ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 147

International Conference on Network, Communication, Computer Engineering (NCCE 2018)

Semi-Transparent Checkerboard Calibration Method for KINECT's Color and Depth Camera

Zhengyang Wu [a)], Wenzhe Zhu, Qing Zhu

Beijing University of Technology, China

[a)] Corresponding Author: zeroxyxy@gmail.com

Abstract. This article describes a method for manually calibrating KINECT color and depth cameras using a semi-transparent checkerboard, and at the same time, the two images can be modeled by 3D projection. A detailed comparison of the resulting images and data is given for the specific experimental procedure and compared to the KINECT original factory calibration. Draws better results from our method.

Key words: KINECT; calibration method; three-dimensional reconstruction.

INTRODUCTION

In recent years, with the popularity of KINECT in the general public, the number of research and applications based on KINECT has also grown exponentially. Since the use of KINECT itself is used as a gesture recognition instrument for a somatosensory game, it can be said to be quite mature in the field of depth image research. However, people have overlooked a fundamental but crucial issue, that is, KINECT also has a high-definition wide-angle color camera. This means that KINECT itself has the conditions to be used as a 3D camera.

We all know that KINECT actually uses two cameras to take color and depth images, so the most critical step in converting it to a 3D camera is to calibrate and fuse the different types of images captured by the two cameras. A three-dimensional coordinate system is established by placing each pixel on the corresponding Z porridge coordinates to form a new 3D image, and then successively forming a frame-by-frame reconstruction 3D image to form a video stream, thereby realized the function of the 3D camera.

In this paper we will only discuss the first step, which is the process of calibrating and merging depth and color images.

Since the two types of image resolution and the size of the visual field captured by KINECT itself are different, the resolution of the RGB camera is 1920*1080 with the KINECT V2 (the latest version of the KINECT), and the resolution of the depth image is only 512*424. Combining one image with the other means that the combined part will have only the overlapping parts of the color image and the depth image, while the color image that exceeds the range of the depth camera will continue to preserve the original color image.

Normally for two shots of a good picture, we only need to use some method to calibrate the two views and zoom the depth map. After the depth map is converted to a grayscale image, a certain ratio can be used to fuse the two images.

However, since our original experiment was to convert KINECT to a 3D camera for use, it means low latency (real-time). The above method is only suitable for processing a single frame of image and is not applicable to this experiment.

At the same time, all factors in the real situation can lead to additional distortion of the camera, which is unexpected geometric distortion. It is also necessary to calibrate each camera to a higher degree of reduction.

So, in summary, there are two parts we need to do. The first is to calibrate each lens, and the second is to calibrate and integrate the depth map and the color map in real time.

Therefore, we will use a manual method of semi-translucent checkerboard calibration. After the color camera and the depth camera are calibrated with a semi-transparent checkerboard, the target objects in the two video streams are passed through the corner points and the viewpoints formula calculations. Specific details will be developed in the following sections.

RELATED WORK

Affine Transformation

Since the camera distortion has been mentioned, then affine transformations have to be mentioned. Since the calibration of the images taken by each camera is the first priority, this part of the principle is also very important.

The affine model adopts the affine transformation opposite to the distortion to cancel the distortion of the image, restore the distortion angle of the image first, and then perform the distance transformation on the image. The center of the image is used as the origin of coordinates. The feature points on the distorted image are extracted, and then the image is rotated and translated using the affine transformation to correct the image. The affine model is a special case of the polynomial model. When the optical axes of the image acquisition system are basically parallel and the target distance is long, the accompanying geometric distortion is ignored. It can be considered that there are only two-dimensional parallel, rotation, and proportional changes between the images. Transformation relations can be described by an affine model.

The definition of affine transformation: If we transform S: Rn → Rn, S (x) = T (x) + a, T is a non-singular linear transformation, a ∈ Rn, the transformation S is called affine transformation. A two-dimensional image affine transformation can be expressed as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{13} & a_{14} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} ,$$

Where A is the deformation matrix and B is the translation vector. In the two-dimensional space, the affine transform deformation matrix can be decomposed in the following four steps: scale, retract, twist, and rotate.

