Journal of Electronic Imaging

JElectronicImaging.org

Automatic checkerboard detection for camera calibration using self-correlation

Yizhen Yan
Peng Yang
Lei Yan
Jie Wan
Yanbiao Sun
Kevin Tansey
Anand Asundi
Hongying Zhao

Yizhen Yan, Peng Yang, Lei Yan, Jie Wan, Yanbiao Sun, Kevin Tansey, Anand Asundi, Hongying Zhao, "Automatic checkerboard detection for camera calibration using self-correlation," *J. Electron. Imaging* **27**(3), 033014 (2018), doi: 10.1117/1.JEI.27.3.033014.

Journal of Electronic Imaging 27(3), 033014 (May/Jun 2018)

Automatic checkerboard detection for camera calibration using self-correlation

Yizhen Yan,[a] Peng Yang,[a] Lei Yan,[a] Jie Wan,[a] Yanbiao Sun,[b] Kevin Tansey,[c] Anand Asundi,[d] and Hongying Zhao[a,*]
[a]Peking University, School of Earth and Space Sciences, Beijing Key Laboratory of Spatial Information Integration and Its Applications, Beijing, China
[b]University College London, Department of Civil, Environmental and Geomatic Engineering, London, United Kingdom
[c]University of Leicester, Leicester Institute for Space and Earth Observation and Centre for Landscape and Climate Research, School of Geography, Geology and the Environment, Leicester, United Kingdom
[d]Nanyang Technological University, School of Mechanical and Aerospace Engineering, Centre for Optical and Laser Engineering, Singapore, Singapore

Abstract. The checkerboard is a frequently used pattern in camera calibration, an essential process to get intrinsic parameters for more accurate information from images. An automatic checkerboard detection method that can detect multiple checkerboards in a single image is proposed. It contains a corner extraction approach using self-correlation and a structure recovery solution using constraints related to adjacent corners and checkerboard block edges. The method utilizes the central symmetric feature of the checkerboard crossings as well as the spatial relationship of neighboring checkerboard corners and the grayscale distribution of their neighboring pixels. Five public datasets are used in the experiments to evaluate the method. Results show high detection rates and a short average runtime of the proposed method. In addition, the camera calibration accuracy also presents the effectiveness of the proposed detection method with reprojected pixel errors smaller than 0.5 pixels.
 [DOI: 10.1117/1.JEI.27.3.033014]

Keywords: corner detection; correlation; structure recovery; camera calibration; photogrammetry.

Paper 171075 received Dec. 11, 2017; accepted for publication Apr. 20, 2018; published online May 15, 2018.

1 Introduction

Camera calibration is an important step to get the intrinsic parameters of cameras for better sensing the world, such as three-dimensional (3-D) reconstruction and thematic information extraction from digital images. Many camera calibration techniques use special patterns to automatically compute the point correspondences, such as Tsai's method[1] and Zhang's method.[2] Among those techniques, the most common pattern is the checkerboard, due to its robustness, low cost, and simple structure. Therefore, the detection of checkerboards becomes a key factor that determines the accuracy and the processing speed in many calibration methods.

When carrying out calibration, corner extraction and structure construction are both necessary, and have been studied by many researchers. Two common camera calibration tools, the Camera Calibration Toolbox for MATLAB[3] and the OpenCV library,[4] include these two steps, but the former needs manual operation to locate four external corners of a checkerboard to form a rectangle, and the latter requires the number of corners in a row and a column, resulting in a complicated and semiautomatic calibration procedure. Other approaches include de la Escalera and Armingol,[5] who proposed an automatic chessboard detection method using the Hough transform to detect straight chessboard edge lines then extracting the corners. Chu et al.[6] detected the chessboard corners using a round template by analyzing the gray distribution in it and computing the centroids of redundant points. These two methods make the calibration task less tedious, which, however, may fail in processing images with strong geometric distortions, such as fisheye camera data. Bennett and Lasenby[7] then developed a new chessboard feature detector that is robust against some poor conditions like noise, whereas their method may detect redundant corners that are outside the checkerboard and do not recover the whole structure of the checkerboards for calibration. Their subsequent work[8] deals with the structure, and obtains good result in finding structures of projected checkerboards with surface distortion, but the method is restricted by its prerequisites and may fail when the angles of the checkerboard squares are far away from $\pi/2$. Placht et al.[9] used an edge graph generation idea to accurately refine the checkerboard corners and the algorithm can process images at extreme poses or with high distortions, and it is also valid for low-resolution images. Fuersattel et al.[10] then made an improvement that uses graphs to represent checkerboards and can find partly occluded checkerboard pattern by graph matching, resulting in a higher detection rate and a shorter runtime. Bok et al.[11] employed circular boundaries of corner candidates to extract the checkerboard ones and conduct the indexing, which also performs well in processing distorted or noisy images. However, the three methods detect only a single checkerboard in one image, which means the camera or the checkerboard needs to be relocated for several times to ensure robust and accurate

*Address all correspondence to: Hongying Zhao, E-mail: zhaohy@pku.edu.cn

1017-9909/2018/$25.00 © 2018 SPIE and IS&T

calibration results. Geiger et al.[12] developed a fully automatic calibration method using a single shot by calculating a corner likelihood at each pixel in the image with two different corner prototypes and detecting all checkerboards in it. Their method is robust in many difficult scenes, such as blurring and distortion, but it can be quite time-consuming when processing large-scale images. Therefore, more accurate and faster automatic checkerboard detection methods still need to be developed.

In this work, we make a trade-off between robustness and runtime for checkerboard detection and calibration with a single shot, and try to improve both. The key idea of our approach is the self-correlation computing of the corner neighborhood, which utilizes the central symmetry property of the checkerboard corner areas. In our solution, potential corners are roughly detected by a Harris detector[13] first, then a square area centred at each pixel in the corner neighborhood is selected and rotated by π; the correlation between the square itself and the rotated one, the aforementioned self-correlation, is calculated afterward to extract more accurate corner candidates, as the correlation values for the checkerboard corners can be quite large, whereas for other points it can be much smaller. Our approach also contains a checkerboard structure recovery solution using gradient, orientation, and distance constraints of the checkerboard edges, which is able to detect multiple checkerboards in a single image through structure expansion and grouping. We evaluate the proposed approach with six datasets including five online public datasets[9,10,14] and compare it with the state-of-the art methods. The main contributions of the proposed method are: (1) detecting checkerboard corners using self-correlation, (2) recovering the structure of a checkerboard by taking the advantages of the gray distribution in the checkerboard square as well as angle and distance constraints, and (3) gaining robust results against distortion with a high detection rate.

This work is organized as follows: Sec. 2 describes the specific procedures to extract checkerboard corners and the refinement of corners. Section 3 gives a detailed introduction to the checkerboard structure recovery and grouping method. The experiments and results of the proposed approach are discussed in Sec. 4. Finally, we draw conclusions based on the evaluation in Sec. 5.

2 Checkerboard Corner Detection

There have been several commonly used feature detectors, such as Harris,[13] Susan,[15] SIFT,[16,17] and FAST,[18] that can extract the checkerboard crossings, but corners outside the checkerboards may also be extracted by them, leading to more outliers. Addressing this problem, our algorithm utilizes the inner feature of the checkerboard to precisely locate its corners and discarding the outliers (as shown in Fig. 1): (1) rough detection by Harris, (2) corner filtering using gray distribution to accelerate the later procedure, (3) self-correlation computation at each pixel in the corner neighborhoods, (4) nonmaximum suppression on the self-correlation map, (5) outlier discarding through some constraints, and (6) refinement. The details of the above steps are described in the following sections, with steps (1) and (2) in Sec. 2.1, steps (3) and (4) in Sec. 2.2, and steps (5) and (6) in Sec. 2.3.

2.1 *Rough Detection and Filtering*

In this part, the Harris detector is used first to roughly detect all potential corners in an image as shown in Fig. 1(b). There are two important parameters in the Harris corner detection that can be changed to obtain better results: the minimum accepted quality MQ and the filter size FS. The MQ parameter is a scalar specifying the minimum accepted quality of corners, and larger values of it can be used to remove erroneous corners. The FS parameter is an odd integer specifying a Gaussian filter that is used to smooth the gradient of the image in the Harris corner detection, and the standard deviation (STD) of the filter is FS/3. The value of the FS parameter can be determined by the image size and resolution, and usually smaller values can be applied on lower resolution images.

