Robust Checkerboard Recognition for Efficient Nonplanar Geometry Registration in Projector-camera Systems

Weibin Sun* Xubo Yang† Shuangjiu Xiao‡ Wencong Hu§
DALAB of School of Software, Shanghai Jiao Tong University

Abstract

Projector-camera systems always need complicated geometry calibration to get a correct display result on nonplanar projection surface. Geometry registration of most calibration methods dealing with arbitrary surfaces is done by projecting a set of structure light patterns or by manually 3D modeling, which are both time-consuming. In this paper, we propose a robust checkerboard calibration pattern recognition method to help nonplanar surface geometry registration. By approximating the nonplanar surface to be composite of many planar quad patches, pixels mapping between the calibration camera and a projector can be got by projecting only one checkerboard calibration pattern recognized by our method. Compared with geometry registration with structure light or encoded points, which need project many images, our method can be more efficient. Our recognition method has two steps: corner detection and checkerboard pattern match. Checkerboard internal corners are defined as special conjunction points of four alternating dark and bright regions. A candidate corner’s neighbor points within a rectangular or a circular window are treated as in different one-point-width layers. By processing the points layers in corner detection, we transform the 2D points distribution into 1D, which simplifies the regions amount counting and also improves the robustness against noises caused by deformation and complex illumination. After corner detection, the pre-known checkerboard grids rows and columns amounts are used to match and decide the right checkerboard corners from the results that have found. Regions boundary data produced during the corner detection also assist the matching process.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Object recognition;

Keywords: checkerboard recognition, geometry registration, projector-camera system

1 Introduction

Geometry registration in projector-camera systems (ProCams) is mainly a process that builds pixels mapping between camera view and projector view. The mapping can be a homography between those two views when dealing with planar projection surface, or be a pixel look-up table in arbitrary surface cases. [Brown et al. 2005] and [Bimber and Raskar 2006] introduce the registration details for both planar and arbitrary surface cases. Planar surface geometry registration can be done by a simple 2D homography computed by projecting a tessellated pattern. To get pixels mapping in arbitrary nonplanar surface cases, structure light technique or manually modeling can be adopted. [Zollmann et al. 2007] uses encoded points to be more robust when detecting but need at least \( \log_2(\text{points count}) \) images. Embedded structure light pattern [Oliver Bimber and Grundhofer 2007] can be time-saving without interruption for projecting patterns when doing registration. However, the registration methods mentioned above are all need more than one pattern images. The reason is that they have to recognize corresponding points between camera and projector views: structure light need more than one image to decide deformation, encoded points need unique binary code to be recognized distinctly. In this paper, we propose a robust recognition method for checkerboard calibration pattern to make it possible to build pixel mapping in geometry registration on nonplanar projection surface using only one pattern image.

Checkerboard pattern is often used in camera calibration [Zhang 2000]. Its alternating bright and dark grids and grid corners features can be a very strong features to detect and recognize distinctly. However, current checkerboard recognition methods in camera calibration can work only on planar calibration surface with a little deformation in camera view. In Zhang’s method [Zhang 2000], the corners are found by intersecting lines. The drawback of this approach is that edges may be in general curved due to radial distortions or deformation. Furthermore, the subsequent ordering of the corners into a regular grid can be complex and unreliable. Bouguet [Bouguet 2000] proposed an interactive method to find the internal corners of planar checkerboard pattern image. This method includes an automatic mechanism for counting the number of squares in the grid and predicting the grid corners. This tool is especially convenient when working with a large number of images. However, the user needs to click on the four extreme corners on each rectangular checkerboard pattern image in order to calculate the data of the corners. The OpenCV [Intel ] function, cvFindChessboardCorners, which is widely used, can do automatic corner extraction, but the algorithm fails rather often under complex illumination and deformation. Wang et. al. [Wang et al. 2007] proposed an approach to automatically recognize and locate the internal target corners of the planar checkerboard pattern image. The proposed approach is based on the characteristics of local intensity and the grid line architecture of the planar checkerboard pattern image. Shu et. al. [Shu et al. 2003] proposed a method, which is based on the algorithm of Watson [Watson 1981], that exploits the topological structure of the checkerboard pattern. The main idea is to use Delaunay triangulation [Bern and Eppstein 1992] to connect the corner points. It can deal with different lighting conditions but also only planar checkerboard pattern.

