D-Snake: Image Registration by As-Similar-As-Possible Template Deformation

Zohar Levi, Craig Gotsman

Abstract—We describe a snake-type method for shape registration in 2D and 3D, by fitting a given polygonal template to an acquired image or volume data. The snake aspires to fit itself to the data in a shape which is locally As-Similar-As-Possible (ASAP) to the template. Our ASAP regulating force is based on the Moving Least Squares (MLS) similarity deformation. Combining this force with the traditional internal and external forces associated with a snake leads to a powerful and robust registration algorithm, capable of extracting precise shape information from image data.

Index Terms—Snake, Registration, MLS deformation

1 Introduction

Finding a specific object within an image, with as little human intervention as possible, has many applications. The advent of 3D volumetric "images" has amplified the challenge, but led to many important applications in the medical field.

A popular family of algorithms for edge detection is based on deformable models, which includes the so-called "snake". A snake is an active contour which evolves under the influence of internal forces emerging from the curve itself and external forces present in the image data. Snakes were first proposed by Kass et al. [1], causing deformable models to become one of the most active and successful research areas in image segmentation. A detailed review of all different types of deformable models can be found in [2].

A 2D snake is a curve $X(s) = [x(s), y(s)]$ that evolves in the spatial domain of an image I to minimize the energy functional

$$E = E_{int} + \kappa E_{ext} + \eta E_{def}$$

(κ and η are constant weighting factors), where the classic internal energy is:

$$E_{int} = \int_0^1 \frac{1}{2}[\alpha\|X'(s)\|^2 + \beta\|X''(s)\|^2]ds$$

(α and β are constants balancing the two components), and the classic external energy is:

$$E_{ext}(x, y) = -\|\nabla I(x, y)\|^2 \qquad ,$$

E_{def} is an additional deformation energy (∇ is the gradient operator). A snake that minimizes E must satisfy the Euler-Lagrange equation

$$\alpha X''(s) - \beta X''''(s) - \kappa\nabla E_{ext} - \eta\nabla E_{def} = 0 \qquad ,$$

which can also be viewed as a force balance equation:

$$F_{int} + \kappa F_{ext} + \eta F_{def} = 0 \qquad . \tag{1}$$

A snake is typically used for image *segmentation*. In this paper we aspire to extend the snake functionality to the more difficult task of *registration*. Segmentation means just identifying a shape in an image, while registration means capturing some of the detailed semantics of the shape. The semantics is usually conveyed through corresponding points between some generic *template* shape, in which these points have some semantic meaning, and the shape extracted from the image. For example, when analyzing a cardiac CT image, we don't want to capture only the general shape of a heart in the image, rather we want to identify each individual ventricle and valve within the heart. If these detailed features have been marked in the template, the registration procedure should identify them also within the CT image.

The classical snake is an evolving contour with no special significance attached to individual points on the contour. These points move freely, and typically there is no special relationship between them and the data. To achieve the registration objective we use a template which evolves like a snake, except that the classic internal energy is augmented by an As-Similar-As-Possible (ASAP) deformation energy. This energy regulates the general shape of the snake to keep it *locally* similar to a given template mesh, which in turn is similar to the shape we are trying to find in the image. We call this modified snake a *D-Snake* (Deformation-Snake). See Figs. 1-2 for a simple illustration of this important difference between the classical snake and our D-Snake.

There exist a number of effective state-of-the-art algorithms (e.g. [3], [4]) for registration of two 3D point clouds under deformation. The reader might wonder why it is not possible to simply adjust these algorithms to deal with image data. We believe that these algorithms would fail in an image environment. Were we dealing with synthetic images containing clear edges, we might

• *Z. Levi and C. Gotsman are with the Technion - Israel Institute of Technology.*
E-mail: zoharl@cs.technion.ac.il, gotsman@cs.technion.ac.il

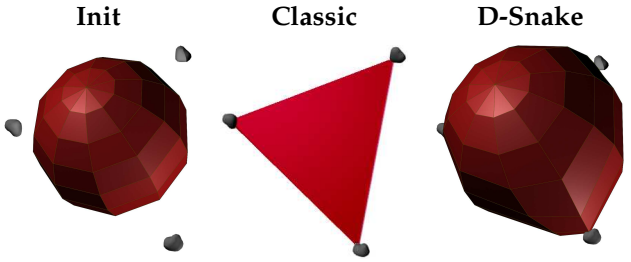


Fig. 1: Classical snake vs. D-Snake. (Left) A snake is initialized to a spherical template attracted to data consisting of 3 points, (Middle) Classical snake is greedily attracted to the points while trying to be as smooth (flat) as possible, (Right) Our D-Snake is also attracted to the data, but also tries to preserve the spherical template shape as much as possible.

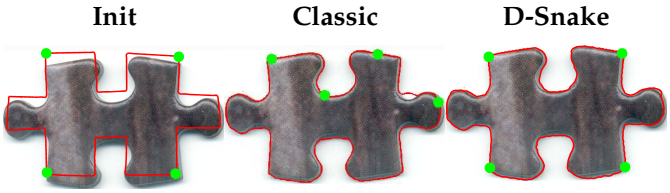


Fig. 2: Feature registration. (Left) Initial snake with a number of important feature points marked in green, and puzzle image data, (Middle) Classical snake is attracted to edges while losing the identity of the feature points, (Right) Our D-Snake is also attracted to edges while retaining the feature points.

be able to convert the edges to a point cloud and apply a point cloud registration algorithm. Unfortunately real-world images don't have a clear and concise edge map, and are usually cluttered with outliers, noise and false edges. The confidence level in an edge is much lower than a point in a point cloud, making the conversion impractical.

1.1 Contribution

This paper describes:

- A robust method for 2D/3D registration of a polyhedral template shape to an image.
- An operator for regulating a shape under the influence of an external data field.

2 Related Work

The main drawback of the classical snake is its total dependence on an edge "force field" to provide the external energy which will attract it to edges in the image. Since the edge force field typically provides very basic guidance - it merely points in the direction of the nearest edge - a snake typically has a hard time traversing through convoluted paths in a maze of edges and noise in an image to reach its mark. As a result, the snake suffers from two main weaknesses. The first is its sensitivity to initialization, i.e., the initial position and shape in the image where it starts to evolve from. The second is the difficulty to penetrate inside concavities, especially tunnels. The classic snake, for example, has to be initialized near the edges, because the edge attraction field does not cover the entire image, causing parts of the snake to not feel any force.

As a consequence, not surprisingly, most of the prior art on snakes is concerned with enhancing the external energy component of the snake model, such as balloons [5], distance potential force [6], Gradient Vector Flow (GVF) [7], and its improvements. The balloon force, for example, adds an expansion (or contraction) force to a snake, usually in the normal direction. The limitations of the balloon method are that the entire snake has to be initialized inside the object area (or outside), and determining the amount of force to apply is a delicate issue. Too much force will cause the snake to leak through holes or push past the target edges. A very successful way to extend the edge force field, which has proven itself over the years, is the GVF method [7]. It extends the initial generated external force field to other regions of the image by homogeneous linear diffusion, yielding a slowly varying vector field. One of the GVF extensions, the Dynamic Directional Gradient Vector Flow (DDGVF), which we take advantage of, will be elaborated on the next section. The VFC method [8] extends the edge force with a kernel matrix applied with a fast FFT. Although it might be faster than GVF, in practice the extended field contains "corners" and behaves less naturally than GVF.

Other variations on the snake try to improve the geometric properties of the snake curve. The B-Snake [9] constructs the snake from B-Splines, avoiding the need for the snake internal force. Topology is another issue. The snake starts with an initial contour which deforms, so cannot in general change topology. T-Snakes [10] try to overcome this limitation by allowing the snake to split or merge. A powerful alternative to the classical snake has emerged over the past few years - the Level Set Method (LSM) [11]. In this approach a 2D snake is considered the zero level set of a three-dimensional surface. This seems to be better suited for handling convoluted topology. In our application we felt more comfortable working with a snake, since the given template already determines the snake topology. Another reason is that while LSM works at pixel resolution, the snake provides control over the contour vertices, which we need to manipulate with our deformation force.

The extension of snake-based methods from 2D to 3D poses by itself a few challenges. Besides the obvious extra complexity of another dimension, which makes it harder to keep the snake well-behaved (e.g. not intersect itself), the component that maintains the discrete snake surface at the correct level of resolution (i.e. at the correct triangle density) is not obvious. Park et al. [12] address these challenges with a multi-resolution version of the snake. At each level of the image pyramid the snake is refined until convergence.

A related problem addressed in the literature is image registration. Unlike our problem which tries to register a contour to a 2D image or a triangulated mesh to a 3D image, image registration tries to register one image to another "reference" image. The common approach is usually to treat the reference image as a regular grid, and deform this to fit the other image. Early work on this was done by Thirion et al. [13], and more recent work by Sykora et. al [14]. The latter uses a regulating force somewhat similar to ours, and could benefit from our enhancements.

We should also mention the Statistical Shape Models (SSM) family [15] that tries to match a statistical model to an image. Probably the best known methods in that area are the Active Shape Models (ASM) by Cootes et al. [16]. A statistical model is constructed from a training set of correctly annotated images or models. The snake is then allowed to deform according to the distribution imposed by the statistical model. The annotation is done manually, and the training set size is of the order of magnitude of 50. Our algorithm requires only one template, and the variations are controlled by the ASAP energy.

2.1 DDGVF

Cheng et al. [17] propose a way to squeeze more information out of the image to improve the attraction of a snake to image edges by taking into account also the edge orientation. More specifically, characterizing an edge in an image by the largest intensity gradient in a local area, the direction of the gradient is the direction of the normal to the edge, and this can be used to guide the snake better. After computing the edge normals in this manner, we can decompose them to the three axes, and a snake vertex is attracted to edges consistent with its own normal. This is not to be confused with the balloon method, where a vertex can advance only in the normal direction. In the DDGVF method, a vertex moves as usual in all directions, but is blind to edges for which none of their normal components agrees with its own. This way a snake can avoid a lot of noise and outliers. It is also attracted to compatible edges inside a tunnel.

