A flexible calibration method of multiple Kinects for 3D human reconstruction.

Palasek, P; Yang, H; Xu, Z; Hajimirza, N; Izquierdo, E; Patras, I

A FLEXIBLE CALIBRATION METHOD OF MULTIPLE KINECTS FOR 3D HUMAN RECONSTRUCTION

Petar Palasek, Heng Yang*, Zongyi Xu, Navid Hajimirza, Ebroul Izquierdo, Ioannis Patras*

Queen Mary University of London
{p.palasek, heng.yang, zongyi.xu, n.hajimirza, ebroul.ezquierdo, i.patras}@qmul.ac.uk

ABSTRACT

In this paper, we present a simple yet effective calibration method for multiple Kinects, i.e. a method that finds the relative position of RGB-depth cameras, as opposed to conventional methods that find the relative position of RGB cameras. We first find the mapping function between the RGB camera and the depth camera mounted on one Kinect. With such a mapping function, we propose a scheme that is able to estimate the 3D coordinates of the extracted corners from a standard calibration chessboard. To this end, we are able to build the 3D correspondences between two Kinects directly. This simplifies the calibration to a simple Least-Square Minimization problem with very stable solution. Furthermore, by using two mirrored chessboard images on a thin board, we are able to calibrate two Kinects facing each other, something that is intractable using traditional calibration methods. We demonstrate our proposed method with real data and show very accurate calibration results, namely less than 7mm reconstruction error for objects at a distance of 1.5m, using around 7 frames for calibration.

Index Terms— Kinect, calibration

1. INTRODUCTION

The rapid development of social networks and communication technologies has moved an individual's social life to a whole new level. People communicate with their family and friends instantly using social networks such as Facebook, exchanging messages even if they live far away from each other. However, this kind of on-line interaction is just a substitute for real human interactions. Although we can see and/or talk with each other, we cannot feel each other because we are not co-located in the same environment. Enabling people at different locations to be able to collaborate and interact in a general environment allows for a lot of different applications, such as distant education or gaming. REVERIE [1] project aims to build such an immersive online environment where people can conduct realistic interpersonal communication and interaction. REVERIE envisages an ambient, content-centric Internet-based environment, highly flexible and secure, where people can work, meet, participate in live events, socialize and share experiences, as they do in real life, but without time, space and affordability limitations. The reconstruction of people's 3D virtual counterparts in order to provide realistic representations of people from different geographical locations is one of the core parts in REVERIE project.

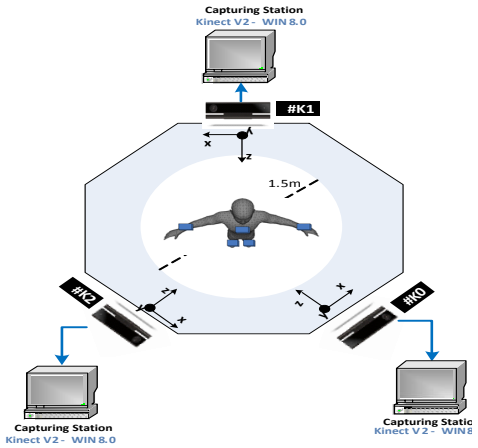


Fig. 1: Top view of the Kinects' spatial arrangement.

The calibration problem has been studied in different ways and many methods have been proposed. The most widely used extrinsic calibration method is proposed by Zhang et al. [2]. It uses a chessboard pattern captured at several 2D views from different angles to estimate the relative positions of two cameras, which is known as stereo calibration. It is mainly used for ordinary RGB cameras as it relies on accurate extraction of the corners of the chessboard. [3] also studied the problem of multiple Kinects calibration but it also requires non-linear optimization. [4] also studied this problem.