Our recognition method will deal with deformed checkerboard pattern on nonplanar surfaces and also, the natural complex illumination condition is concerned. The main idea of our checkerboard pattern detection is to treat a certain point’s neighbor points within
a rectangular or a circular window as in different one-point-width layers and transform the 2D points distribution into 1D to detect regions. Corners are correlated and clustered by the region boundary data to recognize the checkerboard corners.

2 Notations and Motivation

Let \( I \) be the image containing the checkerboard pattern.

\[
I = [\tilde{p}], \quad \tilde{p} = [x, y] \text{ denotes the point in } I.
\] (1)

For a vector point \( \tilde{p} \), we use \((x_{\tilde{p}}, y_{\tilde{p}})\) to represent its Cartesian coordinates, \((\tilde{p}_x, \tilde{p}_y)\) to represent its polar coordinates. In a grayscale image, we make \( I(\tilde{p}) \) or \( I(x_{\tilde{p}}, y_{\tilde{p}}) \) be the intensity of point \( \tilde{p} \), \( I_B(\tilde{p}) \) or \( I_B(x_{\tilde{p}}, y_{\tilde{p}}) \) be the intensity of point \( \tilde{p} \) in a binary image.

A checkerboard corner, which we call it the region corner (Fig. 1(a)), has four alternating dark and bright regions around it. We define a rectangular and a circular window covering the four regions feature surrounding a candidate corner \( \tilde{p} \) to be \( R(\tilde{p}, w) \) and \( C(\tilde{p}, w) \) in eq. 2.

\[
R(\tilde{p}, w) = \{ \tilde{p}_i \mid |x_{\tilde{p}_i} - x_{\tilde{p}}| \leq w, |y_{\tilde{p}_i} - y_{\tilde{p}}| \leq w \}
\]
\[
C(\tilde{p}, w) = \{ \tilde{p}_i \mid |\tilde{p}_i - \tilde{p}| \leq w \}
\] (2)

\( w \) here is always an non-negative integer limit the window’s scope. \( R(\tilde{p}, w) \) is a set of points that are in a rectangular, actually a \( 2w + 1 \) points width square window (we will call it rectangular to be more general in the rest of this paper). \( C(\tilde{p}, w) \) is a set containing the points around \( \tilde{p} \) within a circular window whose radius is \( w \) points and the center is \( \tilde{p} \). The circular window can be generated by a Bresenham [Bresenham 1977] circle when implementing to be efficient and accurate.

The four alternating dark and bright regions around a corner can be deformed seriously on arbitrary surface. The isotropy against deformation can be achieved if points within a window are iterated by circumambulating the corner from the outer to the inner. The layer is defined in eq. 3 to represent the points being checked in each circumambulating iteration.

\[
L_r(\tilde{p}) = \{ \tilde{p}_1, \ldots, \tilde{p}_i, \ldots, \tilde{p}_n \}, i \in [1, n_r]
\]
for circular window \( \tilde{p}_i \in C(p, r) - C(p, r - 1) \)

\[
\theta_{\tilde{p}_i} < \theta_{\tilde{p}_{i+1}}, i \in [1, n_r - 1]
\] (3)

For circular window layers \( L_r(\tilde{p}) \) in \( C(p, w) \), there are \( w \) layers from \( L_{1r}(\tilde{p}) \) to \( L_{wr}(\tilde{p}) \). \( n_r \) is the points amount in \( L_{wr}(\tilde{p}) \). For a \( L_r(\tilde{p}) \) in \( R(\tilde{p}, w) \), \( n_r \) can be calculated by eq. 4.

\[
n_r = 4(2r + 1) - 4 = 8r
\] (4)

