Biometric Technology Today
ISSN (online): 1873-1880

Application of Deep Learning-Based Chessboard Marker Recognition Algorithm in Surgical Positioning

Ruofan Bian[1], Bin Chen[2], Xiaoping Ouyang[3], Dexing Kong[4*],

[1]School of Mathematical Sciences, Zhejiang University, Hangzhou, China.

[2]The First Affiliated Hospital, Zhejiang University School of Medicine, Hangzhou, China

[3]State Key Laboratory of Fluid Power & Mechatronic Systems, Zhejiang University, Hangzhou, China.

[4*]School of Mathematical Sciences, Zhejiang University, Hangzhou, China. Email: dxkong@zju.edu.cn

Abstract: The chessboard grid, as common positioning markers in computer-assisted navigation systems, is crucial for the speed and accuracy of the navigation system. In computer vision, people typically use a chessboard detection function OpenCV: findChessboardCorners to detect the corners of the chessboard grid for camera calibration and pose estimation. However, the applicability of this function is limited, as it may not return accurate results in cases of occlusion, low image quality, small chessboard grid areas, or multiple targets. Additionally, as image resolution increases, the algorithm's processing time significantly increases. These limitations make it challenging to meet the requirements of surgical positioning. To address issues such as small chessboard markers, insufficient clarity, and the presence of multiple chessboard grids simultaneously in surgical scenarios, this paper proposes a two-stage chessboard detection algorithm based on deep learning. This algorithm first segments the approximate location of chessboard grids and then identifies the positions of the grid points. Compared to traditional chessboard detection algorithms, this approach significantly improves the accuracy and efficiency of chessboard detection.

Keywords: Surgical Positioning, Checkerboard Detection, Small Checkerboard, Low Quality Images, Two-stage Recognition Algorithm.

Introduction

The checkerboard detection algorithm [1], [2] is a basic and essential algorithm in computer vision. In three-dimensional vision, the checkerboard grid is commonly used as a calibration object for calibrating camera parameters [3], [4]. In practice, the algorithm can only recognize single checkerboard grid images with specific lighting conditions, high image clarity, and a large proportion of checkerboard grid in the image. However, the checkerboard grid has advantages of distinctive features, simple patterns, and contains attitude information, which means it can be used as feature markers for localization tasks.

This paper uses a checkerboard grid as a feature marker to localize the pose position of surgical instruments. Compared to the standard optical spheres commonly used for high-precision surgical positioning, checkerboard markers offer advantages such as smaller size, lower cost, and no need to modify the shape of surgical instruments. In this scenario, the checkerboard grid accounts for a small percentage of the image, the lighting conditions are not uniform, the sharpness is low and there are multiple checkerboard grids in the image at the same time. In this paper, we expect to find the target feature points (the pixel size of the checkerboard grid is about 100×80) in the high resolution (image resolution of 9280×6992) image to achieve precise positioning of the target surgical instruments. In this extended checkerboard grid usage scenario, traditional checkerboard grid detection algorithms for camera calibration have difficulty in accurately identifying the checkerboard grid and the grid points on it in an image.

To solve the above checkerboard detection problem, this paper proposes a two-stage checkerboard grid detection algorithm: in the first stage, the algorithm detects the approximate location of the checkerboard grid; in the second stage, it calculates the locations of the grid points on the checkerboard grid and completes them. Combined with the deep learning EfficientNetB0 network [5], [6], we can quickly get stable and reliable results in the task of detecting small checkerboard grid markers.

Related Work

The checkerboard grid detection algorithm in OpenCV consists of the following steps: 1. Preprocessing, where the input image is converted to a grayscale image, and preprocessing such as adaptive thresholding and image normalization is applied based on the flag parameter; 2. Corner points detection, where the Harris Corner Detection Algorithm [7] is used to determine the possible corner locations in the image by detecting the changes in the gray value of the pixel points; 3. Corner point screening. The geometric constraints of the checkerboard grid, such as the distance between corner points and the order in which the corner points are arranged, are used to determine the final corner points within the checkerboard grid. In the pre-processing stage, if the illumination on the checkerboard grid is uneven, the thresholding and normalization operations may erase some of the checkerboard points; in the corner detection process, if the checkerboard points themselves are fuzzier, it is easy to fail to detect them; in the corner screening process, the function is impossible to deal with the simultaneous occurrence of checkerboard grids and when checkerboard grids are small compare to the image, it is easy to be determined as a noisy point and discarded.

