A New Calibration Technique for Multi-Camera Systems of Limited Overlapping Field-of-Views

Ziran Xing, Jingyi Yu, Yi Ma

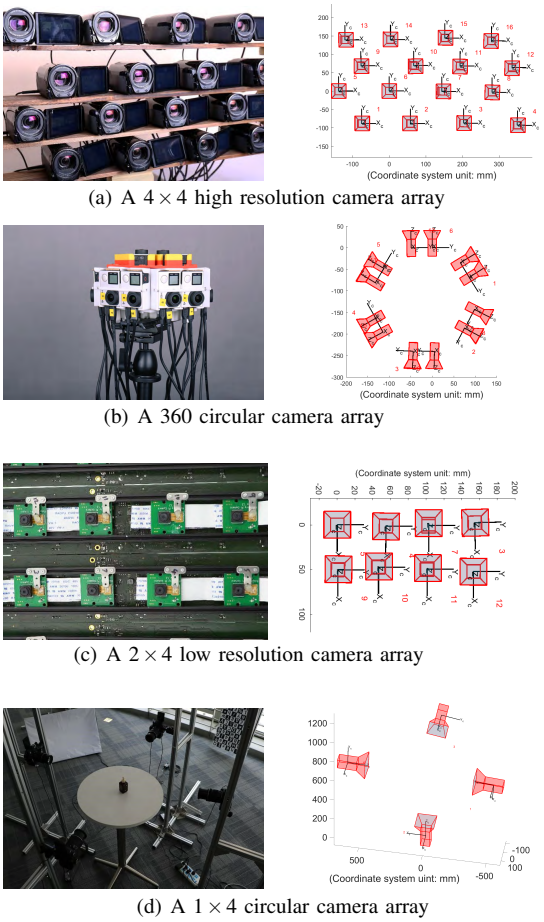
***Abstract*— State-of-the-art calibration methods typically choose to use a checkerboard as the calibration target for its simplicity and robustness. They however require the complete checkerboard be captured to break symmetry. More recent multi-camera systems such as Google Jump, Jaunt, and camera arrays have limited overlapping field-of-view (FoV) and having all cameras viewing the complete checkerboard is extremely difficult in reality. Tailored patterns such as CALTag [1] introduce new image features within the checker blocks for breaking symmetry but they also break the grid topology. We present a new technique using such patterned calibration targets for a broad range of multi-camera systems. Our key observation is that applying directional gradient filters yields to heterogeneous responses on grid vs. non-grid features: the former are isolated and the latter are highly inter-connected. We therefore apply a simple but highly efficient technique to eliminate non-grid outliers based on connected component analysis and gradient histograms. Finally, we recover the complete grid by approximating each local checkerboard as a parallelogram and imposing the topology constraint. We conduct comprehensive experiments on a number of recent multi-camera systems and our technique significantly outperforms the state-of-the-art in accuracy and robustness.**

I. INTRODUCTION

The first step in 3D computer vision tasks is camera calibration. State-of-the-art multi-camera calibration solutions unanimously choose to use a checkerboard as the calibration target for its simplicity and accuracy. The fundamental problem there is to establish correspondences across multiple cameras [2]. The process needs to be fully automatic and the solution needs to be efficient and reliable to noise, illumination inconsistencies, blurs, etc. The simplest choice of the calibration pattern is a checkerboard where reliable corner detection schemes can be directly used. However, the checkerboard patterns also introduce ambiguity: unless the complete checkerboard is captured, the correspondences exhibit ambiguity due to symmetry. Hence an implicit assumption in many state-of-the-art solutions is that the cameras have a large overlapping field-of-view (FoV).

More recent multi-camera systems such as Google Jump, Jaunt, and circular camera arrays employ a very different layout: the cameras are strategically separated to cover different parts of the scene. Such systems attempt to use as few cameras as possible to reduce both the cost and the storage/bandwidth, and therefore the cameras exhibit very limited overlapping FoV. Directly applying the state-of-the-art solutions such as [3], [4] to calibrating the cameras results in capturing many more images than the traditional binocular calibration case since a large proportion of partial checkerboard images are rejected. In short, multi-camera calibration is still far from being solved [5] especially on contemporary camera systems.

All the authors are with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China. {xingzr,yujingyi,mayi}@shanghaitech.edu.cn



(a) A 4×4 high resolution camera array

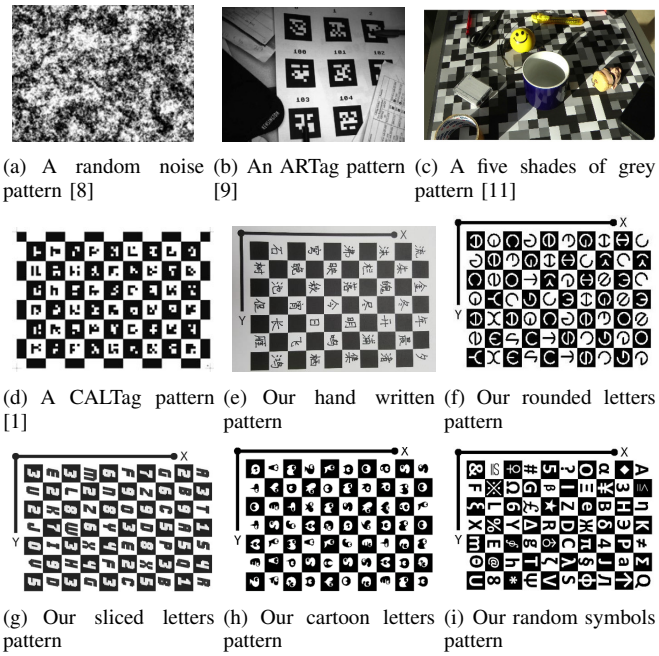
(b) A 360 circular camera array

(c) A 2×4 low resolution camera array

(d) A 1×4 circular camera array

Fig. 1. Real multi-camera systems and calibration results by our method.

