Using Dirichlet Free Form Deformation to Fit Deformable Models to Noisy 3-D Data

Slobodan Ilic and Pascal Fua*

Computer Graphics Laboratory(LIG)
EPFL, 1015 Lausanne, Switzerland
{Slobodan.Ilic, Pascal.Fua}@epfl.ch

Abstract. Free-form deformations (FFD) constitute an important geometric shape modification method that has been extensively investigated for computer animation and geometric modelling. In this work, we show that FFDs are also very effective to fit deformable models to the kind of noisy 3–D data that vision algorithms such as stereo tend to produce.
We advocate the use of Dirichlet Free Form Deformation (DFFD) instead of more conventional FFDs because they give us the ability to place control points at arbitrary locations rather than on a regular lattice, and thus much greater flexibility. We tested our approach on stereo data acquired from monocular video-sequences and show that it can be successfully used to reconstruct a complex object such as the whole head, including the neck and the ears, as opposed to the face only.

1 Introduction

Free-form deformations (FFDs) constitute an important approach to geometric shape modification that has been extensively investigated for computer animation and geometric modelling [15,3,9,2,13]. In the vision community, they have also been used to fit parametric models to medical data [17,1] or animation masks to semi-automatically extracted silhouette data [12]. In this work, we show that FFDs also are also very effective to fit deformable surface models to the kind of noisy 3–D data that vision algorithms such as stereo tend to produce.

The initial FFD approach [15] and all subsequent ones involve embedding the object model into a volume whose shape can be changed by moving a number of control points. Here, we take the embedded object to be a triangulated mesh and the embedding guarantees that each model vertex is influenced by the deformation of the control grid.

In this work, we advocate the use of Dirichlet Free Form Deformations (DFFDs) [13] because, unlike more conventional FFDs, they do not require the control points to lay on a regular rectangular grid. This is achieved by replacing the typical rectangular local coordinates by generalized natural neighbor coordinates, also known as Sibson coordinates[16]. That property give us the ability to place control points at arbitrary locations—that is, on the object, inside of it or outside—rather than on a regular lattice, and thus much greater flexibility. In particular, some of the control points can be important feature points that must be controlled in a specific way.

* This work was supported in part by the Swiss National Science Foundation under contract 21-57075.99

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2351, pp. 704–717, 2002.
© Springer-Verlag Berlin Heidelberg 2002

This flexibility of DFFDs has been put to good use in earlier face-modelling work [12]. That approach, however, takes silhouettes extracted from orthogonal images as input and does not allow for potential errors in the data. By contrast, we expect our input data to be both noisy and possibly incomplete. We have therefore developed a least-squares adjustment framework that lets us impose appropriate regularization constraints on the control mesh and obtain good fitting results even when using poor quality data.

We chose to demonstrate and evaluate our technique mainly in the context of head-modelling: More specifically, we use a technique we developed in earlier work [7] to compute motion parameters for heads seen in uncalibrated video-sequences and compute disparity maps from consecutive image pairs. We derive 3–D point clouds from these maps and fit a complete head model, including face, ears and neck, to them. Given the relative absence of texture one typically finds on faces and the uncalibrated nature of the sequences, the point clouds cannot be expected to be error-free and, yet, we obtain realistic models, whose geometry is correct and for which we can compute good texture maps by averaging the gray levels in all the images of a sequence.

We concentrate on head-modelling to demonstrate our technique because it is the application for which all the tools required to perform the complete reconstruction are available to us. We will, however, argue that the approach is generic and can be applied to any task for which deformable facetized models exist. In particular, we will show that we can also use our approach for high-resolution modelling of the human ear. We therefore view our contribution as the integration of a powerful shape deformation tool into a robust least-squares framework.

In the remainder of this paper, we first describe DFFDs in more detail. We then introduce our least-squares optimization framework and, finally, present results using both calibrated and uncalibrated image sequences.

2 Dirichlet Free Form Deformation(DFFD)

FFDs and their many extensions, such as rational FFD, extended FFD, direct FFD, NURBS based FFD and many others [15,3,2,8,11], are well known in the Computer Graphics community. They are used to globally deform a shape by embedding it into a volume controled by a small number of control points. The volume's shape is then deformed by moving the control points and the motion of the shape points is obtained by interpolating that of the control points.

In the original FFD, the control points must be placed on a regular lattice, which severely limits the range of allowable deformations that can be modeled. Most FFD extensions aim at overcoming this limitation, often by using a more sophisticated interpolant, but without addressing the basic problem that there is little flexibility in the positioning of the control points. By contrast, DFFDs [13] remove the requirement for regularly spaced control points that is the main conceptual geometric limitation of FFDs by replacing rectangular local coordinates by generalized natural neighbor coordinates, also known as Sibson coordinates, and using a generalized interpolant [4]. This idea comes from the data visualization community that relies on data interpolation and, thus, heavily depends on local coordinates.

