Acquisition and Processing of 3D Geometry

ICP Revisited

Final Project

Name: **Yinji Zhu** ID: **18062537** Email: **ucaby22@ucl.ac.uk**

May 29, 2019

Division

Yinji Zhu: " Codes of Kronecker, W including being dropped correspondence, D and U", "Combine the code of non-rigid ICP", Half of visualization", "Half of evaluation" + own report

Xiyue Guo: "Codes of D_L, U_L", "Find libraries, landmarks", "Sub-sampling", "ICP optimization", Half of visualization", "Half of evaluation" + own report

Introduction

Registration two surfaces means to find the mapping from source (template) surface to the target surface[1]. To obtain a good performance of mapping, the total difference between transformed source vertices with target vertices needs to be minimized. In the previous coursework, rigid iterative closest point (ICP) algorithm was used to align two models by rigid transformation, which means that the transformation preserves the relationship between every pair of points.

In the current stage, the source and target are not derived from the rigid models. However, rigid transformation does not change the distance relationship between any two points. In this case, rigid ICP cannot achieve good performance. To choose the correct deformation, nonrigid transformation is developed, which retains the convergence of rigid ICP and regulation is introduced to build morphable models[1].

Method

Two meshes are defined in the nonrigid algorithm, source $\mathcal{S}$ and target $\mathcal{T}$. The source $\mathcal{S} = (\mathcal{V}, \mathcal{E})$ has a set of n vertices $\mathcal{V}$ and m edges $\mathcal{E}$. The aim of the nonrigid algorithm is to compute transformations X, which projects the set of source vertices V to the target mesh $\mathcal{T}$, to a reparameterize version of the original scan.

The algorithm consists of the following steps: Firstly, initialise a transformation matrix X^0, which is a 3×4 transformation matrix per template vertex. Since the existence of n vertices, the matrix is a $4n \times 3$ matrix: $\boldsymbol{X} := \begin{bmatrix} \boldsymbol{X}_1 & \cdots & \boldsymbol{X}_n \end{bmatrix}^T$. Next, use a rigid ICP first to preliminary align two models. Rigid ICP algorithm calls a function from MATLAB library[2]. In this case, nonrigid can find one to one vertices more accurately. Then, a stiffness loop is used to find a series of transformation to let the source model be closer and closer to the target model.

The stiffness parameter determines the amount of acceptable local deformation. It means that large stiffness allows small deformation and small stiffness allows large deformation. Consequently, this step is achieved by starting with a large stiffness value and gradually decreasing stiffness value to let the source vertices match the correct corresponding point. A deformation loop is in the loop of stiffness, which calculates the correspondence between the iterated transformed source and the target. The new correspondence will be used in the next iteration until the value of correspondence converges. After the transformation with the smallest stiffness converges, the result is obtained.

Three terms are used in the function, they are distance term, stiffness term, and landmark term. The full cost function can be written as the sum of these terms:

$$\overline{E}(\boldsymbol{X}) := \overline{E}_d(\boldsymbol{X}) + \alpha E_s(\boldsymbol{X}) + \beta E_l(\boldsymbol{X})$$

Finding the value of transformation in each iteration is the same as finding the minimization of $\overline{E}$ in each iteration.

Distance Term

The distance term is the distance error between deformed source and target. The formula can be written as:

$$\overline{E}_d(\boldsymbol{X}) = \|\boldsymbol{W}(\boldsymbol{D}\boldsymbol{X} - \boldsymbol{U})\|_F^2$$

where $\boldsymbol{W} := \text{diag}\,(w_1, \ldots, w_n)$, $\boldsymbol{D}$ is a $4n \times 3$ sparse matrix which contains the information of source vertices, and the corresponding target vertices can be represented by $\boldsymbol{U}$. The matrix D can be seen as:

$$D := \begin{bmatrix} v_1^T & & & \\ & v_2^T & & \\ & & \ddots & \\ & & & v_n^T \end{bmatrix}$$

Also, the correspondence matrix $\boldsymbol{U}$ can be written as: $U := [u_1, \ldots, u_n]^T$. The correspondence point is found using MATLAB toolbox *knnsearch*[3], which finds the nearest neighbour in source for each query point in target. The assignment of $\boldsymbol{W}$ based on two tests. Firstly, give zero weights to boundary target vertices. Secondly, if the angle between the normals of the meshes at transformed source and u_i is larger than $4/\pi$, the weight will be dropped as well. All other weights will be set to 1. Since most of the elements are zero in $\boldsymbol{W}$, $\boldsymbol{D}$, and $\boldsymbol{X}$, it is better to build them using sparse matrix to save memory. Conversely, all elements have values in $\boldsymbol{U}$ so is a dense matrix.