The most straightforward approach to make partial images useful is to use textural patterns. For example, CALTag [1] exploits rich features on specially designed patterns to break symmetry. A downside though is that it would be difficult to impose the grid topology which is crucial for maintaining accuracy in calibration. We present a new simple but effective calibration technique suitable for a broad range of multi-camera systems by employing a much broader range of patterns. The key to our approach is that applying directional gradient filters yields to heterogeneous responses on grid vs. non-grid features: the former are scattered and



(a) A random noise pattern [8] (b) An ARTag pattern [9] (c) A five shades of grey pattern [11]

(d) A CALTag pattern [1] (e) Our hand written pattern (f) Our rounded letters pattern

(g) Our sliced letters pattern (h) Our cartoon letters pattern (i) Our random symbols pattern

Fig. 2. Examples of related self-identified patterns (a∼d) and our tagged checkerboard pattens (e∼i).

the latter are highly inter-connected. We then develop highly efficient techniques to eliminate the non-grid features based on connectivity and gradient histogram analysis and recover the complete grid via the topology constraint. We conduct comprehensive experiments on a number of recent multi-camera systems. In several cases, we show that by even adding hand written characters onto the checkerboard grid produces highly accurate calibration results (e.g., Fig. 15)). Our technique therefore provides more practical and effective alternative to the state-of-the-art.

II. RELATED WORK

According to early work of [6], [7], it is well known that one can recover both intrinsic and extrinsic parameters of a moving camera at the same time if we have at least three images of the same object (with sufficient feature points) assuming intrinsic parameters remain constant. This is known as camera self-calibration. Nevertheless, in practice it is rather difficult to ensure the accuracy of the parameters through self-calibration from arbitrary, unknown objects [2]. Especially, such self-calibration is not suitable for cases when the camera system cannot be easily moved due to cable restriction or fixtures; also such method does not apply to scenes where there might be multiple moving objects or non-rigid objects (such as human bodies).

Hence in practice, especially in applications where high accuracy of calibration parameters is needed (such as in Robotics or Virtual Reality), calibration is usually conducted beforehand with a pre-designed calibration rig (or pattern). The most classical designed pattern is a checkerboard. Thanks to its regular shape and high contrast appearance, not only can we conveniently compute the correspondence of corner features of the checkerboard with subpixel accuracy, but also can easily estimate the 3D pose of the checkerboard. There are many established methods that can automatically detect checkerboards in images [3], [4], [5]. However, images containing only an incomplete pattern cannot be used for estimating relative pose of the checkerboard due to ambiguity. In reality, ensuring every image containing the whole pattern is very difficult and sometimes impossible [5], especially when we calibrate a system of many cameras, such as the ones shown in Fig. 1.

To overcome this problem, the so-called self-identified patterns are introduced. That is, relative pose of the calibration rig can be estimated even if only part of pattern is detected and matched in images. Many self-identified patterns and detection methods have been introduced in the past several years. [8] has designed a random noise based planar pattern (see Fig. 2(a) for example). Based on this pattern, point correspondences can be solved by matching SURF features. However, feature matching is not robust enough in practice. Calibration may fail due to too many outliers (*see our experiment*).

[9], [10] have designed a pattern by regularly placing some square coded tags on a plane, and each square tag can be uniquely identified by its code (see Fig. 2(b)). [1] has extended this idea by placing tags in the square region of a checkerboard (see Fig. 2(d)). This not only makes the pattern more compact, but also leads to more accurate point location estimation for X-junctions [3]. However, these tag-based methods suffer when the camera is not ideal (e.g., low-resolution or high noise level) [3]. The main reason is that these methods are based on region detection. Since region detection is usually less reliable than edge and corner detection, that makes such tag-based methods more prone to failures in practice than the checkerboard.

[11] has designed a square-based pattern by encoding identity information through variation of neighboring squares' shade of grey (see Fig. 2(c)). Although detecting this pattern can help to estimate camera pose even in real time, this method is not quite suitable for camera calibration since it relies on line fitting and vanishing point estimation which do not work so well when there is large lens distortion [3].

III. TCAD: TAG-FLEXIBLE CHECKERBOARD ADVANCE DETECTOR

In this work, we propose to use the CALTag like pattern, but instead of binary codes, our method allows many different types of identity tags or textures to be used in the square regions of the checkerboard. Since we are flexible with the tags, it is difficult to use any region based methods for detection. So we are compelled to use the low-level corner features for correspondence and recovering geometry. Since each image may only contain a partial region of the whole pattern, one needs to detect as many as possible corners of the visible part of the checkerboard. As the tags inside the squares also contain many "corner-like" feature points, this poses a good challenge: how to extract the true corners of

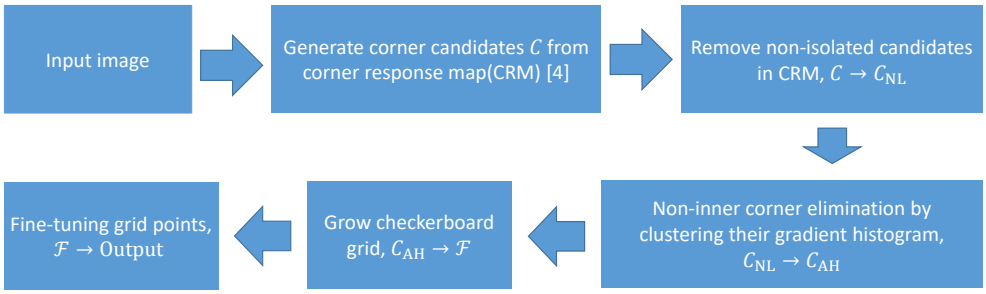


Fig. 3. The processing pipeline of TCAD.

the checkerboard despite all the additional corner features? Our method is essentially to address this challenge.

