between cameras are learned together with the calibration matrix, until the views match exactly. Lens distortion is easily integrated, because for distorted images the linear relation between the views does not hold.

In the following paragraphs the pinhole camera model with lens distortion will be recapitulated. Next, the relation between views by plane homographies will be discussed. The warping and comparison of images will be explained, which is needed for the following optimization procedure. In the last paragraph the number of unknowns that can be extracted from only three views will be discussed.

In this work three views of a plane are used, although the proposed method is not limited to three views only. The views of the plane will be referred to as view one, view two and reference view. In the following, the proposed method is referred to as DLSC.

### 3.1 Camera model

The perspective projection of a pinhole camera as described in [9] is modeled by

$$\mathbf{x} = K\,[\,R \mid \mathbf{t}\,]\,\mathbf{X}, \tag{1}$$

with the 2D homogenous image coordinates $\mathbf{x} = (x, y, z)^{\mathrm{T}}$, 3D homogenous world coordinates $\mathbf{X} = (X, Y, Z, 1)^{\mathrm{T}}$ and the camera matrix $K$. $R$ is the rotation of the camera in relation to the world coordinate system and $\mathbf{t} = -R\tilde{C}$ with $\tilde{C}$ being the coordinates of the camera center. The camera matrix is given by

$$K = \begin{bmatrix} f_x & s & x_0 \\ & f_y & y_0 \\ & & 1 \end{bmatrix}. \tag{2}$$

$f_x$ and $f_y$ are the focal lengths in x and y-direction, $(x_0, y_0)$ are the coordinates of the principal point and $s$ is the image skew.

Image coordinates without lens distortion are given by

$$u = \frac{x}{z} \tag{3}$$

$$v = \frac{y}{z} \tag{4}$$

Lens distortion can be modeled after the perspective projection. A widely used model is Brown's model [3]:

$$u_d = u + \bar{u}(K_1 r^2 + K_2 r^4 + K_3 r^6) + 2P_1\bar{u}\bar{v} + P_2(r^2 + 2\bar{u}^2) \tag{5}$$

$$v_d = v + \bar{v}(K_1 r^2 + K_2 r^4 + K_3 r^6) + 2P_2\bar{u}\bar{v} + P_1(r^2 + 2\bar{v}^2)\,, \tag{6}$$

with $\bar{u} = u - u_p$, $\bar{v} = v - v_p$, $r^2 = \bar{u}^2 + \bar{v}^2$, where $(u_p, v_p)$ are the coordinates of the center of distortion. $(u, v)$ are undistorted image coordinates and $(u_d, v_d)$ are distorted image coordinates.

This work follows the camera model of [1] that is implemented in the OpenCV [2] and MATLAB [19] libraries. It differs in that the distortion is calculated in normalized camera coordinates and that the center of distortion matches the principal point. Since modern cameras have zero skew (the pixels on the sensor are aligned on a rectangular grid), the skew is not further considered. In OpenCV the camera model is extended by a rational model, which is also employed in this work. The final model is given by the following equations:

$$\mathbf{x} = [\,R \mid \mathbf{t}\,]\,\mathbf{X} \tag{7}$$

$$u_n = \frac{x}{z} \tag{8}$$

$$v_n = \frac{y}{z} \tag{9}$$

$$u_{n,d} = u_n \frac{1 + K_1 r^2 + K_2 r^4 + K_3 r^6}{1 + K_4 r^2 + K_5 r^4 + K_6 r^6} + 2P_1 u_n v_n + P_2 (r^2 + 2{u_n}^2) \tag{10}$$

$$v_{n,d} = v_n \frac{1 + K_1 r^2 + K_2 r^4 + K_3 r^6}{1 + K_4 r^2 + K_5 r^4 + K_6 r^6} + 2P_2 u_n v_n + P_1 (r^2 + 2{v_n}^2) \tag{11}$$

$$u_d = f_x u_{n,d} + u_0 \tag{12}$$

$$v_d = f_y v_{n,d} + v_0 \tag{13}$$

where $r^2 = {u_n}^2 + {v_n}^2$. $(u_n, v_n)$ are normalized undistorted image coordinates. $(u_{n,d}, v_{n,d})$ are normalized distorted image coordinates and $(u_d, v_d)$ are the distorted image coordinates.

