Fast drawing method of circular patterns based on Gaussian circles for camera calibration

Soshi Asamura, Yuki Naganawa, and Norishige Fukushima

Nagoya Institute of Technology, Japan

ABSTRACT

Camera calibration is a fundamental technology for image measurement. Calibration using 2D planar patterns is the de facto standard, and is performed by displaying and detecting patterns such as a circle pattern or a checkerboard pattern, and calibrating based on the correspondence between the coordinates of the patterns. Recently, the patterns are often displayed on a liquid crystal display (LCD). While many previous papers have focused on the detection method, this paper focuses on the display method. This is because LCDs have coarser dots per inch than printing devices. In this paper, we propose a method for drawing circular patterns on LCD that accurately and fast detects feature points based on the Gauss circle problem. Experimental results show that the proposed method is 4 times more accurate and 2.5 times faster than previous drawing methods.

Keywords: Camera calibration, Anti-aliased circle drawing, Gauss circle problem

1. INTRODUCTION

Camera calibration plays a fundamental role in 3D computer vision. The ray geometry entering the camera is represented by a position in 3D coordinates and a position in image projection coordinates, which are related using the intrinsic and extrinsic parameters of the camera, respectively. The process of obtaining this is called camera calibration. These parameter estimations were performed using a single camera,[1] a stereo camera,[2] a depth sensor with structured light[3] and unstructured light,[4, 5] camera arrays.[6, 7] In addition, accurate camera parameters are essential from the classical 2D image to 3D image generation[8, 9] to the recent Gaussian splattering.[10]

3D or 2D patterns with known relative positions are often used for camera calibration. The camera parameters are obtained by correspondence between 3D coordinate points and their 2D projected points onto the image. Since it is easier to create 2D patterns than precise 3D patterns, calibration via 2D patterns is often used today. The widely used Zhang's method[1] is a calibration that uses 2D patterns and requires taking 2D patterns in multiple positions and orientations. Usually, this pattern is made of printed paper attached to a flat surface. However, a board manufactured exclusively for this purpose is preferable to ensure printing accuracy and flatness[*], [†]. On the other hand, printing paper and sticking it to a flatboard is more convenient. However, since flatness is not guaranteed, the calibration accuracy tends to be low.

Patterns also have two types: finding the intersection of edges or the center of gravity. The types of intersection of edges include checkerboard (square) ,[11] triangle,[12] star,[13] ArUco.[14] The types of the center of gravity include circle[15] and the ring.[16] The checkerboard detection algorithm is one of the most widely used algorithms. It detects corners by dividing a rectangle and finding the intersection of each edge. Wang et al.[17] proposed a method of fitting two groups of lines to find a pattern grid, De et al. and Hansard et al. proposed finding grid lines using the Hough transform.[18, 19] However, such line-fitting methods are susceptible to lens distortion. For this reason, Rufli et al. proposed a robust algorithm against strong lens distortion implemented in OCamCalib for MATLAB.[20] Highly accurate algorithms for finding the intersection of edges have also been proposed in the triangular pattern[12] and star pattern[13] detections. In general, patterns that find the center of gravity are superior to patterns that find the intersection of edges from triangles, squares, and other shapes in terms of detection accuracy. For circular patterns, corrected conic fitting,[21] confocal conics,[22] and concentric circles[23] have been proposed to remove bias under the projective transformation.

Corresponding author, N. Fukushima (E-mail: fukushima@nitech.ac.jp)

[*]https://calib.io/

[†]https://www.shibuya-opt.co.jp/eng/calibration_board.html

Recently, the method of projecting a pattern on a liquid crystal display (LCD) has been introduced instead of pattern printing due to its ease of preparation.[24,25] However, the LCD dots are large; thus, there is no escape from the quantization error of pattern printing. In particular, since differences in luminance values are essential for detecting the center of gravity, the shading of the circle boundary differs depending on how the circle is drawn, and these differences affect the accuracy of pattern coordinate detection. Most papers have focused on improving the detection accuracy of drawn patterns but have not paid much attention to how to draw them.