However, the Harris detection step will lead to plenty of redundant corners that are not exactly the checkerboard corners when processing high resolution or noisy images, as the Harris algorithm is not designed for checkerboard corner only. Therefore, a filtering step utilizing the gray distribution of corner areas is applied. Given a square area surrounding the checkerboard corner, the square should satisfy the two conditions as shown in Fig. 2(a): (1) containing two bright and two dark parts with a cross shape, and (2) the boundary of the square area also consists of four segments in black or white. The filtering algorithm is designed based on the two constraints. First, a $n \times n$ square centered at each corner is selected as a subimage and transformed to a binary one using its mean pixel value as a threshold. Then, the boundary of each square is scanned to find the endpoints of the segments in it, and the corners without four endpoints in the boundary are removed. Note that, to make it more robust against noise, three squares (with side length $n = 11$, 23, and 35 pixels, respectively) of each corner are selected for the boundary check, and as long as one fits the condition, the corner will not be removed. We also use the four endpoints to construct two cross lines to separate the square into four parts and assign zero and one values to the opposite parts [Fig. 2(b)], then calculate the correlation between the original square and the newly modified one with Eq. (1), so as to check whether the corner meets the first condition. The corners with low correlation are discarded [like the bad example in Fig. 2(b)]. Note that this filter step is optional in the whole procedure, and it aims to shorten the runtime by discarding many outliers beforehand. For low-resolution images, this step is not needed as the processing time for them will not be long, and in another aspect, the step may lead to correct corners being discarded because the edge of the square in low-resolution images may be too short.

$$\mathrm{Corr}(A,B)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{ij}-\overline{A})(B_{ij}-\overline{B})}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{ij}-\overline{A})^2\cdot\sum_{i=1}^{n}\sum_{j=1}^{n}(B_{ij}-\overline{B})^2}}, \tag{1}$$

where A and B are two $n \times n$ matrices, $\overline{A}$ and $\overline{B}$ denote the mean values of all elements in A and B, respectively.

2.2 *Corner Extraction Using Self-Correlation and Nonmaximum Suppression*

The central symmetric property of the checkerboard corner neighborhoods can be well utilized to extract the real

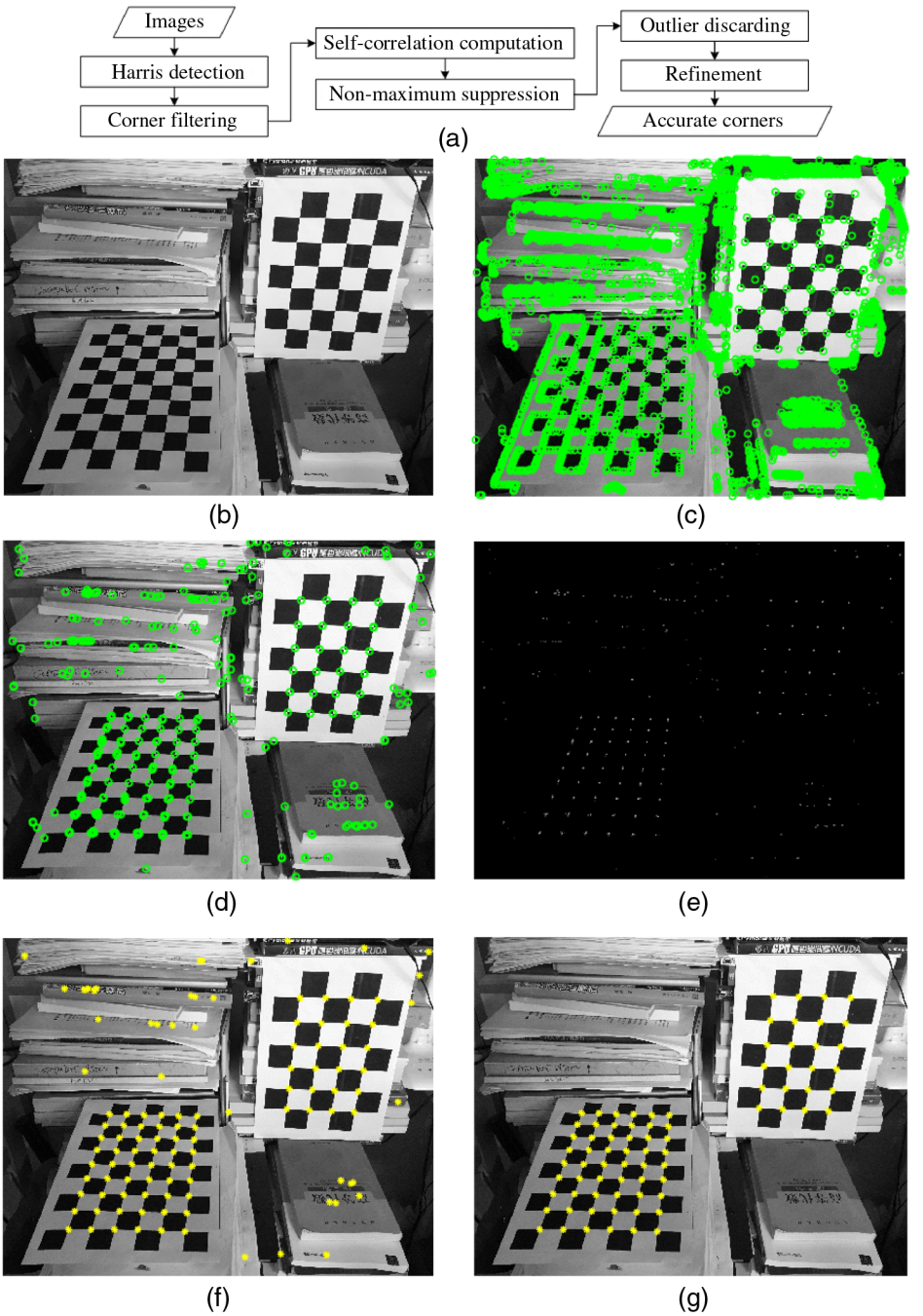

Fig. 1 Processing steps of checkerboard corner detection. (a) Flowchart, (b) original gray image, (c) corners detected by Harris, (d) corners after filtering, (e) self-correlation map, pixels with larger gray values represent higher checkerboard corner likelihood, (f) nonmaximum suppression on the self-correlation map, and (g) result after outlier discarding and refinement.

checkerboard corners from the mass potential corners as shown in Fig. 3(a). Even if the image is taken under strong distortion, the small local checkerboard corner areas still keep their central symmetric property. In this work, we exploit this feature by computing the self-correlation of corner neighborhoods as shown in Figs. 3(b) and 3(c). For each candidate corner detected by the former steps, a $m \times m$ square centered at each pixel in the corner neighborhood (a $k \times k$ area) is selected and rotated by π, then the correlation between the square and its rotated one is calculated using Eq. (1). To better distinguish the checkerboard corners, the self-correlation is magnified by an exponential function as Eq. (2), with a larger value representing a higher corner likelihood. Finally, we can obtain the self-correlation map by assigning zero to other pixels outside the corner neighborhoods and easily extract corners through a nonmaximum suppression step. The key idea of nonmaximum suppression is that pixels are set to zero if they are not part of the local maxima. Therefore, the real corners can be extracted from corner neighborhoods using such algorithms, as described in Refs. 19 and 20.

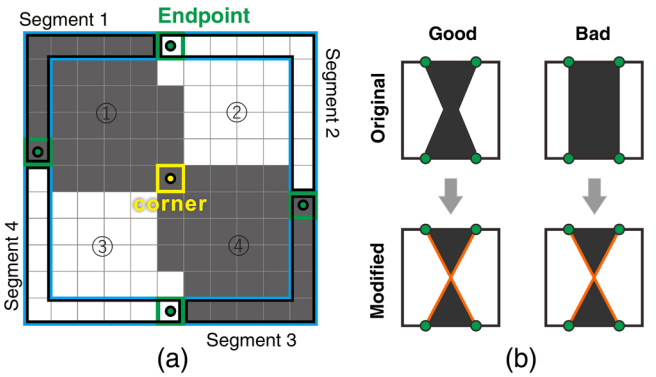

Fig. 2 Schematic diagrams of the corner neighborhood area. (a) Square centered at the corner, with two bright (② and ③) and two dark (① and ④) parts in it; the boundary is marked by blue lines and contains four black and white segments; the endpoints are actually where the signal changes and where the segments separate. (b) The examples of different squares; the endpoints (green circles) are detected the same in the original squares of each example, so the orange cross lines constructed by the endpoints in the modified squares are also the same; but only the original square in the good example is like the real checkerboard corner neighborhood, and its correlation with the modified one is strong, while that for the bad example is not.

$$\mathrm{Corr}_m = \exp^{(\mathrm{Corr}/0.8-1)}, \tag{2}$$

where 0.8 is an empirical value.

Through the above steps, corner clusters detected by Harris, that surround the real checkerboard corner neighborhoods, can be processed to a single corner point and the position can be found more accurately as shown in Fig. 4. There are two details here that should be noted.

(1) Before the self-correlation computation, the STD of each selected square is calculated to remove the obviously wrong corner candidate to improve the efficiency of the algorithm.

(2) The length m is equal to $2k$ to improve robustness against geometric distortion and poor lighting condition like over-exposure, as the large circle area shows in Fig. 4.

In this part, two parameters should be noted: the length k of the square neighborhood in the self-correlation calculation denoted by CS, and the radius of region considered in non-maximum suppression denoted by NMS. Larger values of the two parameters mean larger neighborhoods of corners will be considered and processed. The effect of the values can be obvious when processing images with different resolutions. Generally, smaller values should be set in lower resolution images.

