A Study on Face Reconstruction using Data from a Low Cost Sensor

Jinxiang ZHANG[1,2], Na LI[2]*, Jian ZHANG[2], Wanliang WANG[1], Ye DUAN[3]

1. College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou 310012, China
2. School of Science and Technology, Zhejiang International Studies University, Hangzhou 310012, China
3. Department of Computer Science, University of Missouri-Columbia, Columbia, MO 65211, USA

***Abstract* — This paper presents a system for face reconstruction and warping based on learning from point cloud data captured by Kinect sensor. First, the user is recorded in a natural environment using a low cost 3D acquisition devices--Kinect sensor. Then in order to fix the noisy and incomplete depth data captured by the Kinect sensor, we propose a method to patch the holes in depth image by the symmetric point from the face. In addition, to get enough points to create a 3D face model, a fast point clouds registration method based on SIFT is proposed. Example testifies the effect of the proposed approach.**

Keywords - face reconstruction, Kinect sensor, SIFT, point cloud

I. INTRODUCTION

Nowadays, 3D face reconstruction has received considerate attention from researchers in computer animation, computer vision and the industry domain. Various approaches have been proposed to achieve 3D face reconstruction, but some of them are quite time consuming and difficult to use. To this end, a new trend in this domain is to conduct 3D reconstruction with the aid of some sorts of simple hardware that is easily available.

Among these kinds of hardware, KINECT is an RGB-D sensor providing synchronized color and depth images. It is one of the cheapest 3D acquisition devices (see Table 1 for a comparison between some common 3D acquisition devices) [1]. It contains a normal RGB camera, a depth sensor and a four-microphone array, which are able to provide depth signals, RGB images, and audio signals simultaneously. It can offer RGB images at 640×480 pixels with 8-bit per channel. Kinect also has the option to produce higher resolution images, running at 10 frames/s at the resolution of 1280 × 1024 pixels. Its 3-D Depth Sensor has a practical ranging limit of 0.8m − 3.5m distance, and can offer deep-images at 640×480 pixels [2].

TABLE 1. COMPARISON OF 3D DATA ACQUISITION DEVICES.

Device	Speed (sec)	Size ($inch^3$)	Price (USD)	Acc. (mm)
3dMD	0.002	N/A	>$50k	<0.2
Minolta	2.5	1408	>$50k	~0.1
Artec Eva	0.063	160.8	>$20k	~0.5
3D3 HDI R1	1.3	N/A	>$10k	>0.3
SwissRanger	0.02	17.53	>$5k	~10
DAVID SLS	2.4	N/A	>$2k	~0.5
Kinect	0.033	41.25	<$200	~1.5-50

But the Kinect's data is often noisy and contains holes. To improve quality, a sequence of depth images can be registered and integrated into a 3D model. Due to these capabilities, Kinect has been used in many engineering applications like real-time 3D reconstruction and interaction [3].

Accordingly, in this paper, firstly we propose to generate 3-D models of a user's face from the inexpensive RGB-D sensor. The Kinect sensor provides depth images and RGB images, however, the depth data is often noisy and contains holes. So we use the global Euclidean distance from the specific point to nose tip to reject noisy point and to obtain the holes' symmetrical point or the neighbor fields' means to patch the holes. We use the deep image to segment the foreground and background regions, where the foreground includes the entirety of the user, and the background contains the rest of environment. The method which is proposed by G. Meyer et al. [4] is used to locate the user's head. Then a point cloud registration method based on SIFT is proposed. Because invariant feature transform (SIFT) algorithm has the rotation and scale invariant SIFT descriptor, we use it to find images' feature point, and to match two or more images' feature point. Our method uses three best paired points for the two point cloud registration..

The rest part of the paper is organized as follows: in Section 2, we introduce the method to obtain the 3D face point cloud by Kinect; In Section 3, we present the registration method based on SIFT is presented. Section 4 concludes this paper.

II. FACIAL POINT CLOUD

A. Data acquisition

The Kinect sensor is capable of capturing 640*480 pixels' RGB image and 640*480 pixels' depth images. Using the Kinect Fusion we can obtain some feature's points, such as nose tip point, left eye point, right eye point and mouth corner points.

