Human Body Animation

Sue Baldor, Zaiqing Xu, Florian Mayer, Matthias Wiesen, Sangmin Park,
Kevin Baldor, Young-in, Jessica and Vinay
CS 395T

May 9, 2002

Abstract

We define a new framework for human body animation. Skeletal movements defining simple realistic motions are given. Toolkits have been developed for *inverse kinematics calculations*, *tilings*, *parsing animation commands*, etc. Human body animation is traditionally done for purposes unrelated to scientific visualization. The data sets defining the entire human body for any reasonable resolution (say 512 x 512 x 512) is extremely large. We utilize the graphics capabilities of SGI Onyx2 computer. The performer libraries allow high performance visualization with large data sets.

1 Introduction

Animation of the human body involves both rigid and non-rigid transformations. While bones etc undergo rigid transformations, skin and muscle etc are soft and can deform easily to external forces. We need to depict both deformations and a good model for skin, hair etc to obtain realistic animation. The human body gives us a natural skeletal model to help in rigid animation. Finite element analysis was used to produce other deformations. We are currently working on the two issues seperately and need to develop a framework to integrate them.

2 Input data creation

The raw volume data was preprocessed. We had to reduce noise as well as segment skin, muscles and bones by translating the raw volume data into more recognizable values for easier isocontouring. Then we needed to obtain the data sets from the preprocessed data There are two approaches which already are implemented.

2.1 Tiling

The tiling approach[6][7] focuses on the fact, that the volume data is readable slice by slice. The Isocontouring value is used to partition each slice in an inner and an outer region. The tiling algorithm tries to find connections between the inner regions of two adjacent slices in order to create a mesh surface.

There is a tiling implementation from K. L. Chandra Sekhar. The slice data needs to be expressed by polygons, on slice per file. The output is a number of files, where each one represents the regions between two dataslices and contains triangles connecting the inner regions of the surrounding slices.

In order to apply this tiling code, the raw volume data needs to be parsed and polygons for each slice need to be extracted. There is a program called newTrace with a visual interface, where the polygons can be previewed, and manually deleted in order to get rid of polygons gained from noisy data. A tool was also written for that purpose which is described in the section tools-¿slicescanner. A little java viewer (also described in tools section) helps to visualize the extracted polygons. The TileConverter (described in tools section) merges the output files of the tiling program and converts it to an iv file.

2.2 Parallel isocontouring

Isocontouring [2][3][4][5]is a popular method for extracting geometric data of a particular isovalue from volume data sets. This software from Dr Xiaoyu implements a fully scalable parallel and out-of-core isocontouring algorithm. A raw volume data file is needed to extract isosurfaces according to a certain isovalue. The software is developed for multiprocessing.

The generated iv data is passed to a module for level of detail processing.

2.3 Tools developed

There are several tools which were developed within the above mentioned range of tasks.

- checkiso

  This tool searches for the longest chain of values who are above a certain threshold. Since the conversion from little to big endians and vice versa often caused confusion, this tool was useful to get an idea, if the examined file makes sense.

- filedumper

  Dumps either hex, chars, or integervalues of a file starting with a certain byte. Also useful, to check a raw file for correctness.

- filestats

  Creates a histogram over the datawords in a file starting from a certain byte. Also useful for examination of files.

- float2char

  Converts a raw volume file from floatvalues to unsigned charvalues to reduce the filesize of preprocessed raw volume files.

- meshseparator

  Separates connected objects of a mesh data (input: iv file). Making the assumption, that bones are not connected, this program will separate all bones of an entire mesh.

- rawtools

  Helper for raw volume data. It has two functions: it dumps out the header of a raw file, and it allows to cut a subcube out of the volume data, to focus on a local region of the data. This is useful for finding isovalues in small regions with programs like contour_iv, because one doesn't have to look at the entire dataset at once.

- slicescanner

  Used for preprocessing data for tiling. It reads either a ascii (png like) imagefile, or volume data slice by slice in order to generate polygon files which can be used as input for the tiling_win program from K. L. Chandra Sekhar.

- swapendians

  Swaps the endians in a region of a file. The wordlength is arbitrary.

- TileConverter

  Converts the output of K. L. Chandra Sekhar's tiling_win program into a iv file.

- writefloat

  Writes a 4 byte float at a certain adress into a file. This is used to fix wrong headers of raw volume files.

- writeint

  Writes a 4 byte integer at a certain adress into a file. This is used to fix wrong headers of raw volume files.

3 Bone Segmentation

Segmenatation of regions representing the bones in the visible human data set was an essential requirement as the skeleton aids in rigid animation also.

hence various methods were used to perform the segmentation and various data sets were experimented with.

