

# Pyramidal Blur Aware X-Corner Chessboard Detector

Peter Abeles<sup>1</sup>

**Abstract**—With camera resolution ever increasing and the need to rapidly recalibrate robotic platforms in less than ideal environments, there is a need for faster and more robust chessboard fiducial marker detectors. A new chessboard detector is proposed that is specifically designed for: high resolution images, focus/motion blur, harsh lighting conditions, and background clutter. This is accomplished using a new x-corner detector, where for the first time blur is estimated and used in a novel way to enhance corner localization, edge validation, and connectivity. Performance is measured and compared against other libraries using a diverse set of images created by combining multiple third party datasets and including new specially crafted scenarios designed to stress the state-of-the-art. The proposed detector has the best F1-Score of 0.97, runs 1.9x faster than next fastest, and is a top performer for corner accuracy, while being the only detector to have consistent good performance in all scenarios.

## I. INTRODUCTION

Camera calibration is an essential part in determining the mapping between an image’s 2D pixel coordinate and the 3D world. By observing calibration markers, it is possible to extrapolate a camera’s intrinsic characteristics (e.g. focal length and lens distortion) to a high level of precision [1].

Chessboard (or checkerboard) patterns are arguably the most popular pattern for camera calibration with good reason [2], they are easy to detect and robust to blur due to their symmetry. They are used by themselves or with self-identifying patterns [3], as done in CALTag [4] and ChArUco [5].

In this work, we focus on fully automatic detection of chessboard patterns in diverse environments. Outdoors calibration and in factories is increasingly important and challenging due to poor inconsistent lighting, motion blur, poor focus, harsh shadows, and complex backgrounds. In addition, typical camera resolutions have increased dramatically since much of the past work was proposed and new techniques are needed to handle, the now common, 12+ megapixels images.

A novel chessboard detector is proposed that combines several innovations; a new x-corner detector, blur estimator, robust blur aware edge validation, and overall multi-scale approach. The last three innovations are significant departures from past work and is the first known fiducial detector to estimate the optimal scale to localize a corner, then uses the scale to dynamically sample edges and determine corner connectivity. Enabling heavily blurred and fisheye images to be processed. The final output can be configured to return all found patterns or use prior information and find a single known target. Results include metadata that

is useful for camera diagnostics, autofocus, and estimating corner precision.

To validate performance, a study is performed on what might be the most diverse set of chessboard scenes to date from multiple authors and new ones specifically designed to stress and break detectors in real-world situations. Accuracy is measured using hand labeled corners and synthetic images with known corners. In this study, the proposed detector demonstrated its robustness, accuracy, and speed across all scenarios. Making it the only tested library to produce consistently good results. Both source code and datasets are freely available online<sup>1</sup>.

## II. RELATED WORK

Planar targets are widely used in camera calibration [1] for their ease of use and there are a multitude of toolboxes for them, such as OpenCV [6], Bouguet’s Matlab [7], CamOdoCal [8], and BoofCV [9]. The earliest approaches required human intervention to find corners, while more recent work finds most of the corners automatically, to varying degrees of accuracy and reliability. Deltile patterns have been proposed [10] as an alternative to chessboards for improved handling of blur and distortion.

One approach to chessboard detection involves using contours around squares. In OpenCV’s *findChessboardCorners* [11] the image is adaptively thresholded, eroded to separate the individual squares/quadrangles, then squares are found using contours. The chessboard grid is formed by connecting quadrangles corners. Image borders are challenging since they crop and/or heavily distort squares. With improvements to this general approach being proposed in [12], [8]. Recently, deep learning has been explored for corner detection [13].

Instead of using square contours, x-corners (named after the intersection of two squares) can be detected directly, avoiding image border issues. Harris [14] corner detector is the basis for many x-corner detectors [15], [16], [17]. While Harris is excellent at detecting x-corners, it detects other corner-like shapes too.

Many papers have proposed custom x-corner intensity functions [18], [17], [10], [19]. In [20] x-corners are found by looking at binary centerlines [21]. Filtering corners after detection is standard. In [16], [18] filtering is done using local gradient histogram statistics and based on local intensity patterns in [17].

After candidate corners are found at pixel level precision, they are refined to sub-pixel accuracy [22], [23]. Specific

<sup>1</sup>Peter Abeles is with NINOX 360, Redwood City, CA USA  
pabeles@ninox360.com

<sup>1</sup>Detector and source code has been available since 2019 in BoofCV, with smaller improvements in v0.39. <https://boofcv.org>

approaches include: Fitting a quadratic function [24] or saddle point [15], [20], [10] to x-corner intensity image. Maximize gradient orthogonality [18]. Iterative perspective undistort and redetection [25].

