

## 1. Introduction

Polygon and parametric representation are two different ways to describe a 3D surface. Although polygon mesh is simpler and more flexible to model a 3D object than the parametric way, the latter is more concise. Especially it has significant advantage over polygonal representation on the deformation and shape change.

The ultimate goal of current work is to build warping matrix between two 3D faces. To achieve this goal, we must solve the corresponding problem first if polygon mesh is used for 3D surface description. However, this problem is not well solved yet. Fortunately, the parametric representation may provide another approach to the final answer. More details will be illustrated in next section.

## 2. The parametric representation

A parametrically defined curve in three dimensions can be given by three univariate functions:

$$Q(u) = ( X(u), Y(u), Z(u) )$$

where  $0 \leq u \leq 1$ . Then the extension to parametrical surface from curves is quite straightforward, which is defined by three bivariate functions:

$$Q(u,v) = ( X(u,v), Y(u,v), Z(u,v) )$$

where  $0 \leq u \leq 1$  and  $0 \leq v \leq 1$ . Therefore, we focus on the work of the curves interpolation.

The parametric curve is usually a polynomial or rational polynomial. For example, a polynomial of order  $k+1$  can be written as:

$$Q(u) = p_0 + p_1 * u + p_2 * u^2 + \dots + p_k * u^k$$

Alternatively, we can rewrite the expression above as:

$$Q(u) = \sum_{i=0}^k P_i b_i(u)$$

$b_i(u)$  are called basis functions and the coordinates  $P_i$  are control points.

Just as in  $R^3$  more than one coordinate system can specify the same point in space, there exists more than one basis that can specify the same curve or surface. The most important bases used in computer graphics are Bezier basis and B-Spline basis. Both of them are used as weights. However, there exist two distinct properties between B-spline and Bezier basis functions, i.e.:

- (1) the domain is subdivided by knots, and
- (2) each basis functions is non-zero on a few adjacent subintervals. As a result. B-spline basis functions are quite "local".

These are very useful properties for our purpose of building warping matrix. So in current work, we will concentrate on the B-spline interpolation. As Bezier curves enable an efficient patch-splitting algorithm for rendering, we can take it as the final basis in the process of modeling 3D face.

## 3. Implimentation

The input is 35 3D points divided as 7 groups and 5 points in each group (the sequence of data has critical effect on the final shape of the curves interpolated from these data). The output of the

system is a curve net defined by 2 knot-vectors, namely U-knot vector and V-knot vector. Curves interpolation consists of the following steps:

- (1) Generate parameters and knot vector for each group of data (u-knot vectors)
- (2) Interpolate curves (finding control points) group by group.
- (3) Uniform five u-knot vectors as one U-knot vector.
- (4) Data points for V-curves are obtained by moving along each of U-curves for a given knot value of the U-knot vector. V-curves are formed by interpolating through these data points similar as step (1) and (2).
- (5) Uniform five v-knot vectors as one V-knot vector.

We will exam the method of fitting one group of data points with a B-spline curve in details. In this case, we have 5 data points  $D_0, D_1, \dots, D_4$  and will fit them with a B-spline curve of degree  $p=2$  (order 3). First, I select the “centripetal method” to obtain a set of parameter values  $t_0, t_1, \dots, t_n$  and parameter  $t_k$  corresponds to data point  $D_k$ . Form these parametersm a knot vector of 8 knots is computed. So, we have a knot vector and the degree  $p$ , the only missing part is a set of control points  $P_i$ . Now, it is a question of solving a linear system of unknown  $P_i$ .

From the definition of section 2, we have

$$D_k = Q(t_k) = \sum_{i=0}^4 P_i N_{i,p}(t_k) \quad \text{for } 0 \leq k \leq 4$$

This function can be re-expressed as the form of matrix  $D = N \bullet P$

$$N = \begin{bmatrix} N_{0,p}(t_0) & N_{1,p}(t_0) & N_{n,p}(t_0) \\ N_{0,p}(t_1) & N_{1,p}(t_1) & N_{n,p}(t_1) \\ N_{0,p}(t_n) & N_{1,p}(t_n) & N_{n,p}(t_n) \end{bmatrix}$$

$$D = \begin{bmatrix} d_{01} & d_{02} & d_{0s} \\ d_{11} & d_{12} & d_{1s} \\ d_{n1} & d_{n2} & d_{ns} \end{bmatrix} \quad P = \begin{bmatrix} p_{01} & p_{02} & p_{0s} \\ p_{11} & p_{12} & p_{1s} \\ p_{n1} & p_{n2} & p_{ns} \end{bmatrix}$$

