OCPAD – Occluded Checkerboard Pattern Detector

Peter Fuersattel[§†], Sergiu Dotenco[§], Simon Placht[†], Michael Balda[†], Andreas Maier[§], Christian Riess[§]
[§] Friedrich-Alexander University of Erlangen-Nuremberg
[†] Metrilus GmbH
first.last@fau.de, first.last@metrilus.de

Abstract

Many camera calibration techniques require the detection of a pattern with known geometry, e.g.*, a checkerboard. Typically, the pattern must be fully contained in the field of view. This brings several limitations, one of which is that lens distortion can not reliably be estimated in outer image regions.*

This paper presents the occluded checkerboard pattern detector (OCPAD) to find checkerboards, even in a) low-resolution images, b) images with high lens distortion and if c) the pattern is partly occluded or not completely within the field of view. We exploit that checkerboards can easily be represented by a graph. We use graph matching to find the largest partial checkerboard in the image. Our detector complements a state-of-the-art calibration algorithm. Quantitatively, detection rates are considerably improved over the state-of-the-art. Additionally, estimation of lens distortion is greatly improved at outer image regions. Here, the reprojection error is improved by up to 50%.

1. Introduction

Many established camera calibration techniques rely on point correspondences between 3D objects in the scene and 2D points on the image to estimate the projective transformation and a set of lens model parameters. Commonly, high contrast patterns of known dimensions, like checkerboards, are used to automatically compute these correspondences. Calibration has been practiced for a long time. However, it is still far from being straightforward and user-friendly, since many methods impose constraints on the input data. Examples include different poses of the calibration pattern [18], or well-spread point correspondences throughout the whole image to find accurate lens model parameters. An important factor during camera calibration is the detection of the pattern. Capturing patterns throughout the whole field of view can be very complicated or even impossible, particularly under strong lens distortion. For less experienced users this task may become even more challenging and time-consuming. Thus, with a flexible, robust detection algorithm, calibration can be greatly simplified.

In this work, we present OCPAD, the Occluded Checkerboard Pattern Detector. OCPAD detects checkerboards that are partially occluded or not fully contained in the field of view. From an algorithmic perspective, these cases can be treated identically. By designing the method such that it can flexibly deal with missing and non-detected corners both design goals can be achieved at the same time.

We represent checkerboards as graphs, and use graph matching to find the largest common subgraph between the detected graph and a checkerboard model graph. As a result, it becomes straightforward to use calibration information in traditionally difficult image parts, like the image border and even the image corners. The method excells when estimating lens distortion at image regions that are far from the center pixels.

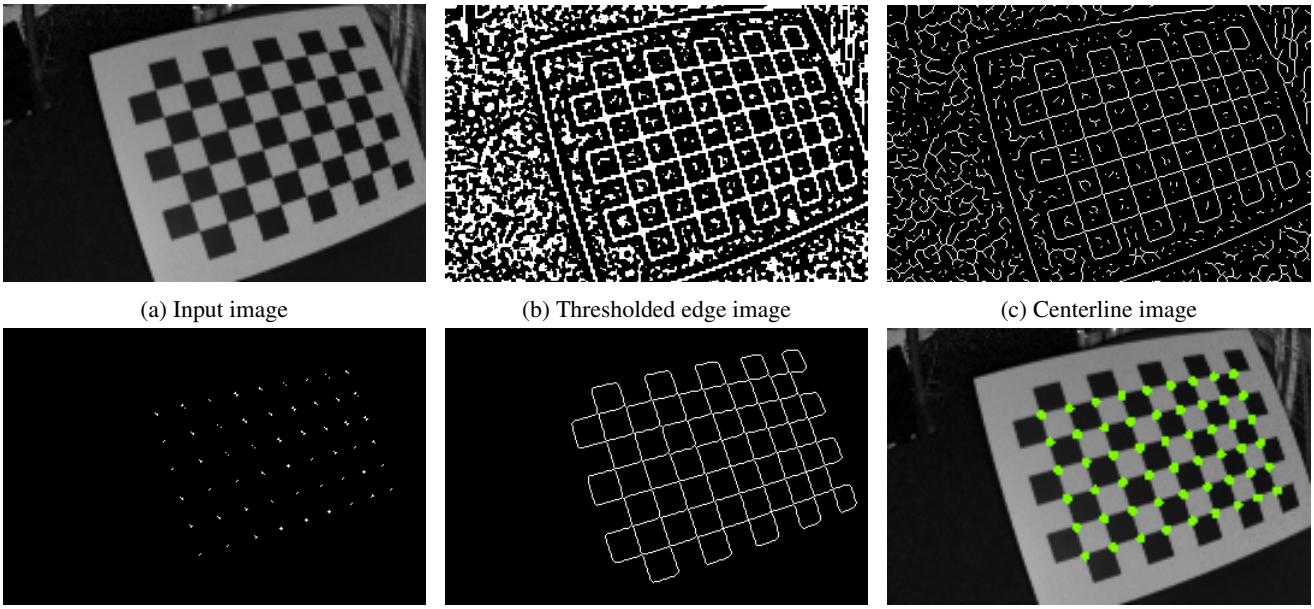
To demonstrate our approach, we use OCPAD in place of the graph matching component of the ROCHADE. The full method "inherits" from ROCHADE its robustness, particularly for low image resolution and strong lens distortion, but clearly improves on the pattern detection rate. OCPAD obtains more accurate results with fewer images and weaker constraints on the pattern position.