Convolutional neural networks [8] are widely used in various tasks of image processing due to their additional feature extraction capabilities. For checkerboard grid detection task, the research target has clear appearance characteristics and is easy for neural networks to learn its unique graphical structure. Rosten and Drummond proposed FAST [9] and FAST-ER [10] to use networks to deal with the checkerboard grid recognition problem. B. Chen, C. Xiong, and Q. Zhang proposed CCDN [11] to solve the problem of checkerboard grid detection for input images with different resolutions, and obtained correct and stable grid detection results. However, the above methods can only deal with the single checkerboard grid detection problem, but fail to recognize multiple checkerboard grids at the same time.

Two-stage Checkerboard Testing Framework

The Overall Algorithmic Framework

The framework of the staged checkerboard grid detection algorithm proposed in this paper is as follows:

Step 1, Chessboard Detection Network: This network takes an original image containing multiple chessboard grids as input and segments the approximate location masks of potential target chessboard grids. Our network is based on an EfficientNetB0 segmentation network, which increases the number of layers and channels of the basic EfficientNetB0 model to adapt to the high-resolution 2320×1748 images used in this paper.

Step 2, Grid Point Detection Network: This network preprocesses the image based on the segmentation results from Step 1. It divides each segmented area from the original image according to the location masks output by the Chessboard Detection Network into small images that may contain chessboard grids. Meanwhile, each segmented area's position on the original image is recorded. The Grid Point Detection Network infers the coordinates of grid points in each segmented area, providing the chessboard grid detection results in the original high-resolution image (Figure 1).

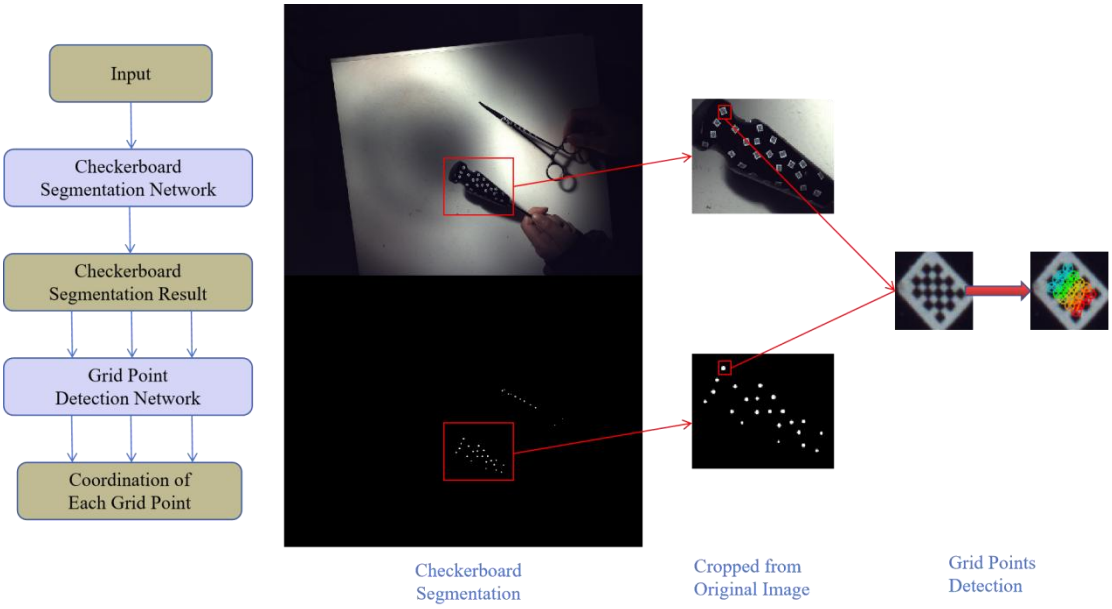

Figure 1. Framework and Demonstration of the Two-stage Checkerboard Detection Algorithm