### 3.2 Plane homography

A plane that coincides with the XY-plane of the world coordinate system is transformed by a pinhole camera from 3D homogenous world coordinates to 2D homogenous camera coordinates by

$$\mathbf{x} = K\,[\,R \mid \mathbf{t}\,] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathrm{K}\,[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{14}$$

where $H = K\,[\,\mathbf{r}_1,\ \mathbf{r}_2,\ \mathbf{t}\,]$ is a homography. If two images of the plane exist, the coordinates of one image can be transformed such that they match those of the reference view by

$$\mathbf{x}_{ref} = H_{ref} H_1^{-1} \mathbf{x}_1. \tag{15}$$

### 3.3 Inverse warping

Assuming all parameters are known, it is now possible to undistort all images and to warp view one and view two such that they match the reference view. Undistorting and warping are done in a single step by inverse warping [17]: For every pixel of the new undistorted warped image the corresponding location in the original distorted image is calculated. Since the calculated locations are generally non-integer values, bilinear interpolation is used to estimate the value of a pixel, which is then copied to the undistorted warped image. Bilinear interpolation is differentiable by following the approach in [12], which is readily implemented in neural network frameworks. Differentiability is a requirement for learning the calibration parameters by backpropagation and gradient descent.

For the reference view, the calculation of pixel coordinates is easily accomplished by applying Equations 10 and 11 in normalized image coordinates. In case of the other views the plane homography $H_{ref} H_i^{-1}$ that relates the $i$th's camera to the reference camera has to be applied prior to the distortion function.

### 3.4 Comparing the images

To compare if the undistorted warped images match the undistorted reference image, feature maps are calculated by a neural network. In [6] a neural network for depth estimation from stereo images is proposed. It uses a Siamese network to extract feature maps from a left and right image, which are then concatenated for depth estimation. The idea was further developed in [4] for depth extraction from flat surfaces.

The feature extraction network consists of several convolutional layers and a spatial pyramid pooling module in order to implement region level features. It transforms an input image of dimension $W \times H \times 3$ into an array of feature maps $\mathbf{F}$ of dimension $W/4 \times H/4 \times F$, where $F$ is the number of feature maps. Further details may be found in [6].

In this work the feature map extraction network is reused with the learned weights from [4]. Finally, a similarity score $sc$ is calculated by the mean cosine similarity between feature vectors of each pixel of the feature map of both images and the reference image:

$$sc = \frac{1}{2\frac{W}{4}\frac{H}{4}} \sum_{i=1}^{2} \sum_{u',v'} \frac{\mathbf{F}_i(u',v') \cdot \mathbf{F}_{ref}(u',v')}{|\mathbf{F}_i(u',v')| |\mathbf{F}_{ref}(u',v')|} . \quad (16)$$

$\mathbf{F}_i(u',v')$ is the feature vector at the coordinates $(u',v')$ of the feature map with index $i$.

### 3.5 Optimization

The similarity score depends on the spatial relation between both cameras and the location of the plane, on the camera matrix and on the distortion parameters. All calculations are implemented in a neural network framework and are differentiable to the parameters. Using a rough estimate of the parameters, they can thus be learned in a training loop through backpropagation and gradient descent, while the weights of the feature extraction network remain unchanged.

### 3.6 Number of unknowns

Since a total of three images of a plane is used and setting aside distortion for now, there are two independent homographies that relate the images to each other. A homography possesses 8 independent variables, thus 16 parameters of the linear camera model can be reconstructed. With respect to the reference camera, 3 translation and 3 rotation parameters are required to describe the location of another camera. The plane is located in front of the reference camera and the distance and 2 rotations are necessary to describe its location. By fixing the distance at 1, the overall scale of the scene is fixed. Altogether 14 parameters describe the layout of the scene, which leaves 2 parameters that can be extracted from two homographies. Let's assume these are $f_x$ and $f_y$.

The warped views only match the reference view under the assumption of the pinhole camera model. The principal point is indirectly estimated by the distortion model, because it matches the center of distortion. The distortion model itself can be learned from only two images as is shown in [18], since every pixel functions as one condition on a system of equations. It follows that if the optimization is able to match the views with distortion, the distortion must have been removed.

## 4. IMPLEMENTATION DETAILS

In the implementation two important aspects have to be considered. These are initialization of parameters and an encoding of rotation parameters and will be described next. Some details about the training procedure and hardware requirements will also be discussed.