2.1 Sibson Coordinates

All FFDs can be viewed as a data interpolation scheme where the interpolating function is a 3–D function specifying point displacements. The displacement is known at the control points and we want to interpolate it to the points of the object to deform. To this end, DFFDs replace standard rectangular or barycentric coordinates that constrain the control grid's shape by Sibson coordinates [16]: Given a set of points $P = \{P_0, P_1, \ldots, P_n\}$, these coordinates let us express any point within the convex hull of P as a linear combination of its *natural neighbors* within P. To find them, we compute a 3-D Delaunay triangulation and its Voronoi dual in which the cells are polyhedra.

A given point p may sit inside several several Delaunay spheres—that is, spheres circumscribed around Delaunay tetrahedra—-and the vertices of all these tetrahedra are taken to be the so-called natural neighbors. In other words, p is inside the sphere of influence of its neighbours. Let this set of natural neighbors be $Q_k \subset P$, where k is number of points inflencing p. The elements of Q_k are the natural neighbors of p and their influence is expressed by the Sibson coordinates u_i such that:

$$p = \sum_{i=0}^{k} u_i P_i, P_i \in Q_k$$

with $\sum_{i=0}^{k} u_i = 1$ and $u_i > 0$.

These coordinates are "natural" in the sense that points in P that are closer to the point p have greater influence on it, that is, the corresponding u_i is larger. They are computed using the partitioning of the space induced by the Voronoi diagram which is the dual of the Delaunay triangulation.

2.2 Introducing Deformations

As will be discussed above, our goal is to deform a *surface triangulation* using the vertices of a much sparser *control triangulation*, as our control points. We therefore take the control points to the vertices of the control triangulation complemented by the corners of the surface triangulation's bounding box, so as to guarantee that the whole object is contained in their convex hull.

Let P be this set of all control points and P_o the set of vertices of the surface triangulation. For each point $p \in P_o$, we find its natural neighbors $Q_k \subset P$ and the corresponding Sibson coordinates. This is referred to as freezing the control mesh to the object. Once computed, Sibson coordinates do not need to be changed when the object is deformed. When we move some of the control points from the set Q_k, the displacement of the model points is computed as follows:

$$\Delta p_o = \sum_{i=0}^{k} \Delta P_i u_i \tag{1}$$

where ΔP_i is displacements of control point from $P_i \in Q_k, i = 0, \ldots, k$. Finally, new object point position is computed as:

$$p'_o = p_o + \Delta p_o, \tag{2}$$

In short, the deformations are local and defined by the natural neighbors, which helps to improve the flexibility of the approach and the realism in the final results.

3 Least Squares Framework for DFFD Fitting

In this section, we introduce the framework we have developed to fit *surface models* such as the ones of Fig. 1 to noisy image data. Our goal is to deform the surface—without changing its topology, that is the connectivity of its vertices—so that it conforms to the image data. In this work data is made of 3–D points computed using stereo. In standard least-squares fashion, for each point $\mathbf{x}_i$, we write an observation equation of the form $d(\mathbf{x}_i, S) = obs_i + \epsilon_i$, where S is a state vector that defines the shape of the surface, d is the distance from the point to the surface and ϵ_i is the deviation from the model. In practice $d(\mathbf{x}, S)$ is taken to be the orthonormal distance of $\mathbf{x}$ to the closest surface triangulation facet. This results in $nobs$ such observations forming a vector

$$F(S) = [..., d(\mathbf{x}_i, S) - obs_i, ...]^t_{1 \leq i \leq nobs} \tag{3}$$

that we minimize in the least squares sense by minizing its square norm

$$\chi^2 = 1/2||F(S)||^2|| \ .$$

In theory we could take the parameter vector S to be the vector of all x, y, and z coordinates of the surface triangulation. However, because the image data is very noisy, we would have to impose a very strong regularization constraint. For example, we have tried to treat the surface triangulation as finite element mesh. Due to its great irregularity and its large number of vertices, we have found the fitting process to be very brittle and the smoothing coefficients difficult to adjust. This is why we chose to use the DFFD deformation approach instead.