The first step in constructing chessboard graphs from x-corners is finding high confidence pairs. Combinatorics can be limited by only considering N-nearest-neighbor corners [16], [10], [19]. Require two corners have compatible orientations/polarities [18], [10]. Reject pairs using image gradient or sample points along edges [16], [26], [10].

After pairs are known, the graph is then constructed. This is a challenging process which often drives the final accuracy. In [18], [10], [19] multiple seeds are used then expanded into a graph. In [8] a spline is used to reject rows if the error is too large.

### III. PROPOSED APPROACH AND CONTRIBUTIONS

The proposed approach is summarized in Algorithms 1 and 2, with a more detailed explanation in the following subsections. "Level" always refers to level in an image pyramid and is synonymous with scale.

X-corner detector contributions:

- 1) *Corner localization at optimal scale*: Select lowest level with a strong response to avoid instability.
- 2) *Explicit handling of x-corner symmetry*: Applying 2x2 box filter for unique local maximum, stabilizing non-maximum suppression, see Figure 2.
- 3) *Sub-pixel accuracy using mean-shift*: Alternative to curve fitting that is less sensitive to noise.

Graph connectivity and structure contributions:

- 1) *Robust edge validation*: Score using  $n$ -best points. Reduces sensitive to outliers from localized lighting.
- 2) *Blur aware edge validation*: Avoid sampling poorly defined edges near blurred x-corners centers.
- 3) *Scale Compatibility*: Only connect a corner to others at same or higher pyramid level. Reduces influence of false positives at lower levels.

#### A. X-Corner Detector

##### Algorithm 1: Pyramidal X-Corner Detection

- 
- 1 Let  $L$  be the set of images in a pyramid
  - 2 **for**  $l_i \in L$  **do**
  - 3   Compute x-corner intensity
  - 4   2x2 box filter for unique maximums
  - 5   Non-maximum suppression
  - 6   Prune using a cascade of filters
  - 7   Mean-shift sub-pixel localization
  - 8   Compute corner orientation and revised intensity
  - 9 **end**
  - 10 Connect corners across pyramid levels
  - 11 Select level for corner location and orientation
- 

The proposed x-corner intensity function is designed to be accurate, fast, and have affine lighting invariance. It will be excited only when encountering x-corners. Other corners and

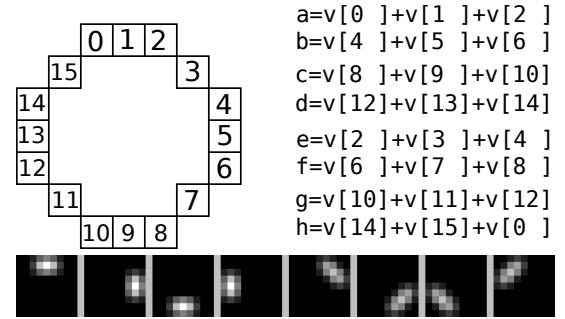


Fig. 1. Left: Circle sampling pattern stored in array 'v'. Right: Workspace variables formulas. Bottom: Visualization of kernels.

lines cause a negative response. The "likelihood" function in [18] is an inspiration, but their formulation has many more logical branches making it expensive to compute.

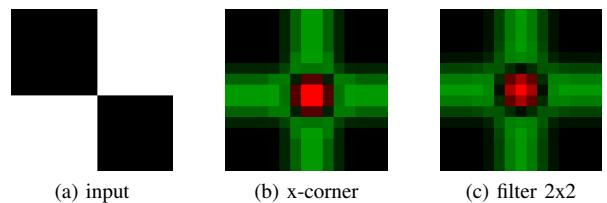


Fig. 2. a) Chessboard corner without perspective distortion. b) Original x-corner intensity. Red and green indicate positive and negative x-corner response, respectively. c) Local maximum after applying 2x2 block filter, which improves corner localization in general.

*Corner Intensity*: 1) Convert image to grayscale and scale pixel values to be 0 to 1. 2) Apply 3x3 Gaussian blur to image. 3) For each pixel, sample a circle as shown in Figure 1. 4) Compute x-corner intensity  $I \in \mathbb{R}$

$$I = \max(\text{xscore}(a, b, c, d), \text{xscore}(e, f, g, h)) \quad (1)$$

where  $\{a, b, c, d, e, f, g, h\}$  are sampled in Step 3.

```
|| fun xscore( v1 , v2 , v3 , v4 )
  ||   μ = (v1+v2+v3+v4)/4
  ||   return (v1-μ)*(v3-μ) + (v2-μ)*(v4-μ)
```

Gaussian blur in Step 2 acts as a computationally efficient template to construct kernels in Figure 1. The *xscore* function is at a maximum when the kernel response  $v_1$  and  $v_3$  are both above or below the mean, then the same is done for  $v_2$  and  $v_4$ . Affine lighting invariance is achieved by subtracting the mean then multiplying the difference. Partial rotation invariance is achieved in Eq. 1 by using two templates offset by 45 degrees.

