Fully Automatic Calibration of Multiple Cameras to a Single World Coordinate System with Bundle Adjustment

Markus Frey
Master of Science, Informatik
Matriculation Number: 3120812

Lab Course Project
Prof. Dr.-Ing. Thomas Brox
Computer Vision Group
Albert-Ludwigs-Universität Freiburg

Abstract. Camera calibration is a central task in a lot of computer vision related problems. There are many different approaches, spanning from expensive calibration bodies to printed checkerboards. The goal of this work was to design a custom calibration body and algorithm providing a more intuitive and robust way of automatically calibrating a set of cameras in a scene. Compared to the classical and tedious procedure using multiple shots of a single checkerboard pattern this gives a combined way of getting internal and external camera parameters with just a simple, one-off setup. To optimize the bundle adjustment step a variety of reprojection error measures are proposed and compared on computer generated test scenes as well as real world examples. As a result this gives a general impression of how closely we can calibrate a multi-camera setup using the proposed calibration body.

Table of Contents

1 Overview

The here proposed methods seek to find a robust way of calibrating multiple cameras to the same world coordinate system using bundle adjustment. Three calibration bodies as seen in the example images in figure 1 are used that allow for automatic camera pose estimation as well as calculation of the internal parameters. For the algorithms to perform best each camera should see as many differently oriented faces of the calibration objects as possible. There must be no camera seeing no face in common with any other camera.

Fig. 1: The three calibration bodies, each consisting of 18 checkerboards in different orientations. The checkerboards facing down can be used in setups where the bodies are placed on a glass pane with cameras filming from below.

The first step is the checkerboard detection. Chapter 2 gives an overview of the used method described by Geiger et al [1]. To get initial values for the internal parameters the differently orientated checkerboards are treated as planar calibration objects. All checkerboards are color coded in order to uniquely find correspondences among the cameras. This is key to calculate egomotion between two viewing positions and therefore absolute external camera parameters. Chapter 3 holds everything needed to get an initial representation of the scene. However, the main focus of this paper lies in finding different bundle adjustment variations that better suite our calibration bodies and therefore outperform the classic approach in most cases. Section 4 explains these ideas.

2 Checkerboard Detection

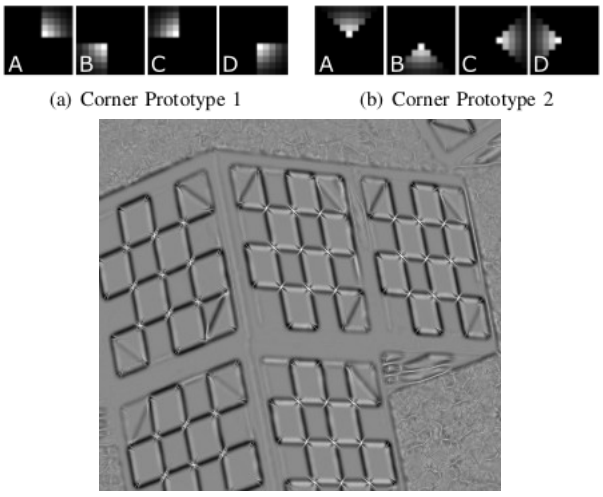
The proposed method uses the checkerboard detector by Geiger et al [1]. The two main tasks are finding corner points and assembling them to a grid which is

assigning them their respective x and y label on the checkerboard. The following chapters give a brief summary. More detailed information can be found in the corresponding paper.

2.1 Finding Corners

The input image is first convolved with different types of corner kernels (see figure 2). To get the corner likelihood c at a given pixel position one has to look at all its kernel responses f_X^i with $X \in \{A, B, C, D\}$ being the kernel and $i \in \{1, 2\}$ being the prototype.

$$
\begin{aligned}
c &= max(s_1^1, s_2^1, s_1^2, s_2^2) \\
s_1^i &= min(min(f_A^i, f_B^i) - \mu, \mu - min(f_C^i, f_D^i)) \\
s_2^i &= min(\mu - min(f_A^i, f_B^i), min(f_C^i, f_D^i) - \mu) \\
\mu &= 0.25(f_A^i + f_B^i + f_C^i + f_D^i)
\end{aligned}
$$



(a) Corner Prototype 1 (b) Corner Prototype 2

Fig. 2: A checkerboard section convolved using different corner detection kernels. The kernel responses yield a corner likelihood c as seen in the image.

This is done for every pixel of the input image. To get the final corner candidates non-maximum suppression is performed [5]. In order to further sort out non-corners a 32 bin orientation histogram is generated from Sobel filter responses around a $n \times n$ neighborhood of a candidate. We calculate its two dominant modes α_1 and α_2 using mean shift [6]. The two edge orientations of a checkerboard corner can directly be inferred from them. We draw a patch of

what the neighborhood's gradient magnitudes would look like if there really was a checkerboard with found edge orientations at the given position. The normalized cross correlation between the encountered neighborhood and our predicted one is multiplied by the previous score c to give us a final likelihood. By thresholding we try to get rid of as many false positives as possible. See figure 3 for an example picture.

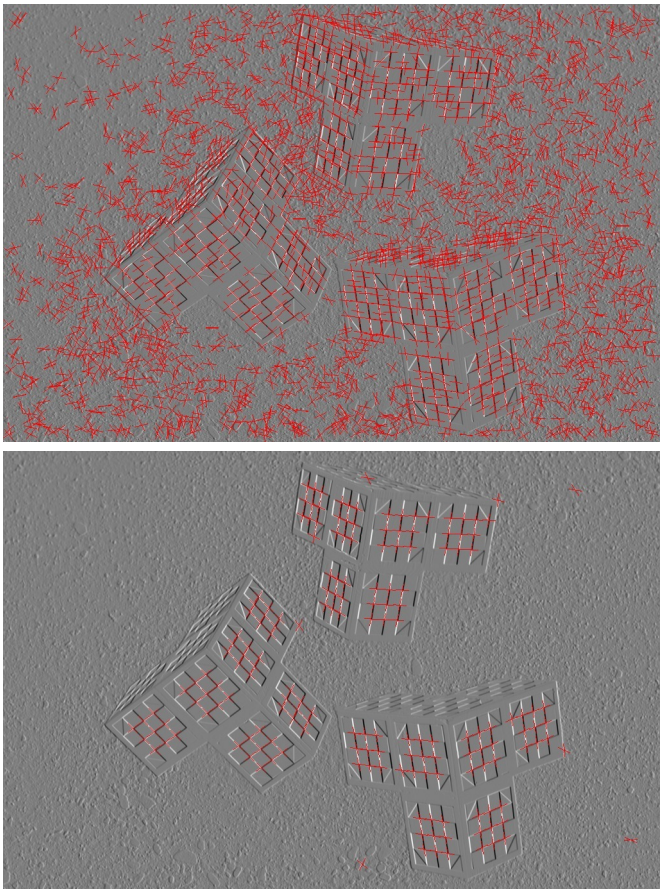


Fig. 3: A rather extreme example of a lot of possible checkerboard corner candidates due to a very structured background. The red lines indicate the two possible edge orientations resulting from the two dominant modes of the orientation histogram of sobel filter responses. The top image is before thresholding the calculated normalized cross correlation and the bottom one after.

To achieve sub-pixel accuracy of a corner position $\mathbf{p} \in \mathbb{R}^2$ we look at the gradients in the 11×11 neighborhood. Only pixels lying on an edge of the pattern yield a strong gradient. Their direction is orthogonal to the direction in

which this pixel is located relative to our detected corner position $\mathbf{p}$. This leads to the following minimization problem with $\mathbf{g}_n$ being the gradient vector of a pixel $\mathbf{n}$ in the neighborhood N around $\mathbf{p}$:

$$\mathbf{p} = \arg\min_{\mathbf{p}'} \sum_{\mathbf{n} \in N(\mathbf{p}')} (\mathbf{g}_n^T(\mathbf{n} - \mathbf{p}'))^2$$

The solution can directly be calculated as follows [1]:

$$\mathbf{p} = (\sum_{\mathbf{n} \in N} \mathbf{g}_n \mathbf{g}_n^T)^{-1} \sum_{\mathbf{n} \in N} (\mathbf{g}_n \mathbf{g}_n^T)\mathbf{n}$$

2.2 Assembling the Grid

The algorithm from the previous chapter gives corner positions and their edge orientations. To assemble them a slightly different approach to Geiger's method is used. The markers of our calibration body form 3×3 grids. Let τ be the set of all row and column triples. Each assembled grid can then be given a score represented by the Energy E:

$$E = \max_{(i,j,k) \in \tau} \frac{||\mathbf{c}_i + \mathbf{c}_k - 2\mathbf{c}_j||_2}{||\mathbf{c}_i - \mathbf{c}_k||_2}$$

The numerator becomes zero if the grid is perfectly square. Because of the denominator bigger grids are allowed to be less square than smaller ones. The lower the score the better the assembled grid.