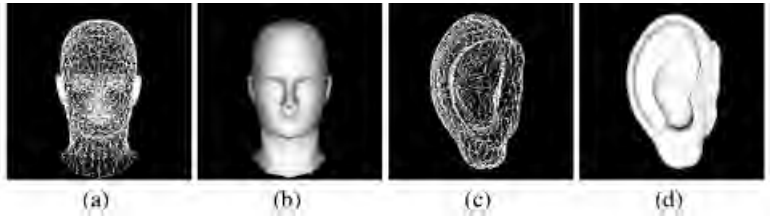


Fig. 1. (a,b) Complete head and neck animation model, shown as a wireframe and a shaded surface. (c,d) generic ear model

3.1 DFFD Parametrization

We therefore introduce *control triangulations* such as the ones of Fig. 2. Their vertices are points located at characteristic places on the human head or ear and defining their rough shapes and serve as DFFD control points. Some of these control points also are vertices of the surface model, while other are simply close to it and either inside or outside of it. This ability to place the control points is unique to DFFDs as compared to all other kinds of FFDs. The control triangulation facets will be used to introduce the regularization constraint discussed below. We tried to use several levels of resolutions

of control meshes, but we found that increasing the number of control points does not influence final results of the deformation. This means that keeping low number of the control points, as we did, greatly saves time for computation.

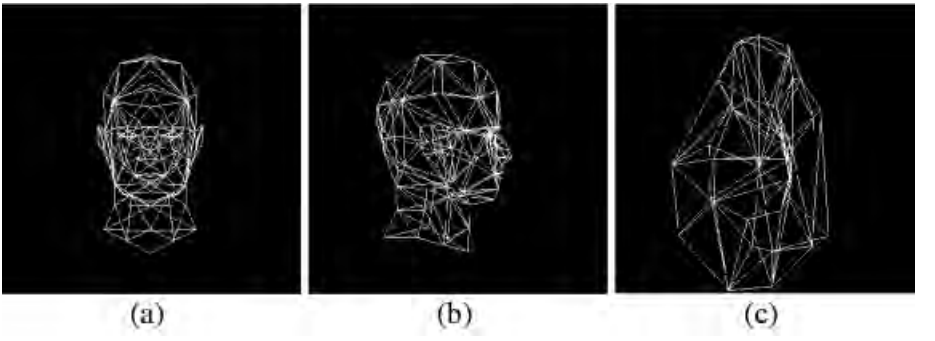


Fig. 2. Control triangulations for head (face (a) and profile (b)) and ear(c).

In our scheme, we take the state vector S to be the vector of 3-D displacements of the DFFD control points, which is very natural using the DFFD formalism: As discussed in Section 2.2, we first freeze the control mesh to the model vertices. This means that for each vertex on the model we compute the influence of certain subset of control points. Those influences are expressed in terms of the Sibson coordinates from Section 2.1 and allows us to express displacement of every model vertex as the linear combination of displacements of control points which influence them.

3.2 3D Observations

We use several sets of stereo pairs or triplets from the sequence of images of a given object as our input data such as those of Fig. 3(a,b,c). We then ran a simple correlation-based algorithm [5] to compute a disparity map for each pair or triplet and by turning each valid disparity value into a 3–D point. This resulted in a large cloud of 3–D points that form an extremely noisy and irregular sampling of the underlying global 3–D surface. To reduce the size of the cloud and begin eliminating outliers, we robustly fitted local surface patches to the raw 3–D points [6]. We then fed the centers of those patches, shown in Fig. 3(d), as input to our surface fitting algorithm.

The center of each patch can then be treated as an attractor. The easiest way to handle this is to model it as a spring attached to the mesh vertex closest to it. This, however, is inadequate if one wishes to use facets that are large enough so that attracting the vertices, as opposed to the surface point closest to the attractor, would cause unwarranted deformations of the mesh. This is especially important when using a sparse set of attractors. In our implementation, this is achieved by writing the observation equation as:

$$d_i^a = 0 + \epsilon_i \tag{4}$$

where d_i^a is the orthogonal distance of the attractor to the closest facet, whose nominal value is zero. It can be computed as a function of the *x*, *y*, and *z* coordinates of the vertices of the facet closest to the attractor.

Because some of the observations, derived on the way explained above, may be spurious, we weigh them to eliminate outliers. Weighting is done as the preprocessing step, before the real fitting is started. In each iteration after fitting is done, we recompute the attachments and also recompute the observation weight w_i and take it to be inversely proportional to the initial distance d_i of the data point to the surface triangulation. More specifically we compute w_i weight of the obs_i as:

$$w_i = exp(\frac{d_i}{\overline{d_i}}), 1 \leq i \leq n \tag{5}$$

where $\overline{d_i}$ is the median value of the d_i. In effect, we use $\overline{d_i}$ as an estimate of the noise variance and we discount the influence of points that are more than a few standard deviations away.

