Freeform Image

Thomas Schiwietz[1,2]
schiwiet@in.tum.de

Joachim Georgii[1]
georgii@in.tum.de

Rüdiger Westermann[1]
westermann@in.tum.de

[1]Technische Universität München
Computer Graphics & Visualization
Munich, Germany

[2]Siemens Corporate Research
Imaging & Visualization
Princeton, USA

Abstract

In this paper we present a technique for image deformation in which the user is given flexible control over what kind of deformation to perform. Freeform image extends available image deformation techniques in that it provides a palette of intuitive tools including interactive object segmentation, stiffness editing and force-based controls to achieve both a natural look and realistic animations of deforming parts. The model underlying our approach is physics-based and it is amenable to a variety of different kinds of image manipulations ranging from as-rigid-as-possible to fully elastic deformations. We have developed a multigrid solver for quadrangular finite elements, which achieves real-time performance for high resolution pixel grids. On recent CPUs this solver can handle about 16K co-rotated finite elements at roughly 60 ms.

1. Introduction and Related Work

Today it seems to be well accepted that as-rigid-as-possible image deformations produce the most intuitive results when the user wants to manually control the shape deformation. As-rigid-as-possible transformations as they were introduced by Alexa et al. [2] for the purpose of shape interpolation are characterized by a minimum amount of scaling and shearing. Such transformations mimic the adaption of the mechanical properties (i.e., the stiffness) of the transformation to enforce rigidity.

In the context of image deformation as-rigid-as-possible transformations have been introduced just recently to avoid undesirable effects like folding and non-uniform scaling typically perceived in previous methods [4, 6, 19]. Igarashi et al. [16] enforce the particular properties of as-rigid-as-possible transformations by minimizing the distortion of triangular elements in a 2D mesh. By separating the rotational part from the scaling the problem is decomposed into the solution of two least squares problems. In principle, this technique can be seen as a variant of constraint mesh deformation techniques, which pose the problem of shape deformation as an energy minimization problem. Such techniques have been used successfully for shape-preserving 3D mesh deformation, for instance in [1, 7, 15, 25], and in the context of image deformation by Kunzhou et al. [26].

Sheffer and Kraevoy [23] also build upon the idea of shape-preserving deformations, and they proposed a rotation invariant shape representation constraint by a set of angles and lengths relating a vertex to its immediate neighbors. Ran Gal et al. [11] presented a method for inhomogeneous texture mapping based on a classification of cells into rigid and elastic. Determining the global displacement field can then be posed as an optimization problem that takes into account the specific cell properties as additional constraints.

An alternative approach to as-rigid-as-possible image deformation was presented by Schaefer et al. [22]. Based on a globally smooth displacement function satisfying the interpolation constraints for a number of control points, for each pixel in the image the best transformation with respect

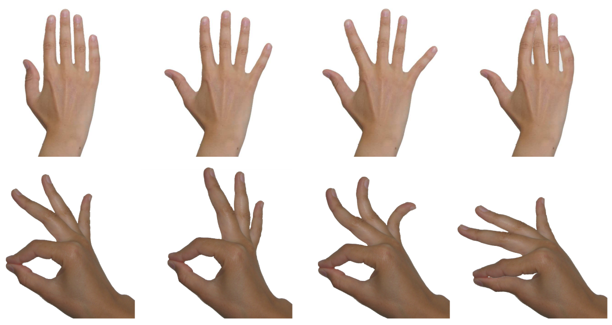
Figure 1. Some results of the image deformation method presented in this work are shown. The left column shows the original images.

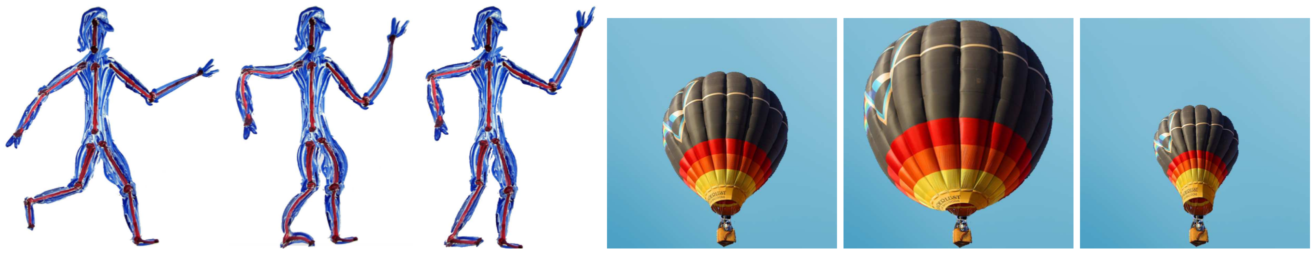

Figure 2. A sketch of a person (left image) is deformed using a uniform stiffness distribution (middle image). By assigning soft stiffness to the joints, a much more realistic deformation can be achieved (right image). Force-based control handles allow blowing the balloon (left) by simply inserting a source vector field (middle). Contrarily, the balloon can be shrunk by a sink vector field (right).

to a local energy constraint is computed. The authors compare different types of transformations and spot as-rigid-as-possible transformations the visually most pleasant ones. In contrast to the method of Igarashi et al. [16] much smoother implicit displacement fields are achieved, and the method performs at significant faster rates due to the small systems of equations to be solved locally for every pixel.

