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], Hongying Zhao[a]*

[a] Beijing Key Laboratory of Spatial Information Integration & Its Applications, School of Earth and Space Sciences, Peking University, Beijing, China, 100871

[b] Department of Civil, Environmental & Geomatic Engineering, University College London, London, UK, WC1E 6BT

[c] Leicester Institute for Space and Earth Observation & Centre for Landscape & Climate Research, School of Geography, Geology and the Environment, University of Leicester, Leicester, UK, LE1 7RH

[d] Centre for Optical and Laser Engineering, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore, 639798

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 in this paper. 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 re-projected pixel errors smaller than 0.5 pixels.

Keywords: Corner detection, correlation, structure recovery, camera calibration, photogrammetry

*Corresponding author, E-mail: zhaohy@pku.edu.cn

1 Introduction

Camera calibration is an important step to get the intrinsic parameters of cameras for better sensing the world, such as three-dimensional 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, whilst the latter requires the number of corners in a row and a column, resulting in a complicated and semi-automatic calibration procedure. Other approaches include De la Escalera and Armingol, who proposed an automatic chessboard detection method using the Hough transform to detect straight chessboard edge lines then extracting the corners [5]. Chu et al. detected the chessboard corners using a round template by analyzing the gray distribution in it and computing the centroids of redundant points [6]. 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 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 doesn't recover the whole structure of the checkerboards for calibration [7]. Their subsequent work [8] deals with the structure, and gets 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 90°. Placht et al. 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 [9]. Fuersattel et al. 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 [10]. Bok et al. 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 [11]. However, the three methods detect only a single checkerboard in one image, and that means the camera or the checkerboard need to be relocated for several times to ensure robust and accurate calibration results. Geiger et al. 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 [12]. 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 tradeoff 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 Harris detector [13] first, then a square area centred at each pixel in the corner neighborhood is selected and rotated by 180° ; 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, while 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 to 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 grey distribution in the checkerboard square as well as angle and distance constraints; (3) gaining robust results against distortion with a high detection rate.

This work is organized as follows. Section 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 Section 4. Finally, we draw conclusions based on the evaluation in Section 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) non-maximum suppression on the self-correlation map; (5) outlier discarding through some constraints; (6) refinement. The details of the above steps are described in the following subsections, with step (1) and (2) in

Subsection 2.1, step (3) and (4) in Subsection 2.2, and step (5) and (6) in Subsection 2.3.

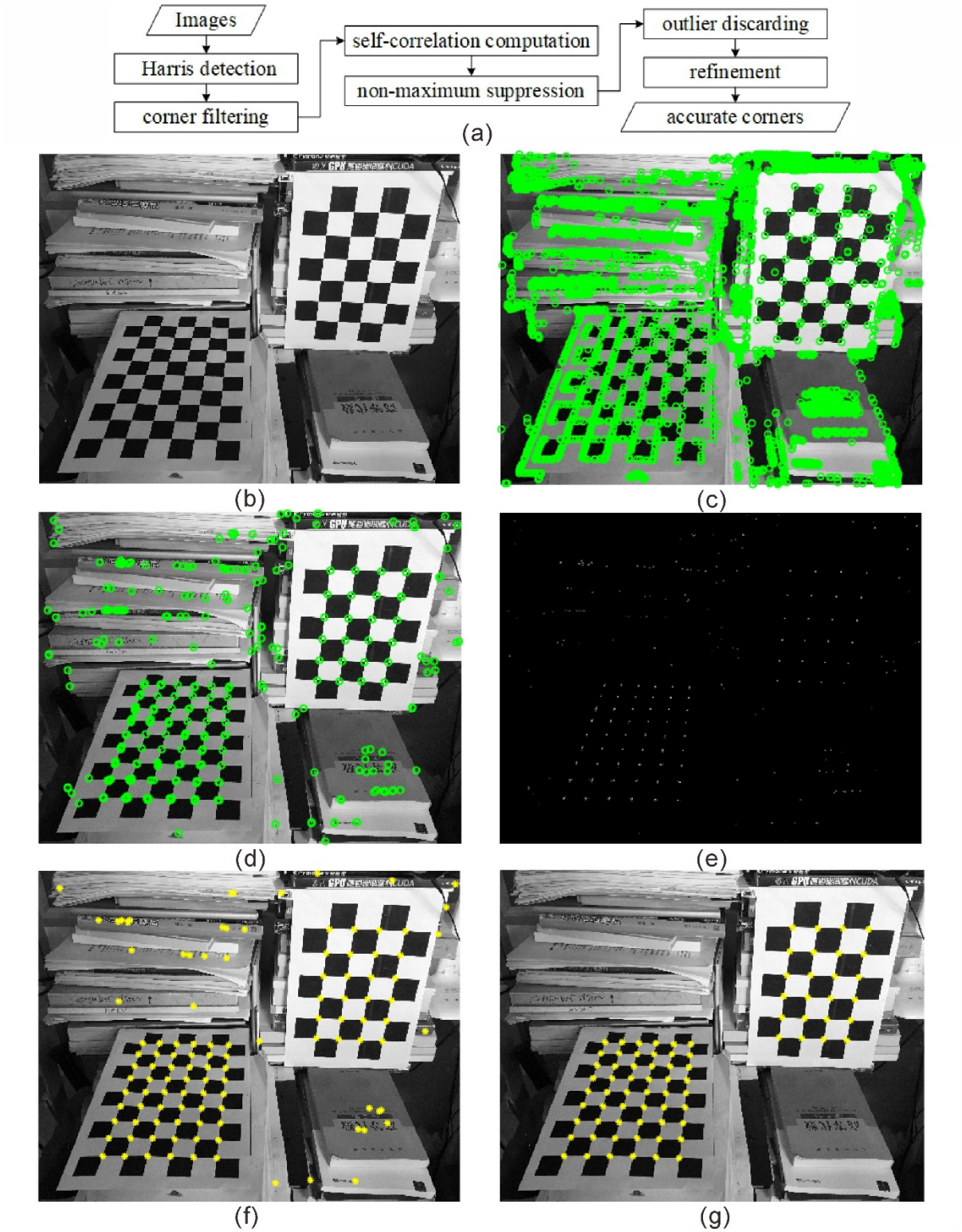

Fig. 1 Processing steps of checkerboard corner detection. (a) Flowchart. (b) Original gray image. (c) Corners detected by Harris. (d) Corners after filtering. (e) Self-correlation map, pixels with larger gray values represent higher checkerboard corner likelihood. (f) Non-maximum suppression on the self-correlation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