There are two drawbacks to the DDGVF method. The first is that the boundary of the shape which we try to capture in an image should be consistent with the direction of the intensity gradient. This is not always the case. For example, a lung boundary in a CT image goes from black to white on the entire boundary, where the black is always the interior of the lung. The liver, on the other hand, is usually not consistent. Orienting from inside to outside, near the lung the liver boundary goes from grey to black, while near the kidney it goes from grey to lighter shades. See Fig. 3. This inconsistency can mislead the snake. The second drawback is instability. The edge force influenced by the approximated normals is less smooth. Consequently the snake becomes unstable and needs a stronger internal force to prevent loops, an extreme case of foldovers, from forming. One solution

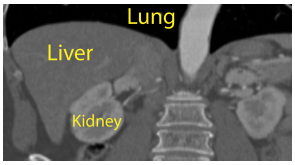


Fig. 3: The gradient directions on the boundary of the liver in a CT image are not consistent. When bordering the lung the gradient points inward, while when bordering the kidney the gradient points outward.

to avoid snake foldovers introduces collision detection forces [12], but this operation is quite costly. To overcome the stability problem we usually give more weight to the internal force to keep the snake together, but this can severely restrict the snake expansion, preventing it from penetrating tight corners. Our deformation force works well with the DDGVF and improves this weak point. It better constrains the snake trying to avoid foldovers, but still permits it to deform more freely and naturally than the classic tightening internal force. Fig. 4 illustrates this.

3 Algorithm Outline

After extracting the edges from the image and construction of the DDGVF, our algorithm proceeds in two phases. In the first phase the snake is initialized to a coarse template mesh, and this mesh evolves to capture the basic outline of the shape based *only* on the deformation and external forces. After convergence a second phase takes place where the snake is gradually refined in order to capture the finer details present in the data. The refinement process is interleaved with a remeshing procedure that keeps a necessary uniform distribution of the mesh vertices. The new vertices evolve based on the classical snake internal and external forces. See Fig. 5 for an outline of the algorithm, and Fig. 6 for a simple synthetic example in 3D.

3.1 Evolving the snake

We will give a brief overview of how the actual snake evolution occurs in 2D without reparameterization. The snake contour is discretized into N ordered points, $\{q_i\}_{i=1}^N = \{(x_i, y_i)\}_{i=1}^N$. The snake is initialized to some arbitrary position (we usually chose it to be the template position), and then starts an iterative evolution process. In each iteration, each vertex q_i is moved to a new position, which is the weighted sum of the new positions for q_i as given by each of the forces F_{int}, F_{ext}, and F_{def} alone for the current time step. The reason for this is

$$q_i(t+1) = q_i(t) + F \quad ,$$

where $q_i(t+1)$ is the new position of q_i in the end of the current iteration, $q_i(t)$ is the position from the previous iteration, and F is some force. More specifically

$$q_i(t+1) = q_i(t) + F_{int} + \kappa F_{ext} + \eta F_{def}$$

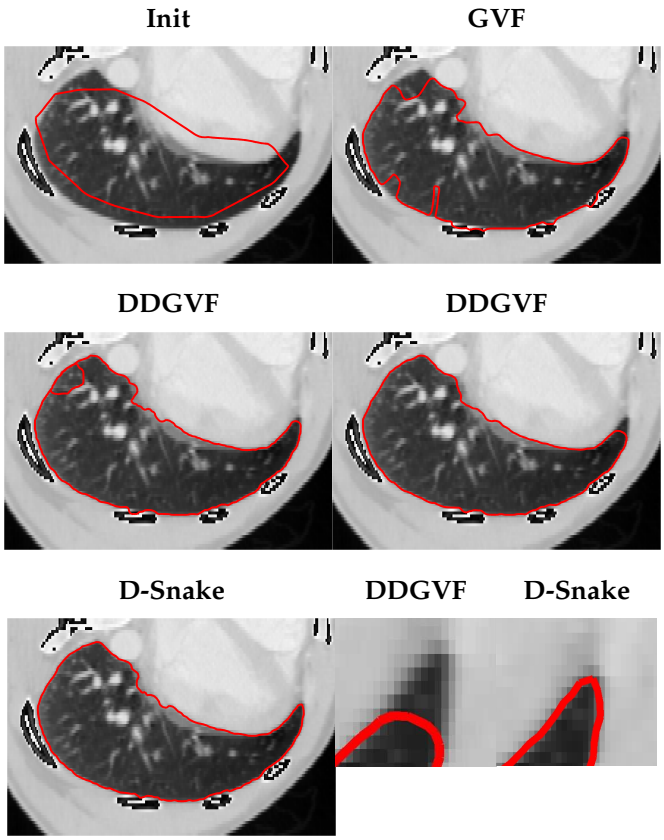


Fig. 4: 2D lung slice from CT image: (Top left) Snake initialized to template (Top right) GVF snake result, (Middle left) DDGVF snake result using a weak internal force. Notice loop formations, (Middle right) DDGVF snake result using a strong internal force. The snake cannot penetrate the corner on the right, (Bottom left) Our D-Snake, (Bottom middle) Zoom of (Middle right), (Bottom right) Zoom of (Bottom left).

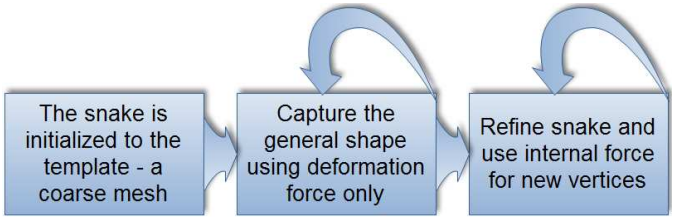


Fig. 5: Algorithm outline

$$= \frac{1}{1+\kappa+\eta}(q_i^{int}(t+1) + \kappa q_i^{ext}(t+1) + \eta q_i^{def}(t+1))$$

where $q_i^{int}(t+1)$, $q_i^{ext}(t+1)$, and $q_i^{def}(t+1)$ are the positions of q_i influenced by each of the forces alone. In the appendix of [1], equations (19) and (20) give the new position for the points that are influenced by the internal, and a basic external force, and our Eq. (5) gives the new position of the points influenced by the ASAP deformation force alone.

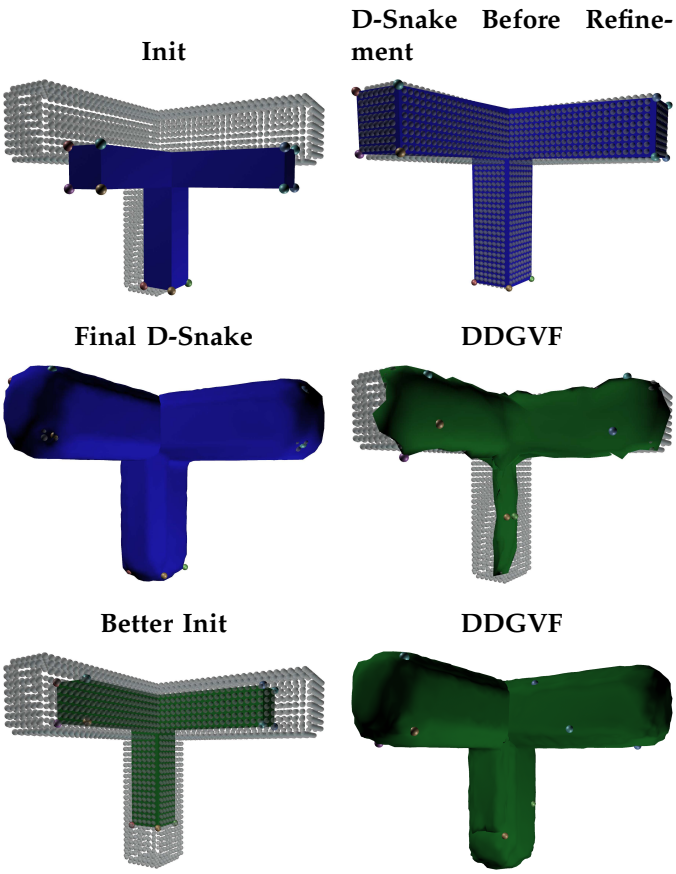


Fig. 6: Synthetic "Tetris" example. (Top left) Template of 20 vertices which serves as snake initialization. The point cloud is a visualization of a sample of the edges in the 3D voxel image, (Top right) Before the refinement step the D-Snake - still with 20 vertices - scales correctly, and fits the image well, (Middle left) Final D-snake. Due to the deformation force, the deformation vertices maintain their relative position in the shape throughout the process, and correspond to their counterparts in the template, (Middle right) DDGVF snake, initialized to the template on (Top left), but with a necessary refinement of additional 400 vertices, as the internal force cannot express itself with only 20 vertices. Since the snake is influenced by internal and external forces only, it cannot penetrate the shape well. Even when initialized partly inside the shape, there are no forces to attract it inside or across tunnels, (Bottom left) A better initial positioning of the template for DDGVF. The snake is scaled up, and positioned such that there is a part of it inside every edge force tunnel, (Bottom right) DDGVF, when initialized with the better template, finally captures the shape in the image, but fails to generate a correct correspondence.