Although as-rigid-as-possible transformations are well suited for the purpose of shape preservation they are not effective at following physics-based constraints like volume preservation or elastic deformations. As images are typically composed of parts, it is often desired to continuously adjust the stiffness of the transformation to permit larger displacements over parts that are known to be very compliant and smaller displacements over parts that are known to be less compliant.

To provide more flexibility in the kind of deformations the user can apply and at the same time achieving numerical stability even for large deformations, Botsch et al. [7] introduced rigid cells as a new structure for shape manipulation. The global displacement field is defined by rigid transformations of a set of hexahedral cells, which are computed in such a way as to minimize a global energy functional. Radial basis functions are used to transfer per-cell transformations onto the vertices of a 2D or 3D render grid.

1.1 Contribution

The main contribution of this paper is an image deformation technique that combines physics-based and as-rigid-as-possible transformations. Interactive deformation of high-resolution pixel grids is achieved by using a numerical multigrid solver for the governing equations of motion of deformable 2D meshes. Several novel extensions have been developed to enable the user to flexibly control the image deformation:

- To avoid undesirable transversal contractions naturally arising in the simulation of elasticity we mimic anisotropic materials based on a modified Hooke's law.
- We have extended the finite element solver to quadrangular elements including bi-linear interpolation functions. This leads to improved simulation accuracy and accelerates the overall simulation time.
- An interactive GPU-based segmentation tool has been integrated to enable fast and intuitive stiffness editing, vertex fixation, and the assignment of depth values to parts of the image.
- We introduce force-based controls that can be shaped and scaled arbitrarily. These controls provide an effective means for smooth and realistic keyframe animations of deformed images.

The proposed method achieves the same advantages as rigid cells, yet it simulates a full physical model at significantly higher simulation rates. Due to the linear time complexity of an implicit multigrid solver we achieve fast and numerically stable image deformations. Physical simulation makes the method amenable to a variety of different kinds of shape manipulations ranging from as-rigid-as-possible to fully elastic deformations. In contrast to previous methods, deformations are initiated by applying external force fields to the vertices of a simulation grid. As a consequence, the method provides simple and intuitive tools to specify areal displacement distributions. Some results of our approach are shown in Figure 1 and 2.

The remainder of this paper is organized as follows: In the next section we will briefly discuss the basics of physical simulation as it is used in our approach. We will then describe a number of novel extensions we have developed to make the simulation applicable for image deformation. Next, we describe the areas where we use a segmentation algorithm, and we review the Random Walker algorithm for image segmentation. Finally, we analyze the proposed method with respect to performance and deformation options, and we conclude the paper with some remarks about future research in this field.

2. Physics-based Deformation

In this section, we give a brief overview of the physical model underlying our approach as well as the numerical scheme used to simulate the behavior of objects obeying to this model. In particular, we consider a linear elasticity model, but as an elastic material wants to resist a change in shape we enforce additional constraints to mimic plasticity. For the purpose of a more transparent discussion of the linear elasticity model we first assume the image to be represented as a 2D triangular grid, where exactly one grid point is placed at each pixel center. Grid points are then connected to their one-ring neighbors via edges. Given such a grid in the reference configuration $x \in \Omega$, the deformed grid is modelled using a displacement function $u(x), u : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ yielding the deformed configuration $x + u(x)$. Later, we will show how to replace triangular finite elements by quadrangular elements to further optimize our technique.

2.1. Linear Elasticity Model

If the object to be simulated obeys to the model of linear elasticity the dynamic behavior is governed by the Lagrangian equation of motion

$$M\ddot{u} + C\dot{u} + Ku = f \tag{1}$$

where M, C, and K are respectively known as the mass, damping and stiffness matrices. u consists of the linearized displacement vectors of all vertices and f are the linearized force vectors applied to each vertex. These vectors are either computed from mouse drags issued by the user or they are taken from pre-computed force templates as described below. The linear elasticity model only considers material that has a linear relationship between how hard it is squeezed or torn (stress) and how much it deforms (strain). This relationship is expressed by the material law, based on which the per-element contributions to the global stiffness matrix K are computed.

Equipped with any suitable discretization, finite element methods typically build the aforementioned matrices by assembling all element matrices to yield a sparse system of linear equations. For the details about the discretization process as well as the numerical schemes used to solve the resulting system let us refer to [3, 9, 21].

2.2. Numerical Simulation

One particular class of acceleration schemes that have recently gained much attention in the graphics community are numerical multigrid schemes [5, 12, 24]. Multigrid methods provide a general means for constructing customized solvers for partial differential equations, and they give rise to scalable solvers exhibiting linear complexity in the number of supporting vertices [8]. At the core of any geometric multigrid scheme a hierarchy of meshes at different resolutions is employed. In a multigrid V-cycle specific quantities are transferred across this hierarchy from fine to coarse and coarse to fine. Using the regular grid structures employed in this work, such a hierarchy can easily be built by regularly sub-sampling the grid vertices and connecting these vertices as described above. While vertex quantities are simply interpolated when going from the coarser to the finer resolution, the inverse step from fine to coarse essentially averages over the quantities at respective vertices in the fine mesh.

