Automatic Detection of Checkerboards on Blurred and Distorted Images

Martin Rufli, Davide Scaramuzza, and Roland Siegwart
Autonomous System Lab, ETH Zurich, Switzerland
ruflim@ethz.ch, davide.scaramuzza@ieee.org, r.siegwart@ieee.org

***Abstract*— Most of the existing camera calibration toolboxes require the observation of a checkerboard shown by the user at different positions and orientations. This paper presents an algorithm for the automatic detection of checkerboards, described by the position and the arrangement of their corners, in blurred and heavily distorted images. The method can be applied to both perspective and omnidirectional cameras. An existing corner detection method is evaluated and its strengths and shortcomings in detecting corners on blurred and distorted test image sets are analyzed. Starting from the results of this analysis, several improvements are proposed, implemented, and tested. We show that the proposed algorithm is able to consistently identify 80% of the corners on omnidirectional images of as low as VGA resolution and approaches 100% correct corner extraction at higher resolutions, outperforming the existing implementation significantly. The performance of the proposed method is demonstrated on several test image sets of various resolution, distortion, and blur, which are exemplary for different kinds of camera-mirror setups in use.**

I. INTRODUCTION

Cameras can appear either with limited field of view (i.e. perspective cameras) or with wide field of view. Wide field of view cameras can be built by using fisheye lenses (e.g. Nikon or Sigma) [1] or by combining a standard perspective camera with a shaped mirror (i.e. catadioptric omnidirectional cameras, Fig. 1) [2].

Accurate camera calibration is necessary for any computer vision task requiring extracting metric information of the environment from 2D images, like in ego-motion estimation and structure from motion. All works on camera calibration can be classified into two different categories. The first one includes methods which exploit prior knowledge about the scene, such as the presence of calibration patterns (e.g. see [3], [4], [5], [6], [7]) or lines ([8] or [5], [9] for an overview). The second group covers techniques that do not use this knowledge; this includes calibration methods from pure rotation or planar motion of the camera [10], and self-calibration procedures, which are performed from point correspondences and epipolar constraint through minimizing an objective function [11], [12], [13].

In the last decade, several toolboxes have been implemented, which allow any user to easily calibrate a camera (perspective, fisheye, or omnidirectional) by using a planar checkerboard as a calibration pattern (for perspective cameras see as an example [14], for fisheye and omnidirectional cameras see [15], [16]). These toolboxes require the user to take several pictures of the pattern shown at a few different positions and orientations. Then, the user is asked to identify the corner points of the checkerboard, which are used as the only input to the calibration routine. The tooboxes given in [14] and [16] require the user to identify the four external corners of the checkerboard in every test image by manually clicking on them. The approximate locations of the remaining corners are then simply interpolated and a Harris corner finder is employed in the vicinity to refine their position. The toolbox given in [15], which is designed for fisheye and catadioptric central omnidirectional cameras, conversely requires the user to click on all corners of the checkerboard. Indeed, this toolbox makes no assumption about the shape of the lens or the mirror, thus the positions of the remaining corners cannot be interpolated from four points. The positions of the clicked points are also refined by a Harris corner finder. In the light of the above elaboration, it was decided to design an automatic checkerboard extractor which could be easily implemented into any and all of the above mentioned toolboxes. Such an add-on dramatically decreases calibration time and increases user experience, while at the same time preserving high calibration accuracy and the correspondence between the same corners over all test images. In order to be useful, such an extraction algorithm needs to work with images of low resolution cameras (at least down to VGA, still widely in use), high distortion (as introduced by omnidirectional cameras) and blur, stemming from the fact that for catadioptric cameras usually not the whole mirror can be made to lie in focus. For this purpose, the checkerboard extraction algorithm by Vezhnevets [17], although developed for planar cameras, was found to yield a good starting point.

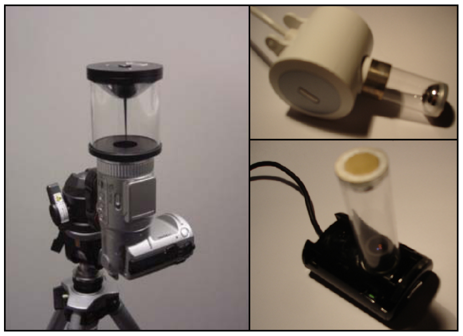
Fig. 1. Left: hyperbolic mirror placed on a video camera. Top right: Philips SPC 300. Bottom right: Philips ToUCam Fun.