In this paper, we propose a fast and accurate method to draw patterns generated on the LCD. In particular, we focus on circle patterns. The midpoint circle algorithm (i.e., a generalization of Bresenham's algorithm[26]) is a method for drawing circles on discrete coordinates. Its anti-aliasing method has also been proposed[27] to reduce the quantization effect. A more straightforward method that reduces the effect of aliasing is to draw a circle on an image with a resolution higher than the display resolution and then downsample it to determine the pixel values, but such drawing methods are cost-consuming. In this paper, we propose a method for determining pixel values by approximating the area of each pixel of a circle using the Gauss circle problem.[28]

2. METHODOLOGY

The Gauss circle problem is required to solve the number of grid points inside a circle of radius r centered at the origin, and the circle area approximates the number of points. Therefore, measuring the number of grid points inside the circle can determine the pixel value. In this paper, we investigate a method for drawing circular patterns based on the Gaussian circle problem. First, we describe a method for drawing a unit circle, then extend it to drawing an arbitrary coordinate containing a projectively transformed ellipse, and finally, we describe a method for drawing multiple circles and ellipses.

2.1 Unit circle drawing

First, we describe how to draw a unit circle. In this paper, we assume the drawing of a unit circle with center $(1, 1)$ and radius 1 when the circle is painted black on a white background.

Let $(i, j) \in \mathbb{N}^2$ be an integer pixel position in image v, $(x, y) \in \mathbb{R}^2$ be a floating-point coordinate, and $(m, n) \in \mathbb{R}^2$ be a floating-point coordinate relative to the top left of the integer pixel separated by a grid point. Considering the Gauss circle problem, the intensity value of image v can be determined by substituting the coordinates of each grid point in all pixels into the circle equation $q(x, y) = (x - 1)^2 + (y - 1)^2 - 1^2$ to determine whether the grid point is inside or outside the circle. Then, we can obtain the intensity as the ratio of the number of grid points counted inside to the number of grid points $N \in \mathbb{N}$ in a pixel. Here, the value is a float type $[0, 1]$. The intensity $v(i, j) \in \mathbb{R}$ of each pixel in the output image $v \mapsto [0, 1]$ is determined by the following formula using the number of all grid points N at that pixel and the number of grid points inside the circle $c(i, j) \in \mathbb{N}$.

$$v(i, j) = 1 - \frac{c(i, j)}{N}, \tag{1}$$

where

$$c(i, j) = \sum_{m=0}^{\sqrt{N}} \sum_{n=0}^{\sqrt{N}} \delta_{i,j}(m, n) \tag{2}$$

$$\delta_{i,j}(m, n) = \begin{cases} 1 & q\left(i + m \cdot h, j + n \cdot h\right) \leq 0 \\ 0 & \text{else} \end{cases} \tag{3}$$

$$h = \frac{1}{\sqrt{N} - 1}, \quad q(x, y) = (x - 1)^2 + (y - 1)^2 - 1^2 \tag{4}$$

The four vertices $\{(i, j), (i, j + 1), (i + 1, j), (i + 1, j + 1)\}$ at the corners of the pixel of interest in the output image can be checked to determine if they are completely inside or not. If all four points are inside the circle, the pixel value is 0; if all the grid points are outside the circle, the pixel value is 1. By skipping calculations this way, the speed can be increased without losing accuracy.

2.2 Ellipse drawing

Next, the unit circle drawing is extended to an arbitrary ellipse drawing. Given a homography matrix $\boldsymbol{H}$ that projects an ellipse to a unit circle, the relation between the coordinates x, y on the ellipse and the coordinates X, Y on the unit circle is as follows using the homogeneous coordinate system.

$$\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \sim \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{5}$$

Note that $x = i + m \cdot h, y = j + n \cdot h$. Expanding, we obtain

$$\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} \dfrac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \\ \dfrac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \\ 1 \end{pmatrix}. \tag{6}$$

Substituting (6) into the unit circle equation $X^2 + Y^2 = 1$, we obtain

$$\left(\frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \right)^2 + \left(\frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \right)^2 = 1. \tag{7}$$

Expanding this, the equation of the ellipse that is equal to 0 is as follows

$$\begin{aligned} &\left(h_{11}^2 + h_{21}^2 - h_{31}^2\right) x^2 + 2\left(h_{11}h_{12} + h_{21}h_{22} - h_{31}h_{32}\right) xy \\ &\quad + \left(h_{12}^2 + h_{22}^2 - h_{32}^2\right) y^2 + 2\left(h_{11}h_{13} + h_{21}h_{23} - h_{31}h_{33}\right) x \\ &\qquad +2\left(h_{12}h_{13} + h_{22}h_{23} - h_{32}h_{33}\right) y + \left(h_{13}^2 + h_{23}^2 - h_{33}^2\right) = 0 \end{aligned} \tag{8}$$

The left side of this equation is $q(x, y)$, and the same process as for the unit circle drawing is used to determine the interior and exterior of the grid points in (3).