2.3 *Outlier Discarding and Refinement*

After the self-correlation step, there are still a few outliers, as the previous steps are designed to detect single corners, including points inside and outside the checkerboard. To eliminate the remaining outliers after the preprocessing, we first calculate the distance between each two extracted corners and find the closest 12 ones for each corner. Let $\mathbf{C}$ denote the set of the corners, i and j be the indices of two corners, then $\mathbf{CN}(i)$ and $\mathbf{CN}(j)$ represent the 12 closest corner index vectors corresponding to corners $\mathbf{C}(i)$ and $\mathbf{C}(j)$, respectively. If $i \in \mathbf{CN}(j)$ and $j \in \mathbf{CN}(i)$, $\mathbf{C}(i)$ and $\mathbf{C}(j)$ are regarded as neighbors and their correlation $\mathrm{Corr}(i,j)$ will be calculated using their square neighborhoods with Eq. (1), with the size of the square set by Eq. (3). Then, all the correlation values between a corner and its neighbors are used to remove outliers with Eq. (4). Only if all the absolute correlation values are too small (less than the threshold), will the corner candidate be discarded.

$$\mathrm{CorrRadius}(i,j) = \min\left[\max\left(\frac{d_{ij}}{4}, 5\right), \frac{d_{ij}}{2}\right], \tag{3}$$

where CorrRadius denotes the half length of the square edge and d_{ij} denotes the distance between $\mathbf{C}(i)$ and $\mathbf{C}(j)$.

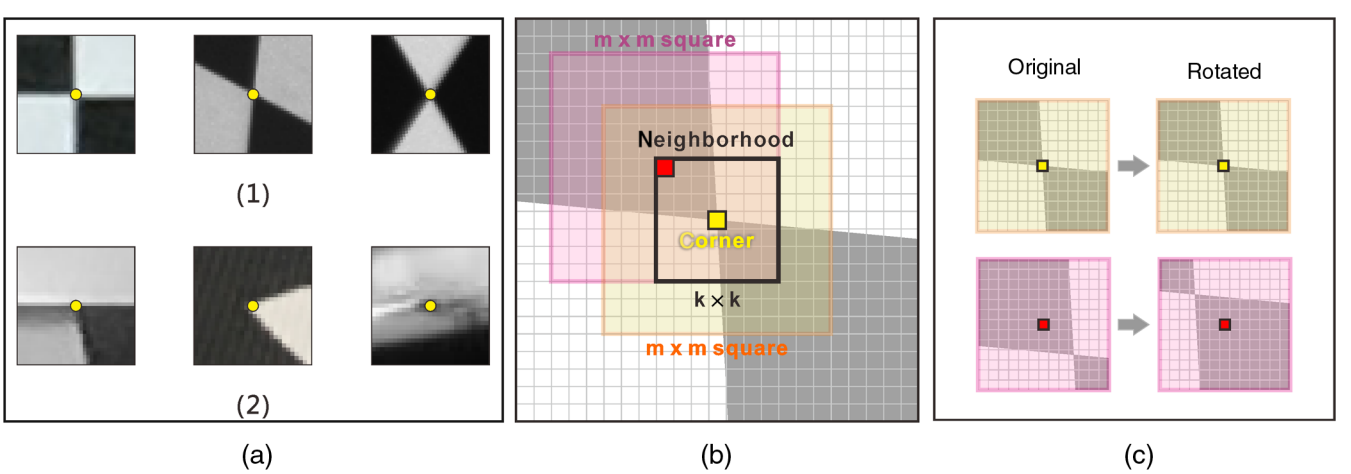

Fig. 3 The central symmetric property of the checkerboard corner neighborhood and its application in corner detection. (a) Some examples of checkerboard corners (1) and other corners (2). (b) The schematic of the square areas processed in correlation calculation. (c) The original and rotated squares of two pixels in the corner neighborhood; the corner is located in the yellow pixel so the square of it is more central symmetric than the farther red pixel, leading to a higher correlation between the original and the rotated square.

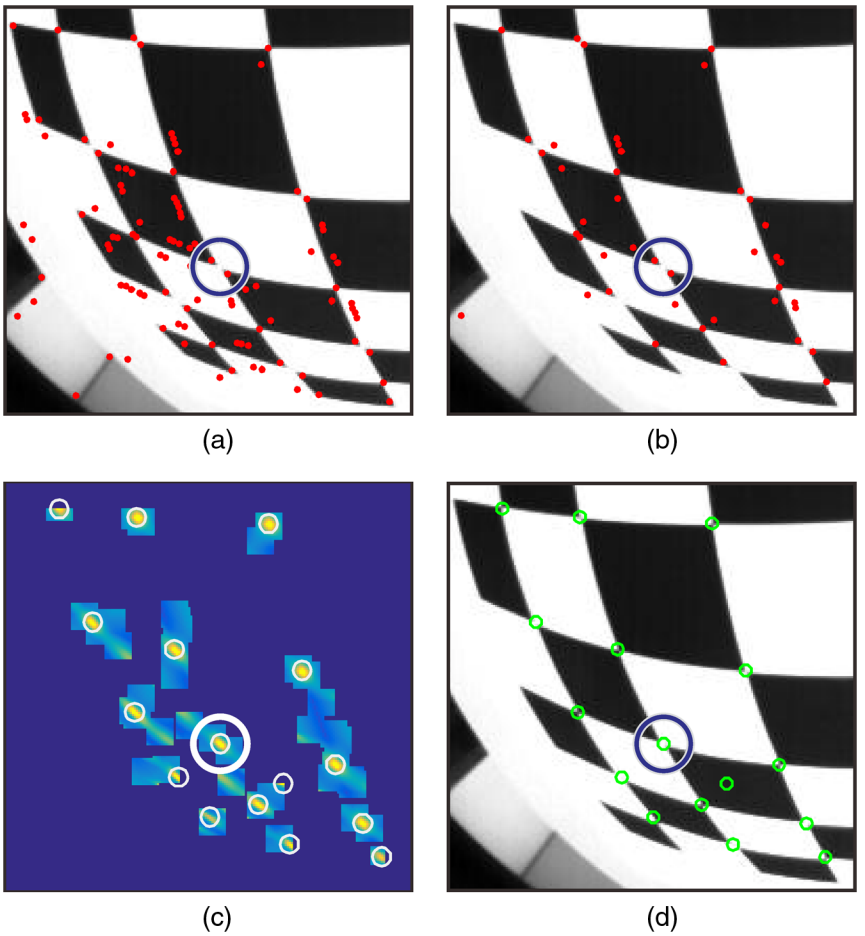

Fig. 4 Self-correlation computation and nonmaximum suppression steps. (a) Candidate corners detected by Harris. (b) Corners after filtering. Cluster corner points surround the checkerboard corner positions, some of which even remain after the filter step. (c) The self-correlation map, where brighter color represents higher self-correlation of the pixels (as the ones in the small white circles show). (d) Corners extracted from the self-correlation map using nonmaximum suppression. After the steps, the corner clusters are processed into single corners with more accurate positions, such as the one in the large circle.

$$\text{remove}(i) = \left(\sum_j \{|\text{Corr}[\mathbf{C}(i), \mathbf{C}(j)]| > \tau\}\right) < 1$$
$$i \in \mathbf{CN}(j)\ \&\ j \in \mathbf{CN}(i), \tag{4}$$

where Corr is the correlation operator, $\text{remove}(i) = 1$ denotes that corner $\mathbf{C}(i)$ will be removed, and τ is a threshold that can be set as 0.5 empirically.

Many researchers[9,21,22] have proposed different corner position refinement algorithms to obtain a higher subpixel accuracy of calibration. The method proposed by Geiger et al.[12] uses both corner position and edge orientation information to do the refinement, leading to a good subpixel result. Therefore in this work, the same refinement method is adopted.

First, two edge orientation vectors can be refined with Eq. (5). As the edge orientations $\mathbf{E}_i$ are almost orthogonal to the image gradients on the edges, the deviation of their normal $\mathbf{E}_i'$ with respect to the gradients $\mathbf{g}_{\mathbf{np}}$ should be as small as possible.

$$\mathbf{E}_i = \arg\min_{\mathbf{E}_i'} \sum_{\mathbf{np}\in\mathbf{N}_i} (\mathbf{g}_{\mathbf{np}}^T \mathbf{E}_i')^2 \quad \text{s.t. } \mathbf{E}_i'^T \mathbf{E}_i' = 1, \tag{5}$$

where $\mathbf{np}$ denotes the neighbor pixel in the corner neighborhood, and $\mathbf{N}_i$ represents the set of neighbor pixels in the buffer area of edges around the corner.