Network Structure

Table 1: Structure of Checkerboard Detection Network

Stage	Operator	Channel	Layers
1	Conv7×7	32	1
2	MBConv1,k3×3	16	1
3	MBConv6,k3×3	24	2
4	MBConv6,k5×5	40	2
5	MBConv6,k5×5	320	4
6	Conv1×1	1280	1
7	Conv3×3	2048	1
8	Conv3×3	512	1
9	Conv3×3	400	1
10	Upsampling		1

The checkerboard detection network structure is shown in Table 1, this paper is based on EfficientNetB0 network for image feature extraction. The input image is first processed using a Conv7×7 convolutional layer, then the image features are extracted by a 4-stage MBConv convolutional module [6] and a 4-stage Conv convolutional layer, and finally up-sampled to the original image size to output the marked point feature heatmap.

The training process uses logistics to normalize the output result's heatmap to [0-1] and then uses cross-entropy as the loss function (equation (1)):

$$\text{Loss} = -\frac{1}{hw}\sum_i \sum_j p(i,j)\log(\hat{p}(i,j)) \quad (1)$$

Where, hw denotes the height and width of the image, respectively, $p(i,j)$ denotes the ground truth of the segmentation, and $\hat{p}(i,j)$ denotes the predicted result.

Table 2: Structure of Grid Point Detection Network

Stage	Operator	Channel	Layers
1	Conv7×7	32	1
2	MBConv1,k3×3	16	1
3	MBConv6,k3×3	24	2
4	MBConv6,k5×5	40	2
5	MBConv6,k5×5	112	4

Stage	Operator	Channel	Layers
6	MBConv6,k3×3	320	1
7	Conv3×3	100×Grid point	1
8	Upsampling		1

Compared to the chessboard detection task (Table 2), grid point detection involves a significantly reduced image resolution and a different output format. In the chessboard detection task, the output is a heatmap of potential target chessboard grids' area. In each heatmap, there are multiple non-connected segmentation areas. The output of the grid point detection task is a feature heatmap of the number of grid points in the chessboard grid, with at most one grid point area in each heatmap (in cases where a channel fails to identify a grid point, there is no clear grid point information in the heatmap). Based on these differences, we modified the network structure. In the grid point detection network, we first use a Conv7×7 convolutional layer to process the input image, followed by 5 stages of MBConv convolution modules and 1 stage of Conv convolutional layer to extract image features. Due to the smaller image resolution, the number of upsampling layers in the network is correspondingly reduced. Finally, the network outputs the grid point feature heatmap. Similarly, the output heatmap is normalized to [0-1] using logistics during the training process, and then cross entropy is used as the loss function (equation (2)):

$$\text{Loss} = -\frac{1}{n\hbar w}\sum_k \sum_i \sum_j p_k(i,j)\log\left(\widehat{p_k}(i,j)\right) \tag{2}$$

where $\hbar w$ denotes the height and width of the image, respectively, $p(i,j)$ denotes the ground truth of the segmentation, and $\hat{p}(i,j)$ denotes the predicted result, k denotes the k$t\hbar$ grid point area, corresponding to the mask image of each grid point.

Dataset Settings

In the training of stage one checkerboard grid detection network, this paper uses a high resolution camera to acquire motion images of target objects with checkerboard grid markers, totaling more than 2000 images. Small checkerboard grids are labeled with dots on the captured images. A total of 27000 checkerboard grids are labeled on more than 2000 images. We randomly selected 1,600 images as the training set and the remaining over 400 images as the validation set.

The training process is expanded to 10 pixels around the circle markers based on their mask. In this stage of the target task, we do not need to completely segment out the checkerboard grid markers, but focus more on the pixel regions that may be small checkerboard grids. Therefore, it is reasonable that we use circle markers of the same size as the mask for the checkerboard grid markers. Meanwhile, after image input, the dataset was expanded using flipped and rotated images to supplement the diversity of marker images and increase the number of samples. The input image of this study comes from a high-resolution camera with a resolution of 9280×6992. In the training process, it needs to be resize to 1/4 size of the original image first, and then split the image and its mask into 4×4 small images and put them into the segmentation network for training.