Building upon the work by Georgii et al. [12] we have implemented an implicit multigrid solver for image deformations amenable to the simulation of heterogeneous materials exhibiting a wide range of stiffness values. One problem that has to be addressed is the well-known lack of rotational invariants of the linear approximation of the strain tensor, which results in artificial forces once elements get rotated out of their reference configuration. Co-rotated finite elements as proposed in [10, 20] effectively cure this problem, and it was shown that they can efficiently be integrated into the multigrid scheme [12].

3. Image Deformation

In the following, we describe novel extensions to the physics-based finite element simulation to account for the particular needs in image deformation. One of the most significant of these extensions is a modified material law that enables as-rigid-as-possible transformations. In addition, we propose algorithms to efficiently handle material updates and to fixate simulation vertices without sacrificing speed.

3.1. Material Laws

The stiffness matrix K accounts for the strain and stress energy, and it depends on the material laws and parameters used to couple stress and strain. In an isotropic and fully elastic body stress (Σ) and strain ($\mathcal{E}$) tensors are coupled through Hooke's law (linear material law)

$$\Sigma = \frac{E\nu}{(1+\nu)(1-2\nu)} \left(\sum_{i=1}^{2} \mathcal{E}_{ii} \right) \cdot I_{22} + \frac{E}{1+\nu}\,\mathcal{E}, \tag{2}$$

where E is the elastic modulus, ν is the Poisson ratio and I_{22} is the identity matrix.

As it has been observed by many others before, shape deformations obeying to Hooke's law are often not desired

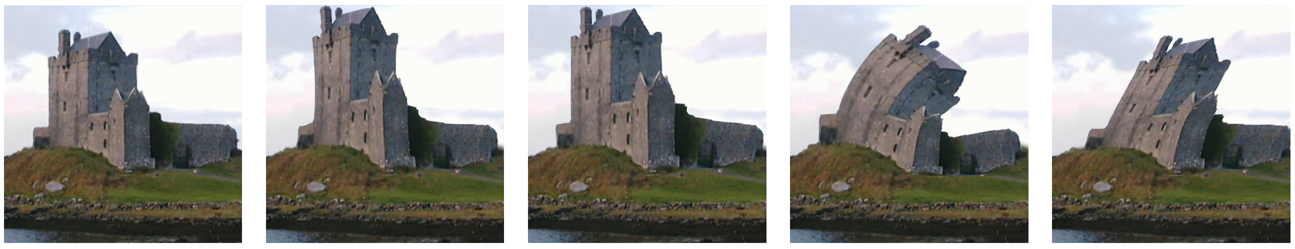
(a) reference image (b) Hooke's law (c) Rigid law ($\alpha = 1$) (d) Rigid law ($\alpha = 100$) (e) Rigid law ($\alpha = 0.1$)

Figure 3. Modified material laws: Images (b) and (c) show the difference between Hooke's law and the rigid law if the castle is stretched vertically. Images (d) and (e) demonstrate the effects of reducing the amount of shearing (d) and rotation (e) while the castle is dragged to the right.

when manipulating images. Especially volume preservation as it is enforced by Hookes law is a physical phenomenon that contradicts shape preserving as-rigid-as-possible deformations. Volume preservation, on the other hand, can be avoided by prohibiting transversal contractions of the material being deformed. This is achieved by appropriately varying the Poisson ratio ν in equation 2, which defines the ratio of transverse contraction to longitudinal stretching in the direction of the force. In particular, we set $\nu = 0$, thus also avoiding any curvature in a direction perpendicular to the direction of stretching or bending. In Figure 3 we compare the deformation of a material under stretching with (b) and without (c) transversal contractions.

To enable as-rigid-as-possible transformations we further enforce the physical simulation to respect to an anisotropic material law. By doing so, we enable the user to flexibly control the resulting deformations by continuously varying between rigid and highly elastic materials within one image. The anisotropic material law is simulated by adding a scaling factor α for the off-diagonal elements in the stress tensor, yielding the rigid material law

$$\Sigma = E \begin{pmatrix} \mathcal{E}_{11} & \alpha\, \mathcal{E}_{12} \\ \alpha\, \mathcal{E}_{21} & \mathcal{E}_{22} \end{pmatrix}. \tag{3}$$

The modified material law allows for transformations minimizing the amount of shearing or rotation, respectively, and it can thus effectively be used to produce as-rigid-as-possible transformations. For a value of $\alpha = 1$ the material law is isotropic. By decreasing the value of α the internal stress is reduced into the direction of x_2 if strain is induced into the direction of x_1, and vice versa. Consequently, such a setting favors shearing instead of rotation. Contrarily, by setting α to a value larger than 1 rotations instead of shearing will be favored. The effects of different values of α are demonstrated in Figure 3 (d) and (e). Note, that generally the rigid material law does not preserve volume. However, a larger value of α leads to volume growth if shearing forces are applied, while small values of α tend to preserve volume in this case.

3.2. Quadrangular Elements

To improve simulation accuracy we have integrated quadrangular finite elements with bi-linear nodal basis functions into the multigrid approach (see Figure 4 for an illustration of how these elements deform under external forces). Quadrangular elements consist of 4 supporting vertices v_k, thus interpolating the deformation in the interior as

$$u(x) = \sum_{k=0}^{3} N_k(x) u_k \tag{4}$$

where

$$N_k(x) = c_0^k + c_1^k x_1 + c_2^k x_2 + c_3^k x_1 x_2.$$

u_k is the displacement of the k-th vertex, and the coefficients c_i^k are derived from $N_k(v_k) = 1$ and $N_k(v_i) = 0$ if $k \neq i$. From the definition of the nodal basis functions as given the element matrices with respect to their reference configuration can now be pre-computed.