The proposed method is evaluated and compared to several other algorithms. Results show that OCPAD outperforms the best compared partial checkerboard detector by more than 29% with respect to detection rate. Furthermore, we investigate the impact of this improvement for camera calibration. Here, OCPAD reduces the reprojection error by up to 50% in outer image regions.

In Sec. 2 we present related work, Sec. 3 summarizes the ROCHADE calibration. The proposed OCPAD algorithm is described in Sec. 4. We evaluate its performance in Sec. 5, and conclude with a brief discussion in Sec. 6.

2. Related Work

Several of the large number of calibration frameworks contain semi-automatic or fully-automatic checkerboard detection algorithms, e.g. [2, 16]. A popular implementation



(a) Input image (b) Thresholded edge image (c) Centerline image

(d) Saddle Points (e) Candidate component (f) Result

Figure 1: Intermediate steps of ROCHADE. (a) input image with completely visible checkerboard. From the masked edge strength image (b) the centerline image is computed (c). A graph representation is built from saddle points (d) and a connected component (e). This graph must match a reference graph of the board (f).

is contained in OpenCV. It includes a checkerboard detector that isolates checkerboard corners by applying different thresholding and morphological operations. The inner corners of the board are found by merging detected quads. Rufli *et al*. [15] increased the robustness of OpenCV's approach with respect to lens distortion. Both algorithms share the same corner refinement method. Placht *et al*. [14] proposed a more accurate corner refinement. Bennet and Lasenby [1] reliably detect checkerboard crossings. However, their method does not recover the complete checkerboard information which is required for camera calibration. Fiala and Shu [8] and Forman *et al*. [9] use self-identifying markers to increase calibration accuracy and to simplify the estimation of the pattern pose. However, markers may easily fail for low-resolution cameras, (*e.g*., Time-of-Flight sensors often provide less than 200×200 pixels). Two approaches aim at finding checkerboard-like structures by applying the Hough transform [6, 10] to detect the parallel checkerboard lines. This method works well with low resolution images, but not under strong lens distortion (which has to be addressed with additional preprocessing).

Several other toolboxes come with a checkerboard detector that is able to find partially occluded patterns. We compare our work to two of these algorithms that are widely used: the Omnidirectional Camera Calibration Toolbox (OCAMCALIB) [16] and the Parallel Tracking and Mapping toolbox (PTAM) [12]. Note that PTAM is a camera tracking system and not primarily made for camera calibration. Nonetheless, PTAM includes calibration functionality with accurate and robust checkerboard detection.

3. ROCHADE

The ROCHADE detection pipeline consists of two steps: initial checkerboard detection and refinement of the inner checkerboard corners. Both steps are briefly presented below. Please refer to the original paper [14] for additional details.

3.1. Checkerboard Detection and Graph Extraction

Figure 1 illustrates the steps of this operation. The input image may optionally be downsampled to reduce computational complexity (Fig. 1a). Next, the image gradient is computed by using the Scharr kernel [17]. Thresholding on the gradient magnitude image may create multiple connected components, which serve as checkerboard candidates (Fig. 1b). Isolated pixels and small branches are removed by several filtering steps. Then, a thinned centerline image [13] (Fig. 1c) and saddle points (Fig. 1d) are computed. The latter two are used to construct a graph.

The detector searches for a connected component with the same number of nodes as the checkerboard (Fig. 1f), and with edges that also match the pattern (Fig. 1f). Thus, ROCHADE can only detect fully visible checkerboards.

3.2. Subpixel Corner Refinement

The location of the checkerboard corners is refined with subpixel accuracy. The refinement method consists of a smoothing operation and a polynomial fit for each corner coordinate. In detail, if the region around a checkerboard corner is interpreted as a surface, then the corner is located exactly at the saddle point. This saddle point is computed by fitting a polynomial to a lowpass-filtered window around the initial corner coordinate. This window size has to be chosen such that the windows only contain a single corner candidate [14].

4. Occluded Checkerboard Detector (OCPAD)

We present a new checkerboard detection algorithm, OCPAD, which exploits the graph representation of checkerboards to find full and partially occluded checkerboards. By including robust subgraph matching into the candidate graph verification it is possible to achieve error-tolerance with respect to missing checkerboard corners. This tolerance leads to multiple improvements: a more simple image capturing process, as the user has to care less about the visibility of the pattern. Additionally, higher detection rates can be expected, and furthermore, the lens model is estimated more accurately.