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. 1b. There are two important parameters in the Harris corner detection that can be changed to get 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 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. 2a: (1) containing two bright and two dark parts with a cross shape; (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 sub-image 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, 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. 2b), then calculate the correlation between the original square and the newly modified one with Eq. 1, so as to check whether the corner meet the first condition. The corners with low correlation are discarded (like the bad example in Fig. 2b). 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 since 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.

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

Where A and B are two $n\times n$ matrixes, $\overline{A}$ and $\overline{B}$ denote the mean value of all elements in A and B, respectively.

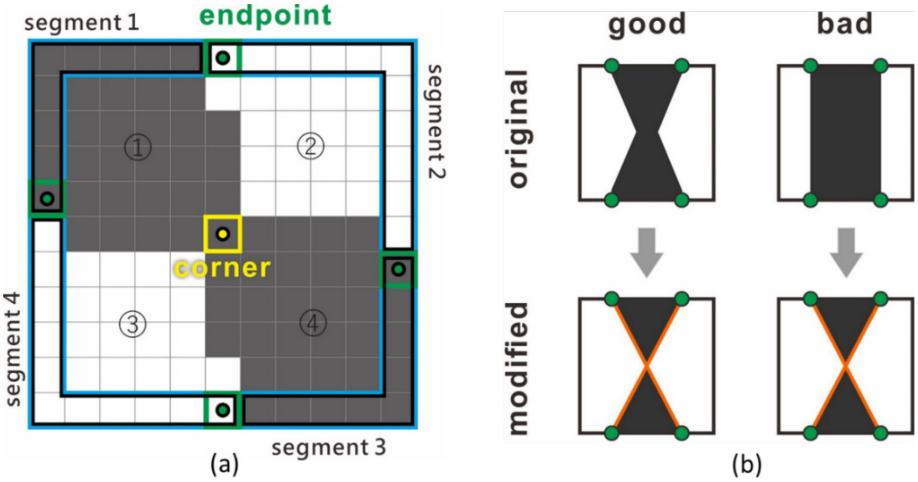

Fig. 2 Schematic diagrams of the corner neighborhood area. (a) Square centered at the corner, with two bright (② and ③) and two dark (① and ④) parts in it; the boundary is marked by blue lines and contains four black and white segments; the endpoints are actually where the signal changes and where the 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.

2.2 Corner extraction using self-correlation and non-maximum suppression

The central symmetric property of the checkerboard corner neighborhoods can be well utilized to extract the real checkerboard corners from the mass potential corners as shown in Fig. 3a. 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 Fig. 3b and Fig. 3c. 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 180°, 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 non-maximum suppression step. The key idea of non-maximum 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 [19, 20].

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

Where 0.8 is an empirical value.

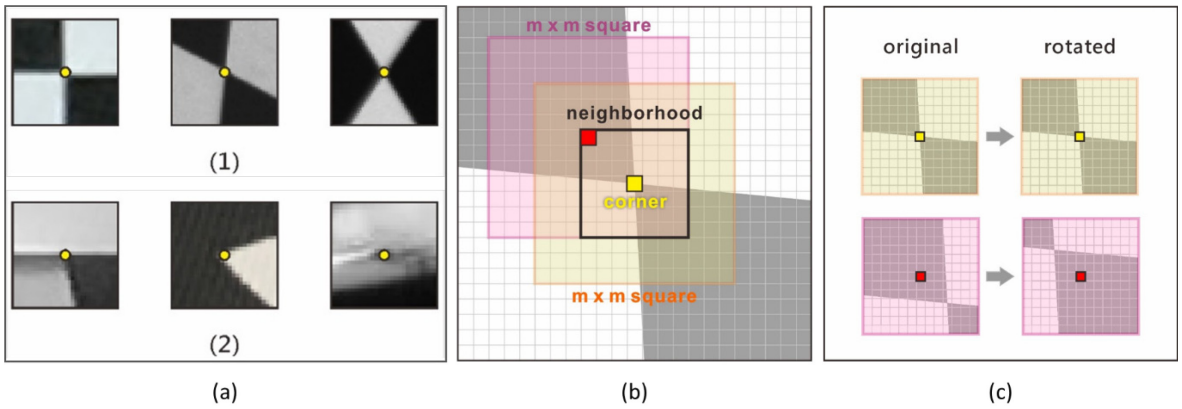

Fig. 3 The central symmetric property of the checkerboard corner neighborhood and its application in corner detection. (a) Some examples of checkerboard corners (1) and other corners (2). (b) The schematic of the square areas processed in correlation calculation. (c) The original and rotated squares 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.

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 standard deviation 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.