This work was supported by European grant FP6-IST-1-045350 Robots@Home$^{®}$

A. Contribution and Outline

The main contribution of this paper is a novel heuristic to detect checkerboards in blurred and highly distorted images. In particular, we show that through this heuristic the detection rate of a standard checkerboard detection algorithm[17] increases from 20% up to 80%, reaching almost 100% using high quality cameras. Furthermore, the code is freely available online, both in its source form and incorporated into our camera calibration toolbox. To our knowledge this is the only such implementation readily available [15].

This document is organized as follows. In Section II, the important steps of an existing corner extraction algorithm [17] are described and its strengths and shortcomings concerning the given task are elaborated. In Section III we propose and discuss several steps for increasing the code's performance. Section IV compares the performance of the improved algorithm against both the performance of the existing implementation and manual selection of corners.

II. THE ALGORITHM BY VEZHNEVETS

OpenCV [18] is an open source computer vision library initially developed by Intel. It features algorithms for many vision applications and is in particular equipped with a checkerboard corner extraction functionality developed by Vladimir Vezhnevets [17]. The function identifies single black checkers of a checkerboard and then tries to merge them back into the original pattern. As a region based method it has the advantage of being much more robust to noise and blur than a line based method would be. What follows is a step-by-step analysis of the important parts of this algorithm. In Section III, we will adapt it to our needs.

A. The Steps of the Algorithm

1) Algorithm Input: Input to the algorithm is an image containing a black-and-white checkerboard of a given size. If a color image is provided, a greyscale conversion is executed thereafter. Then, the algorithm continues with a thresholding step.

2) Adaptive Threshold: Binary thresholding is well suited to separate black from white checkers under most circumstances. The algorithm supports adaptive thresholding, which binarizes the image locally according to a given mask size and method and generally delivers higher level segmentation results for non-uniformly lit images. Two kernel implementations are available: "mean" and "Gaussian". In the original approach "mean" is used, which requires considerably less computational power and is thus well suited for the checkerboard detection from a video stream, where execution time is critical. The checkers in the thresholded black-and-white image tend to be grown together due to blur, noise and/or too coarse sampling. For correct identification, they need to be separated. An erosion step is applied.

3) Erosion: The inclusion of an erosion step (by using a 3x3 "rect" kernel, see Fig. 5 right) is the main ingenious idea behind Vezhnevets' implementation. In this way it is possible to separate the checkerboard at the corners and obtain a set of black quadrangles (four-sided polygons). The contours

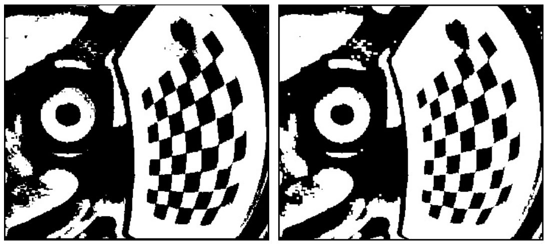
Fig. 2. Left: After adaptive thresholding and one erosion step (run one). Right: After adaptive thresholding and two erosion steps (run two).

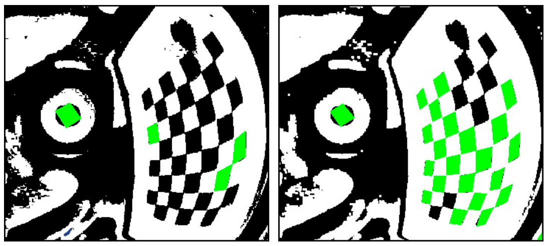
Fig. 3. Left: All found quadrangles after run one. Right: All found quadrangles after run two.

of these quadrangles are then easily found with a binary contour finder. If no pattern is found during the next steps, it can be assumed that the checkers are still grown together. Therefore erosion is gradually increased and the following steps repeated. Fig. 2 illustrates how the checkers shrink and then separate from their neighbors.

