Vector Field Based Shape Deformations

Wolfram von Funck* MPI Informatik
Holger Theisel† MPI Informatik
Hans-Peter Seidel‡ MPI Informatik

Figure 1: Volume-preserving deformation of the hand model (36619 vertices): no skeletal hand model is involved, no self-intersections occur.

Abstract

We present an approach to define shape deformations by constructing and interactively modifying C^1 continuous time-dependent divergence-free vector fields. The deformation is obtained by a path line integration of the mesh vertices. This way, the deformation is volume-preserving, free of (local and global) self-intersections, feature preserving, smoothness preserving, and local. Different modeling metaphors support the approach which is able to modify the vector field on-the-fly according to the user input. The approach works at interactive frame rates for moderate mesh sizes, and the numerical integration preserves the volume with a high accuracy.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

Keywords: shape-deformation, volume-preserving, vector fields

1 Introduction

Shape deformations is a well-researched area in computer graphics and animation with many applications ranging from automotive design to movie production. A variety of techniques have been developed to transform an original shape into a new one under a certain number of constraints. These constraints can be for instance performance, detail preservation, feature preservation, volume preservation, avoidance of (local or global) self-intersections, or local support. In addition, different metaphors for an intuitive definition and handling of the deformation exist, like the free movement of certain handles [Singh and Fiume 1998; Bendels and Klein 2003; Pauly et al. 2003], a two-handed metaphor [Llamas et al. 2003], or the movement of a 9 dof object [Botsch and Kobbelt 2004].

Most existing deformation approaches have in common that they are defined as a map from the original to the new shape, i.e., it does not contain information about intermediate deformation steps. For many applications, the user wants to explore the deformation in an interactive manner, i.e., she wants to see a smooth change from the original to the desired shape moving along certain paths. This means that the deformation has to be recomputed again and again at interactive frame rates. To do so, parts of the deformation can be precomputed and reused for every intermediate deformation, see for instance [Botsch and Kobbelt 2004; Botsch and Kobbelt 2005].

In this paper we introduce an alternative approach to describe shape deformations. We assume that the shape is given as a triangular mesh. We construct a C^1 continuous divergence-free 3D time-dependent vector field $\mathbf{v}$ and obtain the new positions of every vertex $\mathbf{p}$ of the shape by applying a path line integration of $\mathbf{v}$ starting from $\mathbf{p}$. This approach is motivated by two observations. First, it corresponds to the metaphor of smooth deformations by observing the paths of the vertices over time. Second, due to the zero-divergence of $\mathbf{v}$ we get a number of desired properties of the deformation for free. In particular, the following properties hold:

- No self-intersections (neither local nor global) can occur. This is due to the fact that path lines do not intersect in the 4D space-time domain [Theisel et al. 2005].
- The deformation is volume-preserving. This is a well-known property of divergence-free vector fields [Davis 1967].
- The deformation preserves the smoothness of the shape to first order. This is due to the C^1 continuity of $\mathbf{v}$: under a path surface integration, the normals of the evolving shape depend on $\nabla\mathbf{v}$. Hence, for a C^1 continuous $\mathbf{v}$ no discontinuities of the surface normals appear during the integration.
- The deformation preserves details and sharp features in a sense that no smoothing due to an energy minimization occurs.

Figure 2a gives an illustration of the main idea. In addition to the above-mentioned properties of $\mathbf{v}$, we construct it to be non-zero only in a certain area to obtain a local support of the deformation. Although divergence-free vector fields have been used to model the

*e-mail: wfunck@mpi-inf.mpg.de
†e-mail: theisel@mpi-inf.mpg.de
‡e-mail: hpseidel@mpi-inf.mpg.de

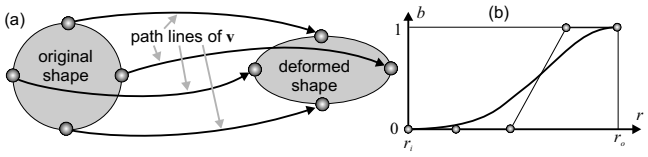

Figure 2: (a) Vector field based shape deformation: every vertex of the original shape undergoes a path line integration of $\mathbf{v}$ to find its new position. (b) Blending function $b(r)$.

flow of fluids [Foster and Fedkiw 2001], we are not aware of any approach to use them for the interactive deformation of solid shapes.

The rest of the paper is organized as follows: section 2 describes related work to shape deformations. Section 3 describes how to construct the locally supported divergence-free vector field $\mathbf{v}$. Section 4 shows a number of modeling metaphors of our technique. Section 5 gives implementation details. Section 6 gives an evaluation of our technique and compares it with other approaches. Conclusions are drawn in Section 7.

2 Related work

Current shape deformation approaches can be classified as surface based methods or space deformation methods. Surface based methods define the deformation only on the shape's surface. A common approach is to specify a number of original and target vertices and compute the remaining vertex positions by a variational approach [Welch and Witkin 1992; Taubin 1995]. Multiresolution methods are well-established because of their ability to speed up computations and preserve features [Zorin et al. 1997; Guskov et al. 1999; Kobbelt et al. 1998; Botsch and Kobbelt 2004]. More recently, approaches have been proposed which rely on the solution of the Laplace/Poisson equations [Alexa 2003; Lipman et al. 2004; Sorkine et al. 2004; Yu et al. 2004; Lipman et al. 2005; Zayer et al. 2005]. These approaches end up in the repeated solution of a large sparse linear system. Space deformation techniques modify objects by deforming their embedded space. Prominent representatives of this are free-form deformation methods which can be classified as lattice-based [Sederberg and Parry 1986; Coquillart 1990; MacCracken and Joy 1996], curve-based [Barr 1984; Singh and Fiume 1998], or point-based [Hirota et al. 1992; Hsu et al. 1992]. Different basis functions to define the space deformation have been applied, like radial basis functions [Botsch and Kobbelt 2005] or swirls [Angelidis et al. 2004a]. [Zhou et al. 2005] extends the Laplacian approach from surface based techniques to a volumetric approach. An often addressed issue when dealing with space deformations is the avoidance of self-intersections [Angelidis et al. 2004b; Mason and Wyvill 2001; Gain and Dodgson 2001]. A number of space deformation techniques are designed to be volume preserving in a global [Hirota et al. 1992; Desbrun and Gascuel 1995; Rappoport et al. 1996; Aubert and Bechmann 1997; Angelidis et al. 2004a] or local [Botsch and Kobbelt 2003] way. The constraint of volume preservation promises to give physically more plausible and natural deformations. [Angelidis et al. 2004a] presented the first space deformation that is both volume-preserving and foldover-free. While this approach is based on volume-preserving swirls, our approach is based on a formal construction of divergence-free vector fields.

