คุณสมบัติ การคำนวณ และการประยุกต์ใช้งานของโครงสร้างกระดูกด้วยวิธีจีโอเดสิก

GEODESIC SKELETONS: PROPERTIES, COMPUTATION AND APPLICATIONS

ปรวัฒน์ วิสูตรศักดิ์

PORAWAT VISUTSAK

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต

สาขาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

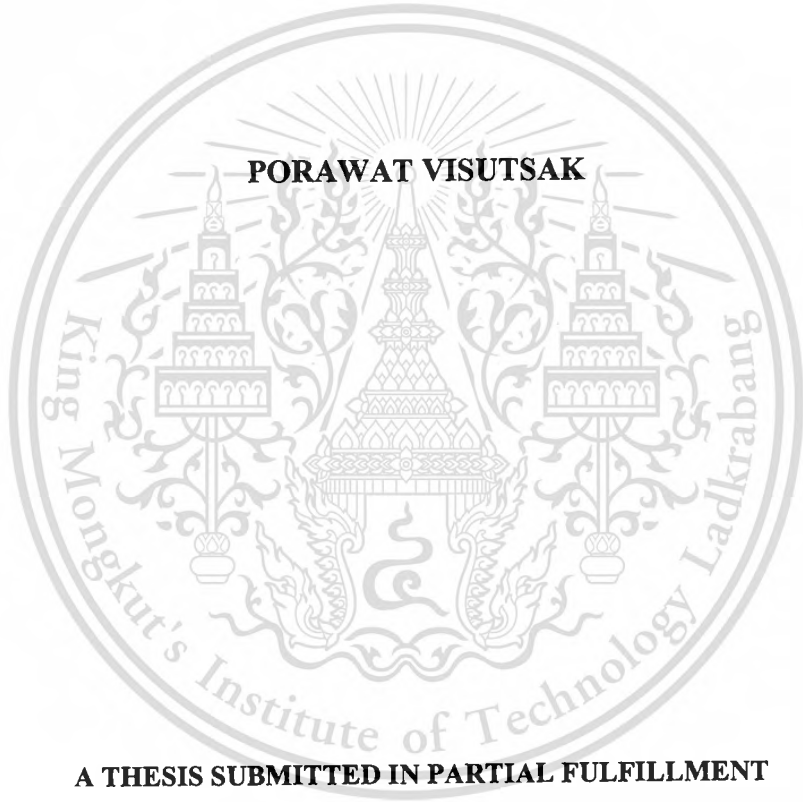
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2555

KMITL-2012-SC-D-002-014

GEODESIC SKELETONS: PROPERTIES, COMPUTATION AND APPLICATIONS

PORAWAT VISUTSAK



A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2012

KMITL-2012- KMITL-2012-SC-D-002-014

หัวข้อวิทยานิพนธ์	คุณสมบัติ การคำนวณ และการประยุกต์ใช้งานของโครงสร้างกระดูกด้วยวิธีจีโอเดสิก
นักศึกษา	นายปรวัฒน์ วิสูตรศักดิ์
รหัสประจำตัว	50067152
ปริญญา	ปรัชญาดุษฎีบัณฑิต
สาขาวิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2555
อาจารย์ที่ปรึกษา	ผศ.ดร.กรกช ประชุมรักษ์

บทคัดย่อ

โครงสร้างกระดูกเป็นส่วนประกอบที่สำคัญสำหรับสร้างการเคลื่อนไหวให้กับตัวละครสามมิติ เทคนิควิธีที่ใช้ส่วนใหญ่ ใช้วิธีการคำนวณมาจากรีปกราฟ และการหาระยะทางที่สั้นที่สุด ซึ่งการคำนวณหาจุดวิกฤตจากอัลกอริทึมระยะทางที่สั้นที่สุด เพื่อนำมาใช้ในการสร้างรีปกราฟนั้นอาจจะทำให้เกิดข้อต่อของโครงกระดูกที่ไม่จำเป็น งานวิจัยนี้จึงได้พัฒนาแนวคิดใหม่เพื่อใช้ในการคำนวณโครงสร้างกระดูกของโมเดลสามมิติโดยใช้วิธีการแปลงมีเดียลแอ็กซิสของบลุม และอัลกอริทึมระยะทางจีโอเดสิก ประโยชน์ของการใช้ฟังก์ชั่นระยะทางจีโอเดสิกและค่าพารามิเตอร์ที่เกี่ยวข้อง จะช่วยให้เกิดความคงสภาพต่อการเปลี่ยนแปลงทางโทโพโลยีของโมเดล งานวิจัยนี้ยังได้เสนอแนวคิดสำหรับการปรับตำแหน่งของข้อต่อที่เรียงต่อกันบนโครงสร้างกระดูก โดยวิธีการหาระยะทางสะสมจากคอร์ดถึงจุดบนโมเดลของฮันและโพสตัน โครงสร้างกระดูกที่คำนวณได้ จึงมีความสอดคล้องกับโครงสร้างของโมเดล และยังสามารถแสดงคุณลักษณะเฉพาะที่สำคัญของโมเดลในรูปแบบที่เหมาะสม โครงสร้างกระดูกที่คำนวณจากแนวคิดนี้ จะถูกนำไปประยุกต์สร้างการเคลื่อนไหวให้กับตัวละครสามมิติต่อไป

คำสำคัญ : โครงสร้างกระดูก, โครงสร้างกระดูกที่ได้รับการปรับแต่ง, ระยะทางจีโอเดสิก, การแปลงมีเดียลแอ็กซิส, ระยะทางสะสมจากคอร์ดถึงจุด

Thesis Title	Geodesic Skeletons: Properties, Computation and Applications
Student	Porawat Visutsak
Student ID	50067152
Degree	Doctor of Philosophy
Program	Computer Science
Year	2012
Thesis Advisor	Asst. Prof. Dr. Korakot Prachumrak

ABSTRACT

Skeleton is at the main interest of 3D character animation. The most common techniques for skeleton computing are based on the Reeb graph and the shortest path finding. Using only the shortest path algorithms for extracting the critical points and constructing the Reeb graph over the surface of the model may generate unwanted skeleton joints. This dissertation is concerned with the development of a new approach to compute the skeleton of the 3D meshed model, based on Blum's Medial Axis and geodesic distance algorithm. This study gains the benefit of geodesic distance functions and parameterization that allow for efficient handling of topological changes of dynamics curves and surfaces. Thus, the new approach can provide the robustness against any changes of a rotation and/or a translation of the 3D meshed model. This study also introduces further improvements in order to adjust the location of consecutive joints along the skeleton, by using Han and Poston's Chord-to-point Distance Accumulation. Consequently, all skeleton joints are regenerated corresponding to the structure of the model, while preserving the essential shape characteristics in a compact form. To motivate this work, the major application based on these novel approaches is presented showing the 3D character animation.

Keywords : Skeleton; Skeleton smoothing; Geodesic distance; Medial axis transform; Chord-to-point distance accumulation.

Acknowledgement

I would like to thank my advisor, Asst.Prof.Dr.Korakot Prachumrak, for her guidance and support throughout the entire course of my studies at King Mongkut's Institute of Technology Ladkrabang. I would also like to thank my other committee members, Asst.Prof.Dr.Nualsawat Hiransakolwong, Asst.Prof.Dr.Nunthika Benjathapanun, Dr.Anantaporn Srisawat, and Dr.La-or Kovavisaruch, for their helpful suggestions and comments. I would also like to thank Dr.Chakrit Watcharopas for his useful comments on the defensive examination day.

Contents

List of Tables

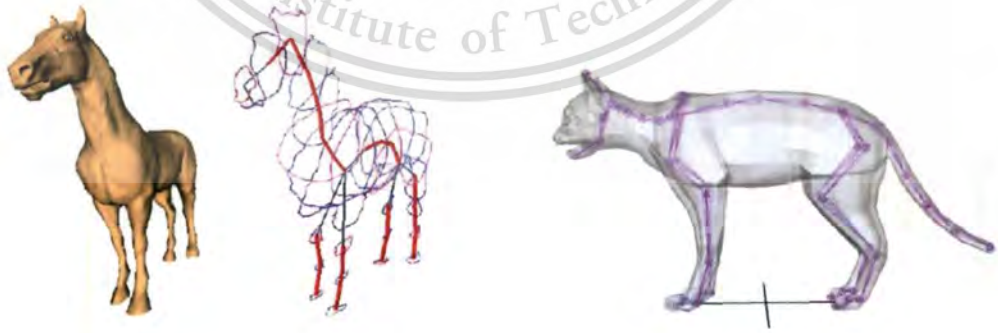
List of Illustrations

Chapter 1

Introduction

The need for a reduced representation of shape was arose from the early ages of computer graphics and computer vision, motivated by the limited resources of computer systems. However, as computers became more sophisticated, more powerful and cheaper, similar progress in imaging technology, 3D scanning and computer graphics produced larger and even more complex shape models. As a result, the quest for efficient shape abstractions is still an active area of research. One such abstraction is the representation known as the skeleton.

The skeleton is a basic structure of representing the 3D objects, frequently used in computer graphics, especially in the areas of character animation and 3D modeling. Using a skeleton as an abstraction of an object has two major benefits. First, it can contain both shape features and topological structures of an original object. Another benefit depends on its characteristic to capture the essential shape of a 3D object in a low-dimension form. Figure 1.1 shows a skeleton of a 3D model, (a) a skeleton of a horse computed by level set diagram [1], and (b) a control skeleton for a character animation drawing by the artist [2].



(a) (b)

Figure 1.1 (a) A skeleton of a horse using a level set diagram [1]. (b) A hand-drawing control skeleton created by an animator [2].

1.1 Problem Statement

A skeleton is an articulated figure composed of the joints and the segments linking them, used to define and drive the animation of a virtual figure such as a human, a creature, and a made-up monster. A skeleton of a 3D model can be either created manually by the professional artists or automatically calculated by a computer programming.

In the case of the realistic animation of a character, the first technique seems to be the most popular option chosen by artists, even though it needs a skilled user and this is a time-consuming task. Actually, a skeleton mock-up can be initially created by artists relatively quickly, but often need to make many adjustments during the rigging process since the skin is very sensitive to the exact location of the skeleton's joints [2]. Therefore, they often have to go back and forth several times between skeleton skinning and testing animation before getting it right. The well-known methods have been developed to generate a skeleton in a graph-like or a curve-like form [1], [3], [4], [5], [6]. Unfortunately, these approaches do not suit for producing a skeleton for use in a character animation because they need the additional processes to eliminate the redundant skeleton branches that may be generated during the skeleton extraction process [7].

Recent work on semi-automatic skeleton extraction is introduced by [2], [8]. This system allows users to select the starting point on the character model, and then it generates a skeleton to match the ones that are created by hand by professionals in most biped and quadruped cases. A method for fully automatic generation of a control skeleton is proposed by [8]. The main task of the system involves discretizing the figure, computing its discrete medial surface, and then using the discrete medial surface both to create the skeleton and to attach the vertices of the model to that structure. However, a major drawback of [8] is only features with a size greater than the voxel size can be taken into account. This often leads to computationally expensive algorithms.

1.2 Aim

In particular this thesis introduces efficient solution to the problems associated with the medial axis according to [7], [9], [10], [11] which produced the smoothed skeleton of 3D objects that are not affected by geometrical transformations (rotation, translation and scaling), and the topology information can be preserved through deformations, twisting, and stretching. Moreover, the proposed algorithm can generate a one-voxel thick skeleton in a graph-like form, which is useful for animation purpose (the implementation will be shown later on in chapter 5).

This work also shows the application of this theory to produce the realistic animation of biped and quadruped models and other objects in general, and to determine if the proposed algorithm has any skeleton, which most often requires some manual intervention from the artists. It is possible, however, to automatically produce such skeletons by using traditional image processing techniques (such as the medial axis transform) which can be extended to 3D. But these techniques require expensive computational time, and they do not suit to use in a character animation since they produce the spurious skeleton joints.

The most significant contribution of this thesis on skeleton research is the use of the medial axis together with the geodesic distance functions. By using the geodesic approximation algorithm proposed by [12] to compute the "single source, all destination" shortest path on a surface of the model, any changes of a rotation and/or a translation do not affect the value of the function. Thus, this becomes the major property of the function of geodesic distance that gives the advantage of being invariant to a rotation and a translation.

The basic idea of this approach is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region determination. Applying the function of geodesic distance can guarantee that the new approach is invariant to a rotation and a translation, and it is also robust against the changes in the connectivity on the 3D shapes. Unfortunately, this raises the problem of producing meaningless joints since the location of skeleton joints does not match the real

bone structure of the model. Thus, the additional filtering process is needed; the smoothed skeleton can be obtained. Chord-to-point distance accumulation [13] then be applied in order to split the smoothed skeleton into the segments, and then the new joints are located correspond to chord-to-point distance accumulation values.

This thesis attempts to test an implementation of the smoothed 3D skeleton for practical applications in a character animation and how it compares with more recent advances in the area. As far as the skeletonization algorithm is concerned, some improvements have been implemented and others introduced, that reduce the overall running time of the skeletonization process and perhaps the overall quality of the end result.

1.3 Contents and structure

This thesis is organized into six chapters. Chapter 2 reviews the relevant literature on the subject of skeletonization, together with any relevant background information. More specifically, the geometric concepts of the medial axis, medial surface, and the geodesic distance function, and the theory behind the smoothed 3D skeleton is more thoroughly investigated. In addition, state of the art, alternative skeletonization methods and their potential uses in modeling and animation are reviewed.

Chapter 3 goes a step further and describes the algorithm behind the smoothed 3D skeleton theory, presented in the previous chapter. The problems and limitations of the original algorithm are explained together with the methods proposed for their solutions and improvements to the algorithm.

In chapter 4 contains testing and the running times. Furthermore, this chapter compares the different choices for algorithms and their impact on the running time, complexity and the overall quality of results.

Chapter 5, design and implementation are mentioned. The smoothed skeletons are then used to produce a character animation and appropriate conclusions are drawn.

Chapter 6 summarizes the research and presents the overall conclusions. Suggestions for further research together with theoretical and algorithmic improvements to this specific approach are discussed.

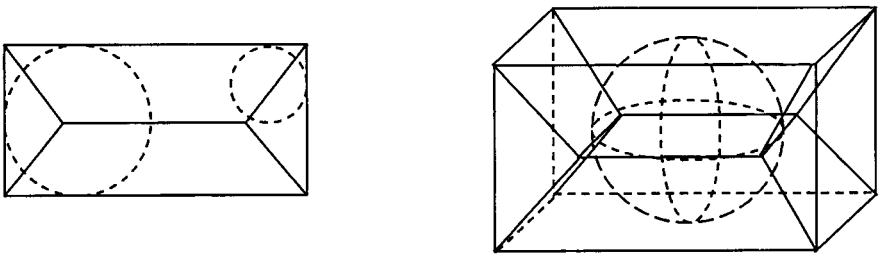
Chapter 2

Literature Review

This chapter provides a briefly detail of both traditionally used and recently developed skeletonization methods, their theoretical concepts, advantages and problems, citing relevant publications as necessary. A special type of skeleton, the smoothed geodesic skeleton, is more closely examined and the theory behind the algorithm used in this thesis is presented. Advances in the area of skeleton applications, especially since the implementation of this thesis are also included as an alternative to the chosen algorithm.