By using quadrangular elements the following beneficial properties are achieved: Firstly, the overall number of elements is significantly reduced, resulting in a faster assembly of the global system matrix. This is important because co-rotated elements as used in our approach require the system matrix to be reassembled in every simulation frame. Secondly, semi-quadratic interpolation as supported by quadrangular elements improves the deformation behavior by increasing simulation accuracy.

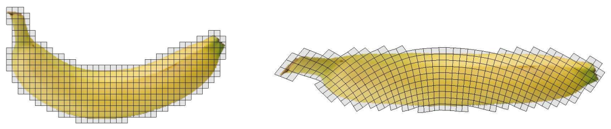

Figure 4. Quadrangular elements in their reference configuration and in a deformed configuration are shown.

3.3. Plasticity Simulation

So far, our discussion was restricted to the deformation of purely elastic materials. In image deformation, however, the user does not expect the image to move back into its reference configuration once the control handles are released. To avoid this behavior forces induced by the user are accumulated into a global force field, and the resulting displacements of grid points are computed just at the very end of the user interaction. This is advantageous because in order to account for the plasticity the system matrix does not have to be updated. Instead, we consider the linear system of equations in the form

$$K(u_{plastic} + u) = f_{plastic} + f$$

where $Ku_{plastic} = f_{plastic}$ are the plastic deformations computed so far, and f are the forces applied by the current user input.

3.4. Stiffness Update

In our opinion one of the most important features an image deformation tool should provide is the possibility to flexibly control the deformation of different parts of the image (see Figure 5). As our method is physics-based this can be achieved in a natural way by assigning specific stiffness values to parts of the image. As we aim at assigning these values interactively while the simulation is running, the simulation method must be able to instantaneously react on such changes.

Fortunately, it can be observed that element matrices do not have to be rebuilt if only the stiffness E of a single finite element is to be changed. This is because in equation (2) and (3) Σ is a multiple of E, and thus E can be factored out of the element matrix. This means that the element matrix only has to be scaled by the factor E_{new}/E_{old}, which can be performed in the reassembly of the global stiffness matrix at no additional cost.

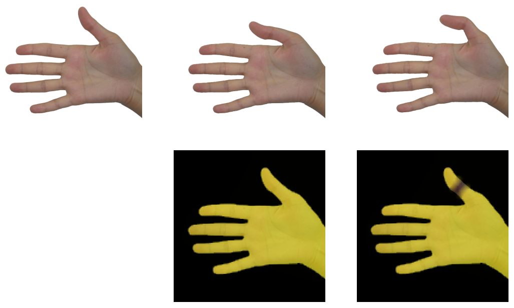

Figure 5. Different stiffness distributions enable flexible control over deformation effects. In contrast to deforming the hand with homogeneous stiffness (middle column), a joint is inserted by making the respective part soft (right column).

4. Image Segmentation

We utilize a segmentation algorithm for assigning specific properties to parts of an image. While it is generally possible to assign these properties using a painting tool on a per-pixel basis this can be a rather tedious and time-consuming task. Instead, we use an image segmentation algorithm to assign the properties on a per-object basis. The segmentation algorithm extracts objects from the image automatically by single mouse clicks or strokes on top of the object (see Figure 6). Once the object's boundaries are found the interior is filled with a user-defined property.

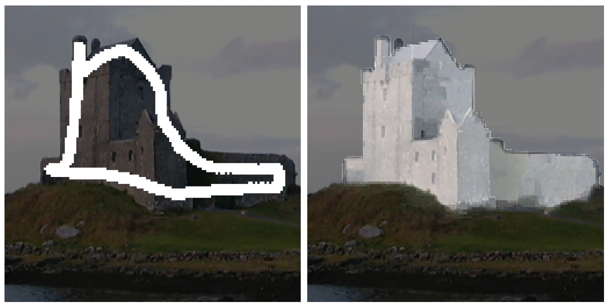

Figure 6. Left: A mouse-stroke painted in white on top of an object is shown. Right: The interactive segmentation algorithm automatically segments the castle shown in white within 30ms.

As stiffness is an important quantity to adjust the deformation behavior, and thus the kind of transformation performed, the primary use of the segmentation algorithms is to assign stiffness to parts of an image. By using an interactive segmentation algorithm stiffness can be assigned to objects in a single mouse click. The real-time segmentation algorithm we describe below even enables on-the-fly stiffness assignment during the deformation process (see Section 5.2), and it can thus be used flexibly to interactively control the resulting deformations. In addition, segmentation is used to select mesh vertices within particular image regions that should be fixed and to establish a rendering order of mesh elements by assigning object-specific depth values.