For the rest groups, the process of curve interpolation is all in the same. Now, we come to the step (3). The 5 knot-vectors we have got are:

```

0.00000 0.00000 0.00000 0.42132 0.72394 1.00000 1.00000 1.00000
0.00000 0.00000 0.00000 0.34609 0.60973 1.00000 1.00000 1.00000
0.00000 0.00000 0.00000 0.35981 0.62294 1.00000 1.00000 1.00000
0.00000 0.00000 0.00000 0.35704 0.61952 1.00000 1.00000 1.00000
0.00000 0.00000 0.00000 0.35981 0.62294 1.00000 1.00000 1.00000
0.00000 0.00000 0.00000 0.34609 0.60973 1.00000 1.00000 1.00000
0.00000 0.00000 0.00000 0.42132 0.72394 1.00000 1.00000 1.00000

```

Because the tensor product representation of a B-spline surface has one knot vector for the

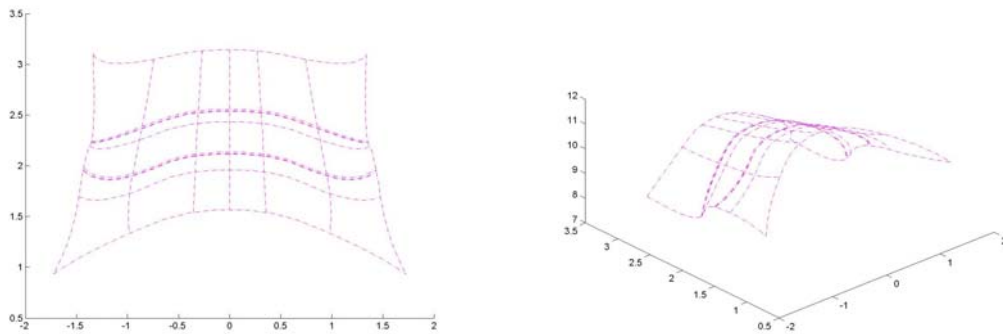
u-direction and one for the v-direction, a single knot vector must be chosen for all the U-curves. This is done by taking the union of all the distinct knot vectors to form one merged knot vector – which we shall call the U-knot vector. Each of the U-curves, in order to be described by the U-knot vector, has additional knots inserted where necessary. The U-knot vector is showed as below:

```
0.00000 0.00000 0.00000
0.34609 0.35704 0.35981 0.42132 0.60973 0.61952 0.62294 0.72394
1.00000 1.00000 1.00000
```

Similarly, we can get the V-knot vector:

```
0.00000 0.00000 0.00000
0.33511 0.33750 0.33774 0.33845 0.35426 0.36166 0.37016 0.37048 0.37168 0.38418
0.46407 0.46424 0.46427 0.46436 0.46642 0.46888 0.47082 0.47090 0.47118 0.47278
0.52721 0.52881 0.52909 0.52917 0.53111 0.53357 0.53563 0.53572 0.53575 0.53592
0.61581 0.62831 0.62951 0.62983 0.63833 0.64573 0.66154 0.66225 0.66249 0.66488
1.00000 1.00000 1.00000
```

The curve net is illustrated in figure 1.



It is obviously that the U-knot vector and V-knot vector are rather redundant. In fact, we can get the same result with much few knots. I introduced the method of deleting knots from the knot vector but keeping the shape of curve meanwhile.

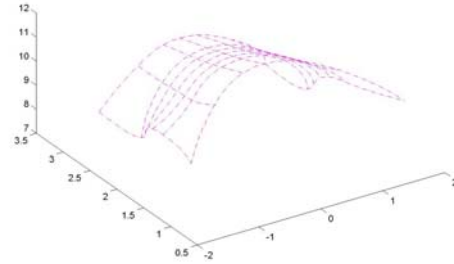
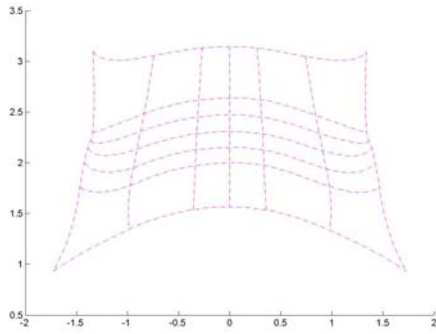
For U-knot vector, now we have

```
0.00000 0.00000 0.00000
0.30000 0.40000 0.50000 0.60000 0.70000
1.00000 1.00000 1.00000
```