Stiffness Term

The stiffness term is the penalty term to limit large difference of the transformations between adjacent vertices. The formula can be written as:

$$E_s(\boldsymbol{X}) = \|(\boldsymbol{M} \otimes \boldsymbol{G})\boldsymbol{X}\|_F^2$$

where $\boldsymbol{M}$ is a node-arc incidence matrix. Each row of $\boldsymbol{M}$ represents an edge and each column represents a vertex. Knowing that each edge has two vertices. The lower numbered of $\boldsymbol{M}$ is assigned to -1, and the larger one is assigned to 1. This is achieved by two libraries *triangulation2adjacency* and *adjacency2incidence* from Toolbox Graph[4], which can compute the adjacency matrix and convert an adjacency matrix to an incidence matrix respectively. Consequently, the node-arc incidence matrix $\boldsymbol{M}$ can be obtained. The weighting matrix $G := \mathrm{diag}(1, 1, 1, \gamma)$ to measure the difference between the rotating part of the deformation and the translational part of the deformation. The value of γ depends on the units of vertices coordinates. Using this parameter, the data can be scaled into the $[-1, 1]^3$ cube.

Landmark Term

The landmark term is used for initialization and it helps to locate correct positions. If this term doesn't exist, some part of source vertices might move to the wrong places. Consequently, this term is an essential part of the cost function. The formula can be written as:

$$E_l(\boldsymbol{X}) = \|\boldsymbol{D}_L\boldsymbol{X} - \boldsymbol{U}_L\|_F^2$$

All landmarks need to be chosen manually. $\boldsymbol{D}_L$ represents the set of landmarks chosen from source and $\boldsymbol{U}_L$ represents the set of landmarks chosen from the target. $\boldsymbol{D}_L$ corresponds to $\boldsymbol{U}_L$. For example, if $\boldsymbol{D}_L$ set a landmark of tip of nose from the source, then $\boldsymbol{U}_L$ should set a landmark of nose tip from a similar position of the target. In this case, $\boldsymbol{D}_L$ vertices will deform closer to $\boldsymbol{U}_L$ vertices after transformation.

Combination

Combining three terms above, the complete cost function can be written as:

$$\overline{E}(\boldsymbol{X}) = \| \begin{bmatrix} \alpha \boldsymbol{M} \otimes \boldsymbol{G} \\ \boldsymbol{W}\boldsymbol{D} \\ \beta \boldsymbol{D}_L \end{bmatrix} \boldsymbol{X} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{W}\boldsymbol{U} \\ \beta \boldsymbol{U}_L \end{bmatrix} \|_F^2$$

where the first row represents stiffness, the second row represents distance, and the third row represents landmark. α influences the flexibility of the source and β influences the strength of landmarks. The formulation is a little different than the formula put forward by Brain[1], since β is a weighting coefficient to the landmark term. It makes no sense if multiplies only one side.

$\overline{E}$ can be minimised by computing the derivative to zero. Set the coefficient matrix of $\boldsymbol{X}$ to $\boldsymbol{A}$ and the coefficient matrix of constant term to $\boldsymbol{B}$, so the equation above can be written as :$\|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_F^2$. Let $\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B} = 0$, to make the equation invertible, multiplying $\boldsymbol{X}^{\boldsymbol{T}}$ to each side, the equation becomes $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{B}$. Knowing that $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$ is invertible, the least square solution can be written as: $\boldsymbol{X} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{B}$. Consequently, the transformation of each iteration can be obtained.

Result and Discussion

The original code was written in c++. However, the program caused a calculation mistake when calculating the inverse of $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$. This was because the inverse statement of Eigen library would stop working if the given matrix was large. After sub-sampling to 5k vertices, it still cannot obtain the correct result. Consequently, the code is compiled based on MATLAB.