### 4.1 Initialization

To start the gradient descent training loop, an initialization of the parameters is necessary. For the loss function to calculate a meaningful output, the images have to overlap at least in parts. That can also be checked visually by transforming the images to the reference view. If no prior knowledge about camera parameters is available, they can be roughly estimated.

#### 4.1.1 Distortion

The parameters of the distortion function are estimated first. A plane homography can only relate two views to each other under the assumption of the pinhole camera model without distortion. That means the better the parameters are estimated, the better a homography relates the views to each other. That is utilized in the following way:

- Points from all views are extracted by the SIFT feature extractor. They are matched between the reference view and view one and between the reference view und view two by a brute-force method and checked for validity by a ratio test.
- A loss function is defined that undistorts feature point coordinates and counts the number of inliers under the homography assumption. The homography is found by a RANSAC method. For undistorting the images, a camera matrix with a principal point that is located at the center is used. It is assumed that $f_x$ and $f_y$ are similar and can be set to the same arbitrary value for initialization.

(a) Distorted reference view.

(b) All three views are blended after undistortion and transformation to reference view with parameters estimated by the initialization method as described in Section 4.1.

(c) All three views are blended after undistortion and transformation to reference view with parameters estimated by the proposed DLSC method.

Figure 1. Images of a flat surface that were used to calibrate the GoPro camera. The tile pattern is not used by the algorithm, but it visualizes the distortion.

- The loss is the negative number of inliers of the RANSAC method. It is optimized by a non-linear optimization method by varying the distortion parameters. In the experiments only $K_1$ and $K_2$ are varied and the optimization is carried out by simulated annealing with Nelder-Mead as a local method.

#### 4.1.2 Focal length

To estimate the focal length parameters a similar approach is used. Once the point coordinates are undistorted, plane homographies can be estimated. With a known camera matrix, each homography can be decomposed into a rotation matrix and a translation vector that relate two cameras to each other, and into a plane normal that describes the orientation of the plane [15]. The decomposition is ambiguous, but the correct solution can be chosen by comparing the plane normal to the physically possible solution. The plane normal from each homography should point into the same direction as it corresponds to the same physical plane. The following approach is used to estimate the focal length parameters:

- A loss function is defined that undistorts feature point coordinates with the previously found distortion parameters and a new camera matrix. The distortion parameters thereby have to be rescaled according to the new focal lengths and only inliers of the RANSAC approach of the previous step are used. It then finds and decomposes plane homographies as in the previous step. The loss is the dot product between the plane normal of the first and the second decomposed homography.
- The loss function is optimized by a non-linear optimization method by varying the focal length parameters. $K_1$ and $K_2$ then have to be adjusted accordingly. The optimization is again carried out by simulated annealing with Nelder-Mead as a local method.

#### 4.1.3 Extrinsic parameters

The locations of the two cameras are found by the homography decompositions of the previous step. Since there is one plane normal per homography decomposition, which are not exactly identical, the normalized mean is taken. The location of the plane in 3D space is found by triangulating the feature points that were inliers of the RANSAC algorithm in the previous step. The intersection of the z-axis of the reference camera and the plane can then be found. Since the distance between the reference camera and the intersection with the plane is fixed at 1 during the optimization (Section 3.6) the length of the translation vectors of camera one and two are adjusted.

The result of the initialization is visualized in Figure 1b. It shows the undistorted reference view of Figure 1a with the blended first and second view. Results for different test cases are shown in Table 2.

### 4.2 Encoding of orientation parameters

To describe a rotation, 3 parameters are required, as opposed to the 9 values that a rotation matrix contains. For the optimization an encoding is needed in order to estimate only as many parameters as necessary. One possibility is to describe the rotations by Euler angles. These have the advantage of consisting of exactly 3 values, but suffer under discontinuities, which make them unsuitable for optimization [13]. Instead, pairs of axis and angles are used to describe the rotations of

the cameras. The axes are described by unit length vectors. Since the condition of unit length is not guaranteed by the gradient descent optimization, they are renormalized after every iteration.

To encode the orientation of the plane, two Euler angles are used, since they are physically restricted to a suitable range. Axis-angle representations and Euler angles are then converted to rotation matrices.