Then, the neighbor pixels along the two refined edge orientations within the corner neighborhood can be found. Given such a neighbor pixel $\mathbf{np}$ of corner $\mathbf{C}_p$, the image gradient of it should be approximately orthogonal to $\mathbf{np} - \mathbf{C}_p$. Therefore, the corner positions can be optimized as

$$\mathbf{C}_p = \arg\min_{\mathbf{C}_p'} \sum_{\mathbf{np}\in\mathbf{N}_c} [\mathbf{g}_{\mathbf{np}}^T (\mathbf{np} - \mathbf{C}_p')]^2, \tag{6}$$

where $\mathbf{N}_c$ represents the set of neighbor pixels along the refined edge orientations within the corner neighborhood.

All the above are the steps of checkerboard corner detection. Table 1 summarizes four key parameters in Sec. 2. Note that, though the values of the four parameters can be adapted to more situations, their empirical values in the experiments (Sec. 4.2) can be used in most cases, so little manual work is needed.

Table 1 Four important parameters in checkerboard corner detection.

Name	Explanation	Step
MQ	A scalar specifying the minimum accepted quality of corners	Harris corner detection
FS	An odd integer specifying a Gaussian filter that is used to smooth the gradient of the image	Harris corner detection
CS	The size k of the square corner neighborhood in the self-correlation calculation	Self-correlation computation
NMS	The radius of region considered in nonmaximum suppression	Nonmaximum suppression

3 Checkerboard Structure Recovery

The structure of the checkerboard can be constructed according to the detected corners. Geiger's method uses an energy function to recover the structure,[12] which is robust against distortion but very time-consuming in dealing with high-resolution images. Bennett and Lasenby's approach[8] takes advantage of the checkerboard crossings' orientation to obtain the structure, and it does well under deformation and noise. But it has some prerequisites and restrictions that may lead to failure when the angles and the edges of the checkerboard squares differ greatly. We recover the structures of checkerboards using constraints related to the correlation of the corner neighborhoods, the gray distribution in a black or white checkerboard square as well as the checkerboard crossings' orientation, which makes it more robust against strong distortion and extreme pose. The detailed steps are shown in Fig. 5: (1) find four neighbor corners from all candidate corners (gray symbols) that reliably locate at the four vertexes of a white or black block in the checkerboard as the initial points [the four red symbols in Fig. 5(a)], (2) expand the quadrangle formed by the four initial corners in the top, right, bottom, and left directions iteratively, until no appropriate corners remain, and label the corners in the expanded structure as belonging to one checkerboard, (3) repeat the two previous steps to recover all checkerboards, until no appropriate initial corners found, and (4) check whether the checkerboard structures overlap, and retain the one with most corners among the overlapping ones. The details of the steps are described in the following sections.

3.1 *Finding Initial Corners*

The selection of the four initial corners is crucial in the whole structure recovery procedure, thus we use strict constraints to find reliable initial corners. Given a random corner as a seed, the other three initial corners are found in the neighbors of the seed corner, and the constraints are related to the distances, angles, and correlation between the seed and its neighbors, as well as the gray distribution inside the quadrangle formed by the initial points. Here, the angle of each two corners (Fig. 6) can be computed as

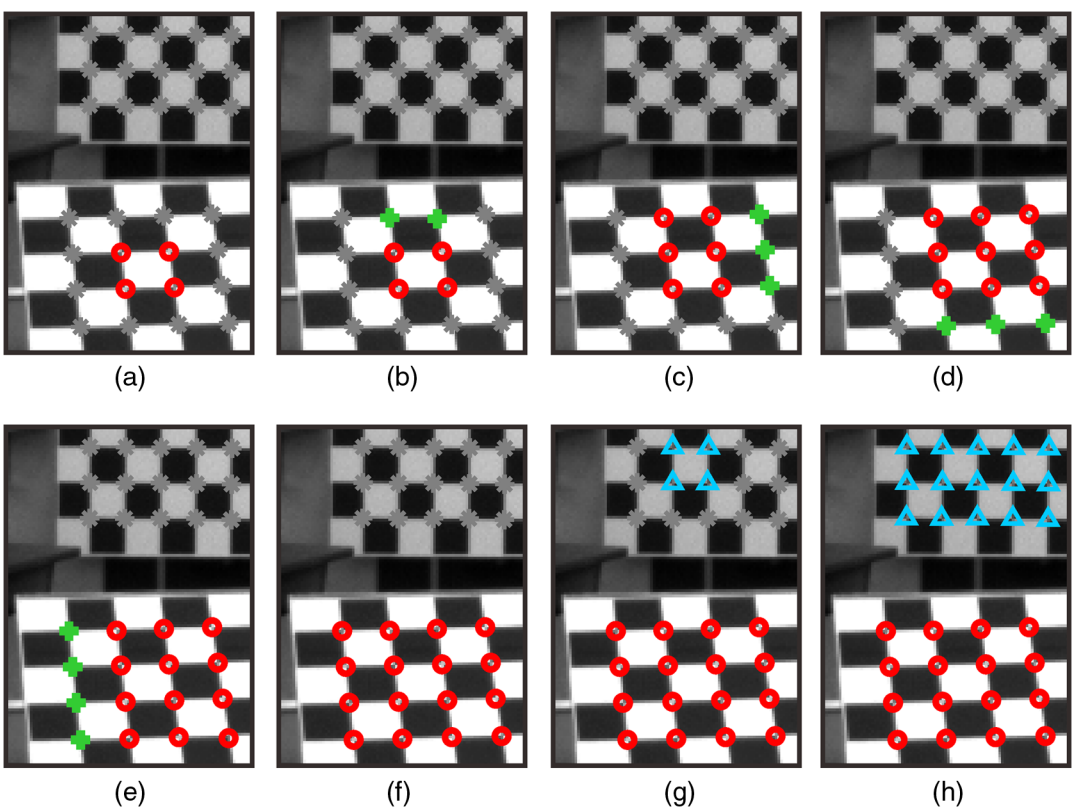

Fig. 5 Steps of checkerboard structure recovery. (a) Four initial corners, (b) expanding in the top direction, (c) expanding in the right direction, (d) expanding in the bottom direction, (e) expanding in the left direction, (f) expansion completed, (g) four new initial corners found, and (h) structure of all checkerboards recovered.

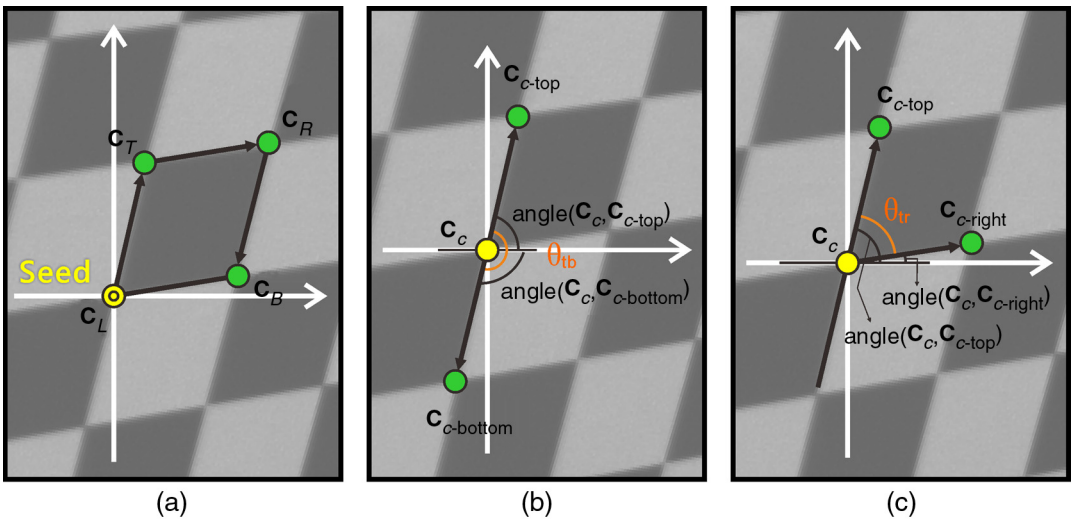

Fig. 6 Schematic diagrams in finding initial corners. (a) A seed is first detected as the left-bottom initial corner, then the top-left, right-top, and bottom right corners are found in turn. (b) and (c) A number of parameters that will be used in the detection of initial corners.

$$\text{angle}[\mathbf{C}(i),\mathbf{C}(j)] = \arccos\left[\frac{\langle \mathbf{C}(i)-\mathbf{C}(j),\mathbf{e}\rangle}{\|\mathbf{C}(i)-\mathbf{C}(j)\|_2}\right], \tag{7}$$

where $\mathbf{e}$ is a unit vector along the horizontal axis direction.

An eligible seed is first detected as the left-bottom initial corner ($\mathbf{C}_L$), then the top-left initial corner ($\mathbf{C}_T$), the right-top corner ($\mathbf{C}_R$), and the bottom-right one ($\mathbf{C}_B$) are selected one by one as shown in Fig. 6(a). The details are shown in the following. If no suitable initial corners are found, the former seed will be labeled as a useless one, and a new random corner among all the other corner candidates will be given as a new seed to continue the process.