Scales: $A1=\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$, s≠0,

Telescopic: $A2=\begin{bmatrix} 1 & 0 \\ 0 & t \end{bmatrix}$ $A_1A_2=\begin{bmatrix} s & 0 \\ 0 & st \end{bmatrix}$,

Distortion: $A3=\begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix}$ $A_1A_2A_3=\begin{bmatrix} s & stu \\ 0 & st \end{bmatrix}$,

Rotation: $A4=\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ $0\leq\theta\leq2\pi$,

In this way, the total deformation can be synthesized as:

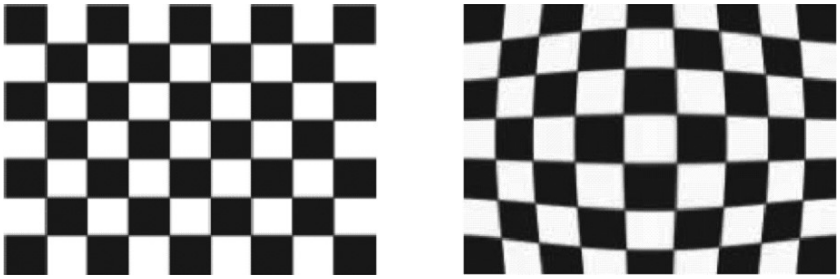
$$A_1A_2A_3A_4=\begin{bmatrix} \cos\theta\ stu\ \cos\theta - st\ \sin\theta \\ \sin\theta\ stu\ \cos\theta + st\ \cos\theta \end{bmatrix} ,$$

It can be seen that the affine transformation contains several typical low-order distortions of the image. Since the distortion of an image is a continuous process, the amount of change in geometric distortion is limited, so affine transformation can be assumed as a model of local deformation. The affine model can be simplified to:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} ,$$

(X', y') is the coordinate position of the captured image, and (x, y) is the coordinate value of the image where the corresponding point is not distorted.

Therefore, the key to achieving image correction is to find the corresponding affine transformation is $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$



(a) (b)

FIGURE 1. (a) Raw image (b) Distortion image.

Checkerboard Calibration and Feature Points

From the literal meaning we can understand that this calibration method is to use a camera to shoot a standard black and white grid checkerboard (figure 1), through a plurality of sets of data, the corresponding corner solution equation, you can perform distortion calibration.

This method was chosen as the base method because the checkerboard calibration method perfectly achieved the expected results in numerous camera calibrations, and it serves as a "fool" method that can simplify our operations, so I chose it as the base method.

The process of how to calculate and amend the specifics will not be described here (too many papers for the checkerboard calibration method), but for the feature points, it is necessary to introduce more here, because the essence of the checkerboard calibration method lies in the correspondence between the feature points and Calculations. Next, we need to use some of the key ideas in our semi-transparent checkerboard calibration method.

As a whole, feature points can be divided into two categories: narrow feature points and generalized feature points. The narrow feature points are defined for the point itself, and its position has regular attribute meanings such as edge points, corner points, intersections, and so on. Generalized feature points are defined based on regions and represent only the locations of feature regions that satisfy certain feature conditions. The generalized feature point can make any relative position of the center, the center of gravity or the feature area of a certain feature area. The feature points referred to in this article refer to generalized feature points. The extraction of feature points in this paper refers to extracting the center of gravity of each black and white area in the standard tessellation. The center of gravity is to find the average of the pixel coordinates in the object (or region) (see the formula below). Where (xi, y1) is the coordinates of the pixels in the region, and (x0, y0) is the bar centric coordinates.

$$(x_0,y_0)=\left(\frac{1}{n}\sum_{i-1}^{n-1}x_i,\frac{1}{n}\sum_{i=1}^{n-1}y_i\right)$$

If the image is not distorted, we need to calculate the resolution of the actual imaging system based on the resolution of the CCD and the magnification of the lens (see formula below). According to the resolution of the imaging system, the feature points on the standard checkerboard are converted into the relative coordinate value of the pixel. This requires knowing the physical size of the standard checkerboard and other related parameters.

Imaging System Resolution = CCD Resolution/Lens Ratio

X-direction relative coordinate value = x-direction relative position × imaging system x-direction resolution

Y-direction relative coordinate value = y direction relative position × imaging system y-direction resolution

Therefore, as long as the center of each area on the standard checkerboard is known, the relative coordinates of the pixels can be calculated according to the above formula. For the acquired distorted image, the image is enhanced, smooth filtered, and binaries to obtain a binary image, and then the resulting binary image is subjected to image morphological processing ("expansion" followed by "corrosion"), and then these the binaries images are connected into geometrically related areas, and the marked areas are sorted according to their size. By setting thresholds, small areas are removed. After obtaining a block area of 7×5 (the size of the used chessboard), an in-depth study of the area block markings can be performed to quickly extract features of each feature point.