4 The ASAP Deformation Energy

The main idea behind our algorithm is to use the As-Similar-As-Possible (ASAP) deformation energy to regulate the snake. Given an image, we would like to perform not just a simple segmentation, but a registration of a specific template shape to the image data. This will allow identifying features of the template in the image. The snake is initialized to the template shape, and as it evolves with respect to external forces, the deformation force regulates the snake by keeping it As-Similar-As-Possible to the template. This means that, locally, the evolving snake is related to the template by a similarity transformation. Let $\{p_i\}_{i=1}^N$ be the vertices of the template mesh, and let $\mathbf{q} = \{q_i\}_{i=1}^N$ be the vertices of the snake, which is initialized to the template. The ASAP deformation energy we minimize is:

$$E_{def}(\mathbf{q}) = \sum_{j=1}^{N} D_j(\mathbf{q}),$$

$$D_j(\mathbf{q}) = \min_{T_j} \sum_{i=1}^{N} w_{ij} \|T_j(p_i) - q_i\|^2.$$

The weights w_{ij} are of the form

$$\tilde{w}_{ij} = ||p_i - p_j||^{-2\alpha}, \qquad w_{ij} = \frac{\tilde{w}_{ij}}{\sum_i \tilde{w}_{ij}} \tag{2}$$

with α being a *fall-off* parameter controlling how strongly the deformation at p_j is influenced by the point p_i. Note that while $\tilde{w}_{ij}$ are symmetric, w_{ij} are not, because of the local normalization. We normalize the weights, so that p_j influences p_i proportional to the importance of p_i in its neighborhood, and not by the original absolute weight w_{ij}. T_j is restricted to the group of similarity transformations of the form

$$T_j(x) = \mu R x + t \qquad ,$$

where R is a rotation matrix, t is the translation component, and μ is a positive scaling factor.

Our ASAP deformation energy was inspired by the Moving Least Squares (MLS) deformation [18]. Given corresponding source and target point clouds, MLS deforms a point x contained in the source domain using the similarity transformation that best explains the mapping between the source and target points in the close vicinity of x. The weight function ensures that the deformation focuses on the neighborhood of x. Analogously, our energy measures the similarity between two shapes. It computes how similar local neighborhoods are between two shapes, and sums the differences. To measure a neighborhood similarity, we compute at each vertex the similarity transformation which best explains (in terms of weighted squared distance) how the neighborhood of a source shape is deformed to the corresponding neighborhood in the target shape. D_j represents the local similarity energy at vertex p_j, and E_{def} is the sum of all the local similarity energies. Note that in the special case

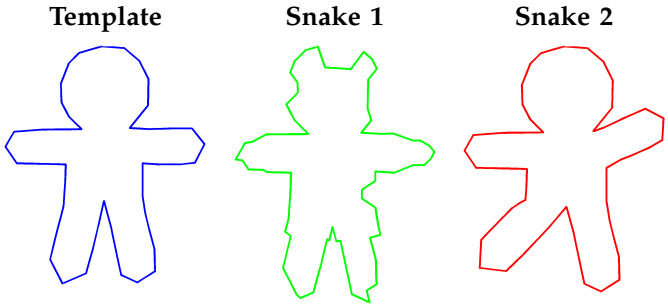


Fig. 7: Measuring the deformation energy between template and snake: (Left) Template, (Middle and right) Possible deformed snakes. Measuring the energy with one global transformation ($w_{ij} \equiv 1$) favors snake 1 over snake 2, while the weighted (by (2)) ASAP energy favors snake 2 over snake 1.

$w_{ij} \equiv 1$, E_{def} reduces to an energy which is minimized by a single global similarity T:

$$E_{def}(q) = \sum_{i=1}^{N} \|T(p_i) - q_i\|^2, \tag{3}$$

The main advantage of the local similarity energy over the global similarity is that it favors $\{q_i\}$ which are an articulated movement of $\{p_i\}$. See Fig. 7 for an example.

The ASAP deformation energy by itself is not really useful. Setting the snake vertices $\{q_i\}$ to any global similarity transformation of the template vertices $\{p_i\}$ would cause the energy to vanish trivially. In practice the ASAP energy must be used in conjunction with a number of other competing energies which would prefer to avoid a global similarity, such as the external energy.

The external energy E_{ext} depends on the image data, so it is not possible to minimize it in one shot (certainly not in closed form). It requires an iterative scheme, such as the gradient descent method. Towards this end, we incorporate the minimization of E_{def} into the snake evolution paradigm, by reducing the energy step by step, vertex by vertex, with a coordinate descent approach. As in Eq. (1), we combine the different forces operating on a vertex, the ASAP deformation force being one of them. The external force causes the snake shape to better fit the data, while the internal force maintains some smoothness, and the ASAP deformation force resists and regulates the snake shape, maintaining its resemblance to the template shape.

Our objective then is to describe a method that minimizes E_{def} gradually, while keeping the snake at each step similar to the template. We need to keep in mind that the minimization process of the total energy E could reach a local minimum due to E_{ext}, and E_{def} would probably not vanish.

4.1 Exact solution for 2D

As discussed in the previous section, in order to minimize E_{def} we use a coordinate descent approach in

the spirit of the basic snake evolution paradigm. At each iteration we determine where each snake vertex q_k should move to in order to minimize E_{def}. Keeping all the snake vertices fixed except q_j fixed, there is an explicit formula for the best local similarity T_j at vertex p_j [18].

Let:

$$p_j^* = \sum_{i=1}^{N} w_{ij} p_i, \qquad q_j^* = \sum_{i=1}^{N} w_{ij} q_i, \qquad \hat{p}_i = p_i - p_j^*,$$

$$B_{li}^j = \mu_j \cdot w_{lj} \begin{pmatrix} \hat{p}_l^T \\ -\hat{p}_l^{\perp T} \end{pmatrix} (\hat{p}_i \quad -\hat{p}_i^{\perp}), \tag{4}$$

$$\mu_j = \frac{1}{\sum_i w_{ij} \hat{p}_i^T \hat{p}_i},$$

where $\perp$ is the 2D orthogonal operator $(x,y)^{\perp} = (-y,x)$, and B_{li}^j are 2×2 matrices. Then

$$\begin{aligned} T_j(p_i) &= \textstyle\sum_{l \neq j} B_{li}^j (q_l - q_j^*) + q_j^* \\ &= \textstyle\sum_{l \neq j} B_{li}^j (q_l - q_j) + q_j, \end{aligned}$$

The last equality follows from the observation that $q_j^* = q_j$, since q_j has an infinite weight. The summation over l in this section is from 1 to N excluding j. In the last equations $l = j$ was omitted from the sum, due to $q_j - q_j^* = 0$, which overrides the infinity weight.

Now that we have T_j, we may proceed to find the solution to the normal equations $\nabla_{q_k} E_{def} = 0$. Using matrix calculus we differentiate each local energy according to q_k,

$$\nabla_{q_k} D_j = 2 \sum_{i \neq j} w_{ij} (g_i^{jk})^T (T_j(p_i) - q_i).$$

where

$$g_i^{jk} = \begin{cases} B_{ki}^j - I & k = i \neq j \\ I - \sum_{l \neq j} B_{li}^j & k = j \\ B_{ki}^j & \text{otherwise} \end{cases},$$

leading to

$$\begin{aligned} \nabla_{q_k} D_j = 2 q_j \sum_{i \neq j} w_{ij} g_i^{jk} (I - \sum_{l \neq j} B_{li}^j) + \\ 2 \sum_{i \neq j} q_i (\sum_{l \neq j} w_{lj} g_l^{jk} B_{il}^j - w_{ij} g_i^{jk} I) \qquad . \end{aligned}$$

Since $\{p_i\}$ are template points, they are fixed during the snake evolution, thus B_{li}^j and g_i^{jk} are constants. Writing $\sum_j \nabla_{q_k} D_j = 0$ results in a single linear equation in q_k.

We can represent this scheme which evolves a vector of points at time t, $q(t)$ (a column stack of x and y coordinates of $\{q_i\}$, for example), to a new position $q(t+1)$, by a matrix A, which depends only on the template p:

$$q(t+1) = Aq(t). \tag{5}$$

The operator A encodes the local shape defined by $\{p_i\}$ up to a similarity, and can be applied successively to any vector q representing any other shape.

From basic numerical analysis we note that updating $q(t)$ iteratively (by multiplying by A) represents a Jacobi method, which generates $q(t+1)$ based only on $q(t)$. This, however, does not guarantee convergence to the correct q, as the energy is not guaranteed to always decrease. It updates a points position using the old positions of the neighbors, which might have changed in the meantime, making the calculation not as accurate as it could be, and this will mislead the global optimization, leading to a possible increase in energy. To apply the coordinate descent method that would give the correct solution, we need to update $q_j(t+1)$ based on the vertices we have updated so far in this iteration, $q_{i<j}(t+1)$, and the vertices from the previous iteration $q_{i>j}(t)$. Following the Gauss-Seidel method, if $A = L + D + U$ is the decomposition of A to lower triangular, diagonal, and upper triangular matrices respectively, we have

$$q(t+1) = A_2 q(t), \qquad A_2 = (I - L)^{-1}(D + U).$$

Applying A in this manner transforms q gradually, until it converges to a shape similar to $\{p_i\}$. At each step of the iteration we are guaranteed to decrease the energy $E_{def}(q)$.

We may control the speed of this process by using a matrix power $A_3 = (A_2)^m$ or a dampening factor $A_4 = (1-\lambda)I + \lambda A_2$, $\lambda \in [0,1]$.

4.2 Exact solution for 3D

The computation of the similarity transformation which minimizes the MLS in 3D involves a SVD operation (see Appendix), and differentiating E_{def} becomes more challenging. Horn [19], [20] offers a better closed form for the best similarity transformation, involving eigenvectors instead of the SVD. However, this requires dealing with roots of a cubic polynomial. Thus no matrix A, analogous to the 2D case, exists. One way to alleviate this is to linearize the 3D rotation matrix in the spirit of [21], but then the solution would be sensitive to large rotation angles.