### 4.3 Training

Table 1. Learning rates used during optimization.

| Parameter | Learning Rate |
|---|---|
| camera rotation | 1e-3 |
| camera translation | 1e-3 |
| plane orientation | 1e-3 |
| $f_x$, $f_y$ | 1e-1 |
| $x_0$, $x_1$ | 1e-1 |
| $K_1$, $K_2$ | 1e-2 |
| $P_1$, $P_2$ | 1e-2 |
| $K_3$ | 1e-3 |
| $K_4$, $K_5$, $K_6$ | 1e-4 |

To decrease the training time, all images are downscaled by a factor of 4 in both dimensions. As soon as the parameters converge, downscaled images with a factor of 2 are used. Finally, images at full resolution are processed. When the scaling of the images is changed, the scaling of the camera matrix also has to be adjusted. The distortion parameters are independent of the camera matrix. The Adam optimizer is used in all experiments. The learning rates are shown in Table 1.

Figure 1c shows the reference view with the blended first and second view after training. All parameters are estimated precisely. As a result, the single images in the figure cannot be distinguished anymore.

### 4.4 Hardware requirements

All calculations are implemented in a neural network framework, which requires GPUs for computation. The current implementation utilizes 8 GB of GPU memory per view at a resolution of 1920 x 1200 pixels. The calculations for every view are calculated on a dedicated GPU.

## 5. RESULTS

In order to evaluate the results, artificial views with artificial distortion of a plane have been created, such that all parameters are known. The parameters are reconstructed by the DLSC method and for comparison by the method described in [1], which is implemented in the OpenCV library. Afterwards, results with real camera images are presented. Not only the camera parameters themselves are compared, but also the resulting distortion and the reprojection error of calibration patterns. In all cases the reference camera is pointed approximately vertically to the plane, which helps finding the correct homography decomposition (Section 4.1.2).

### 5.1 Reprojection error

The correct parameters for real cameras are unknown. The estimated parameters can therefore only be compared to those estimated by another calibration procedure. To make a comparison anyway, the reprojection error of a flat calibration pattern is used.

If one considers the calibration pattern to be fixed in space and a camera freely moving, the coordinates of the pattern are known with $z$ coordinates fixed at zero. With known matches between 3D world coordinates and 2D image coordinates of the pattern, the location of the camera in 3D space can be found by minimizing the error between the projected 3D coordinates and the image coordinates. That can be done for every view separately. The mean of all reprojection errors of all points across all views is considered the overall reprojection error. It has been calculated for artificial and for real images.

Table 2. Results for artificial and real images. The rows marked by - show the real parameters. Parameters found by the initialization method are referred to as Ini. and those found by the proposed method as DLSC. Values shown in grey are not optimized, but stay fixed.