There are some common limitations of current methods. First, since the 2D chessboard only provides 2D information, it usually requires non-linear optimization for 3D pose estimation. Therefore, a good calibration usually requires dozens of frames. Second, in our 3D reconstruction setup shown in Figure 2, the two cameras are facing each other, so each of

* indicates authors contributed equally. This work was supported by the European Integrating Project Reverie under Grant FP-287723.

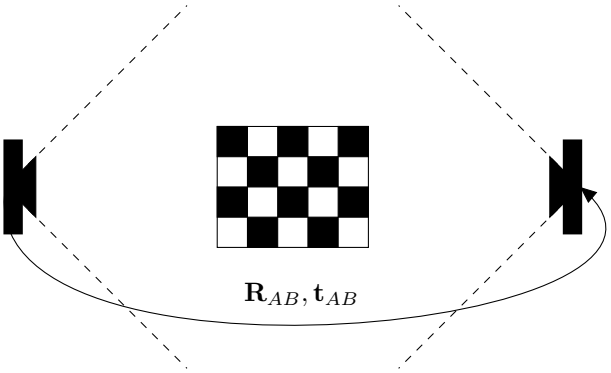


Fig. 2: Top view of two Kinects' spatial arrangement.

the cameras sees only the front or the back side of the subject. In this case, the method of Zhang Z. [2] cannot be used to estimate the extrinsic parameters of these two cameras at the same time due to the invisibility of the common side of the 2D board. Third, in order to achieve reconstruction with Kinect sensors, the 3D point clouds from each separate view need to be registered. The conventional method is designed for calibration of RGB cameras, thus people use the relative position between two RGB cameras to approximate that of the depth camera, which will result in inaccuracy. In this paper we aim to solve the above mentioned problems. We directly find the 3D correspondences between different views based on the corner extraction of the 2D chessboard. We demonstrate our proposed method in a real calibration experiment and obtain accurate calibration using only less than 10 frames.

2. METHOD

In order to calculate the relative position of two arbitrary Kinects, we first need to get a set of point correspondences. Traditional methods of stereo calibration like [2] use 2D calibration patterns [5] where the calculation is based on the planar constraints and homography transformation. In this work we propose a 3D correspondence-based method in which we calibrate the 2D RGB image and the depth image of each Kinect. Given the intrinsic parameters of both the RGB and depth cameras, we can calibrate their relative position and map each 3D point into the RGB image. Thus for each pixel on the RGB image, we can approximately estimate the depth information and further infer the 3D coordinates. Similar to the traditional method, we capture a few image frames of the calibration pattern and extract the locations of the corresponding corners as shown in Fig. 4. Instead of using the 2D correspondences, we infer the 3D coordinates of each point. This allows us to create a set of pairs of 3D correspondences for each Kinect depth camera system.

We describe how we find the 3D correpondences in Section 2.1, the pose calculation in Section 2.2 and the 3D transformation in 2.3.

2.1. 3D correspondences

Since we can easily extract chessboard corners from RGB image (12 corners per image in our case), we assume we have calibrated intrinsic parameters of the RGB and depth cameras using traditional methods [2]. Here we assume both cameras have no distortion. Thus we only need to find the focus length and optical center. The dense mapping procedure is shown in Algorithm 1.

Algorithm 1 Mapping the color pixels to the depth data.

```
procedure GET_MAPPING(depth_image, color_image)
    center_x ← depth_image.width / 2
    center_y ← depth_image.height / 2
    grid ← mat[color_image.height][color_image.width]
    for each depth_pixel ∈ depth_image do
        c_point ← get_corr_color_point(depth_pixel)
        if c_point.color = not available then
            continue
        end if
        z ← depth_pixel.value
        x ← (c_point.x - center_x) * z / flx
        y ← (c_point.y - center_y) * z / fly
        grid[c_point.y][c_point.x].color = c_point.color
        grid[c_point.y][c_point.x].xyz = {x, y, z}
    end for
    return grid
end procedure
```

Once we get this dense mapping, we can find the 3D coordinates of the extracted corners from the RGB images using algorithm 2.