As described earlier in chapter 1, the skeleton is one-dimensional (1D) abstraction of a three-dimensional (3D) object (or a two-dimensional (2D) shape). The simplest result of a skeleton computation process is one where discrete sample points are gradually removed from the object or shape, while preserving its continuity and connectivity, in order to generate a more simple representation of the object or shape [9]. The most common technique to compute a skeleton, that has been the standard for many years, is the medial axis in 2D and medial surface in 3D. The algorithm for computing the skeleton in term of the medial axis is often refers to as the "grassfire" algorithm. The main idea of the "grassfire" algorithm is lighting a fire, started from the border of the object, and let it burn into the object at a constant speed; it will then meet in the medial axis. One starts on the border of the object and strips away one layer of pixels after another until one reaches points that fire reaches from two directions [14]. The medial axis transform of the region is the set of points reached by more than one fire at the same time.

The medial axis [15] of a 2D shape is defined as the locus of points which are equidistant from at least two points on the boundary of the object [16]. In the 3D case, the corresponding representation is called the medial surface [17] because it can also contain surface patches. Figure 2.1 shows the medial axis for a 2D shape and the medial surface of a 3D object.



(a) (b)

Figure 2.1 Medial axis of a 2D shape (a), medial surface of a 3D object (b) and their inscribed disks and spheres respectively.

Although the medial axis/surface and the skeleton are closely related, they are not exactly the same. The other approaches to construct skeleton are Voronoi skeletons and distance transforms. Voronoi skeletons are based on a special property of the Voronoi diagram. More specifically, it has been shown [18] that, under appropriately chosen smoothness conditions and as the sampling rate increases, the vertices of the Voronoi diagram of a set of boundary points will converge to the exact skeleton. This idea has been used successfully [19] in developing skeletonization algorithms in 2D and 3D. This method can preserve the topology and accurately localize skeletal points but only if the boundary is sampled densely enough. Unfortunately, however, the practical results of Voronoi skeletonization methods are not invariant under Euclidean transformations and can require an additional optimization step, which in 3D can have a high computational complexity.

In distance transform method, the distance functions are based on the fact that the locus of the skeletal points is coincidental with the singularities (creases or curvature discontinuities, also known as shocks) of a distance function to the boundary. An appropriate distance metric (Euclidean distance) is first used to calculate the distance transform of the object. The local maxima of this distance function or the corresponding discontinuities in its derivatives are then detected, each of which indicates a skeleton point. The numerical detection of these singularities, however, is in itself a nontrivial problem. Whereas it may be possible to have good localization of the skeleton points, ensuring homotopy with the original object is difficult.

In many applications, however, only an approximation to the true skeleton can be extracted. Therefore, these approximations need to fulfill two main requirements if they are to be considered as true approximations to the skeleton. The first is topological, which is the requirement that the skeleton will retain the topology of the original object [10] (i.e. properties of objects which are preserved through deformations, twisting, and stretching). The second is geometrical, which requires that the skeleton is as thin as possible, connected, in the middle of the object and invariant under the most important geometrical transformations, including rotation, translation and scaling [10].

Certainly these requirements are not exhaustive and may differ for different application areas. For example, thin skeletons (one-voxel thick) with primitive features are required in order for similarity comparisons in the area of the object recognition applications. On the other hand, skeletons with detailed geometric information are necessary for surface reconstruction applications, so that the approximation error in the reconstruction process must be minimized. There are indeed many application areas where the skeleton of an object or shape can be used. Some of them are:

Figure 2.2 Individual matches in the database. The node-to-node matches of the skeletons are also shown.

1) Fast object recognition and tracking: This is the process of determining which, if any, of a given set of objects appear in a given image or image sequence. Since the skeleton retains shape information and is a reduced representation of the original shape it can be used to match shapes/objects much faster than conventional methods, because of the fewer number of points in the skeleton. The same idea lies behind the use of feature tracking, which is actually the correlation of an extracted object from one data set to the next [20]. Figure 2.2 shows the skeletons matching of the objects.

2) Animation control: Animation usually consists of three steps: modeling, deformation and rendering. The step of deformation for polygon based animation is carried out with the help of skeletons. The animator moves the skeleton which in turn causes the object to deform in the same way [21]. Figure 2.3 shows the control skeleton for the character animation.

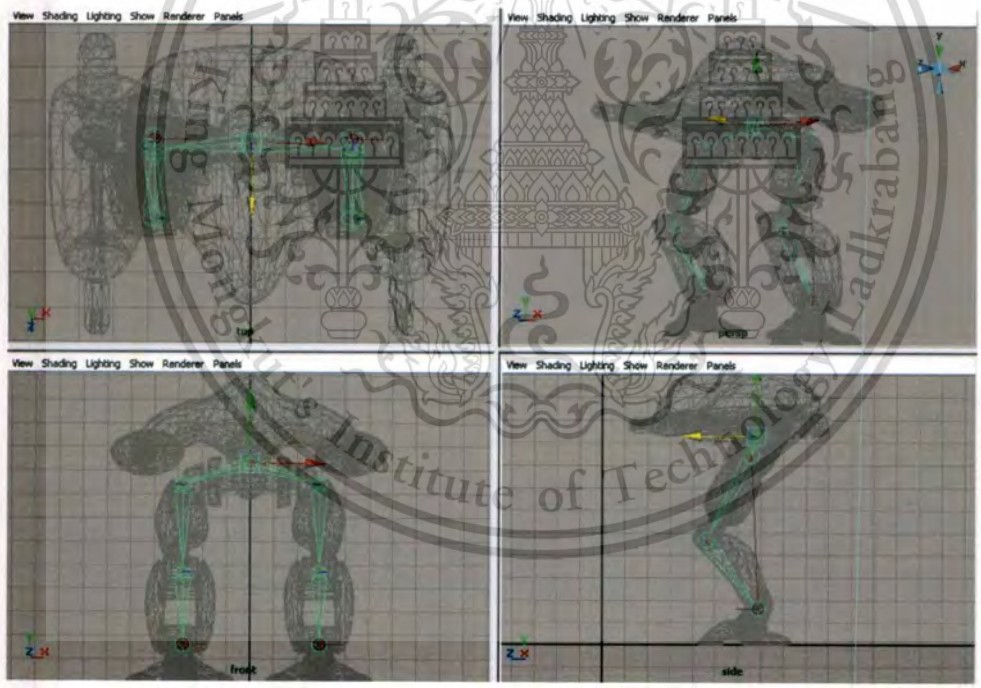


Figure 2.3 The control skeleton for the character animation in Maya software.

3) Surface reconstruction: This is an important process in geometric modeling for generating piecewise smooth surfaces from data captured from real objects. A medial axis transform skeleton

performs well in this case because it retains all the boundary information of the original object. Therefore, by applying the inverse transformation, it is possible to reconstruct the surface of the object [22], [23]. Figure 2.4 shows the surface reconstruction of [23].

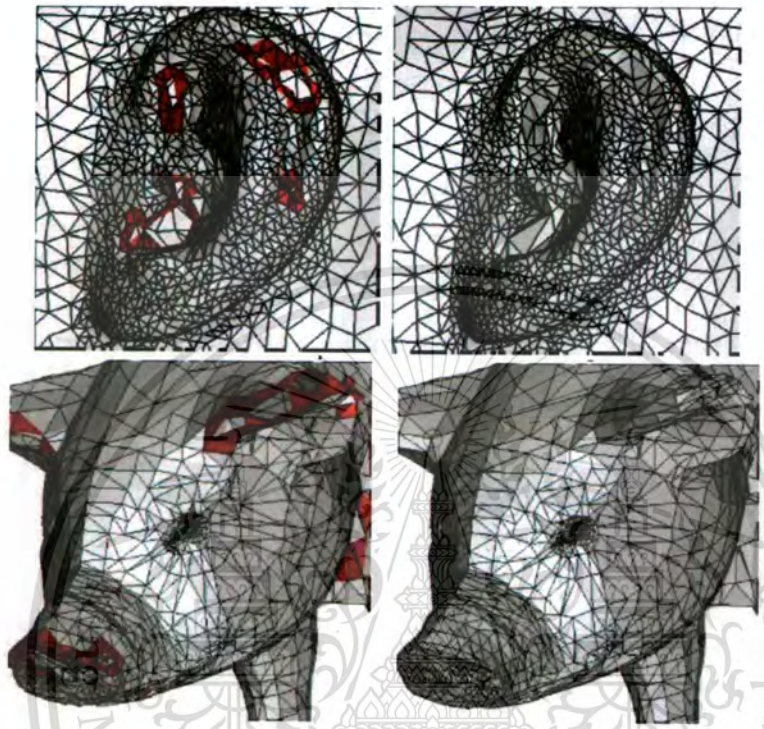


Figure 2.4 The surface reconstruction of the objects. Figure 2.4 (left) shows the holes in the preliminary surface while figure 2.4 (right) shows how they are filled.

4) Shape abstraction: Skeletonization from thinning gives a set of unconnected voxels, which can then be connected into a skeleton tree. By using the distance transform values of these voxels, it is possible to construct a graph, the minimum spanning tree (MST) of which will automatically connect the voxels. The MST contains shape information, and is subject to processing by standard, well known algorithms for searching, traversal and shortest path discovery. This in itself greatly expands the use of a skeleton to several domains [21]. The skeleton thinning is shown in figure 2.5.

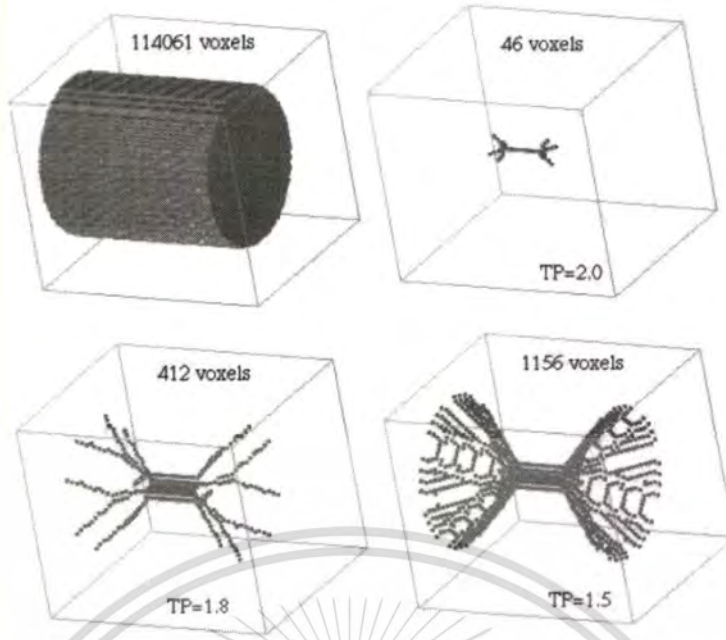


Figure 2.5 The skeleton thinning method.

5) Text processing and character recognition: This is the process of identifying machine printed and handwritten characters in an automated or a semiautomatic manner. Given a character image its skeleton is extracted and then compared with a set of templates (or prototypes) [24].

Figure 2.6 The handwritten character recognition.

6) Automatic navigation: By using skeletonization, the centerline of an object can be detected quite accurately and then can be used as a path for camera navigation in computer graphics and animation, or for path navigation in robotics. If the skeleton is centered, collisions with the object boundaries are avoided [25]. Figure 2.7 shows the distance-field based skeletons for virtual navigation

Figure 2.7 Distance-Field Based Skeletons for Virtual Navigation.

2.1 Traditional Skeletonization Methods

Traditionally, there are two kinds of approaches to constructing the skeleton of an object. One is to compute the skeleton directly from the object surface points, and another is to extract the skeleton by constructing a distance field of an object. Some typical examples of both of these methods are examined here.

2.1.1 Direct Computation Method

The medial axis

Perhaps the most commonly used skeleton is the medial axis, derived from Blum's original work [15]. Basically, the medial axis can be thought of as a branching geometric centerline of a 2D shape. In three dimensions, it becomes a centralized surface and the term medial surface is used instead, as mentioned earlier in chapter 1. The medial axis is defined as the locus of the centers of all the maximal disks, which are inside the shape and intersect the boundary of the shape in at least two different points. Similarly, for 3D, the medial surface is the locus of the centers of the maximal spheres. Both of these concepts can be extended to higher dimensions. Figure 2.8 shows how discrete

points on the skeleton are computed via maximum inscribed circles, and also the medial surface in 3D (solid lines).

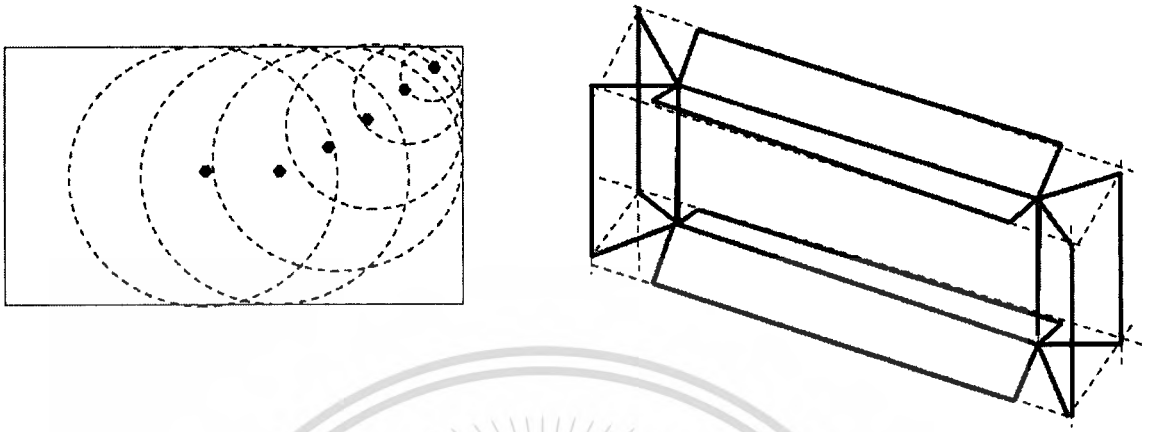


Figure 2.8 Left: Maximal circles interior to the rectangle with their centers. The centers are points on the medial axis. Right: 3D medial surface of a box.

Together with the medial axis, the radius of the maximal disks (or spheres) is stored. The idea behind this, is that the extra information is used to reconstruct the original object. The medial axis in conjunction with this distance function defines the medial axis transform (MAT). The object's boundary and its MAT are equivalent, and one can be computed from the other. Usually, a two dimensional shape will be transformed into a one dimensional structure, while a three dimensional object will be transformed into a 2D surface.