3 Constructing the vector field v

The construction of the deformation vector field $\mathbf{v}$ is the core of our approach. It must be flexible enough to describe a variety of different deformations, but simple enough to be computed and updated on-the-fly. We present the construction of $\mathbf{v}$ both for 2D and 3D. While we use the 3D case for our applications, the 2D case serves mainly for illustrating the concept. Also, we formulate the construction in the static (time-independent) context because the extension to time-dependent fields is straightforward.

It is a well-known fact [Davis 1967] that a 2D divergence-free vector field $\mathbf{v}$ can be constructed as the co-gradient field of a scalar field $p(x,y)$:

$$\mathbf{v}(x,y) = \begin{pmatrix} -p_y(x,y) \\ p_x(x,y) \end{pmatrix}. \tag{1}$$

Here, p_x and p_y denote the partial derivatives $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial y}$, respectively. In 3D, a divergence-free vector field $\mathbf{v}$ can be constructed from the gradients of two scalar fields $p(x,y,z)$, $q(x,y,z)$ as

$$\mathbf{v}(x,y,z) = \nabla p(x,y,z) \times \nabla q(x,y,z). \tag{2}$$

We are going to construct $\mathbf{v}$ as a piecewise field: inside a certain region, $\mathbf{v}$ should be a simple and well-defined field, such as constant (describing a simple translation of parts of the shape) or linear (describing a rotation). We call this region the *inner region* of the deformation. Also, we have an *outer region* in which we have a zero deformation, i.e., $\mathbf{v} \equiv \mathbf{0}$. Between them there is an *intermediate region* in which $\mathbf{v}$ is blended between inner and outer region in a globally divergence-free and C^1 continuous way. We specify the different regions implicitly by defining a scalar *region field* $r(\mathbf{x})$ with $\mathbf{x} = (x,y)$ or $\mathbf{x} = (x,y,z)$, and two thresholds $r_i < r_o$. Then a point $\mathbf{x}$ is in the inner region if $r(\mathbf{x}) < r_i$, $\mathbf{x}$ is in the intermediate region if $r_i \le r(\mathbf{x}) < r_o$, and $\mathbf{x}$ is in the outer region if $r_o \le r(\mathbf{x})$.

Let $e(\mathbf{x})$, $f(\mathbf{x})$ be two C^2 continuous scalar fields which are supposed to define $\mathbf{v}$ in the inner region, i.e. $\mathbf{v} = \nabla e \times \nabla f$ there. Then we can define the piecewise scalar fields p, q as

$$p(\mathbf{x}) = \begin{cases} e(\mathbf{x}) & \text{if } r(\mathbf{x}) < r_i \\ (1-b)\cdot e(\mathbf{x}) + b\cdot 0 & \text{if } r_i \le r(\mathbf{x}) < r_o \\ 0 & \text{if } r_o \le r(\mathbf{x}) \end{cases} \tag{3}$$

$$q(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } r(\mathbf{x}) < r_i \\ (1-b)\cdot f(\mathbf{x}) + b\cdot 0 & \text{if } r_i \le r(\mathbf{x}) < r_o \\ 0 & \text{if } r_o \le r(\mathbf{x}) \end{cases} \tag{4}$$

where $b = b(r(\mathbf{x}))$ is a blending function given in Bézier representation as:

$$b(r) = \sum_{i=0}^{4} w_i B_i^4 \left(\frac{r - r_i}{r_o - r_i} \right) \tag{5}$$

where B_i^4 are the Bernstein polynomials [Farin 2002], $w_0 = w_1 = w_2 = 0$ and $w_3 = w_4 = 1$. Figure 2b illustrates b. Note that in (3) and (4) the term $b \cdot 0$ can safely be removed. We left it in the equation to show that in the intermediate region, p is a weighted combination of e and 0, and q is a weighted combination of f and 0.

(2)–(5) give a C^1-continuous divergence-free vector field $\mathbf{v}$ if the scalar fields e, f, r together with the thresholds r_i, r_o are given and e, f, r are C^2-continuous. This is ensured because the blending function b is designed to have a sufficient number of vanishing derivatives at $r(\mathbf{x}) = r_i$ and $r(\mathbf{x}) = r_o$. A proof of this is in appendix 8. Also note that from e, f, r and their first order partials, $\mathbf{v}$ can be computed in a closed form. Appendix 8 shows this as well.

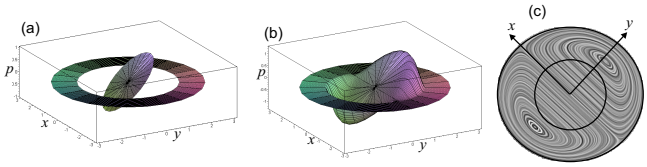

Figure 3: Constructing a constant $\mathbf{v}$ inside the inner region: (a) p as height field in inner and outer region, (b) p in all regions, (c) $\mathbf{v}$ in inner and intermediate region.

We illustrate our concept with a 2D example. Setting the region field $r(x,y) = x^2 + y^2$, $r_i = 1$ and $r_o = 4$, the inner region is the unit circle, while the intermediate region is the ring between the radii 1 and 2. We want $\mathbf{v} = (u,v)^T$ to be constant inside the inner region. To do so, e is the linear field $e(x,y) = v\,x - u\,y$ from which (3) and (1) give $\mathbf{v}$ in all regions. Figure 3a illustrates p in the inner and outer region as height field. Figure 3b additionally considers p in the intermediate region. Figure 3c shows the resulting $\mathbf{v}$ as a Line Integral Convolution (LIC) image in the inner and intermediate region. In the outer region, $\mathbf{v}$ is constant zero.