2.3 Multiple circles and ellipses drawing

A pattern usually contains multiple circles. There are two types of patterns: symmetrical circular patterns (regular grid) and asymmetrical circular patterns (hexagonal grid), both of which can be rendered by the proposed algorithm. In this section, we describe the asymmetric circular pattern. If a symmetric circular pattern is drawn, the radius must be smaller than the grid, or the circles will overlap.

First, the coordinates of multiple circles are specified by the coordinates of the four vertices of the quadrangles that are circumscribed by those circles. For example, draw a circle inscribed in each quadrilateral shown in Fig. 1(a). In the case of a tilted projection of the pattern, as shown in Fig. 1(b), the four vertex coordinates of the convex quadrilateral need not be squares. Note that although the visualization is done with a checkerboard, only the coordinate values are input in reality.

For each drawing circle, the homography $\boldsymbol{H}$ is computed from the coordinates of the four vertices of the adjacent convex quadrilateral and the four vertices of the unit square, and the ellipse equation is derived and then drawn using the method in Sec. 2.2. Multiple circles can be drawn in sequence for all convex quadrilaterals. The pattern and an enlarged circle drawn by the proposed method are shown in Fig. 2.

3. EXPERIMENTAL RESULTS

We verified the accuracy of circle center of gravity detection and drawing time for a circular pattern drawing method using the Gauss circle problems. The computer used in the experiments was Intel Core i9-11900K 3.50-5.30 GHz (8 cores, 16 threads), and the compiler was Visual Studio 2022. A planar pattern with 7×6 asymmetric circles (shown in Fig. 2) was generated in a 512×512 image.

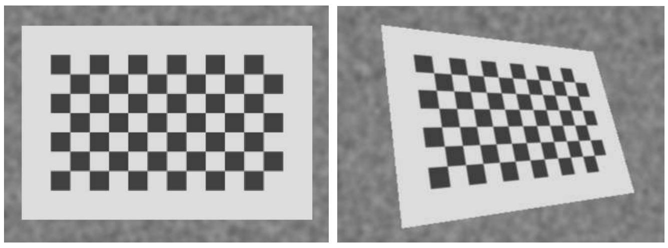

(a) front (b) tilt

Figure 1: Drawing coordinate specification (visualization by a checkerboard).

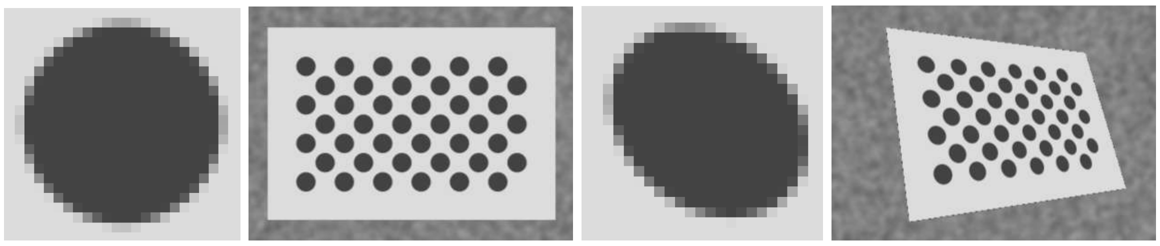
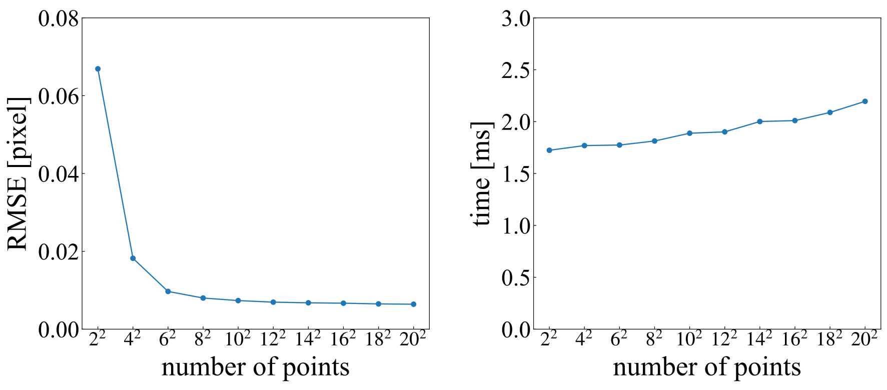

(a) enlarged circle (b) front circle pattern (c) enlarged ellipse (d) tilted circle pattern

Figure 2: Drawing circular pattern and enlarged view.