In general, any segmentation algorithm can be used for each of the aforementioned tasks. For our purposes, however, non-binary segmentation algorithms provide attractive properties since the segmentation results yield a

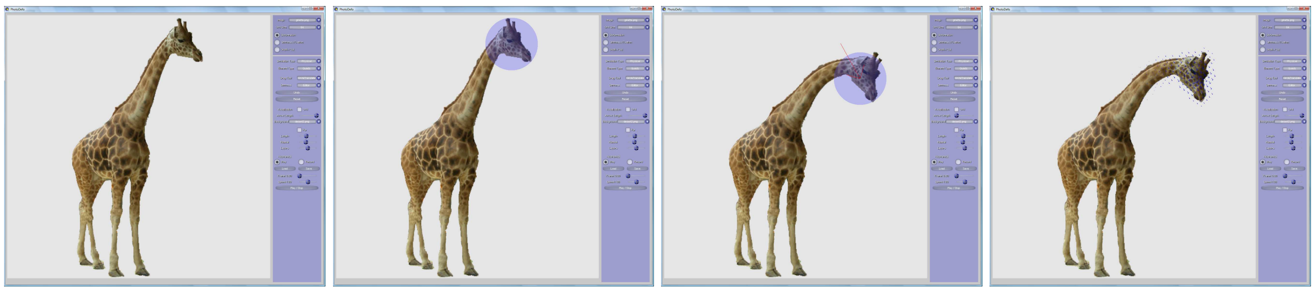

Figure 7. First: the original image. Second: the mouse cursor is moved over the image part to drag. The blue circle indicates the area of force injection. Third: the user drags the mouse to the desired target location. Fourth: the force field generated by the mouse drag is visualized.

smooth transition between the segmented object and the background. This is highly desirable for stiffness assignment because the segmentation result can be mapped directly to stiffness using a transfer function. In this way, we can avoid abrupt stiffness changes in the simulation grid at no extra cost. This feature is especially suited to generate visually pleasant deformations at boundary transitions.

The Random Walker algorithm falls into the category of non-binary segmentation algorithms. It is fast, easy to implement on the GPU and numerically stable. In the following we will give a brief overview of this algorithm and explain how to extend it to work on color images. For a more in-depth discussion of the Random Walker we refer the reader to the initial publication by Grady et al. [13].

The Random Walker Algorithm The algorithm is started on an image that contains a number of seed pixels indicating the object to segment. Given a random walker starting at every other pixel, the probability that this walker reaches one of the seeded pixels is computed. The probability that a random walker travels into a particular direction is determined by the image structure. The change in pixel intensity is a measure for the probability by which a random walker crosses over to a neighboring pixel. Therefore, there is a high likelihood for a random walker to move inside the segmented object, but low likelihood to cross the object's boundary. By multiplying probabilities computed along the paths from pixels to seed points yields a probability distribution representing a non-binary segmentation.

To use the Random Walker algorithm on color images we first convert RGB pixel values in the image to intensity values using the procedure proposed in [14]. Pixel values are first transformed to the Lab color space and the Euclidian length of the Lab vector length is used as intensity. Next, a function to express the probability is defined to map changes in image intensities to crossing weights.

The principle idea is to express random walks by a system of linear equations instead of simulating the random walks itself [17]. With the seed points as boundary conditions the problem can be posed as the solution of a sparse, symmetric, positive-definite system of linear equations.

To make this solution fast, and thus to enable the use of the Random Walker in real-time applications, numerical GPU-solvers can be used as demonstrated in [18]. In the current work we employ a GPU conjugate gradient method that enables segmentation of high-resolution images within milliseconds.

5. User Control

We use a simple paint-and-fill interface to assign stiffness, fixation, and depth to an image. The interface allows painting these properties on top of the image using different brush sizes. As users typically prefer to assign properties on a per-object basis, the segmentation algorithm is built into the interface and can then be used to segment objects from a single mouse click. Upon segmentation, default or selected parameter values are assigned to respective image structures.

5.1. Image Deformation Using Mouse Drags

Once all image properties are set the image is deformed using mouse drags (see Figure 7 for an illustration of this process). Therefore, we first determine the vertex in the simulation grid closest to the mouse position. From the direction and strength of the mouse drag we derive a force vector to be applied to the vertex. By using the mouse wheel the user can generate a radially expanded force field of arbitrary extend. It can be selected whether forces should decay smoothly with increasing distance to the center vertex or should be constant within the selected region. The direction of the force field is everywhere identical to the force applied to the center vertex.

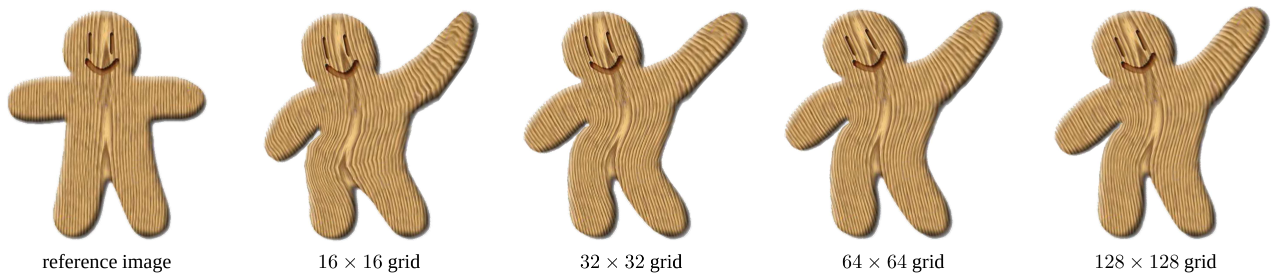

Figure 8. Interactive deformations of differently sized quadrangular grids are shown. Deformation times (including rendering of the deformed grids) are 1ms, 4ms, 17ms and 56ms from left to right, respectively.