3.1 Special Deformations

Theoretically, every divergence-free vector field $\mathbf{v}$ can be considered in the inner region, i.e., arbitrary C^2 scalar fields e, f can be chosen. However, for our applications we particularly used constant, linear, and quadratic vector fields.

A constant vector field $\mathbf{v}$ in the inner region describes a translation inside this region. To get a constant field $\mathbf{v} = (u,v,w)^T$ with $\|\mathbf{v}\| = 1$, we choose two arbitrary orthogonal vectors $\mathbf{u}$, $\mathbf{w}$ with $\|\mathbf{u}\| = \|\mathbf{w}\| = 1$ and $\mathbf{u}\,\mathbf{v} = \mathbf{u}\,\mathbf{w} = \mathbf{v}\,\mathbf{w} = 0$. Also, we have to set a center point $\mathbf{c}$ inside the inner region. This is necessary because e, f have a constant to be added as degree of freedom. The center point $\mathbf{c}$ fixes this by setting $e(\mathbf{c}) = f(\mathbf{c}) = 0$. Then the linear scalar fields

$$e(\mathbf{x}) = \mathbf{u}\,(\mathbf{x}-\mathbf{c})^T \quad , \quad f(\mathbf{x}) = \mathbf{w}\,(\mathbf{x}-\mathbf{c})^T \tag{6}$$

produce $\mathbf{v}$ because (6) yields $\nabla e \equiv \mathbf{u}$ and $\nabla f \equiv \mathbf{w}$.

A linear vector field $\mathbf{v}$ is used to describe a rotation inside the inner region. Given a rotational axis by a center point $\mathbf{c}$ and the normalized axis direction $\mathbf{a}$, the field e is linear with the gradient $\mathbf{a}$, while the field f is quadratic, describing the squared Euclidean distance to the rotation axis:

$$e(\mathbf{x}) = \mathbf{a}\,(\mathbf{x}-\mathbf{c})^T \quad , \quad f(\mathbf{x}) = (\mathbf{a} \times (\mathbf{x}-\mathbf{c})^T)^2. \tag{7}$$

4 Modeling metaphors

Our approach works as a simultaneous path line integration and updating of $\mathbf{v}$. In fact, at a certain time, for every vertex one integration step of a numerical path line integration of $\mathbf{v}$ is carried out. Then $\mathbf{v}$ is updated, i.e., the defining fields e, f, r together with r_i, r_o are changed before the next integration step is carried out. There are different strategies to define and update $\mathbf{v}$, leading to a number of modeling metaphors of our technique. Before describing them in detail, we explain the visualization of the deformation tools, i.e., $\mathbf{v}$ at a certain time. We represent the region field r by a red semitransparent isosurface $r(\mathbf{x}) = r_i$ and a green surface $r(\mathbf{x}) = r_o$ separating the different regions of the deformation. If inside the inner region we use a constant $\mathbf{v}$ (section 3.1), we show it by an arrow whose origin is the central point $\mathbf{c}$ and whose direction denotes $\mathbf{v}$. Figure 4a shows an example where r describes the distance to a certain point and $\mathbf{c}$ is in the center of the inner region (red sphere). If we apply a rotation inside the inner region (section 3.1), we show $\mathbf{c}$ and the central axis. If we combine it with a linear r, the isosurfaces $r(\mathbf{x}) = r_i$ and $r(\mathbf{x}) = r_o$ are planes. In order to make the deformation local, we restrict it to a cylinder with the main axis parallel to ∇r and $\mathbf{c}$ on the main axis. Figure 4b shows the tool. Since this way the deformation vector field $\mathbf{v}$ is discontinuous across the cylinder barrel, the tool is only applicable if no part of the shape intersects the cylinder barrel at the beginning of the deformation.

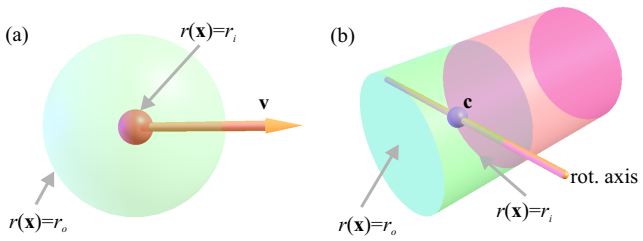

Figure 4: Deformation tools: (a) translation, (b) rotation inside the inner region.

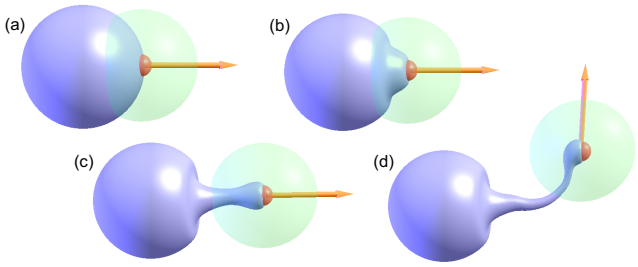

Figure 5: Deforming a sphere with an implicit tool: the parts inside the inner region follow the path of the tool.

4.1 Implicit Tools

For the metaphor of implicit tools, we define an arbitrary scalar field r together with r_i and r_o. Usually, r is a simple function describing the distance to a point (Figure 5) or a line segment (Figure 6). Furthermore, $\mathbf{c}$ is located in the center of the inner region of the deformation. Inside this inner region, $\mathbf{v}$ is constant, describing a translation where its length and direction is determined by position and movement of an interactive input device (e.g. a mouse). When the tool is interactively moved, $\mathbf{v}$ is updated on-the-fly according to the movement. The step size of the path line integration is chosen so that the path line follows the path of the tool: if the interactive motion changes the position of the tool by $\Delta\mathbf{r}$, then the integration inside the inner region moves the points by $\Delta\mathbf{r}$ as well (see Section 5.1 for more detail). This way we get the following property: if at the beginning of the deformation parts of the surface are in the inner region of the tool, they follow the path of the tool. Figure 5 shows an example. If at the beginning of the deformation the inner region is completely outside the shape, the shape will never enter the inner region. Figure 6 illustrates this. Figure 7 shows the result of an extreme deformation by moving the tool toward and through the shape: no self-intersection can occur.