(1) In the four corners of a white or black block of the checkerboard, the correlation between the diagonal corners should be strongly positive, whereas the correlation between two corners on the same edge of the block should be strongly negative. Therefore, for the initial corners there is

$$\begin{aligned}&\text{Corr}(\mathbf{C}_L,\mathbf{C}_T),\text{Corr}(\mathbf{C}_T,\mathbf{C}_R),\text{Corr}(\mathbf{C}_R,\mathbf{C}_B),\\&\text{Corr}(\mathbf{C}_B,\mathbf{C}_L) < -\tau_{\text{corr}},\end{aligned} \tag{8}$$

where τ_{corr} denotes a correlation threshold.

(2) When trying to find the top and bottom corners from the four closest neighbors of a corner ($\mathbf{C}_c$), the two with the smallest angles to the y-axis are regarded as the top ($\mathbf{C}_{c\text{-top}}$) and bottom ($\mathbf{C}_{c\text{-bottom}}$) neighbors of $\mathbf{C}_c$, respectively, and the angle (θ_{tb}) between edge $\mathbf{C}_c-\mathbf{C}_{c\text{-top}}$ and edge $\mathbf{C}_c-\mathbf{C}_{c\text{-bottom}}$ should be close to π shown in Fig. 6(b)

$$\begin{aligned}&-\tau_{\text{angle1}} \le |\text{angle}(\mathbf{C}_c,\mathbf{C}_{c\text{-top}})| + |\text{angle}(\mathbf{C}_c,\mathbf{C}_{c\text{-bottom}})|\\&-\pi \le \tau_{\text{angle1}},\end{aligned} \tag{9}$$

where τ_{angle1} denotes an angle threshold.

(3) When trying to find the right corner from the eight closest neighbors of a corner ($\mathbf{C}_c$), the angle (θ_{tr}) between edge $\mathbf{C}_c-\mathbf{C}_{c\text{-top}}$ and edge $\mathbf{C}_c-\mathbf{C}_{c\text{-right}}$ should not be too small or too large as shown in Fig. 6(c)

$$\begin{aligned}&-\tau_{\text{angle2}} \le |\text{angle}(\mathbf{C}_c,\mathbf{C}_{c\text{-top}}) - \text{angle}(\mathbf{C}_c,\mathbf{C}_{c\text{-right}})|\\&\le \pi-\tau_{\text{angle2}},\end{aligned} \tag{10}$$

where τ_{angle2} denotes an angle threshold.

(4) In a single black or white block of the checkerboard, the gray distribution should be even. Therefore, the STD inside the quadrangle formed by the initial points $qd_{(\mathbf{C}_L,\mathbf{C}_T,\mathbf{C}_R,\mathbf{C}_B)}$ should be small enough, and in case of strong distortion, the quadrangle for STD computation is shrunken into a smaller size one by removing the marginal area

$$\begin{aligned}&\text{STD}[qd_{m(\mathbf{C}_L,\mathbf{C}_T,\mathbf{C}_R,\mathbf{C}_B)}] < \tau_{\text{std}},\\&\text{margin} = \max(1, 0.1L_{\text{edge}}),\end{aligned} \tag{11}$$

where τ_{std} denotes a STD threshold, $qd_{m(\mathbf{C}_L,\mathbf{C}_T,\mathbf{C}_R,\mathbf{C}_B)}$ denotes the shrunken quadrangle, and L_{edge} denotes the mean length (pixel) of the four edges in the $qd_{(\mathbf{C}_L,\mathbf{C}_T,\mathbf{C}_R,\mathbf{C}_B)}$, respectively.

Algorithm 1 lists the steps to find the initial corners with these constraints in detail.

3.2 *Structure Expansion and Checkerboard Grouping*

With the four initial corners found, the structure will be expanded in the top, right, bottom, and left directions iteratively by each corner on the structure edge. An angle constraint between neighbor corners is also used in the expansion process but less strict than that used for the detection of the initial corners.

When expanding in a direction, the angle θ_{bn} between the newly found corner and the border corner should be close to the angle θ_{ob} between the border corner and its nearest neighbor corner in the opposite direction as shown in Fig. 7.

Algorithm 1 finding initial corners

Input data: $\mathbf{S}_c \leftarrow$ Set of corners

Result: $\mathbf{C}_L$, $\mathbf{C}_T$, $\mathbf{C}_R$, $\mathbf{C}_B$

While true do

Initialize parameters: τ_{corr}, τ_{std}, $\Delta\tau_{\text{corr}}$, $\mathbf{C}_L \leftarrow$random($\mathbf{S}_c$);

While $\tau_{\text{corr}} > 0.7$ do

($\mathbf{C}_{L-\text{top}}$,$\mathbf{C}_{L-\text{bottom}}$)$\leftarrow$FindNeighborsClosestTo Y Axis[Neighbor($\mathbf{C}_L$)]

$\mathbf{C}_T \leftarrow$FindTopCorner($\mathbf{C}_{L-\text{top}}$,$\mathbf{C}_{L-\text{bottom}}$); if empty($\mathbf{C}_T$) then go to **P1**

$\mathbf{C}_R \leftarrow$FindRightCorner[Neighbor($\mathbf{C}_T$)]; if empty($\mathbf{C}_R$) then go to **P1**

$\mathbf{C}_B \leftarrow$FindBottomCorner[Neighbor($\mathbf{C}_R$)]; if empty($\mathbf{C}_B$) then go to **P1**

If $\mathbf{C}_L \notin$ Neighbor($\mathbf{C}_B$) or Corr($\mathbf{C}_L, \mathbf{C}_B$) $\geq -\tau_{\text{corr}}$ then go to **P1** else go to **P2**

P1: decrease τ_{corr} by $\Delta\tau_{\text{corr}}$;

end

P2: std[$qd_{m(\mathbf{C}_L,\mathbf{C}_T,\mathbf{C}_R,\mathbf{C}_B)}$]$\leftarrow$CalculateStandardDeviation [$qd_{m(\mathbf{C}_L,\mathbf{C}_T,\mathbf{C}_R,\mathbf{C}_B)}$]

If std[$qd_{m(\mathbf{C}_L,\mathbf{C}_T,\mathbf{C}_R,\mathbf{C}_B)}$] $< \tau_{\text{std}}$ then break;

end

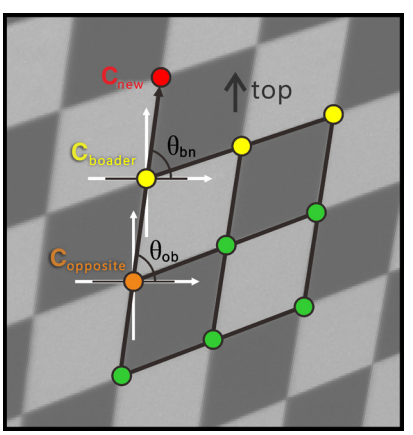

Fig. 7 Structure expansion in one direction.

In addition, the gradient on the boundary of a white or black checkerboard block is calculated and utilized for a more robust and precise structure recovery. On the edges of the white and black blocks in a checkerboard, the absolute gradient values of the pixels should be very similar, thus the mean value of the gradients on the edge between a newly found corner and a border corner should be close to that of the four initial corners. Corners satisfying both the angle and gradient constraints are selected to expand the structure.

In case of strong geometric distortion, where the edge may be a curve, we use the following strategy to compute the mean gradient of a distorted edge as shown in Fig. 8: (1) calculate the length (L_{AB}) of the line between two vertices (A and B) and find its midpoint [the red point in Fig. 8(a)], (2) draw a perpendicular line centered at the midpoint with its length being l [$l = \max(2, 0.1\ L_{\text{AB}})$] pixels, (3) search along the perpendicular line and find the point with max gradient [the orange point in Fig. 8(a)], compute the distance between the point and the midpoint (Dist), (4) if the distance is not larger than two pixels, calculate the gradients of all pixels along L_{AB}, and if not, regard the max-gradient point as a new vertex and repeat the previous steps until all lines meet the distance requirement, and (5) calculate the mean gradient of all pixels along the line segments.

After the structure expansion of a checkerboard, all corners in it will be grouped with the same label, and all the structure recovery steps will be repeated until no corners without a label are found or no appropriate initial corners are found. Then, we check whether the checkerboard structures overlap or not by checking the corner labels. If there are corners with more than one label, the groups with the labels are regarded overlapping and only the one that contains most corners will be retained.

4 Experiments and Evaluation

We evaluate our method in four aspects: the detection rate, the detection speed, the ability to detect checkerboards in images taken under special or extreme conditions, and the accuracy of camera calibration with the detected corners. In the following sections, the experimental data are first introduced in Sec. 4.1, then the results in the four aspects are successively presented in Secs. 4.2–4.5.