In a window \( R(p, w) \) or \( C(p, w) \), there are \( w \) layers from \( L_{1r}(\tilde{p}) \) to \( L_{wr}(\tilde{p}) \). \( n_r \) is the points amount in \( L_{wr}(\tilde{p}) \). For a \( L_r(\tilde{p}) \) in \( R(\tilde{p}, w) \), \( n_r \) can be calculated by eq. 4.

\[
n_r = 4(2r + 1) - 4 = 8r
\] (4)

For circular window layers \( L_r(\tilde{p}) \) in \( C(p, w) \), the Bresenham circle algorithm will decide the \( n_r \). To be used later, we define the first order derivative in a layer \( L_{wr}(\tilde{p}) \) to be:

\[
I'_w(\tilde{p}) = I(\tilde{p}_{i+1}) - I(\tilde{p}_i)
\] (5)

Note that the coordinate index in \( L_w \) should be \( i \) in \( \tilde{p}_i \), this would make the coordinates in \( L_r \) be 1D. An ordered tuple of points which are sorted by the sequence in which points are iterated when circumambulating the corner. The cumambulating can be performed in the order that the polar coordinates \( \theta_{\tilde{p}_i} \)’s incremental direction (note that within each window, we set the polar coordinates origin \((0, 0)\) be the center of the window). Fig. 1(b) shows the layers in a rectangular window around a corner. A layer \( L_r(\tilde{p}) \) is actually a \( n_r \) points long 1D sequential array representing a 1D image signal (Fig. 2). It transforms the 2D point distribution within a window to 1D, then regions in a layer will be line segments after binarization. That will simplify the detection of regions around a corner when corners are detected by recognizing the four alternating regions feature in our method.

To be used by the ring-morphology defined in Section 3.1, we define some operators for set operations:

Set cardinality:

\[
|A|
\]

Translation:

\[
(A)_p = \{ c \mid c = a + p, a \in A\}
\]

Complementary set:

\[
A^c = \{ w \mid w \notin A \}
\] (6)

3 Detection

Checkerboard pattern detection has two steps, corner detection and checkerboard match. The first step finds all candidate checkerboard corners in the image. The second step eliminates noise points and connects all candidate corners with the help of regions boundaries data to match the checkerboard pattern. The connected corners set, which has the same region corners amount to the checkerboard pattern, will be the checkerboard corner.

3.1 Corner Detection

Region corners are detected by checking whether there are four alternating dark and bright regions around a candidate corner within a window scope. To be efficient and robust, firstly, the image \( \tilde{I} \) is resized to a range \([I_{\text{min}}, I_{\text{max}}]\) without changing the width and height ratio. According to the common camera resolutions from 320 \( \times \) 240 to 2048 \( \times \) 1536, the range is set to be \([300, 2000], [2100, 1600]\). There is no difference between the rectangular window and the circular one to achieve isotropy against deformations when circumambulating the points in layers around the candidate corner. We will use the rectangular one in rest of this paper.

---

Figure 1: A region corner and its layers within a window \( R(\tilde{p}, 4) \), from \( L_1(\tilde{p}) \) to \( L_4(\tilde{p}) \), points are labeled with their layer indices.

Figure 2: 1D form of layers from \( L_1(\tilde{p}) \) to \( L_4(\tilde{p}) \) in Fig. 1(b).
The window size parameter, \( w \in \mathbb{R}(\beta, w) \) is decided by the checkerboard grid size. To detect the four alternating dark and bright regions, the window should not cover more than four regions, which are actually four grids surrounding a corner. So window width \( 2w + 1 \) should be less than two times of the grid width (we only consider checkerboard with square grid) \( g_{\text{width}} \). To be efficient, we decide the two windows of two neighbor corners belonging to a same grid should not intersect too much. So \( 2w + 1 < \frac{g_{\text{width}}}{2} \) and \( w \approx \frac{g_{\text{width}}}{2} \). We also define the a window size range \([WS_{\text{min}}, WS_{\text{max}}]\) to limit the window size for extreme large or small grid. \( g_{\text{width}} \) can be calculated by the amount of checkerboard grids rows \( g_r \) and columns \( g_c \) and the image size \( width_t \) and \( height_t \) according to eq. 7 to be a probable value.