SEMI-TRANSPARENT CHECKERBOARD CALIBRATION METHOD

Why Must Be Semi-Transparent

In order to achieve our ultimate experimental goals, we not only need to calibrate two separate cameras, but also perfectly align the depth camera with the color camera and correlate the pixels between the two cameras.

The black and white chessboard is a standard camera calibration item, but why do we need to be translucent? The answer is simple: because infrared (depth camera recording) can pass through the glass. An ordinary black and white chessboard can indeed calibrate a color camera, but it is just an ordinary rectangle in the eyes of a Kinect depth camera because the depth camera is completely color-blind. On the other hand, a translucent chessboard looks like a chessboard with two cameras at the same time, so that two cameras can be calibrated and calibrated at the same time, and the latter is the most important to us.

Make a Semi-Transparent Board

For this kind of accurate manual calibration method, first of all we need to make a relatively accurate semi-transparent chessboard object as our calibration tool.

Of course, making a precise semi-transparent board is not easy. So here I will share with you personally what I think is a relatively simple and practical method: First, print a large grid (with very fine grid lines) on a large piece of paper, like using a large format printer or plotter (copy shops may have large format printers). My goal is to produce 7 columns and 5 rows of tables (the rows and columns all need odd numbers). The size of each mesh is fixed at 9cm x 9cm, so the overall mesh size is 63cm x 45cm. Here I use a PDF file to print this grid, personally think this is the most convenient. The original file is shown in the figure below.

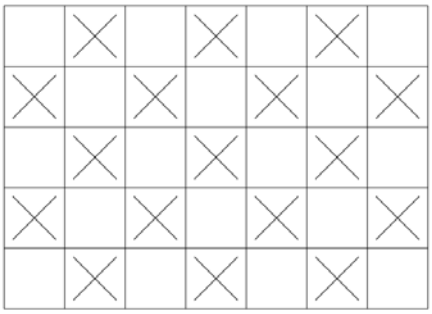


FIGURE 2. The original PDF files.

People may think that the area of the grid is too large, much larger than the board required for the board alignment method in the image. But here I must explain that the mesh in this experiment must be maintained at or below this size, and it cannot be smaller; because if it is too small, it will not be able to reliably calibrate the more distant Kinect, it should be larger; otherwise it will not be suitable for close-range view of Kinect. The mesh size I designed basically fills the entire field of view at the smallest viewing distance of the Kinect and is large enough to be used up to about 2 meters away (this is the upper limit of distance perceived by the Kinect depth).

With the preparations in place, buy a flat glass (approximately 89cm x 71cm, leaving a portion of the perimeter around the grid) and then glue the entire printed grid to the glass plate so that it is approximately centered. Then, use a long metal ruler and a very sharp knife to cut along all horizontal and vertical grid lines to ensure that all grid sheets are completely separated.

Finally, to ensure that all corners are left in place, carefully remove the grid of paper from the glass for each hop and carefully remove the remaining glue from the now transparent grid tiles. The finished product is shown in the figure below.

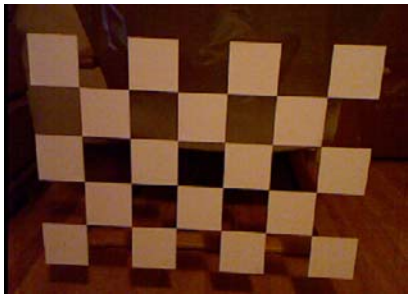


FIGURE 3. The finished product.

Experiment Process

Once we have the calibration target, we use Opens to call both KINECT cameras at the same time and display the screen in two screens. The specific code is not shown here. The result is shown in the figure below.