The database uses two face models from Basel University[5]. To make larger differences, one shorter face and one taller face are adopted, which can be seen in fig.1:

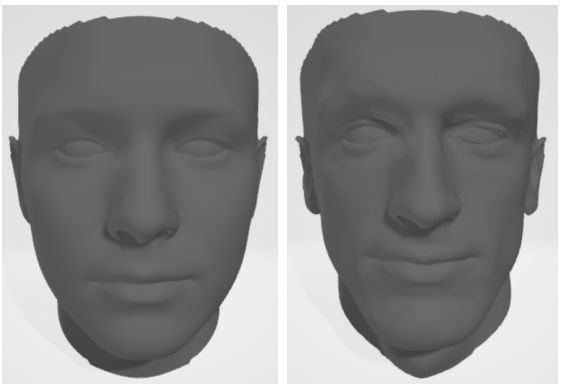

Figure 1: Left: Shorter face Right: Taller face

The left face is chosen as source and the right face is chosen as target. To reduce the

computational cost, two faces are sub-sampled into around 5k vertices.

Since the scale of the faces are 10k, to normalize the scale, all coordinates of models divide 10k rather than change the value of γ. Consequently, the value of γ is set to 1.

Landmarks used for face registration can be seen in fig.2. To obtain better performance,

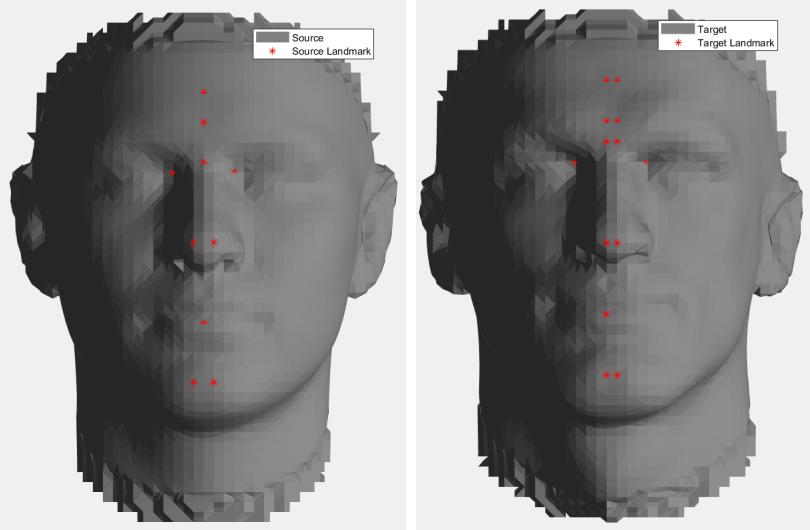

Figure 2: Left:Template landmarks Right: Target landmarks

landmarks at the eyes, mouths, noses, jaws and foreheads are chosen. The value of stiffness starts from 100 and every time it reduces 5 until the value of stiffness equals to 20. The original models and final results without landmark can be seen in fig.3 and fig.4 respectively.

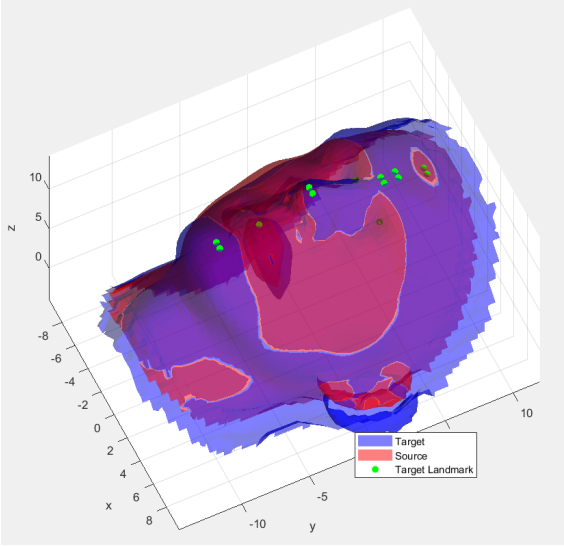

Figure 3: Original models of template and target (left)