In the training of the Stage 2 chessboard grid point detection network, it is important to note that there are associations between the grid points. For each small checkerboard grid, the grid points on it are uniquely numbered, and the numbering rule is determined according to the OpenCV checkerboard point detection function findChessboardCorners. In this stage of training, we directly use the findChessboardCorners function to solve for the pixel coordinates of the checkpoints on the checkerboard grid markers segmented from the original image. A corresponding number of masks are generated based on the grid point numbers as a training set.

We used findChessboardCorners on 27,000 small checkerboard regions on more than 2,000 surgical instrument motion images obtained from the first checkerboard annotation, and finally obtained 15,600 successfully recognized checkerboard grid point mask images. In the process of generating checkpoint mask, we verified that the checkerboard grid detection algorithm based on the traditional method is less effective in detecting checkerboard grids with low quality, uneven lighting conditions, and large shooting angles. The training set generation in this stage can be computed by the checkerboard grid detection algorithm in the first stage and the findChessboardCorners function without manually labeling the data.

During the training process, based on the checkerboard grid masks generated in the first stage, 40-50 pixel points are extended outward from the center of each connected domain to obtain a square region with side

lengths of 80-100 as the input image, and the coordinate positions of the checkpoints are calculated at the sub-pixel level using the findChessboardCorners function. For the pixel regions where the checkpoint information is successfully detected, masks corresponding to the number of checkpoints are generated as the checkpoint detection training set and validation set. After image input, the data set is expanded using flipped and rotated images to supplement the diversity of marker images and increase the number of samples. For the tessellated markers after flipping, due to the axisymmetric nature of the tessellated markers themselves, the numbering order of their checkpoints should be renumbered according to the numbering order of the checkpoints generated by the findChessboardCorners function in order to ensure the uniqueness of the numbering of the checkpoints.

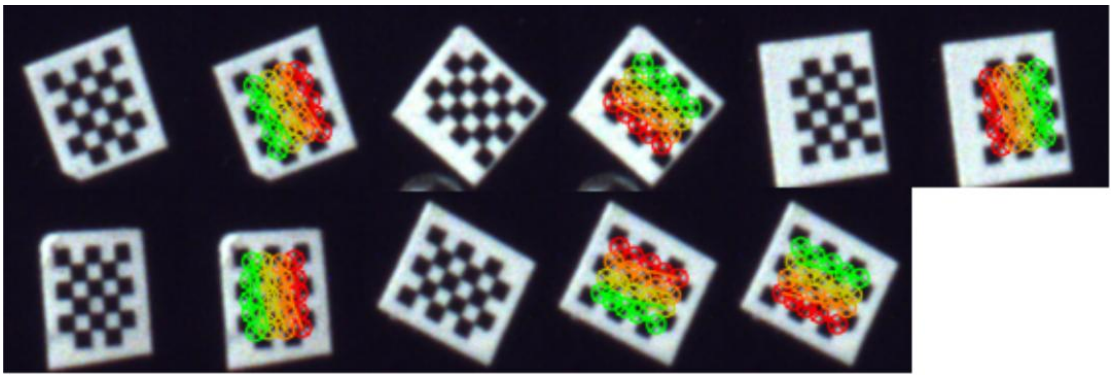

Figure 2. Demonstration of the Checkpoint Detection Results (The order of the grid points in the first five detection results is consistent, but the last checkerboard grid is wrongly numbered although the location of the grid points is detected, and this kind of situation is also regarded as a misdetection)

It is worth noting that the misdetection in the last image of Figure 2 is caused by the change of the grid point serial number after the flip transformation. If the original grid point serial number is used as the training data after the flip transform, the direction information encoded in the serial number will be lost. The rule of the serial number transformation caused by the flip transformation is as follows: for the checkerboard grid in row n and column m, the grid point originally numbered $m \times (i - 1) + j$ in row i and column j is updated to $m \times (n - i) + j$ after the flip transformation. The grid point serial number transformation caused by the flip transformation is shown in Figure 3.