We assume that a graph representation of the input image has already been computed. In our implementation, OCPAD plugs behind the graph construction of ROCHADE, or Fig. 1 (e), respectively. Our algorithm takes this graph as input and robustly matches it to a checkerboard model graph to obtain the largest possible checkerboard. We illustrate important steps of the algorithm in Fig. 2. Here, the input image (Fig. 2 (a)) is partially occluded. We omit illustrations of the graph construction from ROCHADE and immediately show in Fig. 2 (b) the connected component which is used for graph construction. Fig. 2 (c) shows the graph representation of this structure. OCPAD consists of several consistency checks, and an outlier-tolerant matching strategy. We show one intermediate processing stage (described in greater detail in algorithm step 3 below) in Fig. 2 (d). The matching result is shown in Fig. 2 (e).

Since we directly start with the graph representation, all filtering and consistency checks are also performed on the graph structure instead of the intensities of the pixels. The processing steps of the proposed method are:

1. Reject graphs with very few nodes. If only very few feature points are available, estimation of the checkerboard pose becomes unreliable, which may lead to inaccurate calibration results [18]. Hence, we require that the candidate graph must consist of at least 50% of the nodes of the model graph.

2. Spatial consistency of the graph. The two criteria evaluated here are: a minimum inter-node distance and whether the standard deviation of the length of the edges is smaller than the average length of the edges in image domain. Particularly if multiple graph structures are found, this effectively removes background (i.e., bad graph candidates) before the relatively costly subgraph matching algorithm is employed. The threshold for the inter-node distance is set to the same value as the window size in Sec. 3.2, since it effectively represents the lower bound for the minimum distance between corners.

3. Quad graph filter. In this step all single checkerboard quads which are contained in the candidate graph are detected. We remove all nodes that are not a corner of a quad. This filter removes isolated lines and triangles which may be present in the candidate graph.

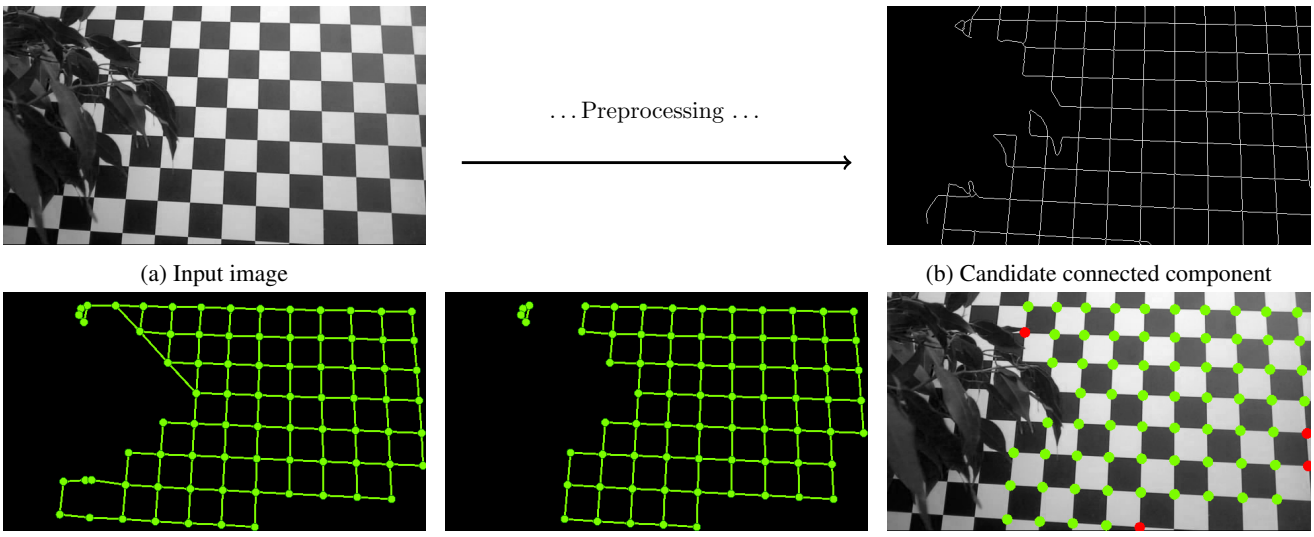
4. Select the largest connected component. By the previous graph editing steps, the candidate graph may have become disconnected. There may exist many ambiguous ways to map multiple disconnected components onto the same checkerboard model graph. To avoid this ambiguity, we only consider the largest remaining connected component of the candidate graph. Although matching this subgraph to the model graph may also lead to non-unique solutions, this is not an issue: for intrinsic camera calibration correspondences between detected corners and three-dimensional corner coordinates are required. However, this mapping does not have to be unique. It is only required the mapping reflects the structure between the detected board and the true checkerboard correctly.