3.3 2D Observations

In our optimization framework besides 3D stereo observations we introduce also 2D observations. For each vertex *(xi, yi, zi)* of the *surface triangulation* whose 2D projection in image *j* is (u_i^j, v_i^j) is known, we can write two observation equations:

$$P_{ru}(x_i, y_i, z_i) = u_i^j + \epsilon_i^u$$

$$P_{rv}(x_i, y_i, z_i) = v_i^j + \epsilon_i^u$$

where P_{ru} and P_{rv} stand for the projection in *u* and *v*. In this way we do not need the explicit 3D position of these feature points, only their 2D image location.

3.4 Regularization

Because there are both noise and potential gaps in the image data, we found it necessary to introduce a regularization term comparable the one proposed in [1]. Since we start with a generic model, we expect the deformation between the initial shape and the original one to be smooth. This can be effectively enforced by preventing deformations at neighboring vertices of the control mesh to be too different. If the control points formed a continuous surface parametrized in terms of two parameters u and v, a natural choice would therefor be to take this term to be

$$\mathcal{E}_D = \sum_{s \in x,y,z} \mathcal{E}_{Ds} \tag{6}$$

$$\mathcal{E}_{Ds} = \iint \left(\frac{\partial}{\partial u} \delta_s(u,v) \right)^2 + \left(\frac{\partial}{\partial v} \delta s(u,v) \right)^2 du\, dv \ ,$$

where $\delta_s(u,v)$ stands for the displacement along the x,y or z coordinates at each point of this control surface. In fact, the control surface is a triangulated one and we only have deformation values at the vertices. We are therefore use a finite element approach to compute $\mathcal{E}_{Ds}$.

Stiffness Matrix for C^0 Triangular Elements. We write the $\mathcal{E}_{Ds}$ term of Eq. 6 as

$$\mathcal{E}_{Ds} = \lambda/2 \sum_{1 \leq j \leq n} \mathcal{E}_{D_s^j}$$

where $\mathcal{E}_{D_s^j}$ represents a summation over a facet j and λ is a regularization coefficient. In fact, we only know the deformations s_1^j,s_2^j and s_3^j at the vertices of the facet and we treat it as a C^0 triangular element. Over this specific facet, we write

$$\delta_s^j(u,v) = (1-u-v)s_1^j + us_2^j + vs_3^j \tag{7}$$

where $u, v \in [0,1]$ and $u+v < 1$. It is then easy to show that $\mathcal{E}_{D_s^j}$ is the quadratic term

$$[s_1 s_2 s_3]\, K_s^j \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad ,$$

where K_s^j is a 3×3 symmetric matrix that only depends on the shape of the triangle and, therefore, does not change during the optimization. These matrices can be summed into a global *stiffness matrix* K_s so that $\mathcal{E}_{Ds}$ becomes

$$s^t K_s s$$

where s stands for the vector of displacements at each vertex in one of the three coordinates. By summing these three terms, we obtain the final quadratic form or our complete regularization term

$$\mathcal{E}_D(S) = \frac{\lambda}{2} S^t K S \tag{8}$$

where S is the complete state vector.

Incorporating the Stiffness Matrix into the Least-Squares Framework. We use the Levenberg-Marquardt algorithm [14] to iteratively minimize the square-norm of the obervation vector $F(S)$ of Eq. 3. At each iteration, given the current state S, the algorithm attempts to find a step dS that minimizes

$$\chi^2(S+dS) = 1/2||F(S+dS)||^2 = 1/2F(S+dS)^t F(S+dS) \quad . \tag{9}$$

At the minimum, we should have

$$\begin{aligned} 0 &= \frac{\partial \chi^2}{\partial dS} \\ &= A^t F(S+dS) \\ &\approx A^t (F(S) + AdS) \quad , \end{aligned}$$

where A is the jacobian of F. dS is therefore taken to be the solution of

$$A^t A dS = -AF(S) \quad . \tag{10}$$

Adding a regularization term means that instead of minimizing simply the χ^2 term of Eq. 9, we minimize

$$\chi^2(S) + \mathcal{E}_D(S) = \frac{1}{2}||F(S+dS)||^2 + \frac{\lambda}{2}(S+dS)^t K(S+dS) \quad . \tag{11}$$

At each iteration, we therefore solve

$$\begin{aligned} 0 &= \frac{\partial \chi^2}{\partial dS} + \lambda K(S+dS) \\ &\approx A^t(F(S) + AdS) + \lambda K(S+dS) \quad . \end{aligned}$$

dS therefore becomes the solution of

$$(A^t A + \lambda K)dS = -A^t F(S) - \lambda K S \quad . \tag{12}$$

Note that solving Eq. 10 or 12 involves the same amount of computation so that our regularization scheme adds very little to the computational complexity of the algorithm. Note also that the proposed optimization scheme is a semi-implicit one very similar to the one proposed for the original active contours [10] and that greatly improves the convergence properties of the algorithm.