4) Quadrangle Generation: A binary contour finder then tries to find closed contours and upon success, tries to fit a quadrangle onto it by gradually approximating a polygon. Notice how after the first erosion run (Fig. 3 left) only two checkers are properly separated and hence only two quads are found. After two erosion steps (Fig. 3 right) the majority of quadrangles, but not all of them, are found. By applying even more erosion steps, the pattern starts to partially dissolve, resulting in non-detection of some (small) checkers.

5) Quadrangle Linking: Quadrangles are then linked according to the following heuristic:

- For every corner of every found quadrangle compute the distance to every corner of every other quadrangle. Store the smallest such distance and the respective corner and quadrangle ID.
- Check whether this distance is smaller than the smallest edge length of both quadrangles. This is intended to make sure, that no quadrangle gets linked to quadrangles too far away.
- If these tests are passed, then the two corners are linked and the extracted corner position is set to the arithmetic mean of their former positions.

The extracted corners finally form a pattern described through their position and neighborhood relation with respect to the other corners.

6) Further Steps: From all erosion runs, the algorithm then selects the corner pattern with the highest number of found corners. No information exchange between different

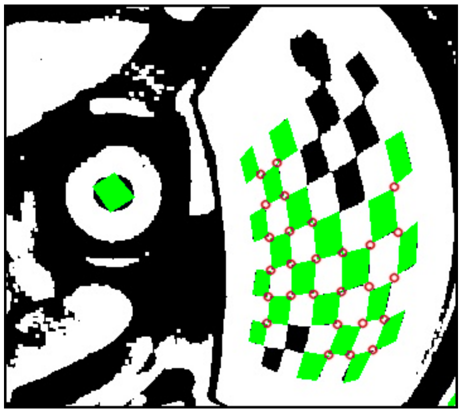
Fig. 4. All found corners after run two.

erosion runs is performed. It is thus assumed that in a single given run every checker is theoretically identifiable. In case the largest pattern features too many corners (i.e. due to an erroneously identified checker because of glare), the ones which result in the smallest convex hull are selected .

B. Limitations

The OpenCV corner finding algorithm was designed for real-time calibration of regular cameras. Focus was laid on fast execution times, hence the use of a "mean" instead of a "Gaussian" mask during the adaptive threshold step. Furthermore, the algorithm only returns a pattern if the complete checkerboard was successfully detected, ignoring the fact that for calibration purposes it is often enough to correctly identify a significant portion of the corners. As will be shown in section IV, the algorithm ceases to function properly with any combination of low resolution (VGA), blurred, and distorted images. Therefore it is of limited use for omnidirectional camera calibration, and thus for implementation into such toolboxes.

III. IMPROVEMENTS TO THE CODE

A. Adaptation of Erosion Kernels

For features of large size in comparison to the kernel used, erosion appears to affect all border pixels uniformly. Upon closer inspection, however, corners tend to get rounded, the exact amount depending on the orientation of the checker and the type of kernel used. This starts to have a significant effect on the checkers if they become of comparable size as the kernels themselves; a condition which is often fulfilled for omni-images taken with VGA resolution. Even though the smallest possible symmetric erosion kernel (a 3x3 maximum filter) was used in the original implementation, some improvements can nonetheless be achieved: the kernel size cannot be made smaller than 3x3, but its shape may be altered. For a symmetric 3x3 kernel it is possible to construct two shapes, namely "cross" and "rect" as depicted in Fig. 5. Alternating between the two has the effect of preserving the aspect ratio of (small) checkers independent of their orientation, i.e. it allows for uniform "shrinking".

0	1	0
1	1	1
0	1	0

1	1	1
1	1	1
1	1	1

Fig. 5. Left: 3x3 "Cross" kernel. Right: 3x3 "Rect" kernel.

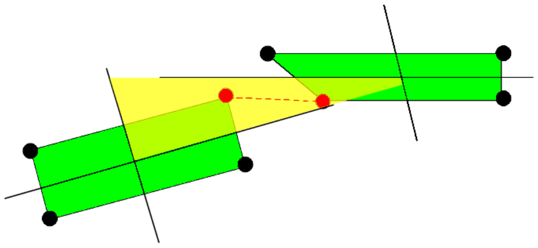
Fig. 6. New heuristic for corner linking: If the two candidate corners (red dots) lie on the same side of each of the four straights (i.e. inside the semitransparent yellow area), they are successfully matched.