While V-knot vector has the same definition

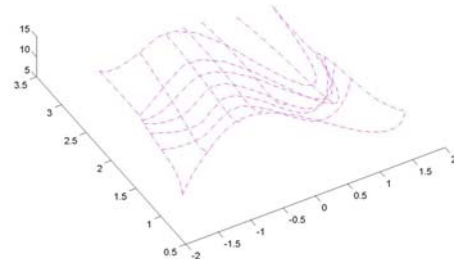
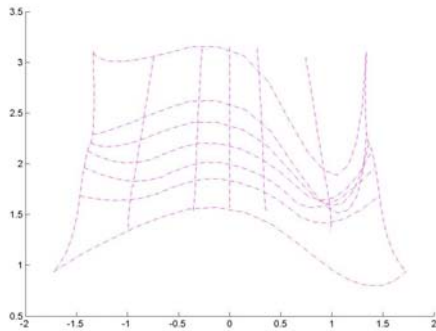
```
0.00000 0.00000 0.00000
0.30000 0.40000 0.50000 0.60000 0.70000
1.00000 1.00000 1.00000
```

The curve net is illustrated in figure 2.



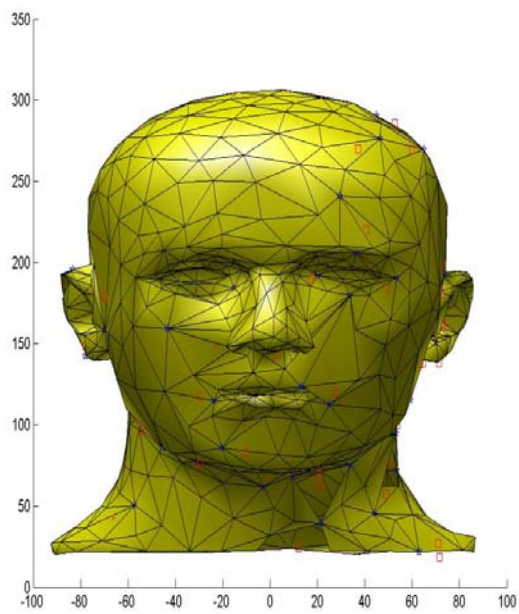
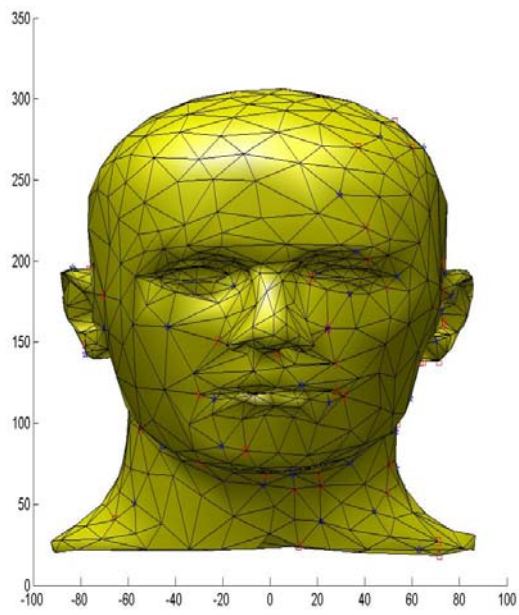
#### 4. Future work

Currently, I use the determinant (Cramer's rule) to solve linear equations  $Ax = b$ . It's not very efficient when the unknown parameters are up to 10 or more. However, because the error introduced by round calculation used in Crout's method, the interpolation result is not very satisfactory although it is very efficient. The result is shown in figure 3.

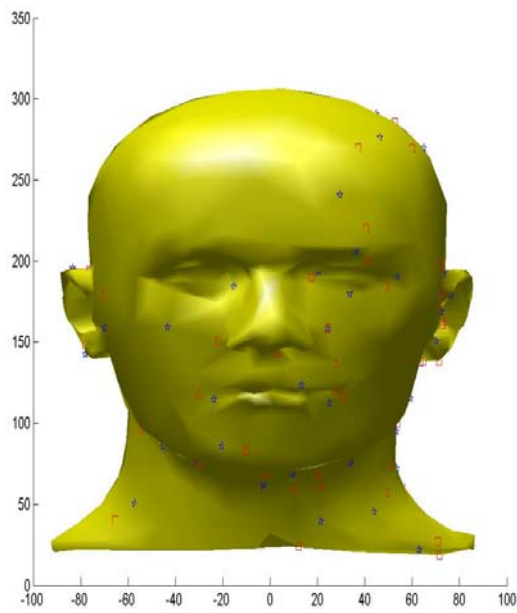


8% vertices are reduced in this result. The topological operator is half-edge collapse. One-sided Hausdorff distance is used to measure the deviation of the current mesh from the original vertices.