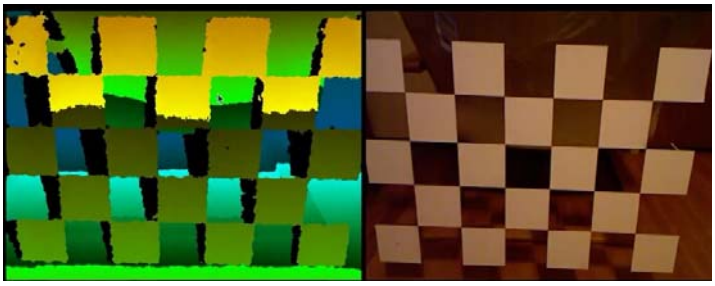


FIGURE 4. The initial screenshot (Left depth image, right color image).

After that, we used the GRID TOOL gadget to construct two virtual grids, one on each side, as same as our built check board. Meanwhile, the corner point of this figure is set as a drag gable point, and the specific code is no longer displayed. The results are shown below.

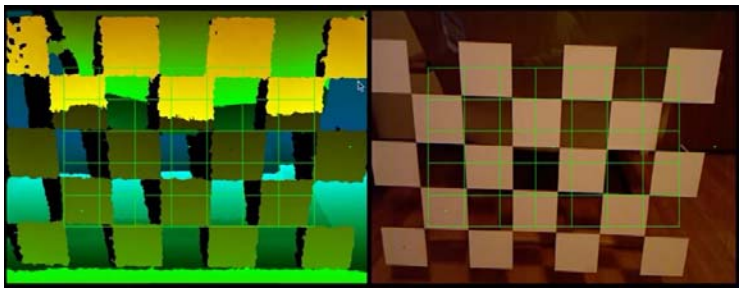


FIGURE 5. The initial image with grid.

After several experiments I discovered that the distance and the angular position of the glass plate are critical to the accuracy of the calibration. So here we will select 4 distances (nearest, stiff, far and far) and three angles (front, left and right) for each distance to calibrate. Specifically, how to do it? We will illustrate it with the map below.

Here is a demonstration of the first angle of the first distance. First of all, we will suspend the transmission of the video stream while ensuring that the entire glass panel is within the screen. At this time, the screen becomes a single-frame screenshot image. Reset out our standard drag gable 7x5 grid, drag 4 corners in turn, and overlap with the corners on the semi-transparent board, as shown.

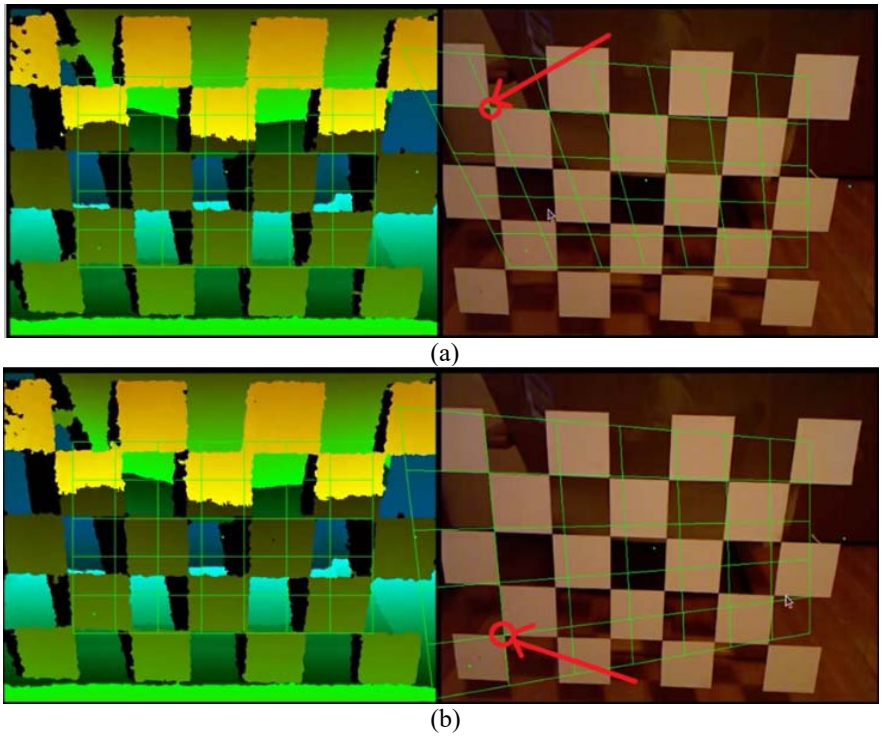


FIGURE 6. Drag 4 corner points in turn.

