The Calibration of Kinect Camera Based on ARtoolkit

Cheng HAN¹, Li-juan BAI¹*, Bao-xing BAI¹, Chao ZHANG¹ and Yun-bo YANG²

¹ Computer Science and Technology, Changchun University of Science and Technology, Changchun, China
² Digital Media Technology, Yanbian University of Technology, Yanji, China

Keywords: Kinect, ARtoolkit, Camera calibration.

Abstract. To deal with mutual occlusion of the virtual and real objects in the augmented reality scene, their depth values should be compared by using ARtoolkit and Kinect application programming interfaces (APIs). To improve the accuracy of the experiment, it has given up the existing default camera parameters file in ARtoolkit, and calibrated the Kinect camera to generate its own parameters data file. And different from the existing Kinect camera calibration methods, it abandoned the method of using the open NI SDK or extra high definition color camera as an intermediate bridge. Experimental results show this method has achieved a high accuracy and the validity by verifying in ARtoolkit.

Introduction

As Kinect can provide the depth map, human skeletal information and color information of the scene, Kinect has a wide development in many areas, such as posture estimation [1], human detection [2], three-dimensional reconstruction [3], and augmented reality (AR).

In the field of AR, we can use Kinect to obtain the depth information of real object and determine the mutual occlusion of the virtual and real object in the fusion scene. A number of methods have been proposed for achieving the correct mutual occlusion in AR. For example, De Gaspari T [4] obtains the relative pose between the Kinect infrared (IR) camera and an extra high-resolution RGB camera by the camera calibration. And then his overlays the depth map on the RGB image with alignment to get the depth information of color image and achieves the mutual occlusion by using the open GL depth testing technology. In addition, the calibration method of Daniel H C [5] is used to calibrate the cameras. This method has calibrated an extra camera and two Kinect cameras with a planar surface and presents a new depth distortion model for the depth sensor.

For the calibration of the Kinect camera, there are many methods have been proposed now. Different from other methods only considering the IR camera parameters, the calibration method considers both intrinsic parameters of the IR camera and IR projector [6]. Besides, it also proposes a new depth error model of the Kinect. Nakazawa M [7] calibrates the Kinect in the case of using multiple Kinects and has used an extra RGB camera to obtain the correspondences between color images from the Kinects. And he optimizes the estimated parameters by minimizing both the errors of correspondences between color images and range data of planar regions. Jin B [8] calibrates the depth camera by using a set of cuboids with known sizes, instead of the check board pattern. Therefore, the length, width and height of cuboids are calculated more directly and robustly in the low-precise depth map. Raposo C [9] has achieved the calibration accuracy similar to Herrera’s calibration method while using less input frames. He uses a series of input frames and several iterative minimization steps to obtain the high accuracy calibration results.

In our case, we use Kinect in ARtoolkit to realize the correct mutual occlusion in the scene. To this end, the depth values of the virtual and real object, which are corresponding to the same pixel in the color image, should be compared. In addition, ARtoolkit has a default camera parameter file including the default camera properties. In order to improve the accuracy of the registration of the virtual object, we should abandon the existing default camera parameters file and use the Kinect’s own camera parameters. But the parameters calibrated by other methods can’t be used in ARtoolkit,
since the data file has its own parameter. Thus a separate parameter file needs to be generated by calibrating the Kinect camera in ARtoolkit calibration program.

**The Calibration of Kinect Camera**

According to the Kinect SDK function, the color and IR camera are considered to have the same intrinsic parameters [11], so we just need to calibrate the Kinect color camera. In this section, ARtoolkit’s two-step calibration method is used to get the camera intrinsic parameters. Meanwhile, the extrinsic parameters can be also known in real time. What’s more, the calibration method doesn't use the open NI or extra camera, which is different from the other calibration methods of the kinect camera.

**Pre-experimental Preparation**

The camera or video capture device, such as USB camera, are usually started by default In ARtoolkit. However, the devices only support video for windows (VFW) and windows driver model (WDM) drivers. There are two cameras in Kinect, color camera and IR camera. The IR camera is usually applied to get the depth data with the IR projector. They are both complementary metal oxide semiconductors (CMOS) and don’t support VFW or WDM drivers. So when Kinect is used in ARtoolkit, one interface should be written to get the color and depth image of the scene.

