GREEDY ALGORITHM AND PHYSICS-BASED METHOD FOR ACTIVE CONTOURS AND SURFACES: A COMPARATIVE STUDY

Julien Mille, Romuald Boné, Pascal Makris and Hubert Cardot

Université François-Rabelais de Tours, Laboratoire Informatique
64 avenue Jean Portalis 37200 Tours, France
{julien.mille, romuald.bone, pascal.makris, hubert.cardot}@univ-tours.fr

ABSTRACT

Deformable models, such as the discrete active contour and surface, imply the use of iterative evolution methods to perform 2D and 3D image segmentation. Among the several existing evolution methods, we focus on the greedy algorithm, which minimizes an energy functional, and the physics-based method, which applies forces in order to solve a dynamic differential equation. In this paper, we compare the greedy and physics-based approaches applied on 2D and 3D models, as regards overall speed and segmentation quality, quantified with an evaluating function mainly based on the mean distance between the model and the desired shape.

Index Terms— 3D segmentation, active surface, greedy algorithm, physics-based

1. INTRODUCTION

Image segmentation by means of deformable models implies the use of iterative evolution methods. In the two-dimensional case, the variational approach [1] and the greedy algorithm [2] are often used to make active contours (or snakes) evolve. In 3D, discrete active surfaces (deformable meshes) are usually driven with physics-based methods [3, 4]. We focus on the greedy algorithm and the physics-based method: the former computes energies in order to minimize a functional, whereas the latter applies forces to solve a dynamic differential equation expressing law of motion. Although these methods are well suited to both discrete contours and surfaces, there have been few implementations of the greedy algorithm in 3D. On the other hand, physics-based methods are not especially chosen for driving 2D contours. In this paper, we propose and compare a greedy and a physics-based approach, relying on a formulation common to the 2D contour and 3D surface. The comparative study uses an experimental protocol dealing both with convergence time and segmentation quality. To represent this last one, we use a Hausdorff-based distance function, considering all pixels (resp. voxels in 3D images) belonging to the contour (resp. surface) by scan-conversion, computing the mean distance between the model and the desired shape.

The paper is organized as follows. Section 2 presents the 2D active contour and 3D active surface models, with adaptive refinement. Section 3 describes the greedy algorithm and the energies. In section 4, we present the physics-based method and the forces. Finally, evaluation of the methods with description of the distance function and experimental results are shown in section 5.

2. THE ACTIVE MODELS

To describe active 2D contour and 3D surface models simultaneously, we introduce a general framework. The contour is a discrete closed curve, whereas the surface model, described in [5], is a triangular mesh built by subdividing an icosahedron [6]. The models have a constant global topology, their initial shape being circular and spherical, respectively. Both are made up of a set of n vertices, denoted $\mathbf{p}_i = (x_i, y_i)^T$ in 2D and $\mathbf{p}_i = (x_i, y_i, z_i)^T$ in 3D. Each vertex $\mathbf{p}_i$ has a set of neighboring vertices, denoted N_i. In the 2D contour, index i is the discrete equivalent of the curve parameter, hence $N_i = \{i-1, i+1\}$ (indices $i+1$ and $i-1$ should be understood modulo n). For each vertex, we also consider the unit vector $\vec{\mathbf{n}}_i$, normal to the contour or surface, oriented in order to point towards the interior of the shape. Since the evolution algorithms described below modify vertex coordinates, all normals should be updated after each iteration (when all vertices have been translated).

To maintain a stable vertex distribution along the contour (resp. surface), adaptive resampling (resp. remeshing) is performed [7, 8]. It insures that every couple of neighbors $(\mathbf{p}_i, \mathbf{p}_j)$ satisfies the constraint $d_{min} \leq \|\mathbf{p}_i - \mathbf{p}_j\| \leq d_{max}$, where d_{min} and d_{max} are two user-defined thresholds. 2D resampling consists in inserting a vertex between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ when their distance exceeds d_{max}, and deleting $\mathbf{p}_{i+1}$ when it gets below d_{min} ($\mathbf{p}_i$ is then connected to $\mathbf{p}_{i+2}$). While resampling the 2D contour is rather straightforward, surface remeshing should take topological constraints into account. One of these is that two neighbors $\mathbf{p}_i$ and $\mathbf{p}_j$ should share exactly two common neighbors: $N_i \cap N_j = \{a, b\}$. When $\|\mathbf{p}_i - \mathbf{p}_j\| > d_{max}$,

