Geometric Algebra for Pose Estimation and Surface Morphing in Human Motion Estimation

Bodo Rosenhahn and Reinhard Klette

University of Auckland (CITR),
Computer Science Department,
Private Bag 92019 Auckland, New Zealand
{bros028, r.klette}@cs.auckland.ac.nz

Abstract. We exploit properties of geometric algebras (GAs) to model the 2D-3D pose estimation problem for free-form surfaces which are coupled with kinematic chains. We further describe local and global surface morphing approaches with GA and combine them with the 2D-3D pose estimation problem. As an application of the presented approach, human motion estimation is considered. The estimated joint angles are used to deform surface patches to gain more realistic human models and therefore more accurate pose estimation results.

1 Introduction

A geometric algebra is a Clifford algebra with a specific geometric interpretation. The term geometric algebra was introduced by D. Hestenes, who applied Clifford Algebras on classical geometry and mechanics in the early 1960's [13]. Due to its properties, geometric algebra unifies mathematical systems which are of interest for computer graphics and computer vision. Examples of such systems are quaternions, dual-quaternions, Lie algebras, Lie groups, screw geometry in Euclidean, affine, projective or conformal geometry. In this contribution we show the applicability of conformal geometric algebra (CGA) [12, 15, 18] for solving the 2D-3D pose estimation problem. We use the example of human motion modeling and estimation [1, 4, 6, 9, 10] to show how it is possible to apply a unified approach to extend a basic scenario to a complex application by exploiting properties of CGA.

Pose estimation is a common task in computer vision. For a definition of the pose problem, we quote [11]: *By pose we mean the transformation needed to map an object model from its inherent coordinate system into agreement with the sensory data.* We deal with the *2D-3D pose estimation problem*: we assume an image of an object captured by a calibrated camera. Additionally to these 2D sensory data we also assume that a 3D representation of an object model is given. 2D-3D pose estimation means to specify a rigid motion (containing both 3D rotation and 3D translation) which fits the object model data with the 2D sensory data. The problem of 2D-3D pose estimation can be tackled from different points of view such as geometric or numerical perspectives. In the literature,

R. Klette and J. Žunić (Eds.): IWCIA 2004, LNCS 3322, pp. 583–596, 2004.

Euclidean, projective and conformal approaches can be found in combination with Kalman-filter, SVD, Newton-Raphson or gradient descent approaches. It is further crucial how objects are represented. The literature deals with point and line based representations, kinematic chains, higher order curves/surfaces, up to free-form contours or free-form surfaces. See [19] for an overview.

Geometric algebras can handle different object representations due to their multi-vector concepts and they allow to transform entities (rotation, translation, screw motion, reflection, refraction, etc.) with the help of the geometric product. It is therefore a useful tool to model geometric and numerical aspects in a unified language. For an introduction to geometric algebras, the reader is referred to [7, 8, 13, 18, 19, 21]. A brief list of homepages and research projects on Clifford algebras (with further links) can be found in [14].

For complex tasks, such as human motion estimation, a representation of the human body as a simple joint and skeleton model can be inadequate. Indeed the coupling of joints within a 2-parametric surface model gives a good initial guess about human motion (see Figure 1), but the human anatomy allows for much more degrees of freedom than, for example, three revolute joints for the shoulder, two for the elbow and two for the wrist. The human being is able to move the shoulder backward and forward, is able to raise and lower the shoulders to certain degrees, and if such additional degrees of freedom are not modeled, the pose results can become inaccurate or worthless. There is a need for an interaction of computer vision and computer graphics: a realistic model is needed to achieve accurate pose estimations, and, vice versa, accurate pose estimations help to refine a realistic model. In this contribution we show how to use geometric algebra for adding surface morphing and joint deformation approaches into surface modeling of a human being. We further show how to use this more complex, but more realistic model in the theory of CGA for pose estimation. In this contribution we are not dealing with the problem of recognizing or identifying

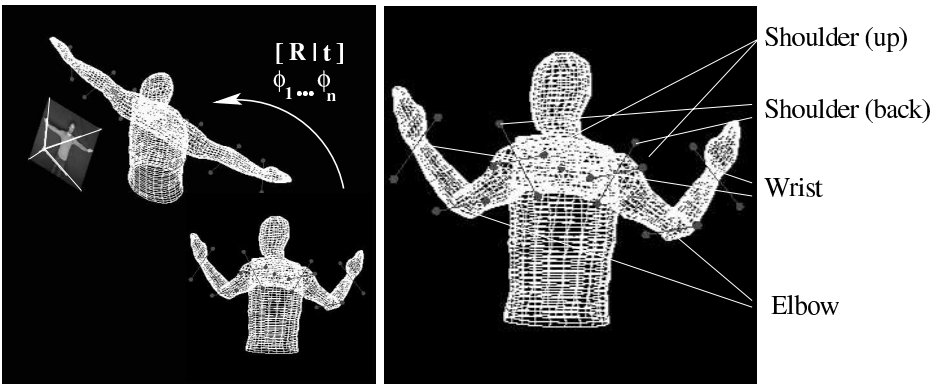

Fig. 1. Left: The pose scenario: the aim is to estimate the pose $\boldsymbol{R}$, $\boldsymbol{t}$ and the joint angles ϕ_i. Right: The names of the used joints

a moving human. Instead we are dealing with the estimation of object-specific parameters, like the pose and the joint angles.