To compose the grid we randomly select a seed corner and try to find the corresponding grid neighbors:

1. **Find the four direct neighbors to the seed corner:** Search the set of all corners for candidates along the seed's edge directions. Found neighbors are thresholded using their deviation to the edge direction and their distance to the seed (candidates too far away get rejected). Choose the neighborhood corners that best fit the edge direction of the central checkerboard corner. Abort if at least one of the four neighbors can't be determined this way.
2. **Find the cornerstones of the grid:** The previous step assembled the central cross structure of the grid. To determine the missing four cornerstones we look at the respective two tips of the cross that form their direct neighbors. We search the set the same way we did the step before only this time a candidate must satisfy the deviation and distance threshold for both its neighbors. If all four cornerstones were found the 3×3 grid now is complete and each member is labeled according to its x and y position in the grid.
3. **Reading the color code:** Having the grid assembled we can determine the angle and distance from the cornerstones in which we need to sample the image to get the color codes. The angle results from the edge directions and the distance is approximately two thirds of the grid spacing.

4. **Rotate the assigned grid labels:** Assume two cameras see the same checkerboard. For the algorithms coming up it is necessary that the detections in their two images have the same rotation of labels. In practice this means that the corner assigned to e.g. grid position $(0, 0)$ should refer to the same physical corner on the calibration body for both camera frames. We use our color codes to rotate the grid assignments in a unique way. Each color is converted to HSV color space. We flip and rotate the grid labels until $(0, 0)$ is the black corner and the hue of $(2, 0)$ is smaller than $(0, 2)$.

Figure 4 visualizes these four steps. Repeat for every corner as the starting seed. In case we assembled overlapping checkerboards check for grids that share at least one corner and keep only the one with the best score. Figure 5 is an example of a fully labeled image.

start step 1 step 2 step 3 step 4

Fig. 4: The steps to assemble the checkerboard. The algorithm starts out on all checkerboard corner detections and a chosen seed corner (start). The first step then is to find the seed's direct neighbors along its edge directions. In order to determine the cornerstones (step 2) of the sought 3×3 grid we search the set for detections satisfying the edge directions of their two neighbors. If all of these steps finished succesfully the grid is complete and we detect the color coding (step 3) and label the detections in a unique rotation (step 4). If step 1 or 2 fails the chosen seed was not a grid's center and we abort. Repeat this procedure until every corner detection was once tested as a seed and collect all resulting grids.

3 Initial Camera Calibration and World Reconstruction

Before any bundle adjustment can be done we need to have an initial camera calibration and an initial point cloud representing the calibration bodies. Since the problem is nonlinear a good initialization is crucial. To estimate the internal camera parameters we use the detected checkerboards as planar calibration objects. For each camera we can calculate a relative position to a seen checkerboard. Starting with an arbitrary camera as reference all other cameras can be added to the same coordinate system by iteratively adding those that share a seen checkerboard. Pairs of cameras are used to triangulate newly exposed calibration body faces.

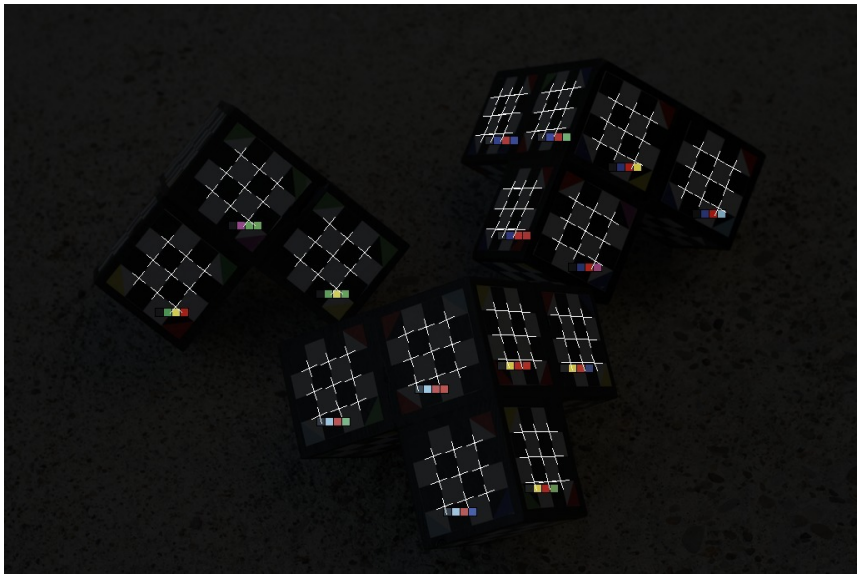


Fig. 5: Detected checkerboards with their respective color codings uniquely rotated. The white crosses are drawn along the estimated edge directions of the checkerboard corner.

3.1 Camera Parameters

The projection of a 3D point $\mathbf{X}$ in world coordinates to its 2D image coordinates $\mathbf{x}$ as seen in a camera's image is defined by the projection matrix P consisting of the internal camera matrix K and the external camera matrix M. Coordinates are given in homogeneous form.

$$\boldsymbol{x} = P\boldsymbol{X}$$

$$P = KM = \begin{pmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

M represents the camera's external parameters namely world position and rotation. It rotates the world such that the camera lies in the coordinate center. K holds the camera's internal parameters. Focal length α_x and α_y, principal point $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ and skew s.

To estimate K the detected checkerboards are used as planar calibration objects. Since they are all identical in shape every detection can be seen as the same calibration object in different orientations. The projection of such a planar object to the image can be described by a homography. Having multiple instances in different orientations then yields a system of equations which can ultimately

be decomposed to get to the internal camera matrix K. For more details see [2]. In the implementation the camera calibration functions of OpenCV version 2.4.9 were used. It is assumed that $\alpha_x = \alpha_y$ and $s = 0$.

3.2 Camera Poses and Triangulation

To generate an initial setup for the bundle adjustment a combination of relative and absolute camera pose estimations as well as point triangulations are necessary. Before briefly going through them a few words about the implementation. The program run in the experiments uses OpenGV as a library for these geometric problems. The point correspondences are represented as bearing vectors. A bearing vector is a 3D unit vector pointing to a seen 3D world point in camera coordinates. Let $\mathbf{x}_u$ be the operation to normalize a vector $\mathbf{x}$ by its length:

$$\mathbf{x}_u := \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|} & \mathbf{x} \neq 0 \\ 0 & \mathbf{x} = 0 \end{cases}$$

Conversion of an image point $\mathbf{p} = \begin{pmatrix} p_x & p_y \end{pmatrix}^T$ to its bearing vector $\mathbf{v} = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T$ given the camera's focal length f and principal point $\mathbf{c} = \begin{pmatrix} x_0 & y_0 \end{pmatrix}^T$ is then done as follows:

$$\mathbf{v} = \begin{pmatrix} p_x - x_0 \\ p_y - y_0 \\ f \end{pmatrix}_u$$

Relative Camera Pose Relative pose between two cameras can be extracted from the 3×3 essential matrix E. For homogeneous point correspondences $\mathbf{q}$ and $\mathbf{q}'$ and estimations of the internal camera matrices K and K' it is defined as follows:

$$\begin{aligned} \hat{\mathbf{q}} &= K^{-1}\mathbf{q} \\ \hat{\mathbf{q}}' &= K'^{-1}\mathbf{q}' \\ \hat{\mathbf{q}}'^{\mathbf{T}} E \hat{\mathbf{q}} &= 0 \end{aligned}$$

E holds the relative rotation R and relative translation t between the two cameras:

$$E = R[t]_x, \quad \text{with } [t]_x = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}$$

The challenging part is to get the essential matrix from a set of point correspondences. Once found R and t can be extracted using singular value decomposition on E. Different approaches have been made mostly named after the minimum number of required correspondences. We use the five point method proposed by Stewenius et al. [3].

Absolute Camera Pose Absolute camera position can be calculated using 2D-3D point correspondences given the camera's internal parameters. The idea is to find the 3D reference points' position in camera coordinates and then retrieving the camera pose by aligning the two point clouds. The here used method introduced by Lepetit et al [4] expresses the 3D reference points $\mathbf{p}_i$ using four control points $\mathbf{c}_{1,...,4}$. One control point is placed in the center of the references, the others should be placed so they form a basis aligned with the principal directions.