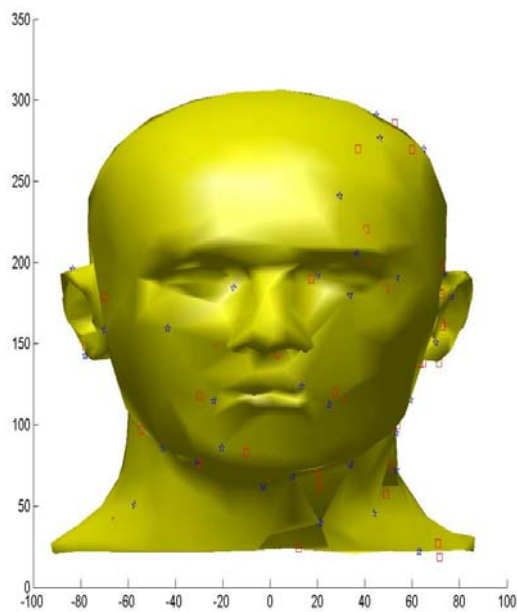
The result shows that the most important features on the face were kept from decimation. If other oracles are added to the algorithm to steer the reduction algorithm, we may get better result.



(a) Un-decimated model with Edges displayed      (b) Decimated model with Edges displayed



(c) Un-decimated model  
without Edges displayed



(d) Un-decimated model  
without Edges displayed

## Summary: applying TPS on warping 3D model

### 1. 3D model deformation

In this summary, the approach of using thin-plate splines (TPS) to warp 3D model is presented. Bookstein (1989) has developed a highly successful approach on analysing deformations for data in two dimensions using a pair of thin-plate splines. We extended the application to three dimensions. The thin-plate splines is given by the function:

$$\begin{aligned}\Phi(t) &= (\Phi_1(t), \Phi_2(t), \Phi_3(t))^T \\ &= c + At + W^T s(t)\end{aligned}$$

where  $t$  is  $(d \times 1)$ ,  $s(t) = (\sigma(t-t_1), \dots, \sigma(t-t_k))^T$ ,  $(k \times 1)$

$$\sigma(h) = \begin{cases} \|h\|^2 \log(\|h\|), & \|h\| > 0, \\ 0, & \|h\| = 0. \end{cases}$$

The parameters of the mapping are  $c$   $(d \times 1)$ ,  $A$   $(d \times d)$  and  $W$   $(k \times d)$ , in which  $d$  and  $k$  respectively represent the dimensions and the amount of landmarks being used as the control points in TPS.

From the analysis of Dryden and Mardia [1], if  $S^{-1}$  exists, the parameters  $c$ ,  $A$  and  $W$  can be calculated on the given  $2k$  landmarks  $t_j$ ,  $j = 1, \dots, k$ , on the source 3D model, which will be mapped exactly into  $y_j$ ,  $j = 1, \dots, k$ , on the destination 3D model (as marked on the Figure 1 (b) and (e)), by:

$$W = (S^{-1} - S^{-1}Q(Q^T S^{-1}Q)^{-1}Q^T S^{-1})Y$$

and

$$\begin{bmatrix} c^T \\ A^T \end{bmatrix} = ((Q^T S^{-1}Q)^{-1}Q^T S^{-1})Y$$

where  $Q = [1_k, T]$ ,  $(S)_{ij} = \sigma(t_i - t_j)$ .

Let  $T = [t_1 \ t_2 \ \dots \ t_k]^T$ ,  $Y = [y_1 \ y_2 \ \dots \ y_k]^T$ .