The coordinates system in ARtoolkit is shown as Fig. 1: the camera screen coordinates \((x_c, y_c)\), the camera coordinates \((X_c, Y_c, Z_c)\) and the maker coordinates \((X_m, Y_m, Z_m)\). Meanwhile, the relationships among the three coordinates can be represented as eq. 1

\[
\begin{bmatrix}
    h_x^c \\
    h_y^c \\
    h
\end{bmatrix}
= P
\begin{bmatrix}
    X^c \\
    Y^c \\
    Z^c
\end{bmatrix}
= P \cdot T_{cm}
\begin{bmatrix}
    X^m \\
    Y^m \\
    Z^m
\end{bmatrix}
= P
\begin{bmatrix}
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    R_{3\times3} \\
    T_{3\times1}
\end{bmatrix}
\begin{bmatrix}
    X^m \\
    Y^m \\
    Z^m
\end{bmatrix}
\]

(1)

![Figure 1. The coordinates system in ARtoolkit.](image)

Where \(f\) is the focal length of the camera, \(s_x\) and \(s_y\) is the scale factor of the x and y axis direction, \(x_0\) and \(y_0\) is the coordinates of the intersection of the camera’s optical axis and the camera screen. The intrinsic parameters in ARtoolkit include: xsizen and ysize is the width and height of the image in pixels, the dist_factor array contains the \(x, y\) coordinate of the image center point \((x_0, y_0)\); the distortion factor dist and the image scale factor \(S\) (aspect ratio), \(3\times4\)
perspective matrix $P$. And the extrinsic parameter $T_{cm}$ is the translation and rotation transformation from the marker coordinates to the camera coordinates.

**The Calibration of Lens Distortion**

This section describes the first step of the calibration to get the $x_{size}, y_{size}$ and $\text{dist\_factor}$ array values. A pattern of 6x4 marker dots spaced equally apart is used in this process. First of all, we need to initialize the distance between each dot, in millimeters. The distance should be 40 mm correctly, however, it will be 39 mm in our experiment. All the dots in the camera screen coordinates can be detected in the grabbed image and many pairs of the marker 3D coordinates $(X_m, Y_m, Z_m)$ and the camera screen coordinates $(x_c, y_c)$ are used for getting the values.

There are three situations when we display the image from the Kinect color camera. The first one is showing a real-time video on the window. When we can get an image that contains each point, freeze the image and proceed to the second situation.

The next one is showing an image and drawing bounding boxes over each dot in the image. It starts with the dot in the upper left corner of the image. After the rectangles drawn, the red-cross should be drawn at the rectangle’s center. And when 5-10 images from different positions are captured and drawn, the $\text{dist\_factor}$ will be calculated as the following steps:

1. The initialized values of the $x_0, y_0, \text{dis}, S$ are $x_{size}/2$, $y_{size}/2$, 0.0, 1.0.
2. Calculate the error of the distortion calibration iteratively according to the $\text{dist\_factor}$ array initial values. Where $x_0$ and $y_0$ add 5 in each iteration calculation respectively, and $x_0, y_0$ meet the condition: $x_0 \in [x_{size}/2 - 50, x_{size}/2 + 50], y_0 \in [y_{size}/2 - 50, y_{size}/2 + 50]$. Save the current values of $x_0$ and $y_0$ when we get the minimum error value.
3. Repeat the step 2, and which the different is the range of the $x_0$ and $y_0$. At this time, the values of the $x_1, y_1$ are the value we got in the step 2, which has the minimum error. In addition, the different condition: $x_0 \in [x_1 - 5, x_1 + 5], y_0 \in [y_1 - 5, y_1 + 5]$, and they add 0.5 in each iteration calculation respectively. As a result, we can get the values of $x_0, y_0, \text{dis}$ with the smallest error.
4. Set $sf = 100.0$ as the maximum scale factor value and the coordinate of a point as $(0.0, y_0)$, and convert it to ideal coordinate, which means that the distortion of the camera is compensated. The ideal coordinate is named $i_x, i_y$, and the $sf$ can be calculated by using the formula:
   \[ sf = (x_0 - i_x) / x_0 \]  
   (2)
5. And repeat the above step with the other points: $(x_{size}, y_0), (x_0, 0.0), (x_0, y_{size})$, and the respectively formulas are:

   \[
   \begin{align*}
   sf &= (i_x - x_0)/(x_{size} - x_0) \\
   sf &= (y_0 - i_y) / y_0 \\
   sf &= (i_y - y_0)/(y_{size} - y_0)
   \end{align*}
   \]  
   (3)
6. At the last, set the smallest one of the four $sf$ values to $S$.