Points in world coordinates have the superscript ^{w} and in camera coordinates the superscript ^{c}. Every $\mathbf{p}_i$ is then defined by its weights $\alpha_{ij}, j \in \{1, ..., 4\}$:

$$\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w, \quad \text{with} \sum_{j=1}^{4} \alpha_{ij} = 1$$

Knowing the control points in camera coordinates $\mathbf{c}_j^c$ we can also write:

$$\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c$$

With the internal camera matrix K and the 2D projections $\mathbf{u}_{1,...,n}$ of the reference points $\mathbf{p}_{1,...,n}$ the projection becomes:

$$\forall i, \; w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = K \mathbf{p}_i^c = K \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c$$

With $\mathbf{u}_i = \begin{pmatrix} u_i \; v_i \end{pmatrix}^T$ and $\mathbf{c}_j^c = \begin{pmatrix} x_j^c \; y_j^c \; z_j^c \end{pmatrix}^T$ this reads:

$$\forall i, \; w_i \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{pmatrix} x_j^c \\ y_j^c \\ z_j^c \end{pmatrix}$$

From the last row it directly follows that $w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$ leaving two equations per reference point:

$$\sum_{j=1}^{4} \alpha_{ij} f x_j^c + \alpha_{ij}(x_0 - u_i) z_j^c \tag{1}$$

$$\sum_{j=1}^{4} \alpha_{ij} f y_j^c + \alpha_{ij}(y_0 - v_i) z_j^c \tag{2}$$

Collecting these for each reference point we get the equation:

$$M\mathbf{x} = 0$$

with $\mathbf{x} = \left[c_1^{cT}, c_2^{cT}, c_3^{cT}, c_4^{cT} \right]^T$ and M being the $2n \times 12$ matrix obtained from the equations 1 and 2 for each reference point. We see that the problem has been

reduced to finding the 12 unknowns which are the four control points in camera coordinates. Now $\mathbf{x}$ can be constructed by a weighted sum of the eigenvectors of $M^T M$ which is a matrix of size 12×12. The weights are calculated by solving a constant number of quadratic equations. For further details see [4].

Triangulation Given point correspondences $\mathbf{x}$ and $\mathbf{x'}$ and the full projection matrices P and P' constructed from the internal camera matrices and the cameras' absolute world positions it holds for homogeneous coordinates:

$$\mathbf{x} \times (P\mathbf{X}) = 0$$
$$\mathbf{x'} \times (P'\mathbf{X}) = 0$$

With $\mathbf{x} = \begin{pmatrix} x & y & z \end{pmatrix}^T$ and $P = \begin{bmatrix} p1, p2, p3 \end{bmatrix}$ this yields three equations:

$$x(p_3^T\mathbf{X}) - (p_1^T\mathbf{X}) = 0$$
$$y(p_3^T\mathbf{X}) - (p_2^T\mathbf{X}) = 0$$
$$x(p_2^T\mathbf{X}) - y(p_1^T\mathbf{X}) = 0$$

Only two of them are linearly independent and we get the linear equation system:

$$\begin{pmatrix} x\mathbf{p}_3^T - \mathbf{p}_1^T \\ y\mathbf{p}_3^T - \mathbf{p}_2^T \\ x'\mathbf{p'}_3^T - \mathbf{p'}_1^T \\ y'\mathbf{p'}_3^T - \mathbf{p'}_2^T \end{pmatrix} \mathbf{X} = 0$$

Solving for $\mathbf{X}$ gives the constructed point in world coordinates.

3.3 Reconstructing the World

To position multiple cameras and triangulated points in a common world coordinate system we make use of the methods from the previous chapter. The color codes of the markers uniquely identify the different calibration body faces. To check whether two detected checkerboards belong to the same physical face of the calibration body we compare the hue values in order to be robust against brightness and contrast changes. If the hue differences of all of the four colors fall below a given threshold we assume the same checkerboard is observed and the corner positions can be used as point correspondences to the pose estimations and triangulations.

1. **Initial camera pair:** We start off by selecting the pair of cameras which see the most checkerboards in common. Their point correspondences are used to calculate the relative pose between them. Setting the coordinate system of one camera as the world coordinate system the relative pose of the other camera directly leads to its absolute pose in this newly defined world frame. Again the point correspondences are used to now triangulate the seen checkerboard corners resulting in points that are also in common world coordinates. The two cameras and checkerboard points start off the reconstructed world.

2. **Iteratively adding the remaining cameras:** Loop over the remaining cameras and check against every already positioned camera counting their commonly seen checkerboards. Choose the pair (c, c') seeing the most checkerboards in common with c being a known camera and c' a yet to be positioned camera. Like in the previous step first calculate the relative position between c and c' using the correspondences and then triangulate the checkerboard corners. Make sure to calculate position and rotation of c' in camera coordinates of c (not the other way around). This positions c' and the checkerboard points as seen from c. To get their world coordinates transform them using the external parameters of camera c. Add c' and all newly triangulated checkerboards to the reconstructed world. Repeat until all cameras are placed.

These two steps result in an initial world reconstruction with all cameras and triangulated checkerboards ready to be bundle adjusted.

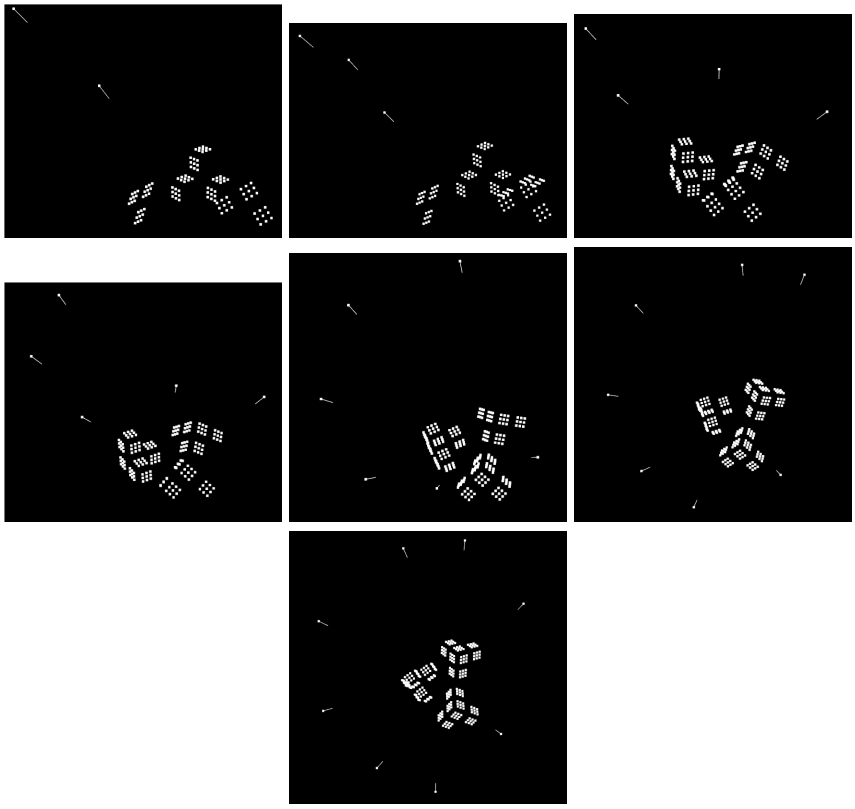


Fig. 6: The world reconstructed step by step. We start off with two cameras and iteratively add the remaining ones while triangulating newly discovered faces of the calibration object.

4 Bundle Adjustment

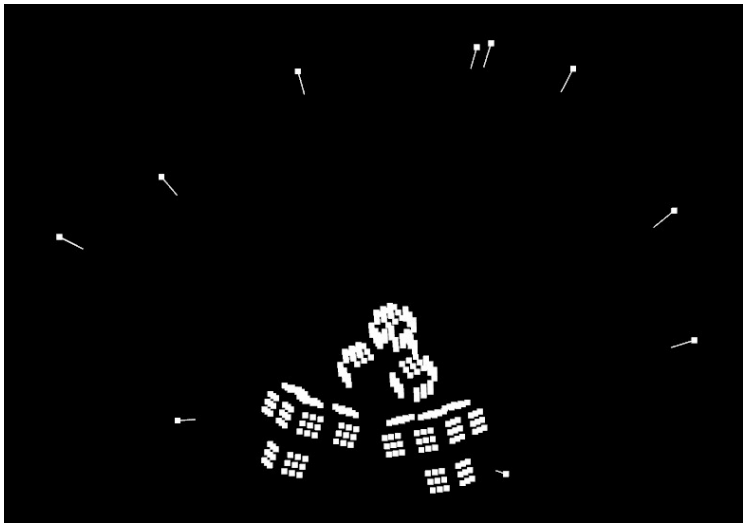


Fig. 7: The input to bundle adjustment: An initual guess for camera positions and world point reconstructions.

Given a set of cameras and world objects bundle adjustment minimizes the reprojection error of world points projected to the camera images by optimizing the cameras' internal and external parameters as well as world point positions. All calculations are done in homogeneous coordinates.

Let $\mathbf{X}_j$ be the j-th reconstructed point in world coordinates:

$$\mathbf{X}_j = \begin{pmatrix} x_j \\ y_j \\ z_j \\ 1 \end{pmatrix}$$

With $f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i$ being the i-th camera's internal and external parameters as seen in chapter 3.1 the projection of a world point $\mathbf{X}_j$ to the camera's image can be defined as function π:

$$\pi(f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{X}_j) = \delta\left(\begin{pmatrix} f^i & 0 & x_0^i & 0 \\ 0 & f^i & y_0^i & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{bmatrix} \mathbf{R}^i & \mathbf{t}^i \\ 0 & 1 \end{bmatrix} \mathbf{X}_j, \quad \kappa_1^i, \kappa_2^i \right)$$

Here $\delta(\mathbf{x}, \kappa_1^i, \kappa_2^i)$ is the camera's radial distortion according to its distortion coefficients κ_1^i and κ_2^i.