\[
g_{\text{width}} = \min\left(\frac{\text{width}_t}{g_r}, \frac{\text{height}_t}{g_c}\right) \tag{7}\]

Here \( g_r \) and \( g_c \) should be known before recognition. Eq. 8 shows the calculating of the window size parameter \( w \).

\[
w = \frac{w'}{2}, w' \text{ is calculated by eq. 9.} \tag{8}\]

\[
w' = \begin{cases} \text{WS}_{\text{min}} & \text{if } g_{\text{width}} < \text{WS}_{\text{min}} \\
g_{\text{width}} & \text{if } g_{\text{width}} \in [\text{WS}_{\text{min}}, \text{WS}_{\text{max}}] \\
\text{WS}_{\text{max}} & \text{if } g_{\text{width}} > \text{WS}_{\text{max}} \end{cases} \tag{9}\]

The window size parameter does not need high precision, we just ignore the 1 in \( 2w + 1 \) window width in eq. 8. The window size range \([WS_{\text{min}}, WS_{\text{max}}]\) reflects the minimum scope in which a region corner can be identified. The region borders around a corner may be smoothed or extended to blocks rather than border lines to lose sharpness because of noise, low camera capturing quality and complex lighting. To get robustness, the window should be able to cover all four regions to detect them. However, too large window size will cause low performance. We set this range be \([11, 21]\) to tolerate the low quality corners in most common camera cases. The \([WS_{\text{min}}, WS_{\text{max}}]\) with value of \([11, 21]\) limits the window widths from 11 to 21 of \( \mathbb{R}(\beta, w) \). This range may be changed when implementing, however, a points block with over 21 points long width should not be recognized to be a border line even in human’s eyes, so for most applications, the range can be fixed to our values without manual change.

For each layer, a mean value of the layer points is calculated to do thresholding as in eq. 10. Let \( L_\beta(\beta) \) be the binarization result of \( L_r(\beta) \).

\[
L_\beta(\beta) = \{ \overline{\beta} \mid \overline{\beta} = \overline{\beta} \land \overline{\beta} \in L_r(\beta) \text{ and } I_\beta(\beta_i) = 0 \text{ if } I(\beta_i) < t \text{ or } I(\beta_i) > t, t = \frac{\sum_{\beta_i | I(\beta_i)} |I(\beta_i)|}{|I(\beta_i)|} \tag{10}\]

This layer-scope threshold can reduce the negative effects caused by noises to achieve a locally adaptive thresholding result. After thresholding, a noise reduction is performed on the binary 1D image signal to remove fake regions, which are mainly caused by lines or blob points. Fig. 3 shows an example of this fake region caused by a line. These fake regions are actually regions with very short widths. We do the binary morphology [Haralick et al. 1987] operations, opening and closing, both of which are with a same 1D structure elements(SE) \( SE_{\text{cd}} \) to remove noises. The \( SE_{\text{cd}} \)'s length is determined by the layer \( L_\beta(\beta) \)'s length \( n_r \). A proportion \( \kappa \) of the \( SE_{\text{cd}} \) length \( l_{se} \) and the layer \( L_\beta(\beta) \)'s length \( n_r \) is used to calculate \( l_{se} \) (eq. 11).

\[
l_{se} = n_r \cdot \kappa \tag{11}\]

A checkerboard corner should have four regions, so \( \kappa \) should be less than \( \frac{1}{4} \). To be robust, we decide to tolerate the deformed region covering about \( \frac{1}{16} \) layer. To produce more redundant corners to avoid missing corners, we tolerate the minimum region taking up only \( \frac{1}{16} \) layer. So we set \( \kappa \) to \( \frac{1}{16} \). To avoid incorrect noise reduction, layers with small width parameter \( r < 2 \) will not do this operation. What’s more, because the last point and the first point of a layer in 1D form are actually continuous in 2D form, a region may break here because of the 2D to 1D transformation. An example is shown in Fig. 4. These broken regions may be treated as fake regions when noise reducing because of their short widths. To avoid this problem, we modify the binary morphology operations by elongating the layer to repeat it one more time behind its last point to be cycle-compatible, these modified operations are called the ring-morphology as in eq. 13.