5.2. On-the-fly Segmentation for Stiffness Assignment

Besides the assignment of properties to parts of an image in a pre-process we employ the Random Walker as a tool for on-the-fly parameter assignment. In this way, the user can flexibly select objects and move them around while the surroundings deform accordingly. On-the-fly assignment starts once the user clicks the initial drag point. Instantaneously, the segmentation algorithm considers the cell of the simulation grid that contains this point as a seed point for the Random Walker. The segmentation is then performed on the undeformed simulation grid, thereby assigning stiffness values to the segmented object. After the stiffness values are downloaded to the simulation engine, which updates the system matrices as described in Section 3.4, the user starts dragging the picked object while the background deforms. Note that by applying soft stiffness to the picked object, local object deformations are possible.

6. Results and Analysis

We have used Freeform image to manipulate a number of different images at varying resolutions. Our examples show snapshots from interactive sessions in which the user applied complex image deformations across a wide range of scales. All of our tests were run on an Intel Core 2 Duo CPU equipped with 2 GB RAM and a NVIDIA Geforce 8800 GTX graphics card. While the CPU cores are used to perform the physics-based simulation including matrix reassembling and multigrid calculations, the GPU is only used for the Random Walker algorithm. In all of our examples the communication of segmentation results and vertex displacements between the CPU and the GPU was below 5 ms.

In Figure 8 deformation results using differently sized simulation grids are compared to demonstrate the efficiency of our approach as well as the quality of the deformations depending on the grid resolution. Note, that some vertices of the simulation grid are fixed, which is accomplished by zeroing all entries in the row and column of the system matrix a fixed vertex belongs to. To keep the matrix at full rank, and thus to keep the solver stable, the diagonal elements in the intersection of the respective rows and columns are set to a default stiffness value that is used for all image parts unless changed by the user.

In particular at lower resolutions, it becomes apparent that the resulting displacement fields are only C0-continuous at element boundaries, a property that directly results from the finite element approach used. In principle this disadvantage can be resolved by using higher order interpolation schemes as proposed in [7] to transfer computed displacements to the pixel grid. On the other hand, we observe that the resulting visual artifacts are greatly alleviated at higher resolution grids, and—as indicated by the timings—these resolutions can still be handled interactively by our approach.

Table 1 gives timings for all parts of the proposed deformation method. The first column contains the numbers of quadrangular elements of the simulation grids used. The values in the second column indicate the amount of time required in every iteration to update the global system matrix. Next, we specify the time spent for updating the multigrid hierarchy from the current system matrix. The fourth column shows the overall time taken by one simulation step using a single thread. The following column demonstrates the performance gain when operations are interleaved by using multiple threads for “matrix update”, “multigrid update” and “multigrid solve” on the dual core PC. The time step used in the current implementation was set to 0.03 seconds. By performing a dynamic simulation using a small time step we avoid that local rotations between successive frames become very large, which otherwise makes element warping more difficult to be applied in a numerically stable way. The last column indicates the time required by the segmentation algorithm on the respective grids once the

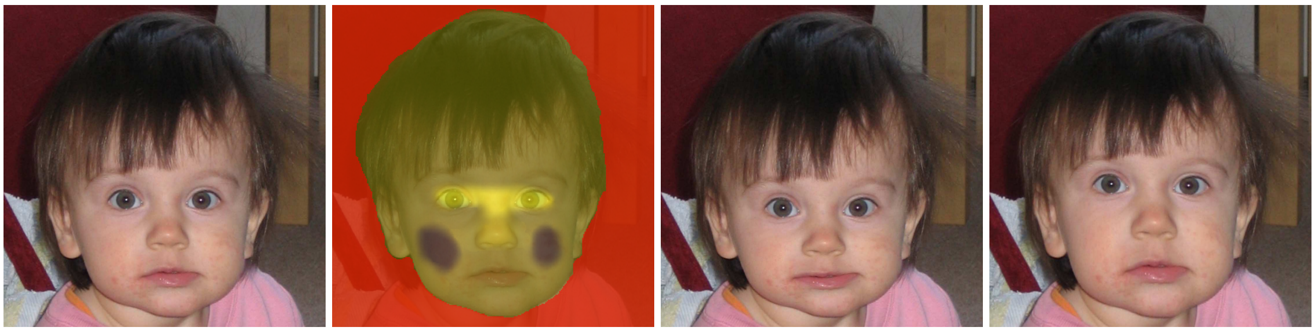

Figure 9. Our system enables easy-to-use face deformations. The second image shows the fixed vertices in red and the intensity of the yellow color illustrates the stiffness intensity (a 128×128 simulation grid was used).

# quads	matrix update	multigrid update	multigrid solve	simulation total (ST)	simulation total (MT)	random-walker
256	1	1	1	2	1	3
$1K$	3	1	2	6	4	8
$4K$	16	7	9	32	17	16
$16K$	47	21	25	93	56	35
$64K$	201	90	98	389	238	79
$256K$	904	411	416	1731	1064	182

Table 1. Timing statistics in [ms] for the main parts of the proposed deformation method. By exploiting both cores of the CPU to interleave numerical operations (column labelled MT) the performance can be nearly doubled.

user selects a new object.