4.1 *Data Introduction*

Six groups of images, including five public data sets, were tested in the experiments. The first three, the Mesa SR4000 (176×144 pixels), the IDS uEye (1280×1024 pixels), and the GoPro Hero 3 (4000×3000 pixels) datasets, are from the evaluation data of Ref. 9. The other two public ones are the partial-full datasets (1280×720 pixels) from Ref. 10 and the images (1392×512 pixels) from the KITTI calibration data.[14] The most significant difference between the KITTI calibration data and the other four datasets is that there are more than 10 checkerboards in each image of the KITTI data, whereas for the others there is only a single one. Table 2 briefly summarizes the characteristics of the five public datasets. The last group is made up of images taken under special or extreme conditions, such as poor lighting and overexposure. These datasets are representative in some characteristics of checkerboard images that can be well utilized to evaluate the corner detection methods.

4.2 *Quantitative Detection Results*

A certain number of point correspondences are needed to conduct the camera calibration, which means that enough checkerboards including their corners must be detected. In real cases, the checkerboards may be recorded by various cameras under different conditions, and a good method should be able to detect as many checkerboard corners as possible regardless of the situation. We use detection rate and corner percentage to evaluate the robustness of the

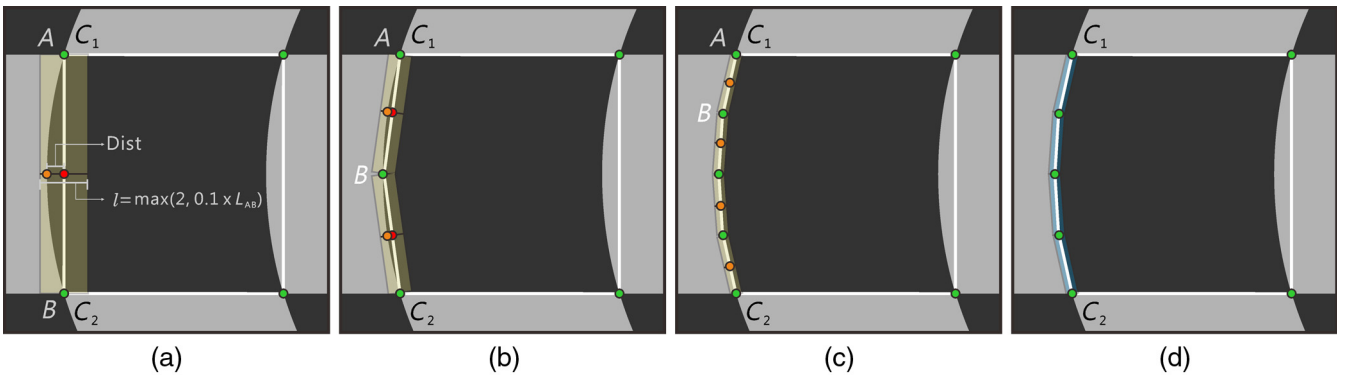

Fig. 8 Steps of mean gradient calculation of a distorted edge. (a) The midpoint (red) of the line between two corners is far from the point (orange) with max gradient. (b) They are closer when the corners are connected with two lines linked by the former max-gradient point. (c) The midpoints and the max-gradient points almost overlap, which also means the lines quite resemble the real edge of a checkerboard block. (d) The pixels along all the constructed lines (light blue) are used to calculate the mean gradient.

Table 2 Simple summarization of the public datasets.

	Resolution	Distortion	Wide angle camera[b]	Fully visible[c]	Noise	Pose	Number of checkerboards[d]
Mesa	Low	Strong	—[a]	All	Large amount	Extreme	Single
IDS	Medium	Little	—[a]	All	Various	Extreme	Single
GoPro	High	Strong	Yes	All	Little	—[a]	Single
Partial-full	Medium	Strong	Yes	Partial	Various	—[a]	Single
KITTI	Low	—[a]	—[a]	All	Various	Various	Multiple

[a]The "—" symbol means that there is no extra explanation of the corresponding characteristic with regard to the corresponding dataset.
[b]The "wide angle camera" column shows whether the images are taken with a wide angle camera.
[c]In the "fully visible" column, "all" means the checkerboard corners are fully visible in all images of a dataset; "partial" means only in some images are the checkerboard corners fully visible.
[d]In the last column, "single" means that there is only one checkerboard in each image; "multiple" means that there are more than one checkerboard in a single image.

proposed method with regard to resolution, geometric distortion, noise, and extreme poses. Here, the detection rate is defined as the ratio of the number of successfully detected images to the total number of images in a group, whereas the "corner percentage" is the ratio of the number of detected corners to the total number of all corners (including the occluded ones of a checkerboard), and "successfully detected" means that the corner percentage of an image is 100%.

Some other automatic checkerboard detection methods and implementations are also applied in the experiments for comparison, including the method using a single shot proposed by Geiger et al.,[12] the Camera Calibrator app included in the "2016b" version of MATLAB,[23] the ROCHADE method,[9] and the OCPAD method.[10] The quantitative detection results of the methods are shown in Table 3.

In the experiments of Table 3, the values of the four parameters (MQ, FS, CS, and NMS introduced in Sec. 2) of the proposed method are set as follows. The MQ value is set to 0.001 for all the datasets, whereas the other parameters vary as they are quite related to the image resolution. The FS, the CS, and the NMS values are set to three, five, and three (pixels), respectively, for the Mesa and KITTI datasets, seven, seven, and five (pixels) for the IDS and partial-full datasets, and seven, thirteen, and nine (pixels) for the GoPro dataset, respectively. Larger values of the three parameters should be used in processing images with higher resolutions, but it is not strictly necessary. The setup of the

Table 3 The number of the successfully detected images in each dataset using different methods.

	Mesa	IDS	GoPro	Partial-full	KITTI
Total images	206	206	100	162	20
ROCHADE[a]	195	205	96	44	—
OCPAD[a]	200	205	**100**	44	—
Geiger's method	201	**206**	**100**	61	17
MATLAB app	**204**	205	99	**64**	—
SCCD (proposed)[b]	**204**	**206**	**100**	63	**18**

Note: Bold values are best results of each dataset.
[a]The result data of ROCHADE and OCPAD are quoted from Ref. 10.
[b]Our proposed self-correlation checkerboard detection method is written in "SCCD" for short.

three parameters for the five datasets is representative for low-, medium-, and high-resolution images, and in fact, the setup for the medium IDS and partial-full datasets can also be used in the high-resolution GoPro dataset, where experiments show the same detection results. It should also be noted that the filtering step (introduced in Sec. 2.1) after the Harris corner detection is applied in the IDS, GoPro, and partial-full datasets during the experiments.

In the cases of the Mesa, IDS, and GoPro data, the performances of the five methods are all good and very similar, in which the numbers of successfully detected images are all close to or equal to the total images. The detection rates of the proposed method and the MATLAB app are highest, 204/206, in the Mesa data results. Geiger's method and the proposed method do best with the IDS data with a successful detection of all images, but the other three also perform well with only one image failing. For the GoPro data, the OCPAD, the Geiger's, and the proposed methods have the highest detection rate, 100%, and the ones of the other two methods are little smaller.

The results of the partial-full data show much lower detection rates of all methods, but it is reasonable because the checkerboards of many images are positioned close to the image boundary in this dataset that many corners are occluded in the images. The MATLAB app has the most successfully detected images, which is 64 images, one more than the proposed method.

For the KITTI data, though only 20 images in size, there are more than 10 checkerboards in each of them, and the conditions of the checkerboards, such as lighting and pose, are various. The proposed method does best on this dataset, with 18 images (226 checkerboards in total) successfully detected. In fact, the total number of detected checkerboards is even more than 226 because some checkerboards in the failed two images are also fully detected. Among the other four methods, only Geiger's method has the ability to detect multiple checkerboards in an image, whose result is 17, a little smaller than the proposed method. However, if the MQ value is set to 0.0001, all the 20 images in the KITTI dataset can be successfully detected using our proposed method.

For the failed images in all the datasets, the proposed method is able to detect partial checkerboards, so higher detection rates can be acquired if lower corner percentages are allowed as shown in Fig. 9. The detection rates of the five datasets all reach 100% when a corner percentage more than 30% is regarded as "successfully detected," and they will all be larger than 95% if the limit of the corner percentage is set to >50%. The detection rates of the partial-full dataset are obviously lower than the others when required corner percentages are large.

The proposed method is robust as shown by its high detection rates of all the datasets. The numbers of successfully detected (corner percentage equal to 100%) images using the proposed method are the largest in four datasets of all the five public experiment datasets and outperform the four other methods, and for the remaining dataset it is only one image less than the MATLAB app.

4.3 *Detection Speed*

When processing a large number of images, the detection speed is also an important factor to judge a detection method. Table 4 shows the average runtime for processing an image of all the experiment datasets using different methods. Here, only Geiger's method and the MATLAB app are used in the speed experiments to ensure the same running platform (the hardware, the system, etc.) for comparison. The programs run on the Windows system with an Intel(R) Core(TM) i7-4790 CPU and 16 GB RAM. They are all written in

Table 4 Average runtime for different datasets using different methods (unit: second).