Algorithm 2 Getting world coordinates of 2D corner points.

```
procedure 3D_CORNERS(depth_image, color_image)
    grid ← get_mapping(depth_image, color_image)
    2D_corners ← find_corners(color_image)
    3D_corners ← []
    for each corner ∈ 2D_corners do
        (x, y) = closest_x_y(grid, corner.x, corner.y)
        3D_corners ← 3D_corners ∪ grid[y][x].xyz
    end for
    return 3D_corners
end procedure
procedure CLOSEST_X_Y(grid, c_x, c_y)
    return (x, y) such that (grid[y][x] ≠ empty and
    distance((x, y), (c_x, c_y)) is minimized)
end procedure
```

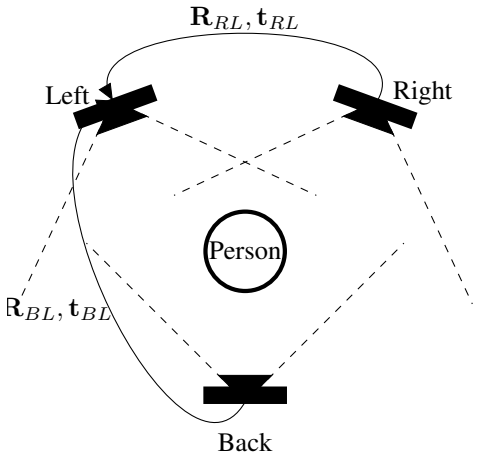


Fig. 3: An illustration of the setup with three Kinects. The right and the back Kinect were calibrated with respect to the left Kinect, which was chosen as the reference device.

2.2. Pose estimation based on 3D correspondences

Assuming that we have n pairs of 3D correspondences $\{(\mathbf{p}_A^i, \mathbf{p}_B^i)\}_{i=1}^n$, where $\mathbf{p}_A^i$ and $\mathbf{p}_B^i$ are respectively the 3D coordinates in camera A and camera B, the rigid transformation can be calculated by optimizing the following function:

$$\underset{\mathbf{R}_{AB}, \mathbf{t}_{AB}}{\operatorname{argmin}} \sum_{i}^{n} ||\mathbf{R}_{AB}\mathbf{p}_A^i + \mathbf{t}_{AB} - \mathbf{p}_B^i||^2. \quad (1)$$

In the next section we will demonstrate how we find the 3D correspondences between two Kinects.

2.3. 3D transformation

After the real-world coordinates of the corners on the calibration pattern are acquired from both devices we are able to do the transformation of the points from the coordinate system of one device into the coordinate system of the other using the calibrated rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$. If we label a point in the coordinate system of Kinect A as $\mathbf{p}_A^i$, we can transform it into the coordinate system of Kinect B as

$$\mathbf{p}_{A'}^i = \mathbf{R}_{AB}\mathbf{p}_A^i + \mathbf{t}_{AB}. \quad (2)$$

If we combine the rotation matrix and the translation vector into a 4×4 transformation matrix

$$\mathbf{T}_{AB} = \begin{bmatrix} \mathbf{R}_{AB} & \mathbf{t}_{AB} \\ \mathbf{0} & 1 \end{bmatrix}, \quad (3)$$

and put the points from each Kinect into a $4 \times n_A$ matrix

$$\mathbf{P}_A = \begin{bmatrix} \mathbf{p}_A^0 & \cdots & \mathbf{p}_A^i & \cdots & \mathbf{p}_A^{n_A} \\ 1 & \dots & 1 & \dots & 1 \end{bmatrix}, \quad (4)$$

the transformed points can be calculated as

$$\mathbf{P}_{A'} = \mathbf{T}_{AB}\mathbf{P}_A. \quad (5)$$

We will denote the set of transformed points (set of all the columns of matrix $\mathbf{P}_{A'}$, without the last row) as $\mathcal{P}_{A'}$.