In the absence of noise and perspective distortion, Eq. 1 will not have unique local maximums. This is resolved by convolving a 2x2 box kernel to the intensity image (Figure 2) and apply (0.5, 0.5) pixels offset to undo box filter induced shift. In typical situations, this helps sub-pixel estimation converge by providing a better initial estimate. For corners at a skewed angle and heavy fisheye distortion the box filter still help break the symmetry.

A cascade of filters is applied to reduce candidate corners in order of least to most computationally expensive. 1) Filter corners based on intensity relative to maximum intensity in top pyramid level. 2) Filter if too many positive x-corner intensity values. Actual x-corners tend to have negative neighbors. 3) Check for expected up-down gray scale pattern. 4) Apply Shi-Tomasi [27] corner Eigenvalue test.

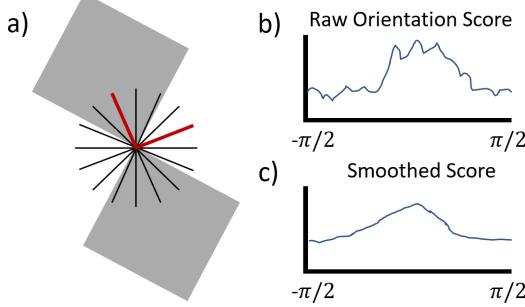


Fig. 3. a) Orientation and a rotationally invariant x-corner intensity are found using the line integral along 32 spokes centered on the corner. Red indicates selected orientation. b) Cartoon showing raw spoke orientation score. c) Unique maximum after smoothing.

Remaining corners are refined to sub-pixel accuracy using mean-shift in the intensity image. Mean-shift implicitly applies a low pass filter, making it less sensitive to noise than curve fitting approaches, which attempt to exactly fit all values. Orientation and an improved x-corner intensity are estimated using spokes, Figure 3. These spokes are conceptually similar to the approximated Radon transform in [19]. Corner half-circle orientation is estimated by computing the difference between each spoke and one offset by 90 degrees. Resulting array is smoothed with a Gaussian kernel to create a maximum in the middle, Figure 3. The selected angle is further refined by fitting a quadratic. The new x-corner intensity is set to the orientation's best score and exhibits better rotational invariance than Eq 1. Corner contrast is computed as the difference in magnitude of spokes for the best orientation and is used later on.

### B. Pyramidal Processing

Unlike blob features [28], e.g. Hessian or Laplacian, corners do not have a unique maximum in intensity across scale-space [29] but have a constant intensity. This is similar to the aperture problem [30] but across scale-space. In fact, if there is no noise or blur, a single corner would be observable in all pyramid levels. In practice, the first level is determined by the amount of blur and sensor noise, while the last level is based on ratio of marker size and distance.

Corners are poorly defined under heavy blur and localization is dominated by noise. Naively assuming higher resolution are better for localization causes instability. To overcome this issue, we proposed that the selected level for localization maximizes resolution and intensity:

$$\max_{level} \text{intensity}(c, level)/(level + 1) \quad (2)$$

where  $\text{intensity}(c, level)$  is a corner's intensity at  $level$ .

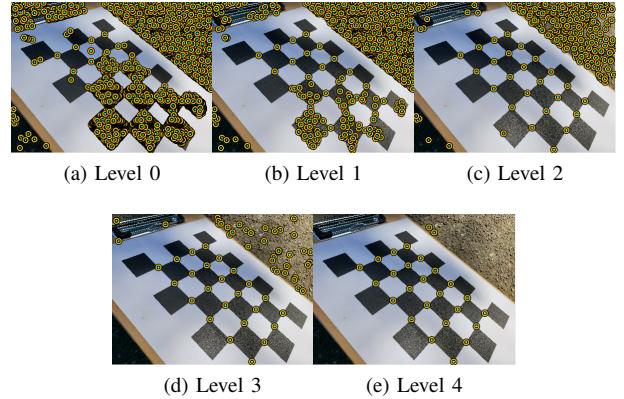


Fig. 4. Corners detected across pyramid levels. Higher levels have lower resolution. Yellow circles are found x-corners.

Metadata from pyramidal processing is useful for end consumers. High confidence corners will typically be viewed across multiple levels. The first observed level indicates the amount of local blur/noise. For example, this could be used to autofocus a camera on chessboard patterns, estimate corner accuracy, or detect faults during calibration.

