Random Walks for Interactive Separation of Segmented Bone Fragments

Johan Nysjö
Centre for Image Analysis, Uppsala University, Sweden
Email: johan.nysjo@cb.uu.se

***Abstract*—Identifying and segmenting individual bone fragments in CT images is an important task in surgery planning. Although it is often straightforward to separate the bone tissue from the rest of the image, extracting individual bone fragments is typically a more difficult segmentation problem. In many cases, the bone fragment of interest is connected to one or several other bone fragments, making it difficult for an automatic segmentation method to identify the object boundary. The aim of this project is to investigate whether a random walks-based mesh segmentation method can be used to extract individual bone fragments from surface meshes that have been obtained with the marching cubes algorithm.**

I. Introduction

A number of mesh segmentation methods have been developed for the purpose of decomposing 3D models into components. In contrast to conventional image segmentation methods, these methods perform the segmentation on explicit surface representations of objects, rather than on voxel grids. In this project, we will investigate a recently developed mesh segmentation method [1] that is based on the random walks paradigm. The objective is to see whether this method, which was originally developed for segmenting CAD models or graphical 3D models such as the one shown in Figure 1, can be used to segment and separate individual bone fragments in CT images—a task that is of great importance in surgery planning. The method will be tested on surface meshes that have been extracted from real CT images.

II. Interactive Mesh Segmentation

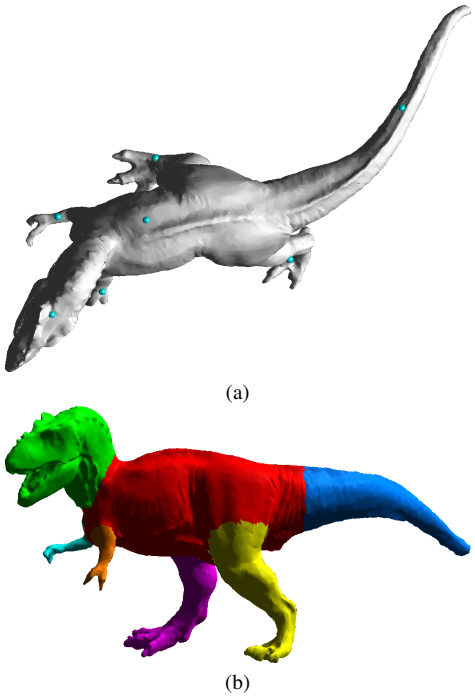
This section describes the different steps in the interactive mesh segmentation method that we have adapted for bone segmentation.

A. Surface Extraction

Given a CT image, we use the marching cubes algorithm [2] to extract an explicit surface representation of the contained bone fragments. This surface is represented as a triangle mesh $T = (V, F)$, where V is the vertex set and F is the face set. The isovalue we use for the surface extraction, $t_{iso} = 100$, is based on the Hounsfield value for bone tissue.

B. Mesh Simplification

A triangle mesh extracted from a CT image may contain millions of faces, making the mesh segmentation potentially time-consuming to compute. To speed up the segmentation process, we follow the advice in [1] and use MATLAB's mesh-simplification function `reducepatch` to reduce the number of faces in the mesh.



(a) (b)

Fig. 1. Random walks-based mesh cutting segmentation of a 3D model. (a) Interactive seeding. (b) Segmentation result.

C. Interactive Seeding

The user initializes the segmentation by placing n seeds on the regions of interest, using a standard 2D mouse and a keyboard as input devices. Each seed corresponds to a single region of interest and will snap to the closest face in the triangle mesh. We denote the resulting seed faces as $s_1, \ldots, s_n$.

D. Random Walks-Based Mesh Segmentation

To extract the selected bone fragments from the surface mesh, we use the random walks-based mesh segmentation method presented by Lai et al [1]. This method, which is based on the same idea as the random walks algorithm for image segmentation [3], allows the user to perform interactive mesh cutting.