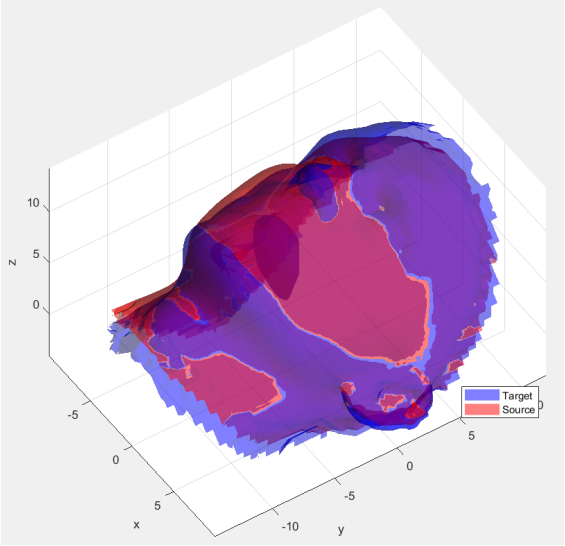

Figure 4: Nonrigid ICP Result without landmark (right)

As can be seen in the graph, the nonrigid ICP algorithm without landmark does not have good performance since the source model does not align its most parts of the face to the target model. It is because the algorithm does not have a guidance of the registration, so vertices align and converge to the wrong positions. Consequently, landmarks are required. The final error is 22.0659, which uses knn statement to obtain the correspondence points and then calculate the norm of differences.

After adding landmarks such as noses, mouths and eyes to the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the results can be seen in fig.5. Comparing with the result without landmarks, this one has

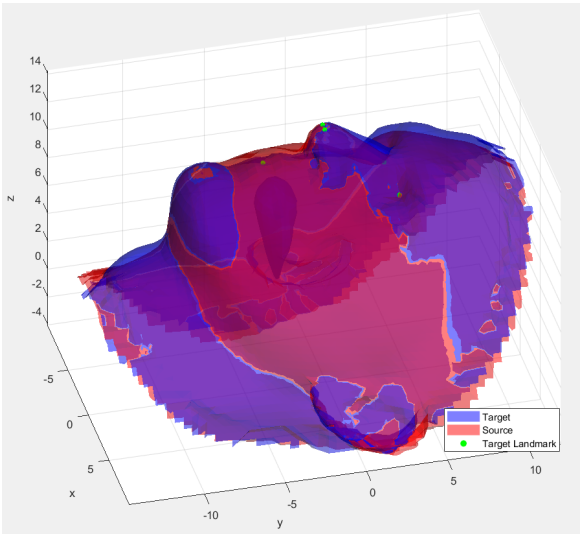

Figure 5: Result with landmarks of noses, mouths and eyes (left)

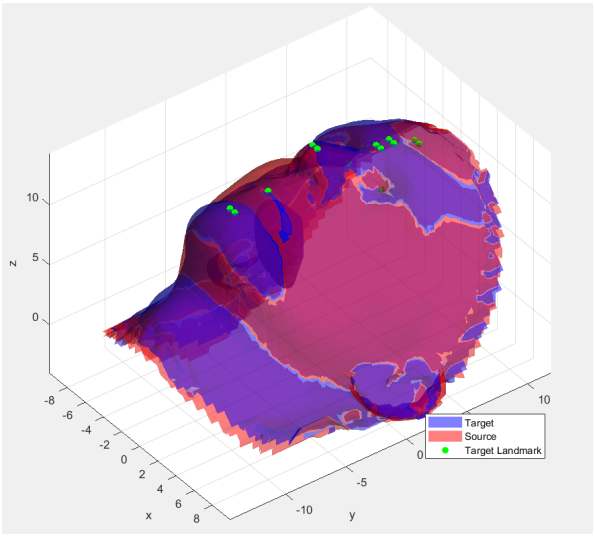

Figure 6: Final results (right)

much better performance, it is because landmarks not only can locate the current vertex but also can help the localisation of neighbours since the existence of neighbour relationships. In this stage, noses, mouths and eyes fit well. The error reduces to 18.062.

To achieve better performance, more landmarks are required to the places which do not have good alignment before. In this case, landmarks at jaws, foreheads, the places between the eye-browses are chosen, which can be seen in fig.6. The models of transformed source and target become almost the same. Comparing to the graph of fig.5, fig.6 has a distinct improvement in the area of foreheads and jaws. It proves that the result seems to have a good alignment. Consequently, the performance of nonrigid ICP can be improved by adding landmarks to the place which does not have good registration before.

