Pattern Recognition Letters 71 (2016) 66–72

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Automated checkerboard detection and indexing using circular boundaries☆

Yunsu Bok[a], Hyowon Ha[b,*], In So Kweon[b]

[a] Division of Future Vehicle, KAIST, Daejeon 34141, Republic of Korea
[b] Department of Electrical Engineering, KAIST, Daejeon 34141, Republic of Korea

ARTICLE INFO

Article history:
Received 3 July 2015
Available online 30 December 2015

Keywords:
Checkerboard detection
Camera calibration
Corner indexing
Partial view

ABSTRACT

This paper presents a new algorithm for automated checkerboard detection and indexing. Automated checkerboard detection is essential for reducing user inputs in any camera calibration process. We adopt an iterative refinement algorithm to extract corner candidates. In order to utilize the characteristics of checkerboard corners, we extract a circular boundary from each candidate and find its sign-changing indices. We initialize an arbitrary point and its neighboring two points as seeds and assign world coordinates to the other points. The largest set of world-coordinate-assigned points is selected as the detected checkerboard. The performance of the proposed algorithm is evaluated using images with various sizes and particular conditions.

1. Introduction

Camera calibration is an essential process for various purposes in the areas of computer vision and robotics. It determines the relation (one-to-one correspondence) between pixels of images captured by a camera and ray directions in the camera coordinate system. In other words, the (geometric) camera calibration is the process of decoding the physical meaning of pixels. Specifically, it calculates intrinsic and extrinsic parameters of the cameras' projection model. Most cameras are assumed to follow the characteristics of the pinhole projection model. A large number of studies of camera calibration under the pinhole model have been published [8,16,18,20]. The method proposed by Zhang [20] is most popular because of its efficiency and robustness.

Most camera calibration methods require a sufficient number of correspondences between world coordinates and image coordinates, which are also known as control points. They utilize images of two- or three-dimensional objects with known geometries. The objects contain patterns on their surfaces, such as checkerboard patterns or circular patterns, which are used to extract features with high accuracy. Most researchers use two-dimensional checkerboard patterns. Assuming that the size of a checkerboard pattern is known, the world-image correspondences can be calculated by accurately extracting the corners of the patterns in images using various techniques.

This paper focuses on the 'automatic' extraction of pattern features from images. In the whole process of camera calibration, the only remaining part that requires user input is feature detection and indexing. Recently, many researchers have started to use a toolbox implemented in MATLAB [1], which requires four points that indicate a checkerboard area with a known number of squares. Moreover, images that contain only a part of the pattern must be discarded. Other open sources work in a similar way.

A number of studies related to automated checkerboard detection have been published. Yu and Peng [19] inserted double-triangle figures in a checkerboard pattern and detected them as a reference. The performance of their method depends on the success ratio of detecting the double-triangle figures. Wang et al. [17] extracted Harris corners [7] and convolved rotating orthogonal masks. Their method may fail in extremely slanted poses and severe radial distortion. Rufli et al. [13] binarized images using an adaptive threshold and detected quadrangles (four-sided polygons). Their method sometimes misses several quadrangles. Fiala and Shu [4] adopted two-dimensional binary markers to assign world coordinates automatically. The indexing ability of this method is good, but its localization accuracy is questionable. De la Escalera and Armingol [3] applied a Hough transform to Harris corners to detect the edge lines of a checkerboard. Geiger et al. [6] computed the corner likelihood of each pixel by comparing two types of corner prototypes; they expanded the seed points into the directions of strong gradients. This algorithm works well even with

☆ This paper has been recommended for acceptance by Dr. A. Koleshnikov.

* Corresponding author. Tel.: +82 423505465.
E-mail addresses: ysbok@rcv.kaist.ac.kr (Y. Bok), hwha@rcv.kaist.ac.kr (H. Ha), iskweon@kaist.ac.kr (I.S. Kweon).

http://dx.doi.org/10.1016/j.patrec.2015.12.008
0167-8655/

large distortion, but it can perform detection only if whole views of the checkerboards are included in the images, not partial views. Placht et al. [11] detected corners by extracting edges using Scharr kernels and checking the connections of the edges' skeletons. The success ratio of this method depends on the edge detection results obtained using the Scharr kernel. Recently, OpenCV also provided a function that detects checkerboard corners automatically. Most of the related works mentioned above have certain weaknesses, such as radial distortion or partial views of the checkerboard. Even recent works [6,11] fail for certain extreme views.

In this paper, we present a simple and efficient algorithm for automated checkerboard detection and indexing. Instead of corner detectors such as Harris corner [7], KLT [15] or FAST [12], we apply an iterative refinement algorithm to the entire region of images to find the candidates of checkerboard corners without loss. We then extract the circular boundary of every candidate and discard outliers using the characteristics of checkerboard corners. A checkerboard is detected automatically using the results of the automated indexing algorithm. The performance of the proposed algorithm is evaluated using images with various sizes. Because the main objective of the proposed algorithm is automated detection and indexing, the accuracy of the detected corners is limited to that of the iterative refinement algorithm. However, this method may be used to determine the initial locations of other refinement algorithms [2,9,10].

2. Checkerboard corner extraction

2.1. Corner search by iterative refinement

Checkerboard corners can be detected using conventional algorithms, such as Harris corner [7] or KLT [15]. However, these algorithms usually extract multiple points from one corner or miss a number of corners due to user-defined parameters. Instead of extracting feature points, we adopt an iterative refinement algorithm that makes arbitrary points converge into nearby corners.