In brief, the method works as follow. Let $f_1, \ldots, f_m$ denote the faces in the surface mesh. The probability that a random walk starting from face f_k will reach a seed s_l

before it reaches any of the other seeds is given by the equation

$$P^l(f_k) = \sum_{i=1}^{\kappa} p_{k,i} P^l(f_{k,i}), \tag{1}$$

where $f_{k,i}$ denotes a neighbor face sharing an edge $e_{k,i}$ with f_k, and κ is the number of such neighbor faces (normally, $\kappa = 3$). The probability that a random walker at face f_k will move across an edge $e_{k,i}$ to a face neighbor $f_{k,i}$ is denoted as $p_{k,i}$, and is computed as

$$p_{k,i} = |e_{k,i}| \exp\left\{-\frac{d_1(f_k, f_{k,i})}{\sigma}\right\}, \tag{2}$$

where

$$d_1(f_k, f_{k,i}) = \frac{\eta}{2}\|N_k - N_{k,i}\|^2 \tag{3}$$

is a difference function measuring the dihedral angle between f_i and $f_{k,i}$, and σ is a weight used to control the influence of d_1. N_k and $N_{k,i}$ denote the face normals, and η is a weighting term. To prioritize concave edges, we set η to $\eta = 1.0$ for concave edges and $\eta = 0.2$ for convex edges. The control parameter σ is set to $\sigma = 1.0$ in all experiments. See [1] for more details. The computed probability values $p_{k,i}$ should be scaled so that they satisfy the equation

$$\sum_{i=1}^{\kappa} p_{k,i} = 1. \tag{4}$$

For each seed $s_1, \ldots, s_n$, Equation 1 defines a sparse linear system $A_{m\times m}P^l = B^l$, where P^l and B^l are column vectors of length m. These systems can be written on the composed form

$$AP = B, \tag{5}$$

where $P = (P_1, \ldots, P_n)$ and $B = (B_1, \ldots, B_n)$. A will be a sparse matrix with at most κ non-zero elements per row. For B^l, it holds that $B^l(f_k) = p_{k,i}$ if one of the neighbors $f_{k,i}$ to f_k happens to be the l^{th} seed; otherwise, $B^l(f_k) = 0$ [1].

Surface meshes extracted with the marching cubes algorithm can often have disconnected components. Experimentally, we have found that linear systems constructed from such meshes are solvable, but it could be of interest to verify that these systems actually have unique solutions.

After solving the sparse linear system in Equation 5, we assign, for $l = 1, \ldots, n$, the label of seed s_l to the mesh faces satisfying the equation

$$P^l(f_k) = \max_{t=1,\ldots,n} P^t(f_k). \tag{6}$$

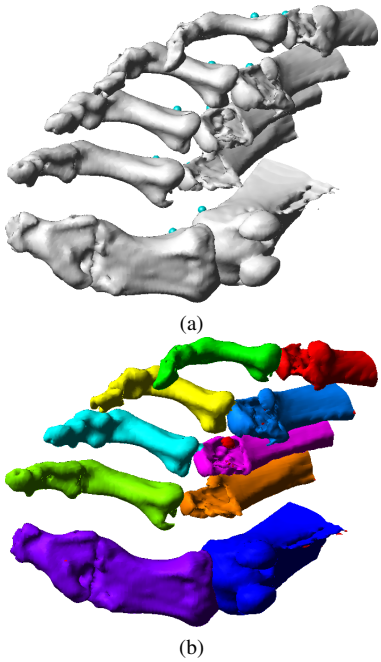
Mesh faces that are not connected to any of the seeds will (in our implementation) be assigned a default label.

III. Implementation

We have implemented the interactive mesh segmentation method described in Section II in MATLAB, using existing functions from a graph theory toolbox[1] to efficiently determine the face neighbors of each face in the triangulation. The matrix A is represented as a sparse matrix, and the corresponding sparse linear system $AP = B$ is solved using MATLAB's direct solver.

It is possible (and desirable) to implement the seeding so that the user can define multiple seeds per label. In our initial implementation, however, the seeding is restricted to one seed per label.