B. Segmentation

The depth image includes depth measurements for the whole scene within view of the sensor. In our experiments, we assume the user is sitting or standing upright near the

fixed Kinect with his head and shoulders clearly visible. To measure the facial point as many as possible, we require that the user's pixels is more than 30 percent of the entire scene. Our method segments the depth image into foreground and background regions, where the user is included in the foreground regions. To determine the foreground regions, we generate a depth pixels histogram. We use ten pixel values to a bin. So the histogram includes 26 bins, and each bin contains a count for a set of pixel value. We search the maximum value of the histogram from bin 1 to bin 26. Then we search the minimum value from the two maximum values. The minimum value is a threshold which is used to distinguish foreground from background regions. To determine the location of the user's head within the foreground region, we use the method from [4].

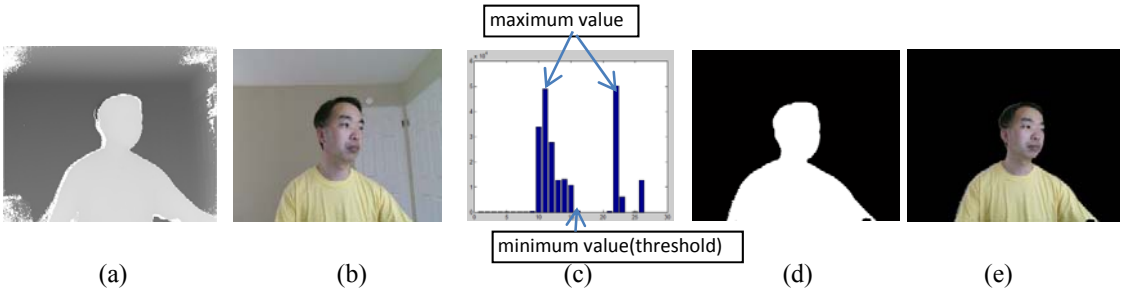

Figure 1. (a) depth image (b) RGB image (c) depth image histogram (d) the foreground of depth image (e) the foreground of RGB image (RGB image maps by the foreground of depth image)

C. *Filtering*

The depth data captured by the Kinect sensor is often noisy and contains holes. The noisy data affects point cloud registration. To improve registration, we correct the noisy data and patch the hole. We correct the noisy data with a global Euclidean distance $Dis(i,j)$. We assume that point (i,j) is in the image, and its spatial coordinates $\boldsymbol{I}(i,j)$ is (x,y,z), (x_1,y_1,z_1) is given to the tip of the nose. In (x), is the threshold. In our experiment, is 0.2.

$$Dis(i,j) = \sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2} \quad (1)$$

$$I(i,j) = \begin{cases} I(i,j) & Dis(i,j) < \delta \\ 0 & Dis(i,j) \geq \delta \end{cases} \quad (2)$$

Because human face is symmetrical, our method eliminates the holes by searching the hole's symmetric point and replacing it. Our method consists of four step:

Step1: We find the plane P by face feature's point, including nose tip point, left eye point, right eye point and mouth corner points. The face is divided into left and right parts by the plane P. Left face and right face is symmetrical.

Step2: We find the hole a_1 and calculate its symmetric point a_2 which is symmetric by plane P.

Step3: We compute the means m_1 of neighborhood of a_1 and compute the means m_2 of neighborhood of a_2.

Step4: If the difference between m_1 and m_2 is less than a threshold, a_1 is replace by a_2. Otherwise a_1 is replace by m_1.

III. REGISTRATION

Because the Kinect sensor is a low-resolution camera, it may not provide enough points to create a 3D model in a single scan. We need to align several point clouds. Iterative closest point (ICP) algorithm has become the dominant method for aligning three dimensional point clouds. ICP was proposed by Besl and Mckay [5] in 1992. Chen and Medioni [6] in 1992 and Bergevin et al.[7] in 1996 proposed a precise registration method for the nearest point by Point-to-Plane searching. In 1998, Schutz et al. [8] proposed Multi-feature matching algorithm for free-form 3D surface registration. Rusinkiewicz and Levoy [9] proposed efficient variants of the ICP algorithm in 2001. Chetverikov et al. [10] proposed the trimmed iterative closest point algorithm in 2005. But ICP algorithm has two problems. One problem is that ICP algorithm can't provide a good effect for aligning point clouds when the point clouds are not input for initialization. We assume that the point number from the first point cloud is N_p, and the point number from the second point cloud is N_r. The time cost of ICP is N_p*N_r. The other problem is that ICP algorithm spends a much longer time to align point cloud when the point number from the point cloud is increased.