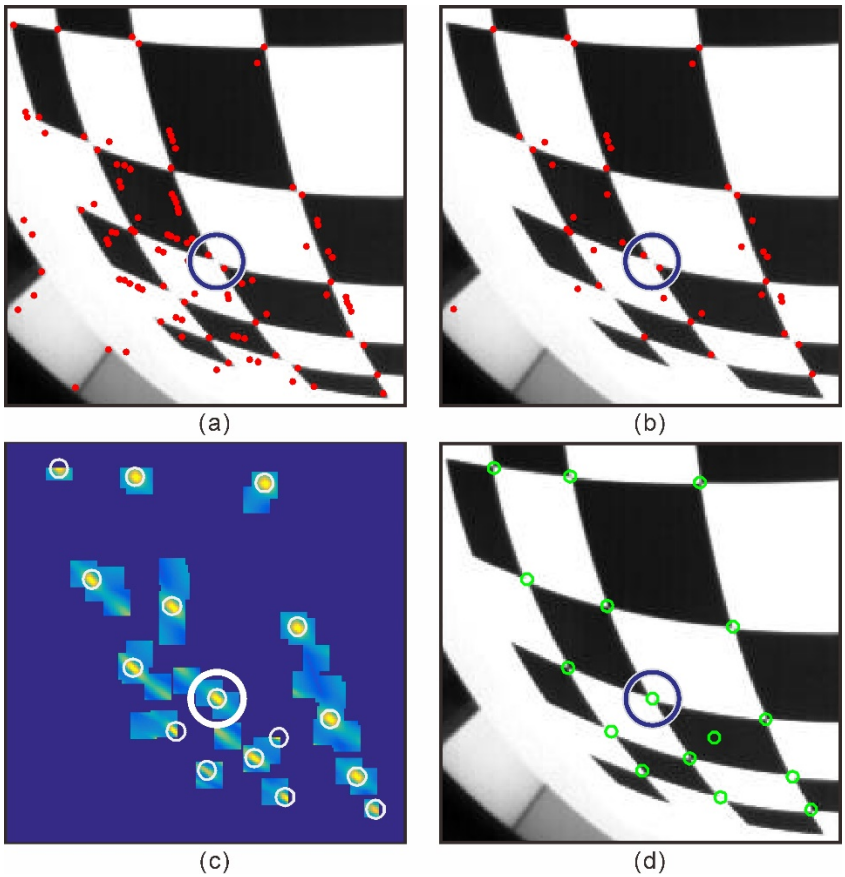

Fig. 4 Self-correlation computation and non-maximum suppression steps. (a) Candidate corners detected by Harris. (b) Corners after filtering. Cluster corner points surround the checkerboard corner positions, some of which even remain after the filter step. (c) The self-correlation map, where brighter 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 non-maximum suppression. After the steps, the corner clusters are processed into single corners with more accurate positions, such as the one in the large circle.

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 $\boldsymbol{C}$ denote the set of the corners, i and j be the indexes of two corners, then $\boldsymbol{CN}(i)$ and $\boldsymbol{CN}(j)$ represent the 12 closest corner index vectors corresponding to corner $\boldsymbol{C}(i)$ and $\boldsymbol{C}(j)$, respectively. If $i \in \boldsymbol{CN}(j)$ and $j \in \boldsymbol{CN}(i)$, $\boldsymbol{C}(i)$ and $\boldsymbol{C}(j)$ are regarded as neighbors and their correlation $Corr(i, j)$ 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.

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

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

$$remove(i) = \left(\sum\nolimits_j \left(\left|Corr\left(\boldsymbol{C}(i), \boldsymbol{C}(j)\right)\right| > \tau\right)\right) < 1, \quad i \in \boldsymbol{CN}(j) \,\&\, j \in \boldsymbol{CN}(i) \tag{4}$$

Where $Corr$ is the correlation operator, $remove(i) = 1$ denotes that corner $\boldsymbol{C}(i)$ 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 sub-pixel accuracy of calibration. The method proposed by Geiger et al. uses both corner position and edge orientation information to do the refinement, leading to a good sub-pixel result [12]. Therefore in this work, the same refinement method is adopted.

First, two edge orientation vectors can be refined with Eq. 5. Since the edge orientations $\boldsymbol{E}_i$ are almost orthogonal to the image gradients on the edges, the deviation of their normal $\boldsymbol{E}_i'$

with respect to the gradients $\boldsymbol{g}_{np}$ should be as small as possible.

$$\boldsymbol{E}_i = \arg\min_{\boldsymbol{E}_i'} \sum\nolimits_{\boldsymbol{np}\in\boldsymbol{N}_i} \left(\boldsymbol{g}_{np}^T \boldsymbol{E}_i'\right)^2 \qquad s.t. \qquad \boldsymbol{E}_i'^T \boldsymbol{E}_i' = 1 \tag{5}$$

Where $\boldsymbol{np}$ denotes the neighbor pixel in the corner neighborhood, and $\boldsymbol{N}_i$ 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 $\boldsymbol{np}$ of corner $\boldsymbol{C}_p$, the image gradient of it should be approximately orthogonal to $\boldsymbol{np} - \boldsymbol{C}_p$. Therefore, the corner positions can be optimized using Eq. 6.

$$\boldsymbol{C}_p = \arg\min_{\boldsymbol{C}_p'} \sum\nolimits_{\boldsymbol{np}\in\boldsymbol{N}_C} \left(\boldsymbol{g}_{np}^T \left(\boldsymbol{np} - \boldsymbol{C}_p'\right)\right)^2 \tag{6}$$

Where $\boldsymbol{N}_C$ 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 Section 2. Note that, though the values of the four parameters can be adapted to more situations, their empirical values in the experiments (Subsection 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 non-maximum suppression	Non-maximum 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 get 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, 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; (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 subsections.