Let $r(\mathbf{x}, x_0^i, y_0^i)$ be the distance of $\mathbf{x} = (x, y, w)^T$ to the camera's principal point $(x_0^i, y_0^i)^T$:

$$r(\mathbf{x}, x_0^i, y_0^i) = \sqrt{(x/w - x_0^i)^2 + (y/w - y_0^i)^2)}$$

The radial distortion then is defined as:

$$\delta(\mathbf{x}, \kappa_1^i, \kappa_2^i) = \begin{pmatrix} x/w + (x/w - x_0^i)(\kappa_1 r(\mathbf{x}, x_0^i, y_0^i) + \kappa_2 r(\mathbf{x}, x_0^i, y_0^i)^2) \\ y/w + (y/w - y_0^i)(\kappa_1 r(\mathbf{x}, x_0^i, y_0^i) + \kappa_2 r(\mathbf{x}, x_0^i, y_0^i)^2) \end{pmatrix}$$

The final minimization term is defined as the sum of the squared euclidean distances of all j world points reprojected to their corresponding detections in the images of all i cameras. Let $\mathbf{X}_j$ be the reconstructed world point and $\mathbf{x}_j^i$ be the image position of its detection in the i-th camera's image (assuming that the camera sees it):

$$\min_{f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{X}_j} \sum_{i,j} d^2(\pi(f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{X}_j), \mathbf{x}_j^i)$$

In the experiments this is called the simple bundle adjustment since it's the bare minimum and holds no further constraints to world point positions or camera parameters. Everything can be altered independently.

4.1 Principal point prior

Since we're expecting the principal point $(x_0^i, y_0^i)^T$ to be roughly in the image center the first bundle adjustment improvement is to add the following residual for every camera i with W and H being the image width and height and ρ the weight for the prior:

$$\rho \sum_i \left(\sqrt{(x_0^i - W/2)^2 + (y_0^i - H/2)^2)} \right)^2 = \quad \rho \sum_i d^2 \left(\begin{pmatrix} x_0^i \\ y_o^i \end{pmatrix}, \begin{pmatrix} W/2 \\ H/2 \end{pmatrix} \right)$$

Our minimization term now reads:

$$\min_{f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{X}_j} \sum_{i,j} d^2(\pi, \mathbf{x}_j^i) + \rho \sum_i d^2 \left(\begin{pmatrix} x_0^i \\ y_o^i \end{pmatrix}, \begin{pmatrix} W/2 \\ H/2 \end{pmatrix} \right)$$

The experiments show that this improvement alone already results in much better calibration since bad local minima with unrealistic principal points are avoided. This minimization term is added to all proposed methods.

4.2 Energy of the checkerboard detection

In the checkerboard detection we achieved sub-pixel accuracy for the detections $\mathbf{p}$ by minimizing the following term (see chapter 2.1):

$$\mathbf{p} = \arg\min_{\mathbf{p'}} \sum_{\mathbf{n} \in N(\mathbf{p'})} (\mathbf{g}_n^T (\mathbf{n} - \mathbf{p'}))^2 \tag{3}$$

Interpreting this as an energy function we can use it as our residual in the minimization problem instead of the euclidean distance. To make this feasable in an optimization algorithm we need to break down the sum.

Let $N(\mathbf{p})$ be the neighborhood around the detected checkerboard corner in the camera's image. With $\mathbf{g}_n^T$ again being the gradients the new distance measure for the corresponding reprojected point $\mathbf{x}$ then becomes:

$$d(\mathbf{x}) = \sum_{\mathbf{n} \in N(\mathbf{p})} (\mathbf{g}_n^T(\mathbf{n} - \mathbf{x}))^2$$

Instead of using absolute reprojected pixel positions $\mathbf{x}$ let's consider positions $\hat{\mathbf{x}}$ relative to the corresponding detection $\mathbf{p}$:

$$\hat{\mathbf{x}} = \mathbf{x} - \mathbf{p}$$
$$d^*(\hat{\mathbf{x}}) = d(\hat{\mathbf{x}} + \mathbf{p})$$

Now d^* is a two dimensional quadratic function with $\hat{\mathbf{x}} = 0$ being its unique minimum (because of equation 3). These types of functions can be written as:

$$d^*(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \mathbf{A} \hat{\mathbf{x}} + c, \text{ with } \mathbf{A} \in \mathbb{R}^{2\times 2}$$

$\mathbf{A}$ is symmetric and positive definite. Therefore:

$$\mathbf{A} = \begin{pmatrix} a_1 & a_0 \\ a_0 & a_2 \end{pmatrix}$$

To determine $\mathbf{A}$ we look at some special evaluations of function d^*:

$$\begin{aligned}
d^*(\begin{pmatrix} 0 \\ 0 \end{pmatrix}) &= c \\
d^*(\begin{pmatrix} 1 \\ 0 \end{pmatrix}) &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} a_1 & a_0 \\ a_0 & a_2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c = a_1 + c \\
d^*(\begin{pmatrix} 0 \\ 1 \end{pmatrix}) &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}^T \begin{pmatrix} a_1 & a_0 \\ a_0 & a_2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} + c = a_2 + c \\
d^*(\begin{pmatrix} 1 \\ 1 \end{pmatrix}) &= \begin{pmatrix} 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} a_1 & a_0 \\ a_0 & a_2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + c = 2a_0 + a_1 + a_2 + c \\
d^*(\begin{pmatrix} 1 \\ -1 \end{pmatrix}) &= \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} a_1 & a_0 \\ a_0 & a_2 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} + c = -2a_0 + a_1 + a_2 + c
\end{aligned}$$

$$\begin{aligned}
\Rightarrow c &= d^*(\begin{pmatrix} 0 \\ 0 \end{pmatrix}) \\
a_1 &= d^*(\begin{pmatrix} 1 \\ 0 \end{pmatrix}) - d^*(\begin{pmatrix} 0 \\ 0 \end{pmatrix}) \\
a_2 &= d^*(\begin{pmatrix} 0 \\ 1 \end{pmatrix}) - d^*(\begin{pmatrix} 0 \\ 0 \end{pmatrix}) \\
a_0 &= \frac{1}{4}(d^*(\begin{pmatrix} 1 \\ 1 \end{pmatrix}) - d^*(\begin{pmatrix} 1 \\ -1 \end{pmatrix}))
\end{aligned}$$

Now that we know how to calculate $\mathbf{A}$ we can simplify the calculation of $d(x)$:

$$d(\mathbf{x}) = d(\mathbf{x} - \mathbf{p} + \mathbf{p}) = d(\hat{\mathbf{x}} + \mathbf{p}) = d^*(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \mathbf{A} \hat{\mathbf{x}} + c$$

Since c is a constant we can ignore it in the new minimization equation:

$$\min_{f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{X}_j} \sum_{i,j} (\pi - \mathbf{x}_j^i)^T A (\pi - \mathbf{x}_j^i)$$

The simple bundle adjustment yields linear costs for deviations no matter in which direction (the squared euclidean distance). Now this new equation weighs them differently according to how well the reprojection fits to the structure of the checkerboard.

4.3 Bivariate gaussian distribution

Especially with noisy input images we have to take into account that detections aren't always spot on. Looking at figure 8 we assume that the detected checkerboard corner (marked in red) is more accurate in its position along the wider edge angle (here roughly along the image's y-axis) than it is along the sharper one (the images x-axis). To account for this observation we construct a bivariate gaussian distribution (marked in blue) with the two main axes along the angle bisections of the edge directions and variances proportional to the angle between the edge vectors. It is then used as the covariance matrix in a mahalanobis distance measure.

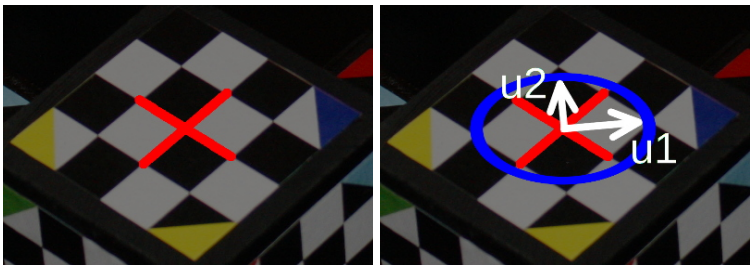


Fig. 8: Constructing a bivariate gaussian distribution along the detected checkerboard edges.