To produce candidate corner points, we use the same corner detector as the Matlab build-in function *detectCheckerboardPoints* [12] which is based on [4]. The work of [4] is essentially developed for plain checkerboard pattern: first some corner candidates are found and then a rectangular grid is grown from these candidates. To apply the same idea to the checkerboard filled with tags, we have designed several additional techniques in order to robustly detect corners of the checkerboard among all corner candidates from the tags. The overall pipeline of our method is shown on Fig.3.

A. Preprocessing

The preprocessing step is to produce a set of possible corner candidates, denoted as C. For each image, we generate two sets of candidates from the corner response maps (CRM) produced by the method of [4] using the 0-degree filter and 45-degree filter, respectively. Fig. 4(b) shows an example of the response map and the suggested corner candidates. The CRM has the same size as the input image, and its value at each pixel represents the likelihood of the pixel to be an inner corner of the checkerboard.

B. Filtering Corner Candidates

As the example shown in Fig. 4(b), due to the filled tags, the CRM produces too many candidates (363 in this case) but only a few of them are the desired corner of the checkerboard (54 in this case). In order to reliably grow the checkerboard grid, all the false candidates introduced by points on the tags should be filtered out.

1) Eliminate large corner clusters in CRM: According to the preprocessing step, inner corners of a checkerboard should be distinctive peaks on CRM. By taking advantage of this property, many incorrect candidates could be removed. We treat CRM as an image, and use Otsu's method [13] to binarize it (see Fig. 4(c)). As we can see, each inner corner represents a highly localized cluster on this image and yet most corner features from the tags form larger clusters. So to detect and remove such (false) large clusters of corners from the tags, we build a graph G from this binary image, with vertices V and edges E defined as follows:

$$G = (V, E), \tag{1}$$

$$V = \{v_i | v_i \in \mathrm{bw}(\mathrm{CRM}), v_i = 1\}, \tag{2}$$

$$E = \{(v_i, v_j) | v_i \in V, v_j \in V, \mathrm{norm}(v_i, v_j) \leq \sqrt{2}\} \tag{3}$$

where $\mathrm{bw}(\mathrm{CRM})$ is the binarized image of CRM. We then remove candidates belonging to large connected components of this graph.

More formally, we want to find the subset of the original candidates which satisfied:

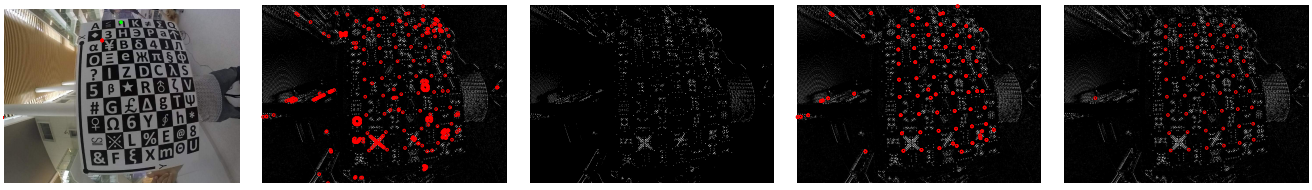
$$C_{\mathrm{NL}} = \{c_i | c_i \in C, \max_{v_j \in V}(\mathrm{dist}(G, c_i, v_j)) \leq L_{\max}\} \tag{4}$$

where $\mathrm{dist}(G, c_i, v_j)$ means the shortest distance from c_i to v_j on G, and $L_{\max}$ is the maximum threshold value of the shortest distance. Since the graph has the same edge weight, we can use BFS (Breadth-First-Search) to compute the length of all paths starting from a candidate to other vertices in its component. Therefore the remaining set C_{NL} does not contain candidates which has a connected path larger than the threshold $L_{\max}$ (see Fig.4(d)).

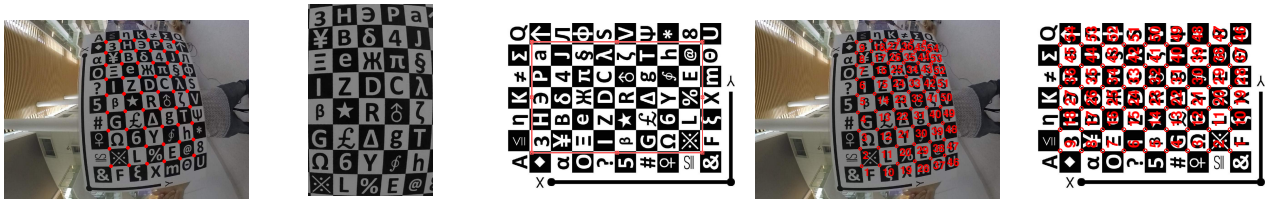
2) Filter corner candidates by gradient histogram: After the previous step, there are still some unwanted candidates since they might belong to a small connected component after binarization. In this step, we further remove false candidates whose gradient characteristic does not resemble that of a checkerboard corner. [4] adjusts the score of each candidate based on how the gradient of the candidate matches that of a correct corner. It works for relatively clean checkerboard. However, such simple method is not robust enough to work for our images (as we will compare extensively in our experiments).