	Mesa	IDS	GoPro	Partial-full	KITTI	Average
Geiger's method	1.43	4.43	126.76	3.42	21.97	31.60
MATLAB app	**0.02**	**0.34**	68.52	**0.19**	—	17.27
SCCD (proposed)	0.59	0.77	**10.40**	0.79	**5.53**	**3.62**

Note: Bold values are best results of each dataset.

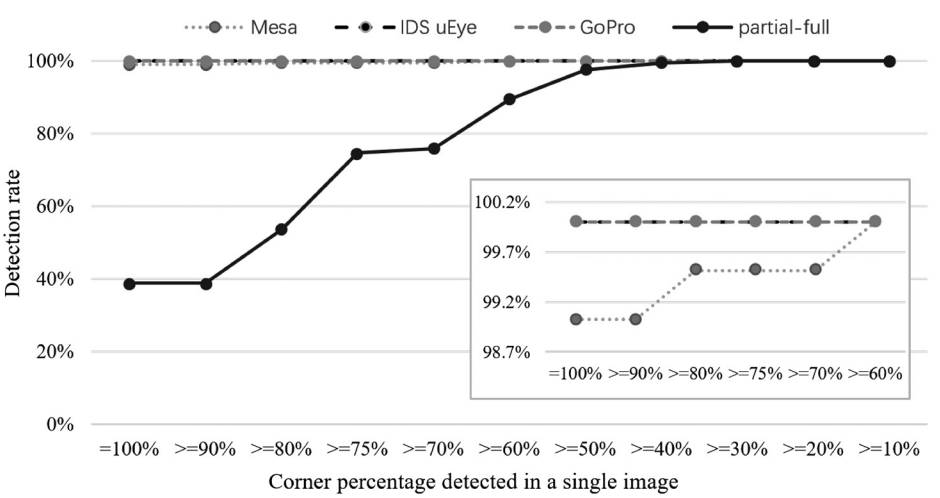

Fig. 9 Detection rate when different corner percentages allowed. The smaller figure in the bottom right box shows the overlapped lines in the top left part of the whole figure.

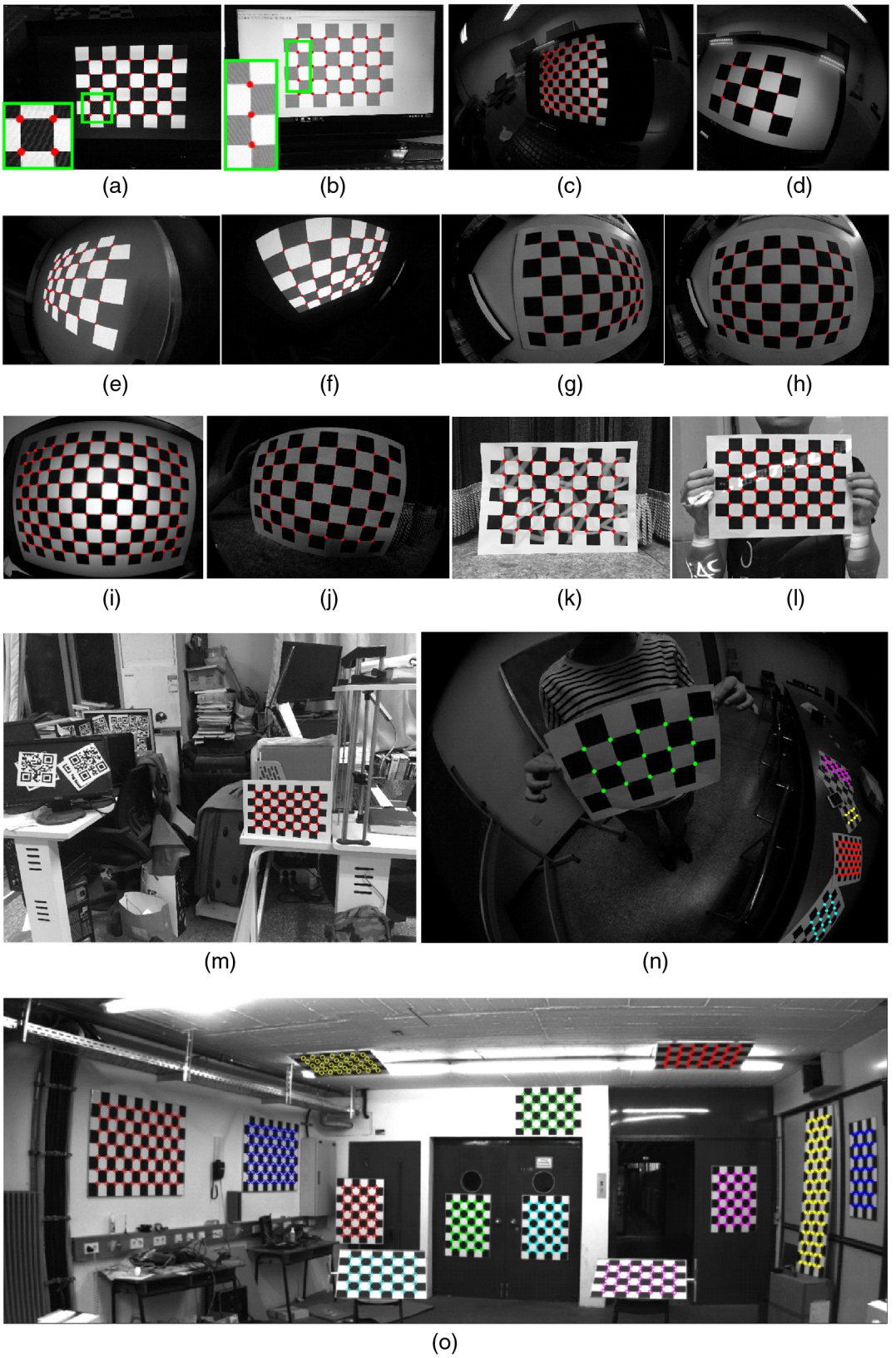

Fig. 10 Detection results. (a–d) Checkerboards shown on screen, (e and f) checkerboards projected on a white wall, (g–i) weakly blurred checkerboards with strong fisheye distortion, (j) checkerboards in a noisy image, (k and l) checkerboards with some slightly shaded areas, (m) a complex circumstance, and (n and o) multiple checkerboards in a single image.

MATLAB and use the CPU. The four important parameters of the proposed method are set the same as explained in Sec. 4.2.

For the Mesa, IDS, and partial-full datasets, the MATLAB app needs the shortest runtime and the ones of the proposed method are all less than 1 s as well. For the high-resolution GoPro data, it is quite time-consuming using Geiger's method and the MATLAB app, which are more than 1 min, whereas it only needs 10.4 s with the proposed method. Overall, the proposed method is the fastest with an average runtime being 3.62 s.

4.4 *Qualitative Detection Results*

Our proposed method has the ability to detect checkerboards in some images taken under special or extreme conditions, such as strong distortion, extreme poses, slight shade, blurring, projected checkerboards, and screen-shown checkerboards. In this section, representative results are presented as shown in Fig. 10.

In Figs. 10(a)–10(d), the checkerboards are shown on screen with different backgrounds, where the first two are interfered with curved stripes (this can be seen when the images are zoomed in). Figures 10(e)–10(f) record checkerboards projected on a white wall with strong geometric distortion. The ones in Figs. 10(g)–10(j) are also significantly distorted, but there are some differences among them—Figs. 10(g), 10(h), and 10(i) are weakly blurred, and the checkerboards appear on different media, whereas Fig. 10(j) is a bit noisy. In Figs. 10(k) and 10(l), some parts of the checkerboards are covered with some projected characters, which may affect the detection process. Figure 10(m) shows a complex circumstance. Figures 10(n) and 10(o) contain multiple checkerboards, and the former is taken by a fisheye camera, whereas the other is from the KITTI dataset. The checkerboard detection succeeds in all the images in Fig. 10 using the proposed method, and different color and shape symbols mean different checkerboards detected in the same image.

4.5 *Camera Calibration Accuracy*

It is important to find the accurate position of the checkerboard corners, as well as recovering the structure, in camera calibration. We calculate the calibration accuracy using the checkerboard corners detected by different methods to evaluate the proposed method. Zhang's camera calibration method[2] is utilized due to its robustness, convenience, and high efficiency, and the calibration functions for normal and fisheye images in the Camera Calibration Toolbox for MATLAB[3] are also applied to conduct the calibration.

For comparison, the detection results of Geiger's method and the MATLAB app are used in the calibration experiments as well. The same images that can be fully detected by all three methods in each dataset are processed in the calibration. As the Mesa and the IDS datasets contain stereo images recorded by binocular cameras (left and right), only the right images of the two datasets are used. It should be noticed that the GoPro images and the partial-full images use the fisheye calibration functions in the toolbox for their strong geometric distortion, whereas the other three use the normal calibration functions. The calibration precision of each dataset is presented by the reprojection pixel error shown in Table 5. After the calibration step, the intrinsic and extrinsic parameters of the cameras can be obtained, so the two-dimensional (2-D) image coordinates of the corners can be calculated later by reprojection using Eq. (12) with the already known 3-D coordinates. Then, the reprojection error can be calculated using Eq. (13).