### C. Corner Connectivity

---

#### Algorithm 2: Chessboard Graph Formation

---

```

1 Let  $L$  be the set of all pyramid levels
2 for  $l_i \in L$  do
3   Let  $C$  be the set of all detected corners in  $l_i$ 
4   for  $c_i \in C$  do
5     Let  $C_n$  be the set of nearest neighbors of  $c_i$ 
      visible in the same level or greater
6     for  $c_j \in C_n$  do
7       Discard  $c_j$  if incompatible orientation
8       Sample grid of points connecting  $c_i$  and  $c_j$ 
9       Score samples using binary intensities
10      Connection if sufficient score
11    end
12  end
13 end
14 for  $c_i \in C$  do
15   Vote on connected neighbors
16   Select neighbors based on perspective geometry
     and graph constraints
17 end
18 Discard ambiguous corners based on votes
19 Sort and enforce graph constraints

```

---

Connected corner pairs are required to construct a topological graph that represents the chessboard. These connections have known topological properties and orientations. Procedure: 1) For each corner, find its  $k$ -nearest-neighbors (KNN) in the same pyramid level or above as candidates for connecting. 2) Reject if the corner orientations are not perpendicular. 3) Reject if edge validation fails.

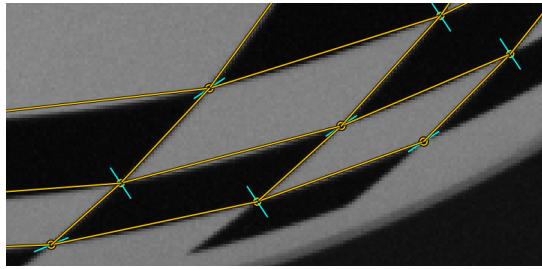


Fig. 5. Visualization of corner orientation (cyan) and corner connections (orange) under fisheye distortion

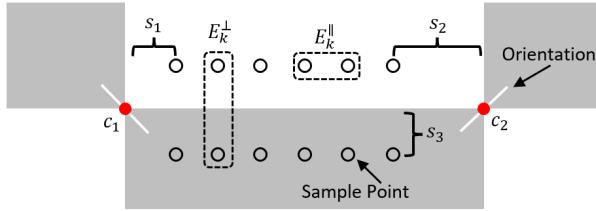


Fig. 6. Diagram illustrating components of edge validation. Sample locations are dynamic based on corner level to avoid blur.

Edge validation is done by sampling points in a grid on either side of the line, Figure 6.  $\{c_1, c_2\}$  are corners with different amounts of local blur. Spacing  $s_i$  is dependent on the pyramid level of  $c_i$ . The proposed novel dynamic spacing based on level avoids sampling poorly defined edges near blurry corners. Perpendicular offset  $s_3$  is determined by the distance between  $c_1$  and  $c_2$ . Perpendicular error  $E^\perp = I_i - I_j$  score is found by sampling adjacent points across the edge and is maximized by high contrast. Longitudinal error  $E^\parallel = |I_i - I_j|$  measures similarity. These scores are sorted and the worst values removed to prevent outliers caused by noise and hard shadows from dominating. The edge intensity score is shown below and edges with values below a threshold are pruned:

$$L_{ij} = \frac{\sum_k E_k^\perp - E_k^\parallel}{\text{contrast}(c_i) + \text{contrast}(c_j)} \quad (3)$$

Corner contrast( $c$ ) is defined in Section III-A and is used as a divisor for lighting invariance.

#### D. Grid Graph Construction

A correctly formed chessboard topological graph will have the following properties: 1) All corners must be mutually connected. 2) Every corner must have 2, 3, or 4 neighbors. 3) There must be one and only one other common corner between two neighbors that are adjacent. Any connection or corner not meeting these conditions is pruned. If there are too many connections and ambiguous corners have been identified, then a vote is taken from the local neighborhood where each corner decides independently which connection is the best fit based on edge intensity and expected geometry. The losing corner(s) are removed.

#### E. Grid Graph to Chessboard Graph

Grid graphs are reformatted into valid chessboard graphs by putting corners into a canonical counter-clock-wise graph order using corner orientation information and ensuring corner (0,0) is connected to a corner square. Optionally, a single complete grid can be enforced by removing stray connections: Outer rows and columns are removed if there are missing corners. This is repeated as required. If the chessboard shape is known in advance then only graphs which match are returned. If only a single chessboard is expected then the pattern with the largest apparent image size is returned. In summary, output can be multiple arbitrary chessboards or a single known chessboard where false positives are pruned with stronger assumptions.

Please consult source code for specifics as algorithm details in this section have been simplified for conciseness.

## IV. PERFORMANCE STUDY

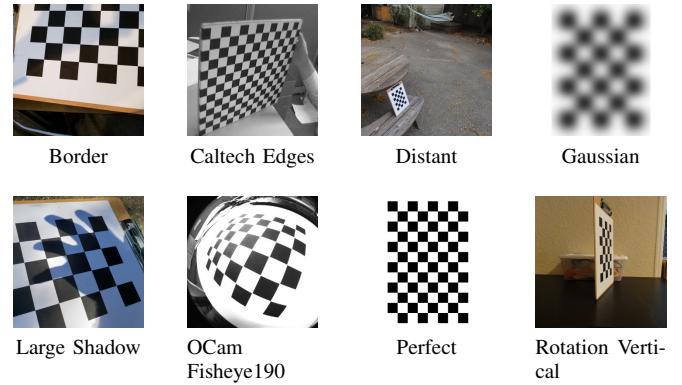


Fig. 7. Select sample images from 8 of 24 the scenarios. A wide range of environments, lighting conditions, and sensors have been tested.