The error changes in each iteration can be seen in fig.8. As can be seen in the graph, the overall variation of the error is from large to small. However, in an optimal iteration step nonrigid ICP algorithm will not always decrease. It is because of the usage of reliability weighting. When the contribution of vertices without correspondence vertices are ignored in distance term, new correspondence relationships are found after each deformation step, which will change some terms of weight. Consequently, errors may increase even though

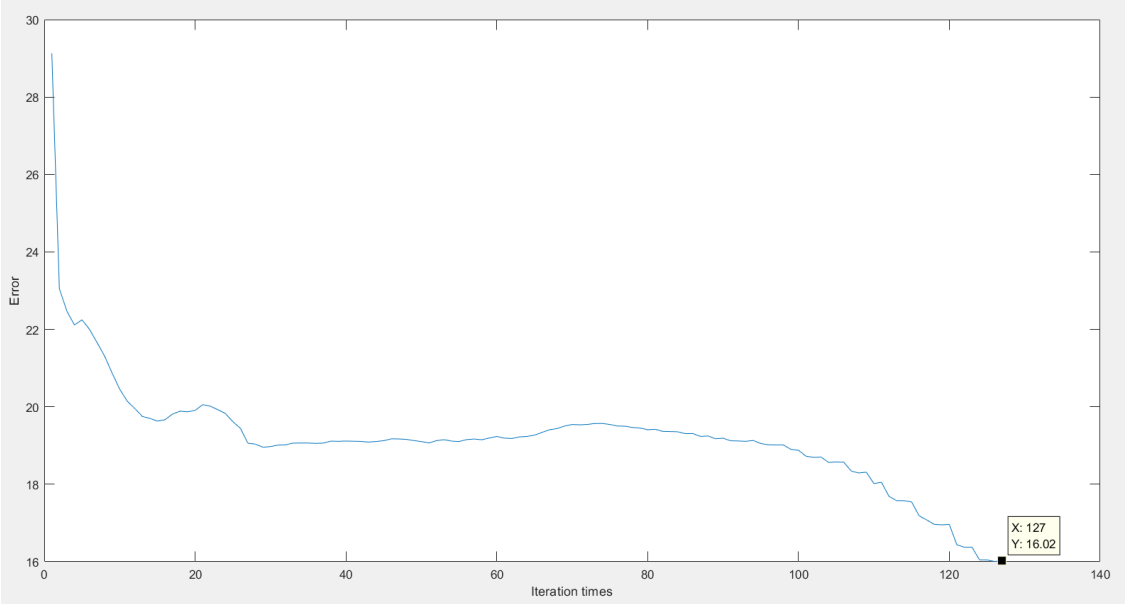

Figure 7: Result with landmarks of noses, mouths and eyes (left)

models have better alignments[1]. Moreover, the final error is 16.023, which is reduced further.

Extension

The extension part uses FAUST dataset[6]. The chosen models can be seen in fig.8, which

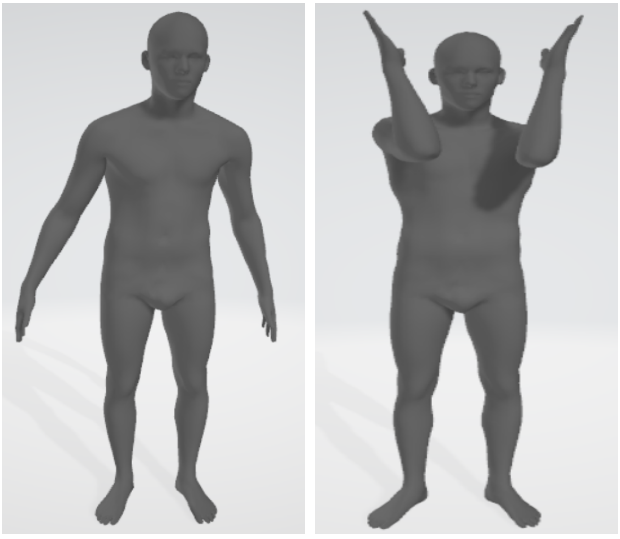

Figure 8: Result with landmarks of noses, mouths and eyes (left)

pay attention to the registration of the arm and other parts of the body show relatively small changes. The starting stiffness is smaller than the facial model since if the starting

stiffness is too large, the source model will flatten out after deformation to keep the relationship between adjacent points. The previous idea was changing partial stiffness value to matrix $\boldsymbol{M}$ or $\boldsymbol{M} \otimes \boldsymbol{G}$. However, it does not have good performance since it influences the alignments of other vertices.