Table 5 Calibration accuracy using different methods.

Pixel error	Mesa	IDS	GoPro	Partial-full	KITTI
Number of used images	101	102	99	61	17
MATLAB app	**0.1233**	0.3740	0.4083	0.3785	—
Geiger's method	0.1455	**0.3690**	**0.4042**	**0.3622**	0.1491
SCCD (proposed)	0.1330	0.3702	0.4061	0.3760	**0.1460**

Note: Bold values are best results of each dataset.

$$s\tilde{\mathbf{x}} = \mathbf{K}[\,\mathbf{R} \quad \mathbf{T}\,]\tilde{\mathbf{X}}, \tag{12}$$

where $\mathbf{K}$ is the intrinsic matrix, $\mathbf{R}$ and $\mathbf{T}$ are the rotation matrix and the translation matrix, $\tilde{\mathbf{X}}$ denotes the homogenous 3-D coordinates of the corners, $\tilde{\mathbf{x}}$ denotes the reprojected homogenous 2-D image coordinates of the corners, and s is a scalar factor, respectively.

$$e_{\text{reprojection}} = \frac{1}{N}\sum \|\mathbf{x}_i' - \mathbf{x}_i\|_2, \tag{13}$$

where N is the number of corners, i is the index of a corner, $\mathbf{x}_i'$ denotes the detected 2-D image coordinates of the corners, and $\mathbf{x}_i$ denotes the calculated 2-D image coordinates, respectively.

All pixel errors are less than 0.5 pixels for the three methods. The MATLAB app does best on the Mesa data, while having the largest pixel errors in all the other datasets. The accuracies of the Mesa and the KITTI data with the proposed method are better than Geiger's method and the others worse, but the difference is small. The results indicate that the proposed method can bring in high calibration accuracy and our method is never the worst one, which shows its overall stability.

5 Conclusion

In this work, a method for automatic checkerboard detection is presented that takes advantage of the central symmetric feature of the checkerboard corners as well as their spatial relationship and grayscale distribution. The method can be used to acquire the intrinsic and extrinsic parameters in camera calibration for 3-D reconstruction and other applications. It contains a corner extraction approach using self-correlation and a structure recovery solution using constraints related to adjacent corners and checkerboard block edges. Experiments indicate good performance of the proposed method, which has the highest detection rates (some reaching 100%) in four out of five public datasets considered when compared with four other advanced checkerboard detection methods. The detection speed of the proposed method is fast and its average runtime for one image is less than 5 s, which is even shorter (<1 s) when processing low and medium resolution images. In addition, the proposed method can detect

multiple checkerboards in a single image, and it is robust against some special and extreme conditions, such as partial checkerboards, screen-shown, and projected checkerboards as well as geometric distortion, slight shade, and extreme poses. The camera calibration results using the checkerboards detected by the proposed method also show good quantified accuracy. However, there are still some problems that need to be further studied. In extreme cases, such as part of the checkerboard being in shadow or the checkerboard being projected onto a patterned surface that leads to great difference between a correct checkerboard corner and all its neighboring corners, the correct corner may also be removed using the proposed method, resulting in only part of a checkerboard being detected, and this can be improved in the future.

Acknowledgments

We thank the anonymous reviewers for their help in the improvement of the paper. This work was supported by the National Natural Science Foundation of China [Grant Number 41571432], the National Key Research and Development Program of China [Grant Number SQ2017YFGX040110], the National Key Research and Development Program of China [Grant Number 2017YFB0503004], and the State Administration of Foreign Experts Affairs Program of China [Grant Number GDT0161100077].

References

1. R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE J. Rob. Autom.* **3**(4), 323–344 (1987).
2. Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(11), 1330–1334 (2000).
3. J. Y. Bouguet, "Camera calibration toolbox for Matlab," 2015, http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#examples (24 August 2017).
4. V. Vladimir, "OpenCV calibration object detection," 2017, http://graphicon.ru/oldgr/en/research/calibration/opencv.html (24 August 2017).
5. A. de la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," *Sensors (Basel)* **10**(3), 2027–2044 (2010).
6. J. Chu, A. GuoLu, and L. Wang, "Chessboard corner detection under image physical coordinate," *Opt. Laser Technol.* **48**, 599–605 (2013).
7. S. Bennett and J. Lasenby, "ChESS—quick and robust detection of chess-board features," *Comput. Vision Image Understanding* **118**, 197–210 (2014).
8. S. Bennett and J. Lasenby, "Robust recognition of chess-boards under deformation," in *20th IEEE Int. Conf. on Image Processing (ICIP)*, pp. 2650–2654 (2013).
9. S. Placht et al., "Rochade: robust checkerboard advanced detection for camera calibration," in *13th European Conf. on Computer Vision (ECCV)*, pp. 766–779 (2014).
10. P. Fuersattel et al., "OCPAD–occluded checkerboard pattern detector," in *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pp. 1–9 (2016).
11. Y. Bok, H. Ha, and I. S. Kweon, "Automated checkerboard detection and indexing using circular boundaries," *Pattern Recogn. Lett.* **71**, 66–72 (2016).
12. A. Geiger et al., "Automatic camera and range sensor calibration using a single shot," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 3936–3943 (2012).
13. C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conf.*, p. 10-5244 (1988).
14. A. Geiger et al., "Vision meets robotics: the KITTI dataset," *Int. J. Rob. Res.* **32**(11), 1231–1237 (2013).
15. S. M. Smith and J. M. Brady, "SUSAN—a new approach to low level image processing," *Int. J. Comput. Vision* **23**(1), 45–78 (1997).
16. D. G. Lowe, "Object recognition from local scale-invariant features," in *7th IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 1150–1157 (1999).
17. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision* **60**(2), 91–110 (2004).
18. E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *9th European Conf. on Computer Vision (ECCV)*, pp. 430–443 (2006).
19. A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *18th Int. Conf. on Pattern Recognition (ICPR)*, pp. 850–855 (2006).
20. P. Kovesi, "Non-maximum suppression," 2017, http://www.peterkovesi.com/matlabfns/Spatial/nonmaxsuppts.m (24 August 2017).
21. L. Lucchese and S. K. Mitra, "Using saddle points for subpixel feature detection in camera calibration targets," in *Asia-Pacific Conf. on Circuits and Systems*, Vol. 2, pp. 191–195 (2002).
22. L. Krüger and C. Wöhler, "Accurate chequerboard corner localisation for camera calibration," *Pattern Recogn. Lett.* **32**(10), 1428–1435 (2011).
23. MathWorks, "Camera calibrator," 2017, https://cn.mathworks.com/help/vision/ref/cameracalibrator-app.html (24 August 2017).

Yizhen Yan received her BS degree in geographic information science from East China Normal University in 2015. Currently, she is an MS candidate at Peking University. Her research interests include image processing, feature detection, light field imaging, and stereo.

Peng Yang received his BS degree in remote sensing science and technology from Beihang University in 2013. Currently, he is a PhD candidate at Peking University. His research interests include calibration, light field imaging, and stereo.

Lei Yan received his BS degree from Nanjing University of Aeronautics and Astronautics in 1981, his MS degree from Navy University of Engineering in 1989, and his PhD from Tsinghua University in 1994. Currently, he is a professor at the School of Earth and Space Sciences at Peking University and the head of Spatial Information Integration and its applications, Beijing Key Laboratory. His current research interests include high-resolution imaging, radiometric calibration, remote sensing of polarization, and photogrammetry.

Jie Wan received his BS degree from Wuhan University in 2011 and his MS degree from the University of Chinese Academy of Sciences in 2014. Currently, he is a PhD candidate at Peking University. His research interests include structure from motion, stereo, and deep learning.

Yanbiao Sun received his BS degree from Wuhan University in 2010 and his PhD from Peking University in 2015. Currently, he is a postdoctoral fellow at University College London. His research interests include feature extraction and matching, photogrammetry, machine learning, and deep learning.

Kevin Tansey is a professor of remote sensing at the University of Leicester, UK. His research interests include algorithm development and image analysis and interpretation from satellite data. He has published 50 papers on this subject since obtaining his PhD in 1999. He is the editor of the International Journal of Remote Sensing and the MDPI Open Access Journal Fire.

Anand Asundi received his PhD from the State University of New York at Stony Brook and joined the University of Hong Kong in 1983, where he was a professor till 1996. Currently, he is a professor and the deputy director of the Advanced Materials Research Centre at the Nanyang Technological University in Singapore. He is the editor of optics and lasers in engineering and on the board of directors of SPIE, the International Society of Optical Engineers.

Hongying Zhao received her BS degree from Changchun University of Technology, China, in 1993, where she also received her MS degree in 1998. She received her PhD from the University of Chinese Academy of Sciences in 2002. Currently, she is an associate professor at Peking University. Her research interests include photogrammetry, remote sensing, and light field imaging.