5. Find the largest common subgraph. We use the algorithm by Cordella *et al*. [5] for finding the largest common subgraph of the candidate graph and the checkerboard model. However, this algorithm is only able to deal with partial matches, or non-matchable extra nodes. To achieve error tolerance during the graph matching, we start with searching for an exact match for a small subgraph of the candidate graph. We then iteratively increase the subgraph in size, until the largest possible subgraph match is found. Additional details on the current implementation of the matching algorithm are presented in Section 4.1.

The matched graph can then be used in the full calibration algorithm. We pass the matching result to the subpixel corner refinement of the pipeline, described in Sec. 3.2.

4.1. Error-Tolerant Sub-Graph Matching

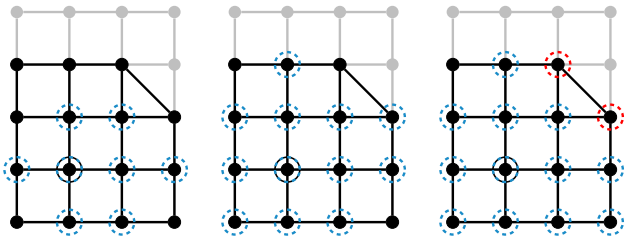
Goal of the matching is to obtain the largest common subgraph of a checkerboard model graph and the detected



(a) Input image (b) Candidate connected component

(c) Graph of the candidate component (d) Two components after quad filtering (e) Result

Figure 2: Intermediate steps of the proposed method. Figure (a) shows the input image. The candidate component (b) and the resulting detected graph shown in Figure (c). This graph might get separated by the quad filter as shown in (d). In the next step the largest component is matched to the original pattern graph. The result is shown in Figure (e). Green and red dots represent the checkerboard corners and indicate whether the corner refinement succeeded or not.



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

Figure 3: Subgraph matching strategy. Illustration for the binary search for a detected graph G_d of order $N = N_0 = 15$. In each iteration the algorithm tries to match the encircled vertices. The anchor vertex is encircled solidly. In the first iteration $G_0 = G_d$ can not be matched due to the wrongly detected diagonal edge. Next a subgraph G_1 of size $N_1 = \lfloor N_0 + 2^{-1}N \rceil = 8$ is evaluated. This succeeds, subsequently the subgraph size is increased to $N_2 = \lfloor N_1 + 2^{-2}N \rceil = 12$ vertices for creating G_2. G_2 can be matched to G_m, therefore G_3 is created with $N_3 = \lfloor N_2 + 2^{-3}N \rceil = 14$ vertices. Finally, G_3 can not be matched to G_m as the diagonal edge is introduced.

graph. This matching algorithm has to fulfill the following requirements: a) it has to find an as-large-as-possible non-injective, non-surjective mapping between the detected graph and the model and, b) it has to tolerate errors in the graph structure, e.g., missing or additional edges or vertices.

In the current implementation, we use the popular VF2 algorithm [5] for subgraph matching. Note that the VF2 algorithm is limited to finding exact matches of a subgraph, instead of finding the largest common subgraph.

An example for this can be found in Fig. 3 (a). The checkerboard model is shown in gray. The detected graph is shown in black. Due to the diagonal edge, which does not exist in the model (shown in gray), the original VF2 algorithm is unable to find a valid mapping between the graphs.

To obtain the required tolerance to missing or extra vertices, we cascade two subgraph search strategies: we first perform a binary search over the nodes of the subgraph, and then complement the graph using region growing. Efficient matching of planar graphs with a certain error tolerance has been considered earlier in the literature [7]. However, we consider in our application the performance of the matching algorithm as not overly critical since only small graphs are matched.

Our search strategy requires the choice of an anchor vertex where the search begins. Ideally this vertex is located in a densely populated region of the graph. This allows to find a large part of the checkerboard with a binary search over the number of nodes. We examined multiple ways to find the anchor vertex, for example the vertex which is closest to the center of mass of the detected graph (in image domain) or the vertex where the sum of all shortest distance to all other vertices is minimal. In practice we found that a different approach based on the quad density works best. We obtain this density for every vertex by counting the number of

Algorithm 1: Binary search for error-tolerant matching

Data: Model graph G_m, Detected graph G_d
Result: Mapping M between G_m and G_d
$i \leftarrow 0$; $N \leftarrow |G_c|$; $N_i \leftarrow N$
$a \leftarrow findAnchor(G_d)$
$D \leftarrow computeDistancesToAnchor(G_d, a)$
while *true* **do**
  $G_i \leftarrow getSubgraph(G_d, D, N_i)$
  $M_i \leftarrow VF2\_match(G_i, G_m)$
  $i \leftarrow i + 1$
  if $validMapping(M_i)$ **then**
    $N_i \leftarrow round(N_{n-1} + 2^{-i}N)$
    $M \leftarrow M_i$
  else
    $N_i \leftarrow round(N_{n-1} - 2^{-i}N)$
  if $N_i >= N_i - 1$ **then**
    break

adjacent vertices which are part of a quad cycle via breadth-first search down to a depth of 3. The vertex with the highest density is then selected as anchor. This approach reduces the influence of non-checkerboard vertices effectively.