4 Results

We demonstrate and evaluate our technique mainly in the context of complete head, that is including face, ears and neck, from calibrated and uncalibrated video sequences. We show its flexibility by also using it to fit a generic ear model using a stereo-pair.

4.1 Calibrated Video Sequence

We first illustrate the effectiveness of our approach using relatively clean stereo data. We use the sequence of forty 512x512 images where some are depicted by the first row of Fig. 3. They were acquired with a video camera over a period of a few seconds by turning around the subject who was trying to stand still. Camera models were later computed using standard photogrammetric techniques at the Institute for Geodesy and Photogrammetry, ETH-Zürich. The centers of the local surface patches fitted to the raw stereo data shown on Fig. 3(d) are used as an input to our surface fitting algorithm.

We initialized the model by manually picking the five 2–D points overlaid in Fig. 3(b). We used them to compute a 4x4 rotation-translation matrix Rt such that five specific 3–D points on the generic model—outside corners of the eyes, corners of the mouth and tip of the nose—once multiplied by this matrix project as close as possible to the hand-picked 2–D location. Because these points are not coplanar, this guarantees that, when we multiply the generic model by this Rt matrix, we obtain an initial head model that is roughly aligned with the point cloud. We use it as the surface model that we deform using our DFFD approach, yielding the results shown in Fig. 3(e,f,g). In this case we did not use additional 2D observations provided manualy. The resulting shape corresponds to the real head shape except around the ears that stick out more from the real head then from the reconstructed model. The reason for this behavior is because of well

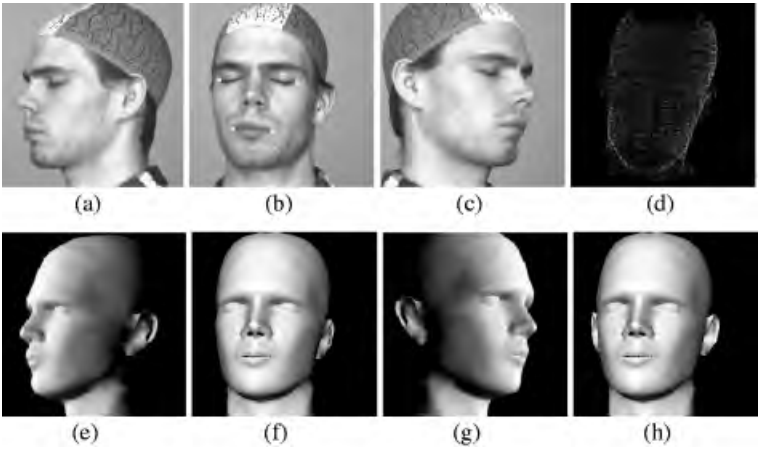


Fig. 3. Calibrated video sequence: (a,b,c) Three images chosen from the calibrated video sequence, courtesy of IGP, ETH Zürich. (d) Centers of local surface patches fitted to the raw stereo data. (e,f,g) Automatically obtained shaded model after fitting to the stereo data and projected using the same perspective transform as that of the images. (h) Shaded model after interactive correction of the ears.

known fact that minimizing orthonormal distance of data points to the closest surface triangulation facet may have difficulty in deforming the model into concave objects [18]. More accurate deformation can be obtained when 2D projected observation are included in the objective function what is used in the examples of uncalibrated video sequence Fig. 4. One of the advantages of DFFD is that this can be fixed manually very quickly and very intuitively by moving one control point per ear to produce the model of Fig. 3(h).

4.2 Uncalibrated Video Sequence

Fig. 4 depicts two examples of reconstruction from uncalibrated images. First row shows images from differnet image sequences: one image from the stereo pair of images Fig. 4(a) and three frames from other uncalibrated video sequence. In both cases, we had no calibration information about the camera or its motion. We therefore used a model-driven bundle-adjustment technique [7] to compute the relative motion and, thus, register the images. We then used the same technique as before [6] to derive the clouds of 3–D points depicted by Fig. 4(e,f,g,h).