To solve the problem of the ICP algorithm, we propose a new method. Firstly we uses the SIFT to find a set of feature points and calculate their descriptors from many faces. We assume that $A=\{a_1, a_2,\ldots,a_{n1}\}$ is the descriptor of feature point set which is searched by SIFT from the first face F_1. Similarly, we obtain the descriptor of feature point set $C=\{c_1,c_2,\ldots c_{n2}\}$ from the second face F_2. Then we obtain the pairs $\{(s_i,d_i)\ \ i=1,2,\ldots,m\ \ s_i \in F_1\ \ d_i \in F_2\}$ by comparing the descriptors A and C. Three specific points in the face can make up of an angel. Whenever the face is enlarged, reduced or rotated, the size of angle is an invariant. So we can search the best three pairs by comparing the angles. We

assume the three pairs are $\{(s_i,d_i),(s_j,d_j),(s_k,d_k)\}$. Firstly we compute two vectors $\overrightarrow{s_is_j}$ and $\overrightarrow{s_is_k}$ where i,j and k are the deferent numbers. Then we compute the angle between vector $\overrightarrow{s_is_j}$ and $\overrightarrow{s_is_k}$ where the operator '•' is vector inner product. We also compute the angle between vector $\overrightarrow{d_id_j}$ and $\overrightarrow{d_id_k}$. At last, the minimum deference T between α_{ijk} and β_{ijk} is searched. We propose that the three pairs are the best pairs when the deference of angle is T.

$$\overrightarrow{s_is_j} = s_j - s_i \quad \overrightarrow{d_id_j} = d_j - d_i \quad (0 < i \neq j \leq m) \tag{3}$$

$$\alpha_{ijk} = \arccos \frac{\overrightarrow{s_is_j} \bullet \overrightarrow{s_is_k}}{|\overrightarrow{s_is_j}| \cdot |\overrightarrow{s_is_k}|} \quad \beta_{ijk} = \arccos \frac{\overrightarrow{d_id_j} \bullet \overrightarrow{d_id_k}}{|\overrightarrow{d_id_j}| \cdot |\overrightarrow{d_id_k}|} \quad (0 < i \neq j \neq k \leq m) \tag{4}$$

$$T = \underset{i,j,k:1 \to m}{\operatorname{argmin}} |\alpha_{ijk} - \beta_{ijk}| \tag{5}$$

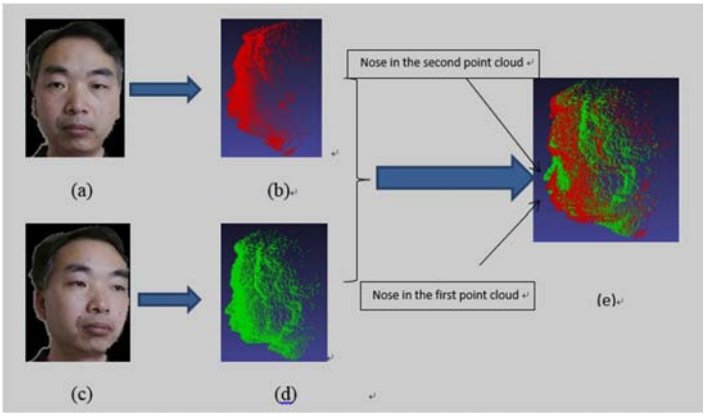

Figure 2. (a) The first RGB image (b) The first raw point cloud (c) The second RGB image (d) The second raw point cloud (e) The point cloud after aligning by ICP and merging the two point clouds

After computing the best three pairs, we can align the two point clouds. We assume the point cloud P for the first face, and Q for the second face. We calculate the two point clouds registration by the following steps.

Step1: we compute the plane's normal vector $\vec{F}$ which includes the points s_i,s_j and sk. The operator '×' in Eq. 6 is vector cross product. As the same, we also compute the plane's normal vector $\vec{G}$ which includes the points di,dj and dk.

$$\vec{F} = \overrightarrow{s_is_k} \times \overrightarrow{s_is_j} \qquad \vec{G} = \overrightarrow{d_id_k} \times \overrightarrow{d_id_j} \tag{6}$$

Step 2: The angle β which is between the normal vectors $\vec{F}$ and $\vec{G}$ is computed by Eq. 7, where $|*|$ is the vector's magnitude. The normal vector $\vec{H}$ of the plane which includes vectors $\vec{F}$ and $\vec{G}$ is computed.