Writing the sought distribution $\mathbf{\Sigma}$ in terms of its eigendecomposition we get:

$$\mathbf{\Sigma} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} u_1 & u_2 \end{pmatrix}^T$$

The two eigenvectors u_1 and u_2 correspond to the two main axes of the distribution and the eigenvalues λ_1 and λ_2 represent the variances along them. We set

u_1 to be the bisection of the smaller angle α and u_2 the bisection of the bigger angle between the checkerboard edges:

$$\mathbf{\Sigma} = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} 1/sin(\alpha) & 0 \\ 0 & sin(\alpha) \end{pmatrix} \begin{pmatrix} u_1 & u_2 \end{pmatrix}^T$$

The minimization equation becomes the mahalanobis distance using $\mathbf{\Sigma}$ as the covariance:

$$\min_{f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{X}_j} \sum_{i,j} (\pi - \mathbf{x}_j^i)^T \mathbf{\Sigma}^{-1} (\pi - \mathbf{x}_j^i)$$

Note that if the checkerboard perfectly faces the camera ($\alpha = 90°$) $\mathbf{\Sigma}$ becomes the identity and the equation breaks down to the squared euclidean distance just like in the simple bundle adjustment.

4.4 Parameterized Bundle Adjustment

Our calibration bodies consist of checkerboards formed by a grid of three by three checkerboard corners. The detection algorithm from chapter 2.2 assembled these grids for us. Instead of moving single world points independently the optimization algorithm now alters translation and rotation of a grid as a whole. The basic idea is to describe a checkerboard grid by rotating and translating a grid prototype. We use the point cloud fitting algorithm by Arun et al [7] to get these transformations.

For each world point $\mathbf{X}_j$ let $col(\mathbf{X}_j) \in \{0, 1, 2\}$ and $row(\mathbf{X}_j) \in \{0, 1, 2\}$ be its column and row position inside the assembled checkerboard grid. The function $l(\mathbf{X}_j)$ then maps each point to the following 3D prototype position in homogeneous coordinates. This basically is the label the detection got assigned in chapter 2.2 (see figure 4, step 4):

$$l(\mathbf{X}_j) = \begin{pmatrix} col(\mathbf{X}_j) \\ row(\mathbf{X}_j) \\ 0 \\ 1 \end{pmatrix}$$

For each checkerboard cloud $G = \{\mathbf{X}^1, ..., \mathbf{X}^9\}$ construct its prototype cloud $G_p = \{l(\mathbf{X}^1), ..., l(\mathbf{X}^9)\}$ and calculate the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between them [7]. Working with homogeneous coordinates this gives us a rigid body transformation $\mathbf{C} = \begin{bmatrix} \mathbf{R}, \mathbf{t} \end{bmatrix}$ for every board.

Let $\mathbf{C}_0, .., \mathbf{C}_n$ be these calculated poses for all n checkerboards and $c(\mathbf{X}_j) \in \{0, ..., n\}$ the mapping assigning a world point $\mathbf{X}_j$ to the checkerboard it belongs to.

Starting with the simple bundle adjustment equation we substitute $\mathbf{X}_j$ with $\mathbf{C}_{c(\mathbf{X}_j)} l(\mathbf{X}_j)$ and the minimization problem becomes:

$$\min_{f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{C}_{c(\mathbf{X}_j)}} \sum_{i,j} d^2 \left(\pi\Big(f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{C}_{c(\mathbf{X}_j)} l(\mathbf{X}_j)\Big),\ \mathbf{x}_j^i \right)$$

To put things simple the optimization parameter $\mathbf{X}_j$ from the classic bundle adjustment equation now became $\mathbf{C}_{c(\mathbf{X}_j)}$ meaning we no longer try to optimize each single world point but the position and orientation of each three by three checkerboard grid as a whole.

4.5 Weighted Bundle Adjustment

Let us consider checkerboards facing a camera at a rather high angle as seen in figure 9. We assume that our corner detection gets less accurate as this angle

Fig. 9: A steep detected checkerboard.

gets bigger. The minimization equation should therefore weigh deviations less for boards not directly facing the camera. Each world point $\mathbf{X}_j$ gets assigned a weight by the function w depending on the angle the checkerboard it belongs to has to the currently considered camera orientation $\mathbf{R}^i$.

With $\cdot$ being the dot product and $\mathbf{C}_{c(\mathbf{X}_j)}$ being the checkerboard's pose $\mathbf{X}_j$ belongs to (see chapter 4.4) w is defined as:

$$w(\mathbf{R}^i, \mathbf{X}_j) = \mathbf{R}^i \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \mathbf{C}_{c(\mathbf{X}_j)} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

This simply calculates the cosine between the checkerboard's normal and the camera's viewing direction. It is one for perfectly facing markers and approaches zero as they turn away from the camera. The minimization term becomes:

$$\min_{f^i, x_0^i, y_0^i, \mathbf{R}^i, \mathbf{t}^i, \kappa_1^i, \kappa_2^i, \mathbf{X}_j} \sum_{i,j} w^2(\mathbf{R}^i, \mathbf{X}_j)\ d^2(\pi, \mathbf{x}_j^i)$$

Reprojections are allowed to differ more from the detections if the checkerboard they belong to is seen tilted. This gives us a mixed set of grids per camera where some must be more accurately reconstructed and some are given more slack.

5 Results

To solve the minimization equations the Ceres Solver library version 1.8.0 was used in the program. Besides testing on real camera images the algorithm also was run on a rendered set of images where the camera matrices were known in order to optimize the parameters and get a sense of accuracy. The figures on the last few pages show mean values and standard deviations for internal camera parameters in various test scenarios. The following abbreviations are used in the graphs:

- **start:** The input to the bundle adjustment. This is the starting point for the optimization.
- **simple:** Simple, classic bundle adjustment without any improvements (see the beginning of chapter 4).
- **simple with pp:** Simple bundle adjustment but with added principal point prior (see chapter 4.1). All the other upcoming methods have this prior as well.
- **energy:** Checkerboard corner energy as used in the detector as distance measure (see chapter 4.2).
- **covariance:** Bivariate gaussian distribution according to the checkerboard's orientation as covariance for a mahalanobis distance measure (see chapter 4.3).
- **parameterized:** Optimize a whole checkerboard pattern instead of single corner detections (see chapter 4.4).
- **weighted:** Weight the squared euclidean distance of the reprojection according to the checkerboard's orientation (chapter 4.5).

The first experiment (figure 10) is a set of rendered images. We know all the internal and external camera parameters and can evaluate the different methods based on that. The only downside to this is that there is generally no noise or distortion (no lenses, sensor noise or lighting difficulties) interfering with the checkerboard corner detections so all the benefits of the here proposed methods don't give as much of a performance boost as they do in real scenes. The input is so good that the simple approach already performs quite well.

This changes for the next set of images (figure 11). Here a Canon 450D with a 50mm f1.6 fixed focal length lense was used. Because of that we would expect low variance in said focal length but especially for low apertures and varying distances to the objects in focus the ground truth actually cannot be zero variance. Real images are prone to slightly off corner detections and thus our proposed methods clearly outperform the classical approach.

To test the bundle adjustment variations for cameras with differing focal lengths figure 12 shows a set of images again taken with a Canon 450D but this time a 18-55mm f3.5-5.6 lense was used. The first four images were shot with less zoom than the last four. Although images five to eight result in roughly the same focal length as the experiment before there is much higher variance. *Parameterized* seems to be the exception but taking a closer look shows that it simply eludes focal variation by yielding higher p_x variance. This could be due

to dim lighting and therefore worse corner detections. But again one can't really tell how much variance really is in the ground truth.

Figure 13 shows the results for an image set taken with the rather cheap camera module of the Nexus 5 smartphone. There's no hardware zoom so a fixed focal length is to be expected except for focussing due to different distances to the objects. Similar to the previous experiment *parameterized* shows less variance in focal length but higher variance in the principal point. With the weight for the principal point prior from chapter 4.1 one can adjust this trade off between centering the principal point and varying the other parameters although it is not clear which side to prefer.

The cheap smartphone camera was tested in better lighting in the next experiment (figure 14). The results show less variance in the focal length but slightly higher variance in the principal point. The overall higher focal mean could very well be true since the aperture changes from the dim scene to this well lit environment.

Our reprojection function π also models radial distortion with distortion coefficients κ_1 and κ_2. All of the implemented methods optimize this as well. In order to test how well these estimates are we took the rendered images and radially distorted them. Figure 15 and 16 show that the bigger the coefficients get the worse the general optimization becomes.

One final evaluation we can only really take with rendered images is the calculation of the cameras' orientations and positions. A little over 100 image sets were generated. For each of them the calibration bodies were randomly positioned and rotated while all cameras were kept fixed. This way we calculated the mean deviations and variances of focal lengths, principal points, camera positions and camera rotations over all of these sets (see figure 17). Again the results have to be taken with a grain of salt since these are noiseless and undistorted rendered images but by far the most accurate method seems to be the parameterized variant of chapter 4.4. Since we can't really do this type of experiment on real images because we simply can't accurately measure the positions of the optical centers we have to rely on this conclusion. Real images generally suffer from more corner detection noise.