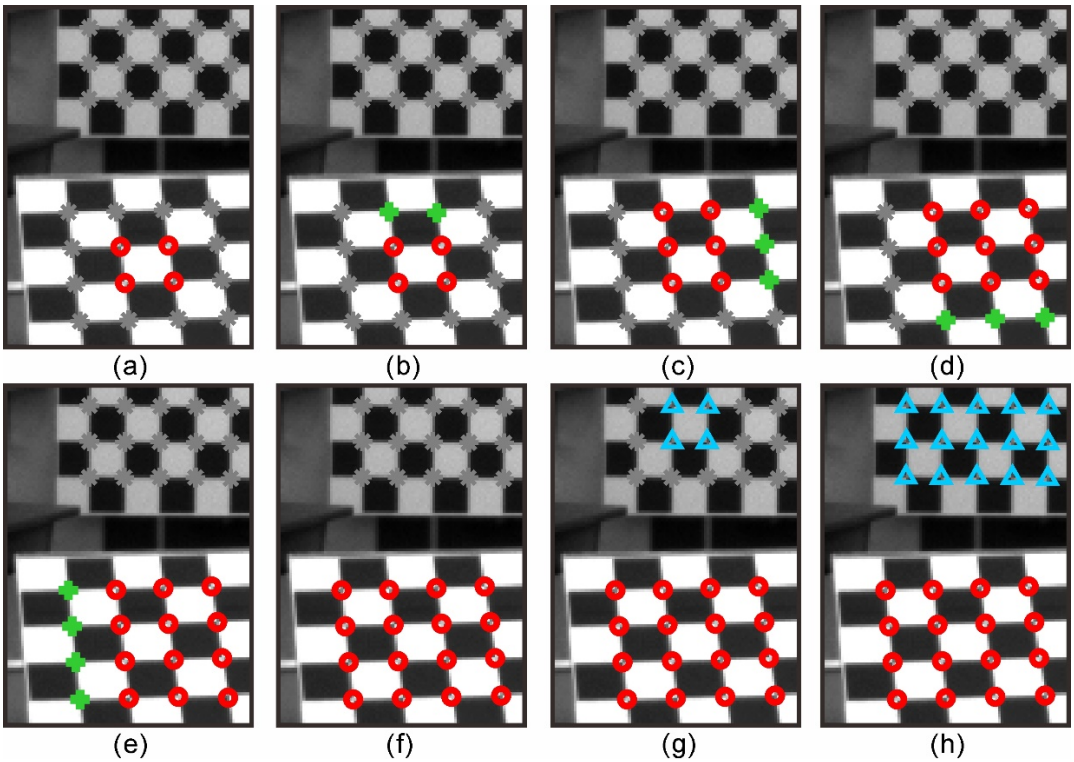

Fig. 5 Steps of checkerboard structure recovery. (a) Four initial corners. (b) Expanding in the top direction. (c) Expanding in the right direction. (d) Expanding in the bottom direction. (e) Expanding in the left direction. (f) Expansion completed. (g) Four new initial corners found. (h) Structure of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

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, the angles and the 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 with Eq. 7.

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

Where $\boldsymbol{e}$ is a unit vector along the horizontal axis direction.

An eligible seed is first detected as the left-bottom initial corner ($\boldsymbol{C}_L$), then the top-left initial

corner ($\boldsymbol{C}_T$), the right-top corner ($\boldsymbol{C}_R$), and the bottom-right one ($\boldsymbol{C}_B$) are selected one by one as shown in Fig. 6a. The details are shown as the following. If no suitable initial corners are found, the former seed will be labelled 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.

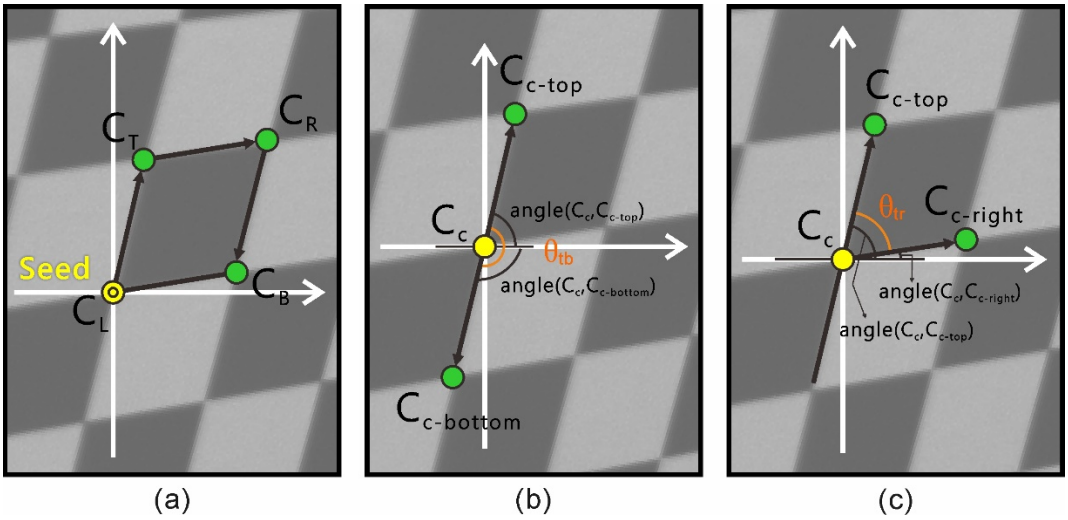

Fig. 6 Schematic diagrams in finding initial corners. (a) A seed is first detected as the left-bottom initial corner, then the top-left, right-top, bottom right corners are found in turn. (b) and (c) show a number of parameters that will be used in the detection of initial corners.