A diverse performance study is conducted by combining several datasets from multiple authors, as well as adding new scenarios specifically designed to stress detectors, Figure 7. Scenarios vary widely in camera type, image size (0.3 MP to 12 MP), lighting conditions, noise level, motion blur, focus, and background clutter. Corner accuracy is measured using hand labeled corners as well as synthetic images with known corner locations. Every image will have one and only chessboard pattern with all corners visible, as is typical for single camera calibration. See technical report [31] for a more complete description of all scenarios.

Performance is measured for chessboard detection, corner localization accuracy, and runtime performance. Detection accuracy is measured using F1-Score statistics. A *true positive* TP is defined as a detected chessboard where every corner is within  $t_c$  pixels of ground truth. A *false positive* FP is defined as a detection with the expected grid shape and one or more corners outside of  $t_c$ . Detections with the wrong shape or missing corners are ignored. To keep it simple, because chessboard orientation can be ambiguous, corner error is computed using the closest match in ground truth. If possible, the known target's shape is given to a library.

Corner labels are used instead of the common post calibration reprojection error because reprojection error does not measure the detector’s accuracy directly. Instead reprojection error measures the calibration system as a whole and can hide systematic bias and report optimistic results due to camera model over parameterization. As an example, if a chessboard detector is biased by 5 pixels the calibration system would compensate by translating the camera and report perfect results.

Hand labeled x-corners are estimated to have an accuracy of around 0.2 pixels when images are in focus. Labeling heavily blurred images is difficult to impossible. For this reason, the *Gaussian* scenario is simulated with exact ground truth and *Gaussian Fisheye* has Gaussian blur added to a real fisheye image. By default,  $t_c$  is set to a generous 5 pixels and for *focus\\_large*  $t_c = 20$  due to labeling accuracy. The relative rankings of each library is insensitive to the choice of  $t_c$ , Figure 8. Libraries which define the pixel at  $(1, 1)$  to be within  $x, y \in (0.5, 1.5]$  are offset by  $(0.5, 0.5)$  pixels.

Runtime is measured using system clock in code while excluding IO. Libraries written in C/C++ are compiled in release mode and built with GCC 9.3.0. Java libraries use JVM 15.0.3 and are given a warm start. Matlab libraries run in headless mode using version R2021a.

**Libraries and Versions:** *Proposed*: [9] BoofCV v0.39, *DelChe*: [10] SHA eed7e86, *Geiger*: [18] libomnical and libcbdetect from website 2021-Aug-15, *OCamCalib*: [32] v3.0, *Binary*: OpenCV 4.5.3 findChessboardCorners and cornerSubPix, *ChessSB*: [19] OpenCV 4.5.3 findChessboardCornersSB. To be clear, *DelChe* is the chessboard detector which was adapted from a deltile pattern detector.

**External Datasets:** *caltech\_edges* [7], *ocam* [32], *stefano\_2012* [33], *challenge* [34], *kaist fisheye* [10].

## V. RESULTS

Corner localization accuracy is reported using 50% error  $E50$  and max error statistics  $E100$ . The same is true for runtime performance, i.e.  $R50$  and  $R100$ . Median or 50% statistics is intended to represent “expected” or nominal performance with outliers removed. Max statistics is the worst case performance and tends to be volatile.  $N$  is the number of images in a specific scenario. See Tech Report [31] for a more complete discussion of these results.

Libraries which crashed, froze, or were excessively slow presented a challenge for summary results. Due to the lack of any better options, summary performance is computed by omitting scenarios a library could not process. This gives some libraries a significant advantage by removing the most difficult scenarios. Here is a list of libraries and the number of omitted scenarios; DelChe 2, and OCamCalib 1.