In current research, we concentrate on the  $d = 3$  dimensional case. Given  $k = 10$  corresponding points on the two 3D models, the matrices  $T$  and  $Y$  are,

$T = 1.0e+003 *$

-0.0467	-0.0638	0.0364	4.3040
-0.0216	-0.0727	0.0342	3.6920
0.0216	-0.0727	0.0342	0.6690
0.0467	-0.0638	0.0364	1.2810
-0.0201	-0.0893	0.0011	5.6500
0	-0.1165	0.0023	2.6470
0.0201	-0.0893	0.0011	2.6270

-0.0277	-0.0819	-0.0347	4.9450
0.0277	-0.0819	-0.0347	1.9220
0	-0.0915	-0.0782	1.9620

Y = -46.6700	2.5340	189.8830	425.0000
-17.2289	-7.9780	192.1740	510.0000
17.4216	-9.0060	189.8360	543.0000
46.6700	-2.5340	190.9890	434.0000
-18.5660	-28.6320	147.8110	632.0000
1.4533	-47.9520	153.8870	678.0000
18.7739	-27.7650	147.8550	637.0000
-29.6596	-7.9230	117.9180	577.0000
30.9331	-5.8700	116.6780	560.0000
3.3820	-20.8615	81.6230	689.0000

It is found that

W = -0.0018	-0.0002	-0.0020
0.0045	0.0016	0.0045
-0.0043	0.0027	0.0010
0.0019	-0.0015	-0.0004
0.0006	-0.0062	-0.0043
0.0001	0.0035	0.0017
-0.0010	-0.0056	-0.0031
-0.0010	0.0036	0.0015
0.0008	0.0045	0.0006
0.0002	-0.0026	0.0004

c = -10.2269
67.9676
122.2715

A = 1.0888	-0.0352	-0.0177
-0.0342	0.9789	-0.0064
0.0177	-0.0326	0.9906

In calculating a deformation 3D model, the parameter matrices W, A, and c are used to interpolate all the points (295×3) on the source 3D model. The results are illustrated in the Figure 1. Here we will briefly explain the results in Figure 1. From (a) to (c), they are frontal views of generic 3D model, destination 3D model and deformed generic 3D model by TPS interpolation, respectively. (d), (e), (f) are corresponding side views. Comparing (c) and (d) to the generic and destination model, it can be seen easily that those features near the control points are more similar to the features on the destination model. On the contrary, for the points that are farther away from the

control points, their deformation are approximately linear.

## 2. Deformation Analysis

The thin-plate spline (TPS) consists of a rigid transformation, i.e.  $c + At$ , and a non-rigid transformation, i.e.  $W^T s(t)$ . For the non-rigid transformation, Bookstein (1989, 1991) introduced the notions of principal and partial warps to help explain its components. We use the partial warps here to decompose the TPS deformation into a series of large scale and small scale components.

Consider the TPS transformation from  $t \in \mathbb{R}^3$  to  $y \in \mathbb{R}^3$ , which interpolates the  $k$  points  $T$  to  $Y$  ( $k \times 3$ ) matrices. An eigen-decomposition of the  $k \times k$  bending energy matrix  $B11$  has non-zero eigenvalues  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{k-4}$  with corresponding eigenvectors  $\gamma_1, \gamma_2, \dots, \gamma_{k-4}$ . The partial warps are defined as the set of  $k-4$  tri-variate functions  $R_j(t), j = 1, \dots, k-4$ , where

$$R_j(t) = Y^T \lambda_j \gamma_j \gamma_j^T s(t)$$

Since

$$W^T s(t) = \sum_{j=1}^{k-4} R_j(t)$$

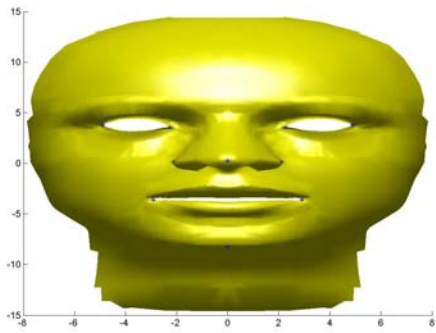
We see that the non-rigid part of TPS deformation can be decomposed into the sum of the partial warps. Furthermore, the way that the partial warps were sorted in ascent order corresponding to the order of eigenvalues encodes the deformation at various scales – the larger values of the eigenvalues (i.e. larger  $j$ ) represent small scale local deformation and the smaller eigenvalues (smaller  $j$ ) represent large scale global deformation. Hence, the first partial warp will be associated with an overall large scale bending of the model and the last partial warp will usually be associated with the landmarks that are closest together, at the smallest scale.

Currently, the question is if the decomposition of TPS transformation can be used for the purpose on analysing the shape differences among faces. So we plot each partial warp to see how big they will affect the final result. In this case, the part of non-rigid transformation was decomposed as five partial warps. We plot them separately in the Figure 2(a) to Figure 2(e). Figure 2(f) is the sum of the five partial warps. Figure 2(g) is the part of rigid transformation of TPS. Figure 2(h) demonstrate the TPS deformation, i.e. the sum of rigid and non-rigid part of TPS deformation. Figure 3 shows the side views to corresponding front views in Figure 2.