(a) (b)

Figure 3: (a) accuracy of circle center of gravity detection to the number of grid points, (b) pattern generation time to the number of grid points.

3.1 Effects of the number of grid points

First, we verified the accuracy of detecting the circle center of gravity and the pattern generation time (median time required to generate one pattern) to the number of grid points N in a pixel. The proposed method generates circular patterns, and detection is performed on patterns rotated in the two-dimensional direction. The root mean squared error (RMSE) between the coordinates of the center of gravity of the circle and the ideal coordinates was used for evaluation. 100 patterns were generated at random rotation angles. The OpenCV function *cv::findCirclesGrid* was used for detection.

Figure 3(a) shows the accuracy results, and Fig. 3(b) shows the speed results. RMSE decreases as the number of grid points increases, and the slope of the graph decreases when the number of grid points exceeds 8^2. The number of grid points increases proportionately to the computational time required to determine whether a grid point is inside or outside a circle.

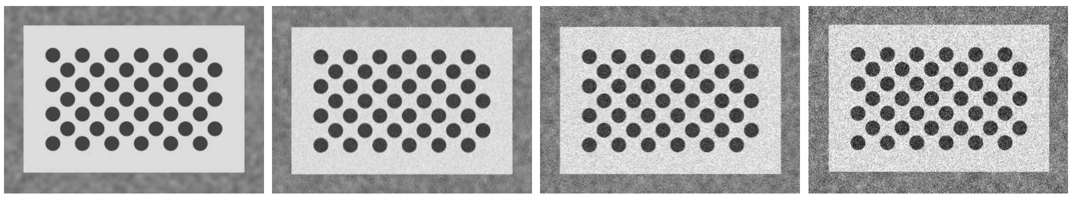

(a) $\sigma = 0$ (b) $\sigma = 10$ (c) $\sigma = 20$ (d) $\sigma = 30$

Figure 4: Drawing circular patterns with changing Gaussian noise levels.

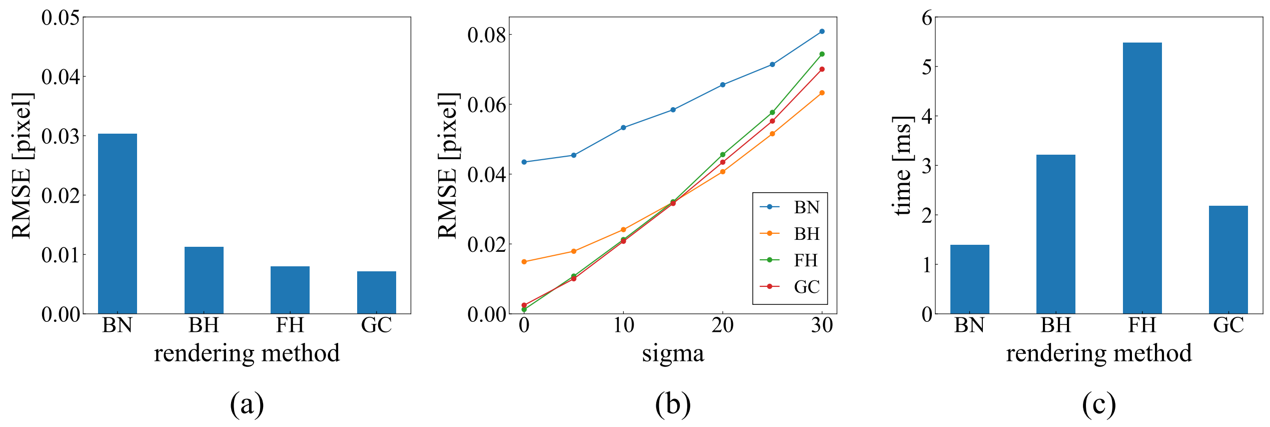


Figure 5: (a) detection accuracy ($\sigma = 0$), (b) accuracy with different noise levels, and (c) generation time. BN: backward normal-resolution, BH: backward high-resolution, FH: forward high-resolution, GC: Gauss circle.