Detection performance as a function of true positive thresholds across all scenarios combined is shown in Figure 8. The proposed approach has the best performance with an F1-score of 0.97 using a 5 pixel threshold. The next best library is Geiger with an F1-score of 0.92. The proposed and Geiger have comparable performance in nominal situations, but the proposed approach outperforms Geiger in degraded

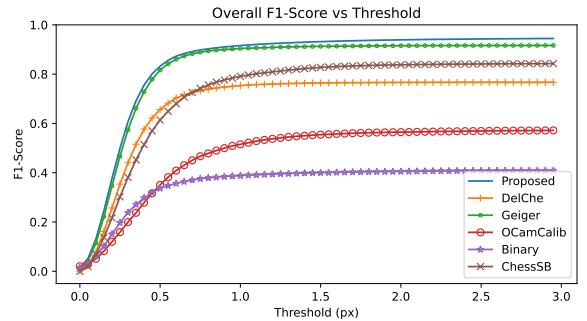


Fig. 8. F1-Score by combining all scenarios with different thresholds

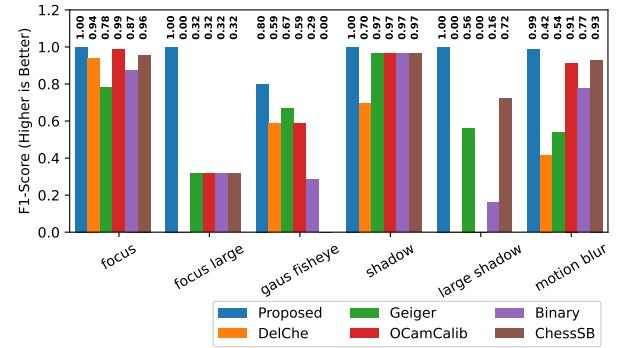


Fig. 9. F1-Score in select challenging scenarios

scenarios. Figure 9 shows performance in select scenarios with blur or adverse environmental conditions. In these individual scenarios the proposed is the top performer, often by a considerable margin, i.e. 3x in “focus large”. The biggest performance difference tends to be in scenarios with larger images ( $\geq 10$  megapixels, see Table II), validating the proposed blur aware pyramidal approach.

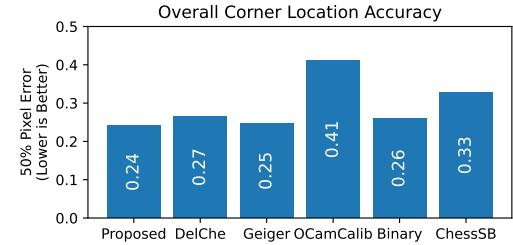


Fig. 10. Overall Corner Location Accuracy

Figure 10 shows nominal corner localization accuracy using 50% error from all true positive detections across all scenarios. Using 50% error avoids putting libraries with more detections at a disadvantage by excluding outliers. The proposed detector is effectively tied for best nominal accuracy. All libraries exhibit reasonable nominal accuracy.

Because overall nominal performance paints an unrealistically optimistic picture, selected specific scenarios are now considered. Table I shows true positives, 50% corner error, and max corner error in scenarios where most detectors exhibited “good” detection rates. *Sloppy* is a printed on wavy

Library	Sloppy			ThetaS			Motion			Stef2012		
	TP	E50	E100									
Proposed	<b>13</b>	<b>0.42</b>	2.17	<b>11</b>	0.25	0.89	<b>37</b>	0.67	4.45	<b>30</b>	<b>0.18</b>	0.83
DelChe	<b>13</b>	0.47	<b>2.12</b>	<b>11</b>	0.26	<b>0.80</b>	10	<b>0.43</b>	<b>2.18</b>	23	0.24	1.15
Geiger	<b>13</b>	<b>0.42</b>	2.97	<b>11</b>	<b>0.24</b>	0.94	14	0.60	3.23	<b>30</b>	0.19	<b>0.81</b>
OCamCalib	10	0.60	3.28	6	0.44	1.44	32	0.84	4.78	26	0.36	4.28
Binary	11	0.44	2.87	9	<b>0.24</b>	4.86	24	0.64	4.89	14	0.20	2.94
ChessSB	<b>13</b>	0.53	2.47	<b>11</b>	0.31	4.22	33	0.67	4.25	26	0.25	3.15

TABLE I

COMPARISON OF CORNER ACCURACY IN SELECT SCENARIOS

paper, *Motion* has motion blur, *Stef2012* regular camera, and *ThetaS* is an  $185^\circ$  fisheye. The reason max error tops out around 5 pixels is because detections are labeled as false positives at that point. Even if a detector has very low 50% error accuracy, calibration results will be poor if the max and false positive rates are high.

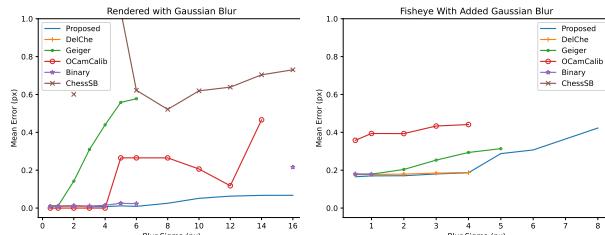


Fig. 11. Accuracy plots with increasing Gaussian blur. Left: Rendered chessboard without perspective distortion. Right: Fisheye with hand labeled corners.