To significantly improve the robustness of checkerboard corner detection, we need to utilize the characteristics of gradients around the candidates in much greater details. In addition, since in our images, most corners (including those of the checkerboard) are subject to significant perspective and lens distortion, it is difficult to generate a template (as [4] did) for the corners of the checkerboard. We need to rely on the consistency of the gradient characteristics of the correct corners to remove all other outliers.

a) Fast gradient histogram computation: We compute the gradients around a candidate in its $n \times n$ neighborhood (by default $n = 20$). To compute the histogram of the gradients, we discretize 360 degrees into 64 bins, and the value of each bin represents the number of pixels whose gradient direction is within $\frac{1}{2} \cdot \frac{2\pi}{\#Bins}$ of this bin.



(a) Input image (b) Corner response map (CRM) and corner candidate set C (c) Binary CRM (d) Removed candidates on large clusters (the set C_{NL}) (e) Removed candidates after histogram filtering (the set C_{AH})



(f) Grown checkerboard (the set $\mathscr{F}$) (g) Warped detected point grid (h) The location of detected point grid (i) Identified corners (j) The pattern template

Fig. 4. Intermediate results of TCAD. Only the results under 0 degree filter are demonstrated here for example, but both 0 degree and 45 degree filters are in the process actually.

Counting gradients in $n \times n$ neighborhoods for all candidates, it requires $O(|C_{\mathrm{NL}}| \cdot n^2)$ running time, where $|C_{\mathrm{NL}}|$ means the size of the set C_{NL}.

To reduce the computational cost, we borrow the idea from integral histogram [14]. Denote H as the histogram operator $H(R) \equiv h$, where R represents an image region and h is the histogram of R. We build a histogram area table (HAT) and each element of HAT represents $\mathrm{HAT}(i,j) \equiv H(R(i,j))$, where $R(i,j) = \{I(x,y) | x \le i, y \le j\}$ in which I is the input image, and it can be efficiently computed by

$$\mathrm{HAT}(i,j) = H(I(i,j)) + \mathrm{HAT}(i-1,j) + \mathrm{HAT}(i,j-1) - \mathrm{HAT}(i-1,j-1). \quad (5)$$

As we see, it takes $O(|I|)$ time to compute HAT, where $|I|$ represents the number of pixel of I. Nevertheless, notice that HAT can be computed in our preprocessing step where we need image gradients to produce the corner candidates. Once we have HAT, it is easy to verify that the histogram of any rectangular region can be computed in $O(1)$ time by

$$H(R) = \mathrm{HAT}(D) + \mathrm{HAT}(A) - \mathrm{HAT}(B) - \mathrm{HAT}(C), \quad (6)$$

where R is a region and A, B, C, D are the four vertexes of R (as shown in Fig.5).

b) Histogram clustering: Since typical corners of the checkerboard resemble a saddle point or an X-junction [3], their histograms will has four dominant bins which other random corners do not have (see Fig. 6). From our experience, conventional similarity measures (such as χ^2 distance)

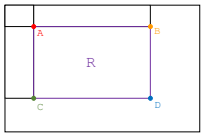
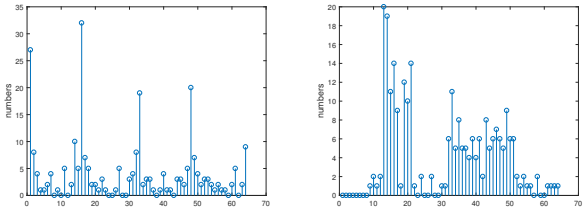


Fig. 5. Demonstration of the relationship between H and HAT.



(a) The gradient histogram of the red point in Fig.4(a) which is an inner conner of the checkerboard (b) The gradient histogram of the green point in Fig.4(a) which is NOT an inner conner

Fig. 6. The difference of histograms between inner corner's and unwanted candidate's.

between histograms are not robust enough as the peaks of the histogram of different corners of the checkerboard may shift due to distortion. Hence, we need a procedure that explicitly identify and match those peaks, and measure how close those peaks are in orientation.

To this end, we extract four peaks of a histogram h_{c_i} for each candidate $c_i \in C_{\mathrm{NL}}$ and denote them as $D(h_{c_i}) \equiv \{d_1, d_2, d_3, d_4\}$, where d_i $(i = 1,2,3,4)$ is the i-th largest bin of h_{c_i}. The similarity between the histograms of two candidates c_i and c_j can be defined as

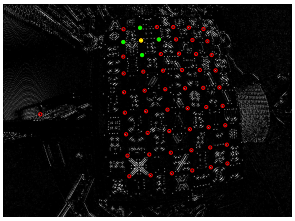
$$S(c_i, c_j) = \begin{cases} 1 & \max(D_A(h_{c_i}, h_{c_j})) < \mathrm{Th}_{\mathrm{angle}}, \\ 0 & \text{otherwise}, \end{cases} \quad (7)$$

where $D_A(h_{c_i}, h_{c_j})$ means the angle difference of any one-one matching between $D(h_{c_i})$ and $D(h_{c_j})$. Based on this measurement, we can define the similarity set of c_i as

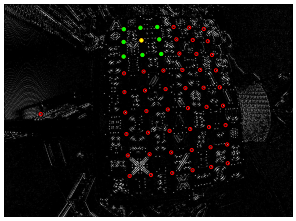
$$T(c_i) \equiv \{c_j | c_j \in C_{\mathrm{NL}}, S(c_i, c_j) = 1\}. \quad (8)$$

The largest similarity set then corresponds to the set of corners on the checkerboard which we denote as

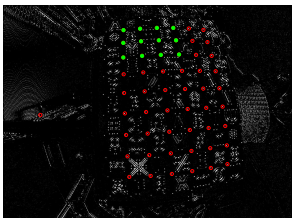
$$C_{\mathrm{AH}} = \arg\max_{T(c_i), c_i \in C_{NL}} |T(c_i)|. \quad (9)$$



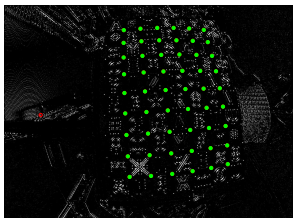
(a) A seed point with its 4 neighbors forming a cross



(b) Growing into a 3×3 point grid



(c) Growing into a 3×4 point grid by expanding a boundary of (b)



(d) The largest point grid for the initial candidate seed

Fig. 7. Intermediate results of checkerboard growing.