B. New Heuristic for Quadrangle Linking

In the original implementation, correctly identified black checkers are connected over their corners according to the heuristic as described in Section II-A.5. It was found to work well for high resolution and mostly undistorted images of checkerboards. For distortions as introduced by omnidirectional cameras, however, not necessarily the closest corner should be matched to a given corner, as Fig. 6 illustrates. Correct corner matching is of utmost importance; mismatches disturb the structure of the extracted pattern and therefore invalidate all further steps. Our proposition for a solution of this issue comes in the form of an enhanced heuristic which can be geometrically verified to work even under severe distortions:

- For every corner of every found quadrangle compute the distance to every corner of every other quadrangle and check whether the distance is shorter than the shortest edge length of both of the two involved quadrangles. If true, accept the two corners as a candidate neighbor pair.
- For each candidate pair, focus on the quadrangles they belong to and draw two straight lines passing through the midsections of the respective quadrangle edges (see Fig. 6).
- If the candidate corner and the source corner are on the same side of every of the four straight lines drawn this way (this corresponds to the yellow shaded area in Fig. 6), then the corners are successfully matched.

C. Adaptive Quadrangle Linking Distance

As mentioned in Section II-A.5, quadrangles only get linked if their corners are less than a certain distance apart. In the original implementation, inaccurately, the shortest edge length of the two involved quadrangles was chosen for this distance limit. If the checkers are large w.r.t erosion, the error introduced is small. But for low resolution

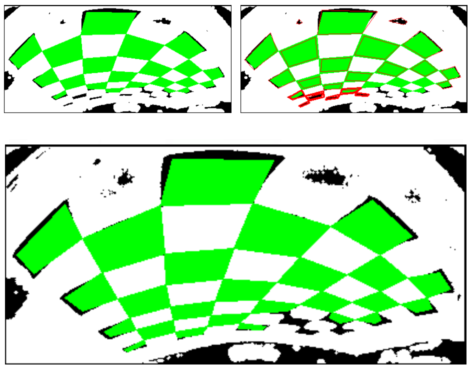

Fig. 7. Visualization of "matching over different dilation runs" procedure. Top: reference pattern (light green). Clearly the bottom checkers have not been identified. Middle: red quadrangles indicate candidate checkers found in another erosion run. Bottom: addition of some of these candidates to the reference pattern (bold red quadrangles).

images, erosion has a large effect on the overall size of the quadrangle, which may result in a drastic reduction of the smallest edge length. Therefore the distance measure was adapted to incorporate the effect of erosion:

$$d_{limit} = shortest\_edge\_length + 2 \cdot erosion, \qquad (1)$$

where the factor two is due to the erosion acting on both quadrangles.

D. Linking of Quadrangles over Multiple Erosion Runs

Through the mirror of an omnidirectional camera, blur is radially unevenly spread: depending on the focal distance of the camera, either points toward the center or toward the border of the image tend to be more blurred. Because of this anisotropy, not all quadrangles are separated during the same erosion run. Some of them may even only start to separate when smaller ones have already completely disappeared. Therefore the problem may be encountered that even though many quadrangles are successfully identified spread over multiple iterations, not all of them appear in a single one. We therefore tried to match patterns of found quadrangles over different erosion runs, by combining partial into complete results. The algorithm was thus expanded as follows: the pattern, where most quadrangles had been found is selected as "reference pattern". In a second (new) part, all previously found quadrangles of all erosion runs are tried to be matched to the border of the above defined reference pattern. Upon successful match, the reference pattern is updated to include the new quadrangle and the whole process is repeated until no more additions are reported. Fig. 7 visualizes this second part in a sequence of images.