Because we used fewer images and an automated self-calibration procedure as opposed to a sophisticated manual one, the resulting cloud of 3–D points is much noisier and harder to fit. Shaded models obtained after the fitting are dispicted on Fig. 4(i,j,k,l). Notice they shaded models are shown in the same projection as the corresponding images on Fig. 4(a,b,c,d). In Fig. 4(d) and (l) we overlay on both the original image and the shaded projection of the mask the outlines of the face. Note that they correspond to the outlines predicted from the recovered 3–D geometry, thus indicating a good fit. These models can also be reused to resynthesize textured images such as the ones of Fig. 5.

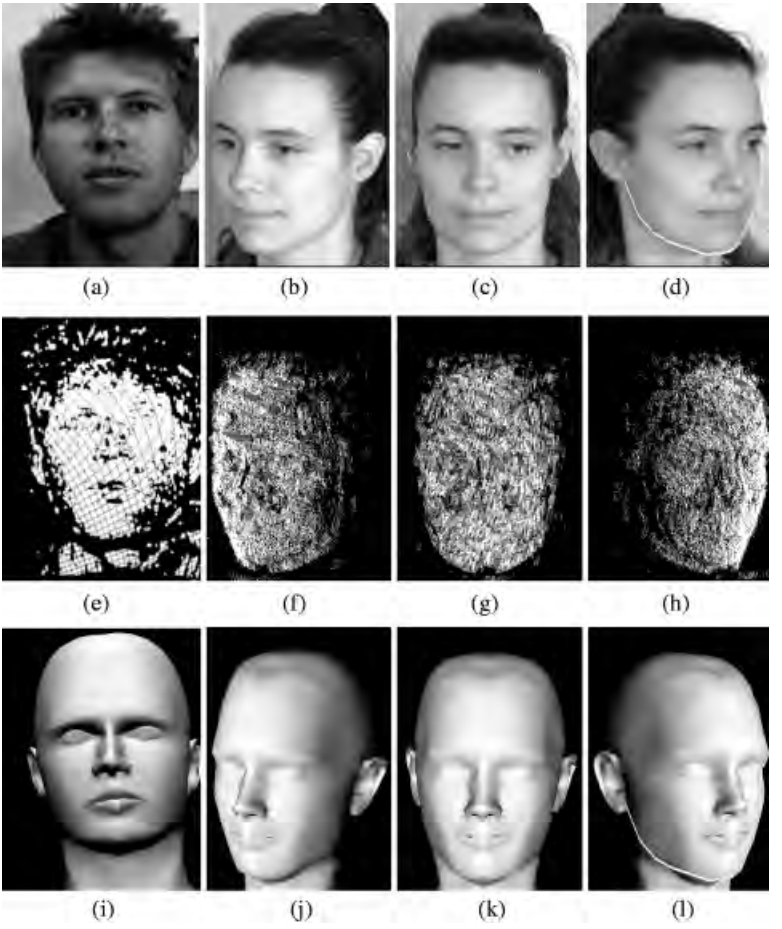


Fig. 4. Uncalibrated video sequence: (a) One image from the stereo pair of images with overlayed manualy provided 2D observations. (b,c,d) Three images from the uncalibrated video sequence out of eight images. On the image (c) 2D observations are despicted. (d) The image with overlaid face outline. (e,f,g,h) Centers of local surface patches fitted to the raw stereo data. (i,j,k,l) Automatically recovered shaded head model projected using the (a),(b),(c) and (d) image camera models. (l) Face outlines overlaid on the shaded projection in image (d).

We also animated generated models after the automatic fitting procedure using DFFD and produced complex facial expressions Fig. 5(i,j).

Least-square adjustment is applied in several iterations for the same value of regularization parameter λ. However, we tested how fitting to the uncalibrated data Fig. 4(b,c,d) is influenced by different choice of regularization parameter, in our case ranging from $\lambda = 1.0$ to $\lambda = 10$, and checked median error of the model to observations distance for certain number of iterations what is depicted on the Fig. 6(a). It is easy to see that final results do not depend on the choice of the regularization parameter since the median