We start this contribution with an introduction to silhouette based 2D-3D pose estimation of free-form contours and free-form surfaces. We want to quote Besl [3] for a definition: *A free-form surface has a well defined surface that is continuous almost everywhere except at vertices, edges and cusps.* Since we already model the pose problem and surface representation in CGA, we will introduce surface morphing in CGA in Section 3, so that we can directly use morphing concepts in the pose scenario without changing our main algorithms. Section 4 presents some experimental results and Section 5 ends with a brief discussion.

2 Preliminary Work

Clifford or geometric algebras [21] can be used to deal with geometric aspects of the pose problem. We only list a few properties which are important for our studies. The elements in geometric algebras are called multivectors which can be multiplied by using a geometric product. It allows a coordinate-free and symbolic representation. We use conformal geometric algebra (CGA) for modeling the pose problem. The CGA is build up on a conformal model which is coupled with a homogeneous model to deal with kinematics and projective geometry simultaneously. In conclusion, we deal with the Euclidean, kinematic and projective space in a uniform framework and can therefore cope with the pose problem within one theory. In the equations we will use the inner product $\cdot$, the outer product $\wedge$, the commutator $\underline{\times}$, and anticommutator $\overline{\times}$ product, which can be derived from the geometric product. Though we will also present equations formulated in conformal geometric algebra, we only explain these symbolically and want to refer to [19] for more detailed information.

2.1 Point Based Pose Estimation

For 2D-3D point based pose estimation we use constraint equations which compare 2D image points with 3D object points. Assume an image point $\boldsymbol{x}$ and the optical center $\boldsymbol{O}$. These define a 3D projection ray $\underline{\boldsymbol{L}}_x = \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})$ as a Plücker line [17]. The motor $\boldsymbol{M}$ is defined as exponential of a twist Ψ, $\boldsymbol{M} = \exp(-\frac{\theta}{2}\Psi)$, and formalizes the unknown rigid motion as a screw motion [17]. The motor $\boldsymbol{M}$ is applied on an object point $\underline{\boldsymbol{X}}$ as versor product, $\underline{\boldsymbol{X}}' = \boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}$, where $\widetilde{\boldsymbol{M}}$ represents the reverse of $\boldsymbol{M}$. Then the rigidly transformed object point $\underline{\boldsymbol{X}}'$ is compared with the reconstructed line $\underline{\boldsymbol{L}}_x$ by computing the error vector between the point and the line. This specifies a constraint equation in geometric algebra:

$$(\boldsymbol{M}\underline{\boldsymbol{X}}\widetilde{\boldsymbol{M}}) \underline{\times} (\mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x})) = 0$$

Note that we deal with a 3D formalization of the pose problem. The constraint equations can be solved by linearization (i.e., solving the equations for the twist-parameters which generate the screw motion) and by applying the Rodrigues formula for a reconstruction of the group action [17]. Iteration leads to a gradient

descent method in 3D space. This is presented in [19] in more detail where similar equations have been introduced to compare 3D points with 2D lines (3D planes), and 3D lines with 2D lines (3D planes).

Joints along a kinematic chain can be modeled as special screws with no pitch. In [19] we have shown that the twist then corresponds to a scaled Plücker line $\Psi = \theta \underline{\boldsymbol{L}}$ in 3D space, which gives the location of the general rotation. Because of this relation it is simple to move joints in space, and they can be transformed by a motor $\boldsymbol{M}$ in a similar way $\Psi' = \boldsymbol{M}\Psi\widetilde{\boldsymbol{M}}$ such as plain points.

2.2 Contour-Based Pose Estimation

We now model free-form contours and discuss their role for solving the pose problem. The pose estimation algorithm for surface models (as introduced in this paper) relies onto a contour based method. Therefore, a brief recapitulation of [19] on contour based pose estimation is of importance. The main idea is to interpret a 1-parametric 3D closed curve as three separate 1D signals which represent the projections of the curve along the x, y and z axis, respectively. Since the curve is assumed to be closed, the signals are periodic and can be analyzed by applying a 1D discrete Fourier transform (1D-DFT). The inverse discrete Fourier transform (1D-IDFT) enables us to reconstruct low-pass approximations of each signal. Subject to the sampling theorem, this leads to the representation of the 1-parametric 3D curve $C(\phi)$ as follows:

$$C(\phi) = \sum_{m=1}^{3} \sum_{k=-N}^{N} \boldsymbol{p}_k^m \exp\left(\frac{2\pi k\phi}{2N+1}\boldsymbol{l}_m\right)$$

The parameter m represents each dimension and the vectors $\boldsymbol{p}_k^m$ are phase vectors obtained from the 1D-DFT acting on dimension m. In this equation we have replaced the imaginary unit $i = \sqrt{-1}$ by three different rotation planes, represented by the bivectors $\boldsymbol{l_i}$, with $\boldsymbol{l_i}^2 = -1$. Using only a low-index subset of the Fourier coefficients results in a low-pass approximation of the object model which is used to regularize the pose estimation algorithm. For pose estimation, this model is then combined with a version of an ICP-algorithm [23].