Figure 11 shows accuracy degrading as a function of Gaussian blur in two scenarios. Ideally a detector would slowly degrade as blur increases and spikes could indicate fundamental instability in the approach. Oddly enough, the rendered scenario with no lens or perspective distortion is very challenging for some detectors, while the fisheye scenario exhibits stable performance. This could be caused by x-corner symmetry discussed previously. In both scenarios, the proposed approach has the most stable response and best accuracy.

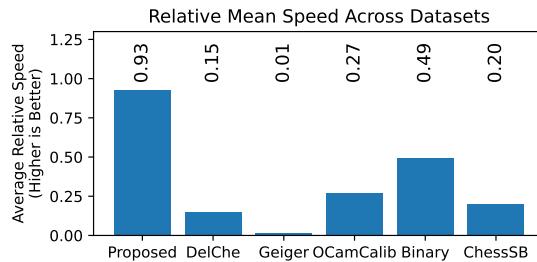


Fig. 12. Average mean relative runtime across all scenarios. A score of 1 would mean it is the top performer in every category.

The proposed library is 1.9 times faster on average than the second fastest and over 90 times faster than the second most accurate library, Figure 12. Proposed has a stable runtime across scenarios Table II. Average runtime metrics hide erratic performance (i.e. very fast or slow on similar images) exhibited by some libraries, making them undesirable for real-time applications. One explanation for erratic

performance is poor handling of excessive false positives, especially in higher resolution images. See Figure 4 for an example of how images can be covered in corners.

Dataset	MP	N	FP	FN	E50 (px)	E100 (px)	R50 (ms)	R100 (ms)
border	12.2	16	0	0	0.29	1.71	103	151
caltech edges	0.3	20	0	1	0.26	0.89	11.9	15.7
challenge	1.4	85	1	20	0.23	4.51	45.3	66.9
close	0.6	12	0	0	0.42	1.40	6.7	8.7
kaist fisheye	1.9	58	0	3	0.21	1.08	26.4	41.6
distance angle	0.5	6	0	0	0.21	0.59	10.6	30.8
distance straight	0.5	8	0	0	0.18	0.67	10.0	16.6
distant	0.5	7	0	0	0.20	0.63	16.4	19.7
focus	0.5	36	0	0	0.44	2.89	7.2	17.2
focus large	12.2	16	0	0	1.89	13.87	118	216
gaus fisheye	0.5	12	1	3	0.18	1.61	5.5	7.3
gaus perfect	0.1	12	0	0	0.02	0.10	2.3	3.2
large	10.0	6	0	0	0.31	1.25	180	232
large shadow	12.2	23	0	0	0.42	3.77	127	605
motion blur	0.3	38	1	0	0.67	4.45	5.4	7.5
ocam fisheye190	0.4	8	0	1	0.22	1.16	8.3	13.0
ocam kaidan omni	0.3	17	0	1	0.26	1.03	10.3	15.2
ocam ladybug	0.8	13	0	0	0.26	0.84	10.8	16.1
ocam mini omni	0.4	15	0	0	0.22	3.25	17.9	27.7
ocam omni	1.2	14	0	0	0.29	1.11	27.0	36.1
perfect	0.3	8	0	0	0.00	0.01	7.5	11.0
theta 5	0.5	11	0	0	0.25	0.89	13.0	16.0
theta V	3.7	46	0	0	0.32	2.59	88.2	134
rotation flat	0.5	12	0	0	0.28	0.97	17.1	20.5
rotation vertical	0.5	7	0	0	0.24	0.66	9.8	15.8
shadow	0.5	15	0	0	0.22	0.63	16.7	28.0
sloppy13x10	2.1	13	0	0	0.42	2.17	34.1	43.6
DSC-HX5V	0.3	13	0	0	0.31	0.85	4.3	5.3
stefano 2012	0.3	30	0	0	0.18	0.83	10.2	13.8

TABLE II  
SUMMARY OF PROPOSED PERFORMANCE ACROSS ALL SCENARIOS

Proposed approach's detection, accuracy, and runtime performance in each scenario is shown in Table II. MP stands for image megapixels. When compared against other libraries in individual scenarios [31], the proposed is always a top performer. Making it the most consistent library.

## VI. CONCLUSION

A new chessboard detector is proposed that is designed to handle larger images while being fast, accurate, and robust. This is accomplished using a new x-corner detector, where for the first time, blur is detected and used to select the best level in scale-space for corner location, dynamically adjust sample points for edge validation, and determine corner connectivity. Several incremental improvements are also proposed, such as correct handling of x-corner symmetry and n-best edge score. Performance is validated across a large set of scenarios using hand labeled and simulated data in what might be the most challenging chessboard dataset yet. Only the proposed detector is a top performer in both overall and all individual scenarios (particularly in larger degraded images), while being the fastest library by a large margin.