Dilation:
\[
A \oplus_{\text{ring}} B = \{ x \mod |A| \mid (B)_{x} \cap (A \cup (A)_{A}) \neq \emptyset \}
\]

Erosion:
\[
A \ominus_{\text{ring}} B = \{ x \mod |A| \mid (B)_{x} \cap (A \cup (A)^{c}) \neq \emptyset \}
\]

Opening:
\[
A \circ_{\text{ring}} B = (A \oplus_{\text{ring}} B) \ominus_{\text{ring}} B
\]

Closing:
\[
A \bullet_{\text{ring}} B = (A \oplus_{\text{ring}} B) \circ_{\text{ring}} B \tag{13}\]

The ring-morphology treats a 1D image as a ring that the head and the end are joined. By performing ring-morphology opening and closing on the binary layers \( L_\beta(\beta) \) after thresholding, we can reduce the fake regions. Then regions count can be calculated by summing absolute value of the first order derivatives \( I_\beta(\beta_i) \) of this binary (only 0 and 1 values) layer. To avoid the similar region breaking problem, we let the first order derivative \( I_\beta(\beta_i) \) at \( \beta_n \) of a layer \( L_\beta(\beta) \) to be:

\[
I(\beta_n) = I_\beta(\beta_1) - I(\beta_n) \tag{14}\]

In checkerboard pattern, one layer should have 4 regions. If all layers of a candidate corner have 4 regions, this candidate should be a corner. However, to get high robustness, we define an acceptance threshold value \( \alpha \) to allow some noises. If there are \( \alpha \) or more portions of layers containing 4 regions, the candidate corner
is accepted to be a checkerboard corner. Too large \( \alpha \) will cause too many noise corners while too small \( \alpha \) also reduces the robustness. This acceptance threshold value can be determined by the acceptance degree of human’s eyes and image noise degree. We set it to be 0.6 to tolerate 40% noise layers in a window for common camera images.

Till now the corner detection result including the region corners is produced. Let the result be \( \mathbb{N} \). Moreover, the regions boundaries positions of a corner are recorded for the later checkerboard match step. Since there are more than one layers within a corner’s surrounding window, only the most outer layer is recorded. We represent the region boundary of a corner \( \vec{p} \) within its window \( \mathbb{R}(\vec{p}, w) \) to be (note that the derivative \( I'_p(\vec{p}) \) is on the thresholded binary layer \( B(\vec{p}) \)):

\[
\mathbb{B}_p = \{ \vec{p}_i | |I'_p(\vec{p}_i)| = 1, \forall \vec{p}_i \in \mathbb{L}^B(\vec{p}) \}
\]

\[
w_{\text{max}} = \max\{w_j | w_j \in [1,w], \sum_{\vec{p}_k \in \mathbb{L}_j(\vec{p})} |I'_p(\vec{p}_k)| = 4\} \tag{15}
\]

### 3.2 Checkerboard Match

#### 3.2.1 Decide Checkerboard Corners Set

The corner detection step can produce noise corners and redundant corners because of the acceptance threshold value and image noises. To reduce those noise, we perform a window neighbors checking on the corner detection result. To remove the redundant corners, firstly we cluster the result points, then in each cluster, the mass center or the point with minimum distances sum to other points of the same cluster is calculated to be the new checkerboard corner, other result corners are removed. After noise and redundance reduction, corners are connected by their region boundary data to match the checkerboard pattern.