Following Sorkine and Alexa [22], we used an alternative method to optimize the energy, which still *guarantees* convergence. The method is iterative and alternates between two phases. In the first phase we fix the vertices and compute optimal rotations and scaling factors. In the second phase we fix the rotations and the scaling factors, and optimize the vertex positions. In the second phase, differentiating the energy with constant rotations and scaling factors results in a set of linear equations for the positions of the vertices $\{q_k\}$:

$$\begin{aligned} q_k = -\frac{1}{2} \sum_{i=1}^{N} [w_{ik}(\mu_k R_k (p_i - p_k^*) - q_i) - \\ w_{ki}(\mu_i R_i (p_k - p_i^*) + q_i)] \end{aligned}$$

Since we use the snake paradigm, when we optimize for a new position for one vertex, all the other vertices are frozen.

4.3 Approximate solution for 2D

Using a large neighborhood in the MLS scheme can improve the results, but also implies more computation, thus is much slower. To accelerate the algorithm, we describe an approximation scheme, which involves less computation and is much faster than the exact algorithm. Unfortunately, convergence is not guaranteed, although if the target surface is close enough to the template, our experiments show that the scheme converges. We first outline the scheme in 2D. In practice, in 2D this scheme replaces the matrix A with a different matrix, so provides no advantage. In 3D, though, it results in a faster algorithm.

Consider a specific vertex q_j: We may decompose E_{def} to its *self-energy* D_j and a sum of energies due to the other vertices:

$$E_{def} = D_j + \sum_{i \neq j} D_i$$

We propose to move q_j, while considering only its self energy D_j, although by doing that we might inadvertently increase the energy due to the other vertices. We will give some intuition for this. The idea is to make each local neighborhood of the snake As-Similar-As-Possible to the template in order to converge to a global similarity. It also conforms with our objective that the energy function favor local neighborhoods in order to deal correctly with articulated poses. The following lemma suggests a way to minimize the self-energy D_j when moving q_j.

Lemma (without proof) To minimize the self-energy D_j, the point q_j should move to

$$\hat{q}_j = \hat{T}_j(p_j), \tag{6}$$

where $\hat{T}_j$ is the similarity transformation which minimizes

$$\bar{D}_j = \sum_{i \neq j} w_{ij} \|\hat{T}_j(p_i) - q_i\|^2.$$

Further development results in:

$$\hat{q}_j \quad = \quad \sum_{l \neq j} B^j_{lj}(q_l - q^*_j) + q^*_j \qquad ,$$

and this time $q^*_j \neq q_j$ which does not participate in the transformation.

4.4 Approximate solution for 3D

Extending the approximation scheme outlined in the previous section to 3D, updating each self-energy one at a time, is straightforward. This still requires the SVD method for the best 3D similarity, as described in the Appendix.

We cannot construct a linear operator to reflect the iteration scheme as in the 2D case, since the similarity transformations do not form a linear subspace of the space of linear transformations in 3D. Still, a large portion of the calculations for updating a vertex q_j can be pre-processed in the initialization step since the template does not change. See Fig. 8, which illustrates applying the ASAP approximation scheme on a mesh. This results in perfect convergence to a global similarity of the template.

5 Remeshing

One important attribute a snake should maintain is a high vertex density, one that is sufficient to cover the edge force field and penetrate into tunnels. We maintain this property, while preserving a high quality mesh, using a remeshing algorithm incorporating the following four stages:

5.1 Uniform distribution

Keeping a uniform distribution of vertices across the mesh is important for a high quality mesh and the snake progress. A simple way to achieve this is an operation similar to Laplacian filtering. The idea is to perform local 1-ring operations that move each vertex to the center of its neighbors on the surface. To avoid foldovers we first parameterize the 1-ring neighborhood (triangles) to a convex polygon lying on a circle embedded in a 2D plane, by arc-length placement, and move the designated vertex to the center of the circle. We then project this vertex back to the surface (the triangles on the 2D plane are mapped to the mesh triangles).

5.2 Mesh connectivity

To achieve a more regular connectivity structure and equilateral triangles, in conjunction with the previous step we perform a Delaunay triangulation. First we partition the mesh into a number of large patches having disc topology - a usual requirement for plane parameterization which allows to handle models having high genus, such as the vertebra model. To achieve this we applied a relatively simple algorithm, which recursively splits patches in two, as long as they are not homeomorphic to a disc. The split is done using Breadth-First-Search (BFS), which marks triangles belonging to one group until it covers half of the patch size. The remaining unmarked triangles then belong to the other group.

In the next step we parameterize each of the patches to the plane using standard parameterization techniques with mean-value coordinates [23]. The mesh boundary is mapped to a circle, and a linear system is solved for the positions of the interior vertices.

Finally we perform Delaunay triangulation on the parameterized vertices in the plane, and adopt this new connectivity structure on the original 3D points.

5.3 Refinement

The refinement process is relatively straightforward. For each triangle, which has an edge longer than a certain threshold, a vertex is added in its barycenter, and the

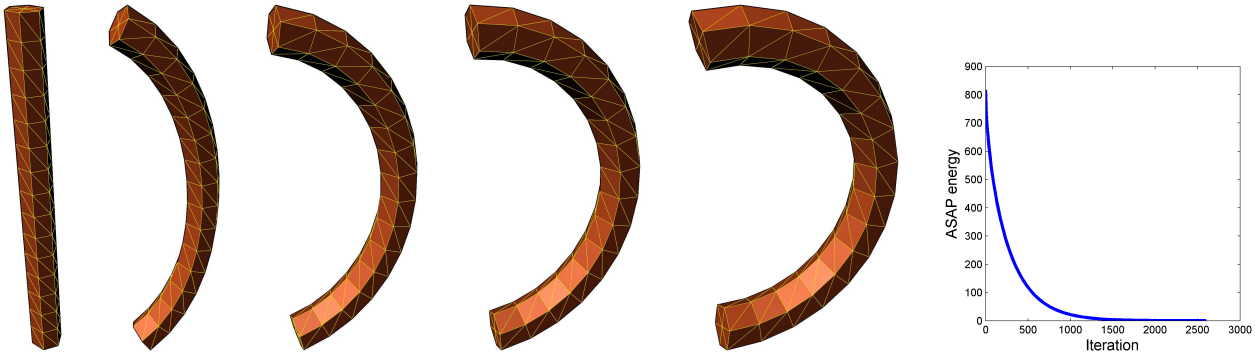


Fig. 8: A few snapshots of applying the 3D approximation scheme on a D-Snake having the shape of a bar on the left, evolving into the template having a 'C' shape on the right. The graph shows the convergence of the ASAP energy to zero.

triangle is subdivided to three triangles. We found that there is no need for anything more sophisticated, since as the snake continues to evolve, the regularity step improves the triangulation quality. See top row of Fig. 9.

5.4 Simplification

After the refinement step and the uniform parameterization, islands of triangles which are overly small may form, which may be redundant. Although an oversampled surface won't harm the smoothness and shape approximation qualities of the surface, the runtime of a single iteration depends on the mesh complexity, and an oversized mesh can slow it down dramatically. To address this problem, we take two steps. One is to limit the minimal edge size of the mesh to the size of a voxel edge. The second is to collapse triangles, which are too small in relation to their curvature [24].

6 Algorithm Flow

The D-Snake is initialized to the template, preferably a coarse mesh. Starting with a relatively low resolution mesh helps the D-Snake preserve the template shape, capture better the general shape of the object in the image, and converge rapidly. Doing the same using just the traditional internal force could cause severe malformations in the result. See Figs. 6, 9, 10.

After the D-Snake evolves to a stable configuration, we start to refine it by gradually lowering the minimum edge length threshold at each iteration. This causes triangles to subdivide, adding new vertices to the mesh. These new vertices continue to evolve based only on the classic internal force. As a result, the D-Snake reaches its highest resolution gradually. This is better than jumping to full resolution in one iteration, since the new vertices would be quite independent, and this could lead to large fissures in places where the vertices are attracted to the edges. By refining the mesh gradually we spread the shape regularization influence originating in the template vertices. From the point we start to refine the D-Snake, we apply the parameterization and simplification steps every few iterations. See the accompanying video for a visual illustration of the complete process.

7 Implementation Details

Unlike the 2D case, in the 3D case each vertex is treated separately during the iterations. Yet a large portion of the process can still be precomputed, since the template is fixed. Thus the computation per vertex is reduced to a multiplication of a matrix whose dimension is three times that of the neighborhood size, and a SVD of a 3×3 matrix.

We used the following parameters for the lung, and similar values for the other models: $\alpha = 0.55$, $\kappa = 0.4$, $\eta = 0.5$.

8 Experimental Results

Most of the calculations were performed mainly in MATLAB with no special optimizations, running on an Intel T9400 2.53GHz laptop. For the computational geometry algorithms we used CGAL [25]. All 3D examples were generated with the approximate algorithm, as described in 4.4. As an example we will give the run times for the input volumetric image of a left human lung having resolution $230 \times 300 \times 80$. The preprocessing step involves construction of the DDGVF map, which takes 40 seconds. The D-Snake was initialized to the template consisting of 309 vertices, and this increased, after 500 iterations, to 7,000 vertices. 300 iterations took 2.5 minutes, while 500 iterations took 16 minutes. The quality of the final mesh is high, and the triangulation is almost fully regular. See Fig. 9 top right.