$$\beta = \arccos \frac{\vec{F} \cdot \vec{G}}{|\vec{F}| \cdot |\vec{G}|} \tag{7}$$

$$\vec{H} = \vec{G} \times \vec{F} \tag{8}$$

Step 3: We make the point cloud P rotate β by the line whose director vector is $\vec{H}$ and through point si. First, we convert the director vector $\vec{H}$ into a unit vector $\vec{A} = [a_x, a_y, a_z]$ by Eq. 9. The matrix $\hat{A}$ and A* are calculated from Eqs. 10 and 11, the matrix M is also calculated from Eq. 12 , and I in Eq. 12 is an identity matrix of size 3. Then we obtain the rotated point cloud P1 by Eq. 13. After rotation transformation, the points si, sj and sk are convert to points s_i',s_j' and s_k'. At this time, the plane which includes the points s_i',s_j' and s_k' is parallel to the plane which includes the points d_i,d_j and d_k.

$$\vec{A} = \frac{\vec{H}}{|\vec{H}|} \tag{9}$$

$$\hat{A} = \vec{A}^T * \vec{A} \tag{10}$$

$$A^{*} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \quad (11)$$

$$M = \hat{A} + \cos\beta \cdot (I - \hat{A}) + \sin\beta \cdot A^{*} \quad (12)$$

$$P1 = (P - s_i) \cdot M^{T} + s_i \quad (13)$$

Step 4: In this step, the two planes are made coplanar. We compute the difference between s_i' and d_i, then all points in the point cloud P' are added to the difference. The points s_i',s_j' and s_k' convert to points s_i",s_j" and s_k", the point cloud P_1 convert to the point cloud P_2, and the point s_i" and d_i coincide.

Step 5: The angel γ which is between the vectors $\overrightarrow{s_i"s_j"}$ and $\overrightarrow{d_id_j}$ is computed. The normal vector $\overrightarrow{H_1}$ of the plane which includes vectors $\overrightarrow{s_i"s_j"}$ and $\overrightarrow{d_id_j}$ is computed.

Step 6: We make the point cloud P_2 rotate by an angle γ about the line whose direction is $\overrightarrow{H_1}$ through point d_i. After this step, the three pairs are aligned, the point cloud P_2 is converted to the point cloud P_3, and the point cloud P_3 and Q are also aligned.

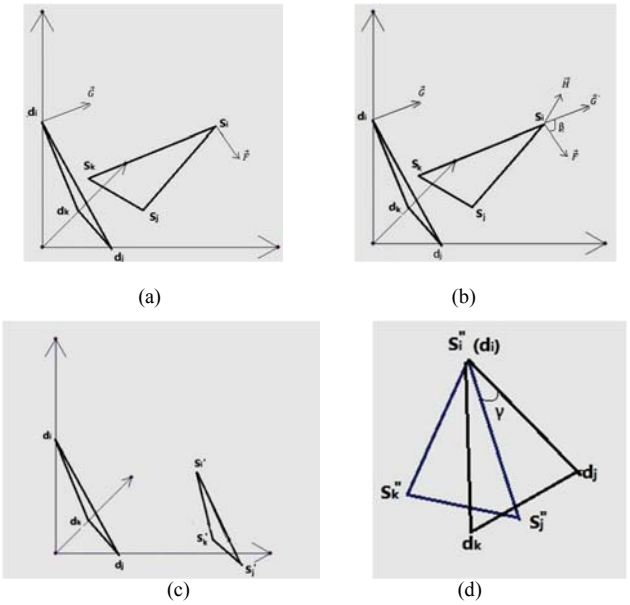

Figure 3. (a) plane $s_is_js_k$ and plane $d_id_jd_k$ with their normal vector $\vec{F}$ and (b) we move the vector $\vec{G}$ from point d_i to s_i, named $\vec{G'}$. The normal vector $\vec{H}$ and angle β between the normal vectors $\vec{F}$ and $\vec{G}$ are computed. (c) After step 3, the two planes are parallel. (d) After step 4, the vectors $\overrightarrow{s_i"s_j"}$ and $\overrightarrow{d_id_j}$ are in the same plane, but do not coincide, the vectors $\overrightarrow{s_i"s_j"}$ and $\overrightarrow{d_id_j}$ have an angle γ..