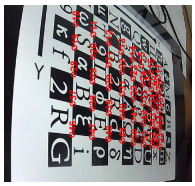
The remaining clusters (points) are all removed. Fig.4(e) shows the result of this step applied to a real image.

C. Growing Checkerboard

As we can see from Fig. 4(e), after the previous filtering steps, there might still be a few outliers left. Notice that so far we have only used local information in previous processing/filtering steps, we now use the global structural information of checkerboard grid to remove the remaining outliers and reconstruct the visible part of the grid. We follow a similar procedure in [4] for recovering the grid, but we here utilize some new techniques to select the candidates and grow the grid more robustly (see Fig. 7 for an illustration of the procedure).

Denote the set of identified inner corners of the checkerboard as $\mathscr{F}$ and initially $\mathscr{F}=\emptyset$. First, a candidate with high score to be an inner corner will be selected as the seed point and added into $\mathscr{F}$. Second, using the 4 directions identified from the histogram clustering in previous step, we can grow $\mathscr{F}$ by finding 4 candidates which not only align with these 4 directions, but also closest to the seed point. After that all points in $\mathscr{F}$ will form a cross. By using our selection method, the initial 3×3 point grid will be generated. We then repeatedly expand the boundary of the point grid until we can no longer make it larger.

Our contribution here is to robustly select candidates in each growing step. While finding candidates along lines of the existing grid $\mathscr{F}$ is easy, it is difficult to locate points off such lines. Notice that the image of each small square of a checkerboard is nearly a parallelogram, even under large lens distortion or severe perspective. So whenever we need to find a candidate point off the lines of the existing grid, we enforce the newly selected point to form a parallelogram with points in $\mathscr{F}$.



(a) Detected grid points

(b) Before fine tuning

(c) After fine tuning

Fig. 8. Example of an inaccurate grid point. (b) Point number 15 is not exactly a corner of the checkerboard. (c) Point location fine-tuned by our method.

$$\arg\min_{c_i\in C_{AH}} \frac{E_s(c_i)}{E_l(c_i)}$$
$$\text{s.t. } E_s(c_i)=\frac{\|e_1+e_2-e_3\|}{\|e_1-e_2\|}, E_l(c_i)=\text{CRM}(c_i). \quad (10)$$

where

$$e_1=c_i-P_1,\quad P_1=\arg\min_{f\in\mathscr{F}}(\text{dist}(c_i,f)), \quad (11)$$
$$e_2=c_i-P_2,\quad P_2=\arg\min_{f\in\mathscr{F}-\{P_1\}}(\text{dist}(c_i,f)), \quad (12)$$
$$e_3=c_i-P_3,\quad P_3=\arg\min_{f\in\mathscr{F}-\{P_1,P_2\}}(\text{dist}(c_i,f)). \quad (13)$$

Small value of $E_s(c_i)$ indicates that c_i satisfy the parallelogram constrain. $E_l(c_i)$ indicates the likelihood of c_i being an inner corner (from CRM). The optimal solution of this problem will be the candidate added to $\mathscr{F}$.

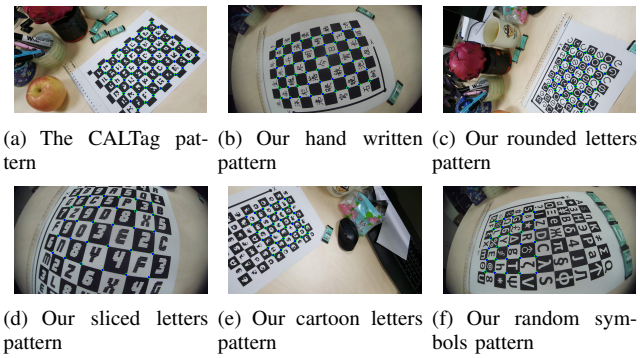
D. Fine-tuning Inaccurate Grid Points

Although very unlikely to happen, a few points in the grid grown might belong to incorrect candidates (see Fig. 8 for example). The main reason is that some poorly designed tags may produce corners very similar to those of the checkerboard.

In order to correct such errors and fine tune the detected grid, we know from [15] that homogeneous coordinates of a point on the planar calibration rig x_i^c and its image coordinates x_i^d satisfy the constraint $x_i^{dT}Fx_i^c=0$ for some matrix $F\in\Re^{3\times3}$. We can robustly estimate F using RANSAC since most of the points in the detected grid are correct. For points that violate this constraint for the estimated F, we search around its neighborhood in C for x_i^d that satisfies this constraint. Fig. 8(c) shows an example of the fine-tuning result.