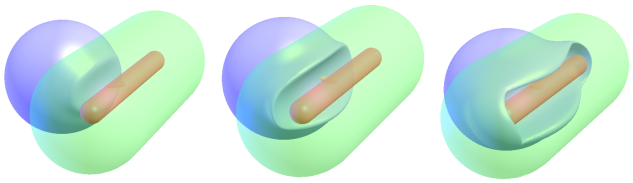

Figure 6: Moving the implicit tool toward the shape: the inner region never enters the shape.

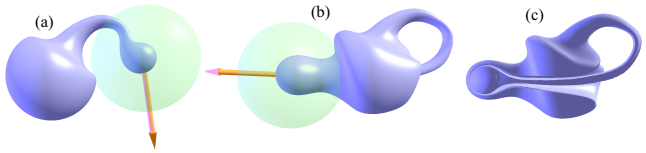

Figure 7: (a),(b) Moving an implicit tool through a sphere shape: no self-intersections occur. (c) Same shape as (b) but with cutting plane.

4.2 Deformation Painting

In this modeling metaphor, the tool is moved along a path on the surface of the shape. For this path, the surface is locally deformed into or out of the shape. If the tool is at the location $\mathbf{x}_s$ on the shape at a certain time, we use $r(\mathbf{x}) = \|\mathbf{x}-\mathbf{x}_s\|$, $r_i = 0$, and r_o is interactively chosen to steering the area of influence of the deformation. This means that the inner region is only the point $\mathbf{x}_s$ for which we use a constant $\mathbf{v}$ in the direction opposite to the surface normal of $\mathbf{x}_s$. Figure 8 shows an example of deformation painting on a hand model.

4.3 Moving point sets

In this metaphor, we mark a number of points on the shape. These points may be isolated or located on a curve. Then we set r as a smooth approximated distance function to this point set, $r_i = 0$, and r_o is interactively chosen. Inside the inner region, a constant $\mathbf{v}$ is used. For the distance function to the point set, we used the approach described in [Biswas and Shapiro 2004]. Figure 9 illustrates an example. The barycenter of all points is used as $\mathbf{c}$. Note that in this scenario the inner and the intermediate regions may consist of multiple unconnected parts.

4.4 Collision tools and shape stamping

In this scenario, the tool is described by an arbitrary closed tool shape for which a repeated collision detection with the deformed shape is carried out. To do so, we used a bounding box hierarchy based approach implemented in [ColDet]. Based on the detected collision points, r is constructed to be zero only in the areas of collision. Similar to moving points sets, we used a smooth approximated distance function for r along with $r_i = 0$. Inside the inner region, $\mathbf{v}$ is constant for every time step, following the path of the input device. Figures 10a,b illustrate the region function for the hand shape. Here, both the inner and the intermediate region consist of several unconnected parts. Figure 11 shows the deformation of the fan data set using the hand tool. Note that the sharp features are preserved under the deformation.

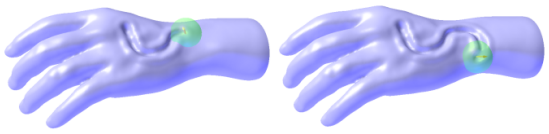

Figure 8: Deformation painting on a hand model.

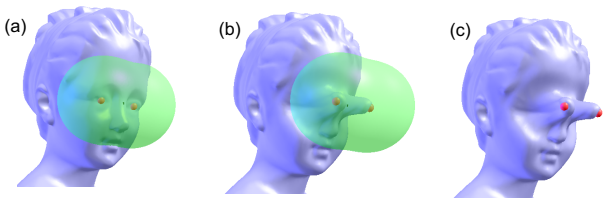

Figure 9: Moving point sets: the inner region consist of two unconnected parts close to the eyes of the bust.

We also used this modeling metaphor for shape stamping: moving the tool shape toward the deforming shape leaves the footprint on the deforming shape. Figures 10c,d show an example stamping a Y-shaped tool shape onto a sphere. Figure 13 shows the deformation of the crater data set by using the Armadillo model as tool.

4.5 Twisting and Bending

Up to now, the vector field inside the inner region was constant. Now we apply linear and quadratic vector fields to get twisting and bending effects. For a twisting, r is linear and its gradient corresponds the direction of the twisting axis. The point $\mathbf{c}$ is on the twisting axis, and the rotational axis for the inner vector field coincides with the twisting axis as well. Inside the inner region we use a linearly increasing rotation defined by $e(\mathbf{x}) = (\mathbf{a}\,(\mathbf{x}-\mathbf{c})^T)^2$, $f(\mathbf{x}) = (\mathbf{a}\times(\mathbf{x}-\mathbf{c})^T)^2$. Figure 14 shows an extreme twisting of the box model (51202 vertices) as well as a moderate twisting of the camel model.

To get a bending effect, we used the bending tool described in Figure 4b. The region field r is linear and its gradient is perpendicular to the rotation axis. The thresholds r_i, r_o are chosen such that $r(\mathbf{c}) = 0.614 r_i + (1-0.614) r_o$. This choice comes from the definition of the blending function: $b(0.614) \approx 1/2$. During the bending, the gradient of r is changed with half the angle speed as the rotation in the inner region. Figure 15 gives an example. $e(\mathbf{x})$ and $f(\mathbf{x})$ describe a rotation (Equation 7). Figure 1 shows some deformations of the hand data. The result looks rather realistic, even though no skeletal hand model is involved. Figure 16 shows the bending of the Armadillo model.

Figure 17 shows two shapes created from a sphere in an interactive session by applying all modeling metaphors described above. The session time for each of the models was approximately 30 minutes.