a new vertex is created at the middle of line segment $\mathbf{p}_i\mathbf{p}_j$ and connected to $\mathbf{p}_a$ and $\mathbf{p}_b$ (middle part of figure 1). When $\|\mathbf{p}_i - \mathbf{p}_j\| < d_{min}$, $\mathbf{p}_j$ is deleted and $\mathbf{p}_i$ is translated to the middle location (right part of figure 1). The distance constraint is checked for every couple of neighbors, after each iteration of the evolution method, in order to insure constant vertex distribution during deformation process. It also prevents neighbors from getting too close, which might result in vertex overlapping and intersections between edges/triangles.

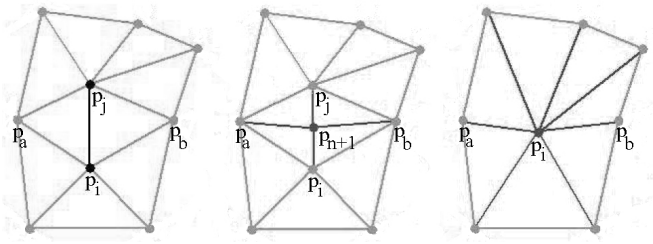

Fig. 1. Remeshing operations in the active surface: vertex inserting and deleting

3. GREEDY ALGORITHM AND ENERGIES

To the previously defined models, we associate a discrete energy functional, being the sum of the energies of each vertex. The greedy approach is an energy-minimizing algorithm introduced for 2D contours in [2] (a 3D extension applied on a triangular mesh was first proposed in [9]). Global minimization is done by means of successive local optimization. Given a square (resp. cubic for the 3D model) window of width w around each vertex, the energy is computed at each pixel (resp. voxel) belonging to the window and the vertex is moved to the location leading to the lowest energy. In contrast with $\mathbf{p}_i$ (the initial location), we denote $\mathbf{p}'_i$ the search location in the window, whose energy is a weighted sum of various internal and external energies, normalized on the whole window.

$$\begin{aligned} E(\mathbf{p}'_i) &= \alpha E_{cont}(\mathbf{p}'_i) + \beta E_{curv}(\mathbf{p}'_i) \\ &+ \gamma E_{grad}(\mathbf{p}'_i) + \delta E_{bal}(\mathbf{p}'_i) \end{aligned} \quad (1)$$

The continuity E_{cont} and curvature E_{curv} are internal energies maintaining the geometrical regularity of the contour. Their expressions involve euclidean distance and neighboring vertices. In following equations, the squared distance is computed rather than the distance, in order to save computational time. The continuity energy is computed as follows:

$$\begin{aligned} E_{cont}(\mathbf{p}'_i) &= \sum_{j \in N_i} \left| \overline{d^2} - \|\mathbf{p}'_i - \mathbf{p}_j\|^2 \right| \\ \overline{d^2} &= \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|N_i|} \sum_{j \in N_i} \|\mathbf{p}_i - \mathbf{p}_j\|^2 \end{aligned} \quad (2)$$

Minimizing it causes the vertices to remain evenly spaced. $\bar{d^2}$ is the global mean squared distance. For 2D snakes, curvature is equivalent to the squared distance between the vertex and the middle of its two neighbors. Extending this principle for the mesh, the curvature of the tested point $\mathbf{p}'_i$ is the squared distance between $\mathbf{p}'_i$ and the centroid of the neighbors of $\mathbf{p}_i$.

$$E_{curv}(\mathbf{p}'_i) = \left\| \mathbf{p}'_i - \frac{1}{|N_i|} \sum_{j \in N_i} \mathbf{p}_j \right\|^2 \quad (3)$$

Assuming that the neighbors are located regularly around the vertex, curvature minimization results in local smoothing. The gradient energy E_{grad} is an external term relating the model to the input image, attracting vertices toward object boundaries. We use the gradient magnitude of image I.

$$E_{grad}(\mathbf{p}'_i) = -\|\nabla I(\mathbf{p}'_i)\| \quad (4)$$

To increase the capture range, the model is allowed to inflate or deflate. Our balloon energy E_{bal} is derived from the inflation force proposed in [10].

$$E_{bal}(\mathbf{p}'_i) = \|\mathbf{p}'_i - (\mathbf{p}_i + k\vec{\mathbf{n}}_i)\|^2 \quad (5)$$

k is a real number, chosen to insure that the resulting vector points outside the window. The sign of weight δ (appearing in equation 1) controls the orientation of the balloon motion and should be set according to the initial location of the model, whether it is inside or outside the object.