3.2 Comparison with competitive methods

Next, we compared the detection accuracy and drawing time between the proposed and competitive methods. The accuracy of detecting the center of gravity of a circle was also verified by changing the Gaussian noise level. The patterns with adding noises are shown in Fig. 4. The following four methods, including the proposed method, are compared. Experimental conditions are the same as in the number of grid points experiment.

backward normal-resolution (BN) A regular circle is drawn with the OpenCV function *cv::circle* to normal resolution image (765×510) and then projected in the backward direction with *cv::warpPerspective*.

backward high-resolution (BH) A regular circle is drawn with the function *cv::circle* at a resolution higher than the display resolution (1530×1020), scaled down to the display resolution, and then projected backward with the function cv::warpPerspective.

forward high-resolution (FH) A regular circle is drawn with the function *cv::circle* at a resolution higher than the display resolution (1530×1020) and then performs a forward projection instead of a backward. The forward projection may cause a hole in the image, but the transformation from a resolution high enough to avoid a hole to a lower resolution. The output is the sum of pixel values spread by linear interpolation from integer pixels to floating-point coordinates of the projection destination.

Gauss circle (GC) Proposed method with the number of grid points per pixel to $N = 20^2$.

The noise-free results for each method are shown in Fig. 5(a). The drawing method with the lowest RMSE is the proposed GC. The proposed method is about 4 times more accurate than the baseline of BN.

Next, the accuracy for adding Gaussian noise is shown in Fig. 5(b). Note that a noise level of σ greater than 25 may cause undetectable cases. The slope of BN and BH is slightly lower than that of FH and GC, and

RMSE of BH is inverted from a certain noise level. This indicates that the noise has a less significant effect on the shading of circles that are not accurately rendered than on the shading of circles that are accurately rendered. However, if multiple exposures significantly suppress the noise level, a high-precision rendering such as the proposed method will produce better results.

Finally, the generation time results are shown in Fig. 5(c). The proposed Gauss circle method had a minor difference in detection accuracy from the forward high-resolution method, which has the next best accuracy; however, the proposed method was about 2.5 times faster.

4. CONCLUSIONS

In this paper, we generated circular patterns based on the Gauss circle problem and compared them with other methods. The proposed method can draw ellipses without projective transformation of the image; thus, the circle's boundary is not blurred. When the number of grid points per pixel reaches 8^2, the accuracy of circle center of gravity detection reaches a ceiling; however, the pattern generation time increases proportionally to the number of grid points. The proposed method is the best in accuracy and generation speed compared to other methods. The proposed method is greatly affected by noise because the shading of the circles is rendered relatively accurately. Therefore, it is recommended that the proposed method be used with noise reduction. In this verification, *cv::findCirclesGrid*, an OpenCV function, was used as the circle detector. However, the accuracy is expected to be improved by creating a circle detector suitable for the proposed method.

ACKNOWLEDGMENTS

This research was supported by JSPS KAKENHI Grant Number JP21H03465, JP24K02960 and Environment Research and Technology Development Fund (JPMEERF20222M01) of the Environmental Restoration and Conservation Agency of Japan. We thank Kowa Optronics Co., Ltd. for the discussions.

REFERENCES

[1] Zhang, Z., "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(11), 1330–1334 (2000). https://doi.org/10.1109/34.888718.

[2] Weng, J., Cohen, P., and Herniou, M., "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(10), 965–980 (1992). https://doi.org/10.1109/34.159901.

[3] Zhang, Z., "Microsoft kinect sensor and its effect," *IEEE MultiMedia* **19**(2), 4–10 (2012). https://doi.org/10.1109/MMUL.2012.24.

[4] Keselman, L., Iselin Woodfill, J., Grunnet-Jepsen, A., and Bhowmik, A., "Intel realsense stereoscopic depth cameras," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1–10 (2017).

[5] Takeda, J. and Fukushima, N., "Poisson disk sampling with randomized satellite points for projected texture stereo," *Optics Continuum* **1**(5), 974–988 (2022). https://doi.org/10.1364/OPTCON.451197.

[6] Wilburn, B., Joshi, N., Vaish, V., Talvala, E.-V., Antunez, E., Barth, A., Adams, A., Horowitz, M., and Levoy, M., "High performance imaging using large camera arrays," *ACM Transactions on Graphics* **24**(3), 765–776 (2005). https://doi.org/10.1145/1073204.1073259.

[7] Fukushima, N., Yendo, T., Fujii, T., and Tanimoto, M., "A novel rectification method for two-dimensional camera array by parallelizing locus of feature points," in *Proceedings of International Workshop on Advanced Imaging Technology (IWAIT)*, (2008).