We adopt the sub-pixel corner finder algorithm implemented in [1]; this algorithm is based on Harris corner [7]. From the given initial corner locations, the algorithm iteratively updates the individual corner locations to the largest gradient values using patch-based structure tensor calculation. Let $G_x^{(i,j)}$ and $G_y^{(i,j)}$ be the image gradients at the location (i, j) along the x- and y-directions, respectively. The updated corner location $(i+\Delta i, j+\Delta j)$ can be computed as follows:

$$e(\Delta i, \Delta j) \equiv I(i+\Delta i, j+\Delta j) - I(i, j) \approx G_x^{(i,j)}\Delta x + G_y^{(i,j)}\Delta j \quad (1)$$

$$\begin{aligned} E(\Delta i, \Delta j) &= \sum_i \sum_j w^{(i,j)} e(\Delta i, \Delta j)^2 \\ &\approx \sum_i \sum_j w^{(i,j)} (G_x^{(i,j)}\Delta i + G_y^{(i,j)}\Delta j)^2 \end{aligned} \quad (2)$$

$$\begin{aligned} &\begin{bmatrix} i+\Delta i \\ j+\Delta j \end{bmatrix} \\ &= \begin{bmatrix} \sum_i \sum_j w^{(i,j)} G_x^{(i,j)} G_x^{(i,j)} & \sum_i \sum_j w^{(i,j)} G_x^{(i,j)} G_y^{(i,j)} \\ \sum_i \sum_j w^{(i,j)} G_x^{(i,j)} G_y^{(i,j)} & \sum_i \sum_j w^{(i,j)} G_y^{(i,j)} G_y^{(i,j)} \end{bmatrix}^{-1} \\ &\times \left(\sum_i \sum_j \begin{bmatrix} w^{(i,j)} G_x^{(i,j)} G_x^{(i,j)} \cdot i + w^{(i,j)} G_x^{(i,j)} G_y^{(i,j)} \cdot j \\ w^{(i,j)} G_x^{(i,j)} G_y^{(i,j)} \cdot i + w^{(i,j)} G_y^{(i,j)} G_y^{(i,j)} \cdot j \end{bmatrix} \right), \end{aligned} \quad (3)$$

where $w^{(i,j)}$ is the weight of the location (i, j), which is defined as the Gaussian weight.

However, when the number of corners is large, the structure tensor calculation becomes the bottleneck of this process due to many gradient calculations that are required for every sampled patch. We modify this algorithm to calculate the structure tensor by directly interpolating the gradients instead of first interpolating the image and second computing the gradients. This modification reduces the computational burden ten times; the results, however, are mathematically identical.

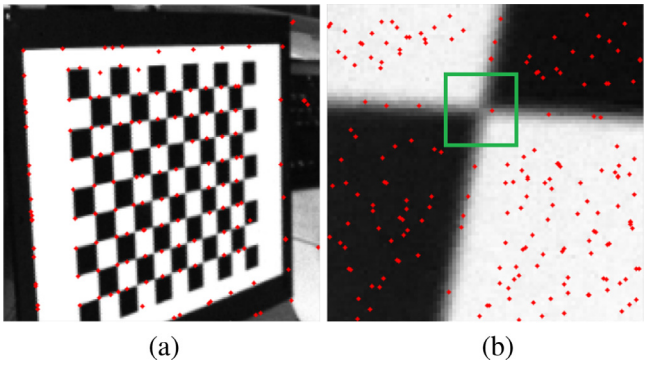

Fig. 1. Examples of inappropriate window sizes. (a) Extraction results for a small pattern using the window size of 17 × 17 pixels. (b) Extraction results for a blurred corner using the window size of 5 × 5 pixels (inside the green box). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

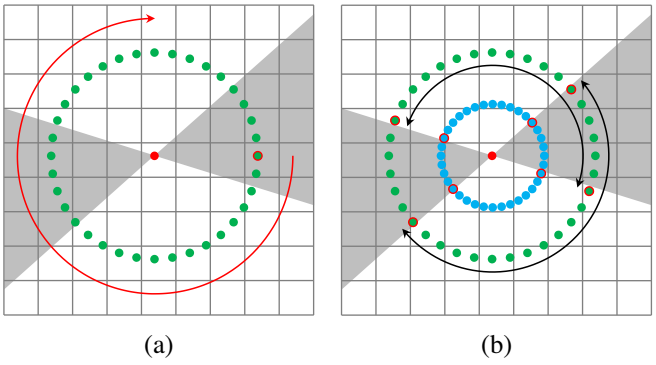

Fig. 2. (a) Example of circular boundaries with 10-degree intervals. The green dots compose the circular boundary of the corner depicted by the red dot. They are reshaped into a $1 \times n$ vector in clockwise order, starting from the pixel at the right center (green dot with red boundary). (b) The characteristics of the circular boundary (green dots). This boundary has four sign-changing indices (dots with red boundaries). The differences (black arrows) between the opposite sign-changing indices are similar to the half-length of the boundary vector. This boundary vector's sign-changing indices are similar to those of a boundary vector with a smaller radius (blue dots). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The performance of the iterative refinement algorithm depends on the window size. A small pattern cannot be detected using a large window, and a blurred pattern cannot be detected using a small window, as shown in Fig. 1. Because we do not know the proper window size for an arbitrary target image, we extract corner candidates using a number of different window sizes. Duplicated candidates extracted from an identical corner using different window sizes are discarded, as detailed in Section 2.2.

2.2. Discarding candidates using circular boundary

The key idea of the proposed algorithm is to utilize the circular boundaries – the list of pixels with the same distances – of checkerboard corners. As shown in Fig. 2(a), we extract n pixels of a circular boundary centered at the corner candidate. Because the locations of the boundary pixels are computed at a sub-pixel level, the pixel intensities are computed using the bilinear interpolation. After computing the intensities, the pixels are reshaped into