| | | Parameters | | | | | | | | | | | | reprojec- | Distortion | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| camera | method | $f_x$ in px. | $f_y$ in px. | $p_0$ in px. | $p_1$ in px. | $K_1$ | $K_2$ | $P_1$ (e-3) | $P_2$ (e-3) | $K_3$ | $K_4$ | $K_5$ | $K_6$ | tion err. in px. | median in px. | mean in px. | MdRAE in % | MRAE in % |
| artificial # 1 | - | 1386.5 | | 1060.0 | 600.0 | -0.240 | 0.050 | | | | | | | 0.05 | 26.7 | 39.5 | ref. | ref. |
| | Ini. | 1323.7 | | 960.0 | 600.0 | 0.176 | 0.019 | | | | | | | 1.69 | 21.8 | 31.6 | 0.55 | 4.12 |
| | OpenCV | 1387.2 | | 1060.6 | 599.3 | -0.240 | 0.050 | | | | | | | 0.04 | 26.6 | 39.5 | 0.01 | 0.03 |
| | DLSC | 1385.6 | | 1058.7 | 600.2 | -0.240 | 0.050 | | | | | | | 0.05 | 26.7 | 39.5 | 0.01 | 0.05 |
| artificial # 2 | - | 1386.5 | 1663.9 | 975.0 | 590.0 | -0.240 | 0.050 | | | | | | | 0.05 | 23.3 | 35.2 | ref. | ref. |
| | Ini. | 1289,8 | 1558,5 | 960.0 | 600.0 | -0.140 | -0.018 | | | | | | | 2.16 | 16.7 | 26.8 | 0.90 | 3.01 |
| | OpenCV | 1387.3 | 1664.7 | 975.6 | 589.4 | -0.240 | 0.050 | | | | | | | 0.04 | 23.3 | 35.2 | 0.01 | 0.03 |
| | DLSC | 1384.8 | 1661.0 | 972.2 | 590.0 | -0.239 | 0.049 | | | | | | | 0.09 | 23.3 | 35.2 | 0.03 | 0.10 |
| artificial # 3 | - | 840.9 | | 975.0 | 590.0 | -0.250 | 0.100 | -1.000 | -0.010 | 0.010 | | | | 0.11 | 61.6 | 65.1 | ref. | ref. |
| | Ini. | 824.3 | | 960.0 | 600.0 | -0.261 | 0.115 | | | | | | | 0.49 | 64.9 | 68.8 | 0.43 | 1.48 |
| | OpenCV | 841.9 | | 974.5 | 588.8 | -0.250 | 0.104 | -0.755 | -0.297 | 0.001 | | | | 0.08 | 61.8 | 68.2 | 0.07 | 1.17 |
| | DLSC | 841.4 | | 975.0 | 589.9 | -0.253 | 0.106 | -0.980 | -0.032 | 0.007 | | | | 0.12 | 61.9 | 65.5 | 0.04 | 0.13 |
| GoPro | Ini. | 695.2 | | 960.0 | 600.0 | -0.146 | 0.020 | | | | | | | 1.42 | 49.7 | 65.1 | | |
| | OpenCV | 840.2 | | 973.2 | 586.5 | -2.701 | 1.808 | -0.232 | 0.249 | 0.688 | -2.416 | 0.963 | 1.430 | 0.20 | 70.5 | 88.3 | | |
| | DLSC | 839.2 | | 974.1 | 590.6 | -0.249 | 0.095 | -1.108 | -0.078 | 0.008 | 0.018 | -0.012 | 0.043 | 0.17 | 61.2 | 82.8 | | |
| industrial # 1 | Ini. | 5247.5 | | 960.0 | 600.0 | | | | | | | | | 0.27 | 0 | 0 | | |
| | OpenCV | 5275.4 | | 894.9 | 690.6 | -1.106 | 4.835 | 3.121 | -2.779 | -9.919 | -10.999 | 2.716 | 9.095 | 0.23 | 0.61 | 0.86 | | |
| | DLSC | 5264.5 | | 960.4 | 601.4 | -0.095 | 1.811 | -5.868 | 2.094 | 1.881 | -0.001 | 0.015 | -0.593 | 0.27 | 0.79 | 1.00 | | |
| industrial # 2 | Ini. | 5247.5 | 5247.5 | 960.0 | 600.0 | | | | | | | | | 0.27 | 0 | 0 | | |
| | OpenCV | 5284.5 | 5280.7 | 882.9 | 682.9 | -13.654 | 4.757 | 2.458 | -3.427 | -10.253 | -13.593 | 2.611 | 9.769 | 0.23 | 0.61 | 0.92 | | |
| | DLSC | 5163.0 | 5104.6 | 944.5 | 591.8 | -0.036 | -0.737 | -5.320 | -1.735 | 28.640 | -0.001 | 0.036 | -2.537 | 0.74 | 0.72 | 0.89 | | |

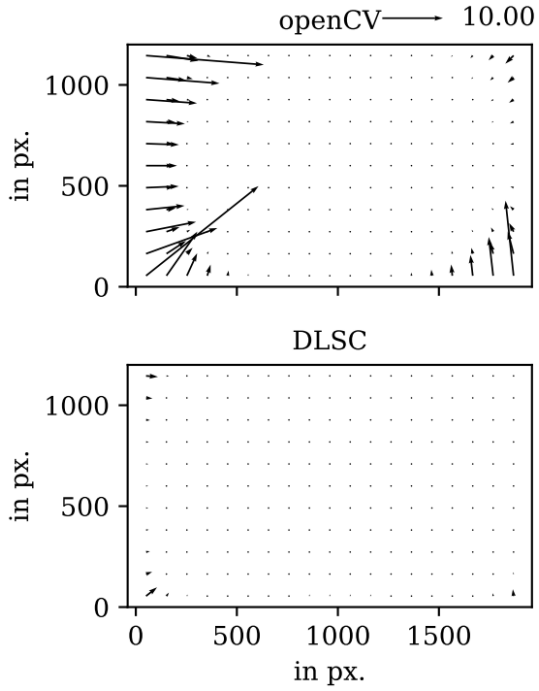


Figure 2. The displacement error generated by the OpenCV library and the proposed DLSC method in the artificial test case #3. The error is represented by arrows with a factor of 30. The scale is located in the upper right corner.