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

$$Corr\left(\boldsymbol{C}_L,\boldsymbol{C}_T\right),Corr\left(\boldsymbol{C}_T,\boldsymbol{C}_R\right),Corr\left(\boldsymbol{C}_R,\boldsymbol{C}_B\right),Corr\left(\boldsymbol{C}_B,\boldsymbol{C}_L\right)<-\tau_{corr} \tag{8}$$

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 ($\boldsymbol{C}_c$), the two with the smallest angles to the y axis are regarded as the top ($\boldsymbol{C}_{c\text{-}top}$) and bottom ($\boldsymbol{C}_{c\text{-}bottom}$) neighbors of $\boldsymbol{C}_c$, respectively, and the angle (θ_{tb}) between edge $\boldsymbol{C}_c$ -$\boldsymbol{C}_{c\text{-}top}$ and edge $\boldsymbol{C}_c$ -$\boldsymbol{C}_{c\text{-}bottom}$ should be close to 180° shown as Fig. 6b,

$$-\tau_{angle1} \le \left|angle\left(\boldsymbol{C}_c, \boldsymbol{C}_{c-top}\right)\right| + \left|angle\left(\boldsymbol{C}_c, \boldsymbol{C}_{c-bottom}\right)\right| - 180^\circ \le \tau_{angle1} \tag{9}$$

Where τ_{angle1} denotes an angle threshold.

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

$$-\tau_{angle2} \le \left|angle\left(\boldsymbol{C}_c, \boldsymbol{C}_{c-top}\right) - angle\left(\boldsymbol{C}_c, \boldsymbol{C}_{c-right}\right)\right| \le 180^\circ - \tau_{angle2} \tag{10}$$

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 standard deviation (*STD*) inside the quadrangle formed by the initial points $qd_{(\boldsymbol{C}_L,\boldsymbol{C}_T,\boldsymbol{C}_R,\boldsymbol{C}_B)}$ 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,

$$STD\left(qd_{m(\boldsymbol{C}_L,\boldsymbol{C}_T,\boldsymbol{C}_R,\boldsymbol{C}_B)}\right) < \tau_{std}, \quad margin = \max\left(1, 0.1L_{edge}\right) \tag{11}$$

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

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

Algorithm 1 finding initial corners

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

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

While true do

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

While $\tau_{corr} > 0.7$ do

$\left(\boldsymbol{C}_{L-top}, \boldsymbol{C}_{L-bottom}\right) \leftarrow FindNeighborsClosestToYAxis\left(Neighbor\left(\boldsymbol{C}_L\right)\right)$

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

$\boldsymbol{C}_R \leftarrow FindRightCorner\left(Neighbor\left(\boldsymbol{C}_T\right)\right)$; if $empty\left(\boldsymbol{C}_R\right)$ then go to ***P1***

$\boldsymbol{C}_B \leftarrow FindBottomCorner\left(Neighbor\left(\boldsymbol{C}_R\right)\right)$; if $empty\left(\boldsymbol{C}_B\right)$ then go to ***P1***

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

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

end

P2: $std\left(qd_{m(\boldsymbol{C}_L,\boldsymbol{C}_T,\boldsymbol{C}_R,\boldsymbol{C}_B)}\right) \leftarrow CalculateStandardDeviation\left(qd_{m(\boldsymbol{C}_L,\boldsymbol{C}_T,\boldsymbol{C}_R,\boldsymbol{C}_B)}\right)$

If $std\left(qd_{m(\boldsymbol{C}_L,\boldsymbol{C}_T,\boldsymbol{C}_R,\boldsymbol{C}_B)}\right) < \tau_{std}$ then break;

end

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.

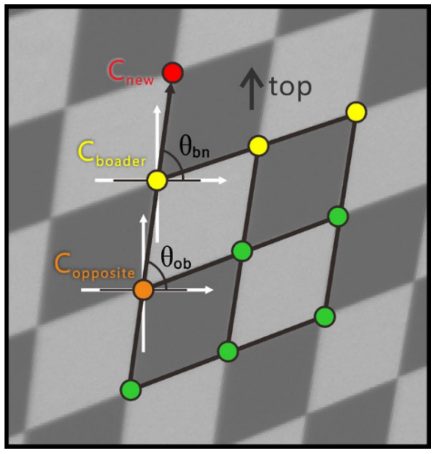

Fig. 7 Structure expansion in one direction.

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. 8a); (2) draw a perpendicular line centered at the midpoint with its length being l (l=max(2, 0.1× L_{AB})) pixels; (3) search along the perpendicular line and find the point with max gradient (the orange point in Fig. 8a), 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; (5) calculate the mean gradient of all pixels along the line segments.

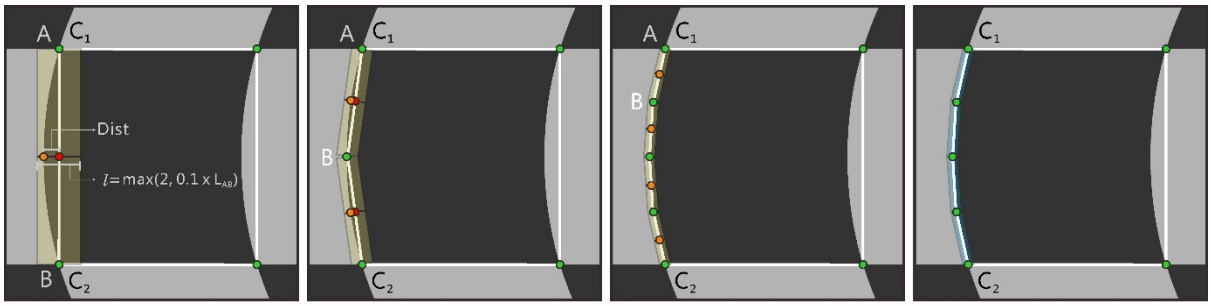

Fig. 8 Steps of mean gradient calculation of a distorted edge. (a) The midpoint (red) of the line between two corners is far from the point (orange) with max gradient. (b) They are closer when the corners are connected with two lines linked by the former max-gradient point. (c) The midpoints and the 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.