2.3 Silhouette-Based Pose Estimation of Free-Form Surfaces

We assume a two-parametric surface [5] of the form

$$F(\phi_1, \phi_2) = \sum_{i=1}^{3} f^i(\phi_1, \phi_2)\mathbf{e}_i$$

with three 2D functions $f^i(\phi_1, \phi_2) : \mathbb{R}^2 \to \mathbb{R}$ acting on the different Euclidean base vectors $\mathbf{e}_i$ $(i = 1, \ldots, 3)$. A two-parametric surface allows that two independent parameters ϕ_1 and ϕ_2 are used for sampling a 2D surface in 3D space. For a discrete number of sampled points, $f^i_{n_1,n_2}$, $(n_1 \in [-N_1, N_1]; n_2 \in [-N_2, N_2]; N_1, N_2 \in \mathbb{N}, i = 1, \ldots, 3)$ on the surface, we can then interpolate the surface by

using a 2D discrete Fourier transform (2D-DFT), and we apply an inverse 2D discrete Fourier transform (2D-IDFT) for each base vector separately. Subject to the sampling theorem, the surface can be written as a Fourier representation

$$F(\phi_1,\phi_2)=\sum_{i=1}^{3}\sum_{k_1=-N_1}^{N_1}\sum_{k_2=-N_2}^{N_2}\boldsymbol{p}^i_{k_1,k_2}\exp\left(\frac{2\pi k_1\phi_1}{2N_1+1}\boldsymbol{l_i}\right)\exp\left(\frac{2\pi k_2\phi_2}{2N_2+1}\boldsymbol{l_i}\right)$$

The complex Fourier coefficients are contained in the vectors $\boldsymbol{p}^i_{k_1,k_2}$ that lie in the plane spanned by $\boldsymbol{l_i}$. We will also call them phase vectors. These vectors can be obtained by a 2D-DFT of the sample points $f^i_{n_1,n_2}$ on the surface. We now continue with the algorithm for silhouette-based pose estimation of surface models.

We assume a properly extracted image contour of our object (i.e., in a frame of the sequence). To compare points on the image silhouette with the 3D surface model, we consider rim points on the surface (i.e., which are on an occluding boundary of the object). This means we work with the 3D silhouette of the surface model with respect to the camera. To ensure this, we project the 3D surface on a virtual image. Then the contour is calculated and from the image contour the 3D silhouette of the surface model is reconstructed. The contour model is then applied within the contour-based pose estimation algorithm. Since aspects of the surface model are changing during ICP-cycles, a new silhouette will be estimated after each cycle to deal with occlusions within the surface model. The algorithm for pose estimation of surface models is summarized in Figure 2, and it is discussed in [20] in more detail.

2.4 Human Motion Estimation

We continue with our way how to couple kinematic chains within a surface model. Then we present a pose estimation algorithm which estimates the pose and angle configurations simultaneously.

A surface is given in terms of three 2-parametric functions with respect to the parameters ϕ_1 and ϕ_2. Furthermore, we assume a set of joints J_i. By using an extra function $\mathcal{J}(\phi_1,\phi_2)\rightarrow[J_i|J_i:i\text{th joint}]$, we are able to give every node a joint list along the kinematic chain. Note that we use $[,]$ and not $\{,\}$, since

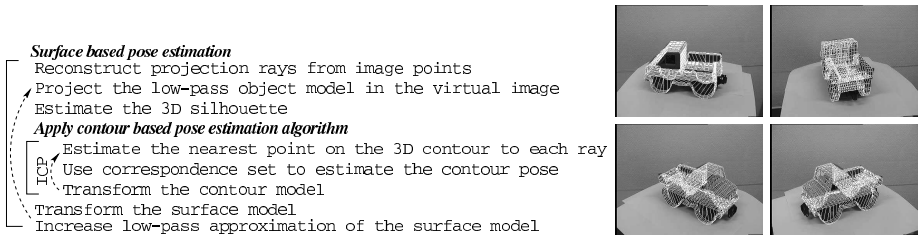

Fig. 2. Left: The algorithm for pose estimation of surface models. Right: A few example images of a tracked car model on a turn-table

the joints are given as an ordered sequence along the kinematic chain. Since the arms contain two kinematic chains (i.e., for the left and right arm, separately), we introduce a further index to separate the joints on the left arm from the ones on the right arm. The joints themselves are represented as objects in an extra field in form of a look-up table, and their parameters can be accessed immediately from the joint index numbers. Furthermore, it is possible to transform the location of the joints in space (as clarified in Section 2.1). For pose estimation of a point $\underline{\boldsymbol{X}}_{n,i_n}$ attached to the nth joint along the kinematic chain, we generate constraint equations of the form

$$(\boldsymbol{M}(\boldsymbol{M}_1 \dots \boldsymbol{M}_n \underline{\boldsymbol{X}}_{n,i_n} \widetilde{\boldsymbol{M}_n} \dots \widetilde{\boldsymbol{M}_1}) \widetilde{\boldsymbol{M}}) \underline{\times} \mathbf{e} \wedge (\boldsymbol{O} \wedge \boldsymbol{x}_{n,i_n}) = 0$$

To solve a set of such constraint equations we linearize the motor $\boldsymbol{M}$ with respect to the unknown twist Ψ, and the motors $\boldsymbol{M}_i$ with respect to the unknown angles θ_i. The twists Ψ_i are known a priori.