An outline of the binary search approach is given in Algorithm 1. In the following we describe the algorithm as we analyze the example shown in Fig. 3. First, all vertices are ordered via breadth-first search with respect to their distance to the anchor vertex (solidly encircled). The model graph G_m is shown in gray. We perform a binary search over the $N = N_0 = 15$ vertices of the detected graph G_d, which is drawn black. For the binary search, let N_i be the number of vertices that are selected in iteration i. The first iteration's graph G_0 equals G_d. Due to the diagonal edge G_d can not be matched to G_m, thus the tested number of nodes in the next iteration is set to $N_1 = \lfloor N_0 - 2^{-1}N \rceil = 8$. G_1 can be matched to G_m, as shown in Fig. 3 (a). As a consequence, the binary search continues to search for a larger number of nodes $N_2 = \lfloor N_1 + 2^{-2} \rceil = 12$ (Fig. 3 (b)). The subgraph G_2 can be matched to G_m, therefore the number of nodes is increased to $N_3 = \lfloor N_2 + 2^{-3} \rceil = 14$. Due to the diagonal edge G_3 can not be matched as shown in Fig. 3 (c). This process is repeated until N_i converges.

Note that due to potential extra nodes or merged nodes in the image graph, the binary search may fail before the full number of nodes is found. In this case, we continue to add single vertices in a breadth-first manner with respect to the anchor point, and iteratively match the obtained graph to the model. That way, the matched graph can grow around the erroneously detected nodes.

Typically, this approach requires only a few iterations, as the detected graph is in most cases not strongly fragmented. The theoretical maximum number of iterations is proportional to the number of vertices in the detected graph. However, due to the strong connectivity of checkerboard graphs, large parts of the board can be matched in the binary search phase which drastically reduces the amount of required iterations. Timing results on the benchmark datasets are shown in Sec. 5.4.

5. Evaluation

We first show qualitative results for several images to demonstrate the performance of OCPAD in Sec. 5.1. Then, we evaluate the overall detection rate and compare it to three state-of-the-art algorithms in Sec. 5.2. In Sec. 5.3, we evaluate the impact of OCPAD on the accuracy of the resulting calibration. Finally, we provide and compare measured runtimes in Sec. 5.4.

Five data sets are used in this evaluation. The first three sets belong to the publicly available ROCHADE evaluation data, containing low resolution images with strong lens distortion (Mesa SR4000), medium resolution images with little distortion (IDS uEye) and high resolution images with strong fisheye distortion (GoPro Hero 3). The fourth and fifth data sets have been recorded with a wide angle camera at a resolution of 1280×720 pixels. The fourth data set contains only fully visible boards. The fifth data set contains fully and partially visible boards positioned close to the image border.

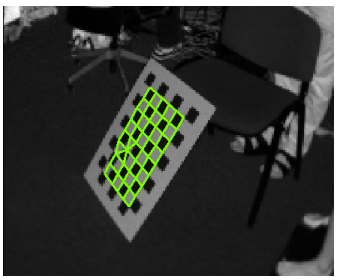
We use the publicly available implementations of the OCamCalib toolbox [16] and the PTAM algorithm [12], and our own implementation of ROCHADE [14]. Note that PTAM is a camera tracking system for augmented reality applications and not primarily designed for camera calibration. Nonetheless, PTAM comes with calibration functionality which requires robust calibration pattern detection. We provide a publicly available implementation of OCPAD[1].

5.1. Qualitative Examples

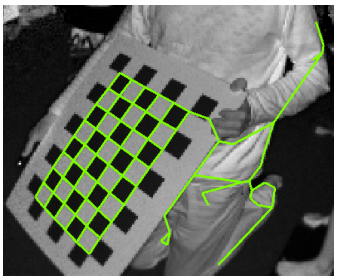
Representative results are shown in Fig. 4. In Fig. 4a, two checkerboard x-junctions are merged erroneously during preprocessing. This results in a wrong number of vertices and edges, such that ROCHADE fails. This vertex is not removed by the quad filter as it belongs to multiple complete quads, even though it creates triangle structures. With OCPAD, the majority (42 of 54 corners) of the pattern are detected accurately and can be used for calibration.

Two additional examples are shown in Fig. 4b and Fig. 4c. In both images ROCHADE fails as the intermediate candidate graph is connected to the background. OCamCalib finds only 26 of the 54 corners shown in Fig. 4c, due to foreshortening and the low resolution. OCPAD detects both cases, due to its robustness towards low resolution and error-tolerant subgraph matching.