By selecting one of the Kinect devices to be the reference device, the previously described methods for calibrating two Kinects can be used to get a 3D point cloud from three cameras. The cameras were positioned in such a way that they form a triangle (see Figure 3) and the left Kinect was chosen as the reference device. The remaining two Kinects were then calibrated with respect to the reference one, which allows the point clouds from all three cameras to be transformed and shown in the same coordinate system. If we denote the transformation matrix from the right Kinect's coordinate system into the left Kinect's coordinate system as $\mathbf{T}_{RL}$ and the transformation matrix from the back to the left Kinect coordinate system as $\mathbf{T}_{BL}$, the final point cloud from all three Kinects is calculated as

$$\mathcal{P}_{final} = \mathcal{P}_{left} \cup \mathcal{P}_{right'} \cup \mathcal{P}_{back'}, \quad (6)$$

where $\mathbf{P}_{back'} = \mathbf{T}_{BL}\mathbf{P}_{back}$ and $\mathbf{P}_{right'} = \mathbf{T}_{RL}\mathbf{P}_{right}$ represent the points transformed into the left Kinect's coordinate system.

3. EXPERIMENT

3.1. Implementation details

The reconstruction platform is a real-time and convenient way to acquire accurate 3D human models by using three commodity Kinect V2[1] sensors. Compared to Kinect V1, Kinect V2 offers greater precision, responsiveness, and intuitive capabilities to accelerate the development of applications that respond to movement, gestures. The sensor's color camera is enhanced with full 1080p video that can be displayed in the same resolution as the viewing screen. In addition, the new version can now track as many as six people and the tracked positions are more anatomically correct and stable. The higher depth fidelity makes it significantly easier to capture smaller objects and to capture all objects with more detail. Given their high precision performance, we rely on them for creating the moving 3D human reconstruction.

The set-up of the system is shown in Figure 1. The human subject can move freely in the center of a circle surrounded by three Kinects. Each of the Kinects records the subject from different viewpoints, followed by registration of these three views. In order to merge the separate views, consisting of the cloud points and color information, into a complete human model, the critical step is to estimate the relative position between sensors in a flexible way. This is the main focus of this paper.

The project was written in *C++* and it uses the Kinect SDK developed by Microsoft in order to communicate with the devices. The SDK provides, among others, functions for

[1] http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx

acquiring data from the *RGB* and the depth camera, a method for segmenting people from the background and a method that maps the depth data to the color data (method denoted as $get\_corr\_color\_point$ in Algorithm 1). A method was developed to combine the color data with the depth information, which allowed us to get real-world coordinates of corners on the calibration pattern extracted from the color camera. Knowing the 3D coordinates of the corners made the calculation of the previously discussed transformation matrix used to transform points from one Kinect's coordinate system to the other's possible. In this paper we only focus on the calibration and illustrate how we acquire the 3D cloud points. The 3D mesh reconstruction is beyond the scope of this paper.

3.2. Results

Since it is difficult to get the ground truth of $\mathbf{R}$ and $\mathbf{t}$, we propose to evaluate the performance of our calibration method by comparing the distance of the 3D points transformed from camera A's viewpoint to the viewpoint of camera B to the 3D points captured from camera B. We estimate $\mathbf{R}$ and $\mathbf{t}$ using only n captured frames, while the m corners from the remaining frames are used to calculate the calibration error. We calculate the 3D location of an extracted chessboard corner by the algorithm presented in Section 2.1. Then we transform them from view (A) to view (B) using the caculated $\mathbf{R}$ and $\mathbf{t}$. The mean error between $\mathbf{P}_{A'}$ and $\mathbf{P}_B$ is then calculated as follows:

$$ME = \frac{1}{m}\sum_{i}^{m} ||\mathbf{p}^i_{A'} - \mathbf{p}^i_B||. \quad (7)$$

The performance over the number of frames used for calibration is shown in Fig. 5. As we can see, using only around 7 frames, we are able to obtain an error level of 6mm for right-left cameras and 7 mm for front and back cameras at a distance of 1.5m, which is reasonably good for 3D human reconstruction. We show an example result reconstructed from three Kinects in Fig. 6.