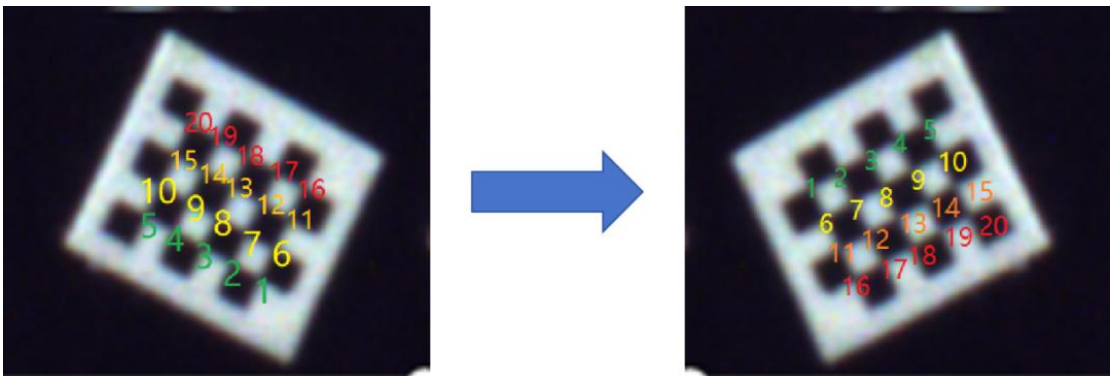

Figure 3. Grid Point Serial Number Change after Flip Transformation

The above process is for single checkerboard grids. As for multi-marker detection, it is necessary to adjust the structure of the over network, change the number of output Channels of Stage7 to output the coordinate position of the corresponding numbered checkerboard points with different channels, and use different networks to complete the checkerboard point detection task.

Grid Point Complementation

When the image quality of the checkerboard grid is comparatively poor, the OpenCV: findChessboardCorners function will miss some of the grid points in the second stage of the corner detection stage, resulting in recognition failure due to missing grid points in the third stage of the corner screening and sorting process. The grid point detection algorithm in the framework of this paper calculates the coordinates of the grid points corresponding to the serial number through different channels, and the results are relatively independent, so that in the case of failing to recognize all the coordinates of the grid points, the coordinates of some of the grid points can still be output according to the serial number. This paper adopts the post-processing method of grid point complementation, which can further improve the recognition rate of the checkerboard grid.

For unrecognizable lattice points, the output coordinates of the checkerboard lattice detection algorithm are set as (-1, -1). In the training stage, we use the channel sequence number to represent the checkerboard grid point sequence number, which directly obtains the geometric constraints of the checkerboard grid point ordering. Based on these grid point constraints, we can calculate the checkerboard grid center point position, long axis short axis and other information. The specific calculation method is as follows:

Let the point on the checkerboard grid with serial number 1 be the checkerboard origin (x_0, y_0), the single cell coordinates of the long axis on the checkerboard grid are expressed as (x_a, y_a), the unigram coordinates of the short axis are expressed as (x_b, y_b) (Using the checkerboard grid in Figure 3 as an example, the vectors from ordinal number 1 to ordinal number 2 are single cells on the long axis, and the vectors from ordinal number 1 to ordinal number 6 are single cells on the short axis). For a checkerboard grid of row n column m, the coordinate of the grid point row i column j is $(x_0 + i \times x_a + j \times x_b, y_0 + i \times y_a + j \times y_b), i = 0,1,..,n-1, j = 0,1,..,m-1$ (equation (3))

$$\begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} x_a & x_b \\ y_a & y_b \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix}. \tag{3}$$

First centering the above equation, approximating the constant term and simplifies the expression to (equation (4)):

$$\begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix} = \begin{pmatrix} x_a & x_b \\ y_a & y_b \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix}. \tag{4}$$

Then split the above equation into two linear equations (equation (5)):

$$\begin{cases} x_{i,j} = (i \quad j) \begin{pmatrix} x_a \\ x_b \end{pmatrix} \\ y_{i,j} = (i \quad j) \begin{pmatrix} y_a \\ y_b \end{pmatrix} \end{cases} \tag{5}$$

Based on the output of the grid-point detection algorithm, we obtain a number of the above equations, and when the number of detected grid-points is greater than or equal to three and not on the same straight line, using the least-squares method, we can obtain $\begin{pmatrix} x_a & x_b \\ y_a & y_b \end{pmatrix}$.