Finally, the eight corner points in the depth map and the color map are all overlapped and corresponded, and the situation shown in the following figure is obtained. At this time, we store the two grids that we dragged and formed as the original of the back-calibration calculation data.

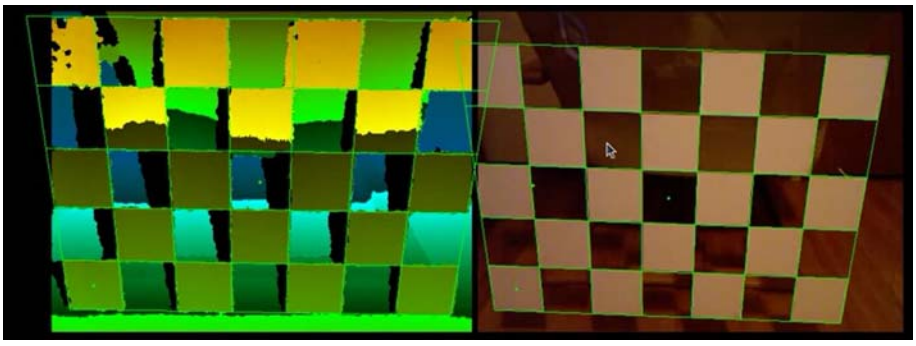


FIGURE 7. The finished screenshot (first distance, first angle).

After that we can reset the grid and continue our different calibrations at different distances. The operation is basically the same. We will not go into details here. Here we will release screenshots of different distances in the middle for better understanding.

Why do I recommend getting connection points from at least four different distances and each distance from three angles Because I want to skip almost all angles if possible (in fact, the maximum is about 60°) The angle pose is very important for establishing the constraints of the depth conversion formula, which converts the original depth value of the Kinect report into a true 3D distance.

After that, we directly carry out the fusion mentioned in the previous article (that is, 3D projection). The actual 3D distance of the pixel corresponding to the color image is the Z-axis coordinate, 3D projection, and the final fusion calibration experiment results can be obtained. As shown below

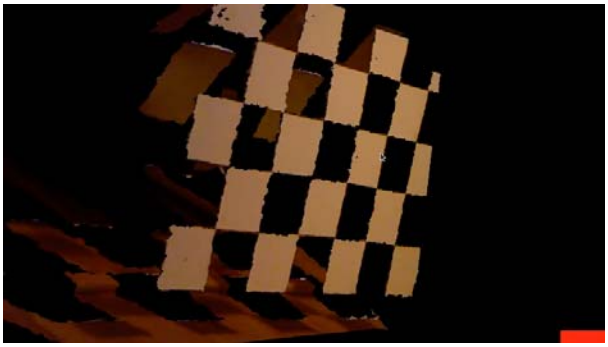
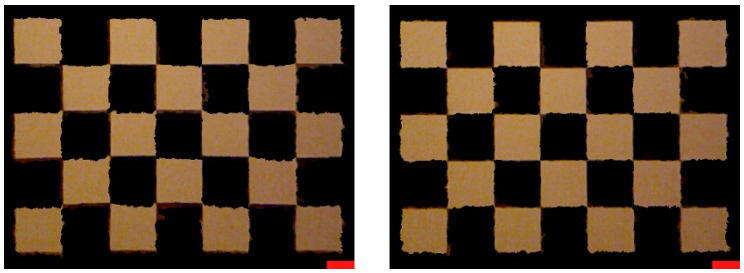


FIGURE 8. The result of 3D projection.

EXPERIMENTAL RESULTS AND ANALYSIS

Experimental Results 2D Image Comparison with Factory Calibration Results

In order to do a simple evaluation of the experimental results, I used reverse-compilation to implement the original factory-calibrated SDK in KINECT. Some of the functions included in the KINECT were similar to the ones I used, and there were some bits and pieces that I could not understand at the moment. But how good is it? We also use the factory calibrated data to create a 3D projection. These two 2D front view is shown below.



(a) (b)

FIGURE 9. The 2D front view (a) the factory calibration (b) mine.

Analysis of Results

Observed by the naked eye, our manual calibration is definitely more accurate than the other. What is the specific reason? Let's do some in-depth analysis below.

The first is the data error. The original factory calibrated the conversion formula from the (depth-corrected) raw disparity value (the content in the Kinect depth frame) to the camera space Z value, which ignores the horizontal offset of the depth image pixels, so the result Horizontal details seem less than satisfactory.