### 5.2 Distortion error

The overall distortion cannot be compared by looking at the parameters alone, since different parameters can lead to a similar distortion. For that reason, the distortion is calculated as the shift that each pixel undergoes. The shift of pixels is compared between calibration techniques and the shift that predefined parameters produce.

The error is given as the mean pixelwise relative absolute error (MRAE) and the median pixelwise relative absolute error (MdRAE) that the predefined parameters produce.

### 5.3 Results for artificial images

For the evaluation of the proposed method, three views of a plane are generated per test case. The texture of the plane is a photograph in order to produce rich features. Artificial images are created by inverse warping, which requires the inverse of the distortion model (Equations 7-13). It is calculated by an approximate iterative algorithm [2].

A set of artificial views of a calibration pattern is also created per test case. Camera parameters are recovered by utilizing the OpenCV library. A second set of such images is created to calculate the reprojection error. All images have a resolution of 1920x1200 pixels. Each pattern consists of 481 points. The first and second set each consist of 14 training and 3 test images, the third set consists of 10 training and 3 test images. The following results that are shown in Table 2 were produced:

- In the artificial test set #1 the principal point is shifted 100 pixels in horizontal direction and the pictures are radially distorted. The results show that the DLSC method is capable of reconstructing the camera intrinsic and distortion parameters, although the OpenCV library reaches a higher precision.
- Test set #2 shows an example of an aspect ratio of the pixels which is not equal to 1. Again, the results show that the real parameters are reconstructed.
- The artificial camera of test set #3 resembles the real GoPro camera of the next test case. Here, a higher degree of radial distortion and tangential distortion is implemented. The results show a high accuracy of the

reconstructed parameters. The MdRAE and MRAE produced by the DLSC method are significantly smaller than those produced by the OpenCV library. The reason is the higher accuracy of the distortion model. Figure 2 compares the displacement error that the DLSC method and the OpenCV library produce. Especially in the outer parts of the image, the DLSC method can use every pixel to estimate the distortion model in contrast to methods that rely on feature points. This is very important due to the poor extrapolation capability of the distortion model.

### 5.4 Results for real camera images

To test the proposed method on real camera images, a GoPro action camera with wide-angle lens and strong distortion (see Figure 1) and an industrial camera with telephoto lens and low distortion are used. A calibration target of the same size and dimension as in the artificial experiments is produced. Two sets of images are created for each camera: One for use with the OpenCV library and one for calculating the reprojection error. In both cases the full distortion model with all parameters is used.

For the GoPro camera, 52 images are used to find the parameters with the OpenCV library, and 51 different images are saved for calculating the reprojection error. The resolution is down sampled and cropped to 1920 x 1200 pixels. The results are shown in Table 2. The amount of predicted distortion differs between the OpenCV and the DLSC calibration, but since the real distortion is unknown, only the reprojection error can be used for comparison. It shows that the DLSC method slightly outperforms the method implemented in the OpenCV library. If one compares the result with the artificial test case #3, this is probably due to the better distortion parameters.

For the industrial camera, 106 images are used to find the parameters with the OpenCV library, and 36 are saved for the reprojection error. The resolution is 1920 x 1200 pixels. Table 2 shows two test cases: In the first the aspect ratio is fixed at 1. The reprojection errors of both methods are comparable. The principal point seems to be too far away from the center for the OpenCV library, while the DLSC method places it more centrally. Due to the low distortion, the DLSC method cannot outperform the initialization method with respect to the reprojection error. However, the overall amount of distortion is similar between the OpenCV and the DLSC result.

In the second test case the aspect ratio is not fixed. This experiment shows the limit of the DLSC method with only three input images. Due to the low distortion, the center of distortion and thus the principal point is more difficult to find. As a result, the other parameters of the calibration matrix cannot be accurately estimated. The reprojection error increases. Normally, when a digital camera is calibrated, one can assume that the aspect ratio is equal to 1 and this problem can be avoided.