Finally, according to the long and short axis vectors, the coordinates of the checkerboard grid origin are introduced and the coordinates of the undetected grid points are complemented to complete the checkerboard grid detection post-processing.

Experiments

For the first step tessellation detection network, this paper focuses on the detection rate of tessellated regions. We additionally label 100 images with checkerboard markers as the validation set. The detection rate of tessellations on the validation set is higher than 97%.

For the second step of the checkerboard detection network, we capture a 15-minute-long video of surgical instrument motion trajectories with checkerboard markers. For this video, key frames are extracted every 1 second and fed into the checkerboard detection network to obtain 27,384 small checkerboard masks, which are segmented into small checkerboard regions according to the checkerboard masks, which are pre-processed by using the original OpenCV: findChessboardCorners function, multi-scale zoom in and out, color saturation adjustment, etc., respectively. findChessboardCorners function with our grid point detection network. The results are shown in Table 3:

Table 3: Structure of Grid Point Detection Network

Algorithms	FindChessboardCorners	Preprocessing +FindChessboardCorners	Ours
Recognized	12236	15603	24831
Not Recognized	15148	11781	2553
Recognition rate	44.68%	56.98%	90.68%
Speed(s)/Frame	0.5~1s	0.5~8s	less than 0.1s

The experimental results show that the Efficient Net-based checkerboard grid detection algorithm is much higher than the checkerboard grid detection function that comes with OpenCV in terms of accuracy and efficiency. It should be pointed out that the images to be recognized in the table are the images generated by the first step of the checkerboard grid detection network that may be checkerboard grids, and there are some images with poor quality or not checkerboard grids. Therefore, the actual recognition accuracy of our checkerboard point detection network will be higher.

Conclusion and Outlook

Aiming at the poor generalization performance of OpenCV: findChessboardCorners function and the inefficiency of the algorithm, this paper proposes a phased checkerboard checkerboard detection framework based on EfficientNet, which significantly improves the accuracy of checkerboard checkerboard detection and the efficiency of the algorithm. Further, this paper realizes the detection of multiple small checkerboard squares in the image, solves the problem of mutual interference of multiple checkerboard squares in the existing algorithmic framework, and enables small checkerboard squares to be used as simple and stable target localization markers.

Based on the checkerboard lattice detection framework in this paper, the use of small checkerboard markers instead of standard optical blobs can be realized in the future for high-precision tracking and localization tasks of smaller and harder-to-mark target objects (e.g., surgical instruments). However, with deep learning-based checkerboard point detection algorithms, a network can only handle one specification of checkerboard grid detection task. For different specifications of checkerboard grids, it is necessary to change the output layer of the model and retrain the network, i.e., the generalization performance is worse than the existing checkerboard grid detection algorithms.

Acknowledgments

This research was funded by the Natural Science Foundation of Zhejiang Province, China (Grant no. LSD19H180005).

References

[1] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. Sebastopol, CA: O'Reilly Media, Inc., 2008.

[2] OpenCV Team. OpenCV API Reference.[Online]. Available: https://docs.opencv.org/4.8.0/

[3] M. E. Loaiza, A. B. Raposo, and M. Gattass, "Multi-camera calibration based on an invariant pattern," *Computers & Graphics*, vol. 35, no. 2, pp. 198-207, 2011.

[4] A. Duda and U. Frese, "Accurate detection and localization of checkerboard corners for calibration," in *British Machine Vision Conference*, vol. 126, Sep. 2018.

[5] M. Tan and Q. Le, Q. (2019, May). "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, May 2019, pp. 6105-6114.

[6] A. G. Howard *et al*., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv preprint:1704.04861*.

[7] C. G. Harris and M. J. Stephens, "A combined corner and edge detector," *Alvey Vision Conference*, 1988, doi: 10.5244/C.2.23.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, 2012, doi: 10.1145/3065386.

[9] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, May 2006, pp. 430-443.

[10] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105-119, 2008.

[11] B. Chen, C. Xiong, and Q. Zhang, "CCDN: Checkerboard corner detection network for robust camera calibration," in *Intelligent Robotics and Applications: 11th International Conference, ICIRA 2018, Newcastle, NSW, Australia, August 9–11, 2018, Proceedings, Part II 11*, Aug. 2018, pp. 324-334.