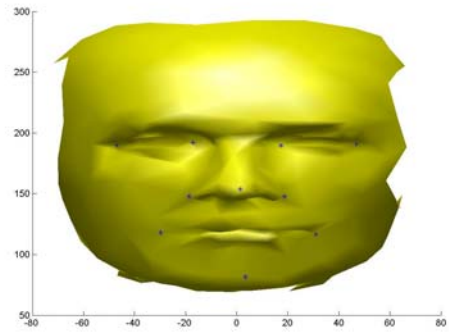
$B11 = 1.0e-003 *$

0.2865	-0.3983	0.0820	-0.0190	0.0153	0.0370	0.0487	-0.0938
0.0216	0.0200						
-0.3983	0.8265	-0.3764	0.0820	-0.2962	0.0076	0.0464	0.1150
-0.0039	-0.0027						
0.0820	-0.3764	0.8265	-0.3983	0.0464	0.0076	-0.2962	-0.0039
0.1150	-0.0027						

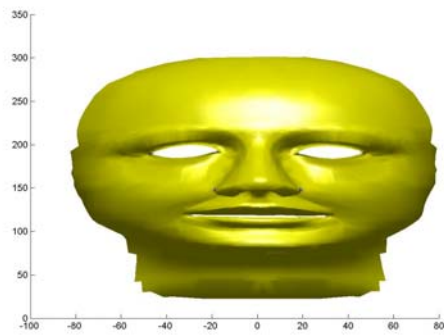
-0.0190	0.0820	-0.3983	0.2865	0.0487	0.0370	0.0153	0.0216
-0.0938	0.0200						
0.0153	-0.2962	0.0464	0.0487	0.7762	-0.1852	-0.1475	-0.3364
0.0108	0.0679						
0.0370	0.0076	0.0076	0.0370	-0.1852	0.1340	-0.1852	0.0961
0.0961	-0.0452						
0.0487	0.0464	-0.2962	0.0153	-0.1475	-0.1852	0.7762	0.0108
-0.3364	0.0679						
-0.0938	0.1150	-0.0039	0.0216	-0.3364	0.0961	0.0108	0.3357
-0.0178	-0.1274						
0.0216	-0.0039	0.1150	-0.0938	0.0108	0.0961	-0.3364	-0.0178
0.3357	-0.1274						
0.0200	-0.0027	-0.0027	0.0200	0.0679	-0.0452	0.0679	-0.1274
-0.1274	0.1296						



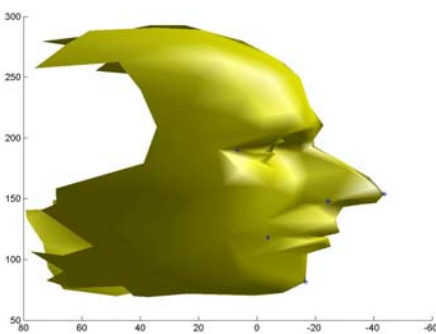
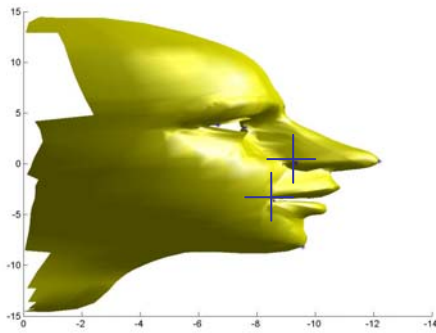
(a) Source 3D Model



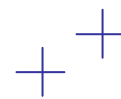
(b) Destination 3D Model



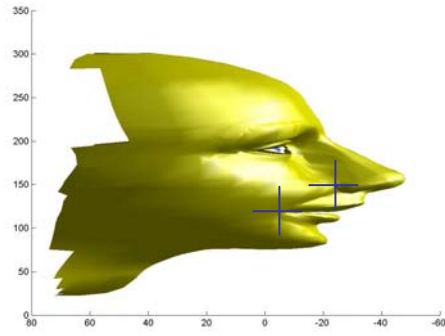
(c) Using warping matrix (TPS) to interpolate source 3D model