The last situation is drawing the red lines to check if these parameters are correct (see Fig. 2). If they are correct, these lines will pass through the centers of each dot of every image.

Figure 2. The image of checking the calibration results.
The Calibration of Perspective Matrix

The second step is similar with the first one and the coordinates of the center point, distortion focal length and size adjustment factor will be used. The grid pattern used in this step contains of 7 horizontal lines and 9 vertical lines. The distance between each line is also 40 mm and we need move the pattern 100 mm away from the Kinect camera every time. The details are described as follows:

1. Similar with the situations of the first step of the calibration, it has two modes. When the mode is 0, the window will display a live video until we found all the lines are in view and the grid is as large as possible.
2. After we grab the image, the mode will be 1 and a white line will appear in the image. Move and draw it by calling the ARtoolkit functions when the white line is aligned with the grid line and it will be blue. Repeat the operation for the 16 lines.
3. Once all the lines have been placed, we should calculate the perspective matrix by calling the function.

Firstly, the \( \text{inter\_coord}[k][j][i + \text{hno}] \) means the world coordinates of the Intersection point between the \( j \)th vertical line and the \( i \)th horizontal line in the \( k \)th image. Its coordinates should be calculated with the formula:

\[
\begin{align*}
\text{inter\_coord}[k][j][i + \text{hno}][0] &= \text{dist1} \times i \\
\text{inter\_coord}[k][j][i + \text{hno}][1] &= \text{dist1} \times j \\
\text{inter\_coord}[k][j][i + \text{hno}][2] &= \text{dist1} \times k
\end{align*}
\]  

Where \( \text{hno} = 7 \) is the number of the horizontal lines, \( \text{dist1} = 40 \) is the distance among lines, and \( \text{dist2} = 100 \) is the distance to move the pattern.

Meanwhile, the line array contains the coordinates of the starting point and the ending point, named as \( \text{sx, sy, ex, ey} \). This process will get the screen coordinates of the intersection point of the \( j \)th vertical line and the \( i \)th horizontal line in the \( k \)th image. The formula is expressed as follows:

\[
\begin{align*}
a &= \text{sy}_1 - \text{ey}_1; b = \text{ex}_1 - \text{sx}_1; c = \text{sx}_1 \times a + \text{sy}_1 \times b \\
d &= \text{sy}_2 - \text{ey}_2; e = \text{ex}_2 - \text{sx}_2; c = \text{sx}_2 \times d + \text{sy}_2 \times e \\
g &= a \times e - b \times d; \text{screen\_coord}[0] = (c \times e - b \times f) / g \\
&= (a \times f - c \times d) / g
\end{align*}
\]  

Finally, we can calculate automatically the camera parameters in the function and save them in data file.

Calculation of the Transformation Matrix

In ARtoolkit, we use the center of the market as the origin of the marker coordinates. Thus, the transformation matrix \( T_{cm} \) from the marker coordinates to the camera coordinates can be considered as the position of the camera.

When we detect the marker, the rectangular area is template matched with the pattern. During the process, the information of the rectangular area will be stored after contour detection, which is in idea screen coordinate without the distortion. The information contains that the number of pixels in this region\_area, the marker identified number\_id, confidence value to be a marker \_cf, the center of the region\_pos, the parameters of these four line segments\_line, the coordinates of the four points of the region\_vertex and the direction of the region\_dir. The \_dir can be 0, 1, 2 or 3, and it can tell about the line order of the detected marker (which line is the first one).

According to the value of line, the relationship of the two parallel sides of the marker can be represented as the following:
\[ a_1x+b_1y+c_1=0, \quad a_2x+b_2y+c_2=0 \]  
(6)

Meanwhile, the equations of getting the screen coordinates \((x_c, y_c)\) with the perspective projection matrix \(P\) are the following:

\[
P = \begin{bmatrix}
    s_x f & 0 & x_0 & 0 \\
    0 & s_y f & y_0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}, \quad \begin{bmatrix}
    h x_c \\
    h y_c \\
    h \\
    1
\end{bmatrix} = \begin{bmatrix}
    X_c \\
    Y_c \\
    Z_c \\
    1
\end{bmatrix}
\]  
(7)

We can obtain the rotation matrix \(R_{3x3}\) according to eq. 8.