E. Identify Detected Grid Points

Since the checkerboard pattern is textured with tags, may methods can be used to identify the detected grid points against the original calibration pattern. For challenging conditions, one could use powerful features such as SIFT or classification tools such as DNN (Deep Neural Networks) to recognize those tags. But based on our experience with all the experiments presented in this paper, we found that a simple patch-based matching technique suffices our purpose. We first correct the perspective and scale distortion by warping



(a) The CALTag pattern (b) Our hand written pattern (c) Our rounded letters pattern

(d) Our sliced letters pattern (e) Our cartoon letters pattern (f) Our random symbols pattern

Fig. 9. Patterns of our Dataset with grids detected by our method.

the image with a homography estimated from detected point grid and a standard grid (see Fig. 4(g) for an example). Then we use NCC (Normalized Cross Correlation) to match the regions of the warped image to those of the calibration pattern. Once the regions are matched, the identity of each point (against the calibration pattern) can be found (see Fig. 4(i)).

IV. Experimental Evaluation

We evaluate our method TCAD by comparing with the state-of-the-art calibration methods, namely Matlab [4], OCPAD [5], and CALTag [1] on 1. some popular standard datasets and our own dataset (that further tests robustness of various methods); 2. calibrating several real-world multi-camera systems (as those shown in Fig. 1).

A. Evaluation on Calibration Datasets

The datasets we used in this paper are: various standard checkerboard datasets used in [3], [5] and our tagged checkerboard dataset.[1] We measure the detection performance by counting the number of images with missed detections (MD, reporting nothing on a valid image), false detections (FD, reporting at least one false detected corner), and successful detections (SD, reporting a point grid without error).

The detection performance of all the methods on all datasets are summarized in Table I (Each method we used for comparison is implemented by their authors). Although Matlab and OCPAD slightly outperform our method on the low-quality dataset MesaSR4000, our method is the only method that performs perfectly on the remaining datasets.

In order to evaluate our method's performance and robustness under different resolution, distortion, and pattern designs, we build a comprehensive dataset of images taken with a cellphone camera (5312×2988), a cheap fisheye camera (1920×1080), and GoPro Hero 4 (under WVGA mode, 848×480). Six tagged patterns are used for our dataset, as shown in Fig.9. Except for the CALTag pattern, 84 images are taken for each pattern by the three cameras (28 images each).The number of CALTag pattern images (Fig. 9(a)) is 155, which is more than the others. The reason is to give more thorough comparison with the original CALTag method [1]. The evaluation results are shown and compared in Table I. As one can see, our method significantly outperforms all existing methods on this dataset. Out of all the 575 images, and our method only failed on five whereas other methods all fail badly on some (or all) of the patterns or on images taken by some cameras. This suggests our method is indeed very robust to practical variations of real cameras.

Our method is designed for off-line calibration tasks, therefor reducing the running time is not our first priority. However, we show the running time of our method in Table II for those readers who care about real time applications. Our method is implemented in MATLAB (not optimized) and the test platform is a desktop computer with a Core i7 CPU (Intel Core i7-6950X) and 128 GB of DDR4 memory.

B. Evaluation on Real Multi-camera Systems

We now evaluate and compare our method by calibrating several real multi-camera systems for various practical :

- I The first one is a large baseline light-field system which consists of 4×4 Canon DV cameras, and each camera's resolution is 1920×1080 (see Fig. 1(a)).
- II A 360 circular camera array for 3D panorama which consists of 12 GoPro Hero 4 and each cameras' resolution is 1440×1080 (see Fig. 1(b)).
- III A 2×4 low-resolution webcam array and each cameras' resolution is 192×144 (see Fig. 1(c)).
- IV A 1×4 circular camera array for reconstruction which consists of 4 high resolution Canon DSLR (Digital Single Lens Reflex) cameras and each camera's resolution is 6000×4000 (see Fig. 1(d)).
- V A self-made stereo cameras are consist of 2 GoPro Hero 4 (under 5MP mode, 2560×1920), with a baseline about 30cm. The calibration rig is simply some words written on a standard checkerboard printed on an A4 paper.

We evaluate our method by calibrating these camera systems and comparing with existing methods such as the random pattern based method and CALTag. Since system I, II, III and V are made up of pre-calibrated cameras with fixed focal length and system IV is made up of DSLR cameras with adjustable focal length, our calibration tasks are (i) extrinsic calibration of system (I-III,V) and (ii) joint intrinsic and extrinsic calibration of system IV. The calibration results are compared by measuring the mean vertical parallax (MVP) of each pair after epipolar rectification. Since the random pattern based method fails to calibrate the 4×4 camera array, we only compares the result of our method with CALTag [1]. Since our method can also work on the CALTag pattern, we use only the CALTag pattern to compare, to be fair for [1].

a) The 4×4 light-field system: Since for this system, the baseline is large (400mm) and each camera's FoV is quite small (about 50 deg in horizontal and 30 deg in vertical), it is impossible to ensure all cameras see the entire calibration pattern. In this experiment, we randomly move the CALTag pattern in front of the camera array and capture 20 images for

[1]We will release our dataset and code upon publication.

TABLE I

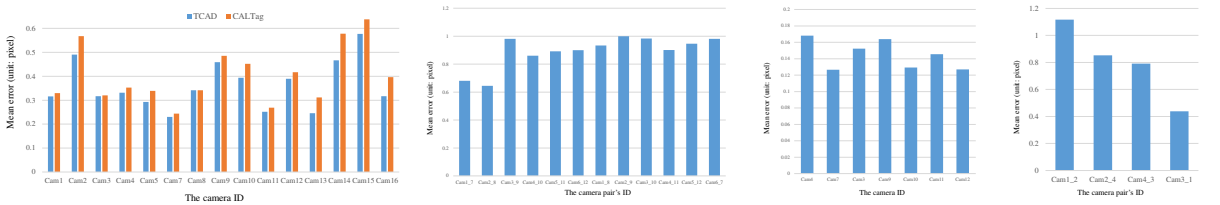
COMPARISON OF DETECTION RESULTS FOR MATLAB [4], OCPAD [5], CALTAG [1] AND OUR METHOD (TCAD). THE PERFORMANCE IS MEASURED BY COUNTING THE NUMBER OF IMAGES WITH MISS DETECTIONS (MD), FALSE DETECTIONS (FD), AND SUCCESSFUL DETECTIONS (SD).