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 subsections, the experiment data are first introduced in Subsection 4.1, then the results in the four aspects are successively presented in Subsection 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 [9]. The other two public ones are the Partial-full datasets (1280×720 pixels) from [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 ten checkerboards in each image of the KITTI data while 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.

Table 2 Simple summarization of the public datasets.

	Resolution	Distortion	Wide angle camera	Fully visible	Noise	Pose	Number of checkerboards
Mesa	low	strong	—	all	large amount	extreme	single
IDS	medium	little	—	all	various	extreme	single
GoPro	high	strong	yes	all	little	—	single
Partial-full	medium	strong	yes	partial	various	—	single
KITTI	low	—	—	all	various	various	multiple

Note: 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.

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 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, while 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 (bold values are best results of each dataset).

In the experiments of Table 3, the values of the four parameters (*MQ*, *FS*, *CS* and *NMS* introduced in Section 2) of the proposed method are set as follows. The *MQ* value is set to 0.001 for all the datasets, while the other parameters vary since 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. 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 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 Subsection 2.1) after the Harris corner detection is applied in the IDS, GoPro and Partial-full datasets during the experiments.

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	195	205	96	44	—
OCPAD	200	205	**100**	44	—
Geiger's method	201	**206**	**100**	61	17
MATLAB app	**204**	205	99	**64**	—
SCCD (proposed)	**204**	**206**	**100**	63	**18**

Note: a. the result data of ROCHADE and OCPAD are quoted from [10].

b. our proposed self-correlation checkerboard detection method is written in 'SCCD' for short.

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 ten 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.

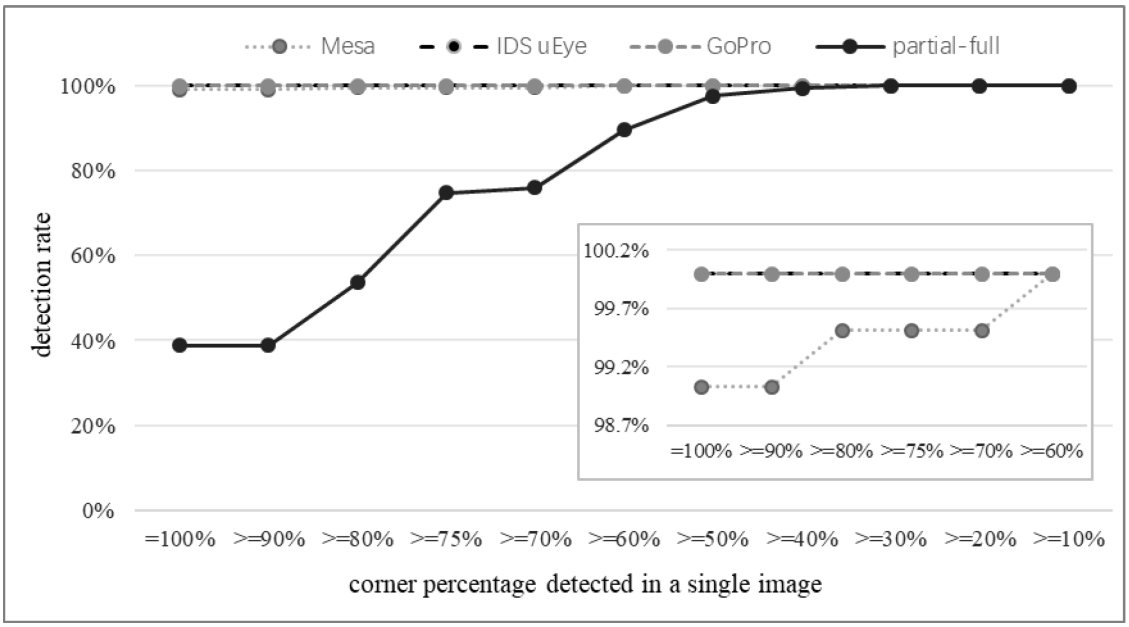

Fig. 9 Detection rate when different corner percentages allowed. The smaller figure in the bottom right box shows the overlapped lines in the top left part of the whole figure.

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 more than 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 MATLAB and use the CPU. The four important parameters of the proposed method are set the same as explained in Subsection 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 one second 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 one minute, while it only needs 10.4 seconds with the proposed method. Overall, the proposed method is the fastest with an average runtime being 3.62 seconds.

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**

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 etc. In this subsection, representative results are presented as shown in Fig. 10.

In Fig. 10 (a–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). Fig. 10 (e–f) record checkerboards projected on a white wall with strong geometric distortion. The ones in Fig. 10 (g–j) are also significantly distorted, but there are some differences in them—(g), (h) and (i) are weakly blurred, and the checkerboards appear on different media, while (j) is a bit noisy. In Fig. 10 (k) and (l), some parts of the checkerboards are covered with some projected characters, which may affect the detection process. Fig. 10 (m) shows a complex circumstance. Fig. 10 (n) and (o) contain multiple checkerboards, and the former is taken by a fisheye camera, while 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.