The timing statistics indicate that our method performs considerably faster than the methods proposed by Igarashi et al. [16] and Botsch et al. [7]. While Botsch et al. solve for energy minimizing affine transformations of about 5K elements at one second using a Newton solver for the resulting non-linear system, the same number of elements can be physically simulated at roughly 40ms seconds on a single core using our extended multigrid. On the other hand, we observe that our method is clearly slower compared to the MLS method by Schaefer et al. [22], which can compute as-rigid-as-possible deformations for 10K elements in roughly 4ms. It is worth noting, however, that the performance of the MLS method greatly benefits from using pre-computed quantities that solely depend on the initial rest positions of the issued control points. In an interactive deformation session, where the user frequently selects arbitrary scales and regions to deform, these quantities have to continuously be re-computed and we expect the performance differences to become significantly smaller.

In terms of deformation options and user control our method distinguishes from previous approaches in several aspects. The most striking difference is that it adheres to a full physical model thus supporting a wide range of different types of global and local image deformations. In particular, our method provides a large repertoire of physics-based deformations ranging from as-rigid-as-possible to fully elastic and volume conserving deformations. The method is thus much more flexible than previous approaches in selecting the most appropriate deformation for specific image parts. By combining interactive stiffness editing and anisotropic material laws the user can flexibly use a magnitude of different deformations in one single image.

Figure 9 shows an example where different deformations have been applied to parts of the images. These deformations were selected to mimic real material values for specific parts at the one hand and to artistically manipulate other parts on the other hand. As can be seen, there is almost no restriction on the kind of deformations applied, yet still allowing for interactive manipulation by the user.

One additional aspect that makes our method distinct from previous approaches is the particular control mechanism it provides. Although entirely hidden by the user, every deformation is driven by external forces acting on the finite elements to be deformed. This kind of user control is different to mechanisms solely based on control points as used in [7, 16, 22], both with respect to user interaction and deformation effects. By constraining the object transformation via control points the user can specify very precisely to which position particular structures should be displaced. Our approach, on the other hand, is less precise in that the displacements are not specified directly but their movement due to external forces is computed. Consequently the displacements are less predictable and more effort is required to transform particular vertices to a desired position.

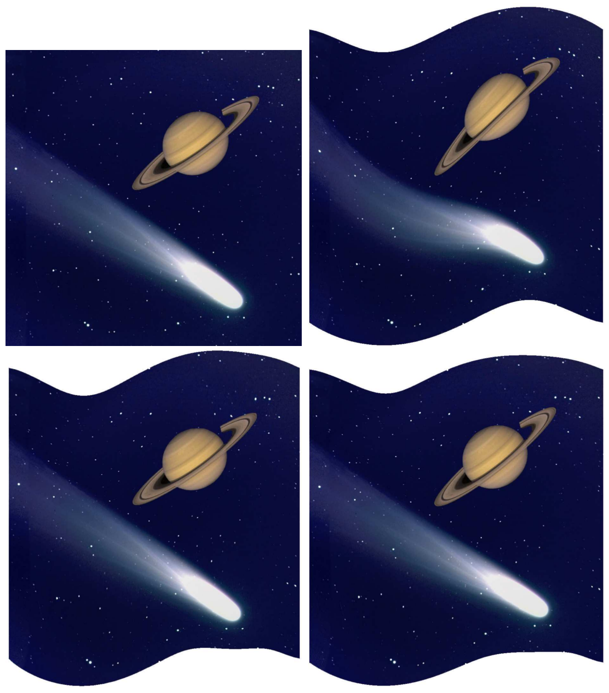

Figure 10. A wave force template is applied to the image. In the upper right image, a uniform stiffness distribution is used resulting in a very distorted planet. On the lower right, the planets were assigned a hard stiffness whereas the space was assigned soft stiffness. In the image on the lower left, the planets were fixed by vertex fixation. All examples use a 128×128 simulation grid.

Force-based control handles, on the other hand, have the advantage that arbitrary force fields can be used to deform the image. Figure 10 shows an example where a global force field simulating wave-like deformations was applied to an image, once with the planet having the same stiffness than the background, once with the planet fixed, and once with the planet having a very high stiffness. An additional application is shown in Figure 11, where a cartogram is generated by using multiple force-controls at once. Cartogram are maps in which area is not preserved, and instead another thematic mapping variable is substituted for area. By applying radially symmetric force fields in each state to either grow or shrink the area depending on such a mapping variable, the generation of cartograms can be performed in a very simple, controllable, and automatic way.

Furthermore, the use of forces fields enables realistic and fast keyframe animations by either interpolating between multiple global force fields or by specifying animation paths for the movement of local force fields over time (see Figure 12). Because in either case in-between displacements are computed by simulating the deformation from the interpolated or animated force field, the resulting displacements show a very smooth behavior and are in accordance with the underlying physical model.