Using the greedy algorithm, two neighbors $\mathbf{p}_i$ and $\mathbf{p}_j$ might overlap at iteration t only if their respective windows overlapped at iteration $t-1$. However, two square windows of width w do not overlap if their centers are w pixels away in at least one dimension. Using the infinite norm $\|\mathbf{p}_i - \mathbf{p}_j\|_\infty = \max(|x_i - x_j|, |y_i - y_j|, ...)$, we redefine the remeshing criterion:

$$w < \|\mathbf{p}_i - \mathbf{p}_j\|_\infty < 2w \quad (6)$$

4. PHYSICS-BASED METHOD AND FORCES

This approach is the most used one for deformable mesh [3, 4]. In this approach, segmentation is a force equilibrium reaching problem, according to a law of motion. Based on mechanics, the method is also referred to as dynamic approach, because time appears in the evolution equation. First and second-order inertial terms are introduced, making each vertex evolve independently according to a Newtonian law of motion:

$$m\frac{d^2\mathbf{p}_i}{dt^2} + \mu\frac{d\mathbf{p}_i}{dt} = \vec{\mathbf{F}}(\mathbf{p}_i) \quad (7)$$

where m and μ are the mass and viscosity, respectively. Mass is set to zero, thus removing acceleration (equation is then known as Lagrangian dynamics). An explicit evolution scheme is obtained using Euler time-integration, with time-step Δt.

$$\begin{aligned} \frac{\mu}{\Delta t}(\mathbf{p}_i^{(t+1)} - \mathbf{p}_i^{(t)}) &= \vec{\mathbf{F}}(\mathbf{p}_i^{(t)}) \\ \mathbf{p}_i^{(t+1)} &= \mathbf{p}_i^{(t)} + \frac{\mu}{\Delta t}\vec{\mathbf{F}}(\mathbf{p}_i^{(t)}) \end{aligned} \quad (8)$$

At each iteration, $\mathbf{p}_i$ is translated by global force $\vec{\mathbf{F}}$, which is a weighted sum of internal and external forces. Since $\vec{\mathbf{F}}$ is already a weighted sum, equation 8 is simplified. Time-step and viscosity notions are included in the weights.

$$\begin{aligned}\vec{\mathbf{F}}(\mathbf{p}_i) &= \alpha\vec{\mathbf{F}}_{cont}(\mathbf{p}_i) + \beta\vec{\mathbf{F}}_{curv}(\mathbf{p}_i) \\ &+ \gamma\vec{\mathbf{F}}_{grad}(\mathbf{p}_i) + \delta\vec{\mathbf{F}}_{bal}(\mathbf{p}_i) \end{aligned} \quad (9)$$

These forces are vector equivalences of the energies presented in the previous section. Indeed, force and energy are closely related since force is the negative spatial derivative of energy, apart from a constant. Given a point-derivative formulation, $\partial f/\partial \mathbf{p} = (\partial f/\partial p_x, \partial f/\partial p_y, \partial f/\partial p_z)^T$, differentiating a squared distance (in our case, an energy) gives:

$$\frac{\partial \left\|\mathbf{p}-\mathbf{q}\right\|^2}{\partial \mathbf{p}} = 2(\mathbf{p}-\mathbf{q}) \quad (10)$$

Following this principle, finding $\mathbf{p}$ minimizing $\left\|\mathbf{p}-\mathbf{q}\right\|^2$ is done by translating the current $\mathbf{p}$ with force $\mathbf{q}-\mathbf{p}$. As regards internal forces, the continuity force is similar to the one used in [7]. It involves the same mean squared distance $\overline{d^2}$ defined in the previous section. The curvature force pushes a vertex towards the centroid of its neighbors.

$$\vec{\mathbf{F}}_{cont}(\mathbf{p}_i) = \sum_{j\in N_i} (\overline{d^2} - \left\|\mathbf{p}_i-\mathbf{p}_j\right\|^2) \frac{\mathbf{p}_i-\mathbf{p}_j}{\left\|\mathbf{p}_i-\mathbf{p}_j\right\|} \quad (11)$$

$$\vec{\mathbf{F}}_{curv}(\mathbf{p}_i) = (\frac{1}{|N_i|}\sum_{j\in N_i}\mathbf{p}_j) - \mathbf{p}_i \quad (12)$$