6 Summary and Outlook

The idea behind the proposed methods generally is to loosen the constraints and provide varying error distributions across the image for example depending on the checkerboard's orientation or basically trying to deal with slightly off detections. This was done exploiting certain properties of the calibration body. For rendered test images the checkerboard detection is already so robust that this is not necessarily needed and simple bundle adjustment already performs well. But that means evaluating the different approaches is not that easy. In scenarios where we know the ground truth the different methods don't really play out their full potential and for real camera images we have no way of knowing how close we are to the real values because of depth of field, hardware production

errors or camera firmware doing various post processing like bad radial distortion correction. We can only say that we are in the right ballpark but especially for real cameras it is not clear whether low variance in focal length really is the better result since even when keeping the zoom fixed aperture changes and refocussing influences the focal length. All in all the parameterized method seems to be the most robust one also looking at the astonishing positioning results of figure 17.

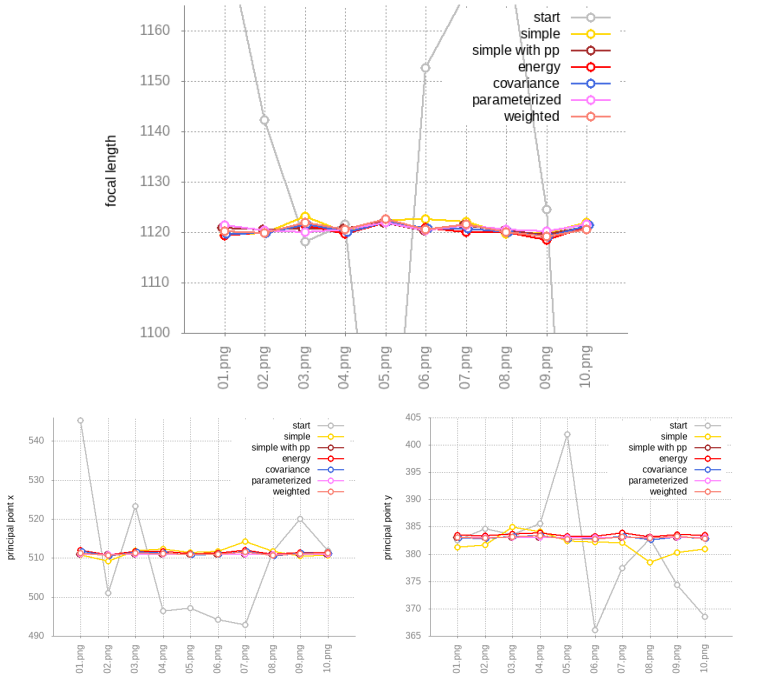
Improvements can still be done for finding correct color code correspondences. Light turquoise for example easily gets mixed up with deep blue when in the shade. It is best to have the calibration body in a well lit surrounding with uncolored and clear lights. A single wrong checkerboard correspondence ruins the whole bundle adjustment procedure. Currently the colors' hue values are compared for possible matches. Maybe some sort of auto white balancing or even different ways of uniquely marking a checkerboard face could make calibration in real world scenes even more robust.

Follow-up experiments can be done testing combinations of the different methods. One could for example use the parameterized grid approach in conjunction with the energy distance measure. Another different approach in the triangulation step could be to implement the point correspondence improvement by Chum et al [8]. There points lying on a common plane are refined before doing the triangulation so that their reconstruction also lies on a plane. This could be done for every checkerboard face in order to start off with a better initial world.

In summary the presented camera calibration method using the custom calibration body gives robust results with the most critical part being point correspondences due to difficulties in matching the color codes in bad lighting. The biggest advantage is the intuitive procedure with only one image per camera and simultaneous calculation of internal and external camera parameters.

References

1. Geiger, A., Moosmann, F., Car, O., Schuster, B.: Automatic Camera and Range Sensor Calibration using a single Shot. ICRA, 2012.
2. Zhang, Z.: A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11): 1330-1334, 2000.
3. Stewenius, H. D., Engels, C., Nistér, D.: Recent developments on direct relative orientation. ISPRS Journal of Photogrammetry and Remote Sensing, 60(4): 284–294, 2006.
4. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate O(n) solution to the pnp problem, International Journal of Computer Vision (IJCV), 81(2): 578–589, 2009.
5. Neubeck, A., Van Gool, L.: Efficient Non-Maximum Suppression. ICPR '06 Proceedings of the 18th International Conference on Pattern Recognition, Volume 03: 850-855, 2006.
6. Comaniciu D., Meer, P.: Mean Shift: A Robust Approach Toward Feature Space Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5): 603-619, 2002.
7. Arun, K. S., Huang, T. S., Blostein, S. D.: Least-Squares Fitting of Two 3-D Point Sets. IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5): 698-700, 1987.
8. Chum, O., Pajdla, T., Sturm, P.: The Geometric Error for Homographies. Computer Vision and Image Understanding 97: 86-102, 2005.



	f (variance)	p_x (variance)	p_y (variance)
start	1123.81 (2867.21)	509.34 (249.32)	380.72 (91.76)
simple	1121.10 (1.86)	511.43 (1.54)	381.84 (3.02)
simple with pp	1120.76 (0.36)	511.00 (0.00)	383.00 (0.00)
energy	1120.25 (0.96)	511.42 (0.15)	383.49 (0.06)
covariance	1120.52 (0.76)	511.12 (0.17)	382.98 (0.06)
parameterized	1120.82 (0.38)	511.00 (0.00)	383.00 (0.00)
weighted	1120.67 (1.04)	511.07 (0.07)	382.99 (0.04)

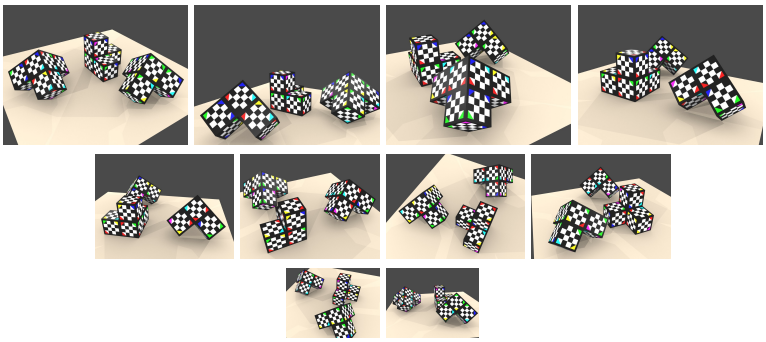
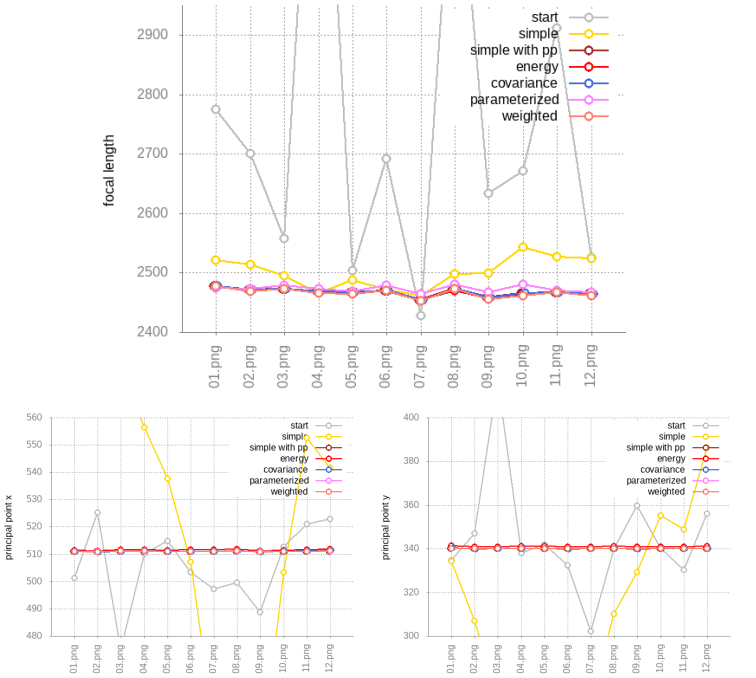


Fig. 10: Rendered scene results with known focal length of 1120. For these synthetic images the simple bundle adjustment with added principal point constraint performs best. Generally speaking the other proposed methods try to make room for outliers due to slightly off checkerboard detections. Clear and noiseless input images don't really benefit from that.



	f (variance)	p_x (variance)	p_y (variance)
start	2769.31 (102182.37)	505.92 (202.89)	344.95 (673.33)
simple	2500.60 (626.64)	519.22 (2954.71)	304.44 (2104.94)
simple with pp	2467.40 (39.49)	511.00 (0.00)	340.00 (0.00)
energy	2466.58 (40.63)	511.51 (0.03)	341.00 (0.01)
covariance	2467.28 (38.67)	511.05 (0.03)	340.00 (0.01)
parameterized	2472.82 (27.66)	511.00 (0.00)	340.00 (0.00)
weighted	2465.71 (50.68)	511.02 (0.01)	340.00 (0.00)

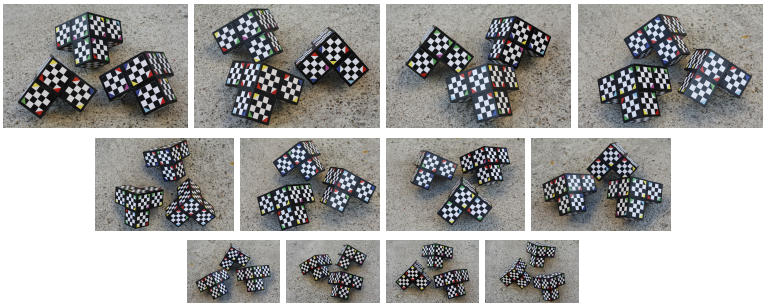


Fig. 11: Images taken with a Canon 450D, 50mm f1.6. Variations in the focal length are to be expected due to the lens focussing differently from image to image especially for low apertures. The results show that this set of images mostly gains from the principal point constraint.