When doing noise reduction on the corner detection result \( \mathbb{N} \), each corner \( \vec{c}' \)'s neighbors within a window \( \mathbb{R}(\vec{c}', w) \) is checked to see whether there are enough corners in the window. If there are enough more than a given \( \tau \) corners, all corners within the window are reserved, otherwise the checked corner is removed. This operation is similar to binary erosion, we also need to a \((2w_{se} + 1) \times (2w_{se} + 1)\) structure element \( SE_{cr} \) to perform the neighbors checking. By defining \( SE_{cr} \) in eq. 16, the refined result \( \mathbb{N}' \) can be calculated in eq. 17, in which we also use the translation operation and set cardinality defined in eq. 6.

\[
SE_{cr} = \{ \vec{p}_{ij} | i \in [1,2w_{se} + 1], j \in [1,2w_{se} + 1], x_{\vec{p}_{ij}} = i - w_{se} - 1, y_{\vec{p}_{ij}} = j - w_{se} - 1 \} \tag{16}
\]

\[
\mathbb{N}' = \{ \vec{p} | |(SE_{cr})\vec{p} \cap \mathbb{N}| \geq \tau, \forall \vec{p} \in \mathbb{N} \} \tag{17}
\]

The \( \tau \) to reduce noise in eq. 17 should be no larger than \( w_{se} \). We define \( w_{se} \) to be 2 to check a \( 5 \times 5 \) neighbors patch and \( \tau \) to be 2 to eliminate the isolated noise corner, which is the only one corner within that neighbors patch when \( \tau \) is 2.

Redundant corners are reduced by clustering. Corners result \( \mathbb{N}' \) after noise reduction is clustered according to their distances between each other. The result \( \mathbb{N}' \) is treated as a node set and each corner in \( \mathbb{N}' \) is a node. If two corners have a distance no larger than \( SE_{cr} \)'s width \( 2w_{se} + 1 \), there is an edge connecting them. By defining nodes and edges, we get an undirected graph \( G' \) (eq. 18).

\[
G' = (\mathbb{N}', \mathbb{E}')
\]

\[
\mathbb{E}' = \{ (\vec{p}_i, \vec{p}_j) | |\vec{p}_i - \vec{p}_j|| \leq 2w_{se} + 1, \forall \vec{p}_i, \vec{p}_j \in \mathbb{N}' \} \tag{18}
\]

Each connected component \( G'_c (\mathbb{N}', \mathbb{E}') \) in \( G' \) is a cluster. For each cluster \( G'_c \), the mass center or the point \( \vec{p}_c \) with minimum distances sum to other points of the same cluster is calculated and made to be the new corner to replace others in the cluster. \( \vec{p}_c \) is calculated by eq. 19. There may be more than one \( \vec{p}_c \), just pick a random one when implementing.

\[
\vec{p}_c : \vec{p}_c \in \mathbb{N}' \text{ and } \sum_{\vec{p}_j \in \mathbb{N}'} |\vec{p}_c - \vec{p}_j| = \min \{ \sum_{\vec{p}_j \in \mathbb{N}'} |\vec{p}_c - \vec{p}_j| | \forall \vec{p}_j \in \mathbb{N}' \} \tag{19}
\]

Because mass centers of some clusters may be not in \( \mathbb{N}' \), the region boundary positions of those mass centers are not recorded in corner detection. This problem can be solved by calculating the mean region boundary positions of corners in a cluster. \( \vec{p}_c \) does not have this problem although the mass center is always more accurate in position than \( \vec{p}_c \) to represent the cluster’s position. So when implementing, \( \vec{p}_c \) will be an efficient selection. We let \( \mathbb{N}'' \) be the corners result after redundant corners reduction.

After noise and redundance reduction, the region boundary positions of each found corner are used to calculate the connectedness of corners. We also use the graph theory to assist this process. Each corner in \( \mathbb{N}'' \) is a node. An edge connecting two corners will exist if those two corners are on a same region boundary line (fig. 5). An edge actually connects two checkerboard corners sharing the same boundary of one checkerboard grid. These two corners are neighbor corners of a quad grid. The edge set is defined by eq. 20.