As regards external forces, the gradient force $\vec{\mathbf{F}}_{grad}$ is related to the spatial derivative of the gradient energy. To express the balloon force, translation along the normal vector may be directly applied.

$$\vec{\mathbf{F}}_{grad}(\mathbf{p}_i) = -\nabla \left\|\nabla I(\mathbf{p}'_i)\right\| \quad (13)$$

$$\vec{\mathbf{F}}_{bal}(\mathbf{p}_i) = \vec{\mathbf{n}}_i \quad (14)$$

Here, a preliminary comparison is done between the two evolution methods. Expressing algorithmic complexity with respect to vertex quantity, physics-based approach gives $O(n)$, whereas the greedy algorithm gives $O(nw)$. As regards energy weighting, in the greedy algorithm, weight parameters only have a relative meaning (one can be fixed and the other tuned), since they intervene in an expression to be minimized. In the other approach, weights include a time-step notion, and have an absolute meaning (if they are too high, the resulting force might cause the vertices to skip the real object boundaries). In addition, the greedy algorithm is based on step-wise motions, since the vertices are moved in a window, given a spatial step equivalent to the image sampling (to some extent, vertex coordinates could be stored as integer values). Conversely, there is no such constraint on physics-based motions.

5. EVALUATION

The comparative study between the evolution methods is done according to three criteria: the number of iterations needed to converge to the boundaries, the overall computational time, and the quality of the obtained shape. To evaluate this last one, a function quantifying how well the model fits to the real object boundaries should be used. We use a function taking into account the overall distance between the desired shape and the model, which we believe to be representative of the quality of segmentation. Let $\mathcal{S}$ be the set of points belonging to the contour/surface, and $\mathcal{B}$ the set of points belonging to the object boundaries. For each contour point, we consider the distance to the nearest boundary point, leading to the oriented Hausdorff distance, also used in [11].

$$\mathcal{D} = \frac{1}{|\mathcal{S}|}\sum_{\mathbf{s}\in\mathcal{S}} \min_{\mathbf{b}\in\mathcal{B}} \left\|\mathbf{s}-\mathbf{b}\right\| \quad (15)$$

Since the evaluation is performed on pixels/voxels (and not only on vertices), connections between vertices need to be discretized. For the 2D contour, edges are scanned using Bresenham's line-drawing algorithm. For the mesh, triangles are voxelized using a 3D scan-conversion algorithm [12] (see figure 2).

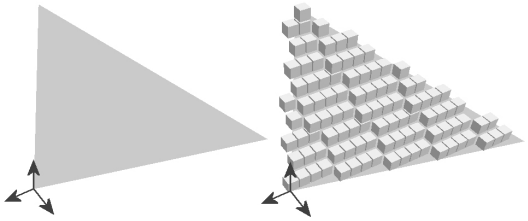

Fig. 2. Triangle voxelization

Segmentation tests were performed on 2D images: one with 2 crossing ellipses (a) and one with a symbol with strong concavities (b), and 3D images: one with 3 crossing ellipsoids (c) and one with a helicoidal spiral shape (d). All of them were 200 pixels/voxels wide, and highly corrupted with gaussian noise (we applied smoothing before computing gradient). The gradient weight λ was set to 1 for all experiments (higher values caused the boundaries to be skipped by physics-based motion). To achieve equivalent compromises between speed and quality, differents sets of weights were used for the two methods.

For images (a) and (c), models were initialized outside the object and remeshing was disabled (contour and surface had respectively 100 and 2562 vertices). Average weight values were ($\alpha = 0.5$, $\beta = 0.3$, $\delta = 0.8$) for the greedy approach and ($\alpha = 0.1$, $\beta = 0.1$, $\delta = 0.15$) for the physics-based one. For images (b) and (d), remeshing was enabled. For the symbol, initial contour was around the object and had 100 vertices. For both approaches, two values of w were used, leading to different contour finenesses (we used the same remeshing criterion (6) for the two approaches, which explains why w appears for the physics-based method). Coefficients α and β were set to 0 to allow the vertices to move deep into the concavities. We used $\delta = 0.3$ and 0.8 for the greedy and physics-based approach, respectively. For the spiral, the mesh was initialized as a small sphere holding 12 vertices, inside