(d) Source 3D model profile

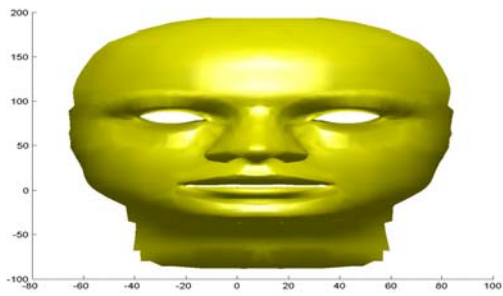


(e) Destination 3D model profile

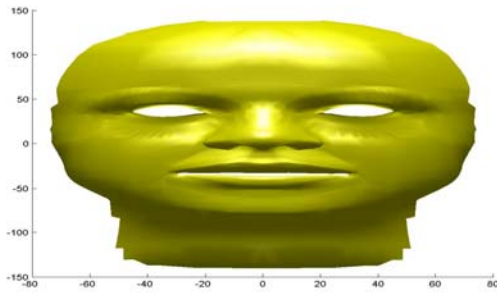


(f) Using warping matrix (TPS) to interpolate source 3D model profile

Figure 1 Using TPS to warp 3D model

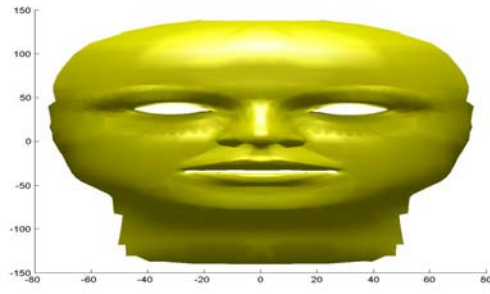


(a) first partial warp



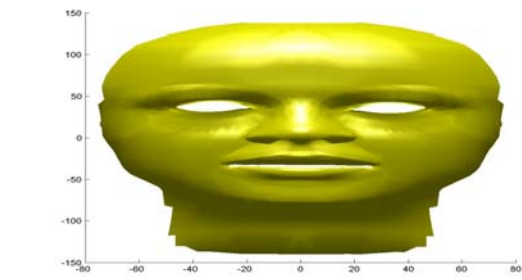
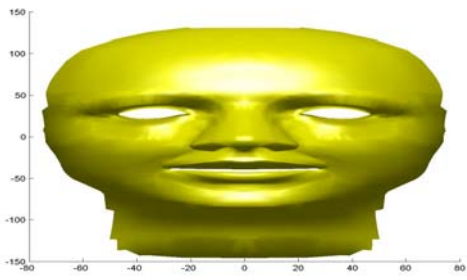
(b) second partial warp





(c) third partial warp

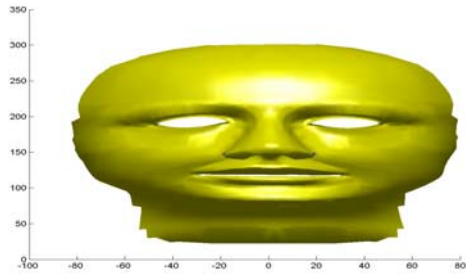
(e) fourth partial warp



(d) fifth partial warp

(f) sixth partial warp

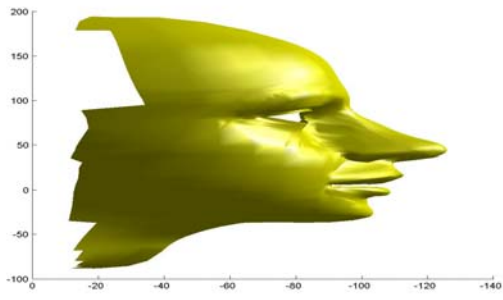




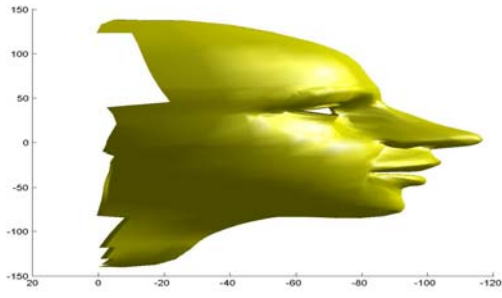
(g) sum of partial warps

(h) sum of rigid and non-rigid transformation

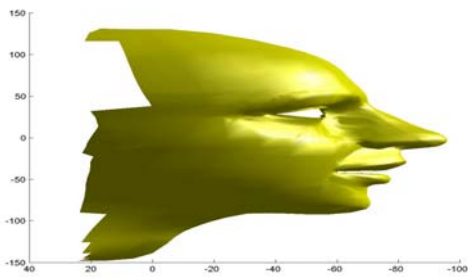
Figure 2 The components of TPS deformation (front views)