[1]URL: http://www.mathworks.com/matlabcentral/fileexchange/5355-toolbox-graph



(a) (b)

Fig. 2. Interactive mesh cutting applied on a triangle mesh extracted from a CT image of a human foot. (a) Interactive seeding. (b) Segmentation result.

IV. Experiments and Results

We tested the mesh segmentation method on two surface meshes that had been extracted from CT images of a human foot and the hand of a mouse. In addition, we also tested the method on the 3D model shown in Figure 1, to verify that our implementation provides similar results as the implementation used in [1].

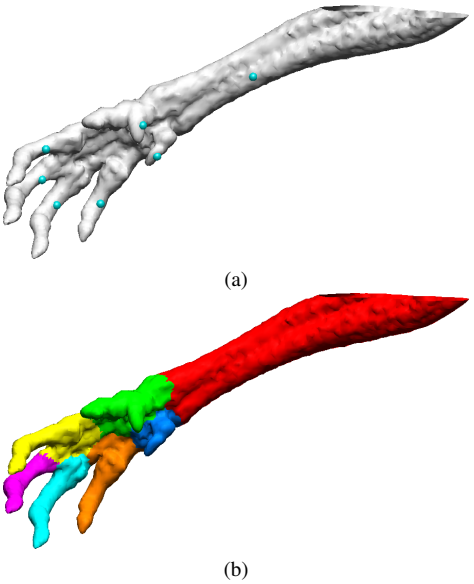
Figures 2 and 3 show the mesh segmentation result obtained for the two CT images. Given a few seeds, the segmentation method is able to identify the object boundaries, although additional seeds would be required to obtain an accurate segmentation of the mouse hand.

The segmentation result shown in Figure 1b illustrates that the method is able to decompose simple graphical models.

Although not presented in this report, the computation times required to solve the sparse linear systems seems to be consistent with the timing results reported in [1]. However, for large surface meshes (i.e., meshes with more than 100000 faces), the computation time is dominated by the task of creating the graph and setting up the sparse linear system, indicating that we need to optimize some parts of the implementation.

V. Conclusions

Overall, the interactive mesh segmentation method we have investigated here seems to perform well on the



(a) (b)

Fig. 3. Interactive mesh cutting applied on a triangle mesh extracted from a CT image of a mouse hand. (a) Interactive seeding. (b) Segmentation result.

task of segmenting individual bones in surface meshes extracted from CT images. However, the method tend to produce suboptimal cutting contours when the seeding is limited to one seed per label, indicating that we need to extend the implementation to handle multiple seeds per label. Placing more seeds around the desired cutting contour may improve the segmentation accuracy, but will also make the segmentation more tedious. The constrained random walks-based mesh cutting method by Zhang et al. [4] overcomes this problem by allowing the user to specify additional seeds on the cutting contours. These seeds are used as hard or soft boundary constraints to improve the accuracy of the mesh cutting.

To evaluate the accuracy of the obtained mesh segmentation results, we could use the framework described in [5], which provides a benchmark for evaluation of 3D mesh segmentation algorithms.

Finally, it would be interesting to investigate whether the mesh segmentation results can be propagated to the underlying voxel data.

ACKNOWLEDGMENTS

The 3D model used in this report has been obtained from the AIM@SHAPE Shape Repository[2].

REFERENCES

[1] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, "Rapid and effective segmentation of 3d models using random walks," *Computer Aided Geometric Design*, vol. 26, no. 6, pp. 665–679, 2009.

[2] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[3] L. Grady, "Random walks for image segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[4] J. Zhang, J. Zheng, and J. Cai, "Interactive mesh cutting using constrained random walks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 3, pp. 357–367, 2011.

[5] X. Chen, A. Golovinskiy, and T. Funkhouser, "A Benchmark for 3D Mesh Segmentation," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, pp. 1–12, 2009.

[2]URL: http://shapes.aim-at-shape.net/viewmodels.php