4. REFERENCES

[1] Julie Wall, Ebroul Izquierdo, Lemonia Argyriou, D Monaghan, N OConnor, Steven Poulakos, Aljoscha Smolic, and Rufael Mekuria, "Reverie: Natural human interaction in virtual immersive environments," in *ICIP*. 2014, IEEE.

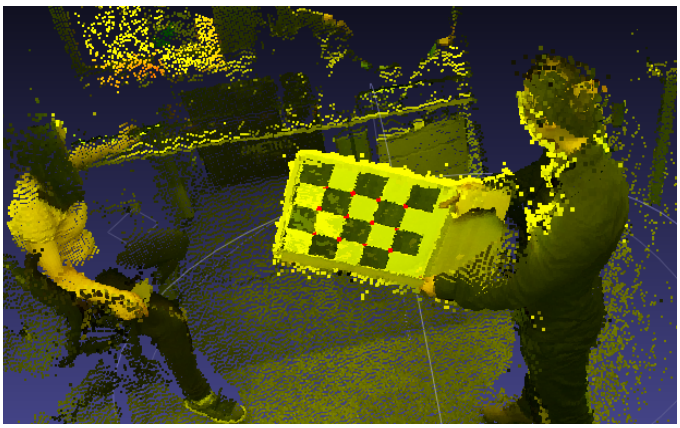
[2] Zhengyou Zhang, "A flexible new technique for camera calibration," *T-PAMI*, vol. 22, no. 11, pp. 1330–1334, 2000.

[3] Jean-Clement Devaux, Hicham Hadj-Abdelkader, and Etienne Colle, "A multi-sensor calibration toolbox for kinect: Application to kinect and laser range finder fusion.," in *16th International Conference on Advanced Robotics*, 2013.

[4] Shashwat Rohilla, *A study of a multi-kinect system for human body scanning*, Ph.D. thesis, IIT Bombay, 2014.

[5] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

(a) View from the left camera. (b) View from the back camera.



(c) Combined view from the back and left Kinects, with the detected corners plotted on the calibration board displayed in 3D.

Fig. 4: Calibration of the back and left Kinect cameras.

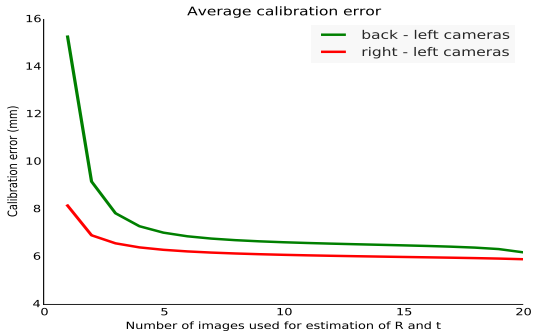


Fig. 5: Calibration error vs. the number of frames used for calibration.

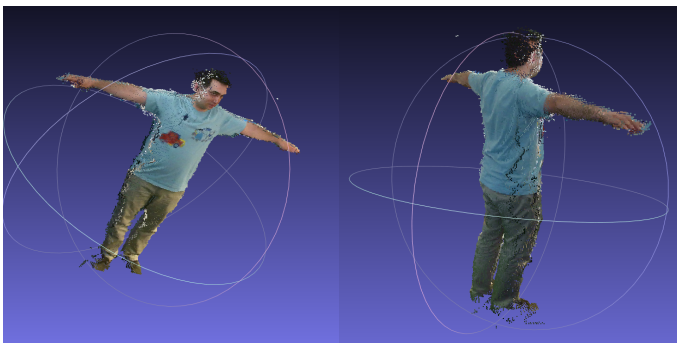


Fig. 6: Example of 3D human reconstruction.