Computerized Medical Imaging and Graphics 40 (2015) 120–127

Contents lists available at ScienceDirect

Computerized Medical Imaging and Graphics

journal homepage: www.elsevier.com/locate/compmedimag

Interactive 3D medical data cutting using closed curve with arbitrary shape

Hai Ning, Rongqian Yang*, Amin Ma, Xiaoming Wu

Department of Biomedical Engineering, South China University of Technology, 510006 Guangzhou, China

ARTICLE INFO

Article history:
Received 29 July 2013
Received in revised form 31 August 2014
Accepted 6 October 2014

Keywords:
Interactive 3D cutting
Binary mask image
Coordinate transformation
Octree decomposition
Image segmentation

ABSTRACT

Interactive 3D cutting is widely used as a flexible manual segmentation tool to extract medical data on regions of interest. A novel method for clipping 3D medical data is proposed to reveal the interior of volumetric data. The 3D cutting method retains or clips away selected voxels projected inside an arbitrary-shaped closed curve which is clipping geometry constructed by interactive tool to make cutting operation more flexible. Transformation between the world and screen coordinate frames is studied to project voxels of medical data onto the screen frame and avoid computing intersection of clipping geometry and volumetric data in 3D space. For facilitating the decision on whether the voxels should be retained, voxels through coordinate transformation are all projected onto a binary mask image on screen frame which the closed curve is also projected onto to conveniently obtain the voxels of intersection. The paper pays special attention to optimization algorithm of cutting process. The optimization algorithm that mixes octree with quad-tree decomposition is introduced to reduce computation complexity, save computation time, and match real time. The paper presents results obtained from raw and segmented medical volume datasets and the process time of cutting operation.

1. Introduction

Volume clipping is typically used in visualizing medical data, preparing surgical plans and processing images, because it conveniently clips away selected parts of volume data. Volume clipping is also a useful approach in exploring the interior details of volume data [1]. Accurate volume data clipping can meet users' requirements to design more complicated volume model by cutting and pasting existing volume data [2]. Combined with image segmentation algorithms, clipping operation can be used as a pre-processing step to extract region of interest (ROI) in volume images, such as tumor organizations or important brain functional areas. Just because volume cutting plays an so important role to remove and carve the occluding materials and extract different regions of interest [4,9], we explore a 3D cutting tool in depth which can arbitrarily select volume data and cut them away in real-time. Moreover, the cutting algorithm introduced in the paper has been experimented to reveal hidden parts of brain volume data for surgical planning and extract cardiac chambers from heart volume images effectively.

Most volume clipping methods use common clipping geometry that can be structured by implicit functions to clip medical images. Many researchers had used a set of volumetric data clipping methods, including plane, bounding box, ball, and contour surface cutting [3]. Several methods based on volumetric textures and depth fields constructed various geometric primitives to cut unwanted volumetric data away [4]. However, depth-based clipping methods hardly handled concave geometry clipping. To solve the problem, Xie et al. [5] proposed a clipping method that uses the clipping distance field to represent clipping geometry and implement a clipping algorithm on the fragment shader. McGuffin et al. [6] presented a method for browsing the interior of volumetric data with interactive manipulation widgets. Some researchers also presented multifunctional tools to segment volume data flexibly. Huff et al. [7] exploited programmable hardware and proposed three tools (eraser, digger, and clipper) to uncover hidden structures in volume images. Correa et al. [8] proposed a collection of operators (peeler, retractors, pliers, and dilators) that can be placed anywhere on or within volume data. Chen et al. [9] utilized point radiation techniques and interactive parallel region growing algorithm to extract different regions of interest. There were also some researchers combine sketch-lines with subdivision surface to create novel envelops to cut unwanted region away [10]. Users needed to define the depth of penetration into the volume image before the cutting technique was implemented. The painting metaphor

* Corresponding author. Tel.: +86 15002020271.
E-mail addresses: ningtian2009@126.com (H. Ning), bmeyrq@gmail.com (R. Yang), aminma419@163.com (A. Ma), bmxmwus@scut.edu.cn (X. Wu).