The medial axis has been studied extensively, especially as a means for shape representation [26], [27], object decomposition for mesh generation [28], shape morphing and animation [29] and motion planning [30]. This will be briefly shown in the next paragraph, how the MAT produced undesirable skeleton branches when the curvature of the object surface varies everywhere (noisy surface). Since this is a morphologically meaningless property of the MAT, some studies have focused on the simplification of a noisy MAT by means of pruning [31], [32].

To illustrate this, considering the medial axis of the rectangle from Figure 2.1(a) above and a small bump is added in the boundary of the shape. The resulting skeleton can be seen in Figure 2.9(a). It is obvious that the medial axis can be very sensitive to small changes in the shape. The medial axis is also very sensitive to noise. To illustrate this "salt and pepper" noise was added to the original rectangle and its skeleton was computed. As can be seen in Figure 2.9(b), the skeleton attempts to connect each noise point to the skeleton obtained from the noise-free image.

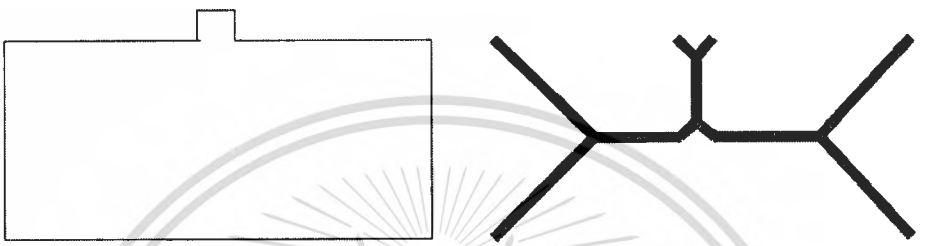


Figure 2.9 (a) Rectangle with small bump and its skeleton.

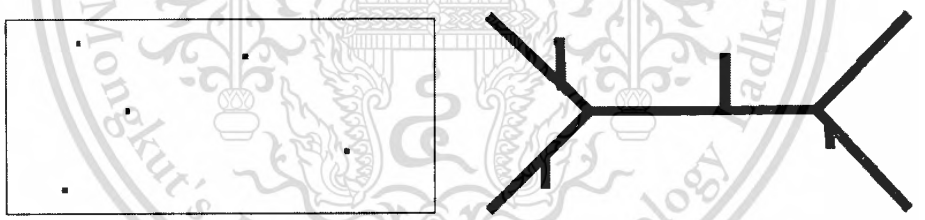


Figure 2.9 (b) Noisy rectangle and its corresponding skeleton.

However, this is not the only problem with the medial axis. The medial axis is not invariant under translation and rotation transformations (i.e. the medial axis of a transformed shape is not the same as that obtained by applying the same transformation to the original medial axis). This is best illustrated in Figure 2.10, where the medial axis of an elliptical pie shape is shown. When the shape is transformed, the calculated medial axis has additional branches. Clearly, it does not correspond to the original axis. More detail of this problem will be revisited again, later in the next chapter.

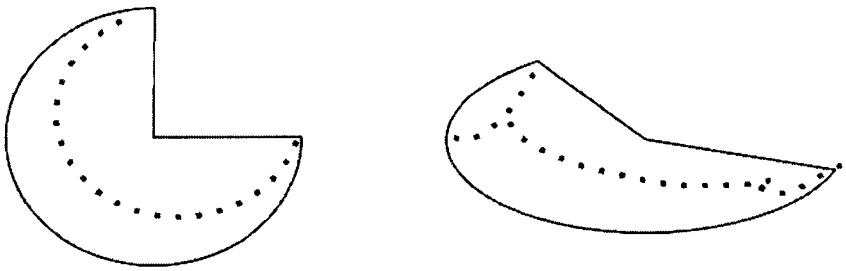


Figure 2.10 A concave curve and its medial axis, and after transformation.

Voronoi diagram

Exact computation of the medial axis is difficult in general. [33] and [34] have created algorithms for computing the exact medial axis for some special classes of shapes but even these algorithms had to deal with the numerical instabilities associated with the medial axis computation. An alternative method for the exact computation of the medial axis is the creation of the Voronoi diagram from the shape boundary points, and then the approximation of the MAT from that. The Voronoi diagram is the partition of n points in a plane into n convex polygons such that each polygon contains exactly one point and every point in a given polygon is closer to the point in that polygon than to any other point (Figure 2.11). The generated polygons are known as Voronoi polygons, as illustrated in Figure 2.11, the points where the boundaries of these polygons meet, form an approximation to the medial axis.

Voronoi diagrams have been widely adopted in the construction of 2D and 3D skeletons [35]. More recently, [22] have proposed the "Power Crust" algorithm for MAT approximation and surface reconstruction, based on the idea of α-shapes (a generalization of the convex hull). The algorithm first computes the Voronoi diagram of the scattered data points, and then retrieves a set of polar balls by selecting candidates from the Voronoi balls that have maximal distance to the sampled surface. After labeling these polar balls, the object described by the data points is transformed into a polar ball representation. Connecting the polar ball centers gives a good approximation to the MAT. Although

this algorithm works well on dense data sets, it faces some difficulties when the data is under sampled by producing holes or other artifacts in the vicinity of the under-sampling.

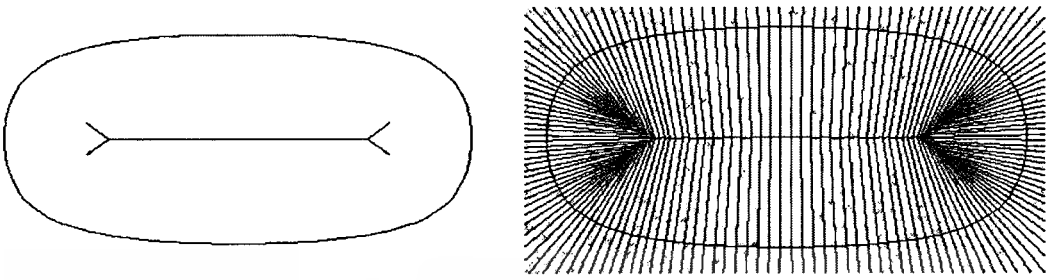


Figure 2.11 On the left a convex curve and its medial axis, and on the right the Voronoi diagram of the curve, together with its approximated medial axis.

Reeb Graph

A relatively up-to-date direct computation method is to use the Reeb graph. This graph was introduced by [5], based on Morse theory. Given an object surface M and a smooth valued function f defined on it, Morse theory provides the relationship between the critical points of f (points where the tangent plane is horizontal) and the global topology of M. Morse theory states that the shape of the pair (M, f) is represented by the sequence of the homology groups of the level sets (i.e. the set of points x from M such that $f(x)<k$, $k \in \mathbb{R}$. The critical points of f determine the homology groups of M. The homology groups contain the original shape's properties encoded as the number of connected components, holes and cavities of the shape. It is possible to fully describe the surface of the shape by a finite collection of level-sets, and also get a more complete description rather than simply knowing the global homology. By varying k, it can be introduced topological changes on the level-sets and obtained a discrete representation of the shape. This can be then encoded into a topological graph called the Reeb graph.

The Reeb graph can be represented as a one dimensional skeleton, provided by a continuous scalar function on the surface M. Although the global homology of the surface M does not change as f

varies, the topology of the level-sets depends on f. In this way, the choice of f determines a collection of shape descriptors with properties dependent on the function f that characterize the object's surface depending on the application context. Typically, f is chosen as the height function $(\forall p=(x,y,z) \in \mathbb{R}^3, f(x,y,z)=z)$ whose critical points (i.e. peaks, pits and passes) are useful for shape description. The main drawback of using the height function is that it produces graphs which are dependent on the orientation of the object in space. Figures 2.12 and 2.13 show how the Reeb graphs of real and generated data look.

Figure 2.12 Reeb skeleton (bold) of real 3D data.

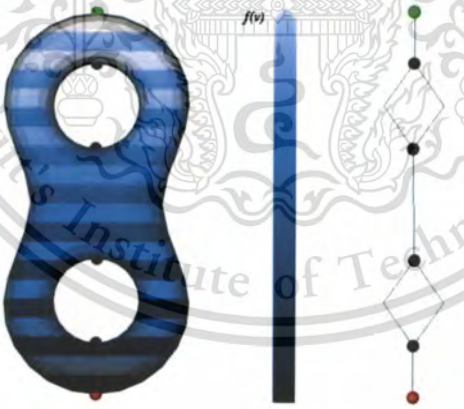


Figure 2.13 Reeb graph of a torus. The graph will correspond to the topology of the shape.

Some research on Reeb graphs, has been carried out using the geodesic distance (which is the minimum length of the paths connecting two vertices u and v of a finite graph) as the mapping function. More specifically, [3] have used the idea of Reeb graphs to construct a skeleton called the

Level-Set Diagram, in which the geodesic distance from a source point is used as the function *h*. Their solution however, is dependent on the choice of source points, and for that reason it is not unique. Different Level-Set Diagrams can be produced for the same model. [2] have defined the Multi-resolution Reeb Graph based on the geodesic distance, and used it as a search key for topology matching. They have developed a series of Reeb graphs for an object at various level of details, partitioning the object into regions using the height function at every level.

The skeleton produced by the Reed graph has some interesting properties that make is especially useful as object recognition and matching key. Firstly, a Reeb graph will always be a one dimensional graph structure and will never have any higher dimensional components such as degenerate surfaces. Secondly, by correct choice of the continuous function *f*, it is possible to obtain a Reeb graph that is invariant to translations, rotations and connectivity changes that occur from simplification, subdivision and re-mesh. The re-meshing property also helps to ensure that such a Reeb graph is quite resistant to noise.

2.1.2 Distance Field Method

The methods we have seen above might not be suitable for large input datasets. For example, building a 3D Voronoi diagram, given N data points, requires at least $O(N^2)$ computational time complexity [36]. Therefore, another equally popular approach is to use the distance field of the object to extract the skeleton. First, the distance field of the object is constructed, then the algorithm finds the local maxima of the distance field, and finally these maxima are connected in order to find the skeleton.

There have been several distance field methods proposed for skeleton computation. Most notably, [37] have implemented a two dimensional MAT by minimizing the distance field energy of an active contour model. [38] have proposed a voxel-coding method based on recursive voxel propagation. Their algorithm works by using a set of seed voxels and a coding scheme to construct connectivity relationship and distance fields. [39] proposed a penalized distance algorithm for skeleton extraction from volumetric data. The algorithm uses a distance field and Dijkstra's algorithm to get a rough

approximation of the skeleton, which is then refined by discarding redundant voxels. [40] have proposed a skeletonization method for virtual navigation based on the distance from boundary (DFB) field, which contains the Euclidean distance from each voxel inside the 3D volumetric environment to the nearest object boundary. Even more recently, [8] have presented a distance field method that uses the Euclidean distance, for automatic generation of the control skeleton of an articulated object (i.e. a skeleton that can be used to control the animation of the object by specifying stick-like skeletons, and often used in this way for animating avatars in virtual environments with the stick like skeletons defined in the HAnim standard [HAnim]). The discrete medial surface (DMS) is used as a first approximation of the medial axis transform, and via simplification of voxel paths their system produces a relatively good control skeleton for 3D objects.

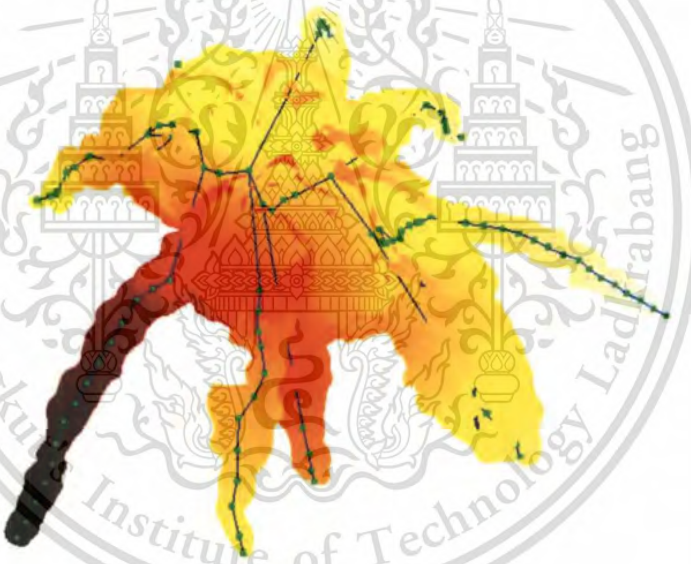


Figure 2.14 The skeleton extracted from the distance field method.

2.2 The Proposed Skeletonization Method

[7] used the skeleton pruning method based on the medial axis together with the geodesic distance function as the distance measurement between each joint on the skeleton. Being invariant to a rotation and a translation is the major advantage of the geodesic distance function, thus, any changes of a

rotation and/or a translation do not affect the value of a measured distance. The basic idea of the skeleton pruning method is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region construction since the generation of redundant skeleton joints will seriously disturb the topology of the skeleton.

(a) The original 3D mesh model

(b) Computing the degree of each point for pruning the spurious skeleton joints

(c) Constructing the pruned skeleton

Figure 2.15 The skeleton pruning method.

Figure 2.15 shows an overview of the skeleton pruning method. The first step of the method is to compute the degree of each point from the 26-neighborhood structure, and determine which point are the end points, the middle points, the connection points, and the branched connection points. By the assumption that the pruned skeleton is one-voxel thick, the skeleton end point is a point that has a degree one. The middle point is a point that has a degree two. The connection point is a point that has a degree three or higher and all the other neighbors are either the end points or the middle points. The branched connection point is a point that has a degree three or higher and at least one of its neighbors is neither an end point nor a middle point. The 26-connected branched connection points will form the branched region. The second step is to determine the centroids based on Blum and Nagel's definition of a skeleton, and find the terminator points in each branched region (the terminator points are all the end points and the middle points that are 26-adjacent to the branched region). The next step is the computation of the geodesic distance from the particular centroids to all the terminator points. The Dijkstra's algorithm is applied in this step in order to find the shortest paths between the centroids and the terminator points. The branched connection points that are not on any of the shortest paths have been removed, and the pruned skeleton can be generated from the remaining points.

In the next chapter, the detail and the algorithms of the recent method will be presented later on. Where possible, the developed algorithms which provide a one-voxel thick in a graph-like form will be illustrated.

Chapter 3

The Skeleton Pruning-Smoothing algorithm

The previous chapter attempted to provide an overview of many applications involving skeletons from various scientific fields, such as object recognition and tracking, animation control, shape abstraction and others. It also discussed the theory behind the skeletonization methods, and contrasted the drawbacks of each method. In this chapter, the study will show the algorithms together with the mathematical concept of the proposed algorithms. It begins with the distance function for measuring the distance of each skeleton point, and then it will describe the method for extracting the skeleton. Finally, the skeleton adjusting and the locating of the smoothed joints by using the cord-to-point distance accumulation will be illustrated at the end of this chapter.