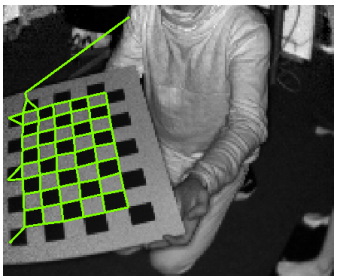
[1] http://www.metrilus.de/software/downloads



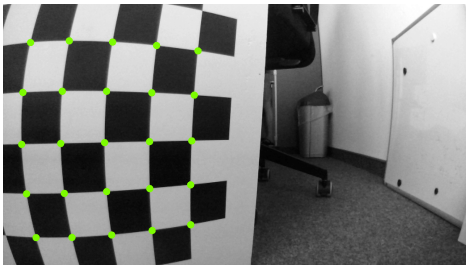
(a) Wrongly merged vertices



(b) Connected outlier vertices



(c) Connected outlier vertices



(d) Partially occluded, distorted pattern

(e) Occluded pattern

Figure 4: Qualitative Results. (a) No complete detection possible as the saddle points have been merged wrongly. However, 42 of the 54 correctly detected corners can be used for calibration. (b-c) In these examples the underlying method fails as the candidate graph is connected to the background. In (c) OCPAD finds the complete pattern, while OCamCalib finds only 26 corners. In figures (d) and (e) examples for partially occluded and distorted patterns are shown. In (d) OCPAD is able to detect more corners as OCamCalib (25 vs. 18). In (e) the central part of the pattern is occluded. Both detectors find 66 corners. Figures are viewed best in color.

Figures 4d and 4e show two additional examples. In both images the patterns are either partly occluded or outside of the field of view. In Fig. 4d the pattern is also heavily distorted by the camera's lens. In Fig. 4d OCPAD detects 25 of the 26 visible corners while OCamCalib finds only 18 corners. In the occluded example, shown in Fig. 4e, both detectors find all possible corners.

5.2. Detection Rates

Obtained detection rates are listed in Table 1. For partial detectors, different tolerance thresholds for missed corners are evaluated. We considered a detection successful if either 100%, 90% or 75% of all corners have been found.

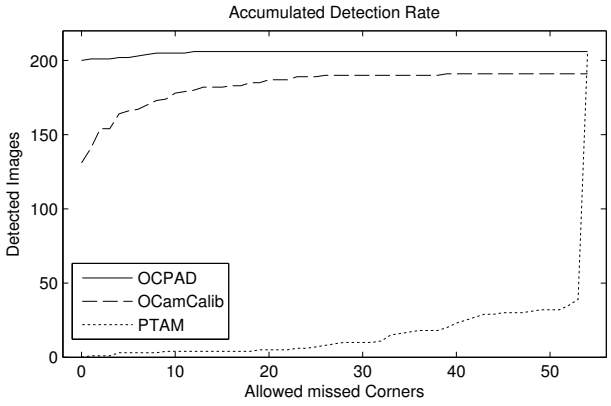
In the less challenging IDS uEye and GoPro data sets, ROCHADE, OCamCalib and OCPAD find full checkerboards in almost all images. However, in the more challenging Mesa SR4000 data set, OCPAD outperforms all other partial detectors by more than 10%. Furthermore, the presented method is the only method that is able to find all patterns at a tolerance threshold of 75%. ROCHADE is the second best method with this data set if 100% of the corners shall be detected, which indicates that its preprocessing is part of the reason for the good performance. However, OCPAD outperforms ROCHADE due to the more robust checkerboard matching.

In the full board data set at 100% required corners, OCPAD detects 100% of the corners in the full board data in seven more checkerboards than the best baseline method OCamCalib. Nonetheless, OCamCalib benefits from the high image resolution. In the full and partial set at 100%, OCamCalib slightly outperforms OCPAD, but only by two images. OCPAD consistently outperforms the other methods if only 90% or 75% of the corners are required for detection. The only exception are full and partial boards at 75%, where OCamCalib and OCPAD perform comparably.

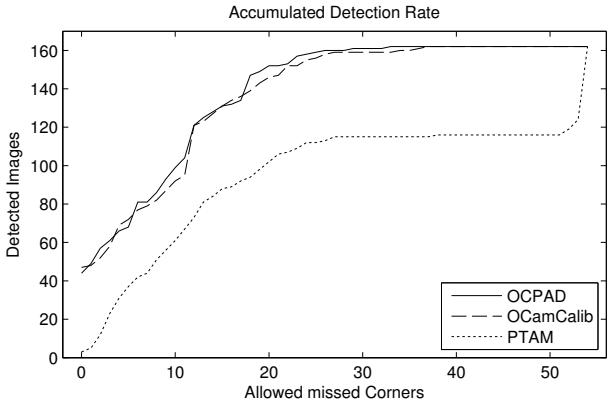
Figure 5 visualizes the detection rates for the two most challenging data sets, the MESA SR4000 set and the full and partial board set. In Fig. 5a, when allowing the detector to miss 12 corners, the proposed method is able to find the checkerboards on every image of the MESA set. OCamCalib is able to detect 191 patterns, if detections are considered to be valid with only 50% of the whole pattern found. The detection algorithm included in PTAM cannot find any complete patterns in the image set. No parameter set could be found which allows to improve the detection rate of this method. The reason for the poor performance might be the low resolution of the data set. Figure 5b shows the results for the full and partial boards data set. This plot supports the results given in Tab. 1, as OCamCalib and OCPAD perform comparably well. Due to the higher resolution the detector