## REFERENCES

- [1] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

- [2] J. Mallon and P. F. Whelan, "Which pattern? biasing aspects of planar calibration patterns and detection methods," *Pattern recognition letters*, vol. 28, no. 8, pp. 921–930, 2007.
- [3] M. Fišála and C. Šuh, "Self-identifying patterns for plane-based camera calibration," *Machine Vision and Applications*, vol. 19, no. 4, pp. 209–216, 2008.
- [4] B. Atcheson, F. Heide, and W. Heidrich, "Caltag: High precision fiducial markers for camera calibration," in *VMV*, vol. 10, 2010, pp. 41–48.
- [5] S. Garrido, "Charuco: Chessboard + aruco," <https://opencv.org/>, 2015.
- [6] Various, "Opencv library," <https://opencv.org>, 2000.
- [7] J.-Y. Bouguet, "Camera calibration toolbox for matlab," 2001.
- [8] L. Heng, B. Li, and M. Pollefeys, "Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1793–1800.
- [9] P. Abeles, "Boofcv's camera calibration tools," <http://boofcv.org/>, 2011–2021.
- [10] H. Ha, M. Perdoch, H. Alismail, I. So Kweon, and Y. Sheikh, "Deltille grids for geometric camera calibration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5344–5352.
- [11] V. Vezhnevets and P. Gruebele, "Part of the opencv image processing library: findchessboardcorners()," <https://opencv.org/>, 2010.
- [12] M. Rufli, D. Scaramuzza, and R. Siegwart, "Automatic detection of checkerboards on blurred and distorted images," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3121–3126.
- [13] S. Donné, J. De Vylder, B. Goossens, and W. Philips, "Mate: Machine learning for adaptive calibration template detection," *Sensors*, vol. 16, no. 11, p. 1858, 2016.
- [14] C. G. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, no. 50, 1988.
- [15] L. Lucchese and S. K. Mitra, "Using saddle points for subpixel feature detection in camera calibration targets," in *Asia-Pacific Conference on Circuits and Systems*, vol. 2. IEEE, 2002, pp. 191–195.
- [16] A. Kassir and T. Peynot, "Reliable automatic camera-laser calibration," in *Proceedings of the 2010 Australasian Conference on Robotics & Automation*. ARAA, 2010.
- [17] Y. Liu, S. Liu, Y. Cao, and Z. Wang, "Automatic chessboard corner detection method," *IET Image Processing*, vol. 10, no. 1, pp. 16–23, 2016.
- [18] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3936–3943.
- [19] A. Duda and U. Frese, "Accurate detection and localization of checkerboard corners for calibration," in *BMVC*, 2018, p. 126.
- [20] S. Placht, P. Fürsattel, E. A. Mengue, H. Hofmann, C. Schaller, M. Balda, and E. Angelopoulou, "Rochade: Robust checkerboard advanced detection for camera calibration," in *European Conference on Computer Vision*. Springer, 2014, pp. 766–779.
- [21] C. W. Niblack, P. B. Gibbons, and D. W. Capson, "Generating skeletons and centerlines from the distance transform," *CVGIP: Graphical Models and image processing*, vol. 54, no. 5, pp. 420–437, 1992.
- [22] J.-M. Lavest, M. Viala, and M. Dhorne, "Do we really need an accurate calibration pattern to achieve a reliable camera calibration?" in *European Conference on Computer Vision*. Springer, 1998, pp. 158–174.
- [23] W. Sun and J. R. Cooperstock, "Requirements for camera calibration: Must accuracy come with a high price?" in *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05)-Volume 1*, vol. 1. IEEE, 2005, pp. 356–361.
- [24] W. Förstner and E. Gülich, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, 1987, pp. 281–305.
- [25] A. Datta, J.-S. Kim, and T. Kanade, "Accurate camera calibration using iterative refinement of control points," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 2009, pp. 1201–1208.
- [26] V. N. Dao and M. Sugimoto, "A robust recognition technique for dense checkerboard patterns," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 3081–3084.
- [27] J. Shi and C. Tomasi, "Good features to track," in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, pp. 593–600.
- [28] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. IEEE, 1999, pp. 1150–1157.
- [29] A. P. Witkin, "Scale-space filtering," in *Readings in Computer Vision*. Elsevier, 1987, pp. 329–332.
- [30] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of IEEE conference on computer vision and pattern recognition*, 1994, pp. 593–600.
- [31] P. Abeles, "Yet to be published detailed technical report," in *TBD*, 2021.
- [32] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 5695–5701.
- [33] S. Mattoccia, "Stereo images from videredesign monochrome megad," <http://vision.deis.unibo.it/smart/stereo/Calibration.html>, 2006.
- [34] T. Peynot, S. Scheding, and S. Terho, "The marulan data sets: Multi-sensor perception in a natural environment with challenging conditions," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1602–1607, 2010.