01.png to 04.png	f (variance)	p_x (variance)	p_y (variance)
start	1692.16 (2463.85)	497.05 (299.39)	340.88 (2.7)
simple	1555.5 (64.8)	511.4 (338.66)	348.56 (54.38)
simple with pp	1579.77 (120.62)	511.0 (0.0)	340.0 (0.0)
energy	1576.42 (97.47)	510.81 (0.88)	341.32 (1.37)
covariance	1577.81 (97.53)	510.35 (0.69)	340.36 (1.63)
parameterized	1581.35 (35.6)	506.53 (10.18)	344.13 (8.13)
weighted	1577.44 (101.4)	510.69 (0.13)	340.07 (0.11)

05.png to 08.png	f (variance)	p_x (variance)	p_y (variance)
start	2516.93 (28095.72)	501.52 (133.46)	347.5 (394.58)
simple	2282.12 (1298.68)	487.66 (860.21)	379.16 (281.41)
simple with pp	2283.85 (550.79)	511.0 (0.0)	340.0 (0.0)
energy	2283.11 (455.77)	510.82 (0.32)	340.92 (0.34)
covariance	2283.98 (458.87)	510.36 (0.27)	339.92 (0.31)
parameterized	2281.09 (47.58)	508.02 (10.45)	341.55 (0.81)
weighted	2287.29 (756.19)	510.82 (0.05)	340.0 (0.03)

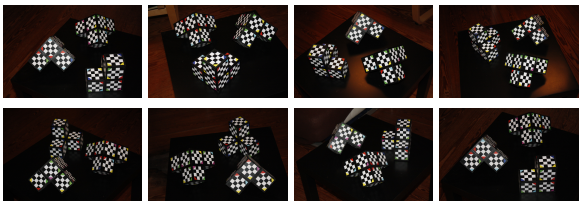
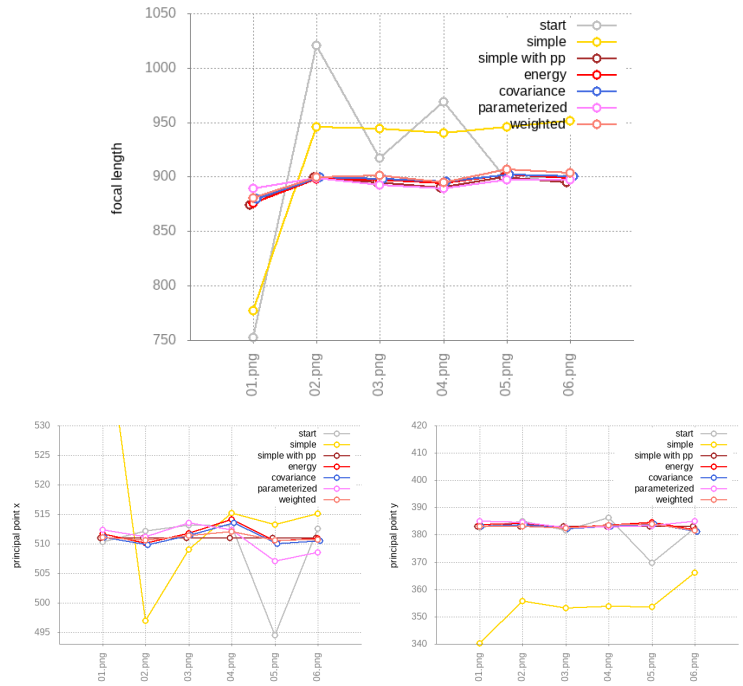
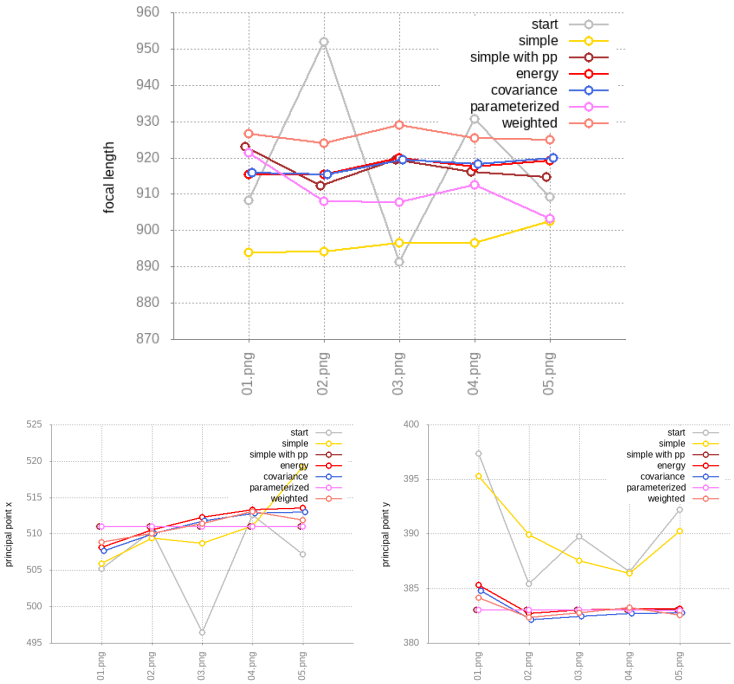


Fig. 12: Images taken with a Canon 450D, 18-55mm f3.5-5.6. Images 01.png to 04.png were taken with less zoom than 05.png to 08.png. The results table is divided into these two groups. Again it is hard to say whether lower variance really means better calibration since hardware dependent bigger focal length means less depth of field and causes refocussing from image to image.



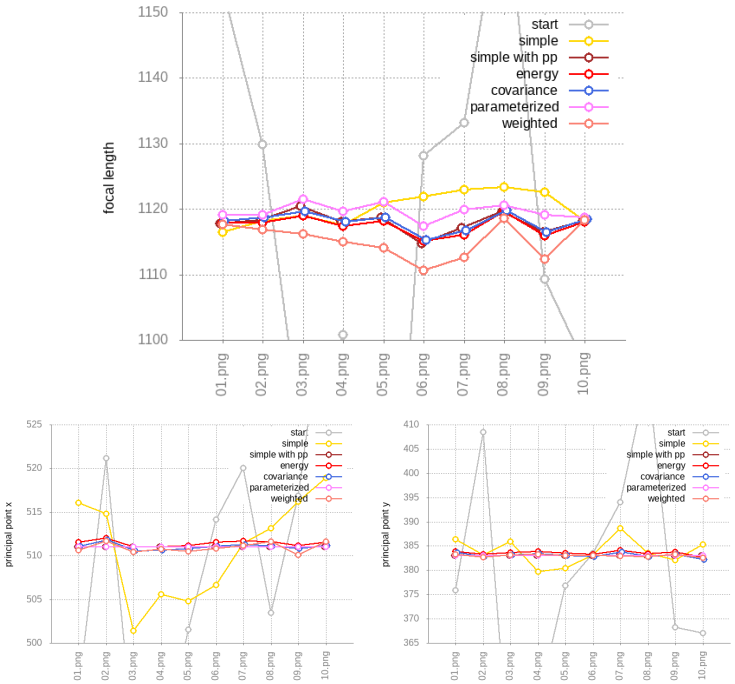
	f (variance)	p_x (variance)	p_y (variance)
start	909.54 (6832.49)	509.24 (44.64)	381.26 (28.83)
simple	917.32 (3948.56)	516.80 (279.31)	353.76 (56.63)
simple with pp	892.21 (78.95)	511.00 (0.00)	383.00 (0.00)
energy	894.18 (72.45)	511.48 (1.76)	383.37 (0.95)
covariance	895.83 (58.29)	511.05 (1.60)	382.85 (0.89)
parameterized	894.11 (16.23)	510.80 (5.37)	383.87 (0.94)
weighted	897.66 (69.55)	511.04 (0.29)	382.94 (0.51)

Fig. 13: Images taken with a Nexus 5 smartphone. In order to improve contrast in this low lighting setting the flash was turned on and all automatic image improvements like high dynamic range mode were disabled where possible.