3.1 The Geodesic Distance Function

A Riemannian space is a mathematical geometry concept that studies curves and surfaces in higher dimensions, giving a precise meaning to concepts like angle, length, area, volume and curvature. On a Riemannian system, the geodesic distance is the distance between two points on the surface of the model, computed by using the Dijkstra's algorithm to find the shortest path between the two points on the surface made up by n points [11]. The following definitions characterize the geodesics based on [12].

Definition 1. Given R^3 be a surface in a Riemannian space G, and source vertex $v_s \in G$, an explicit representation of the geodesic distance function D: $G \rightarrow R^3$. For any point $p \in G$, this function $D(p)$ returns the length of the geodesic path from p back to the source v_s. The approximation of $D(p)$ can be given by

$$||x_i - x_j|| \approx D(x_i, x_j) \text{ when } x_j \approx x_j \qquad (3.1)$$

Consider the length of a curve S, represented in R^3 by equations

$$S: x^i = x^i(t), t_1 \leq t \leq t_2, \tag{3.2}$$

Definition 2. The distance s between two points t_1 and t_2 on a curve $x^\gamma = x^\gamma(t)$ in R^3 is given by

$$s = \int_{t_1}^{t_2} \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}\, dt, \qquad (\infty, \beta = 1, \ldots, n). \tag{3.3}$$

The minimum of eq. (3.3) will be yielded a geodesic of the space, by using Euler's or Lagrange's equations:

$$\frac{\partial F}{\partial x^k} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{x}^k}\right) = 0 \tag{3.4}$$

with

$$F = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta} \tag{3.5}$$

Thus,

$$\frac{\partial F}{\partial x^k} = \frac{1}{2}\left(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta\right)^{-1/2} \frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta \tag{3.6}$$

$$\frac{\partial F}{\partial \dot{x}^k} = \frac{1}{2}\left(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta\right)^{-1/2} 2g_{\infty k}\dot{x}^\infty \tag{3.7}$$

since

$$\frac{ds}{dt} = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta} \tag{3.8}$$

Euler's equations can be written in the form

$$\frac{d}{dt}\left(\frac{g_{\infty k}\dot{x}^\infty}{\dot{s}}\right) - \frac{1}{2\dot{s}}\frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta = 0, \tag{3.9}$$

and, carrying out the indicated differentiation, which obtain

$$g_{\infty k}\ddot{x}^\infty + \frac{\partial g_{\infty k}}{\partial x^\beta}\dot{x}^\infty \dot{x}^\beta - \frac{1}{2}\frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta = \frac{g_{\infty k}\dot{x}^\infty \ddot{s}}{\dot{s}} \tag{3.10}$$

by writing

$$\frac{\partial g_{\infty k}}{\partial x^{\beta}}\,\dot{x}^{\infty}\dot{x}^{\beta} = \frac{1}{2}\left(\frac{\partial g_{\infty k}}{\partial x^{\beta}} + \frac{\partial g_{\beta k}}{\partial x^{\infty}}\right)\dot{x}^{\infty}\dot{x}^{\beta} \qquad (3.11)$$

Thus,

$$g_{\infty k}\ddot{x}^{\infty} + [\infty\beta, k]\,\dot{x}^{\infty}\dot{x}^{\beta} = \frac{g_{\infty k}\dot{x}^{\infty}\ddot{s}}{\dot{s}} \qquad (3.12)$$

if the curve length is used as parameter, $\dot{s} = 1, \ddot{s} = 0$, then the equation becomes

$$g_{\infty k}\ddot{x}^{\infty} + [\infty\beta, k]\,\dot{x}^{\infty}\dot{x}^{\beta} = 0 \qquad (3.13)$$

multiplying by $g^{\gamma k}$, then the equation becomes

$$\ddot{x}^{\gamma} + [\infty\beta, r]\,\dot{x}^{\infty}\dot{x}^{\beta} = 0 \qquad (3.14)$$

These are the desired equations of geodesics. In eq. (3.14), dots denote the differentiation with respect to the length parameter of the curve *S*. According to [41], the geodesic distance function given by eq. (3.14) is rotation and translation invariant.

3.2 Extracting the skeleton

According to [7], the following definitions are used to formally define the skeleton in the proposed method:

Definition 3. The *degree* of a point is defined as a finite number of points in its 26-neighborhood (figure 3.1 shows the 26-neighborhood structure).

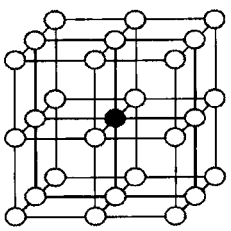


Figure 3.1 The 26-neighborhood structure.

Definition 4. By the assumption that the skeleton is one-voxel thick, the skeleton *end point* is a point that has a degree one. The *middle point* is a point that has a degree two. The *connection point* is a point that has a degree three or higher and all the other neighbors are either the end points or the middle points. The *branched connection point* is a point that has a degree three or higher and at least one of its neighbors is neither an end point nor a middle point. The 26-connected branched connection points form the *branched region*.

Figure 3.2 shows the steps of pruning the skeleton, the original 3D object. The method involves of nine different steps:

(1) Compute the degree of each point.

(2) Determine which point are the end points, the middle points, the connection points, and the branched connection points. If there are no branched connection points, exit the program.

(3) Organize the branched connection points into the branched regions. Each branched region consists of 26-adjacent branched connection points.

(4) In each branched region, find all the end points and the middle points that are 26-adjacent to this branched region, and mark each as "terminator".

(5) Determine the centroids, based on Blum's definition of a skeleton; the skeleton points must be the centers of the maximal disks contained in the curve *C*.

(6) Compute the geodesic distance from the particular centroids to all points in step 4.

(7) Apply the Dijkstra's algorithm, in order to find the shortest paths between the centroids and their "terminator".

(8) Remove the branched connection points that are not on any of the shortest paths.

(9) Generate the pruned skeleton from the remaining points.

```
Input: A set of points on the surface of model, P = {p1, p2, ..., pn}.
Output: The pruned skeleton of the given model.
// Read P and determine which p is an endpoint, a middle point, a connection point, and a branched connection point.
for all p ∈ surface
    compute degree of p
        if degree of p = 1
            p ← endpoint
                if degree of p = 2
                    p ← middle point
```

```
                    if degree of p ≥ 3 and all the other neighbors are either the end points or the middle points
                        p ← connection point
                            if degree of p ≥ 3 and at least one of its neighbors is neither an end point nor a middle
                            point
                                p ← branched connection point
                                    store p in ListOfBranchedConnectionPoint
                            end if
                    end if
                end if
            end if
        end for
        // Construct a branched region from p in ListOfBranchedConnectionPoint.
        for all p ∈ ListOfBranchedConnectionPoint
            if NumberOfNeighbours(p) = 26
                construct BranchedRegion(p)
            end if
        end for
        // Find the end points and the middle points that are 26-adjacent to this branched region, and mark each as
        "terminator".
        for all p ∈ BranchedRegion
            if degree of p = 1
                p ← endpoint
                p ← terminator
                    if degree of p = 2
                        p ← middle point
                        p ← terminator
                    end if
            end if
        end for
        // Determine the centroids in each branched region.
        // Compute the geodesic distance and find the shortest path correspond to each centroid.
        for all p ∈ BranchedRegion
            CentroidOfBranchedRegion = (∑_{i=1}^{N} Xi/N, ∑_{i=1}^{N} Yi/N, ∑_{i=1}^{N} Zi/N)
                if p ≈ CentroidOfBranchedRegion
                    p ← centroid
                        geod_dist (p, p_i)      // compute the geodesic distance from centroid to all p_i, where p_i = terminator
                        Dijkstra's_shortest_path (p, p_i)  // apply the Dijkstra's algorithm
                                                           // to find the shortest path from centroid to all p_i
                        construct ShortestPath(P) where P = (p, p_i)
                end if
        end for
        // Remove the branched connection points that are not on any of the shortest paths.
        // Construct the pruned skeleton.
        for all p ∈ BranchedRegion
            for all p ∈ ListOfBranchedConnectionPoint
                if p ∉ ShortestPath
                    remove p
                    update ListOfBranchedConnectionPoint
                    construct PrunedSkeleton(p)     // construct the pruned skeleton from the remaining points in the list
                end if
            end for
        end for
```

Figure 3.2 The algorithm of the skeleton pruning method.

3.3 Adjusting the skeleton

Although the skeleton pruning method described in figure 3.2 can eliminate a large number of spurious skeleton joints, the next problem is raised that is the meaningless points which align unreasonably along a skeleton. These points can affect on the later creation of segments and joints, therefore the pruned skeleton is subjected to a smoothing operation. The smoothing process involves calculating the average position of consecutive points along the pruned skeleton, reassigning the new joints by splitting a skeleton into the segments [9].

Once, the pruned skeleton has been originated with a one-voxel thick, in a graph-like form as shown in figure 3.3(a), it is ready to be used as a skeleton. Before applying the chord-to-point distance accumulation to the pruned skeleton, the skeleton is needed to be smoothened in a curve form. This can be accomplished by using the average filtering over a sliding window consisting of five consecutive points to adjust the positions of consecutive points along the skeleton.

Let $C = \{p_i = (x_i, y_i, z_i), i = 1,\ldots, N\}$ be the set of coordinate of points on the skeleton. The ith point in the set is denoted by p_i and p_{i+1} is its neighboring point. To calculate the average at the point p_i, the method makes an approximation of the change in the positions of two consecutive points before p_i and two consecutive points after p_i then divide them by 5 (the number of points between $p_{i-2}, \ldots, p_i, \ldots, p_{i+2}$). Thus, the average position (v_i) over the point p_i can be defined as equation (3.15). The filtering process is illustrated in figure 3.3.

$$v_i = \frac{\sum_{i-2}^{i+2} p_i}{5} \qquad (3.15)$$

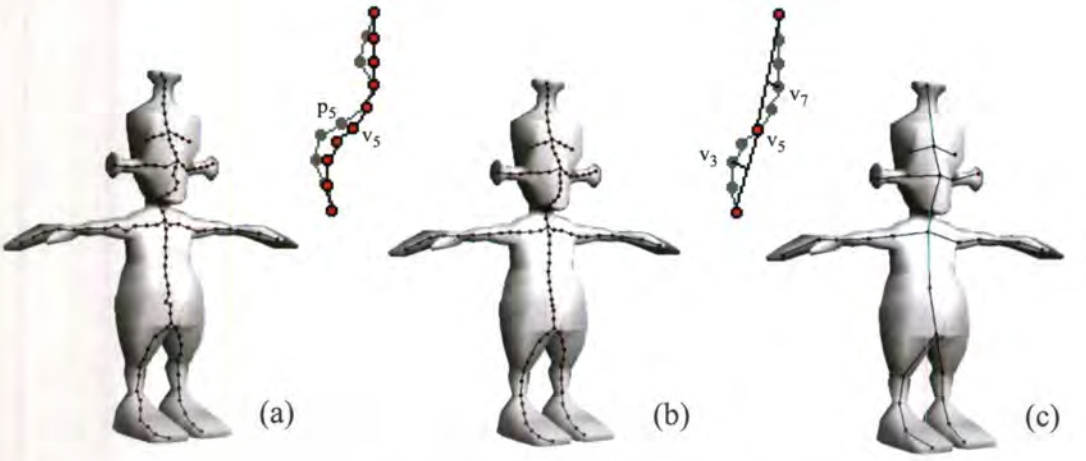


Figure 3.3 (a) Finding the average position over the point. (b) Smoothing the skeleton. (c) The result skeleton.

The method for smoothing the pruned skeleton is shown in figure 3.3. Figure 3.3 (a) shows the average position v_5 over $p_5 = (p_3+p_4+p_5+p_6+p_7)/5$, in order to obtain the smoothed skeleton, the position v_i is computed for each p_i on the pruned skeleton. Figure 3.3 (b) shows the result of applying the smoothing operation to the pruned skeleton from figure 3.3 (a), once the smoothed skeleton is formed, chord-to-point distance accumulation is applied for splitting the smoothed skeleton into segments in order to form bones and joints. Notice that the best split is formed at v_5 since the value of chord-to-point distance accumulation at v_3 and v_7 are equal. The new joints are formed as a result of the best splitting based on chord-to-point distance accumulation value, as shown in figure 3.3 (c).

After the smoothed skeleton is formed, the next step is to determine which points should be split. Splitting of the skeleton is accomplished by considering chord-to-point distance accumulation value. The study applied chord-to-point distance accumulation proposed by Han and Poston [13], unlike Han and Poston's method, this study used the geodesic distance instead of the Euclidean distance.

The study took the benefit of the geodesic distance function which is the invariant to rotation and translation [12] as the distance feature for the smoothing method. Han and Poston defined chord-to-point distance accumulation as the distance measurement from the line to a point in the curve segment. The interpretation of chord-to-point distance accumulation can be derived as follows.

Let L be a fixed integer value defines a line L_i from each point p_i to p_{i+L}, where $i+L$ is taken modulo N. The perpendicular distance D_{ik} is computed from L_i to the point p_k. The distance is positive if p_k is on the left-hand side of the vector $(p_{i+L} - p_i)$, negative otherwise. Chord-to-point distance accumulation for a point p_k and a chord length L is the sum h_L of the D_{ik} as i moves from k-L to k. That is,

$$h_L(k) = \sum_{i=k-L}^{k} D_{ik} \tag{3.16}$$

Since $D_{kk} = 0$, Thus,

$$\begin{aligned} h_L(k) &= \sum_{i=k-L}^{k} D_{ik} \\ &= \sum_{i=k-L+1}^{k-1} D_{ik} \end{aligned} \tag{3.17}$$

Figure 3.4 illustrates chord-to-point distance accumulation, and shows an example in the case of D_{ij} is positive and D_{ik} is negative.

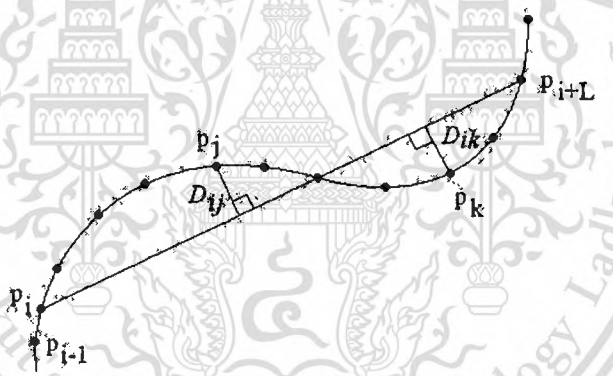


Figure 3.4 Chord-to-point distance accumulation.

The basic idea of locating the new joints depends on chord-to-point distance accumulation value. After the smoothed skeleton is formed, the points that align along the smoothed skeleton can be classified into three classes. The end point is a point that has only one adjacent neighbor. The junction point is a point that has three or more adjacent neighbors. The intermediate point is a point that has exactly two adjacent neighbors. The end points and the junction points split the smoothed skeleton into a set of connected segments or bones.