Datasets \ Methods		number of images	Matlab[4]			OCPAD[5]			CALTag[1]			TCAD		
			MD	FD	SD	MD	FD	SD	MD	FD	SD	MD	FD	SD
MesaSR4000(StereoWideBaseline)		206	1	2	**203**	5	0	201	206	0	0	11	2	191
uEye(StereoWideBaseline)		206	0	0	**206**	0	0	**206**	206	0	0	0	0	**206**
GoPro Hero3		100	0	1	99	23	0	77	100	0	0	0	0	**100**
OCPAD(full)		162	0	0	**162**	3	0	159	162	0	0	0	0	**162**
OCPAD(partial-full)		162	0	4	158	95	0	67	162	0	0	0	0	**162**
Our Dataset	CALTag Pattern 9(a)	155	27	64	64	129	0	26	27	4	124	1	0	**154**
	Hand Written Pattern 9(b)	84	31	38	13	52	1	31	84	0	0	1	0	**83**
	Rounded Letters Pattern 9(c)	84	18	50	16	70	0	14	83	1	0	0	0	**84**
	Sliced Letters Pattern 9(d)	84	21	55	8	76	0	8	83	1	0	1	1	**82**
	Cartoon Letters Pattern 9(e)	84	60	22	2	66	0	18	84	0	0	1	0	**83**
	Random Symbols Pattern 9(f)	84	18	53	13	83	0	1	84	0	0	1	0	**83**

TABLE II

COMPUTATION TIMES OF TCAD. THE PERFORMANCE IS MEASURED BY THE AVERAGE AND MAXIMUM COMPUTATION TIMES (AT AND MT).

	MesaSR 4000	uEye	GoPro3	OCPAD (full)	OCPAD (partial)	Pattern 9(a)			Pattern 9(b)			-
						phone	fisheye	GoPro4	phone	fisheye	GoPro4	-
AT	2.05s	4.02s	4.67s	3.61s	2.52s	5.98s	5.74s	4.63s	6.15s	5.69s	4.13s	-
MT	6.80s	6.86s	18.89s	7.77s	4.14s	12.52s	10.10s	9.95s	9.87s	9.03s	6.86s	-
	Pattern 9(c)			Pattern 9(d)			Pattern 9(e)			Pattern 9(f)		
	phone	fisheye	GoPro4	phone	fisheye	GoPro4	phone	fisheye	GoPro4	phone	fisheye	GoPro4
AT	5.06s	4.63s	4.41s	5.82s	6.09s	5.02s	5.48s	5.59s	4.33s	5.01s	5.27s	4.72s
MT	8.37s	8.43s	8.73s	7.99s	11.54s	6.98s	8.33s	10.01s	7.37s	7.63s	9.07s	9.51s



(a) MVP for the 4×4 light field system (b) MVP for the 360 panorama system (c) MVP for the 2×4 camera system (d) MVP for the 1×4 DLSR camera system

Fig. 10. Compare the mean vertical parallax (MVP) after rectifying the images captured by three real camera systems (see Fig. 1). Since CALTag fails in detecting patterns on images from the 360 panorama system and the 2×4 low-resolution light field system, we only show results of our method in (b) and (c).

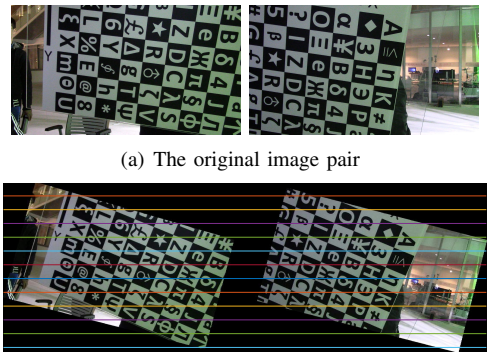
each camera synchronously. We use these images to estimate camera poses and take another 20 images of our tagged pattern to verify the rectification results. Fig. 11 shows an example of rectification result based on our method. In addition, Fig. 10(a) compares mean vertical parallax (MVP) for CALTag and TCAD. Cam6 was picked as the reference camera and other cameras are rectified against it. As we see, TCAD's mean error is always less than 0.6 pixel for each camera, consistently less than CALTag.

b) The circular 360 panorama system: Calibrating this system is more tricky than the first one since GoPro cameras have large lens distortion and the overlap region is even smaller. We capture 26 images by moving the calibration pattern around the systems. Since the CALTag method fails, we only show the results of TCAD in Fig. 12 and Fig. 10(b).

c) The low-resolution 2×4 *light field system:* Calibrating this system is more difficult than others. The main reason is that CALTag cannot even detect the pattens from this system due to its low resolution. The experiment settings are exactly same to the first one, and we show the results of TCAD in Fig. 13 and Fig. 10(c), and the Cam5 is the reference camera and other cameras are rectified against it.

d) The 1×4 *DLSR camera system:* Calibrating this system requires more human effort than other systems. We capture 40 images by randomly moving the pattern inside the shooting scope. Each camera's intrinsic parameters are computed at first, then relative poses of these cameras are recovered as the second system. We show the results of TCAD in Fig. 14 and Fig. 10(d).