Since the image contains significant noise, the D-Snake is sensitive to initialization, and the closer to the edges it is initialized - the better. The ASAP deformation force also helps to overcome that. In Fig. 6, 9, 10 we compare the DDGVF snake which uses only the standard internal force, and our D-Snake which uses the ASAP deformation force. The bronchi in the lung create false edges which are closer to the top of the D-Snake than to the upper wall of the lung. Thus the D-Snake is attracted to them first. The ASAP deformation force better guides the

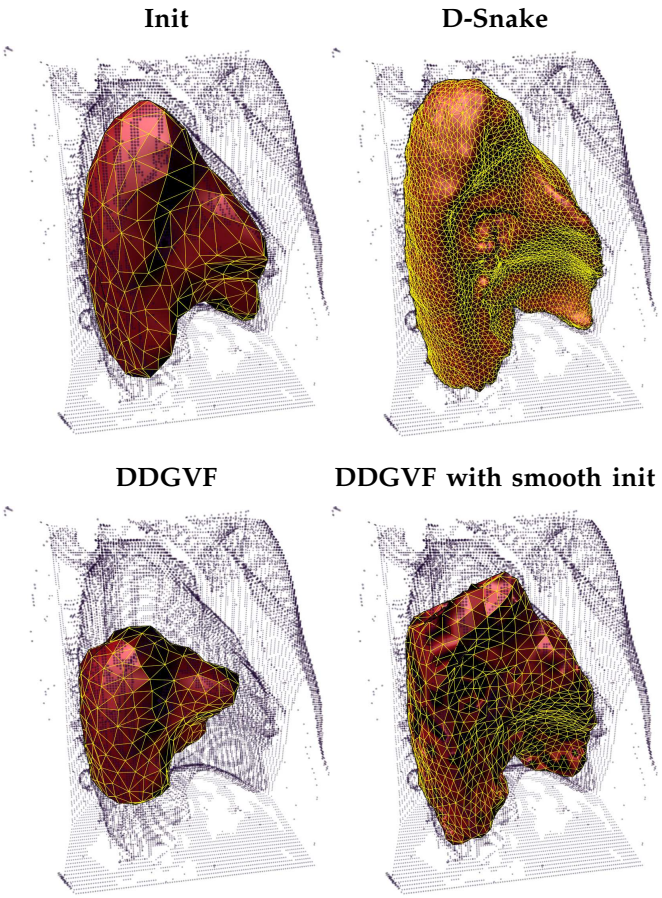


Fig. 9: Registration to a CT image of a left human lung. (Top left) Template lung, which serves to initialize the snake process. The point cloud gives a rough visualization of the image edges position, (Top right) D-Snake after 500 iterations, (Bottom left) State-of-the-art DDGVF snake. The initial template mesh is too coarse, so the internal force shrinks the shape, (Bottom right) A snake using the same forces as (Bottom left), but initialized to a refined version of the template to make it easier for the internal force to retain some shape. Nonetheless, as opposed to D-Snake which has deformation vertices that maintain the shape and push it beyond noisy edges, DDGVF cannot pass the noisy edges created by the bronchi in the lung.

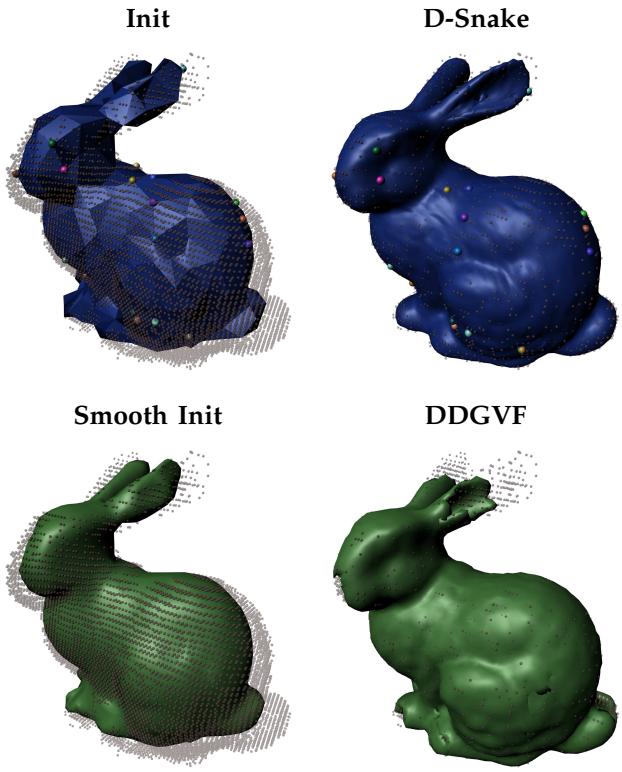


Fig. 10: CT scan of the Stanford Bunny. (Top left) Template containing 358 vertices, which serves as initialization for snake. The template is a simplified version of the original model which underwent light anisotropic scaling and ear twisting, (Top right) Final D-Snake after 500 iterations, (Bottom left) Refined version of the template (one level of subdivision). (Bottom right) DDGVF snake result when using bottom left refined template.

D-Snake, forcing it to expand upwards in order to keep its shape As-Similar-As-Possible to the template. Fig. 19 compares two slices of the final snake to the template. Fig. 20 shows the color-coded correspondence between some of the template vertices and their counterparts in the final D-Snake after the deformation process.

We also compared the D-Snake to the GVF snake under conditions of outliers, noise, and perturbation. We performed the tests on a synthetic 3D image of a star, having dimensions 80x80x80; see Fig. 13-18. In this experiment we compared the D-Snake to the GVF snake, since the latter performed better than the DDGVF snake, probably due to the normals smoothness which contributes to instability in the DDGVF snake. In order for the test to be objective, we used the same GVF external force in the D-Snake. As in the rest of the paper, both snakes start with the same shape - the template of the D-Snake - where the GVF snake is refined from 26 vertices to 386 vertices before the first iteration. As we discussed previously, this refinement step, performed during initialization, is crucial for the derivative calculation and stability of snakes that relies only on the internal force. We used $\alpha = 0.5$ and $\kappa = 0.2$ for the GVF snake. Using a larger κ results in a stronger external force, and the GVF snake then becomes unstable and intersects itself. On the other hand using a smaller κ results in a stronger internal force, which limits the snake flexibility, and prevents it from reaching all the corners of the star. We determined empirically that $\kappa = 0.2$ is the best compromise. For D-Snake we used $\kappa = 0.1$ and $\eta = 0.5$. For the D-Snake we show only the result of the first important step, where the snake fits itself to the general shape of the image, before starting the refinement process.

As in the rest of the paper, both snakes were initialized

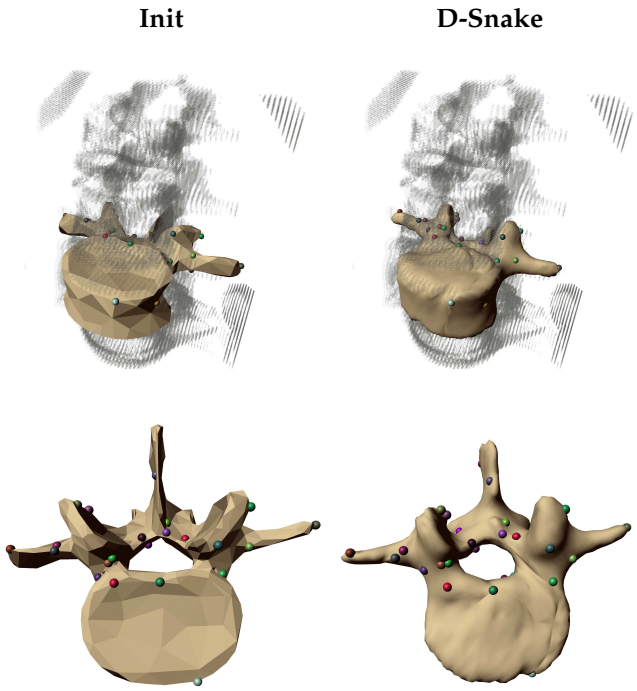


Fig. 11: Vertebra. (Left) D-Snake is initialized to a coarse template (539 vertices). (Right) Final D-Snake after 300 iterations (two minutes) preserves topology and obtains a good registration.

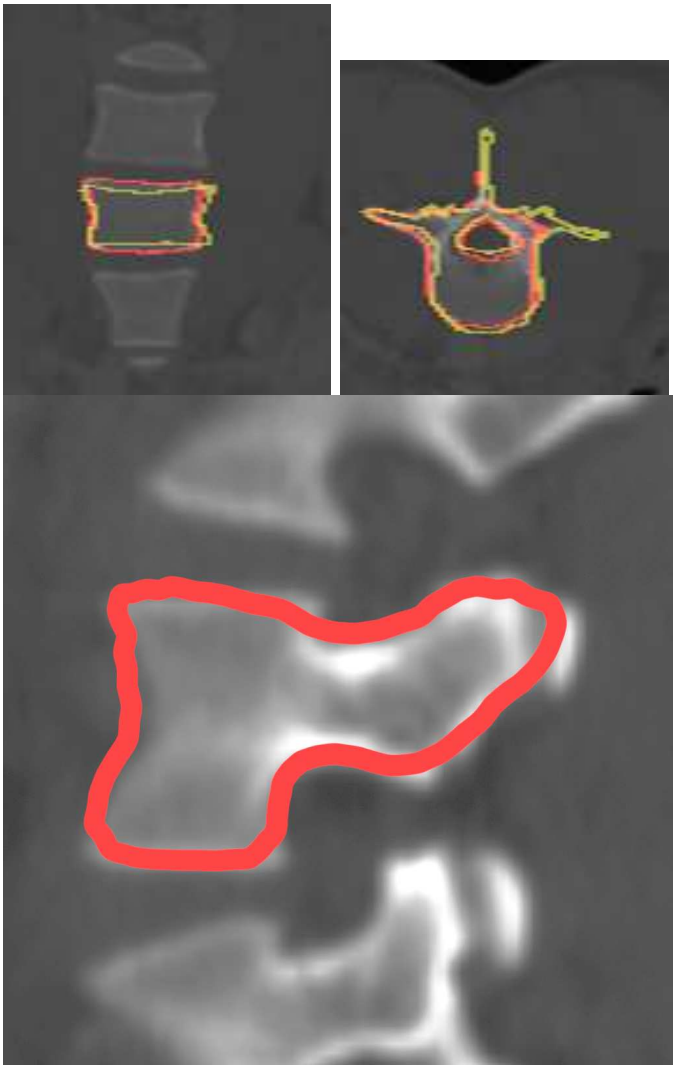


Fig. 12: Slices of the vertebra image with the initial (template) snake in yellow, and the final D-Snake in red.

to the same position shown in Fig. 13. Like before, the cloud of points in the bottom-left image is a clean illustration of the shape in the image, which is based on the edges present in the image. Fig. 14 shows the result of the two snakes on a clean image. GVF succeeds in reaching all the corners, and except for a few artifacts the resulting mesh is reasonable. The corners of the GVF snake are rounded due to the internal force, while D-Snake allows sharp corners for deformation vertices.