The study used the end points and the junction points to form the initial joints of the skeleton. In each segment of the skeleton (see figure 3.3 (b)), this work iteratively selects the point from v_{i+1} to v_{N-1} and computes chord-to-point distance accumulation correspond to each selected point. The selected point becomes the splitting point if $||D_{ij}|| = ||D_{ik}||$ e.g. v_5 is the splitting point since chord-to-point distance accumulation value at v_3 equals to chord-to-point distance accumulation value at v_7. Then two bones can be formed from the corresponding segment, the first bone whose joints are located on v_1 and v_5, another bone whose joints are located on v_5 and v_{10} respectively. In the case of more than one splitting point is created, the sub segment will be formed (this sub segment becomes a bone later on), and more joints may be created from the corresponding sub segment. Figure 3.3 (c) shows the final result of the smoothed skeleton generated from the method. The algorithm for smoothing the skeleton is also shown.

```
//Given skeleton V consists of N points places in a queue Q.
Smoothen(V, N)
    If N ≤ 0
        Return V
    Repeat
        Q ← V
        N ← 1
        Repeat while Q ≠ ϕ
            v ← POP(Q)
                Compute h_L(k)
                    If ||D_ij|| = ||D_ik||
                        V ← V-v
                        N ← N + 1
Return v
```

Figure 3.5 The algorithm of the iterative smoothing.

As shown in the algorithm, to compute chord-to-point distance accumulation value for all chordal points in the skeleton curve is not practical. The solution of this drawback is using the appropriate threshold for eliminating this computation time. This study used the simplest threshold based on the mean curvature [42]. Computing the mean curvature of the surfaces is troublesome. Fortunately, the mean curvature is the divergence of the vector field of the optimal normals, and the divergence can be approximated by finite differences. Figure 3.6 shows the vector field of the surface and its normal.

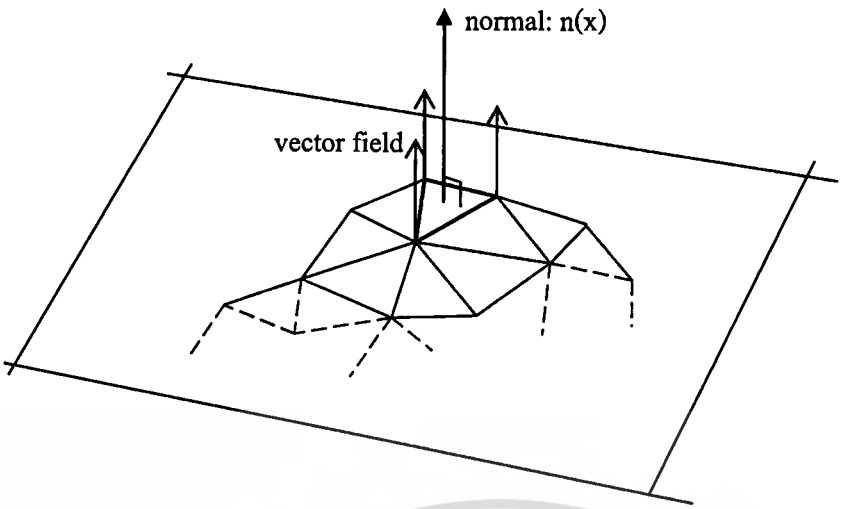


Figure 3.6 A vector field on a surface and its normal.

The mean curvature and its normal are denoted by $H(x)$ and $n(x)$, respectively. The finite difference is the discrete analogue of the derivative (an infinitesimal change in the function with respect to whatever parameter it may have). The divergence computed by finite differences is:

$$H(x) = \nabla . n(x) \tag{3.18}$$

where in 3 dimensions, the divergence is, in Cartesian form:

$$\nabla . n(x) = \frac{\partial n_x}{\partial x}, \frac{\partial n_y}{\partial y}, \frac{\partial n_z}{\partial z} \tag{3.19}$$

Each of the partial derivatives may then be approximated by finite differences in the usual way. According to [42], the easiest and fastest way to calculate them is to use the intermediate difference operator. With this operator, for each axis the method looks at the neighbor and the optimal normal of the current voxel, thus a total of two voxels can be computed. So, for the x-axis the operator is:

$$\frac{\partial n_x}{\partial x} = \frac{n_x(x + \Delta x) - n_x(x)}{\Delta x} \tag{3.20}$$

and can be extended in the y and z axes as follows.

$$\frac{\partial n_y}{\partial y} = \frac{n_y(y + \Delta y) - n_y(y)}{\Delta y} \tag{3.21}$$

$$\frac{\partial n_z}{\partial z} = \frac{n_z(z + \Delta z) - n_z(z)}{\Delta z} \tag{3.22}$$

According to [42], the appropriate values of the mean curvature, $H(x)$ are 0.4-0.6, the study used these values as the threshold where the chordal points (joints on the skeleton) will be selected if their absolute mean curvature values are greater than the threshold:

$$\nabla . n(x) > |H(x)|; \text{ where } |H(x)| \in [0.4,0.6] \tag{3.23}$$

In the next chapter, the evaluation of the proposed algorithms will be presented, and compared with the well-known skeletonization algorithm. In the rest of this work, the study will use the smoothed 3D skeleton in a character animation, the analysis and design, together with implementations will be shown in chapter 5.

Chapter 4

Testing and the running times

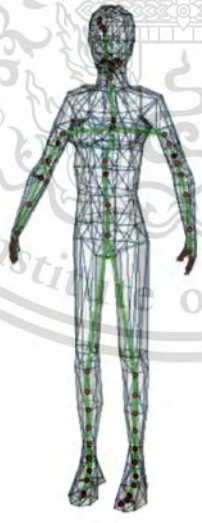
This chapter will provide the evaluation results by comparing the results of the proposed algorithms (pruning and smoothing algorithms) with [3]. The proposed algorithms and [3] were tested on the Princeton Shape Benchmark database [43], and a standard handmade model (from Autodesk's Maya software). A brief comparison between the results of the proposed methods and their original skeleton, extracted by using the Multi-Resolution Reeb Graph (MRG), proposed by [3] is provided.

The pruning method can reduce the number of the branched connection points, which is similar to the number of critical points in the original Dijkstra's algorithm therefore the spurious skeleton joints can be removed. These joints will not influence the structure of the skeleton because they are not on any of the shortest paths computed by Dijkstra's algorithm. The pruned skeleton can then be generated in the sense that it can capture the essential shape of a 3D object while preserving a compact form of its data structure. Table 4.1 shows the experimental results.

Table 4.1 compares the number of joints after applying the pruning algorithm with the number of joints extracted by [3]. The pruning method can reduce the number of critical points in the graph's nodes, the branched connection points in a skeleton graph. Therefore, the numbers of the spurious skeleton joints are reduced comparable to the number of unnecessary joints generated by the MRG method.

The study also demonstrated the smoothing algorithm on several pruned skeletons generated from pruning algorithm (see figure 3.2), as shown in Table 4.1. A brief comparison between the results of the smoothing algorithm and their pruned skeletons is provided. The smoothing algorithm can produce a useful control skeleton which will be used for animating purpose in the next chapter, the number of joints can be significant reduced in the sense that it is still enough for use in animation. Most of the control skeletons produced from the proposed algorithm are centralized, and run along to the ends of the main branches of the models.

Table 4.1 Comparison of the number of joints between MRG (No pruning), the pruning method, and the smoothing method.

Mesh	Number of joints			Threshold used (in the smoothing method)	% of the reduced joints (Pruning Vs. Smoothing)
	MRG [3]	The pruning algorithm	The smoothing algorithm		
Boy: No. of faces: 32007 No. of vertexes: 16017 	17342 	860 	19 	0.55	97.79 %
Femme: No. of faces: 1002 No. of vertexes: 503 	635 	50 	26 	0.45	48 %

Mesh	Number of joints			Threshold used (in the smoothing method)	% of the reduced joints (Pruning Vs. Smoothing)
	MRG [3]	The pruning algorithm	The smoothing algorithm		
Alien: No. of faces: 1436 No. of vertexes: 732	429	40	30	0.45	25 %
Horse: No. of faces: 1058 No. of vertexes: 535	783	56	27	0.45	51.78 %
Dinopet No. of faces: 3999 No. of vertexes: 2039	1219	120	31	0.55	74.16 %

Figure 4.1 shows several objects of a human-like model smoothed by the proposed algorithm. Segments and joints relate fairly well to surface features; however there is room for improvement. For instance for the boy model, there is no control joints for fingers or toes are produced.

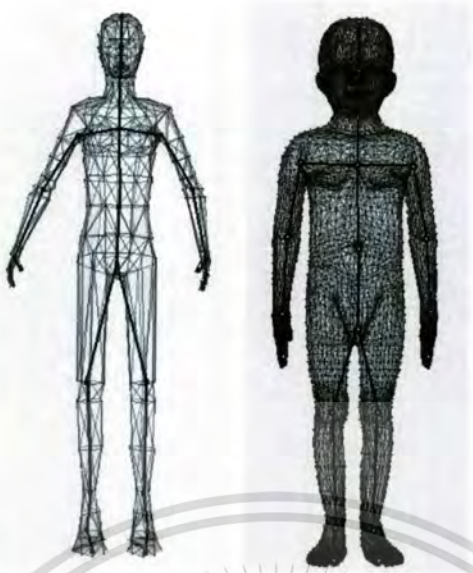


Figure 4.1. The control skeletons extracted by geodesic-based skeleton smoothing.

The computational complexity of the approach can be derived here. Let n be the number of points in the meshed model. The time complexity of the proposed algorithm can be considered as follows.

The pruning algorithm: the computation of degree of each point and the point analysis takes $O(n)$ time. The computation of testing and finding the maximal disks in the neighborhood of each point takes $O(n)$ time. The computation of the single source shortest paths (in term of the geodesic distance from that source vertex) to all points on the mesh takes $O(n \log n)$ time [44]. Thus, the overall computation takes $O(n \log n)$ time.

The smoothing algorithm: the computational complexity of averaging position of v_i over the point p_i in the filtering process takes $O(n)$ steps. The computational complexity of chord-to-point distance accumulation takes $O(n)$ steps. The implementation of the algorithm is done by placing n points in a queue data structure, and most of the times are spent on the *POP* operation, thus the time complexity of this process is $O(\log n)$ steps. Thus, the overall computational complexity of the smoothing algorithm takes $O(n)$ steps (this is not include the complexity analysis of the skeleton pruning algorithm which takes $O(n \log n)$ steps).

In the next chapter, the study will implement the result skeletons to produce the realistic 3D character animations.

Chapter 5

Design and Implementation

This chapter describes the introductory phases of the software engineering process, in which the requirements for the software are established and specified in detail for further development. The chapter starts with the requirements definition, which is an abstract description of the services the software should provide and the constraints under which it must operate. The chapter concludes with the design of the software, followed by a brief discussion of why the particular design was chosen over competing methodologies, and any problems this study had to overcome.

5.1 Requirements definition and specification

This section will start with the general description of the software, followed by the definition of the requirements. These requirements have been divided into functional, which are just statements of the system services, and how the system should react to particular inputs, and to nonfunctional, which are constraints imposed on the functionality of the system and the development process.

5.1.1 Functional requirements

The system has the ability of loading Mesh formats like OFF and objects files from the user and also creates a new text based formats. It also stores the OFF file in computer memory in such a way that it can be processed later by other parts of the system. The system also has the ability of saving the deformed mesh and creating a unique format for handling key frames based animation with full options to control and edit animation. The system also provides the enhanced features such as adding frames, removing frames, loading animation, and recording to AVI video file with any codec installed in the system.

The abilities of reading files formats of the system are shown below:

1. Reading OFF format option

The OFF format (.off) includes the data as follows:

The name of the format, the numbers of vertex, faces and texture.

```
OFF 8145 6564 24088
0.49167 1.53549 0.28482
0.52069 1.47633 0.26981
0.50562 1.47848 0.28101
0.48597 1.52223 0.29346
0.54712 1.39891 0.27045
0.53161 1.40082 0.28174
```

Figure 5.1 The OFF file format.

2. Text format option: After dealing with the reading of the OFF format, the study has started to create two different text format for the skeleton and the weights, which were in a text format. An example of the skeleton file is shown below.

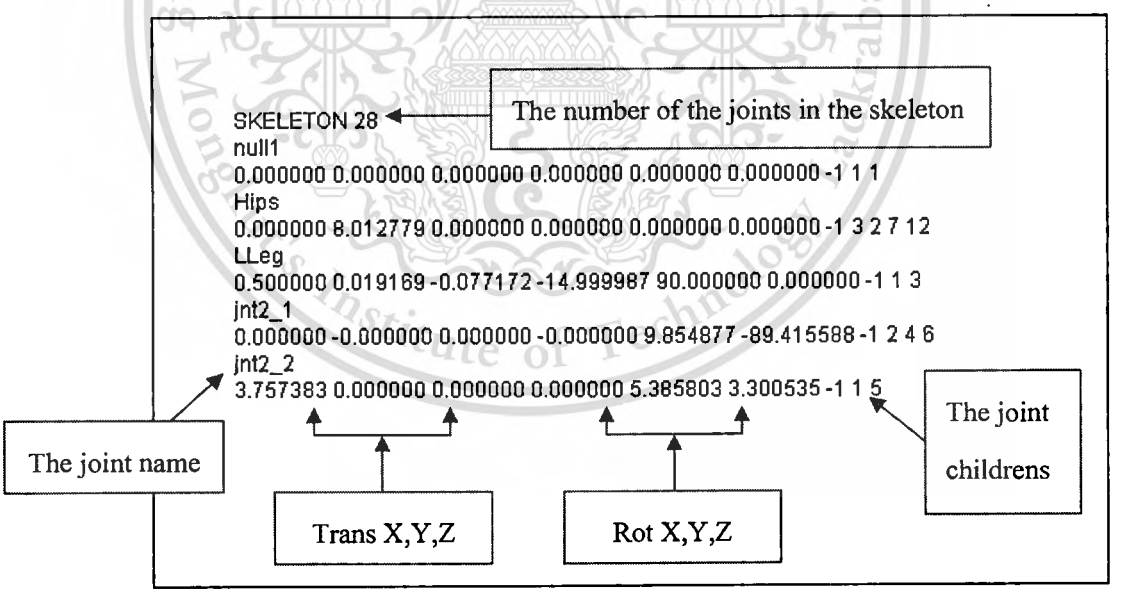


Figure 5.2 The skeleton text format.