5 Implementational Details

5.1 Integration with adaptive stepsize

Different approaches for a numerical stream/path line integration have been studied [Nielson et al. 1997], where higher order techniques with an adaptive stepsize turned out to have the best trade-

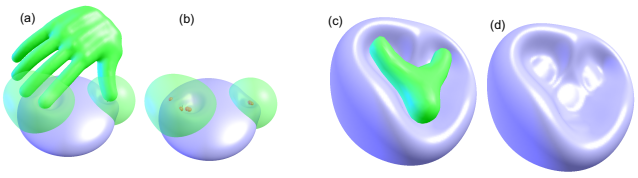

Figure 10: Collision tools. (a),(b) The inner and the intermediate region consists of unconnected parts. (c),(d) Shape stamping: a Y-shaped tool is stamped onto a sphere.

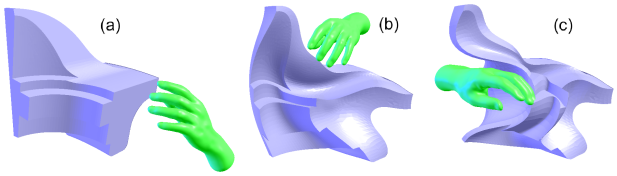

Figure 11: Deforming the fan data set.

off between speed and accuracy. Thus, in our implementation we used a fourth order Runge-Kutta integration with adaptive stepsize. If a constant stepsize were chosen, the interplay between integration and updating $\mathbf{v}$ is simple: for each vertex, one integration step is carried out, then $\mathbf{v}$ is updated. For an adaptive step size, the synchronization between integration and updating $\mathbf{v}$ is explained in the following example: suppose we use an implicit tool where the mouse moves the central point from $\mathbf{c}_i$ at the time t_i to $\mathbf{c}_{i+1}$. Further assuming that $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$ are in the inner region of the deformation, $\mathbf{v}$ has to fulfill

$$\mathbf{v}(\mathbf{c}_i, t_i) = \mathbf{v}(\mathbf{c}_{i+1}, t_{i+1}) = (t_{i+1} - t_i) \cdot (\mathbf{c}_{i+1} - \mathbf{c}_i) \tag{8}$$

Following the description of path lines in [Theisel et al. 2005], we integrate the 4-dimensional vector field

$$\tilde{\mathbf{v}}(\mathbf{x}, t) = \begin{pmatrix} (1 - \frac{t-t_i}{t_{i+1}-t_i})\mathbf{v}_i + \frac{t-t_i}{t_{i+1}-t_i}\mathbf{v}_{i+1} \\ 1 \end{pmatrix} \tag{9}$$

with an adaptive stepsize from a point $(\mathbf{x}, t_i)$ until it reaches a point $(\acute{\mathbf{x}}, t_{i+1})$. In (9), t_{i+1} is chosen to fulfill (8) and $\mathbf{v}_i$, $\mathbf{v}_{i+1}$, are computed from $r(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_i\|$ and $r(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_{i+1}\|$ respectively.

5.2 Remeshing

Obviously, large deformations on triangle meshes can cause unpleasing artifacts due to an undersampling of the surface (Figure 18a). Furthermore, undersampling can lead to significant volume changes. We deal with this problem by resampling (remeshing) the mesh. For doing so, a variety of approaches exist (see [Alliez et al. 2005] for a survey). For our algorithm, we used some ideas from [Gain and Dodgson 1999]. The basic idea is to do the remeshing not on the deformed but on the original shape, and let the new vertices undergo the same deformation as the original vertices.

1. While the user performs a deformation on the mesh, the undeformed mesh M and the deformation path (the subsequent translations/rotations) are stored.

2. When the user finishes the operation (in our implementation: when the user releases the mouse button), all edges (ordered by length) of the deformed mesh M' are tested for refinement: if an edge is longer than a certain threshold or the angle between the normals of the end-vertices is large, an edge split is performed both on M and M'. Using the stored deformation path, all new vertices of M are deformed, i.e., path line integrated. Finally, the new vertex positions are copied to the corresponding vertices of M'.

3. In order to guarantee a uniform distribution of the vertices and to eliminate slivers (long, thin triangles), we apply a diffusion of the vertices in M': first, all moved vertices and their immediate neighbors are marked for diffusion. Each marked vertex is moved toward the barycenter of its 1-ring. Afterward, the vertex is projected back onto the surface of the undiffused mesh. This operation is repeated for a fixed number of steps.

4. Subsequent edge splits increase the overall vertex valence and some deformations produce surfaces that are oversampled. Both issues are tackled by a decimation step on M': all edges whose length is smaller than a certain threshold and whose vertex normals enclose a small angle are collapsed at their midpoints.

5. Step 3 is performed again, where the vertices involved in edge collapses are considered as moved vertices.

Figure 18 (b) shows a mesh after an application of the above algorithm.

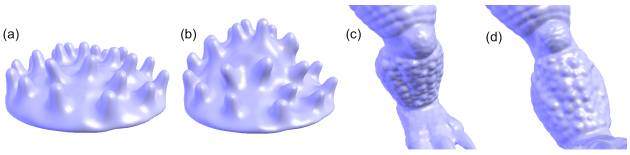

Figure 12: Feature preservation: (a),(b) Deforming a model with small details. (c),(d) Bending Armadillo's leg.

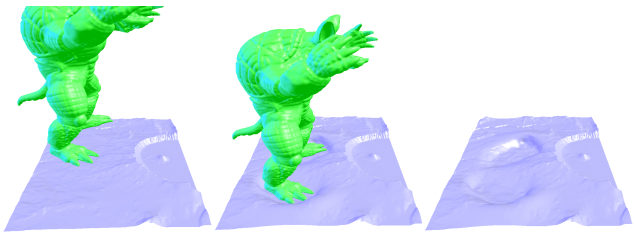

Figure 13: Armadillo on the moon: deforming the crater data set with the Armadillo as tool.

5.3 GPU Implementation

Being based on path line integration of single points and being without the need for additional information like mesh connectivity or a skeleton, our deformation approach is highly parallelizable using graphics hardware. We implemented a vertex program to perform an adaptive fourth order Runge-Kutta path line integration of points. All necessary parameters like translation vector, rotation axis, contact points etc. are passed to the shader as uniform variables. The resulting positions are rendered to a floating point framebuffer. Due to the incremental nature of the algorithm (collision detection after each small deformation, painting on a continually deforming surface etc.) a read-back of the computed points has to be performed after each deformation. Although this drops performance, the computation is still about ten times faster than on the CPU. A detailed performance evaluation can be found in section 6.