	f (variance)	p_x (variance)	p_y (variance)
start	918.17 (439.88)	506.35 (30.95)	390.24 (18.41)
simple	896.70 (9.44)	510.86 (19.95)	389.87 (9.49)
simple with pp	917.09 (13.89)	511.00 (0.00)	383.00 (0.00)
energy	917.47 (3.39)	511.57 (4.07)	383.45 (0.89)
covariance	917.82 (3.45)	511.08 (4.02)	382.96 (0.88)
parameterized	910.55 (37.84)	511.00 (0.00)	383.00 (0.00)
weighted	925.97 (3.04)	511.03 (2.18)	383.01 (0.42)

Fig. 14: Images taken with a Nexus 5 smartphone with better lighting than the set before. From the results table it seems that the different methods are a trade-off between lower principal point variance and lower focal length variance. Of course by tweaking the weight of the principal point constraint one can drastically effect this behaviour but without ground truth data there's no way of knowing the best parameter settings for that.



	f (variance)	p_x (variance)	p_y (variance)
start	1114.61 (1238.10)	508.73 (244.82)	378.34 (556.29)
simple	1120.08 (5.71)	510.89 (31.30)	383.82 (6.86)
simple with pp	1117.92 (2.29)	511.00 (0.00)	383.00 (0.00)
covariance	1117.98 (1.89)	511.02 (0.12)	383.06 (0.20)
energy	1117.52 (1.82)	511.42 (0.10)	383.53 (0.17)
parameterized	1119.59 (1.33)	511.00 (0.00)	383.00 (0.00)
weighted	1115.22 (6.82)	510.91 (0.29)	382.99 (0.06)

	κ_1 (variance)	κ_2 (variance)
start	0.000e+00 (0.000e+00)	0.000e+00 (0.000e+00)
simple	-1.043e-04 (8.206e-11)	1.746e-08 (3.208e-16)
simple with pp	-9.213e-05 (4.094e-12)	-3.605e-10 (1.128e-20)
energy	-9.185e-05 (3.520e-12)	-3.748e-10 (1.375e-20)
covariance	-9.199e-05 (3.893e-12)	-3.595e-10 (1.338e-20)
parameterized	-9.507e-05 (2.310e-12)	9.044e-10 (1.232e-18)
weighted	-8.760e-05 (8.840e-12)	-4.208e-10 (1.848e-21)

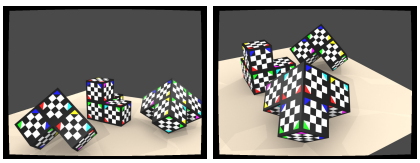
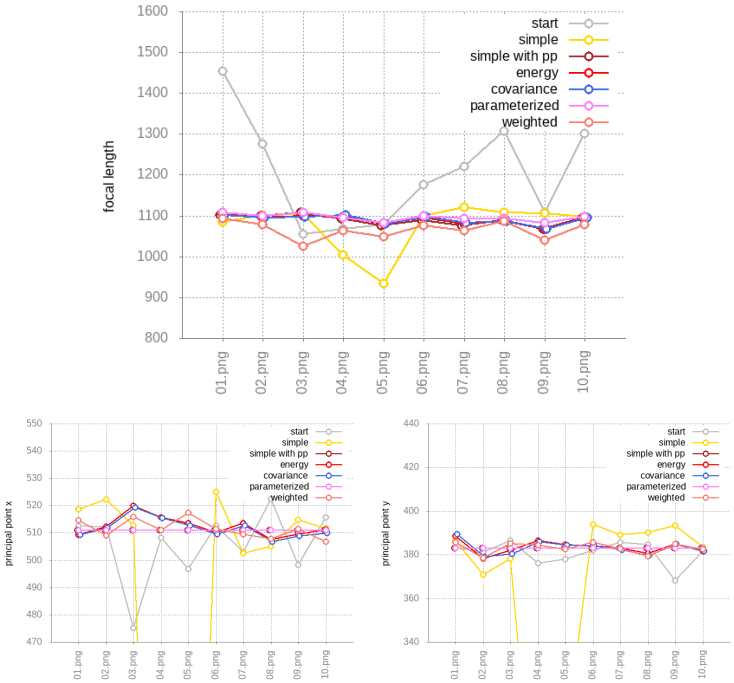


Fig. 15: Rendered scene (same as figure 10) radially distorted with $\kappa_1 = -1 \cdot 10^{-4}$ and $\kappa_2 = -1 \cdot 10^{-8}$. The results now start to show similar behaviour to the real image sets, which is slight variance in the principal point and focal length. Radial distortion indeed is another big influence especially with the rather few checkerboard corners we work with.



	f (variance)	p_x (variance)	p_y (variance)
start	1204.19 (15486.64)	505.65 (160.16)	380.66 (26.39)
simple	1075.54 (3157.46)	459.56 (12065.15)	360.91 (2717.09)
simple with pp	1089.25 (146.49)	511.00 (0.00)	383.00 (0.00)
energy	1090.07 (109.31)	512.23 (11.84)	383.37 (8.09)
covariance	1090.12 (106.91)	511.62 (12.10)	383.10 (9.17)
parameterized	1095.80 (69.71)	511.00 (0.00)	383.00 (0.00)
weighted	1065.10 (423.04)	511.46 (10.93)	383.11 (5.74)

	κ_1 (variance)	κ_2 (variance)
start	0.000e+00 (0.000e+00)	0.000e+00 (0.000e+00)
simple	-3.032e-04 (6.271e-08)	6.024e-08 (8.167e-14)
simple with pp	-3.209e-04 (4.339e-10)	-7.090e-11 (1.445e-18)
energy	-3.180e-04 (2.816e-10)	-2.195e-10 (1.265e-18)
covariance	-3.195e-04 (3.513e-10)	-1.507e-10 (1.378e-18)
parameterized	-3.386e-04 (2.504e-10)	1.582e-08 (1.113e-16)
weighted	-2.894e-04 (1.726e-10)	-9.565e-10 (1.987e-19)

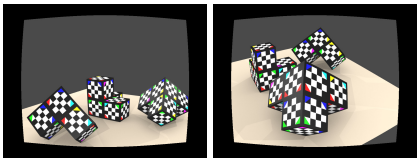
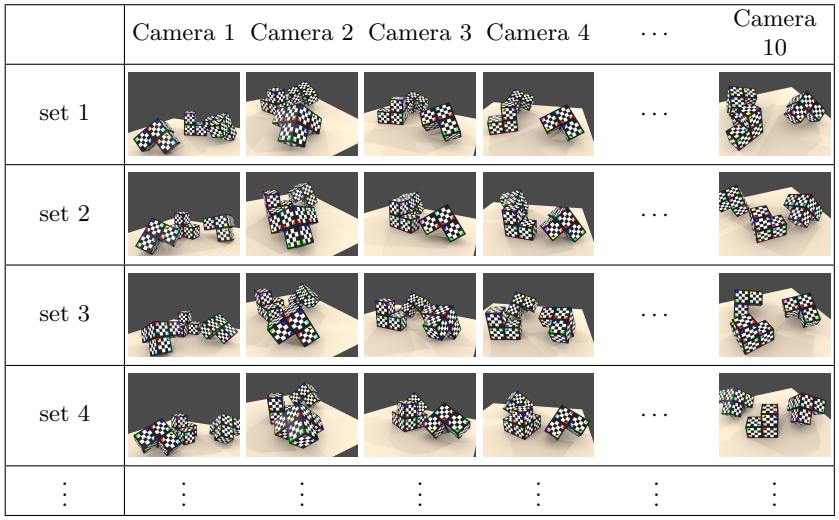


Fig. 16: Rendered scene (same as figure 10) radially distorted with $\kappa_1 = -5 \cdot 10^{-4}$ and $\kappa_2 = -1 \cdot 10^{-8}$. Where the distorted images were rather well calibrated in the set before, this is where it starts to get messy. k_1 is a little too low and variances get bigger.



mean errors	f	pp in pixels	position in mm	normal in degrees
start	37.1 (3.12e+03)	15.7 (181)	30.7 (1e+03)	0.0334 (0.00125)
simple	3.86 (39.1)	5.04 (125)	7.14 (52.9)	0.00501 (7.89e-05)
simple with pp	1.2 (1.92)	1.36 (0.156)	6.92 (52.4)	0.00248 (7.63e-06)
energy	1.15 (2.06)	0.833 (0.182)	6.91 (50.6)	0.00225 (7.88e-06)
covariance	1.19 (1.92)	1.36 (0.153)	6.95 (53.7)	0.00248 (7.64e-06)
parameterized	0.985 (0.843)	1.41 (6.93e-07)	0.527 (0.161)	0.00246 (7.59e-06)
weighted	1.45 (3.78)	1.39 (0.0484)	7.34 (40)	0.00253 (7.61e-06)

Fig. 17: In this experiment the calibration methods were tested on over 100 image sets where the calibration objects were randomly repositioned for each set with fixed cameras. The table shows the mean errors and variances of the calibration. The principal point error was calculated using the squared distance to the true value and is taken in pixels. This also is the first time evaluating estimated camara positions since for renderings the ground truth is known. It is assumed that the calibration body has a leg length of 14.4cm. The world coordinate system is scaled to match these dimensions.