		Mesa SR4000	IDS uEye	GoPro Hero 3	Full Boards	Full + Partial Boards
t_m	Total images	206	206	100	162	162
100%	ROCHADE	195	205	96	153	44
	OCamCalib	131	**206**	**100**	155	**46**
	PTAM	0	165	31	3	3
	OCPAD	**200**	205	**100**	**162**	44
90%	OCamCalib	180	**206**	**100**	**162**	95
	PTAM	4	192	57	118	73
	OCPAD	**203**	**206**	**100**	**162**	**104**
75%	OCamCalib	182	**206**	**100**	**162**	127
	PTAM	4	192	60	119	84
	OCPAD	**206**	**206**	**100**	**162**	**128**

Table 1: Detection rate results computed on the image sets of ROCHADE and new data. Different thresholds t_m for minimum required corners are evaluated. OCPAD excells particularly on the challenging Mesa and Full+Partial datasets.



(a) MESA SR4000 data set (176x144 pixels), 206 total



(b) Full and partial boards (1280x720 pixels), 162 total

Figure 5: Detection rates by missed corners. On the x-axes are the missed corners. On the y-axes are the images where at least $54 - x$ corners were detected.

included in PTAM finds larger parts of the checkerboards.

5.3. Calibration Accuracy

In this section we show that including partial checkerboards which cover the outer image regions leads to more accurate calibrations. This is particularly the case towards the image border where lens distortion typically becomes more severe. Furthermore, we present results which imply that this claim holds independent of the lens model. We also demonstrate that the accuracy achieved with a combination of fully and partially visible boards cannot be obtained if only full boards are used.

In this experiment, the full boards set and the full and partial boards set are used. For assessing the suitability of the calibration images we randomly sample 50 images of each data set and estimate the intrinsic parameters with Zhang's method [18]. This step is repeated 25 times in order to minimize the impact of the selection of images, which results in 25 sets of calibration parameters for image set.

We first used Brown's lens model [3], which is commonly used for camera calibration. For this model we use three radial and two tangential distortion terms, similarly as in OpenCV or other calibration toolboxes [2]. Kannala and Brandt [11] presented a generic lens model $d(\theta)$. Unlike Brown's lens model, it does not use the distance to the principal point, but instead the angle θ between the incoming light ray and the principal axis,

$$d(\theta) = k_1\theta + k_2\theta^2 + k_3\theta^3 + \dots \quad (1)$$

Experiments by Chtchetinine have shown that even-order terms up to the power of 6 are well-suited for camera calibration [4]. Note that this model does not include tangential distortion.

For measuring the reprojection error, cameras are calibrated on the two benchmark data sets. Using this calibra-

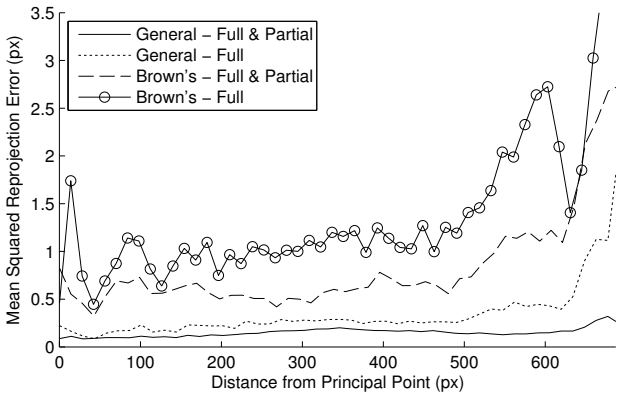


Figure 6: Evaluation of the calibration accuracy for two lens models.

tion, the radially-dependent reprojection error is calculated on a third (validation) set. This set contains checkerboard images which are evenly distributed throughout the whole image plane. The patterns in the third are detected with OCPAD. We report the reprojection error for a point and its distance to the principal point. The resulting errors are binned into 50 equidistant sections and averaged to improve clarity of the plots.

Figure 6 shows that Brown's model is not suitable for capturing the characteristics of the camera lens, due to the fact that the reprojection error increases with the distance to the principal point. It can also be seen that including partially visible checkerboards at the image border leads to a considerably lower overall reprojection error. However, the overall reprojection error can clearly be reduced by using the general lens model. This leads to the insight that the general model better fits this lens.

For the general lens model and Brown's lens model, the radially dependent reprojection error is considerably lower if partially visible patterns at the image border are included. Generally, OCPAD achieves the lowest reprojection errors using the general lens model. Note that due to the fact that parts of the patterns are occluded, less point correspondences can be obtained. Yet, having correspondences in highly distorted areas outweighs this disadvantage.