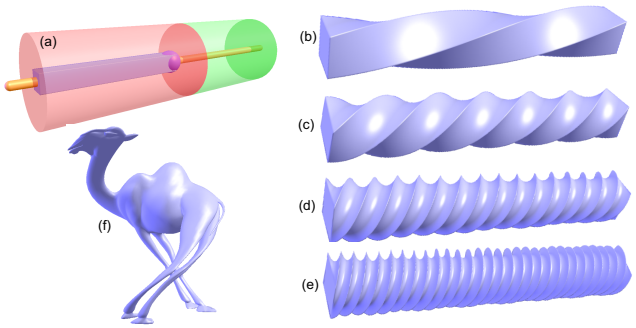

Figure 14: Twisting the box model: (a) placing the tool, (b)-(e) twisted models, (f) twisted camel.

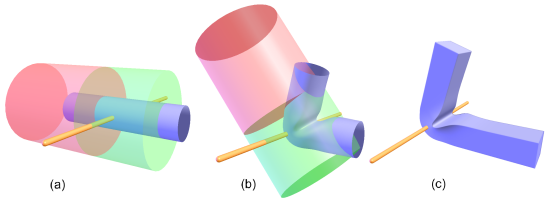

Figure 15: (a),(b) Bending a cylinder. (c) Box after bending.

6 Evaluation and Comparison

In this section we give an evaluation of our technique and compare it with other deformation approaches. We do so in terms of visual quality, other modeling metaphors, speed and accuracy.

Visual quality: To get a comparison with existing techniques, we apply our technique to a number of standard test data sets for which other deformation approaches have been reported in the literature. The twisting of a box (Figure 14) has been considered in [Yu et al. 2004; Lipman et al. 2005; Zhou et al. 2005]. Our result shows the behavior of a volume-preserving twisting even for an extreme deformation. The effect of bending a cylinder has been demonstrated for different approaches in [Botsch and Kobbelt 2003; Botsch and Kobbelt 2004; Zhou et al. 2005]. Our result (Figures 15a-b) shows a realistic looking bend without self intersections. Also, the bending of the box (Figure 15c) and the deformation of the hand (Figure 1) look plausible and do not contain self-intersections. Furthermore, small scale features are deformed in a plausible manner (Figure 12).

Other modeling metaphors: Our implicit tool metaphor using r as the distance to a point (Figures 5, 7) is similar to the swirling sweepers metaphor [Angelidis et al. 2004a]. However, our metaphor is more flexible in the sense that other implicit functions can be used (Figure 6). Moreover, contrary to swirling sweepers our method does not have to choose an appropriate number of basic swirls to approximate the final deformation.

Many modeling metaphors [Botsch and Kobbelt 2004; Sorkine et al. 2004; Botsch and Kobbelt 2005] work by setting areas of zero deformation and areas of full deformation on the surface. Then the full deformation is defined by a sequence of translations and rotations. Our tools have a similar metaphor, with the main difference that we define the regions of zero and full deformation implicitly, i.e. by marking the underlying space. This may create problems in areas where surface parts of zero and full deformation are spatially close to each other (for instance the fingers of a hand). For these cases we use the cylinder tool (Figure 4b) and restrict the definition to the area inside the cylinder.

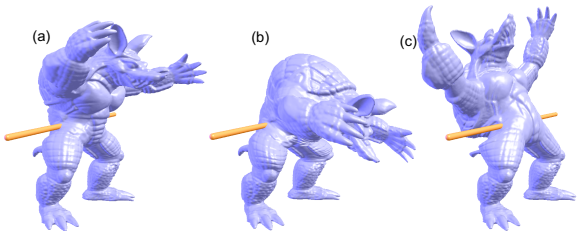

Figure 16: Bending the Armadillo model.

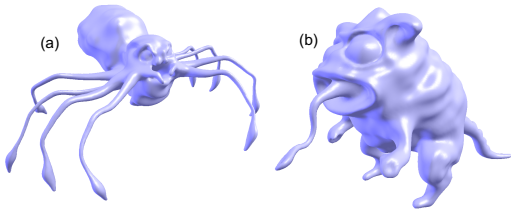

Figure 17: Shapes created from spheres in an interactive session.

Speed: The performance of our approach depends on a number of factors: the number of vertices inside the inner region of the deformation, the number of vertices in the intermediate region, the chosen modeling metaphor, and the chosen region field r. In general, vertices in the intermediate region are more expensive to integrate than vertices in the inner region because $\mathbf{v}$ has a more complicated form there. Also, a simple r (such as the distance to a point in an implicit tool) gives a higher performance. Finally, for the metaphor of shape stamping, the additional collision-detections drops the performance. To get an evaluation of the performance of our approach, we consider a number of benchmark deformations. We placed an implicit tool describing the distance to a point in such a way that no vertex of the shape is in the outer region (i.e., that all vertices have to be integrated), and that most of the vertices are in the (most expensive) intermediate region. Then we applied a rather strong deformation. Figure 19 shows the four benchmark deformations by the initial and final shapes as well as used deformation tools. Here, the sphere-shaped intermediate region (green) has been cut out at the black boundary lines. The following table shows the performance on a AMD Opteron 152 (2.6 GHz) with 2 GB RAM and a GeForce 6800 GT GPU. There, #vert denotes the number of vertices of the model, #steps denotes the number of integration steps to come from the original to the final shape, sps(CPU) gives the number of integration steps per second for the CPU implementation, and sps(GPU) does so for the GPU implementation.

model	#vert	#steps	sps(CPU)	sps(GPU)
bust	30696	212.737	31.65260	292.2210
hand	36619	186.194	26.94950	257.8860
armadillo	172974	152.656	5.79448	61.8539
dragon	437645	249.196	2.29102	27.2049

It turns out that even for rather large meshes the deformations can be carried out in an interactive manner. All the examples in the accompanying video are captures from interactive sessions.