E. Adaptation of the Polygonal Approximation Level

As described in Section II-A.4, extracted contours are sent to a polygonal approximator, which tries to fit quadrangles onto them. Depending on how much the approximated polygon is allowed to deviate from the true contour (deviation threshold), due to blur connected checkers are sometimes mistakenly approximated as a single quadrangle, which again disturbs the resulting pattern. Decreasing the deviation threshold leads to the identification of a substantially smaller amount of quadrangles. At the same time, false positives detection is reduced as well. Therefore we decided to restrict the approximation of contours to a conservative level (i.e. select a low deviation threshold) in the first part of the algorithm, practically guaranteeing the extraction of correct quadrangles at the price of the number of found objects. The now smaller reference pattern is then introduced into the new part two of the algorithm (see Section III-D), where the polygonal approximation threshold is again increased.

The idea is then to try matching quadrangles found during the most strongly eroded run to the reference pattern first (i.e. introducing runs in reverse order), as there the chance of separated checkers is highest. Addition of heavily eroded quadrangles to the reference structure decreases corner localization, however. With this adaptation, correct pattern extraction is therefore favored over corner accuracy.

F. Relative Importance

The adaptation of the erosion kernels and especially the introduction of a new linking heuristic were found to be the most important enhancements. They both deal with the changes to the checker pattern as introduced by omnidirectional camera distortions, while at the same time preserving the detection rate of the original implementation for regular images. The other improvements only start having a significant effect on very low resolution and blurred images (see Section IV-B).

IV. TEST IMAGE ANALYSIS

In this section, 6 test image sets containing 10 images each are analyzed. Typical camera-mirror setups of various quality have been considered. The number of found corners per image and the corner localization accuracy is compared between the original OpenCV implementation and our proposed method. First, however, prerequisites for successful corner extraction are discussed.

A. Prerequisites

Corner extraction using both OpenCV and our method is dependent on a black and white checkerboard of any reasonable size (sizes of 5x6 and 6x7 inner corners have been shown to work well), with a white border around it of at least one checker width (see Fig. 8). If you plan on using the algorithm in cases of extreme back light or overhead lighting, consider using a checkerboard with an even wider white border. Additionally, use a camera with as high a resolution as possible, try to minimize overall blur, but especially around small checkers and make sure that none of the checkers touch the border or got occluded.

B. Results

For an overview of the test image sets chosen, refer to Table I. Sets no. 1-3 have been taken with a Sony XCD-SX910 camera (high resolution) combined with a hyperbolic mirror; sets no. 4 and 5 with a Philips ToUCam Fun camera

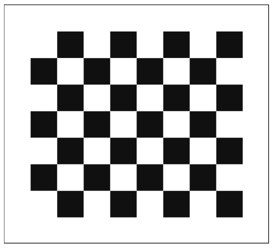

Fig. 8. A 7x6 inner corner checkerboard with a white border of exactly one checker width.

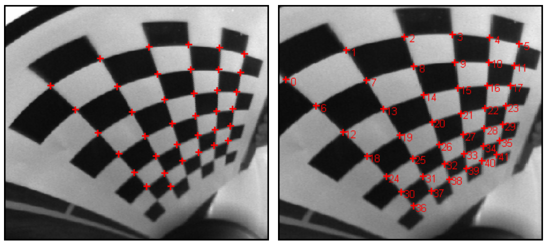

Fig. 9. Calibration images which best reflect the average performance of the algorithm for test set 1. Left: OpenCV. Right: our approach.

TABLE I
TEST IMAGE SETS

Img. set	Resolution	Blur	Brightness	Camera-mirror shape
Set 1	1280x960	no	daylight	hyperbolic, central
Set 2	1280x960	no	reduced iris	hyperbolic, central
Set 3	1280x960	yes	daylight	hyperbolic, central
Set 4	640x480	no	daylight	Christmas ball, non-central
Set 5	640x480	no	daylight	spherical, central
Set 6	640x480	yes	daylight	spherical, central

TABLE IV
TEST IMAGE SET 3

Method	OpenCV	Our method
	Number of found corners	Number of found corners
Mean	11.4 of 30	29.7 of 30
Min	3	28
Max	21	30
	Corner inaccuracy [pxl]	Corner inaccuracy [pxl]
Mean	1.48	0.62
Variance	0.23	0.63

(low resolution, large depth of field, Fig. 1 bottom) combined with a Christmas ball and a spherical mirror respectively; set no. 6 with a Philips SPC 300 camera (low resolution, narrow depth of field, Fig. 1 top) combined with a spherical mirror. For set no. 4, we used a Christmas ball in order to show that our method also works for other concave mirrors.