## 6. CONCLUSION

A new method for camera self-calibration has been presented. The results show that it is capable of reconstructing intrinsic camera parameters as well as distortion parameters to a high precision. The effort is reduced to a minimum, since only three views of a flat surface are required. This is in contrast to established methods, where a calibration pattern has to be fabricated and many pictures thereof have to be taken. Compared to previous self-calibration techniques, it is the only method that is capable of fitting a complex distortion model with a high number of parameters.

Although not shown here, the method can easily be adapted to calibrate stereo cameras. If the focal length has been calibrated beforehand and the distance between cameras is known, the distortion parameters of both individual cameras together with the extrinsic parameters can be estimated from a single stereoscopic picture of a flat surface. This is clearly more practical than methods such as described in [22], where structured light is used for this purpose.

The method can also be easily extended to include more than three views. The limiting factor is the GPU memory. A future work could focus on reducing the amount of memory needed by reducing the size of the feature extraction network.

## REFERENCES


[1] J.-Y. Bouguet. Camera Calibration Toolbox for Matlab, 2015.

[2] G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

[3] D. C. Brown. Close-range camera calibration. Photogramm. Eng, 37(8):855–866, 1971.

[4] H. Brunken and C. Gühmann. Incorporating Plane-Sweep in Convolutional Neural Network Stereo Imaging for Road Surface Reconstruction. In Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, pages 784–791. SciTePress, 2019.

[5] A. Cefalu, N. Haala, and D. Fritsch. Structureless bundle adjustment with self-calibration using accumulated constraints. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, III-3:3–9, June 2016.

[6] J.-R. Chang and Y.-S. Chen. Pyramid Stereo Matching Network. arXiv preprint arXiv:1803.08669, page 9, 2018.

[7] C. S. Fraser. Automatic Camera Calibration in Close Range Photogrammetry. Photogrammetric Engineering & Remote Sensing, 79(4):381–388, Apr. 2013.

[8] Y. Furukawa and J. Ponce. Accurate Camera Calibration from Multi-View Stereo and Bundle Adjustment. International Journal of Computer Vision, 84(3):257–268, Sept. 2009.

[9] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. Cambridge University Press, Cambridge, UK; New York, 2003.

[10] D. Herrera, C. J. Kannala, and J. Heikkila. Forget the checkerboard: Practical self-calibration using a planar scene. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1–9, Lake Placid, NY, USA, Mar. 2016. IEEE.

[11] K. Ikeuchi, editor. Computer Vision - A Reference Guide. Springer US, Boston, MA, 2014.

[12] M. Jaderberg, K. Simonyan, A. Zisserman, and others. Spatial transformer networks. In Advances in neural information processing systems, pages 2017–2025, 2015.

[13] G. Korn and T. Korn. Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review. Dover Civil and Mechanical Engineering. Dover Publications, 2013.

[14] H. Li and R. Hartley. Plane-Based Calibration and Auto-calibration of a Fish-Eye Camera. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, P. J. Narayanan, S. K. Nayar, and H.-Y. Shum, editors, Computer Vision – ACCV 2006, volume 3851, pages 21–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[15] E. Malis and M. Vargas. Deeper understanding of the homography decomposition for vision-based control. PhD Thesis, INRIA, 2007.

[16] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual Modeling with a Hand-Held Camera. International Journal of Computer Vision, 59(3):207–232, Sept. 2004.

[17] R. Szeliski. Computer Vision. Texts in Computer Science. Springer London, London, 2011.

[18] J.-P. Tardif, P. Sturm, and S. Roy. Plane-based self-calibration of radial distortion. In 2007 IEEE 11th International Conference on Computer Vision, pages 1–8, Rio de Janeiro, Brazil, 2007. IEEE.

[19] The MathWorks. MATLAB Computer Vision Toolbox, 2019.

[20] B. Triggs. Autocalibration from planar scenes. In G. Goos, J. Hartmanis, J. van Leeuwen, H. Burkhardt, and B. Neumann, editors, Computer Vision — ECCV'98, volume 1406, pages 89–105. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[21] Z. Zhang. A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330–1334, Nov. 2000.

[22] H. Zhao, Z. Wang, H. Jiang, Y. Xu, and C. Dong. Calibration for stereo vision system based on phase matching and bundle adjustment algorithm. Optics and Lasers in Engineering, 68:203–213, May 2015.