The basic pose estimation algorithm is visualized in Figure 3. We start with simple image processing steps to gain the silhouette information of the person by using a color threshold and a Laplace operator. Then we project the surface mesh in a virtual image and estimate its 3D contour. Each point on the 3D contour carries a given joint index. Then we estimate the correspondences by using an ICP-algorithm, generate the system of equations, solve them, transform the object and its joints, and iterate this procedure. During iteration we start with a low-pass object representation and refine it by using higher frequencies. This helps to avoid local minima during iteration.

First results of the algorithm are shown on the left of Figure 4. The figure contains two pose results; on each quadrant it shows the original image and overlaid the projected 3D pose. The other two images show the estimated joint angles in a virtual environment to visualize the error between the ground truth and the estimated pose. The tracked image sequence contains 200 images. In this sequence we use just three joints on each arm and neglect the shoulder (back) joint. The right diagram of Figure 4 shows the estimated angles of the joints during the image sequence. The angles can easily be identified with the sequence. Since the movement of the body is continuous, the estimated curves are also "relatively smooth".

3 More Realistic Human Models

We are interested in using skinning approaches to model "more realistic" human motions during pose estimation. The hypothesis at this stage is that the more accurate the human being is modeled, the more accurate the pose result will be. This requires two things in a first step, namely, to model joint transformations and surface deformations during joint motions. So far, skin and muscle deformations are not yet modeled. Take, for example, the shoulder joint: If the shoulder is moving, muscles are tensed and the skin is morphing. The task is to model such deformations dependent on the joint angle. To achieve surface morphing,

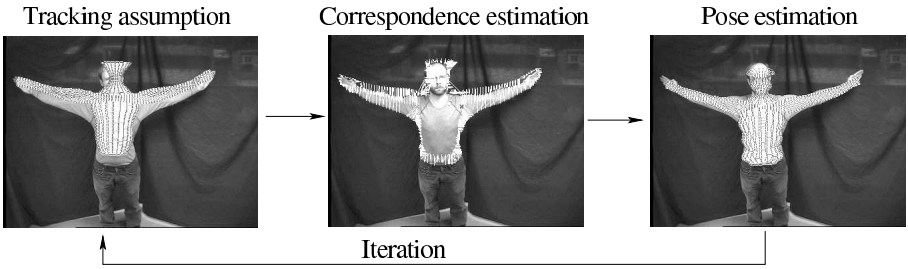

Fig. 3. The basic algorithm: Iterative correspondence and pose estimation

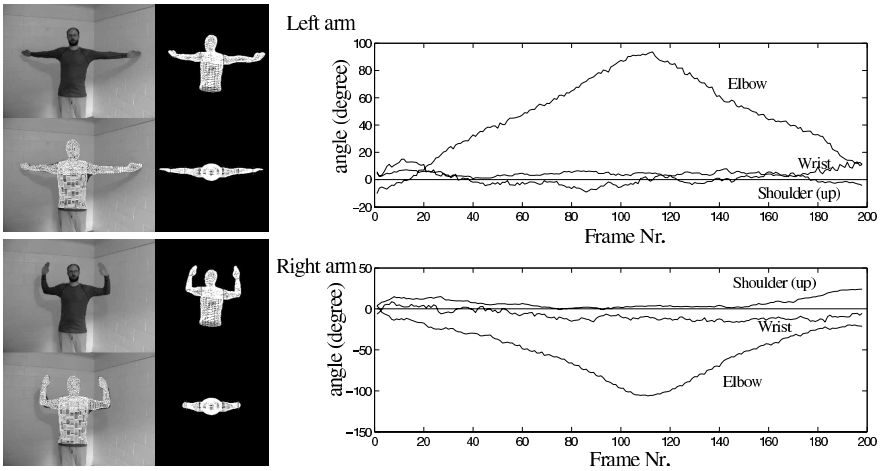

Fig. 4. Left: First pose results with a 6 DOF kinematic chain. Right: Angles of the left and right arm during the tracked image sequence

we will express two known approaches for surface morphing in CGA. These are a global approach and a local approach; the latter one uses radial basis functions.