Steps

- Choosing the Dataset
  - Cryoslices
  - freshCT
  - frozenCT
- Image Processing
  - thresholding
  - morphology
- Model production
  - iso-contours
  - tiling

3.1 Choosing the Dataset

To produce the bone geometry dataset, we considered three datasets:

- Cryoslices : these images were obtained by photographing slices of the body. They posess several attributes which make them desirable.
  - high resolution - allows for fine details to be resolved
  - low noise - the high resolution is usable
  - color - bone,internal organs,etc are segmentable
- FreshCT : these images were obtained through computed tomography. This technology has its advantages and disadvantages
  - non-destructive - can be perfomed on living people
  - noisy - X-ray opaque objects produce shadows
  - fresh - organs are more visible
- Frozen CT : these images were obtained after the subject was frozen
  - high contrast - bones are fairly easy to find
  - some noise - when the arms protrude beyond the imageing area, they produce shadows which are not accounted for in other scans producing streaks
  - missing bones - the elbows are outside of the imaging area Of these, we chose the frozen CT dataset because we were unable to produce an automated segmentation scheme for finding bone in the cryoslice data, and the frozen CT dataset had the lowest noise.

3.2 Image Processing

To segment the bone, some image processing was required:

- Thresholding : a variable threshold was used to highlight bone
  - variable threshold - allows for different gray values associated with bone. This was caused by the different imaging distances
- Morpholgy : these techniques alter the shape of the data to remove artifacts
  - erode - removes small objects but decreases the size of everything
  - dilate - fills small holes but increases the size of everything
  - close - tends to connect fine lines which were lost - useful in the skull

3.3 Model Production

We considered two approaches to model production.

- Isocontouring : the model is defined by a mesh which is defined by a constant value.
  - benifits - it is useful for models for which the values have significance

- detractors - doesnt take advantage of the properties of the desired body part in the visual human dataset
- compormise - preprocess the image to accentuate the desired feature

- Tiling : the model is produced by connecting the outline of a body on two adjacent slices.
  - benifits - well suited to the dataset
  - detractors - code was unavailable.

This task was acheived through image processing and isocontouring. However, the isocontouring code did not handle binarized data well and created models which could not be downsampled using our techniques.

4 Skin and Muscle Models

The skin and muscle models were extracted from the cryosection slice data from the Visible Human Project male dataset. Below, an example slice from the dataset is pictured. The cryosection data was chosen for a number of reasons. First of all, muscle does not show up well (or at all) on CT scans. Muscle tissue is easily segmented from the rest of image by simply searching for red. Secondly, the skin was distorted in the CT scans in the area of the nose. When they "zoomed in" to get higher resoltution images of the brain, the area of the nose was outside the scanner's field of view. As you can see in the second image, the nose remains perfectly intact in the cryosection data.

Despite the attractive features of this data set, one must work quite hard to segment out useful features. The approach taken here, was to perform segmentation on each two-dimensional slice. The resulting black and white image can be used for producing isocontours.

Figure 1: Slice from Midsection of Body

Figure 2: Slice from Head Region

4.1 Creating a Mask

The first thing to do is to define the body region within the slice. This mask will narrow down the area within the image to search for various anatomy within the body.

Holes inside the slices and the ice around the body are colored blue. Nothing natural inside the body is colored blue. This is the first piece of information used to determine the body area. To do this, the RGB image is first transformed into the HSV color space (Hue, Saturation, Value). This space is more suitable for identifying specific colors (blue, red) which differ in darkness.

Upon completion of the transformation, a search is performed within the image for values of "blue".

This is accomplished by searching for a range of hues. The range of hues of interest is modeled as a gaussian distribution. The hue of interest is defined as one which is plus or minus one standard deviation of the mean. In this case, all the values of interest, the blue hues, are set to zero. The figure below shows the result. (Note that a black border is drawn around the image and a threshold is placed on value.)

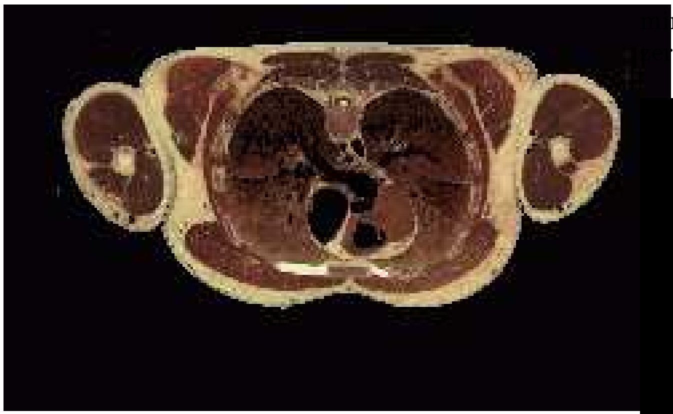

Figure 3: Blue Hues Removed

Each non-zero value is set to be white. The resulting image is rather noisy as shown.