For sets no. 1, 2, 4, and 5 (no blur) corner inaccuracy is measured with respect to a reference extraction (manual preselection followed by a Harris corner extraction in the selected area). For sets no. 3 and 6 (blur), manual corner selection alone is defined as reference. Images displaying the average number of found corners for test sets no. 1 and 6 are depicted in order to convey a feeling for the relative performance between the two implementations at different test conditions (Fig. 9 and 10).

C. Discussion

The results show that our approach consistently outperforms OpenCV, except on high-resolution and nearly planar images (test set no. 1) where they are on equal footage. Our algorithm notably also works well in conjunction with non-hyperbolic mirrors (test sets no. 4, 5, and 6). Furthermore, corner localization is shown to have an average error of less than one pixel, compared to reference. If not much blur is present in the calibration images, a Harris corner finder as implemented into most toolboxes is able to negate this error.

D. Issues with Our Approach

The following two examples are intended to give the reader an understanding on issues which could arise during

TABLE II
TEST IMAGE SET 1

Method	OpenCV	Our method
	Number of found corners	Number of found corners
Mean	37.2 of 42	42 of 42
Min	18	42
Max	42	42
	Corner inaccuracy [pxl]	Corner inaccuracy [pxl]
Mean	1.46	0.68
Variance	0.13	0.17

TABLE III
TEST IMAGE SET 2

Method	OpenCV	Our method
	Number of found corners	Number of found corners
Mean	37.4 of 42	41.5 of 42
Min	30	37
Max	42	42
	Corner inaccuracy [pxl]	Corner inaccuracy [pxl]
Mean	1.43	0.69
Variance	0.15	0.20

TABLE V
TEST IMAGE SET 4

Method	OpenCV	Our method
	Number of found corners	Number of found corners
Mean	28 of 42	41.7 of 42
Min	14	40
Max	39	42
	Corner inaccuracy [pxl]	Corner inaccuracy [pxl]
Mean	1.77	0.97
Variance	0.55	0.84

TABLE VI
TEST IMAGE SET 5

Method	OpenCV	Our method
	Number of found corners	Number of found corners
Mean	26.8 of 42	41.4 of 42
Min	10	36
Max	36	42
	Corner inaccuracy [pxl]	Corner inaccuracy [pxl]
Mean	1.05	0.79
Variance	0.51	0.31

TABLE VII
TEST IMAGE SET 6

Method	OpenCV	Our method
	Number of found corners	Number of found corners
Mean	8.3 of 42	33.4 of 42
Min	3	23
Max	15	42
	Corner inaccuracy [pxl]	Corner inaccuracy [pxl]
Mean	1.08	1.05
Variance	0.35	0.90

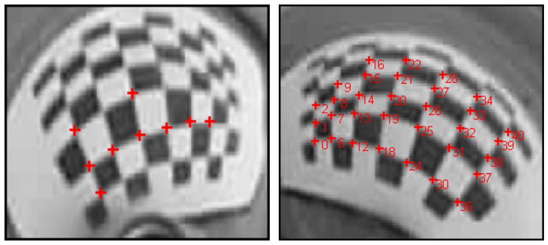
Fig. 10. Calibration images which best reflect the average performance of the algorithm for test set 6. Left: OpenCV. Right: our approach.

the checkerboard pattern extraction.

1) Importance of a Wide Border around the Checkerboard: When taking pictures against bright light sources, the adaptive threshold is disturbed into believing that the white checkerboard border is actually black. We stress the importance of a wide enough white border.

2) Small Checkers in Low Resolution Images: Figure 11 belongs to test image set no. 5. Close inspection of the matching process shows that during one erosion run the bottom right checkers are too small to be recognized as quadrangles; during the next erosion run, however, they are already grown together with their neighbor checker. In such cases, which happen only for very small checkers in low-res images, corner extraction for the involved checkers fails.