5.4. Detection Runtime

Average runtime measurements of evaluated algorithms for all five datasets are listed in Table 2. In OCPAD, the majority of the processing time is used for graph matching. Variations in image dimensions barely affect the algorithm. Instead, OCPAD's runtime depends on how challenging the matching of the detected graphs is. This property can be observed best with the GoPro data set. The baseline methods require multiple seconds, but OCPAD finds the patterns in less than 500 ms. OCamCalib is extremely fast on the Mesa SR4000 data set. However, its detection performance is not competitive (see Tab. 1). On the full and partial data set, OCPAD is somewhat slower than OCamCalib, due to additional iterations during graph matching.

Image set	OCamCalib	PTAM	OCPAD
Mesa SR4000	**69** ms	437 ms	367 ms
IDS uEye	604 ms	3891 ms	**379** ms
GoPro Hero 3	12431 ms	7310 ms	**477** ms
Full Boards	374 ms	4605 ms	**361** ms
Full + Partial Boards	**412** ms	3760 ms	439 ms

Table 2: Average runtime for different image sets.

6. Conclusion and Outlook

We present a new checkerboard detector, OCPAD, that considerably improves detection rates in challenging cases higher over state-of-the-art methods. For less challenging cases, the performance does not degrade compared to the other methods. Particularly, OCPAD is able to find partially occluded patterns. OCPAD allows the user to compute more accurate calibrations from a lower number of images.

In our implementation, we use subgraph matching for finding the largest possible calibration pattern. The method is robust to various types of graph detection errors. We present extensive experiments which evaluate the detection rate and the resulting accuracy of the camera calibration. The results show that the proposed algorithm clearly outperforms the baseline algorithms when applied on challenging data sets. If partially visible checkerboards are used, the accuracy of intrinsic camera calibrations can be considerably increased by up to 50% in outer image regions.

The current graph matching algorithm which does not yet exploit all properties of planar graphs that occur in two-dimensional calibration patterns. We will investigate this in future work.

Acknowledgements

This work was partly supported by the German Federal Ministry of Education and Research as part of the Spitzencluster Medical Valley program 13GW0029A. This work was partly supported by the Research Training Group 1773 "Heterogeneous Image Systems", funded by the German Research Foundation (DFG). The authors gratefully acknowledge funding of the Erlangen Graduate School in Advanced Optical Technologies (SAOT) by the German National Science Foundation (DFG) in the framework of the excellence initiative.

References

[1] S. Bennett and J. Lasenby. ChESS - quick and robust detection of chess-board features. *Computer Vision and Image Understanding*, 118:197–210, 2014. 2

[2] J. Y. Bouguet. Camera calibration toolbox for Matlab, 2008. 1, 7

[3] D. C. Brown. Decentering distortion of lenses. *Photometric Engineering*, 32(3):444–462, 1966. 7

[4] A. Chtchetinine. Radial distortion in low-cost lenses: numerical study. *Optical Engineering*, 47(2):023001–023001, 2008. 7

[5] L. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub) graph isomorphism algorithm for matching large graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10):1367–1372, 2004. 3, 4

[6] A. de la Escalera and J. M. Armingol. Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration. *Sensors*, 10(3):2027–2044, 2010. 2

[7] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 632–640, 1995. 4

[8] M. Fiala and C. Shu. Self-identifying patterns for plane-based camera calibration. *Machine Vision and Applications*, 19(4):209–216, 2008. 2

[9] C. Forman, M. Aksoy, J. Hornegger, and R. Bammer. Self-encoded marker for optical prospective head motion correction in MRI. *Medical Image Analysis*, 15(5):708–719, 2011. 2

[10] M. Hansard, R. Horaud, M. Amat, and G. Evangelidis. Automatic detection of calibration grids in time-of-flight images. *Computer Vision and Image Understanding*, 121:108–118, 2014. 2

[11] J. Kannala and S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(8):1335–1340, 2006. 7

[12] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 6th IEEE and ACM International Symposium on*, pages 225–234, 2007. 2, 5

[13] C. W. Niblack, P. B. Gibbons, and D. W. Capson. Generating skeletons and centerlines from the distance transform. *CVGIP: Graphical Models and Image Processing*, 54(5):420–437, 1992. 2

[14] S. Placht, P. Fürsattel, E. Mengue, H. Hofmann, C. Schaller, M. Balda, and E. Angelopoulou. ROCHADE: Robust checkerboard advanced detection for camera calibration. In *Computer Vision, European Conference on*, volume 8692 of *LNCS*, pages 766–779. 2014. 2, 3, 5

[15] M. Rufli, D. Scaramuzza, and R. Siegwart. Automatic detection of checkerboards on blurred and distorted images. In *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pages 3121–3126. IEEE, 2008. 2

[16] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5695–5701, 2006. 1, 2, 5

[17] H. Scharr. Optimal filters for extended optical flow. In B. Jähne, R. Mester, E. Barth, and H. Scharr, editors, *Complex Motion*, volume 3417 of *LNCS*, pages 14–29. Springer Berlin Heidelberg, 2007. 2

[18] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000. 1, 3, 7