Figure 4: Non Zero Values Set to 255

Much of the noise can be removed using morphological operations. (Linear filters were not chosen so that sharp boundaries would be preserved.) In this case, an open (erosion followed by dilation) with a circular window is applied to the entire image. This removes much of speckel noise. The remaining small blobs are removed by applying a minimum blob size. Blob size is determined by recursively going through each blob and counting the number of pixels in that blob. If the number of pixels present is less than the minimum blob size, the pixels in that blob are set to zero.

Figure 5: Cleaned up Body Blob

There are "holes" still present in the interior of the blob. To remove these, the outer region of the image is flood-filled starting at (0,0) with a value other than 0 or 255 (say 128). The pixels which are not equal to the fill value are set to 255. The outer region is then set back to 0. The resulting image is absent of holes. This result is the final mask. These mask slices are used to construct the skin volume data.

2 Segmenting Out Muscular Tissue

Since we now have a mask to start with, segmenting muscular tissue is much easier. We first and the entire image with the mask. This produces an image which is much like the first one produced in creating the mask. As before, the color space is transformed to the HSV color space. Then, we look for red hues.

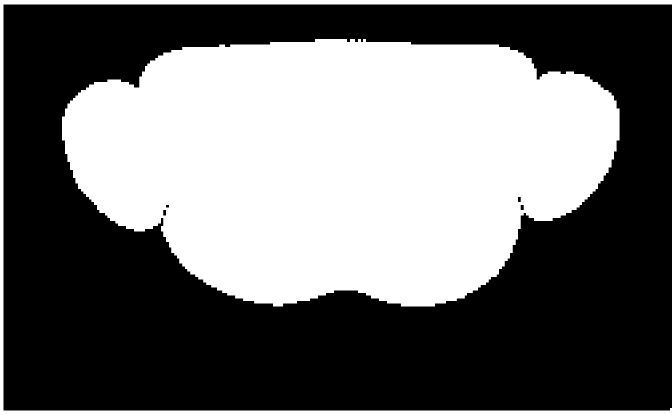

Figure 6: Mask

Figure 8: Binarized Muscle Regions

Instead of zeroing out the matching pixels, the non-matching pixels are zeroed.

Figure 7: Matched Muscle Regions (Red Hues)

The image is noisy, as was seen in the first binarized image for the mask. The image is improved using an open operation and by applying a minimum object size. The flood fill would not help remove inner holes since they are connected to the rest of the image. These holes may actually be desireable as they are an artifact of the muscle texture.

Figure 9: Final Extracted Muscle Segmentation

4.3 Model Production

The slices are saved to individual files. A later tool compiles the slice files into volume data. The volume is saved as .raw format. This file should then be able to be used in existing isocontouring applications to create isosurfaces. Below is an example created from a portion of the skin volume.

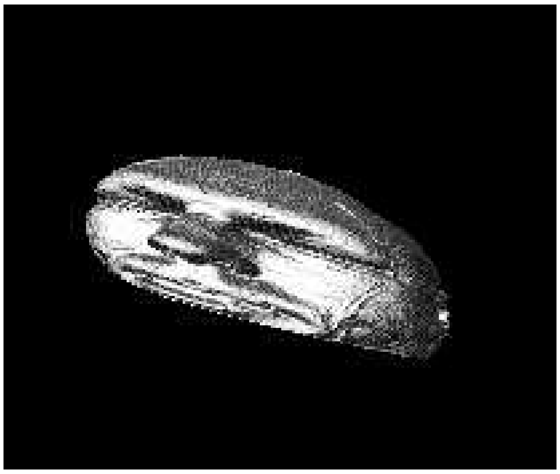

Figure 10: Isosurfaces from Head Area of Skin Volume

5 Deformation of surfaces

5.1 Volumetric Mesh Construction

Various techniques have been proposed for construction of volume meshes [9][10][11][12].