V. CONCLUSION

In this paper an existing method for identifying checkerboards on calibration images was analyzed. This method then served as the starting point for the adapted and improved method described in Section III. The enhancements to the code proved to dramatically increase the corner output for low resolution and blurred images, consistently returning 80% and more of the corners present, compared to as low as 20% for the existing method. On higher resolution images, nearly 100% corner recognition was obtained. False positive detection was found to be very low, provided the image acquisition prerequisites from Section IV-A are fulfilled. This shows the strength of the algorithm: it builds on top of and enhances the openCV implementation using the refinements described earlier, and thus works just as well on nondistorted images as the original approach, but consistently outperforms it in low resolution, highly distorted and/or blurred images. We therefore believe it to be well suited for implementation into a wide range of camera calibration toolboxes, which could notably benefit from automatic calibration routines.

A standalone executable of the algorithm was then generated. Via a Matlab based interface it may be easily integrated into any of the available camera calibration toolboxes. The complete source code, the executable and a sample interface are available online [15].

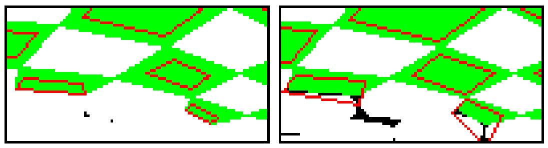
Fig. 11. Bottom checkers of the board are not recognized: During one erosion run they are too small for recognition (left image), during the next, however, already grown together with their neighbor checker (right image).

REFERENCES

[1] B. Micusik, *Two View Geometry of Omnidirectional Cameras.* PhD thesis, Center for Machine Perception, Czech Technical University in Prague, 2004.

[2] R. Benosman and S. B. Kang, editors. Panoramic vision: sensors, theory, and applications. Monographs in computer science. Springer Verlag, New York, 2001.

[3] R.Y. Tsai, An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, pp. 364-374, 1986.

[4] Zhengyou Zhang. A Flexible New Technique for Camera Calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 22, Issue 11, pp.: 1330 - 1334. November 2000.

[5] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[6] Scaramuzza, D., Martinelli, A. and Siegwart, R. A Toolbox for Easy calibrating Omnidirectional Cameras. *In Proc. of IROS'06*, pp. 5695-5701, China, October 2006, Beijing, (2006).

[7] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *IEEE International Conference on Robotics and Automation*, April 2007.

[8] C. Geyer and K. Daniilidis, "Paracatadioptric camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 687–695, may 2002.

[9] 19. Y. Ma, S. Soatto, J. Kosecka, S. Sastry, An invitation to 3D vision, from images to geometric models models, Springer Verlag, ISBN-0-387-00893-4. 2003.

[10] J. Gluckman and S. Nayar, "Ego-motion and omnidirectional cameras," in *6th International Conference on Computer Vision*, pp. 999–1005, 1998.

[11] S. Kang, "Catadioptric self-calibration," in *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 201–207, 2000.

[12] Micusik, B. and Pajdla, T. Estimation of omnidirectional camera model from epipolar geometry. *In Proc. of CVPR*. ISBN 0-7695-1900-8, US, June 2003, IEEE Computer Society, Madison, (2003).

[13] S. Bougnoux, "From projective to euclidean space under any practical situation, a criticism of self-calibration," in *6th International Conference on Computer Vision*, pp. 790–796, 1998.

[14] Bouguet, J.-Y. Camera Calibration Toolbox for Matlab: $http://www.vision.caltech.edu/bouguetj/calib\_doc$

[15] Scaramuzza, D. Omnidirectional Camera Calibration Toolbox for Matlab: Google for "*ocamcalib*", or go directly to $http://asl.epfl.ch/\tilde{}scaramuz/research/Davide\_Scaramuzza\_files/Research/OcamCalib\_Tutorial.htm$

[16] Mei, C. Omnidirectional Camera Calibration Toolbox for Matlab: $http://www.robots.ox.ac.uk/\tilde{}cmei/Toolbox.html$

[17] Vezhnevets, V. OpenCV Calibration Object Detection: $http://graphics.cs.msu.ru/en/research/calibration/opencv.html$

[18] Intel Corporation. Open Source Computer Vision Library: $http://www.intel.com/technology/computing/opencv$