3.1 Joint Motions

Joints along the kinematic chain can be modeled as screws with no pitch. We already have shown that its twist then corresponds to a scaled Plücker line $\Psi = \theta \underline{\boldsymbol{L}}$ in space, which gives the location of the general rotation. Because of this relation it is simple to move joints in space, and they can be transformed by a motor $\boldsymbol{M}$ in a similar manner $\Psi' = \boldsymbol{M}\Psi\widetilde{\boldsymbol{M}}$ as 3D points. To interpolate between two given joint locations Ψ and Ψ', we can use a motor $\boldsymbol{M} = \exp(-\frac{\rho}{2}\Xi)$, with the property that for $\rho = 2\pi$ it holds $\boldsymbol{M}\Psi\widetilde{\boldsymbol{M}} = \Psi'$. Then $\boldsymbol{M}_t = \exp(-\frac{t2\pi}{2}\Xi)$ for $t \in [0 \ldots 1]$ leads to a motor which interpolates the joint location Ψ via $\boldsymbol{M}_t\Psi\widetilde{\boldsymbol{M}}_t$ towards Ψ'.

3.2 Global Surface Interpolation

We assume two free-form surfaces given as two-parametric functions in CGA as follows:

$$F_1(\phi_1,\phi_2)=\sum_{i=1}^{3} f_1^i(\phi_1,\phi_2)\mathbf{e}_i \quad \text{and} \quad F_2(\phi_1,\phi_2)=\sum_{i=1}^{3} f_2^i(\phi_1,\phi_2)\mathbf{e}_i$$

For a given parameter $t \in [0 \dots 1]$, the surfaces can be linearly interpolated by evaluating

$$F_t(\phi_1,\phi_2)=\left(\sum_{i=1}^{3} f_1^i(\phi_1,\phi_2)\mathbf{e}_i\right) t+\left(\sum_{i=1}^{3} f_2^i(\phi_1,\phi_2)\mathbf{e}_i\right)(1-t)$$

We perform a linear interpolation along the nodes, and this results in the following:

$$F_t(\phi_1,\phi_2)=\begin{cases}\sum_{i=1}^{3} f_1^i(\phi_1,\phi_2)\mathbf{e}_i = F_1(\phi_1,\phi_2)\ , \text{for}\ \ t=1\\ \sum_{i=1}^{3} f_2^i(\phi_1,\phi_2)\mathbf{e}_i = F_2(\phi_1,\phi_2)\ , \text{for}\ \ t=0\end{cases}$$

Figure 5 shows examples of morphing a male into a female torso. Note that we are only morphing surfaces with known and predefined topology. This means that we have knowledge about the correspondences between the surfaces, and morphing is realized by interpolating the corresponding nodes on the mesh.

The linear interpolation can be generalized by using an arbitrary function $\omega(t)$ with the property

$$\omega(t)=\begin{cases}0\ , \text{for}\ \ t=1\\ 1\ , \text{for}\ \ t=0.\end{cases}$$

Then, an interpolation is still possible by using

$$F_t(\phi_1,\phi_2)=\left(\sum_{i=1}^{3} f_1^i(\phi_1,\phi_2)\mathbf{e}_i\right)\omega(t)+\left(\sum_{i=1}^{3} f_2^i(\phi_1,\phi_2)\mathbf{e}_i\right)(1-\omega(t))$$

Figure 6 shows different possible functions which result in different interpolation dynamics. and Figure 7 illustrates these different dynamics: using the square root function for weighting leads to a faster morphing at the beginning, which slows down at the end, whereas squared weighting leads to a slower start and a faster ending. Therefore, we can use non-linear weighting functions to gain a natural morphing behavior dependent on the joint dynamics.

Figure 8 shows a comparison of the non-modified model (right) with a morphed joint-transformed model (left). As it can be seen, the shoulder joint is moving down and in-wards during motion, and, simultaneously, does the surface of the shoulder part morph. The amount of morphing and joint transformation is steered through the angle of the shoulder (up) joint (left and right, respectively). As can be seen, the left motion appears more natural than the right one.

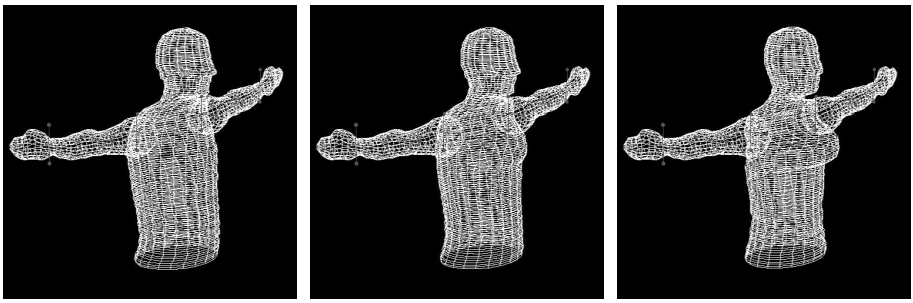

Fig. 5. Morphing of a male into a female torso

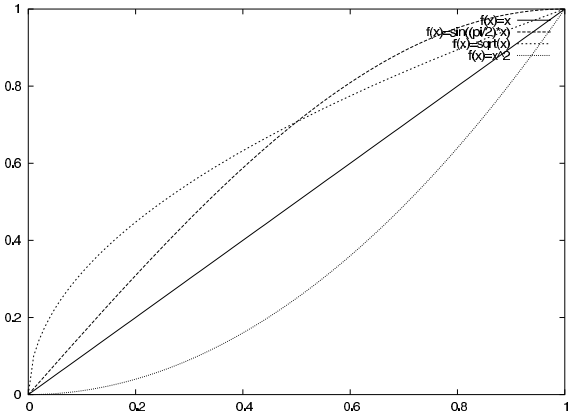

Fig. 6. Different weighting functions during interpolation

3.3 Local Surface Morphing

The use of radial basis functions for local morphing is common practice for modeling facial expressions.