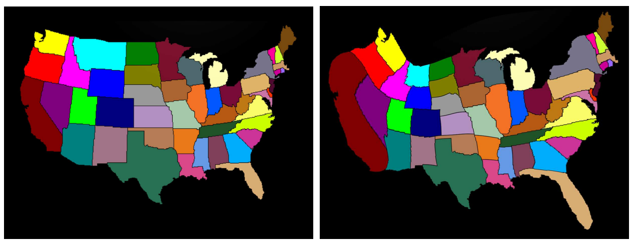

Figure 11. Using the proposed force templates, information visualization can be achieved. Here, each US state is scaled appropriately by using a radially symmetric force field to achieve the cartogram on the right (a 256×256 simulation grid was used).

7. Conclusions and Future Work

In this work, we have presented a technique for image deformation that enables flexible control over the kind of deformations to perform. We have shown that physics-based simulation combined with advanced numerical schemes is an effective and efficient tool for image deformation. Built upon such a simulation framework we have presented several novel extensions that enable intuitive freeform deformations including different kinds of transformations. By integrating application-specific material laws and by establishing stiffness as a powerful modeling parameter we can flexibly select between as-rigid-as-possible and plastic deformations. It is especially the combination of rigidity and elasticity that offers a variety of additional

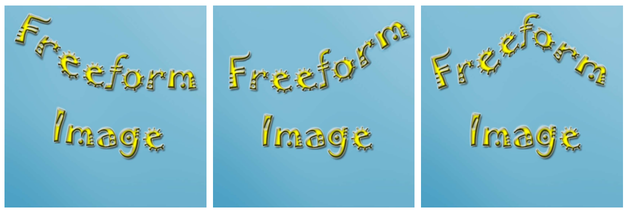

Figure 12. With just a few mouse-clicks, visual pleasant animations can be created. The images show two captured keyframes followed by a frame interpolated along an animation path.

deformation options in comparison to previous methods.

Due to the efficiency of the physics-based simulation engine, including highly optimized operations to assemble and update the system matrices, we were able to couple this CPU engine with a real-time segmentation algorithm running on the GPU. We have shown that on-the-fly segmentation and stiffness assignment are very powerful and useful modeling options, and we have described several ways the user can interact with the image being deformed.

In the future, we will investigate how to further improve the control mechanisms provided by our deformation tool. We have the feeling that in some cases control points or lines as used in some of the previous methods allow for a more precise control of the deformation. To achieve this, we will investigate how to solve static elasticity problems with mixed boundary conditions (forces and displacements) efficiently and how to adapt such schemes to the problem of image deformation.

Acknowledgements

We would like to thank Scott Schaefer for providing us the image of the wooden guy (Figure 8).

References

[1] M. Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, V19(2):105–114, May 2003.

[2] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[3] K.-J. Bathe. *Finite Element Procedures*. Prentice Hall, 2002.

[4] T. Beier and S. Neely. Feature-based image metamorphosis. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 35–42, New York, NY, USA, 1992. ACM Press.

[5] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. In *Proc. SIGGRAPH '03*, pages 917–924, 2003.

[6] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989.

[7] M. Botsch, M. Pauly, M. Wicke, and M. Gross. Adaptive space deformations based on rigid cells. In *Proceedings of Eurographics 2007*, 2007.

[8] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial, Second Edition*. siam, 2000.

[9] M. Bro-Nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proc. Eurographics*, 1996.

[10] O. Etzmuß, M. Keckeisen, and W. Straßer. A fast finite element solution for cloth modelling. In *Pacific Conference on Computer Graphics and Applications '03*, 2003.

[11] R. Gal, O. Sorkine, and D. Cohen-Or. Feature-aware texturing. In *Proceedings of Eurographics Symposium on Rendering*, pages 297–303, 2006.

[12] J. Georgii and R. Westermann. A multigrid framework for real-time simulation of deformable volumes. In *Workshop On Virtual Reality Interaction and Physical Simulation*, 2005.

[13] L. Grady and G. Funka-Lea. Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. In *ECCV 2004 Workshops CVAMIA and MMBIA*, pages 230–245, 2004.

[14] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. In *Proceedings of the Fifth IASTED International Conference on Visualization, Imaging and Image Processing*, pages 423–429, Benidorm, Spain, Sept. 2005. ACTA Press.

[15] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, 25(3):1126–1134, 2006.

[16] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1134–1141, New York, NY, USA, 2005. ACM Press.

[17] S. Kakutani. Markov processes and the Dirichlet problem. *Proc. Jap. Acad.*, 21:227–233, 1945.

[18] J. Krüger and R. Westermann. Linear algebra operators for GPU implementation of numerical algorithms. *ACM Trans. Graph.*, 22(3):908–916, 2003.

[19] S.-Y. Lee, K.-Y. Chwa, and S. Y. Shin. Image metamorphosis using snakes and free-form deformations. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 439–448, New York, NY, USA, 1995. ACM Press.

[20] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *ACM SIGGRAPH/EG symp. on Computer animation '02*, 2002.

[21] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH '99*, 1999.

[22] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 533–540, New York, NY, USA, 2006. ACM Press.

[23] A. Sheffer and V. Krayevoy. Shape preserving mesh deformation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, 2004.

[24] L. Shi, Y. Yu, N. Bell, and W.-W. Feng. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph.*, 25(3):1108–1117, 2006.

[25] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM Press.

[26] Y. Weng, W. Xu, Y. Wu, K. Zhou, and B. Guo. 2D shape deformation using nonlinear least squares optimization. *Vis. Comput.*, 22(9):653–660, 2006.