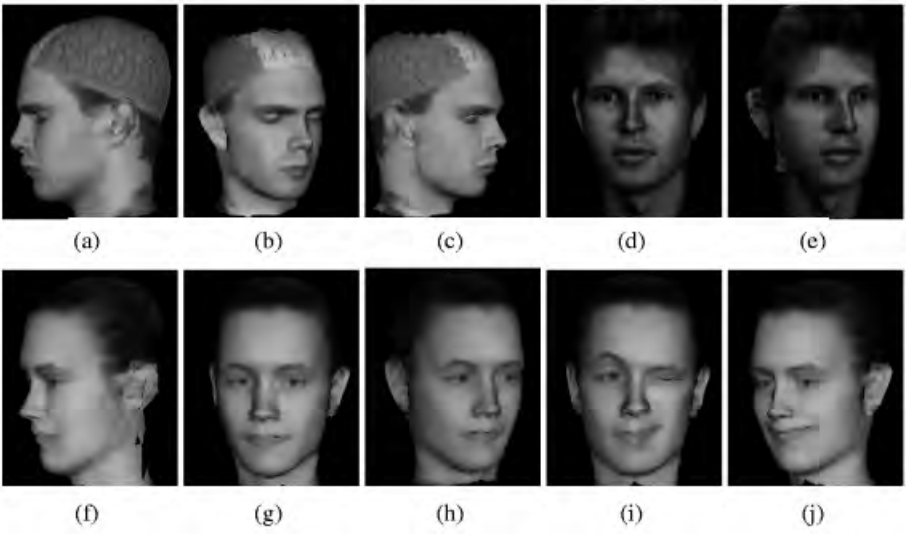
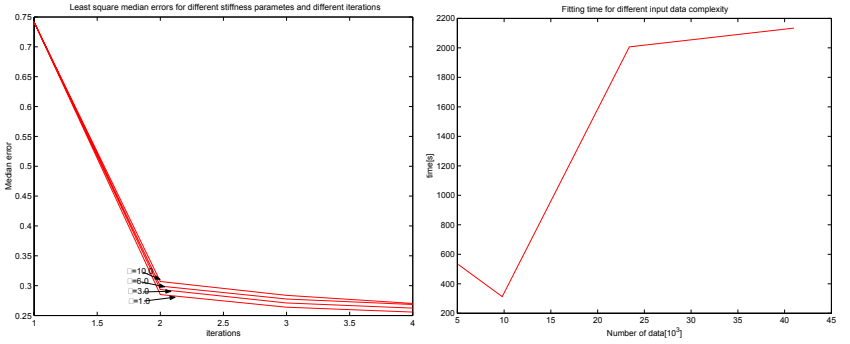


Fig. 5. Textured models we created from the calibrated video sequence:(a,b,c), from stereo pair (d,e) and from uncalibrated video sequence:(f,g,h); Animated model showing complex facial expressions(i,j)



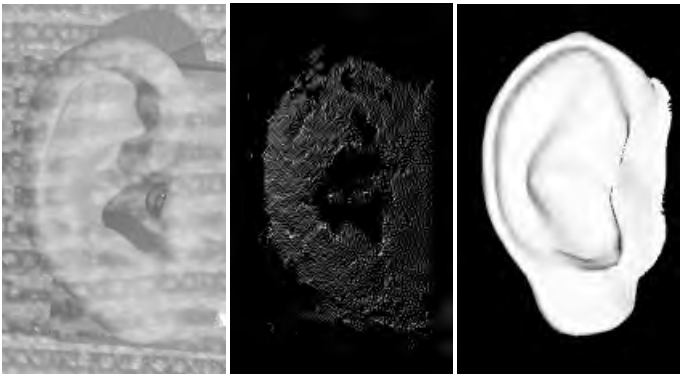
(a)

Fig. 6. (a) Least square median error for different regularization parameter λ ranging from *1.0* to *10.0* in respect to the number of iterations for video sequence whose three images are Fig. 4(b,c,d). (b) Fitting time in respect to the input data complexity

error of the model to observations distance does not greatly change with the increasing of the regularization parameter.

4.3 Ear Modelling

Finally, we will show that we are also able to model some other complex geometrical shapes such as human ear. The model we have is of very bad quality and we got it from the web Fig. 1(c,d). Data are produced from two images taken with the structure light Fig. 7(a). The control mesh is created manually Fig. 2(c), but this time there are no control points which exactly match some of the points on the model. Once again we demonstrated the power of DFFD based fitting to the noisy data. Results we obtained are again realistic and reliable. Deformed model on Fig. 7(c) is close to the ear shown on the Fig. 7(a). Using such complex shape to deform we can argue that the approach used is generic enough that it could be applied to any shape for which we have triangulated model.



(a) (b) (c)

Fig. 7. (a) Ear image "Courtesy of U. Maryland", (b) data cloud, (c) automatically obtained shaded deformed model shown in the same perspective as the ear on the image

4.4 Performance Measures

In our framework we use generic model of the human head including ears and neck from Fig. 1(a,b) which consists of 1396 vertices and 2756 facets. This generic model is used in all tests we performed to model heads. Control mesh is the one from Fig. 2(a, b). Generic ear model from Fig. 1(c,d) is built of 1167 vertices and 1620 facets, while its corresponding control mesh is shown on Fig. 2(c) has 46 vertces and 87 facets. System is tested on ancient Silicon Graphic Octane work station with R12000 prosessor working on 300MHz, and with 512Mb RAM memory.