The basic idea is as follows: we move a node on the surface mesh, and we move the neighboring nodes in a similar manner, but decreasingly with increasing distance to the initiating node. It is further possible to deform the radial basis function to allow a realistic morphing in the presence of bones or ligaments. The classic equation for a radial basis function is

$$r(x,y) = \exp\left(-\frac{(x-c_x)^2}{r_x}\right)\exp\left(-\frac{(y-c_y)^2}{r_y}\right)$$

with the centre (c_x, c_y) and the radius (r_x, r_y). The values $(c_x, c_y) = (0,0)$ and $(r_x, r_y) = (1,1)$ lead to the classic Gaussian form as shown on the left of Figure 9.

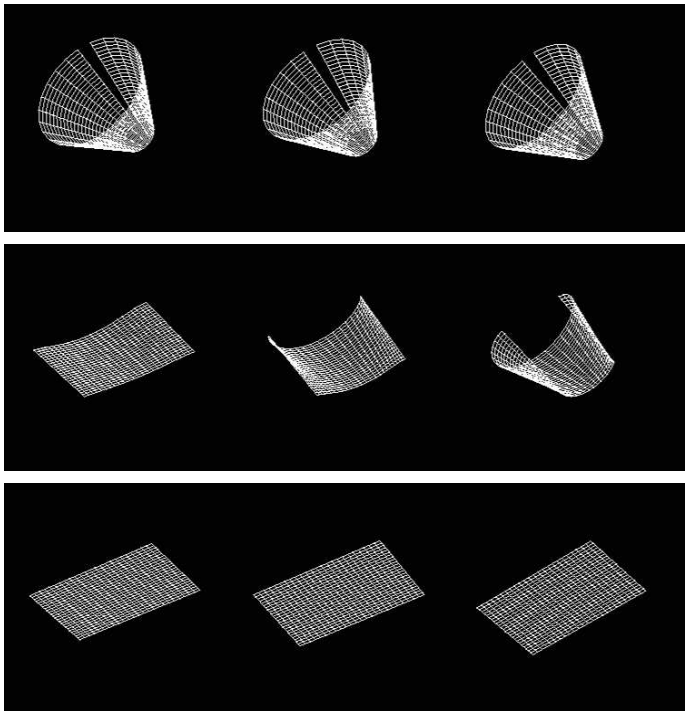

Fig. 7. Different interpolations. Left: square root, middle: linear and right: square interpolation

The coupling of a radial basis function with the surface mesh leads to

$$F_R(\phi_1, \phi_2) = \boldsymbol{T}_r \left(\sum_{i=1}^{3} f^i(\phi_1, \phi_2) \mathbf{e}_i \right) \widetilde{\boldsymbol{T}_r}$$

with

$$\boldsymbol{T}_r = 1 + \frac{\mathbf{e}\boldsymbol{t}}{2} \quad \text{for} \quad \boldsymbol{t} = \mathbf{e}_3 r(\phi_1, \phi_2)$$

$\boldsymbol{T}$ is a translator which translates a node along an orientation, and the amount of translation is steered through the value of the radial basis function. Note that $\boldsymbol{T}$ is dependent on (ϕ_1, ϕ_2) and different for each node. In this case we model a deformation along the $\mathbf{e}_3$-axes, but it can be any orientation, and the Gaussian function can be arbitrarily scaled. For our human model we additionally steer the amount of morphing through the joint angle θ_1 of the shoulder (back) joint. This means, if the shoulder is not moving forwards or backwards, we will not have any morphing, but the more the arm is moving, the larger will be the amount of morphing. With a scaling parameter λ, the morphing translator is completely given as

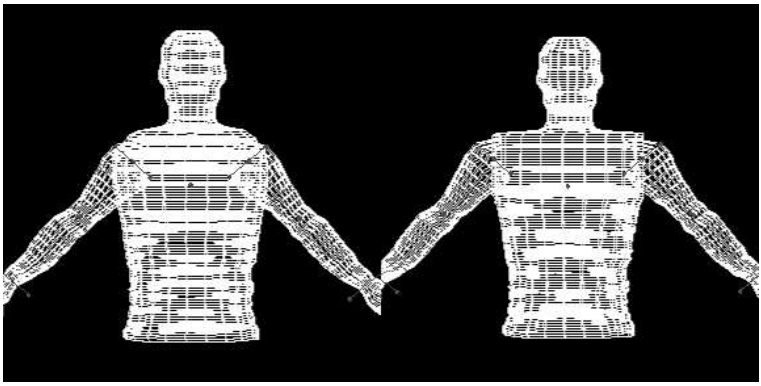

Fig. 8. Different arm positions of the morphed joint-transformed model (left) and non-modified model (right)

$$\boldsymbol{T}_r = 1 + \frac{\mathbf{e}\boldsymbol{t}}{2} \quad \text{with} \quad \boldsymbol{t} = \lambda\theta_1 r(\phi_1, \phi_2)\mathbf{e}_3$$

In contrast to global morphing, local approaches have the advantage that they can more easily be used in the context of multiple morphing patches. For example, simultaneous shoulder morphing up/down and forwards/backwards is hardly possible with a global approach, but simple with a local one.