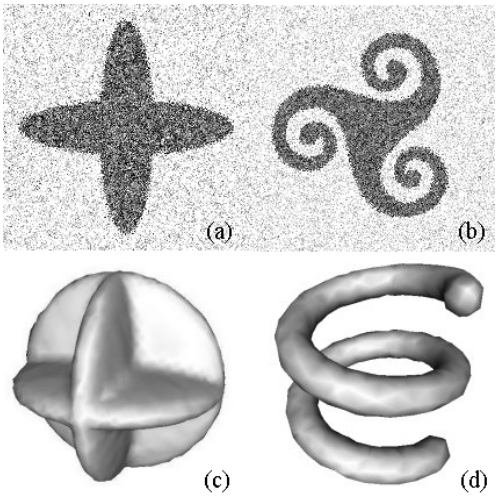

Fig. 3. Up: 2D test images. Down: 3D test images (surfaces)

the shape. As previously, two different mesh finenesses were tested. Weights were ($\alpha = 0$, $\beta = 0.25$, $\delta = -0.5$) and ($\alpha = 0$, $\beta = 0.1$, $\delta = -0.95$), respectively. Results are shown in the following table (time is expressed in ms).

			Nb. iter.	Time	Distance
2ellipses	Greedy	w=3	49	7.6	0.434
		w=5	25	8.8	0.452
	Physics		250	11	0.526
Symbol	Greedy	w=3	260	160	0.002
		w=5	150	105	0.008
	Physics	w=3	370	76	0.185
		w=5	440	52	0.197
3ellipsoids	Greedy	w=3	61	730	0.530
		w=5	39	1000	0.529
	Physics		250	1710	0.658
Spiral	Greedy	w=3	350	1910	0.497
		w=5	192	730	0.723
	Physics	w=3	440	3469	0.931
		w=5	448	1150	1.161

The comparison is done on well-segmented shapes, i.e. the distance values correspond to satisfactory results. The greedy approach needs significantly fewer iterations to converge. This yields to lower completion time, except for the symbol image (strongly concave objects tend to reduce the efficiency of greedy algorithm). It also gives better boundary fitting. Explanation is that the physics-based method causes vertices to oscillate around the boundaries, enabling them to have non-integer coordinates, which does not happen when using the greedy algorithm, since vertices evolve according to an integer step-wise motion (hence, the greedy algorithm is inherently more stable). This is particularly visible on the symbol, which has a high perimeter.

6. CONCLUSION

In this paper, we presented two related greedy and physics-based approaches, relying on a common formulation, to deform active contours and surfaces used to segment 2D and 3D images. A comparison was made, dealing with computational time and segmentation quality. The comparison turns out to be in favor of the greedy algorithm, which yields globally better computation cost and segmentation accurateness. Future work may include the development of an hybridation between energy-based and force-based evolution method, in order to take advantage of both principles.

7. REFERENCES

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[2] D.J. Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.

[3] J.O. Lachaud and A. Montanvert, "Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction," *Medical Image Analysis*, vol. 3, no. 2, pp. 187–207, 1999.

[4] J. Montagnat and H. Delingette, "4D deformable models with temporal constraints: application to 4D cardiac image segmentation," *Medical Image Analysis*, vol. 9, no. 1, pp. 87–100, 2005.

[5] J. Mille, R. Boné, P. Makris, and H. Cardot, "3D segmentation using active surface: a survey and a new model," in 5^{th} *IASTED Int. Conf. on Visualization, Imaging & Image Processing (VIIP)*, Benidorm, Spain, 2005, pp. 610–615.

[6] T. McInerney and D. Terzopoulos, "A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis," *Computerized Medical Imaging and Graphics*, vol. 19, no. 1, pp. 69–83, 1995.

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

[8] G. Slabaugh and G. Unal, "Active polyhedron: surface evolution theory applied to deformable meshes," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, San Diego, USA, 2005, vol. 2, pp. 84–91.

[9] A.J. Bulpitt and N.D. Efford, "An efficient 3D deformable model with a self-optimising mesh," *Image and Vision Computing*, vol. 14, no. 8, pp. 573–580, 1996.

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

[11] M-P. Dubuisson and A.K. Jain, "A modified Hausdorff distance for object matching," in *Proceedings of* 12^{th} *International Conference on Pattern Recognition (ICPR)*, Jerusalem, Israel, 1994, pp. 566–568.

[12] A. Kaufman, "Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes," *Computer Graphics*, vol. 21, no. 3, pp. 171–179, 1987.