In Fig. 15 we added 1% of random strong outliers to the image, such that the images still stay binary (with two colors). These few outliers disrupt the external force field of the GVF and challenge the snake. In Fig. 16 we see that 1% of outliers attract the GVF snake, and prevent it from detecting the entire shape. We can also see that the strong external force that we used for the GVF snake, allowing it to stretch and spread better, was actually too strong, and more artifacts can be seen. D-Snake was able to overcome 1% outliers, but failed for 3% outliers. Due to a strong deformation force, D-Snake failed gracefully and still retained a star shape.

In Fig. 17 we tested the snakes on images with Additive White Gaussian Noise (AWGN). The source image is an 8-bit (0-255) grayscale, with background value 170 and foreground value 85. Addition of AWGN with $\sigma = 30$ confuses the GVF snake, while D-Snake succeeds for $\sigma = 30$, but fails for $\sigma = 35$. In Fig. 18 we perturbed the pixels of the star image. Each foreground pixel was perturbed in a random direction by a random distance up to some radius r. The resulting image remains binary. The GVF snake fails for $r = 2$, while D-Snake succeeds and fails only for $r = 4$.

The video accompanying this paper illustrates the various stages of our registration algorithm and its results.

9 Discussion and Conclusions

We have presented the D-Snake - a method which incorporates space deformation techniques from computer graphics and snake-based evolution ideas from computer vision to solve the problem of mesh registration to 3D volume data. We introduced the MLS-based similarity operator for regulating a shape in a dynamic environment, and showed how it helps the snake to better capture a shape defined by a dataset while preserving features present in the template.

While very effective, the MLS-based deformation energy used in our D-Snake framework can be replaced by

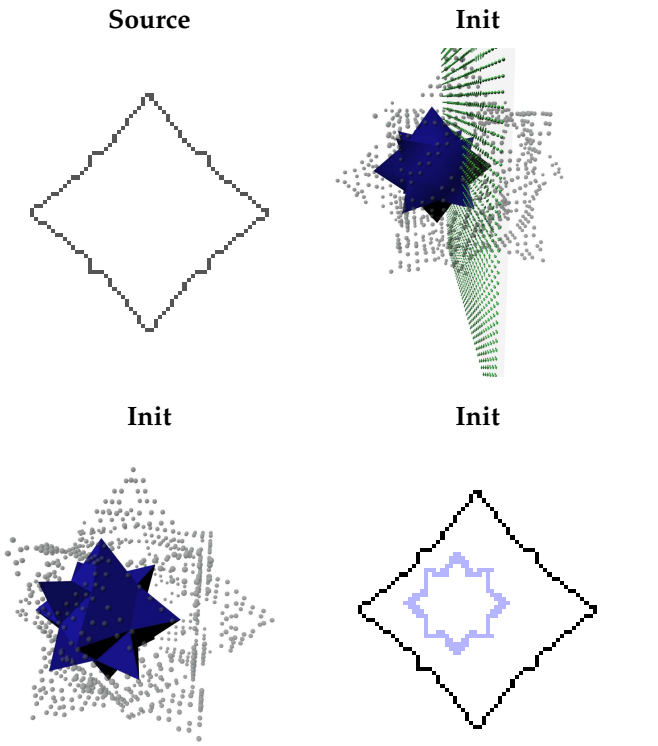


Fig. 13: Star initialization. (Top left) A middle slice from the source image, (Top right) The slice position in space, (Bottom left) Initialization for both snakes , (Bottom right) A slice from the previous image.

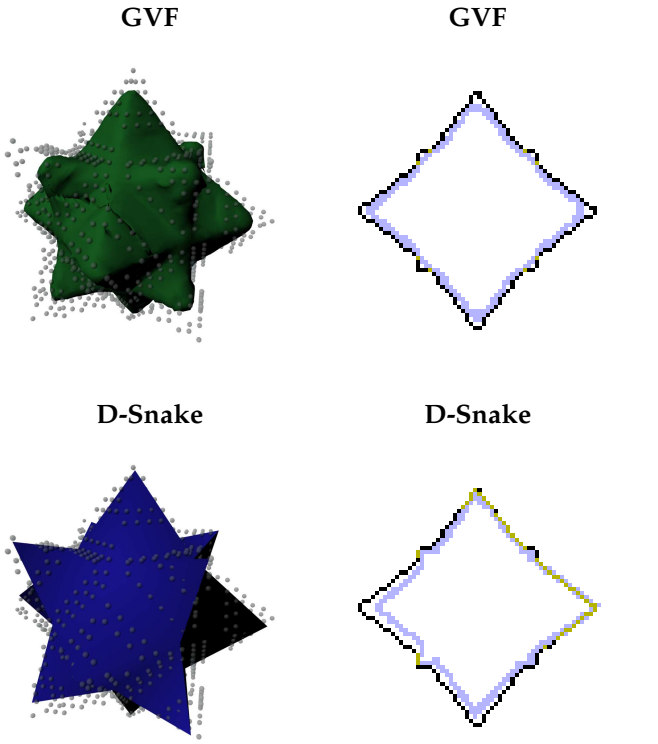


Fig. 14: A clean star. (Top) GVF result, (Bottom) D-Snake result.

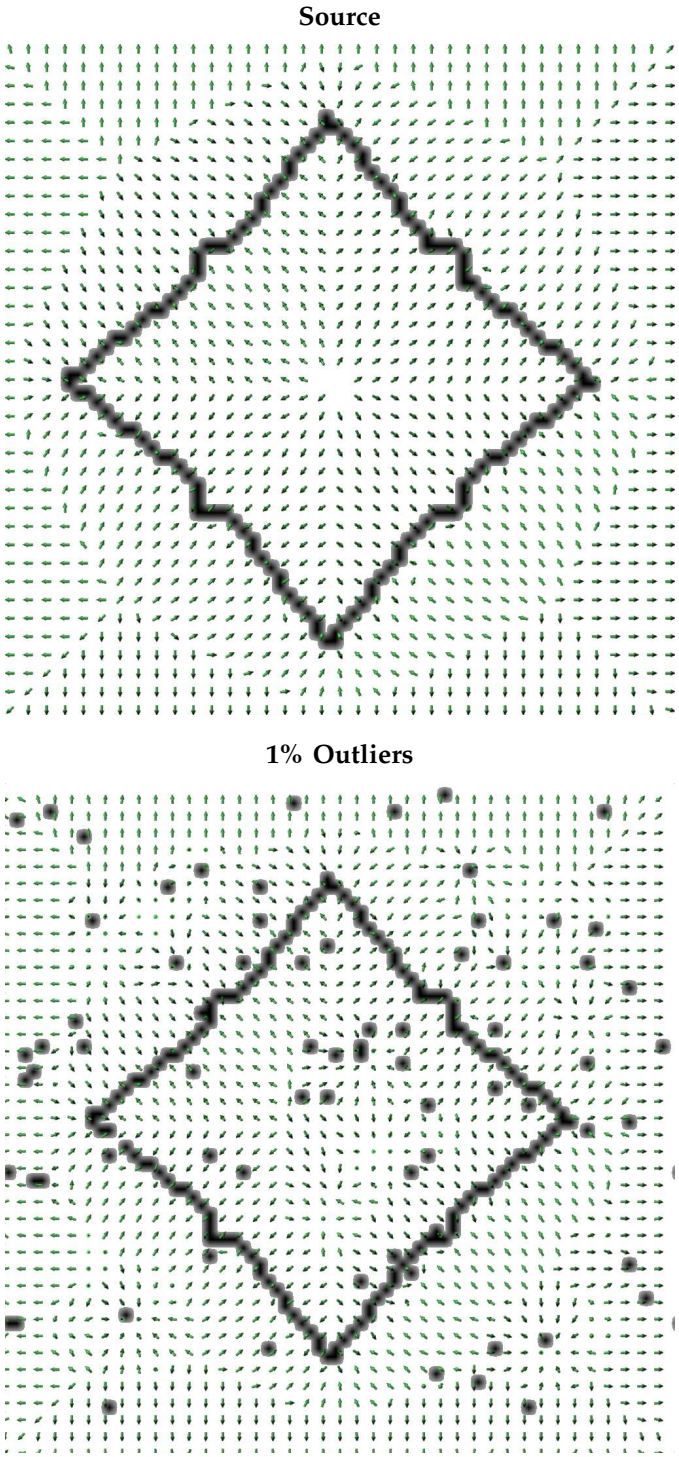
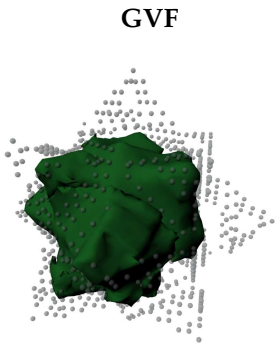
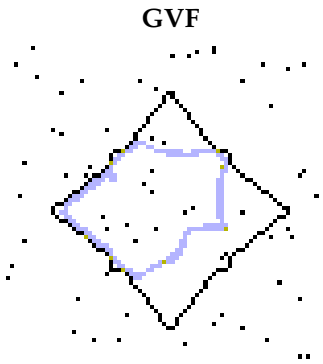
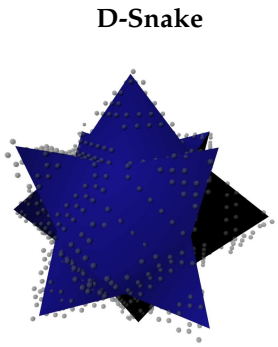
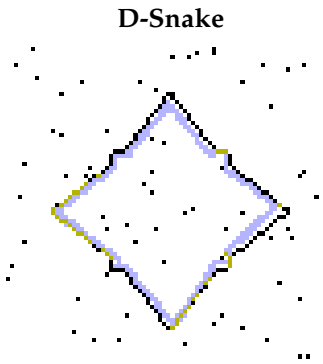