Figure 9 shows on the left a typical radial basis function to realize local surface morphing. The images on the right show a double morphing on the shoulder: moving the arms up or down and forwards or backwards leads to a suited deformation of the shoulder patch and a similar motion of the joint locations.

4 Experiments

This section presents a few experiments using global and combined (i.e., global and local) morphing methods. The implementation is done in `C++` on a standard Linux PC (2.4 GHz) and we need 100ms for each frame, including image processing, pose estimation and surface morphing.

The morphing effect during an image sequence can be seen in Figure 10. A person is moving his arms down, and as it can be seen in the left images, the shoulder is moving downwards, too. The pose result for the morphed/joint transformed model is shown in the middle image, and the result for the non-modified model is shown in the right image. As shown, the matching result of the morphed/joint transformed object model is much better.

Figure 11 shows example images for a double-morphing in the shoulder area. The shoulder is morphing downwards when the arms are moving down, and forwards when the arms are moving forward. Both motions can occur simultaneously, leading to a more realistic motion behavior then using rigid joints. Since

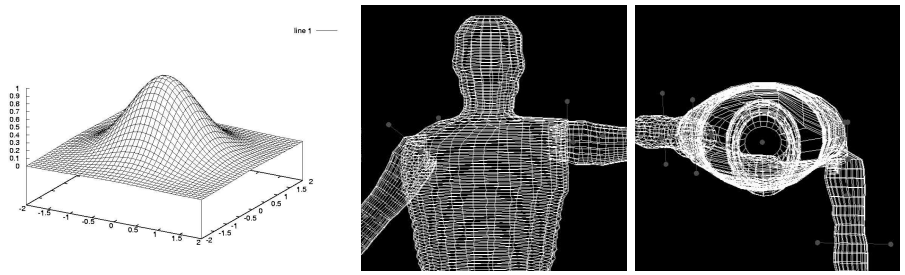

Fig. 9. Left: A 2D-radial basis function. Right: Double surface morphing on the shoulder

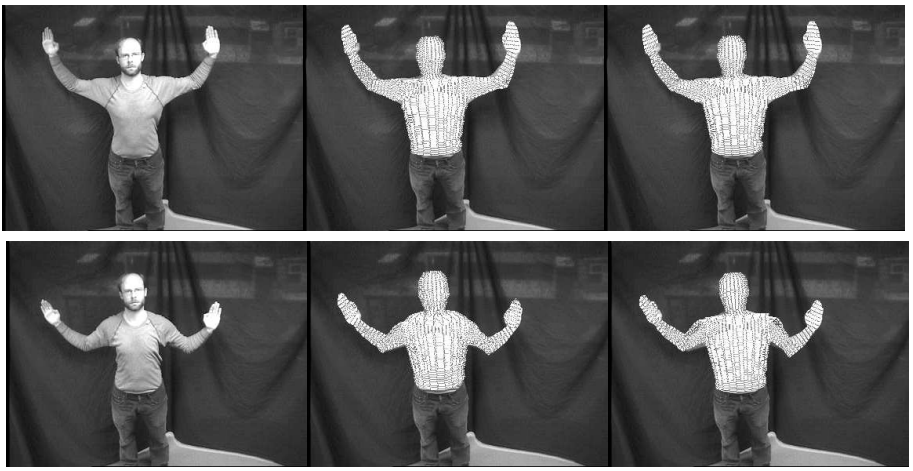

Fig. 10. Comparison of pose results for the morphed/joint transformed model and the non-modified model

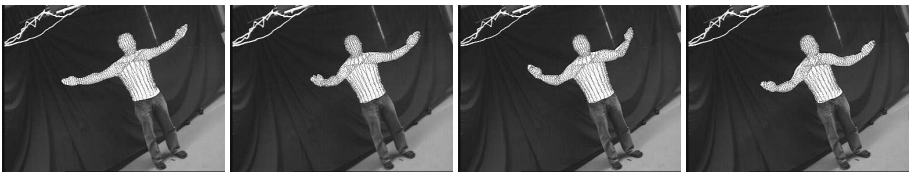

Fig. 11. Pose results using two shoulder morphing operators

the morphing effect can be seen more easily during the whole sequence (in contrast to a few snap shots), the reader is invited to see the sequence at `http://www.citr.auckland.ac.nz/~bodo/DMorph.mpg`.