Accuracy: The statement that our approach is volume-preserving holds only if every surface point of the shape undergoes an exact path line integration. In reality, we carry out a numerical integration only for discrete surface points: the mesh vertices. Thus, slight changes of the volume during the deformation can be expected. However, the following table shows that they are minimal even for strong deformations. Here, we measured the error

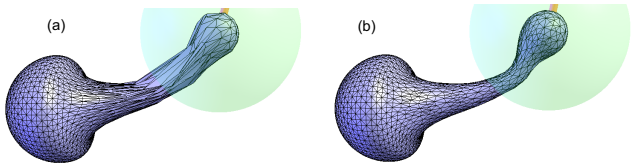

Figure 18: Remeshing. (a) Mesh during deformation, (b) mesh after deformation and remeshing.

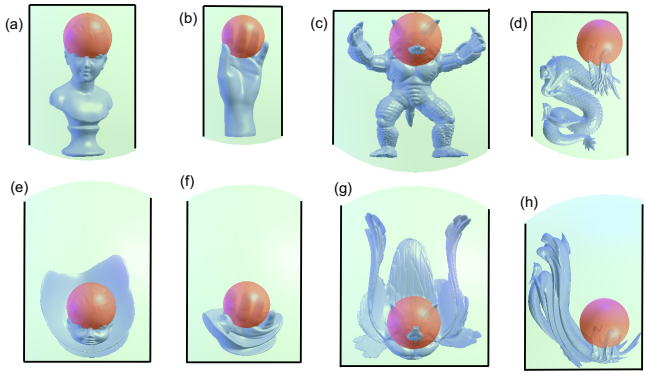

Figure 19: Benchmark deformations. First line: original shapes and parts of tools. Second line: deformed shapes and parts of tools.

as $error = \dfrac{\text{volume(deformed shape)}}{\text{volume(original shape)}} - 1$.

model	orig. shape	deformed shape	*error*
sphere	fig. 5a	fig. 7b	-0.001060
box, twisted	fig. 14a	fig. 14d	0.000781
box, bent	fig. 14a	fig. 15c	0.000751
fan	fig. 11a	fig. 11c	-0.000007
armadillo	fig. 16a	fig. 16c	0.001344
dragon	fig. 19d	fig. 19h	-0.001520
spider	fig. 5a	fig. 17a	0.001875
own monster	fig. 5a	fig. 17b	0.000070

7 Conclusion and Future Work

In this paper we introduced an alternative approach to shape deformations: by carrying out a path line integration of a time-dependent vector field for each shape vertex. This way, simple properties of the vector field lead to useful properties of the deformation: a divergence-free vector field gives a volume-preserving deformation, self-intersections cannot occur, and sharp features are preserved. Also small scale features are deformed realistically. We have also shown that the performance of the deformation suffices for real-time applications for moderately large meshes. The accuracy in volume-preserving is rather high.

There is a number of issues for future research. First, the performance can further be increased by a multi-processor parallelization of the integration. This is possible because the integration of the vertices can be carried out independently of each other. Second, since the method does not rely on any connectivity information of the mesh, an application to point-based shape representations seems possible. Finally, the modeling metaphor should be extended such that the regions of full and zero deformation can be marked explicitly on the surface instead of implicitly in the embedding 3D space.

Acknowledgments

The authors would like to thank Thomas Annen and Carsten Stoll for their advice on GPU programming. The Armadillo and Dragon models are courtesy of Stanford University. The Fandisk, Camel and Hand models are from the AIM@SHAPE shape repository.

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2, 105–114.

ALLIEZ, P., UCELLI, G., GOTSMAN, C., AND ATTENE, M. 2005. *Recent Advances in Remeshing of Surfaces*. Springer.

ANGELIDIS, A., CANI, M.-P., WYVILL, G., AND KING, S. 2004. Swirling-sweepers: Constant-volume modeling. In *Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, 10–15.

ANGELIDIS, A., WYVILL, G., AND CANI, M.-P. 2004. Sweepers: Swept user-defined tools for modeling by deformation. In *Proceedings of Shape Modeling and Applications*, IEEE, 63–73.

AUBERT, F., AND BECHMANN, D. 1997. Volume-preserving space deformation. *Comput. and Graphics 21*, 5, 6125–639.

BARR, A. 1984. Global and local deformations of solid primitives. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 21–30.

BENDELS, G. H., AND KLEIN, R. 2003. Mesh forging: editing of 3d-meshes using implicitly defined occluders. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 207–217.

BISWAS, A., AND SHAPIRO, V. 2004. Approximate distance fields with non-vanishing gradients. *Graphical Models 66*, 3, 133–159.

BOTSCH, M., AND KOBBELT, L. 2003. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum 22*, 3, 483–491. (Proceedings Eurographics 2003).

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3, 630–634.

BOTSCH, M., AND KOBBELT, L. 2005. Real-time shape editing using radial basis functions. *Computer Graphics Forum 24*, 3, 611–621. (Proceedings Eurographics 2005).

COLDET. Free 3D collision detection library. http://photoneffect.com/coldet.

COQUILLART, S. 1990. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 187–196.

DAVIS, H. 1967. *Introduction to vector analysis*. Allyn and Bacon, Inc., Boston.

DESBRUN, M., AND GASCUEL, M.-P. 1995. Animating soft substances with implicit surfaces. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 287–290.

FARIN, G. 2002. *Curves and Surfaces for CAGD*, 5th ed. Morgan Kaufmann, San Francisco.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 23–30.

GAIN, J. E., AND DODGSON, N. A. 1999. Adaptive refinement and decimation under free-form deformation. *Eurographics UK '99*.

GAIN, J. E., AND DODGSON, N. A. 2001. Preventing self-intersection under free-form deformation. *IEEE Transactions on Visualization and Computer Graphics 7*, 4, 289–298.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 325–334.

HIROTA, G., MAHESHWARI, R., AND LIN, M. 1992. Fast volume-preserving free form deformation using multi-level optimization. In *Proceedings Solid Modeling and applications*, 234–245.