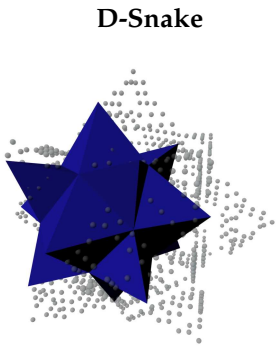
Fig. 15: Star GVF field. (Top) A slice from the GVF field of a clean image, (Bottom) A slice from the GVF field of an image with 1% outliers.











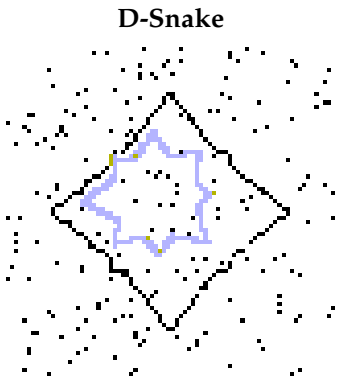
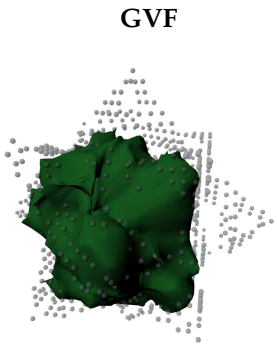
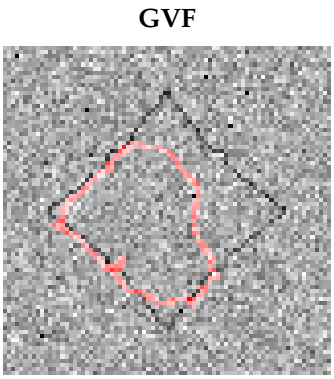
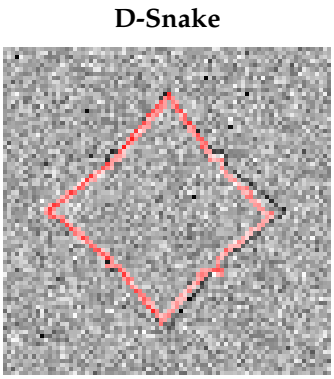


Fig. 16: Star with outliers. (Top) GVF snake result for 1% outliers, (Middle) D-Snake result for 1% outliers, (Bottom) D-Snake result for 3% outliers.







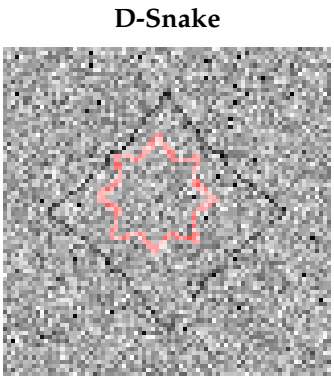


Fig. 17: Star with noise. (Top) GVF snake results for $\sigma = 30$, (Middle) D-Snake results for $\sigma = 30$, (Bottom) D-Snake results for $\sigma = 35$.

other forms of deformation energy. The main requirement from the energy is that it is based on positional data only. Moreover it should be possible to minimize the deformation energy point-wise, avoiding global optimization such as that used in Laplacian surface editing [26] or mean-value encoding [27] used by several sketch-base modeling techniques [28]–[30]. It should be noted that although [26] is fast, it uses a linearized version of the rotation operator, which can be sensitive to large rotations. In the method of Kraevoy et al. [27], vertex coordinates are encoded by decomposition into a tangential component computed in the projection plane and a normal component based on the vertex offset from the plane, which is rotation-invariant. This seems to be quite effective, although the resulting mean-value weighted function requires a non-linear optimization.

We mentioned in the related work the ASM, which builds a statistical model from a training set. If one has such a training set at his disposal, one can then build a shape space [31] based on the ASAP energy, which guides the snake deformation even better.

A few words about initialization. A snake in general is quite sensitive to the initial positioning of the template relative to the dataset. For simple examples such as Fig. 6, the extra deformation force of D-Snake alleviated this sensitivity somewhat, but for complex data sets such as the lung (Fig. 9), a good initialization is crucial. Currently the degrees of freedom for the initial position of the template are Euclidean transformations. However, our D-Snake permits the medical professional to specify

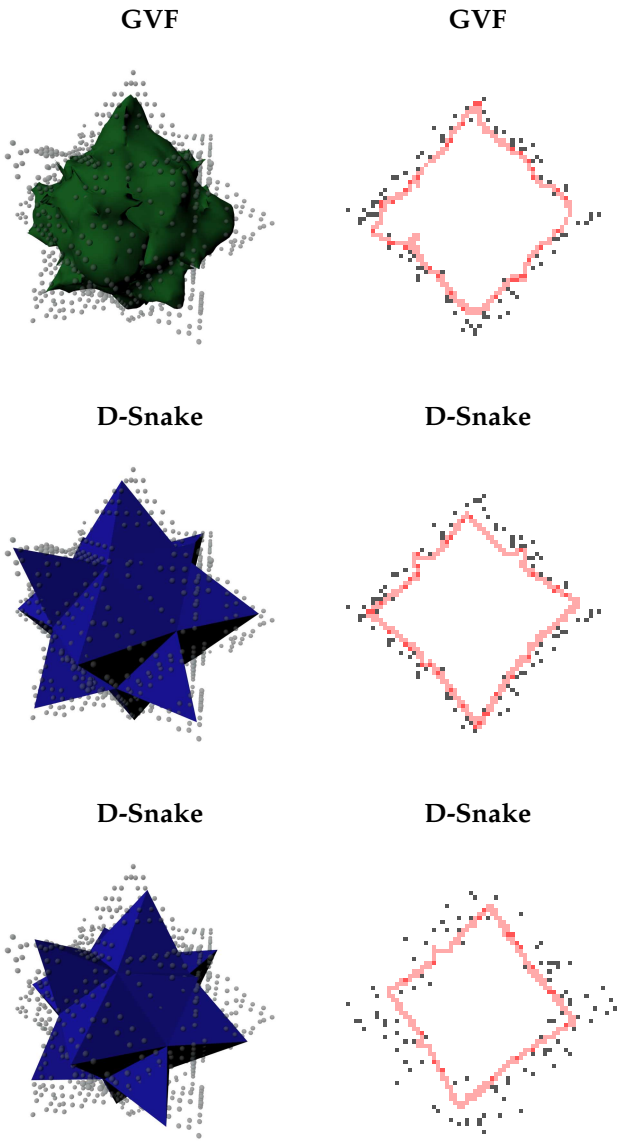


Fig. 18: Star with perturbations. (Top) GVF snake result on $r = 2$, (Middle) D-Snake result on $r = 2$, (Bottom) D-Snake result on $r = 4$.

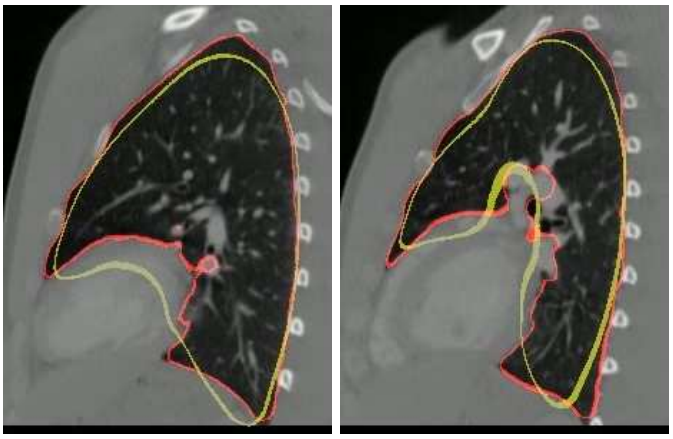


Fig. 19: Two slices of the lung image with the initial (template) snake in yellow, and the final D-Snake in red.