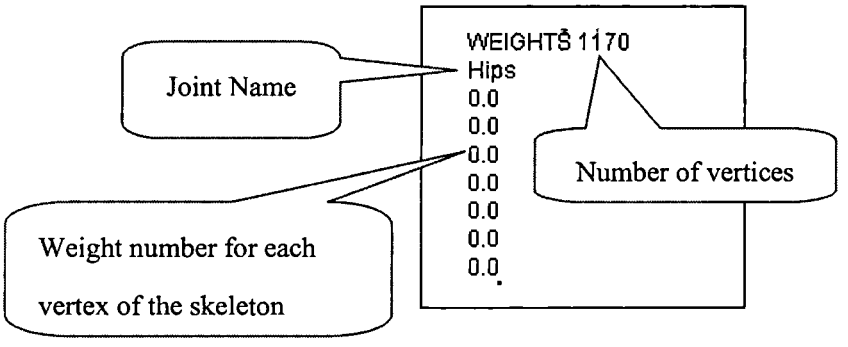


Figure 5.3 The weight text format.

5.1.2 Nonfunctional requirements

1. Usability: Because of the system requires no user interaction and supervision, apart from the initial data and parameter input, users should be able to use all the system functions with minimal training.

2. Portability: The system should be implemented in a language where it would be possible to compile it for different platforms. The obvious choice would be STD C++, and if possible ISO C++.

3. Reliability: The system can be deal with the 3D object file with medium to high level of detail. It also saves a full package of animation, skeleton, mesh, weights linked to a project file and exports the animation to AVI movie file format without failure.

4. Robustness: There must be zero probability that the original data might be corrupted and low probability of corruption of the produced data, after failure.

5. Speed and cost: The design and implementation of the system (together with choice of language) should not considerably increase the estimated execution time of the algorithm. The size of the executable program is ~272 Kbytes.

5.2 System design

At the beginning of the design process, the study chose a procedural design for the software. However, it was obvious in that stage, that many parts of the system required input and functionality from other parts, and this proved to be quite complicated for a procedural design to handle. Furthermore, as the scale of the design increased, and many portions of the system were constantly reused, therefore it would has been better to use an object-oriented methodology to capture this complexity, the communication needs and code reuse requirements. In addition to that, this study is largely based on computational geometry, which itself is built from simple components such as points, lines, angles, and so on, which are combined in order to build a more complex structure. Object-oriented design can capture this behavior easily because of its special characteristics, such as inheritance, aggregation, data abstraction and so on. The class diagram of the system is shown below.

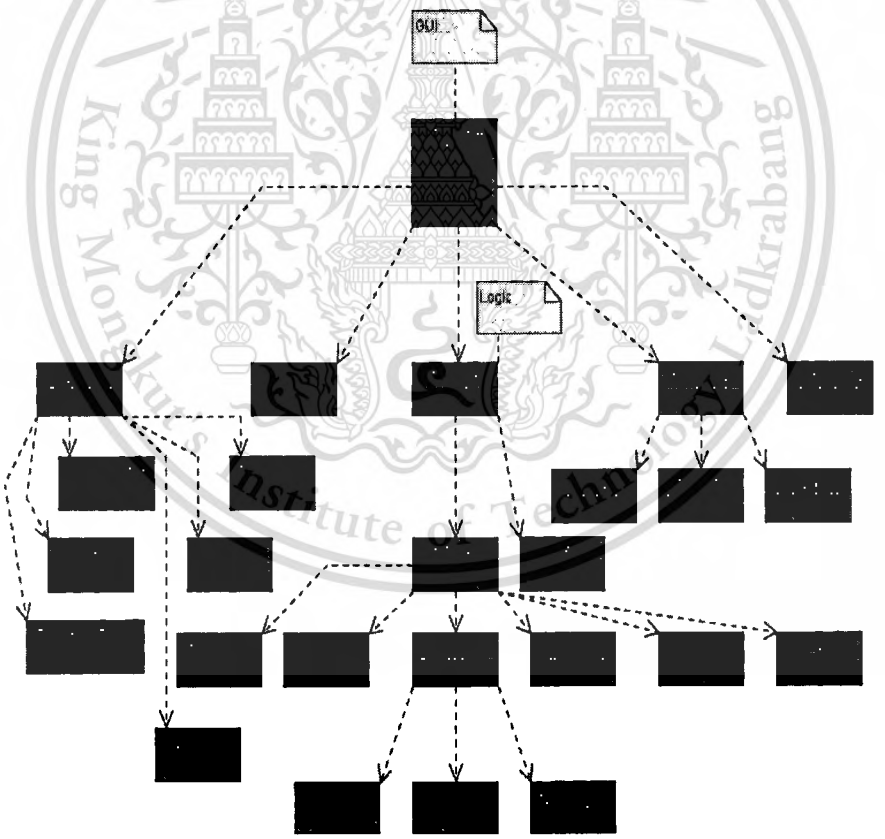


Figure 5.4 Class diagram.

5.3 Animating the skeleton

5.3.1 The key frames idea

Once a frame has been created, the system can animate the image using the key frame method of animation. This method uses two frames which depict the start and end frames, then creates an animation sequence by adding frames (a series of views) between the two existing frames (key frames). The following steps illustrate how to create the frames between the two key frames:

1. Calculating the frame gap between two key frames.
2. Finding the relative index between the frame to set to, and the current key frame.
3. Calculating the motion percent between the relative index divided to the frame gap.
4. Assigning each key frame multiply by the motion percent into the translation, rotation, and scaling.

Then automatically adds the required number of frames between these key frames to produce an animated smooth sequence.

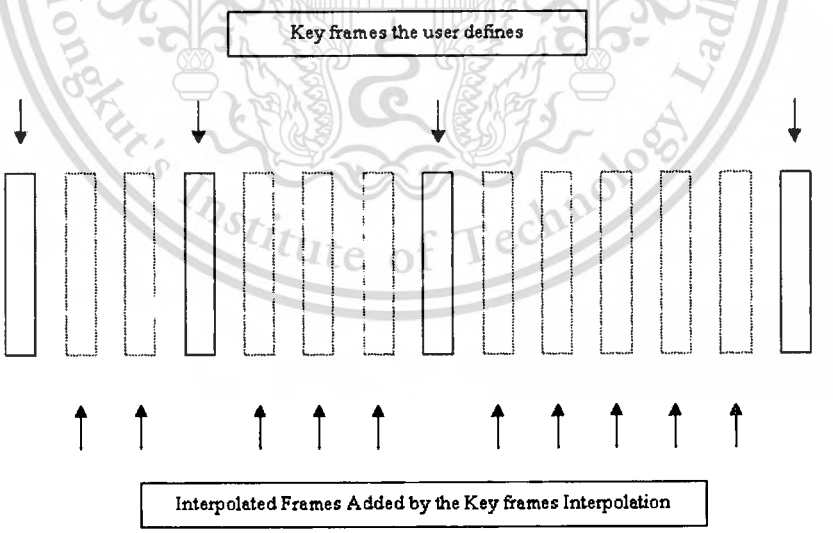


Figure 5.5 The Key Frames with the interpolation frames.

5.3.2 GUI General description

The system consists of 5 panels; the layout of the system is shown below

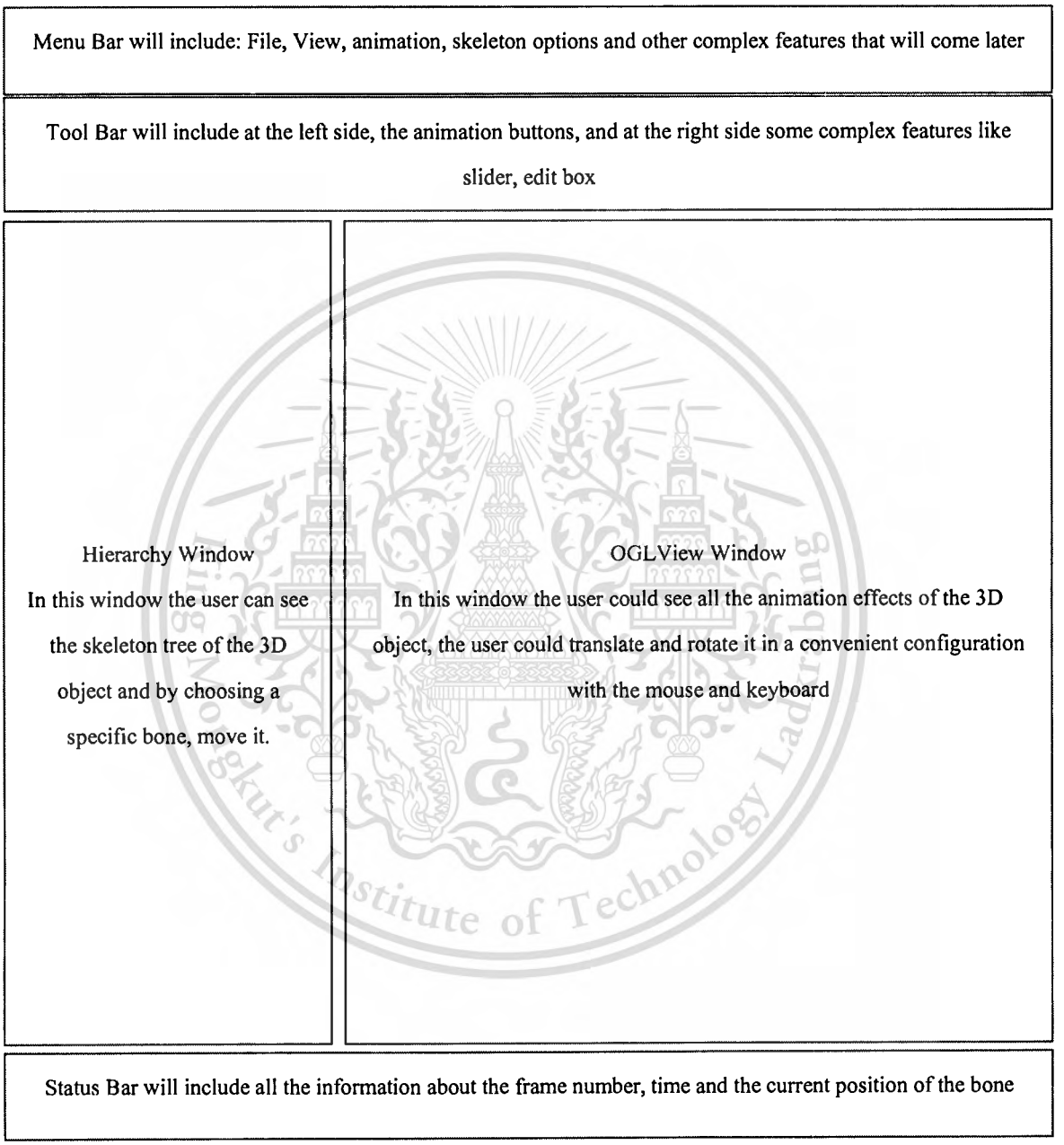


Figure 5.5 The GUI of the system.

5.3.3 The animation file format

The animation file format includes the key frames number on the left side of the frame, and the name of the format is Animate3D2. The animation format with some explains is shown in figure 5.6.

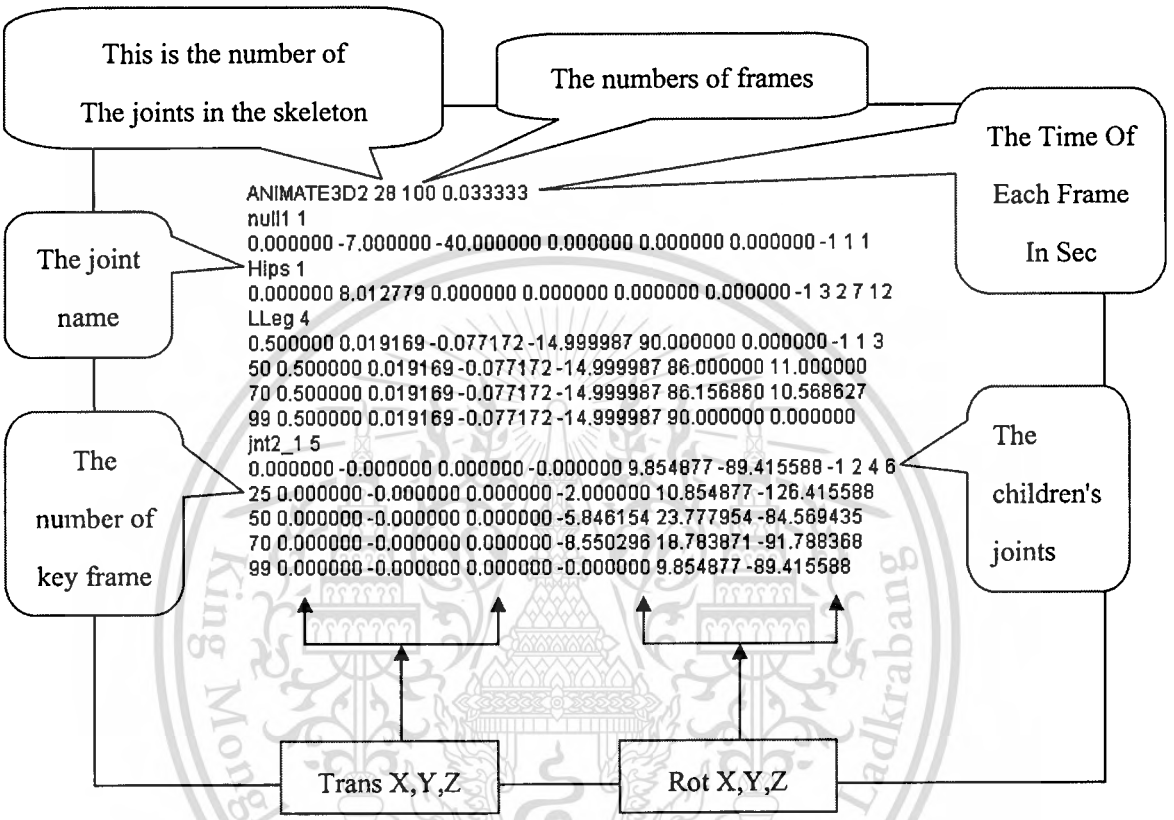


Figure 5.6 The animation file format.

5.4 Implementation and results

This section described the implementation of the result skeletons extracted from the proposed method (referred to chapter 4) to generate the 3D character animation. The GUI of the animation software is shown in figure 5.7. The implementation of the animation is fulfilling both functional requirements and non-functional requirements. The final software provides the abilities such as loading the OFF files, saving deformed objects in two formats (.obj) and (.off), loading skeleton and weights files in

text format, creating of two animation formats (.a3d) which are the framed-based animation and key frame-based animation. The software also provides the capability of creating, loading and saving animation with the GUI panel for controlling and editing the animation easily.