5 Discussion

This contribution presents an embedding of global and local morphing techniques in CGA. The motivation for this paper was to show the applicability of geometric algebras to model complex geometric problems. At first we recalled the 2D-3D pose estimation problem for free-form surface models. Then we extended a surface model by joints and used the human motion estimation problem as an example scenario for discussing CGA. Due to the complexity of human motions, we introduced local and global morphing approaches in CGA to gain a more realistic human model. The amount of deformation is steered through a related joint angle. It is further possible to deform (e.g., the shoulder patch) even with non-linear weighting functions or as coupled local and global deformation. The experiments showed the usefulness of this approach to obtain more accurate tracking results for human motions.

Acknowledgments

This work has been supported by the EC Grant IST-2001-3422 (VISATEC) and by the DFG grants RO 2497/1-1 and RO 2497/1-2.

References

1. Allen B., Curless B. and Popovic Z. Articulated body deformation from range scan data. In *Proceedings 29th Annual Conf. Computer Graphics and Interactive Techniques*, San Antonio, Texas, pp. 612 - 619, 2002.
2. Arbter K. and Burkhardt H. Ein Fourier-Verfahren zur Bestimmung von Merkmalen und Schätzung der Lageparameter ebener Raumkurven. *Informationstechnik*, Vol. 33, No. 1, pp. 19-26, 1991.
3. Besl P.J. The free-form surface matching problem. *Machine Vision for Three-Dimensional Scenes, Freemann H. (Ed.)*, pp. 25-71, Academic Press, 1990.
4. Bregler C. and Malik J. Tracking people with twists and exponential maps. *Conf. on Computer Vision and Pattern Recognition*, Santa Barbara, California, pp. 8-15, 1998.
5. Campbell R.J. and Flynn P.J. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, Vol. 81, pp. 166-210, 2001.
6. Chadwick J.E., Haumann D.R. and Parent R.E. Layered construction for deformable animated characters *Computer Graphics*, Vol. 23, No. 3, pp. 243-252, 1989.
7. Dorst L. The inner products of geometric algebra. In *Applied Geometric Algebras for Computer Science and Engineering, Dorst L., Doran C. and Lasenby J. (Eds.)*, Birkhäuser Verlag, pp. 35-46, 2001.
8. Dorst L. Honing geometric algebra for its use in the computer sciences. In [21], pp. 127-152, 2001.
9. Fua P., Plänkers R., and Thalmann D. Tracking and modeling people in video sequences. *Computer Vision and Image Understanding*, Vol. 81, No. 3, pp.285-302, March 2001.

10. Gavrilla D.M. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, Vol. 73 No. 1, pp. 82-92, 1999.
11. Grimson W. E. L. *Object Recognition by Computer.* The MIT Press, Cambridge, MA, 1990.
12. Hestenes D., Li H. and Rockwood A. New algebraic tools for classical geometry. In [21], pp. 3-23, 2001.
13. Hestenes D. and Sobczyk G. *Clifford Algebra to Geometric Calculus.* D. Reidel Publ. Comp., Dordrecht, 1984.
14. Homepages Clifford (geometric) algebra
`http://www.ks.informatik.uni-kiel.de`
`http://www.clifford.org/`
`http://modelingnts.la.asu.edu/GC_R&D.html`
`http://www.mrao.cam.ac.uk/~clifford/`
`http://www.science.uva.nl/ga/`
`http://clifford.physik.uni-konstanz.de/~fauser/P_cl_people.shtml`
`http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Clifford.html`
15. Li H., Hestenes D. and Rockwood A. Generalized homogeneous coordinates for computational geometry. In [21], pp. 27-52, 2001.
16. Mikic I., Trivedi M, Hunter E, and Cosman P. Human body model acquisition and tracking using voxel data *Computer Vision* , Vol. 53, Nr. 3, pp. 199–223, 2003.
17. Murray R.M., Li Z. and Sastry S.S. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.
18. Perwass C. and Hildenbrand D. Aspects of Geometric Algebra in Euclidean, Projective and Conformal Space. An Introductory Tutorial. *Technical Report 0310, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2003.
19. Rosenhahn B. Pose Estimation Revisited. (PhD-Thesis) *Technical Report 0308, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2003. Available at `www.ks.informatik.uni-kiel.de`
20. Rosenhahn B., Perwass C. and Sommer G. Pose estimation of free-form surface models. In *Pattern Recognition, 25th DAGM Symposium*, B. Michaelis and G. Krell (Eds.), Springer, Berling, LNCS 2781, pp. 574-581, 2003.
21. Sommer G. (Ed.). *Geometric Computing with Clifford Algebra.* Springer, Berlin, 2001.
22. Theobalt C., Carranza J., Magnor A. and Seidel H-P. A parallel framework for silhouette based human motion capture *Proc. Vision, Modeling, Visualization 2003*, Munich, Nov. 19-21, pp. 207-214, 2003.
23. Zang Z. Iterative point matching for registration of free-form curves and surfaces. *Computer Vision*, Vol. 13, No. 2, pp. 119-152, 1999.