At the same time, the depth camera of Kinect has very obvious nonlinear depth distortion, especially in the corner of the depth image, and correcting this is a very simple process. In the calibration data, the correction is likely to be represented by two sets of coefficients of the bivariate polynomial, but I did not find any similar or related in the original calibration.

The most important thing is that I found the logic of the original factory calibration to be very interesting. It is exactly the opposite of what we do. The original calibration is a mapping from a depth image to a color image, that is, the pixels of the depth image are projected nonlinearly into the color image. This does in fact fit the original intention of KINECT itself as a recognizer (i.e. to find the position of an object that has been identified by that color). Our logic, on the other hand, is to project color pixels onto a depth map (which we can understand), which does not make much sense for the intended purpose of KINECT. But when we use it as a 3D camera, it is clear that the latter logic is more applicable.

Below we will use the most direct measurement method and compare with the size of the original chessboard to make the most direct data feedback.

TABLE 1. Error rate comparison.

	Actual size	My calibration		Factory calibration	
			Error		Error
Top edge	**44.5cm**	**44.38cm**	**-0.166%**	**45.13cm**	**1.549%**
Bottom edge	**44.5cm**	**44.69cm**	**0.543%**	**45.49cm**	**2.349%**
Left edge	**26.7cm**	**26.73cm**	**0.229%**	**27.18cm**	**1.895%**
Right edge	**26.7cm**	**26.84cm**	**0.819%**	**27.41cm**	**2.771%**

From the experimental data, we can see that the overall error rate of the original factory calibration is more than four times that of the semi-transparent chessboard manual calibration method, and it is clear at a glance. The reason for this I think is mainly due to two completely different mapping logics, but it has to be said that when the original factory calibration was used to project the depth image pixels non-linearly into the color image, it also calibrated the two cameras at the same time. Very bright, but in the case of this texture mapping 3D geometry, it is not as accurate as my logic.

CONCLUSION AND FUTURE WORKS

This article proposes a new KINECT dual-lens calibration method: a semi-transparent checkered manual calibration method, which is significantly better than the original factory calibration in the results (when using KINECT as a texture projection under the premise of a 3D camera).

REFERENCES

1. David Nester. An Efficient Solution to the Five-Point Relative Pose Problem. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2004.
2. Konolige, Kurt, Agrawal, Motile. Frame SLAM: From bundle adjustment to real-time visual mapping. IEEE Transactions on Robotics; Special Issue on Visual Slam. 2008.
3. Amina A, Chen Y, Curwen R W, et al. Coupled B-snake grids and constrained thin-plate splines for analysis of 2-D tissue deformations from tagged MRI. IEEE Transactions on Medical Imaging. 1998.
4. Farnia Parastoo, Ahmadian Alireza, Sedighpoor Mahdi, Khoshnevisan Alireza, Siyah Mansoory Meysam. On the performance of improved ICP algorithms for registration of intra-ultrasound with pre-MR images; a phantom study. Conference proceedings: Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference. 2013.
5. Hill D Maurer C R, Studholme Fitzpatrick J Mohawks D J. Correcting scaling errors in tomographic images using a nine degree of freedom registration algorithm. Journal of Computational Finance. 1998.
6. Collins DL, Neelin P, et al. Automatic 3D intersubjective registration of MR volumetric data in standardized Talairach space. Journal of Computational Finance. 1994.
7. Woods R Grafton S Watson J D, Sicotte N L, Mazziotta J C. Automated image registration: II. Intersubjective validation of linear and nonlinear models. Journal of Computational Finance. 1998.
8. Yale Amit. A nonlinear variation problem for image matching. SIAM Journal on Scientific Computing. 1994.
9. AKBARZADEH A, FRAHM J, MORDOHAIP, et al. Towards urban 3d reconstruction from video. 3D Data Processing, Visualization, and Transmission, Third International Symposium on. 2006.
10. Knur Arad, Nira Dyn, Daniel Reisfeld, Yehezkel Yeshurun. Image warping by radial basis functions: applications to facial expressions. CVGIP Graphical Models and Image Processing. 1994.
11. Shao H Cow C Chau L Ahmet al. 3D thin-plate spline registration for Drosophila Brain Surface Model. IEEE International Conference on Image Processing. 2013.