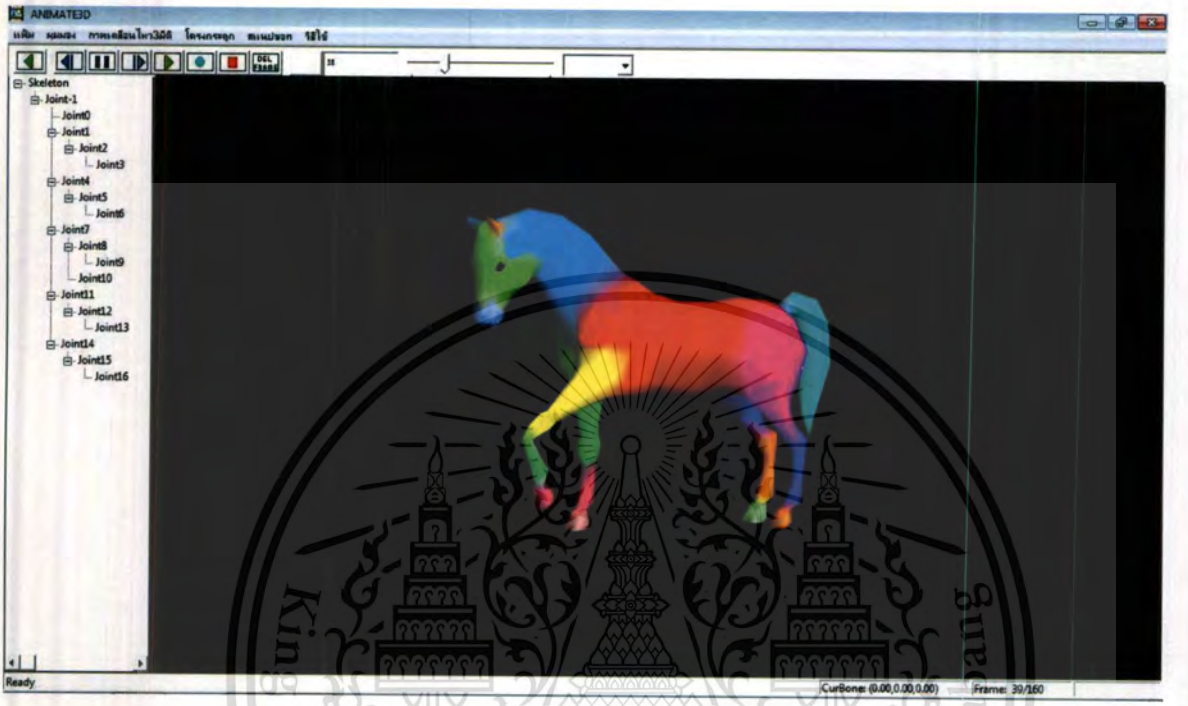


Figure 5.7 The snap shot of the animation.

In the chapter 6, the thesis will summarize the research and presents the overall conclusions. The further research is also discussed.

Chapter 6

Conclusion

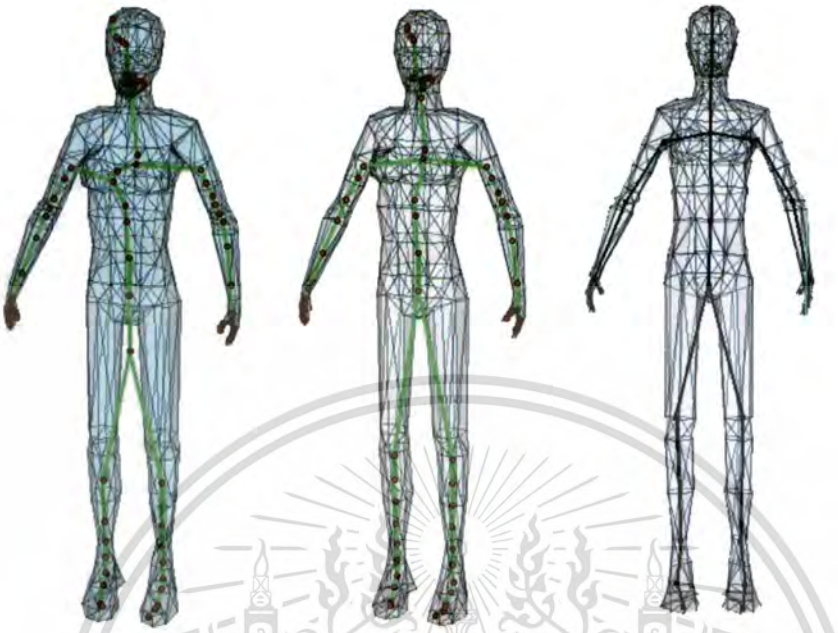
This is the final chapter of the thesis and it contains the summary of the study so far and the overall conclusions. In addition, it also provides the discussion of the certain ideas about possible future works on how to improve the proposed algorithms, and possible application areas and extensions.

6.1 Summary

This thesis presented a new method for generating a skeleton, based on the medial axis transform and the geodesic distance function associated with a 3D meshed object. The study also presented the smoothing technique in order to relocate the new joints for the later use in the animation production. The skeleton, generated by the new method captures the overall shape as well as the topology of an object. The proposed method can generate a one-voxel thick skeleton, in a graph-like form, which maintains the robustness against any changes of a rotation and/or a translation of the 3D meshed model, which are the major properties of 3D objects. This work gains this benefit from the geodesic distance used in chord-to-point distance accumulation to construct the rotation and the translation invariant control skeleton, which is quite important when the model is subject to be deformed during the animation production, since the bones and joints must be kept inside the model all the time while the animator deforms the character model.

In figure 6.1 (a), it is easy to observe that the skeleton extracted by using the MRG is very sensitive to the connectivity of the boundary representation and it also has a problem in producing the unwanted skeleton joints. The joints are also generated outside the model which is the major drawback of the MRG skeleton. The pruning algorithm can overcome these drawbacks; the results are shown in figure 6.1 (b) and 6.1 (c). In the smoothing step, the collection of joints produced from

the smoothing algorithm is transformed into a collection of bones which can be used for 3D animation.



(a) (b) (c)

Figure 6.1 The skeleton extracted by using the MRG (a), the pruned skeleton (b), and (c) the result of the smoothing method.

Another example is shown in figure 6.2. Even though the MRG method can extract the skeleton in a curve-like form suitable for use as the topology description, it is not appropriate in animation manner since the curve contains many points consecutively which is not practical to use as the control segments in a character animation. This study can overcome this drawback by producing the useful control skeletons with a small number of control segments (joints). It also suits for producing skeletons for more complex objects (the object which have a large number of vertex, and faces), figure 6.2 shows an example of the model with the number of faces are 32007, since it can reduce the number of control segments comparable to the number of unnecessary segments generated by the skeleton pruning method. Figure 6.2 (c) shows the meaningful characteristics of the 3D meshed model, and the generated skeletons whose joints can be associated with the 3D meshed model.

Figure 6.3 Moving horse. These images are taken from an animation created using the skeletoniztion algorithm proposed by this study.

6.2 Future works

Given the skeleton construction process with the proposed approach of geodesic-based skeleton smoothing, significant improvements are desirable, either by decreasing the time complexity of the algorithm. Another area of applications using the skeleton extraction is still in the room for improvement such as the similarity matching of the 3D objects, the knowledge-based approach for skeleton extraction, and so on.

References

[1] F. Lazarus, and A. Verroust. *"Level set diagrams of polyhedral objects"*. In Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications, ACM Press, pages 130–140, 1999.

[2] G. Aujay, F. Hetroy, F. Lazarus, and C. Depraz. *"Harmonic skeleton for realistic character animation"*. In Symposium on Computer Animation (ACM SIGGRAPH), 2007.

[3] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. *"Topology matching for fully automatic similarity estimation of 3D shapes"*. Computer Graphics Annual Conference Series (Proc. Of SIGGRAPH), pages 203–212, 2001.

[4] T. Oda, Y. Itoh, W. Nakai, K. Nomura, Y. Kitamura, and F. Kishino. *"Interactive skeleton extraction using geodesic distance"*. In Proceeding of the 16th International Conference on Artificial Reality and Telexistence, 2006.

[5] G. Reeb. *"Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [On the singular points of a complete integral pfaff form or of a numerical function]"*. Comptes Rendus Acad.Science Paris 222, pages 847–849, 1946.

[6] M. Sabry Hassouna and Aly A. Farag. *"On the extraction of curve skeletons using gradient vector flow"*. Proc. of IEEE International Conference on Computer Vision ICCV, Rio de Janeiro, Brazil, October 14-20, 2007.

[7] P. Visutsak and K. Prachumrak. *"The 3D skeleton pruning for removing undesired joints"*. In Proceedings of The Second IEEE International Conference on Computer, Control and Communication IEEE-IC4 2009, Karachi, Pakistan, February 17-18, 2009.

[8] L. Wade and R.E. Parent. *"Automated generation of control skeletons for use in animation"*. The Visual Computer 18, 2002.

[9] P. Visutsak and K. Prachumrak. *"The smoothed 3D skeleton for animation"*. In Proceedings of the 5th International Conference on INC, IDC, IMS, Seoul, Korea, August 25-27, 2009.

[10] P. Visutsak and K. Prachumrak. *"Geodesic-Based Skeleton Extraction"*. The 25th International Tech nical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2010), Pattaya, Thailand, July 4-7, 2010.

[11] P. Visutsak and K. Prachumrak. *"Geodesic-based Skeleton Smoothing"*. International Journal of Mathematical Models and Methods in Applied Sciences, pages 713-721, Issue 4 Vol.5, 2011.

[12] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. *"The discrete geodesic problem"*. SIAM J. of Computing 16(4), pages 647-668, 1987.

[13] J.H. Han and T. Poston, *"Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature"*. Pattern Recognition Letters, vol. 22, pp. 1133-1144, 2001.

[14] M. K. Agoston. *"Computer graphics and geometric modeling: implementation and algorithms"*. Springer, ISBN 1852338180, page 185, 2005.

[15] H. Blum and R.N. Nagel. *"Shape description using weighted symmetric axis features"*. Pattern Recognition 10, pages 167-180, 1978.

[16] A. Lieutier. *"Any open bounded subset of R^n has the same homotopy type than its medial axis"*, ACM Shape Modeling and Applications, 2003.

[17] W. Gong and G. Bertrand. *"A simple parallel 3D thinning algorithm"*. IEEE Pattern Recognition, 188-190, 1990.

[18] M. Schmitt, "*Some examples of algorithms in computational geometry by means of mathematical morphological techniques*", Lecture Notes in Computer Science, Geometry and Robotics, pp.225-246, 1989.

[19] E. C. Sherbrooke, N. Patrikalakis, and E. Brisson, "*An algorithm for the medial axis transform of 3d polyhedral solids*", IEEE Transactions on Visualization and Computer Graphics, pp.44-61, 1996.

[20] H. Sundar et al, "*Skeleton Based Shape Matching and Retrieval*", Proceedings, Shape Modelling and Applications Conference, SMI , pp.n/a, 2003.

[21] N. Gagvani and D. Silver, "*Parameter controlled skeletons for 3D visualization* ", Proceedings IEEE Symposium on Volume Visualization, pp.n/a, 1998.

[22] N.Amenta, S.Choi and R.Kolluri, "*The Power Crust*", Proceedings of the sixth ACM Symposium on Solid Modelling and Applications, pp.249-260, 2001.

[23] T.Dey and S.Goswami, "*Tight Cocone: A Watertight Surface Reconstructor*", Proceedings of the eighth ACM symposium on Solid modeling and applications, pp.127-134, 2003.

[24] M.Ahmed and R.Ward, "*A Rotation Invariant RuleBased Thinning Algorithm for Character Recognition*", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.1672-1678, 2002.

[25] M. Wan, F. Dachille and A. Kaufman, "*Distance Field Based Skeletons for Virtual Navigation*", Proceedings of the 12th IEEE Visualization Conference, pp.n/a, 2001.

[26] L. Nackman and S. Pizer, "*ThreeDimensional Shape Description Using the Symmetric Axis Transform I:Theory*", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.187-202, 1985.

[27] F. Wolter, "*Cut Locus and Medial Axis in Global Shape Interrogation and Representation*", MIT Sea grant report, pp., 1992.

[28] D.Sheeny et al, "*Shape Description by Medial Axis Construction*", IEEE Transactions on Vizualisation anc Computer Graphics, pp.62-72, 1996.

[29] M.Teichman and S.Teller, "*Assisted Articulation of Closed Polygonal Models*", Proc. 9th Eurographics Workshop on Animation and Simulation, pp., 1998.

[30] S. Wilmarth et al, "*Motion Planning for rigid body using random networks on the medial axis of the free space*", ACM Symp. on Computational Geometry, pp., 1999.

[31] D.Shaked and A. Bruckstein, "*Pruning Medial Axes*", Computer Vision and Image Understanding, pp.156169, 1998.

[32] R. Ogniewicz, "*Automatic Medial Axis Pruning by Mapping Characteristics of Boundaries Evolving Under the Euclidean Geometric Heat Flow onto Voronoi Skeletons*", Harvard Robotics Laboratory Technical Report, pp., 1995.

[33] T.Culver, j.Keyser and D.Manocha, "*Accurate computation of the medial axis of a polyhedron*", Solid Modeling '99, pp.179-190, 1999.

[34] C. Hoffman, "*How to construct the skeleton of CSG objects*", Proc. Fourth IMA Conf., The Mathematics of Surfaces, pp., 1990.

[35] R. Ogniewicz and M.Ilg., "*Voronoi Skeletons: Theory and Applications*", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.63-69, 1992.

[36] M.Golin and H-S Na, "*On the average complexity of 3D Voronoi diagrams of random points on convex polytopes*", Proc. of the 12th Annual Canadian Conference on Computational Geometry, pp.127-135, 2000.

[37] F.Leymarie and M.Levin, "*Simulating the Grassfire Transform Using an Active Contour Model*", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.56-75, 1992.

[38] Y.Zhou and A.Toga, "*Efficient Skeletonization of Volumetric Objects*", IEEE Transactions on Visualization and Computer Graphics, pp.195-206, 1999.

[39] I.Bitter, A.Kaufman and M.Sato, "*Penalized distance volumetric skeleton algorithm*", IEEE Transactions on Visualization and Computer Graphics, pp.195-206, 2001.

[40] M.Wang et al, "*Distance field based skeletons for virtual navigation*", IEEE Proceedings of the conference on Visualization 2001, pp., 2001.

[41] S. Baloch, H. Krim, I. Kogan, and D. Zenkov. "*Rotation invariant topology coding for 2D and 3D objects using Morse theory*". In Proceeding of IEEE International Conference on Image Processing, ICIP 2005, pages 796-799, Sep 2005.

[42] Betelu S., Sapiro G., Tannenbaum A., "*Affine invariant erosion of 3D shapes*", In Proceedings of the 8th International Conference on Computer Vision, pp.174-180, 2001.

[43] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. "*The Princeton shape benchmark*". In Shape Modeling International, pages 167-178, 2004.

[44] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. "*Loops in reeb graphs of 2-manifolds*". In Symposium on Computational Geometry, pages 344-350, 2003.

Appendix

Publications

P. Visutsak and K. Prachumrak. *"Geodesic-based Skeleton Smoothing"*. International Journal of Mathematical Models and Methods in Applied Sciences, pages 713-721, Issue 4 Vol.5, 2011.