[8] Mori, Y., Fukushima, N., Yendo, T., Fujii, T., and Tanimoto, M., "View generation with 3d warping using depth information for ftv," *Signal Processing: Image Communication* **24**(1), 65–72 (2009). https://doi.org/10.1016/j.image.2008.10.013.

[9] Matsuo, T., Fukushima, N., and Ishibashi, Y., "Weighted joint bilateral filter with slope depth compensation filter for depth map refinement," in *Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP)*, 300–309 (2013). https://doi.org/10.5220/0004292203000309.

[10] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G., "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics* **42**(4) (2023). https://doi.org/10.1145/3592433.

[11] Bradski, G., "The opencv library.," *Dr. Dobb's Journal: Software Tools for the Professional Programmer* **25**(11), 120–123 (2000).

[12] Ha, H., Perdoch, M., Alismail, H., Kweon, I. S., and Sheikh, Y., "Deltille grids for geometric camera calibration," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 5354–5362 (2017). https://doi.org/10.1109/ICCV.2017.571.

[13] Schöps, T., Larsson, V., Pollefeys, M., and Sattler, T., "Why having 10,000 parameters in your camera model is better than twelve," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2532–2541 (2020). https://doi.org/10.1109/CVPR42600.2020.00261.

[14] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M., "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition* **47**(6), 2280–2292 (2014). https://doi.org/10.1016/j.patcog.2014.01.005.

[15] Heikkila, J., "Geometric camera calibration using circular control points," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(10), 1066–1077 (2000). https://doi.org/10.1109/34.879788.

[16] Datta, A., Kim, J.-S., and Kanade, T., "Accurate camera calibration using iterative refinement of control points," in *Proceedings of IEEE International Conference on Computer Vision Workshops (ICCVW)*, 1201–1208 (2009). https://doi.org/10.1109/ICCVW.2009.5457474.

[17] Wang, Z., Wu, W., Xu, X., and Xue, D., "Recognition and location of the internal corners of planar checkerboard calibration pattern image," *Applied Mathematics and Computation* **185**(2), 894–906 (2007). https://doi.org/10.1016/j.amc.2006.05.210.

[18] De la Escalera, A. and Armingol, J. M., "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," *Sensors* **10**(3), 2027–2044 (2010). https://doi.org/10.3390/s100302027.

[19] Hansard, M., Horaud, R., Amat, M., and Evangelidis, G., "Automatic detection of calibration grids in time-of-flight images," *Computer Vision and Image Understanding* **121**, 108–118 (2014). https://doi.org/10.1016/j.cviu.2014.01.007.

[20] Scaramuzza, D., Martinelli, A., and Siegwart, R., "A toolbox for easily calibrating omnidirectional cameras," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5695–5701 (2006). https://doi.org/10.1109/IROS.2006.282372.

[21] Mallon, J. and Whelan, P. F., "Which pattern? biasing aspects of planar calibration patterns and detection methods," *Pattern Recognition Letters* **28**(8), 921–930 (2007). https://doi.org/10.1016/j.patrec.2006.12.008.

[22] Kim, J.-S., Gurdjos, P., and Kweon, I. S., "Euclidean structure from confocal conics: Theory and application to camera calibration," *Computer Vision and Image Understanding* **114**(7), 803–812 (2010). https://doi.org/10.1016/j.cviu.2010.03.004.

[23] Kim, J.-S., Gurdjos, P., and Kweon, I.-S., "Geometric and algebraic constraints of projected concentric circles and their applications to camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(4), 637–642 (2005). https://doi.org/10.1109/TPAMI.2005.80.

[24] Yang, B., "Distortion-free camera calibration," in *Proceedings of SPIE, Dimensional Optical Metrology and Inspection for Practical Applications IX*, **11397** (2020). https://doi.org/10.1117/12.2558274.

[25] Han, S., "Improvement of an lcd-based calibration site for reliable focus-dependent camera calibration," *KSCE Journal of Civil Engineering* **26** (2021). https://doi.org/10.1007/s12205-021-5464-x.

[26] Bresenham, J. E., "Algorithm for computer control of a digital plotter," *IBM Systems Journal* **4**(1), 25–30 (1965). https://doi.org/10.1147/sj.41.0025.

[27] Wu, X., "An efficient antialiasing technique," in *Proceedings of Annual Conference on Computer Graphics and Interactive Techniques*, **25**(4), 143–152 (1991). https://doi.org/10.1145/122718.122734.

[28] Hardy, G. H., [*Ramanujan: Twelve lectures on subjects suggested by his life and work*], vol. 136, American Mathematical Soc. (1999).