The process starts with freezing the control mesh to the surface triangulation computing necessary Sibson coordinates. This is done once at the very begining and it is used for all input data. Freezing the control mesh of the head takes 65s. On the Fig. 6(b) is shown how the time for fitting depends on complexity of input data. Fitting time incerases with the complexity of the data, but also depends on its configuration. For this test the number of fitting steps is fixed to three iterations and the stiffness parameter is set to $\lambda = 1.0$. Input data of the size 10^3, are fitted for the shorter time then the one of the lower complexity $5 \cdot 10^3$, since its configuration is initially closer to the generic model, so the least-square minimization converges faster.

5 Conclusion

In this work, we proposed to use the powerful DFFD extension to the conventional FFD shape deformation approach to fit deformable surface models to noisy 3–D image data. DFFDs give us the ability to place control points at arbitrary locations rather than on a regular lattice, and thus much greater flexibility.We demonstrated the effectiveness and robustness of our technique in the context of complete head modeling. We also showed that, in fact, we can model any complex shape for which a deformable triangulated model exists.For the specific application we chose, DFFDs offer the added benefit that they can be used to animate the head models we create and to produce realistic human expressions. The proposed framework could thus be extended to head motion tracking.

In future work we intend to investigate the possibility of using DFFDs to deform implicit surfaces as opposed to the explicit ones we use here. In unrelated body-modelling work, we found that implicit surface formulations lend themselves extremely well to fitting to the kind of data because they allow us to define a distance function of data points to models that is both differentiable and computable without search. Combining both approaches should therefore produce an even more powerful modelling tool.

References

1. E. Bardinet, L.D. Cohen, and N. Ayache. A Parametric Deformable Model to Fit Unstructured 3D Data. *Computer Vision and Image Understanding*, 71(1):39–54, 1998.
2. Y. Chang and A. P. Rockwood. A Generalized de Casteljau Approach to 3D Free-form Deformation. *Computer Graphics, SIGGRAPH Proceedings*, pages 257–260, 1994.
3. S. Coquillart. Extended Free-Form Deformation: A sculpturing Tool for 3D Geometric Modeling. *Computer Graphics, SIGGRAPH Proceedings*, 24(4):187–196, 1990.
4. G. Farin. Surface over Dirichlet Tessellations. *Computer Aided Design*, 7:281–292, 1990.
5. P. Fua. A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features. *Machine Vision and Applications*, 6(1):35–49, Winter 1993.
6. P. Fua. From Multiple Stereo Views to Multiple 3–D Surfaces. *International Journal of Computer Vision*, 24(1):19–35, August 1997.
7. P. Fua. Regularized Bundle-Adjustment to Model Heads from Image Sequences without Calibration Data. *International Journal of Computer Vision*, 38(2):153–171, July 2000.
8. W.M. Hsu, J.F. Hugues, and H. Kaufman. Direct manipulation of Free-Form Deformations. *SIGGRAPH*, 26(2):177–184, 1992.
9. P. Kalra, A. Mangili, N. Magnenat Thalmann, and D. Thalmann. Simulation of Facial Muscle Actions Based on Rational Free Form Deformations. In *Eurographics*, 1992.

10. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
11. H.J. Lamousin and W.N. Waggenspack. NURBS-based Free-Form Deformations. *Computer Graphics and Applications*, 16(14):59–65, 1994.
12. W.S. Lee and N. Magnenat Thalmann. Fast Head Modelling for Animation. *Journal Image and Vision Computing*, 18(4), August 2000.
13. L. Moccozet and N. Magnenat-Thalmann. Dirichlet Free-Form Deformation and their Application to Hand Simulation. In *Computer Animation*, 1997.
14. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes, the Art of Scientific Computing*. Cambridge U. Press, Cambridge, MA, 1986.
15. T.W. Sederberg and S.R. Parry. Free-Form Deformation of Solid Geometric Models. *Computer Graphics, SIGGRAPH Proceedings*, 20(4), 1986.
16. R. Sibson. A vector identity for the Dirichlet Tessellation. In *Math. Proc. Cambridge Philos. Soc.*, pages 151–155, 1980.
17. R. Szeliski, S. Laval, and e. Matching anatomical surface with non-rigid deformations using octree-splines. *International Journal of Computer Vision*, 18(2):171–186, 1996.
18. C. Xu and J. Prince. Snakes, Shapes, and Gradient Vector Flow. *IEEE Transactions on Image Processing*, 7(3):359–369, March 1998.