- Perturbation method Given one surface, perform perturbation to get a parallel surface Prism mesh can be constructed between the two parallel surfaces
- Extraction directly from volume data (Seedset Algorithm) Given two iso-values for the volume data, extract tetrahedral mesh between the two iso-surfaces
- Specified boundary Given one specified boundary and an iso-value, to extract tetrahedral mesh from volume data

5.2 Deformation Calculation

- Scaling Set deformation function, e.g. distance
- Finite element method[8] (Abaqus)

6 The animation toolkit

This animation can only support inverse kinematics and was developed for the purpose of human body animation (although any rigid skeletal animation can be performed). Please refer to GATI[1] for another toolkit. It can be used as both a stand alone application as well as a library.

6.1 Usage as a library

- create an *Animation object*;
- intialise it by calling *init_animation(Animation* animation)*;
- load all control data by calling *load_control_data(Animation* animation, char* fname)*;

6.2 The data files required

The actual data files is of no concern to the animation toolkit, the individual rigid objects can be anything. The parameter fname in the load fucntion should be the first comman string of the file names for the following files.

- The tree structure It should end with ".tre" . It contains the description of the hierarchy as a tree structure.
- State diagram It should end with ".dgm" .
- Database of positions It should end with ".pos" .

6.3 Input commands

Animation commands are given as prescribed by the grammer. This should also be made to be extensible. New commands are entered to the library with the standard opengl keyboard interface or through a call *int is_valid_command(Animation* animation, char* cmd)*.

6.4 Output transformations

At every desired interval, one can get the next transformation to be performed with the call to *get_next_transformation(Animation* animation, Transformation* tranformations)*;

7 Conclusions and Future work

Human Body animation using scientific data is especially valuable in the Medical field. We have attempted to solve the various issues involved in it. Special hardware and high-performance libraries were used to obtain efficient rendering. User interaction was increased with an animation toolkit which allowed run time inputs to the system. A large part of the tools required to visualize and animate the human body data set has been developed. We need to next explore issues like flythrough compression, skin and hair modelling, correct muscular deformations, effect of external forces on shape of body etc. A database of movements is also required.

8 Acknowledgements

We thank Dr. Zhang for providing us with a fast parallel isoontouring program. We also thank Anthony Thane from CCV, University of Texas at Austin for providing us with a time varying geometry viewer.

References

[1] The GATI Client-Server Animation Toolkit
C. Bajaj, S. Cutchin Proceedings: Computer Graphics International, CGI93, Communicating with Virtual Worlds, edited by N. Thalmann and D. Thalmann, Springer Verlag, (1993), 413-423

[2] Scalable Isosurface Visualization of Massive Datasets on COTS-Cluster
X. Zhang, C. Bajaj, W. Blanke Proc. of IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics, pg. 51-58, San Diego, CA, 2001

[3] Parallel Accelerated Isocontouring for Out-Of-Core Visualization,
C. Bajaj, V. Pascucci, D.Thompson, X.Y. Zhang In Proceedings of IEEE Parallel Visualization and Graphics Symposium, October 24-29,1999 San Francisco, CA, pages 97 - 104.

[4] Seed Sets and Search Structures for Optimal Isocountour Extraction,
C. L. Bajaj, V. Pascucci, D. R. Schikore TICAM report #99-35

[5] Progressive IsoContouring,
C. Bajaj, V. Pascucci TICAM report #99-36

[6] Tetrahedral Meshes from Planar Cross Sections,
C. Bajaj, E. Coyle, K-N. Lin Computer Methods in Applied Mechanics and Engineering, Vol. 179 (1999) 31-52

[7] Arbitrary Topology Shape Reconstruction from Planar Cross Sections C. Bajaj, E. Coyle, K. Lin Graphical Models and Image Processing,, vol. 58, no. 6 (1996), 524-543

[8] Automated Finite Element of Human Mandible with Dental Implants,
FtterlingS.,Klein,R.,Strasser,W.,Weber,H., The Sixth International Conference in Central Europe on Computer Graphics and Visualization 98.

[9] Simplificaiton of 3-Dimensional Density Maps,
Vijay Natarajan, Herbert Edelsbrunner, Category: Technical Paper

[10] Marching Cubes: A High Resolution 3D Surface Construction Algorithm,
W. E. Lorensen, H. E. Cline, Computer Graphics, vol.21, no.4, pp.163-169, July 1987.

[11] Volumetric Data Exploration Using Interval Volume,
I. Fujishiro, Y. Maeda H. Sato and Y. Takeshima, IEEE Transactions on Visualization and Computer Graphics, Vol. 2, No. 2, pp. 144-155, June 1996.

[12] Interval volume tetrahedrization,
Nielson, G.M., Junwon Sung, Proceedings of the 8th IEEE Visualization '97 conference.