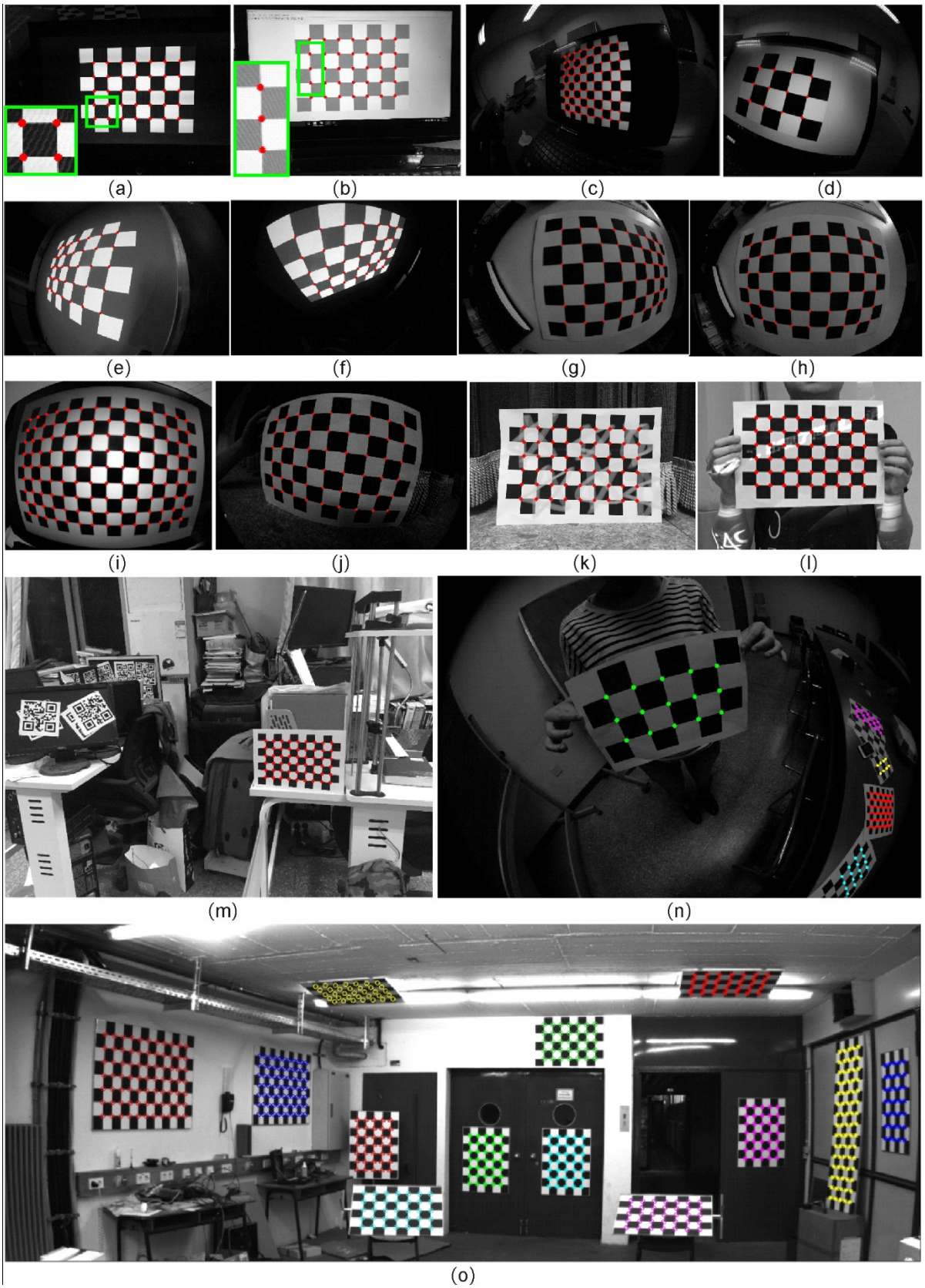

Fig. 10 Detection results. (a–d) Checkerboards shown on screen. (e-f) Checkerboards projected on a white wall. (g-i) Weakly blurred checkerboards with strong fisheye distortion. (j) Checkerboards in a noisy image. (k-l) Checkerboards with some slightly shaded areas. (m) A complex circumstance. (n-o) Multiple checkerboards in a single 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. Since 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, while 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 image coordinates of the corners can be calculated later by reprojection using Eq. 12 with the already known three-dimensional coordinates. Then the reprojection error can be calculated using Eq. 13.

$$s\tilde{\boldsymbol{x}} = \boldsymbol{K}\begin{bmatrix}\boldsymbol{R} & \boldsymbol{T}\end{bmatrix}\tilde{\boldsymbol{X}} \tag{12}$$

Where $\boldsymbol{K}$ is the intrinsic matrix, $\boldsymbol{R}$ and $\boldsymbol{T}$ are the rotation matrix and the translation matrix, $\tilde{\boldsymbol{X}}$ denotes the homogenous three-dimensional coordinates of the corners, $\tilde{\boldsymbol{x}}$ denotes

the reprojected homogenous two-dimensional image coordinates of the corners, and s is a scalar factor.

$$e_{reprojection} = \frac{1}{N}\sum \left\| \boldsymbol{x}_i' - \boldsymbol{x}_i \right\|_2 \tag{13}$$

Where N is the number of corners, i is the index of a corner, $\boldsymbol{x}_i'$ denotes the detected two-dimensional image coordinates of the corners and $\boldsymbol{x}_i$ denotes the calculated two-dimensional image coordinates.

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**

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 three-dimensional 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 five seconds, which is even shorter (less than one second) 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]; 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 Journal on Robotics and Automation, 3(4), 323-344 (1987).

[2] Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on pattern analysis and machine intelligence, 22(11), 1330-1334 (2000).

[3] J. Y. Bouguet, "Camera Calibration Toolbox for Matlab," 14 October 2015 (accessed 24 August 2017). [http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#examples]

[4] V. Vladimir, "OpenCV calibration object detection," 24 August 2017 (accessed 24 August 2017). [http://graphicon.ru/oldgr/en/research/calibration/opencv.html]

[5] A. de la Escalera, and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," Sensors (Basel), 10(3), 2027-44 (2010).