\[
\begin{align*}
    (a_1 s_x f X_c + b_1 s_y f Y_c + (a_1 x_0 + b_1 y_0 + c_1)) Z_c &= 0 \\
    (a_2 s_x f X_c + b_2 s_y f Y_c + (a_2 x_0 + b_2 y_0 + c_2)) Z_c &= 0 
\end{align*}
\]  
(8)

At the same time, the screen coordinates and marker coordinates of four vertexes in the marker detected are named as \((x_c, y_c)\) and \((X_m, Y_m, Z_m)\). So we can calculate the translation matrix \(T_{3x1}\) with the known values of \(P, R_{3x3}\). Thus, we have the initial transformation matrix \(T_{cm}\) with error. So we need to refine iteratively by using the nonlinear least square method. The optimal value of \(T_{cm}\) is obtained by finding the minimum of the quadratic sum of the distances between the vertices of the detected rectangular region and the transformed coordinates with \(T_{cm}\). And the formula of the optimization is the following:

\[
\text{err}^2 = \frac{1}{4} \sum_{i=0}^{3} ((x_{ci} - x_{ci'})^2 + (y_{ci} - y_{ci'})^2) \rightarrow \text{min}
\]  
(9)

**Experimental Results**

In this experiment, we calibrate both 5 images in the first and second step of this method, which is less than others’. In addition, we calibrate the Kinect color camera with the check-board calibration pattern in MATLAB. To evaluate our calibration results, we compare the errors among our method, Herrera I’s method, Barreto J P’s method [11] and MATLAB. Table 1 clearly shows that our method produces the smaller error result with less calibration images.

| Barreto P’s method | Herrera I’s method | Our method | MATLAB |
|--------------------|--------------------|------------|--------|
| No DC | 0.495 | 0.743 | 0.18 | 0.32 |
| DC | 0.369 | 0.602 | | |

**The Verification of the Camera Parameters**

After calibrating the Kinect camera, we save the parameters into a parameter data file, which will be read when the ARtoolkit program starts and opens the video device. Fig. 3 shows the difference of the accuracy of the registration of the virtual object between using different parameter files. And it proves the high accuracy of our calibration method.
Conclusion

In this paper, we propose a method of calibrating Kinect’s camera with using ARtoolkit. This two-step calibration method can obtain the high calibration accuracy. Moreover, we apply the calibration results in ARtoolkit to improve the registration of the virtual object when we use Kinect’s color camera to capture the real scene. Then, the error is minimized to 0.18 in the process of the calibration.

As the future work, we will try to apply the calibration results to realize the mutual occlusion in the AR scene. Because we can convert the 3D coordinates of the virtual object into the 2D coordinates in the color image by using the camera parameters. And we will try to compare the depth values of the virtual and real object which have the same coordinates in the color image.

Acknowledgment

This research was financially supported by the National Natural Science Foundation of China (No. 61602058), Natural Science Foundation of Jilin Province (No. 20150101015JC), Jilin province major science and technology bidding project (20170203004GX) and Jilin province major science and technology bidding project (20170203003GX).

References

[1] Obdrzalek S, Kurillo G, Ofli F, et al. Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population [J]. 2012.

[2] Xia L, Chen C C, Aggarwal J K. Human detection using depth information by Kinect[C]/CVPR 2011 Workshops. IEEE, 2011:15-22.

[3] Izadi S, Kim D, Hilliges O, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera[C]/ACM Symposium on User Interface Software and Technology. ACM, 2011:559-568.

[4] De Gaspari T, Sementille A C, and Vielmas D Z, et al. Arstudio: a virtual studio system with augmented reality features [J]. 2014:17-25.

[5] Daniel H C, Kannala J, Heikkilä J. Joint Depth and Color Camera Calibration with Distortion Correction [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2012, 34(10):2058-2064.

[6] Yamazoe H, Habe H, Mitsugami I, et al. Easy depth sensor calibration [J]. 2012:465-468.

[7] Nakazawa M, Mitsugami I, Habe H, et al. Calibration of multiple Kinects with little overlap regions [J]. IEEE Transactions on Electrical & Electronic Engineering, 2015, 10(S1):S108-S115.
[8] Jin B, Lei H, Geng W. Accurate Intrinsic Calibration of Depth Camera with Cuboids [M]/Computer Vision—ECCV 2014. Springer International Publishing, 2014:788-803.

[9] Barreto J P, Nunes U. Fast and Accurate Calibration of a Kinect Sensor[C]/International Conference on 3dtv-Conference. IEEE, 2013:342-349.