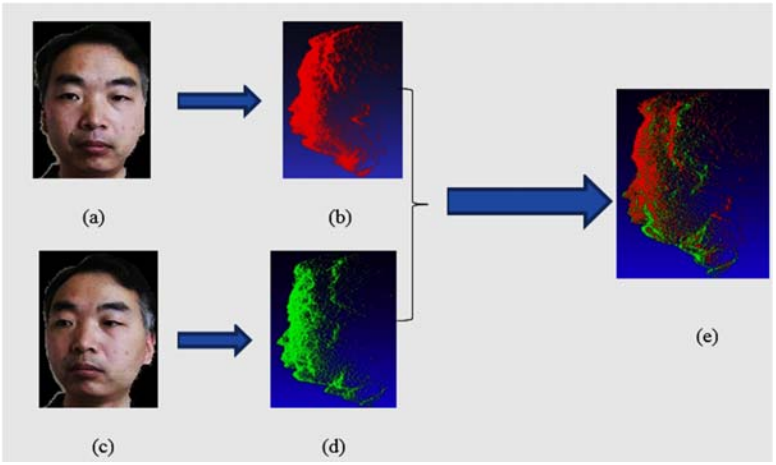

Figure 4. (a) first RGB image (b) first raw point cloud (c) second RGB image (d) second raw point cloud (e) point cloud after aligning by our method and merging the two point clouds

In most cases, our method is effective. When the three pairs which we search is not precise enough, the point cloud is not always well aligned. To this case, we need continue to precisely align. We assume the pairs $\{(p_i,d_i)\ \ i=1,2\ldots m\ p_i \in P3\ \ d_i \in Q\}$. The error pairs are removed according to the distance between p_i and d_i. if the distance between p_i and d_i is greater than a threshold δ, we assume this pair is an error pair. After error pairs are deleted, we store reserved p to $\boldsymbol{U}=\{u_1,u_2,\ldots,u_n\}$. Then U is regarded as source point cloud, and Q is regarded as reference point cloud. We can get the rotation matrix TR and translation vector TT for the ICP algorithm. At last we use (14) to get P4.

$$P4 = P3 * TR + TT \tag{14}$$

IV. CONCLUSION

We propose a novel approach to reconstructing realistic 3D face model from the Kinect sensor. Our main contributions are two parts. First, we propose a method to patch the holes in depth image by the symmetric point from the face. Second, develop a fast point clouds registration method based on SIFT is proposed. We provide examples in the paper to show the 3D reconstruction results.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (No. 61303143), Zhejiang Provincial Natural Science Foundation of China (No. LY17F020009, No. LQ14F020003), the State Scholarship Fund of China (No. 201508330643), and the Scientific Research Fund of Zhejiang Provincial Education Department (No. Y201223425).

REFERENCES

[1] B. Y. L. Li, A. S. Mian, W. Liu, and A. Krishna, "Using kinect for face recognition under varying poses, expressions, illumination and disguise", In WACV, pp. 186–192, 2013.

[2] J. Han , L. Shao , D. Xu and J. Shotton, "Enhanced computer vision with MicrosoftKinect sensor: A review", IEEE Trans. Cybern., vol. 43, pp. 1318-1334, 2013.

[3] S. Izadi , D. Kim , O. Hilliges , D. Molyneaux , R. Newcombe , P. Kohli , J. Shotton , S. Hodges , D. Freeman , A. Davison and A. Fitzgibbon, "KinectFusion: Real-time 3D reconstruction and interaction using a movingdepth camera", Proc. 24th Annu. ACM Symp. User Interface Softw. Technol., pp. 559-568, 2011.

[4] G. Meyer, M. Do, "Real-time 3D face modeling with a commodity depth camera", Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on (Jul. 2013), pp. 1–4, 2013.

[5] Paul J. Besl, Neil D. McKay, "A method for registration of 3-D shapes", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.14, pp.239-256, 1992.

[6] Y. Chen, G. Medioni, "Object Modeling by Registration of Multiple Range Images", Image and Vision Computing, vol. 10, pp. 145-155, 1992.

[7] R. Bergevin, M Soucy, H. Gagnon, et al, "Towards a general multi-view registration technique", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 18, pp. 540-547, 1996.

[8] C. Schutz, T. Jost, H. Hugli, "Multi-feature matching algorithm for free-form 3D surface registration", Proceedings of the 14th International Conference on Pattern Recognition, Vol. 2, pp. 982–984, 1998.

[9] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm", 3D Digital Imaging and Modeling Int. Conf., pp. 0-145, 2001.

[10] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust Euclidean Alignment of 3D Point Sets: The Trimmed Iterative Closest Point Algorithm", Image Vision Computing, vol. 23, pp. 299-309, 2005.