One possible idea is to regard the parts of the body which have small changes as rigid parts, the smaller differences, the less number of landmarks. That is to say, shoulders and elbows need a large amount of landmarks. Also, some small body parts and edges need more landmarks for registration, such as palms and toes. Consequently, landmarks are sampled uniformly at most part of the body and more landmarks are added to the parts like shoulders, palms, and elbows. After nonrigid ICP iterations, the result can be seen in fig.9. where the red model is the deformed source, the blue model is the target, and

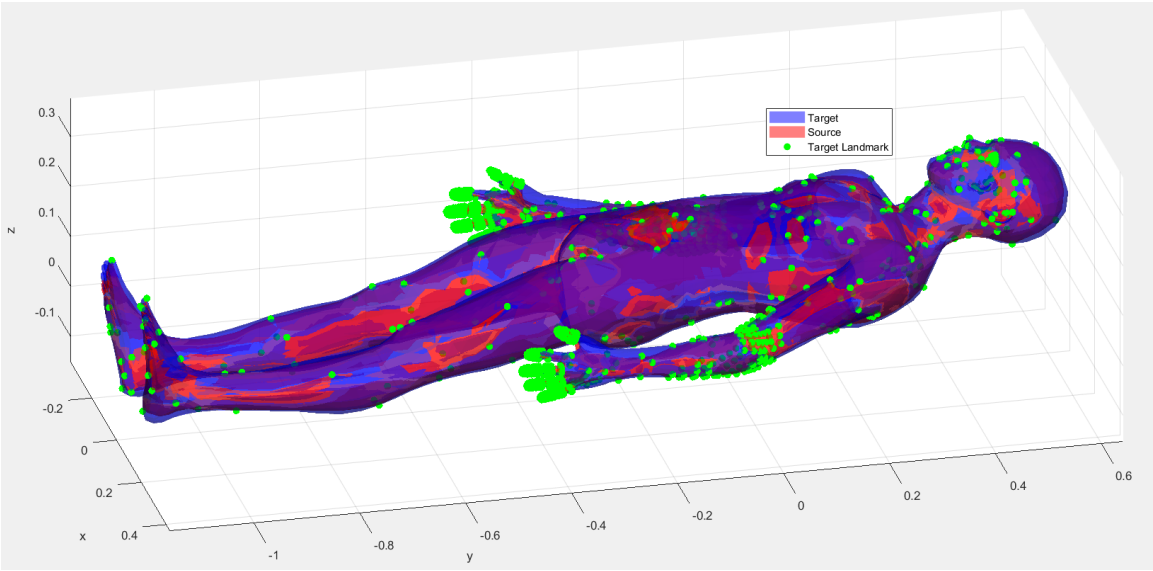

Figure 9: Result with landmarks of noses, mouths and eyes (left)

green points are landmarks. It seems to have good alignment in most parts. However, it does not work well in the wrists. It seems that not enough landmarks are added to the wrists so the template is collapsed into a point. Adding some landmarks to the specific vertices can fix this problem.

A possible way to improve this algorithm is doing nonrigid ICP separately. Firstly, separate the body to front arm, back arm, and the rest body. Firstly, do rigid ICP to the body for localization and then do nonrigid ICP to this part. Next, do rigid ICP to the back arm and then do nonrigid ICP as well. Then, apply the same algorithm to the front arm. Finally, knowing faces and vertices relationships, these three parts can be combined and then do nonrigid ICP again. The results can be obtained. However, in the current are not arranged in order. Consequently, it is hard to separate the entire arm.

References

[1] B. Amberg, S. Romdhani, and T. Vetter, "Optimal step nonrigid icp algorithms for surface registration," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[2] J. Wilm. (2012) Iterative closest point. [Online]. Available: https://ww2.mathworks.cn/matlabcentral/fileexchange/27804-iterative-closest-point?requestedDomain=zh

[3] MATLAB. (2019) knnsearch. [Online]. Available: https://ww2.mathworks.cn/help/stats/knnsearch.html

[4] G. Peyre. (2019) Toolbox graph. [Online]. Available: https://ww2.mathworks.cn/matlabcentral/fileexchange/5355-toolbox-graph

[5] *A 3D Face Model for Pose and Illumination Invariant Face Recognition*. Genova, Italy: IEEE, 2009.

[6] F. Bogo, J. Romero, M. Loper, and M. J. Black, "FAUST: Dataset and evaluation for 3D mesh registration," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Piscataway, NJ, USA: IEEE, Jun. 2014.