http://dx.doi.org/10.1016/j.compmedimag.2014.10.004
0895-6111/

defined a selection tool that is similar to "brush strokes" to select voxels [11]. In some of the commercial medical image processing software, for example, zioTerm, similar cutting approaches based on closed curve were adopted to clip away or retain selected data. Industrial design software, such as AutoCAD, used mesh, curved surface or other 3D model to cut objects through intersecting them. Many approaches had been proposed to simulate the cutting process of deformable objects and mesh data. Zhang et al. [12] proposed a hybrid cutting method combining non-progressive cutting with progressive cutting for the real-time simulation of soft-tissue cutting. The method that dynamically modifying the mesh topology of deformable objects achieved better simulation effects [13]. Lenka et al. [14] used two synchronized geometrical models at different resolutions to simulate the interactive cutting of viscoelastic objects.

In this paper, a flexible and timesaving method adopting free closed curve is proposed to address two main issues in clipping 3D medical images, namely, (1) finding the correspondence between clipping geometries and volume data in 2D space, and (2) reducing computation complexity and saving computation time.

The proposed method differs from those using common clipping geometries, which mainly uses an interactive tool to draw an arbitrary closed curve to retain or clip away volume data. Thus, the clipping geometry can be convex or concave. A binary mask image with same size as the rendering window is built according to the closed curve to conveniently compute the coordinate relationship between the volume data and the closed curve in screen frame. The 2D screen coordinates of volume data are computed by the transformation from the world coordinate frame to the screen coordinate frame. To increase cutting speed and reduce computation complexity, the optimization algorithm mixing octree with quad-tree decomposition is introduced to decompose eligible volume data. Octree has been widely utilized as spatial data structure for organizing 2D or 3D data, such as in ray tracing [15], mesh simulation [16], mesh coding [17], and geometric modeling [18–20]. Eligible octants are decomposed to eight parts, and the other octants are clipped away or retained according to their coordinate relationship with the closed curve.

The rest of the paper is organized as follows: Section 2 introduces the proposed volume clipping algorithm, which includes filling the closed curve geometry, transforming the coordinate system, and using the optimization algorithm to decompose volumetric data. Section 3 describes how it can perform multiple cutting. Section 4 describes several experiments, including manually segment cardiac images, and Section 5 presents our conclusions.

2. Method

2.1. Binary mask image generation

Selecting or surrounding arbitrarily shaped ROI in volumetric data in a simple and intuitive way and then representing the region with a geometric model, is a complex task [10]. A closed curve is usually drawn on screen to clip away or keep some regions of volume data. According to the coordinates of pixels on the closed curve, which voxels of volume data inside the curve and which ones outside would be decided. Unfortunately, the pixels on the curve are discontinuous. Thus, adjacent pixels as well as the starting and terminal points should be linked to construct a closed curve.

Next, a 2D image with the same size as the rendering window is created. To describe two statuses inside and outside the curve, the image is defined as binary type [9]. All pixels are initialized to 1. An interactive tool is used to plot an arbitrarily shaped curve in the rendering window as shown in Fig. 1(a).

Pixels coordinates of the curve in Fig. 1(a) are extracted to construct the binary mask image in Fig. 1(b). Two containers of points ($\boldsymbol{P}$, $\boldsymbol{Q}$) are utilized to save the pixels coordinates. The pixels of the curve are set to 0, and all the discontinuous pixels coordinates are included in $\boldsymbol{P}$. The curve must be closed to fill the region inside the curve. Adjacent points, such as $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$ and $\boldsymbol{p}_{i+2}$, are often discontinuous and should thus be linked together, as shown in Fig. 1(c). Several points between $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ should be also linked to make the curve closed. Thus, all the pixels between $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ are filled with 0 (Fig. 1d), and the points between the starting and end points are also linked. The linked curve is shown in Fig. 1(d). The new continuous pixels coordinates are placed inside $\boldsymbol{Q}$. The 95 pixels in $\boldsymbol{P}$ increase to 627 in $\boldsymbol{Q}$, as shown in Fig. 1(d).