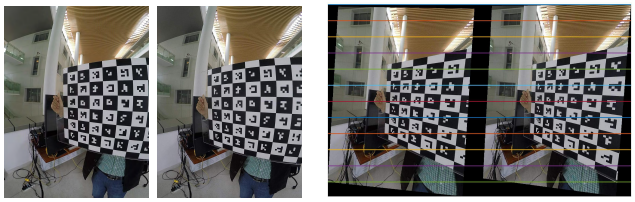
e) The stereo pair by hand-writing pattern: The hand writing pattern used here is shown in Fig. 9(b), some ancient Chinese characters are written on a checkerboard printed on an A4 paper. 10 images are used to estimate the camera poses, and the rectification mean error is 0.3415 pixel on the remaining 18 image pairs. One rectification result is shown as Fig. 15.



(a) The original image pair

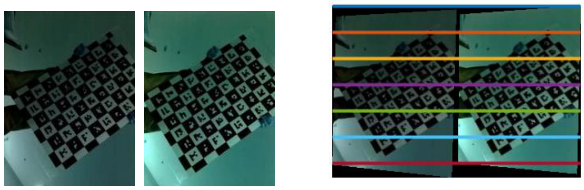
(b) The rectified image pair

Fig. 11. A rectification result of the 4×4 light field system.



(a) The original image pair (b) The rectified image pair

Fig. 12. Rectification result of the 360 panorama system.



(a) The original image pair (b) The rectified image pair

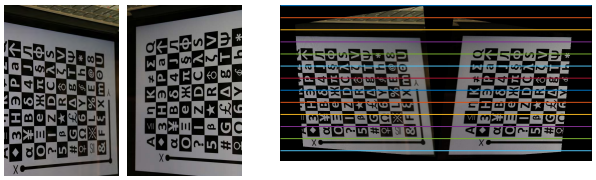
Fig. 13. Rectification result of the 2×4 light field system.

V. Conclusion

Due to the emerging applications of VR and AR, the demand for high-accuracy multi-camera systems is increasing explosively. However people continue to struggle with accurately calibrating a multi-camera system under diverse practical conditions [5]. In this paper we have developed a robust and accurate calibration method TCAD (including both calibration patterns and algorithms) that addresses various practical difficulties in calibration: diverse camera configurations, severe perspective, large lens distortion, low-resolution, and limited FoV and incomplete images of the calibration pattern, etc. Through extensive experiments on standard calibration datasets and on our own dataset and multiple practical camera systems, the new method significantly outperforms existing calibration methods in terms of both accuracy and robustness.

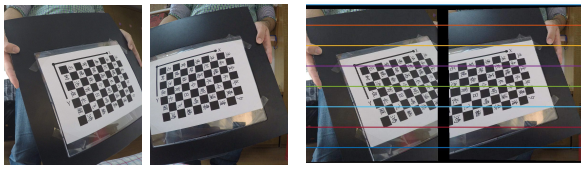
References

[1] B. Atcheson, F. Heide, andW. Heidrich. CALTag: High Precision Fiducial Markers for Camera Calibration. In Vision, Modeling, and Visualization Workshop 2010, Siegen, Germany, pages 41-48, 2010.



(a) The original image pair (b) The rectified image pair

Fig. 14. Rectification result of the 1×4 DLSR camera system



(a) The original image pair (b) The rectified image pair

Fig. 15. Rectification result between self-made stereo system.

[2] Z. Zhang. A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis & Machine Intelligence, 22(11):1330-1334, 2000.

[3] S. Placht, P. Fürsattel, E. A. Mengue, H. Hofmann, C. Schaller, M. Balda, and E. Angelopoulou. ROCHADE: Robust Checkerboard Advanced Detection for Camera Calibration. Springer International Publishing, 2014.

[4] A. Geiger, F. Moosmann, ÖCar, and B. Schuster. Automatic camera and range sensor calibration using a single shot. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 3936-3943, 2012.

[5] P. Fuersattel, S. Dotenco, S. Placht, M. Balda, A. Maier, and C. Riess. OCPAD: Occluded checkerboard pattern detector. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1-9, 2016.

[6] Q. T. Luong and O. D. Faugeras. Self-Calibration of a Moving Camera from Point Correspondences and Fundamental Matrices. International Journal of Computer Vision, 22(3):261-289, 1997.

[7] S. J. Maybank and O. D. Faugeras. A theory of selfcalibration of a moving camera. International Journal of Computer Vision, 8(2):123-151, 1992.

[8] B. Li, L. Heng, K. Koser, and M. Pollefeys. A multiplecamera system calibration toolbox using a feature descriptorbased calibration pattern. In 2013 IEEE/RSJ International Conference on Intelligent Robots & Systems, pages 1301- 1307, 2013.

[9] M. Fiala. ARTag, a fiducial marker system using digital techniques. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 590-596 vol. 2, 2005.

[10] M. Fiala and C. Shu. Self-identifying patterns for planebased camera calibration. Machine Vision and Applications, 19(4):209-216, 2008.

[11] A. Herout, I. Szentandrsi, M. Zachari, M. Dubsk, and R. Kajan. Five Shades of Grey for Fast and Reliable Camera Pose Estimation. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1384-1390, 2013.

[12] Matlab 2016b. Documentation: detectCheckerboardPoints. http://cn.mathworks.com /help/vision/ref/detectcheckerboardpoints.html, 2017.

[13] N. Otsu. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1):62-66, 1979.

[14] Porikli, Fatih. Integral histogram: A fast way to extract histograms in cartesian spaces.n Computer Vision and Pattern Recognition (CVPR), 2005 IEEE Conference on, pages 829-836, 2005.

[15] R. Hartley and S. B. Kang. Parameter-free radial distortion correction with center of distortion estimation. IEEE Transactions on Pattern Analysis & Machine Intelligence, 29(8):1309-1321, 2007.