\[
\mathbb{E}'' = \{ (\vec{p}_i, \vec{p}_j) | \exists \vec{p}_h \in \mathbb{B}_{\vec{p}_i}, \vec{p}_h \in \mathbb{B}_{\vec{p}_j} : \\
(\vec{p}_h - \vec{p}_i) \cdot (\vec{p}_j - \vec{p}_i) \leq \vec{p}_h - \vec{p}_i \parallel \vec{p}_j - \vec{p}_i \parallel \in [-1, 1] \} \tag{20}
\]

\( \epsilon \) here should be very close to \(-1\). \([-1, 1]\) defines an acceptance range to tolerate the inaccurate region boundaries. The inaccuracy is mainly caused by the serious deformation within a checkerboard grid. If two corners \( \vec{p}_i \) and \( \vec{p}_j \) are on a same boundary of a checkerboard grid, \([-1, 1]\) defines the cosine value(calculated by \( \frac{(\vec{p}_h - \vec{p}_i) \cdot (\vec{p}_j - \vec{p}_i)}{||\vec{p}_h - \vec{p}_i|| ||\vec{p}_j - \vec{p}_i||} \)) range of the angle between two region boundaries vectors: \( \vec{p}_h - \vec{p}_i \) and \( \vec{p}_j - \vec{p}_i \), here \( \vec{p}_h \) and \( \vec{p}_i \) are on the same grid boundary on which \( \vec{p}_i \) and \( \vec{p}_j \) locate. eq. 20 ignores the relative position of the two corners when deciding edges, in our experiment results in Section 4, this ignorance is tolerable. Considering other cases that can tolerate the non-planar grid and to be robust under serious deformation, we decide \( \epsilon \) to be \(-0.8\) to tolerate a \( \pm \arccos(-0.8) \approx \pm 36^\circ \) bending deformation of straight boundaries within a grid.

Now the graph \( G'' = (\mathbb{N}'', \mathbb{E}'') \) is produced. With the assistance of checkerboard grids rows \( g_r \) and columns \( g_c \), we can find the checkerboard pattern corners by finding the connected component \( G''_c \) that has exact \( (g_r - 1) \times (g_c - 1) \) nodes. If there is the only one \( G''_c \) having \( (g_r - 1) \times (g_c - 1) \) nodes, it is the set that contains all the right checkerboard corners.

#### 3.2.2 Corners Match

In \( \mathbb{N}'' \), corners are in three classes that have different numbers of neighbors. Fig. 5 shows the corners with 2, 3, 4 neighbors respectively. In \( \mathbb{N}'' \), there are four corners that have 2 neighbors, they are the most left-top, most right-top, most left-bottom and most right-bottom corners. We limit the checkerboard pattern rotation within \(+45^\circ\). Then those four corners can be identified and matched first. Corners with 3 neighbors lie in the four most outside boundaries of checkerboard pattern. After the identifying of the four 2-neighbor-corners, we can identify and match the 3-neighbor-corners one by one.
one along the most outside boundaries. Now the most outside corners are matched. Then those corners and their edges are removed from \( G'' \) and there will be three classes of corners with different neighbors again. We can repeat the matching method above the identify all corners from the most outside to the center. Till now, the recognition is completed and the corner points correspondence is built, which can be used to calculate the pixel mapping between camera view and projector view when doing geometry registration.

4 Result and Comparison

4.1 Recognition Result and Comparison

We compare our method with the popular checkerboard detection method, FindChessboardCorner function in OpenCV, to evaluate our recognition of the checkerboard pattern on nonplanar surface and bad illumination condition.

OpenCV’s cvFindChessboardCorners does thresholding on the global image to transform the dark and bright grids into black and white. This thresholding can always fail even with adaptive method under complex illumination. The results in fig. 6(a) and 6(b) show the failed detection of region corners by OpenCV.

Fig. 6(c) and 6(d) show that our method can find all checkerboard region corners and match them within the checkerboard pattern successfully while OpenCV fails. Besides the comparison, we also evaluate our method with plenty of other deformed and bad illuminated checkerboard pattern images. Those images, which are projected checkerboard on arbitrary surface or deformed surface with checkerboard texture, are captured by general low-cost cameras with an highest interpolated resolution up to 1280 × 960. Fig. 7 and 9 show some representative results of our experiment on arbitrary surface.