To distinguish pixels inside and outside $\boldsymbol{Q}$, the area inside the closed curve should be filled with 1, and outside it with 0 pixels. The size of the rendering window is 500×500 (Fig. 1d). Introducing an iterating method to fill the outside region of the curve would be time consuming, because all the pixels in some rows or columns are located outside the curve. In such a case, the smallest bounding box of the closed curve should be defined first to speed up the filling process. Let x_{max} and y_{max} be the maximal coordinates in $\boldsymbol{Q}$ along the x and y coordinate axes, respectively, and x_{min} and y_{min} be the minimal coordinates in $\boldsymbol{Q}$ along the same coordinate axes. Thus, the smallest bounding box with four vertices $\boldsymbol{p}(x_{min}, y_{min})$, $\boldsymbol{p}(x_{min}, y_{max})$, $\boldsymbol{p}(x_{max}, y_{min})$, and $\boldsymbol{p}(x_{max}, y_{max})$ are obtained. The size of the bounding box is 207×235 (Fig. 1e). Compared with the rendering window, the bounding box significantly reduces the search range.

To fill the area between the box and the closed curve, pixels of four edges of the bounding box are placed in a list $\boldsymbol{L}$. The 4-connected neighborhoods searching algorithm [21] is introduced to determine whether pixels inside bounding box are set to 0. If $\boldsymbol{p}_i \in \boldsymbol{L}$, the four neighbors are searched for in the region inside the bounding box. If the value of a neighbor in the mask image is 1, this neighbor is added to $\boldsymbol{L}$. After all neighbors are checked, $\boldsymbol{p}_i$ in the mask image is set to 0, and $\boldsymbol{p}_i$ is removed from $\boldsymbol{L}$. This procedure is repeated until $\boldsymbol{L}$ is empty. The pixels on the curve are previously set to 0; thus, the filling operation is finished until all the pixels between the bounding box and the curve are filled (Fig. 1e). The pixels outside the bounding box can easily be set to 0 according to $\boldsymbol{p}(x_{min}, y_{min})$, $\boldsymbol{p}(x_{min}, y_{max})$, $\boldsymbol{p}(x_{max}, y_{min})$, and $\boldsymbol{p}(x_{max}, y_{max})$. The filling result is shown in Fig. 1(f).

2.2. Relationship of coordinate frames

If the voxels in the volume data projected inside the closed curve require removal, the relationship between these voxels and the curve should be determined first. For convenience, five coordinate frames (i.e., the model coordinate frame $\boldsymbol{F}^m$, world coordinate frame $\boldsymbol{F}^w$, camera coordinate frame $\boldsymbol{F}^c$, normalized view coordinate frame $\boldsymbol{F}^n$, and screen coordinate frame $\boldsymbol{F}^s$) are built to describe how the voxels in the volume data are projected onto the screen through a rendering pipeline. Each set of volume data has its own $\boldsymbol{F}^m$ to describe the coordinates of each voxel. All volume data are located within an $\boldsymbol{F}^w$; thus, the relationship between $\boldsymbol{F}^m$ and $\boldsymbol{F}^w$ should be determined. If we let v_m and v_w be the coordinates of an arbitrary voxel in $\boldsymbol{F}^m$ and $\boldsymbol{F}^w$, respectively, then the rigid motion transformation is expressed as

$$\boldsymbol{v}_w = \boldsymbol{R}_{mw}\boldsymbol{v}_m + \boldsymbol{T}_{mw} \tag{1}$$

where $\boldsymbol{R}_{mw}$ represents a 3×3 rotation matrix and $\boldsymbol{T}_{mw}$ is a 3×1 translation vector.

The camera model is described as a classical pinhole model having intrinsic $\boldsymbol{A}$ with focal length, principal point, pixel skew factor, and extrinsic parameters, including a rotation matrix $\boldsymbol{R}_{wc}$ and a translation vector $\boldsymbol{T}_{wc}$. In $\boldsymbol{F}^c$, the origin O_c is placed on the viewpoint of the camera, and Z_c aligns with the direction of the optical axis. X_c