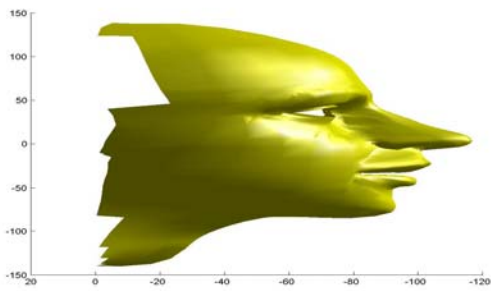
(a) first partial warp



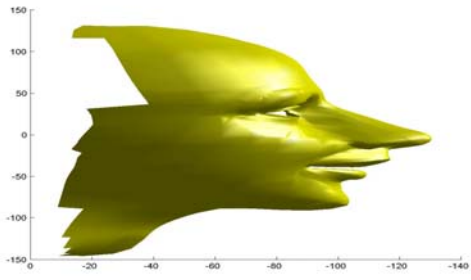
(b) second partial warp



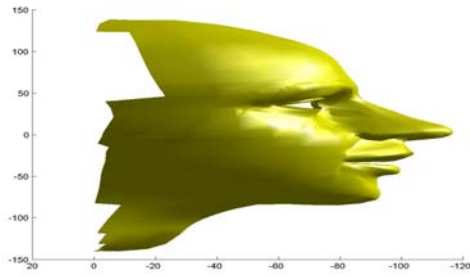
(c) third partial warp



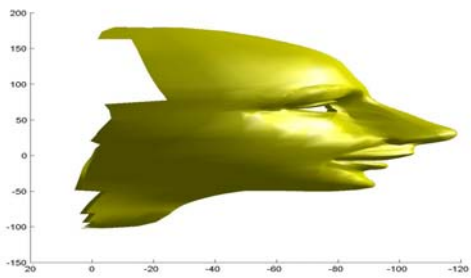
(e) fourth partial warp



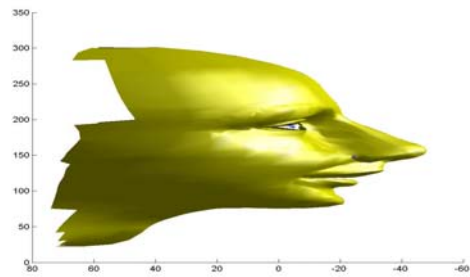
(d) fifth partial warp



(f) sixth partial warp

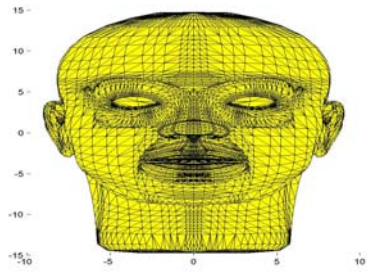
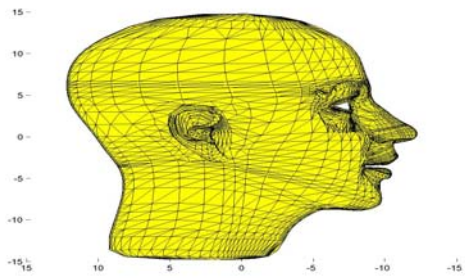


(g) sum of six partial warps



(h) sum of rigid and non-rigid transformation

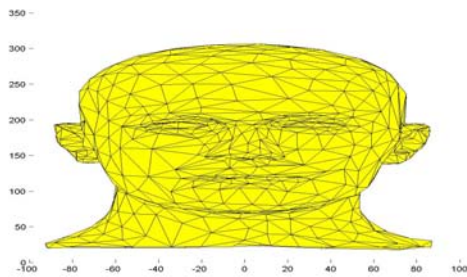
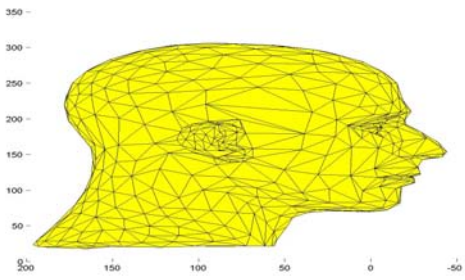
Figure 3 The components of TPS deformation (side views)



Source 3D model:

Vertices: 5828

Faces: 11370



Destination 3D model: Vertices: 689

Faces: 1355