[6] J. Chu, A. GuoLu, and L. Wang, "Chessboard corner detection under image physical coordinate," Optics & Laser Technology, 48, 599-605 (2013).

[7] S. Bennett, and J. Lasenby, "ChESS – Quick and robust detection of chess-board features," Computer Vision and Image Understanding, 118, 197-210 (2014).

[8] S. Bennett, and J. Lasenby, "Robust recognition of chess-boards under deformation," 20th IEEE International Conference on Image Processing (ICIP), 2650-2654 (2013).

[9] S. Placht, P. Fürsattel, E. A. Mengue *et al.*, "Rochade: Robust checkerboard advanced detection for camera calibration," 13th European Conference on Computer Vision (ECCV), 766-779 (2014).

[10] P. Fuersattel, S. Dotenco, S. Placht *et al.*, "OCPAD – Occluded Checkerboard Pattern Detector," 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), 1-9 (2016).

[11] Y. Bok, H. Ha, and I. S. Kweon, "Automated checkerboard detection and indexing using circular boundaries," Pattern Recognition Letters, 71, 66-72 (2016).

[12] A. Geiger, F. Moosmann, C. Ö *et al.*, "Automatic camera and range sensor calibration using a single shot," 2012 IEEE International Conference on Robotics and Automation (ICRA), 3936-3943 (2012).

[13] C. Harris, and M. Stephens, "A combined corner and edge detector," Alvey Vision Conference, 10-5244 (1988).

[14] A. Geiger, P. Lenz, C. Stiller *et al.*, "Vision meets robotics: The KITTI dataset," The International Journal of Robotics Research, 32(11), 1231-1237 (2013).

[15] S. M. Smith, and J. M. Brady, "SUSAN—A new approach to low level image processing," International Journal of Computer Vision, 23(1), 45-78 (1997).

[16] D. G. Lowe, "Object recognition from local scale-invariant features," 7th IEEE International Conference on Computer Vision (ICCV), 1150-1157 (1999).

[17] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, 60(2), 91-110 (2004).
[18] E. Rosten, and T. Drummond, ″Machine learning for high-speed corner detection,″ 9th European Conference on Computer Vision (ECCV), 430-443 (2006).
[19] A. Neubeck, and L. Van Gool, ″Efficient non-maximum suppression,″ 18th International Conference on Pattern Recognition (ICPR), 850-855 (2006).
[20] P. Kovesi, “Non-maximum suppression,” 24 August 2017 (accessed 24 August 2017). [http://www.peterkovesi.com/matlabfns/Spatial/nonmaxsuppts.m]
[21] L. Lucchese, and S. K. Mitra, ″Using saddle points for subpixel feature detection in camera calibration targets,″ Asia-Pacific Conference on Circuits and Systems, 191-195 vol.2 (2002).
[22] L. Krüger, and C. Wöhler, “Accurate chequerboard corner localisation for camera calibration,” Pattern Recognition Letters, 32(10), 1428-1435 (2011).
[23] MathWorks, “Camera Calibrator,” 24 August 2017 (accessed 24 August 2017). [https://cn.mathworks.com/help/vision/ref/cameracalibrator-app.html]

Yizhen Yan received her B.S. degree in geographic information science from East China Normal University in 2015. She is currently a M.S. candidate of Peking University. Her research interests include image processing, feature detection, light field imaging and stereo.

Peng Yang received his B.S. degree in remote sensing science and technology from Beihang University in 2013. He is currently a Ph.D. candidate in Peking University. His research interests include calibration, light field imaging, and stereo.

Lei Yan received his B.S. degree from Nanjing University of Aeronautics & Astronautics in 1981, the M.S. degree from Navy University of Engineering in 1989, and the Ph.D. degree from Tsinghua University in 1994. He is now a Professor in School of Earth and Space Sciences in Peking University and the head of Spatial Information Integration & 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 B.S. degree from Wuhan University in 2011 and his M.S. degree from University of Chinese Academy of Sciences in 2014. He is currently a Ph.D. candidate in

Peking University. His research interests include structure from motion, stereo and deep learning.

Yanbiao Sun received his B.S. degree from Wuhan University in 2010 and his Ph.D. degree from Peking University. He is now a post-doctoral fellow in University College London. His research interests include feature extraction and matching, photogrammetry, machine learning and deep learning.

Kevin Tansey is 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 editor of the International Journal of Remote Sensing and the MDPI Open Access journal Fire.

Anand Asundi received his Ph.D. 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 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 B.S. degree from Changchun University of Technology, China in 1993, where she also received her M.S. degree in1998. She received the Ph.D. degree from University of Chinese Academy of Sciences in 2002. She is currently an associate professor of Peking University. Her research interests include photogrammetry, remote sensing and light field imaging.

Caption List

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) Non-maximum suppression on the self-correlation map. (g) Result after outlier discarding and refinement.

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.

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 non-maximum 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 non-maximum suppression. After the steps, the corner clusters are processed into single corners with more accurate positions, such as the one in the large circle.

Fig. 5 Steps of checkerboard structure recovery.

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, bottom right corners are found in turn. (b) and (c) show a number of parameters that will be used in the detection of initial corners.

Fig. 7 Structure expansion in one direction.

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.

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-f) Checkerboards projected on a white wall. (g-i) Weakly blurred checkerboards with strong fisheye distortion.

(j) Checkerboards in a noisy image. (k-l) Checkerboards with some slightly shaded areas. (m) A complex circumstance. (n-o) Multiple checkerboards in a single image.

Table 1 Recommended font sizes and styles.

Table 1 Four important parameters in checkerboard corner detection.

Table 2 Simple summarization of the public datasets.

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

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

Table 5 Calibration accuracy using different methods.