HSU, W., HUGHES, J., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 177–184.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 105–114.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, IEEE Computer Society Press, 181–190.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

LLAMAS, I., KIM, B., GARGUS, J., ROSSIGNAC, J., AND SHAW, C. 2003. Twister: a space-warp operator for the two-handed editing of 3d shapes. *ACM Trans. Graph. 22*, 3, 663–668.

MACCRACKEN, R., AND JOY, K. 1996. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 181–188.

MASON, D., AND WYVILL, G. 2001. Blendeforming: Ray traceable localized foldover-free space deformation. In *CGI '01: Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, Washington, DC, USA, 183.

NIELSON, G., HAGEN, H., AND MÜLLER, H. 1997. *Scientific Visualization*. IEEE Computer Society.

PAULY, M., KEISER, R., KOBBELT, L., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM Trans. Graph. 22*, 3, 641–650.

RAPPOPORT, A., SHEFFER, A., AND BERCOVIER, M. 1996. Volume-preserving free-form solids. *IEEE Transactions on Visualization and Computer Graphics 2*, 1, 19–27.

SEDERBERG, T., AND PARRY, S. 1986. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 151–160.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 405–414.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Eurographics Association, 179–188.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 351–358.

THEISEL, H., WEINKAUF, T., HEGE, H.-C., AND SEIDEL, H.-P. 2005. Topological methods for 2D time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics 11*, 4, 383–394.

WELCH, W., AND WITKIN, A. 1992. Variational surface modeling. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 157–166.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3, 644–651.

ZAYER, R., RÖSSL, C., KARNI, Z., AND SEIDEL, H.-P. 2005. Harmonic guidance for surface deformation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, Blackwell, Dublin, Ireland, vol. 24, Eurographics, 601–609.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph. 24*, 3, 496–503.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 259–268.

8 Appendix

Given the C^2 continuous scalar fields e, f and the C^2 continuous region field r with the thresholds r_i, r_o, we show that $\mathbf{v}$ constructed by (2)–(5) is divergence-free and C^1. The zero-divergence follows directly from (2) [Davis 1967]. For showing the C^1 continuity, we have to consider the boundaries of the regions, i.e., the locations $\mathbf{x}$ with $r(\mathbf{x}) = r_i$ and $r(\mathbf{x}) = r_o$.

For $r(\mathbf{x}) = r_i$, (5) gives

$$b = 0 \quad , \quad \frac{db}{dr} = 0 \quad , \quad \frac{d^2 b}{dr^2} = 0. \tag{10}$$

For $b = b(r(\mathbf{x}))$, basic rules in differential calculus give

$$\nabla b = \frac{db}{dr}\nabla r \quad , \quad \mathbf{J}(\nabla b) = \frac{db}{dr}\mathbf{J}(\nabla r) + \frac{d^2 b}{dr^2}\nabla r \nabla r^T. \tag{11}$$

To prove that $\mathbf{v}$ is C^1, we have to show

$$\nabla e \times \nabla f = \nabla((1-b)\,e) \times \nabla((1-b)\,f) \tag{12}$$
$$\mathbf{J}(\nabla e \times \nabla f) = \mathbf{J}(\nabla((1-b)\,e) \times \nabla((1-b)\,f)) \tag{13}$$

(2)–(4) give that the left-hand side of (12) describes $\mathbf{v}$ in the inner region, while the right-hand side describes $\mathbf{v}$ in the intermediate region. (13) does so for the Jacobian of $\mathbf{v}$ in inner and intermediate region. Applying basic rules of differential calculus, we get

$$\nabla((1-b)\,e) = (1-b)\cdot\nabla e - e\nabla b \tag{14}$$
$$\nabla((1-b)\,f) = (1-b)\cdot\nabla f - f\nabla b \tag{15}$$

and

$$\mathbf{J}(\nabla((1-b)\,e)) = (1-b)\,\mathbf{J}(\nabla e) - \nabla e\,\nabla b^T - e\,\mathbf{J}(\nabla b) - \nabla b\,\nabla e^T \tag{16}$$
$$\mathbf{J}(\nabla((1-b)\,f)) = (1-b)\,\mathbf{J}(\nabla f) - \nabla f\,\nabla b^T - f\,\mathbf{J}(\nabla b) - \nabla b\,\nabla f^T \tag{17}$$

Inserting (10),(11) into (14),(15),(16),(17) we get

$$\nabla((1-b)\,e) = \nabla e \quad , \quad \nabla((1-b)\,f) = \nabla f \tag{18}$$
$$\mathbf{J}(\nabla((1-b)\,e)) = \mathbf{J}(\nabla e) \quad , \quad \mathbf{J}(\nabla((1-b)\,f)) = \mathbf{J}(\nabla f) \tag{19}$$

which gives (12),(13). In fact, (18),(19) show that p and q defined in (3), (4) are C^2 across locations with $r = r_i$.

For $r(\mathbf{x}) = r_o$, (5) gives

$$b = 1 \quad , \quad \frac{db}{dr} = 0. \tag{20}$$

To prove that $\mathbf{v}$ is C^1, we have to show

$$\nabla((1-b)\,e) \times \nabla((1-b)\,f) = \mathbf{0} \tag{21}$$
$$\mathbf{J}(\nabla((1-b)\,e) \times \nabla((1-b)\,f)) = \mathbf{0} \tag{22}$$

where the left-hand side of (21) describes $\mathbf{v}$ in the intermediate region and the right-hand side in the outer region. Inserting (11),(20) into (14), (15) gives (21). Inserting (11),(20) into (16), (17) together with $\mathbf{J}(\mathbf{a}\times\mathbf{b}) = \mathbf{J}(\mathbf{a})\times\mathbf{b} + \mathbf{a}\times\mathbf{J}(\mathbf{b})$ gives (22). It shows that at locations with $r = r_o$, p and q are only C^1, whereas $\mathbf{v}$ is C^1 as well. The C^1 continuity of p and q is sufficient here because we have the additional condition that $\mathbf{v}$ equals zero.

Note that (14),(15) together with (11) and $\frac{db}{dr} = \frac{2r}{r_o - r_i}$ from (5) gives the closed form of $\mathbf{v}$ if e, f, r and their first order partials are given.