4.2 Geometry Registration Experiments in ProCams

To evaluate our method in geometry registration of ProCams, we use one projector and one camera to build a simple projector-camera system. To simplify the experiment, we set the viewer and the calibration camera to be the same position. Then we can ignore the homography between camera and viewer. Our geometry registration looks nonplanar display surface to be quad patches that approximate planar. This will model the display surface in the 3D surface modeling way. So planar geometry registration with one homography can be adopted within each quad patch. Fig. 8 shows the checkerboard quad patch and its quad mesh. Let \( Q^c_i \) be the patch in camera view image, \( Q^p_i \) be its corresponding quad in projector view image. Then available display surface is \( \delta^c = \{ Q^c_i \mid i = 1 \ldots (g_r - 2) \cdot (g_c - 2) \} \) and available projector image area is \( \delta^p = \{ Q^p_i \mid i = 1 \ldots (g_r - 2) \cdot (g_c - 2) \} \). Note that the available display surface area is smaller than the projecting scope. With the help of recognized checkerboard corners in camera view and their corresponding corners in projector view, for each quad in

Figure 5: Three classes of corners (marked by yellow circles) with different number of neighbors (marked by red circles).

Figure 6: Checkerboard pattern finding result comparison with OpenCV.

Figure 7: Some of our checkerboard finding result (Enlarge images to see detail)
checkerboard we can calculate a homography as perspective transform matrix between camera and projector view. Let $H_{p2c}$ be the homography transforming points from $Q_p$ to $Q_c$. To warp an image $I$, first we transform it into $S_p$ without changing the height and width ratio. Then each point $p$ in $S_p$ can be transformed to its corresponding $q$ in $S_c$ according to eq. 21 to get warped image $I_w$.

$$q = H_{p2c} \cdot p, p \in Q_p, q \in Q_c$$  \hspace{1cm} (21)

To be more efficient, mapping between each corresponding point pair can be pre-computed, a pixel mapping look-up table will be more convenient.

Fig. 9 and Fig. 10 show our experiment results. Fig. 9 shows the recognition result of projected calibration checkerboard pattern. Fig. 10 shows some test images and their warped results and the final registration results. Here we can achieve human-eyes acceptable correctness in geometry registration by projecting only one calibration checkerboard pattern and warping image with a set of quad patch homographies. Multiple projectors display can also be achieved with our method. In the multiple projectors case, each projector needs one calibration checkerboard image, however, that will be also more efficient than contemporary registration methods that each projector will need more than one calibration image.

5 Conclusion and Future Works

A robust checkerboard pattern recognition method is described in this paper to help geometry registration in ProCams. This method addresses the problem of detecting checkerboard pattern and recognizing the corresponding corners under nonplanar deformation without encoded points or current structure light patterns that require multiple calibration pattern images. Only one checkerboard calibration pattern image is needed to do geometry registration in ProCams to calculate homographies between quad patch pairs from camera and projector views. In our recognition process, the checkerboard internal corners surrounded by four alternating dark and bright regions and their boundaries are detected robustly. The experiment results in Section 4 show that our method can deal with most common checkerboard images captured by general cameras with serious deformation on arbitrary surface. We also evaluate our method to do geometry registration in ProCams and the result (in Section 4.2) shows that it is effective and efficient.

Our next step is to accelerate this method. To achieve the most robustness, we detect the region corners and match checkerboard pattern by recognizing and using almost all their features including surrounding regions amount, region boundaries positions and neighbor corners’ boundary vectors angles. Both intensity and geometry information are used. This can ensure the robustness but costs much time. In future, we will focus on the speed. Some features may be simplified to save time without losing robustness. The window-based corner detection may perform window-operation only on the pre-detected corners by other much faster methods rather than every point.

Acknowledgements

This work is sponsored by 863 National High Technology R&D Program of China (No.2006AA01Z307).

References


Figure 10: Geometry registration result (Enlarge images to see detail.)


INTEL. OpenCV computer vision library.