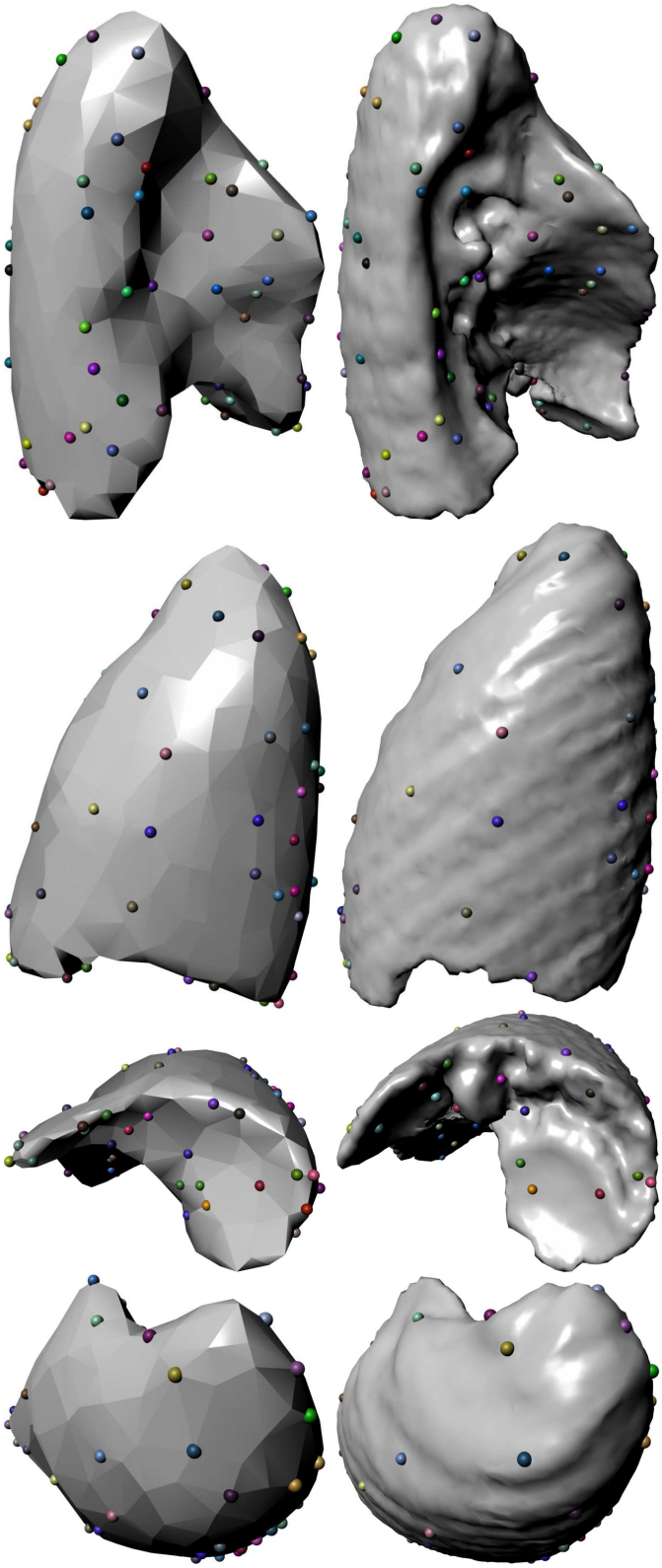


Fig. 20: Color-coded correspondence between (Left) template and (Right) final D-Snake.

better guidance for the snake by placing markers. Since the deformation vertices have semantics, markers can be placed inside the 3D image to guide important features to their exact location. We avoided use of markers in our experiments in order to keep the D-Snake on equal footing with the classical snake.

REFERENCES

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *J. of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[2] C. Xu, D. Pham, and J. Prince, "Image segmentation using deformable models," in *Handbook of Medical Imaging*, vol. 2, 2000, pp. 129–174.

[3] Q.-X. Huang, B. Adams, M. Wicke, and L. J. Guibas, "Non-rigid registration under isometric deformations." *Comput. Graph. Forum*, vol. 27, no. 5, pp. 1449–1457, 2008.

[4] H. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans." *Comput. Graph. Forum*, vol. 27, no. 5, pp. 1421–1430, 2008.

[5] L. Cohen, "On active contour models and balloons," *Computer Vision, Graphics, and Image Processing. Image Understanding*, vol. 53, no. 2, pp. 211–218, 1991.

[6] L. Cohen and I. Cohen, "Finite-element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 11, pp. 1131–1147, 1993.

[7] C. Xu and J. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Trans. on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.

[8] B. Li and S. Acton, "Active contour external force using vector field convolution for image segmentation," *IEEE Trans. on Image Processing*, vol. 16, no. 8, pp. 2096–2106, 2007.

[9] Y. Wang and E.-K. Teoh, "Object contour extraction using adaptive B-snake model," *J. Math. Imaging Vis.*, vol. 24, no. 3, pp. 295–306, 2006.

[10] T. Mcinerney and D. Terzopoulos, "T-snakes: Topology adaptive snakes," in *Medical Image Analysis*, 1999, pp. 840–845.

[11] S. Osher and J. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. of Comp. Physics*, vol. 79, no. 1, pp. 12–49, 1988.

[12] J.-Y. Park, T. McInerney, D. Terzopoulos, and M.-H. Kim, "A non-self-intersecting adaptive deformable surface for complex boundary extraction from volumetric images," *Computers & Graphics*, vol. 25, no. 3, pp. 421–440, 2001.

[13] J.-P. Thirion, "Fast non-rigid matching of 3D medical images," INRIA, Research Report RR-2547, 1995.

[14] D. Sýkora, J. Dingliana, and S. Collins, "As-rigid-as-possible image registration for hand-drawn cartoon animations," in *Proc. of Non-Photorealistic Animation and Rendering*, 2009, pp. 25–33.

[15] T. Heimann, *Statistical Shape Models for 3D Medical Image Segmentation*. VDM Verlag, 2009.

[16] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models their training and application," *Comput. Vis. Image Underst.*, vol. 61, no. 1, pp. 38–59, Jan. 1995.

[17] J. Cheng and S. Foo, "Dynamic directional gradient vector flow for snakes," *IEEE Trans. on Image Processing*, vol. 15, no. 6, pp. 1563–1571, 2006.

[18] S. Schaefer, T. McPhail, and J. Warren, "Image deformation using moving least squares," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 533–540, 2006.

[19] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. of the Optical Society of America*, vol. 4, no. 4, pp. 629–642, 1987.

[20] B. K. P. Horn, H. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. of the Optical Society of America*, vol. 5, no. 7, pp. 1127–1135, 1988.

[21] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas, "Registration of point cloud data from a geometric optimization perspective," in *Proc. Symposium on Geometry Processing*, 2004, pp. 23–31.

[22] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proc. Symp. on Geometry Processing*, 2007, pp. 109–116.

[23] M. Floater, "Mean value coordinates," *Comp. Aided Geometric Design*, vol. 20, pp. 19–27, 2003.

[24] M. Meyer, M. Desbrun, P. Schro"der, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and mathematics*. Springer-Verlag, 2002, pp. 35–57.

[25] "CGAL, Computational Geometry Algorithms Library," http://www.cgal.org.

[26] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proc. Symposium on Geometry Processing*, 2004, pp. 175–184.

[27] V. Kraevoy and A. Sheffer, "Mean-value geometry encoding," *Intl. J. Shape Modeling*, vol. 12, no. 1, pp. 29–46, 2006.

[28] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1142–1147, 2005.

[29] J. Zimmermann, A. Nealen, and M. Alexa, "Silsketch: automated sketch-based editing of surface meshes," in *Proc. Symp. on Sketch-based Interfaces and Modeling*, 2007, pp. 23–30.

[30] V. Kraevoy, A. Sheffer, and M. van de Panne, "Modeling from contour drawings," in *Proc. Symp. on Sketch-Based Interfaces and Modeling*, 2009, pp. 37–44.

[31] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović, "Mesh-based inverse kinematics," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 488–495, Jul. 2005.

Zohar Levi received his MSc in computer science from Tel-Aviv University, Israel, in 2007. He is currently pursuing the PhD degree at the Faculty of Computer Science at the Technion at Haifa, Israel. His research interests include computer graphics and computer vision.

Craig Gotsman received a PhD in computer science from the Hebrew University of Jerusalem. Since 1991, he has been on the Faculty of Computer Science at the Technion at Haifa, Israel, where he co-founded the Center for Graphics and Geometric Computing (CGGC). His main research interests are computer graphics, geometry processing and geometric modeling. He is currently the Founding Director of the Technion-Cornell Innovation Institute (TCII) at the Cornell-NYC Tech campus.

APPENDIX
THE 3D MLS SIMILARITY TRANSFORMATION

For a given source point cloud $\{p_i\}$, a target point cloud $\{q_i\}$, and a given point v we seek the similarity transformation T which minimizes

$$\sum_i w_i \|T(p_i) - q_i\|^2, \qquad w_i = \frac{1}{\|p_i - v\|^2}.$$

Let:

$$p^* = \sum_i w_{ij} p_i, \qquad q^* = \sum_i w_{ij} q_i,$$

$$\hat{p}_i = p_i - p^*, \qquad \hat{q}_i = q_i - q^*.$$

As described in [1], if (by performing SVD)

$$U \Lambda V^T = \sum_i w_i \hat{p}_i \hat{q}_i^T,$$

then

$$T(x) = M(x - p^*) + q^*, \qquad M = \mu V U^T$$

$$\mu = trace(\Lambda)/trace(P^T P), \qquad P = (\sqrt{w_1}\hat{p}_1 ... \sqrt{w_n}\hat{p}_n).$$

(P is a $3 \times n$ matrix). To avoid reflection solutions (having negative determinant), see the modification described in [2]. Alternative solutions can be found in [3]–[5].

REFERENCES

[1] Y. Zhu and S. Gortler, "3d deformation using moving least squares." Harvard Univ.: TR-10-07, Tech. Rep., 2007.

[2] O. Sorkine, "Least-squares rigid motion using svd," New-York University, Tech. Rep., 2009.

[3] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. of the Optical Society of America*, vol. 4, no. 4, pp. 629–642, 1987.

[4] B. K. P. Horn, H. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. of the Optical Society of America*, vol. 5, no. 7, pp. 1127–1135, 1988.

[5] A. Cuno, C. Esperanca, A. Oliveira, and C. P. Roma, "3D as-rigid-as-possible deformations using MLS," in *Proc. of Computer Graphics International Conference*, 2007.