P. Visutsak and K. Prachumrak. *"Knowledge-Based Approach for Skeleton Approximation"*. Doctoral Research Workshop, The 7th International Conference Computer Graphics, Imaging and Visualisation (CGiV 2010), Sydney, Australia, August 7-10, 2010.

P. Visutsak and K. Prachumrak. *"Geodesic-Based Skeleton Extraction"*. The 25th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2010), Pattaya, Thailand, July 4-7, 2010.

P. Visutsak, V. Boonjing, and K. Prachumrak. *"Knowledge-Based approach for 3D skeleton extraction"*. In Proceedings of The 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, pages 796-801, Seoul, Korea, November 24-26, 2009.

P. Visutsak and K. Prachumrak. *"The smoothed 3D skeleton for animation"*. In Proceedings of the 5th International Conference on INC, IDC, IMS, Seoul, Korea, August 25-27, 2009.

P. Visutsak and K. Prachumrak. *"The 3D skeleton pruning for removing undesired joints"*. In Proceedings of The Second IEEE International Conference on Computer, Control and Communication IEEE-IC4 2009, Karachi, Pakistan, February 17-18, 2009.

Geodesic-based Skeleton Smoothing

Porawat Visutsak and Korakot Prachumrak

***Abstract*—** Skeleton is at the main interest of 3D character animation. The most common techniques for skeleton computing are based on the Reeb graph and the shortest path finding. Using only the shortest path algorithms for extracting the critical points and constructing the Reeb graph over the surface of the model may generate unwanted skeleton joints. In this paper, we present a new approach to compute the skeleton of the 3D meshed model in a Riemannian space, based on Blum's Medial Axis Transform and geodesic distance algorithm. We gain the benefit of geodesic distance functions and parameterization that allow for efficient handling of topological changes of dynamics curves and surfaces. Thus, our approach can provide the robustness against any changes of a rotation and/or a translation of the 3D mesh model. We are able to generate one-voxel thick, graph-like skeleton. Han and Poston's Chord-to-point Distance Accumulation then be applied for adjusting the locations of consecutive points along the skeleton. The smoothed skeleton is split in order to create segments and joints corresponding to its shape. The new skeleton can be regenerated later on. Therefore, the new skeleton produced from our method can capture the essential shape characteristics in a compact form, while preserving the meaningful anatomical information of the 3D character models. The demonstration of the approach with several examples is also provided.

***Keywords*—** Skeleton; Skeleton smoothing; Geodesic distance; Medial axis transform; Riemannian space; Chord-to-point distance accumulation.

I. Introduction

THE skeleton is a basic structure of representing the 3D objects, frequently used in computer graphics, especially in the areas of character animation and 3D modeling. Using a skeleton as an abstraction of an object has two major benefits. First, it can contain both shape features and topological structures of an original object. Another benefit depends on its characteristic to capture the essential shape of a 3D object in a low-dimension form. Numerous algorithms have been developed to generate the skeleton in graph-like or a curve-like form [7], [8], [10], [11], [12]. Unfortunately, these approaches do not suit for producing a skeleton for use in animation since they need the additional processes to eliminate the redundant skeleton branches that may generate during the skeleton extraction process. The most common technique to represent the 3D objects, that has been the standard for many years, is the Reeb graph [11], originally defined by Reeb. The Reeb graph is obtained by applying the continuous function, usually a height function, to encode the topological structure. Using a height function to build a Reeb graph does not guarantee that the graph is invariant to the affine transformations, which is an essential feature of the skeletal structure of the model [12]. The extended version of the Reeb graph, called the Multi-Resolution Reeb graph (MRG) [7] has been proposed. To construct the MRG, the topological characteristics of the shape must be defined in terms of the critical points of a function on the manifold; this function is called a mapping function. The mapping function maps the points from the manifold of the shape to the domain of the function, and the configuration of the critical points of the mapping function can represent the topology of the shape [14]. This configuration can be embedded by the Reeb graph, and becomes an essential property of the shapes later on. When the mapping function is defined, the model is then partitioned into regions that correspond to equal intervals of the mapping function [15]. Each partition of the model is represented as a node in the Reeb graph, and adjacent nodes are linked by an edge that connects the corresponding nodes.

To apply skeleton for use in character animation and 3D modeling, skeleton animation is a common technique for animating a 3D model. Controlling the movement of a skeleton in a way that is designed to appear naturally is accomplished using a control skeleton (sometimes called an Inverse Kinematics or IK skeleton). IK skeleton is an articulated structure of segments and joints combined with information detailing how the surface geometry of the figure is anchored to that structure. Recent work on semi-automatic skeleton extraction is introduced by Aujay et al. [2]. This system allows users select the starting point on the character model, and then it generates a skeleton to match the ones that are created by hand by professionals in most biped and quadruped cases. A method for fully automatic generation of a control skeleton is proposed by Wade and Parent [18]. The main task of the system involves discretizing the figure, computing its discrete medial surface (DMS), and then using the discrete medial surface both to create the skeleton and to attach the vertices of the model to that structure. However, a major drawback of Wade and Parent's method is only features with a size greater than the voxel size can be taken into account. This often leads to computationally expensive algorithms.

In this paper, we present a novel approach to compute the skeleton of 3D mesh model based on Blum's Medial Axis Transform and geodesic distance algorithm. Blum et al. [4] propose to use the medial axis to define the skeleton. The algorithm for computing the skeleton in terms of the medial axis is often refers to as the "grassfire" algorithm. The main idea of the "grassfire" algorithm is lighting a fire, started from the border of the object, and let it burn into the object at a

constant speed; it will then meet in the medial axis. One starts on the border of the object and strips away one layer of pixels after another until one reaches points that fire reaches from two directions [1]. The medial axis transform of the region is the set of points reached by more than one fire at the same time. Unfortunately, the medial axis transform does not provide robustness against a rotation and a translation of the objects. Therefore, a more sophisticated algorithm is needed in order to solve this problem, and that algorithm is the geodesic distance functions. By using the geodesic approximation algorithm proposed by [9] to compute the "single source, all destination" shortest path on a surface of the model, any changes of a rotation and/or a translation do not affect the value of the function. Thus, this becomes the major property of the function of geodesic distance that gives the advantage of being invariant to a rotation and a translation. The basic idea of our approach is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region determination. Applying the function of geodesic distance can guarantee that the new approach is invariant to a rotation and a translation, and it is also robust against the changes in the connectivity on the 3D shapes. Unfortunately, this raises the problem of producing meaningless joints since the location of skeleton joints does not match the real bone structure of the model. Thus, the additional operation is needed, by using the filtering process we can obtain the smoothed skeleton. Chord-to-point distance accumulation [6] then be applied in order to split the smoothed skeleton into the segments, and then the new joints are located correspond to chord-to-point distance accumulation values. A brief review of the medial axis transform and the geodesic distance function are illustrated in sections 2. The skeleton smoothing method based on Chord-to-point is outlined in section 3; we also demonstrate the usability of the smoothed 3D skeleton with several figures, and the experimental results are shown in section 4. Finally, the conclusion and the evaluation of our approach are also provided.

II. Geodesic-based Skeleton Smoothing

A. The Medial Axis Transform

The idea of using a skeleton as an abstraction of the shape goes back to [4]. Blum et al. define a skeleton in terms of the medial axis (MA), which is a set of curves that roughly run along the middle of an object, as shown in figure 1. According to [4], the medial axis of a curve S is the locus of the centers of the maximal disks contained in S. Let R^n be a symmetry set (where n is a number of dimensional space) which is defined similarly to the medial axis, except that it also includes the circles not contained in S and thus the medial axis is a subset of the symmetry set. The medial axis can then be defined as follows:

Definition 1. Let S be an arbitrary curved surface; Let $D^n(p, r)$ be a closed disk with a radius r centered at a point p, where $S(p, r) \subseteq R^n$. A maximal disk in S is a closed disk $D^n(p, r)$ contained in S.

Property 1. If X is a maximal disk in S, then S is not properly contained in any other closed disk in S.

Definition 2. Let $S(p, r) \subseteq R^n$. The medial axis (MA) of S is the locus of the centers of the maximal disks contained in S. The medial axis of a 3D object denoted R^3 is sometimes called the medial surface. The continuous function of a real-value that assigns to each center of a maximal disk in S is called the radius function of that medial axis.

The medial axis together with the associated radius function of the maximal disks is called the medial axis transform (MAT). Then, the medial axis can be defined:

Definition 3. The medial axis transform (MAT) of an object consists of its medial axis together with the associated radius function.

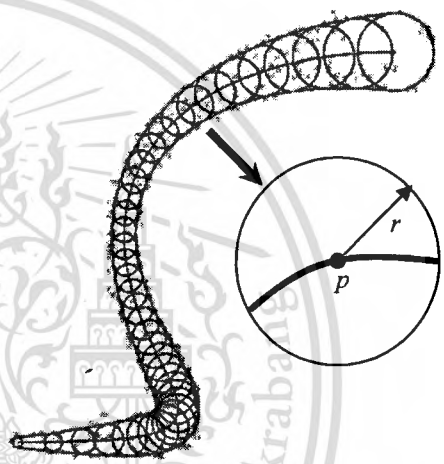


Figure 1. The medial axis of an object.

B. The Geodesic Distance Function

A Riemannian space is a mathematical geometry concept that studies curves and surfaces in higher dimensions, giving a precise meaning to concepts like angle, length, area, volume and curvature. On a Riemannian system, the geodesic distance is the distance between two points on the surface of the model, computed by using the Dijkstra's algorithm to find the shortest path between the two points on the surface made up by n points [14]. The following definitions characterize the geodesics based on [9].

Definition 4. Given R^3 be a surface in a Riemannian space G, and source vertex $v_s \in G$, an explicit representation of the geodesic distance function D: $G \rightarrow R^3$. For any point $p \in G$, this function $D(p)$ returns the length of the geodesic path from p back to the source v_s. The approximation of $D(p)$ can be given by

$$\|x_i - x_j\| \approx D(x_i, x_j) \text{ when } x_j \approx x_j$$

Consider the length of a curve S, represented in R^3 by

equations

$$S: x^i = x^i(t), t_1 \le t \le t_2,$$

Definition 5. The distance s between two points t_1 and t_2 on a curve $x^\gamma = x^\gamma(t)$ in R^3 is given by

$$s = \int_{t_1}^{t_2} \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}\, dt, \qquad (\infty, \beta = 1, \ldots, n). \quad (1)$$

The minimum of (1) will be yielded a geodesic of the space, by using Euler's or Lagrange's equations:

$$\frac{\partial F}{\partial x^k} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{x}^k}\right) = 0$$

with

$$F = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}$$

we have

$$\frac{\partial F}{\partial x^k} = \frac{1}{2}(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta)^{-1/2} \frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta$$

$$\frac{\partial F}{\partial \dot{x}^k} = \frac{1}{2}(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta)^{-1/2}\, 2 g_{\infty k}\dot{x}^\infty$$

since

$$\frac{ds}{dt} = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}$$

Euler's equations can be written in the form

$$\frac{d}{dt}\left(\frac{g_{\infty k}\dot{x}^\infty}{\dot{s}}\right) - \frac{1}{2\dot{s}}\frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta = 0,$$

and, carrying out the indicated differentiation, we obtain

$$g_{\infty k}\ddot{x}^\infty + \frac{\partial g_{\infty k}}{\partial x^\beta}\dot{x}^\infty \dot{x}^\beta - \frac{1}{2}\frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta = \frac{g_{\infty k}\dot{x}^\infty \ddot{s}}{\dot{s}}$$

by writing

$$\frac{\partial g_{\infty k}}{\partial x^\beta}\dot{x}^\infty \dot{x}^\beta = \frac{1}{2}\left(\frac{\partial g_{\infty k}}{\partial x^\beta} + \frac{\partial g_{\beta k}}{\partial x^\infty}\right)\dot{x}^\infty \dot{x}^\beta$$

we obtain

$$g_{\infty k}\ddot{x}^\infty + [\infty\beta, k]\,\dot{x}^\infty \dot{x}^\beta = \frac{g_{\infty k}\dot{x}^\infty \ddot{s}}{\dot{s}}$$

if we use curve length as parameter, $\dot{s} = 1, \ddot{s} = 0$, then the equation becomes

$$g_{\infty k}\ddot{x}^\infty + [\infty\beta, k]\,\dot{x}^\infty \dot{x}^\beta = 0$$

multiplying by $g^{\gamma k}$, we obtain

$$\ddot{x}^\gamma + [\infty\beta, r]\,\dot{x}^\infty \dot{x}^\beta = 0 \quad (2)$$

These are the desired equations of geodesics. In equation (2), dots denote the differentiation with respect to the length parameter of the curve S. According to [3], the geodesic distance function given by Eq. (2) is rotation and translation invariant.

III. Implementation Details

A. Pruning the skeleton

According to [16], the following definitions are used to formally define the skeleton in our approach:

Definition 6. The *degree* of a point is defined as a finite number of points in its 26-neighborhood (figure 2 shows the 26-neighborhood structure).

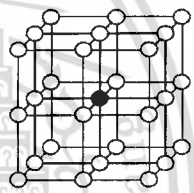


Figure 2. The 26-neighborhood structure.

Definition 7. By the assumption that the skeleton is one-voxel thick, the skeleton *end point* is a point that has a degree one. The *middle point* is a point that has a degree two. The *connection point* is a point that has a degree three or higher and all the other neighbors are either the end points or the middle points. The *branched connection point* is a point that has a degree three or higher and at least one of its neighbors is neither an end point nor a middle point. The 26-connected branched connection points form the *branched region*.

Algorithm 1 shows the steps of pruning the skeleton, the original 3D object, and the result of pruning are shown in figure 3. The method involves of nine different steps:

(1) Compute the degree of each point.
(2) Determine which point are the end points, the middle points, the connection points, and the branched connection points. If there are no branched connection points, exit the program.
(3) Organize the branched connection points into the branched regions. Each branched region consists of 26-adjacent branched connection points.
(4) In each branched region, find all the end points and the middle points that are 26-adjacent to this branched region, and

Input: A set of points on the surface of model, $P = \{p_1, p_2, \ldots, p_n\}$.
Output: The pruned skeleton of the given model.
// Read P and determine which p is an endpoint, a middle point, a connection point, and a branched connection point.
for all $p \in$ *surface*
  compute *degree of p*
    if *degree of* $p = 1$
      $p \leftarrow$ *endpoint*
        if *degree of* $p = 2$
          $p \leftarrow$ *middle point*
            if *degree of* $p \geq 3$ **and** all the other neighbors are either the end points or the middle points
              $p \leftarrow$ *connection point*
                if *degree of* $p \geq 3$ **and** at least one of its neighbors is neither an end point nor a middle point
                  $p \leftarrow$ *branched connection point*
                  store p in *ListOfBranchedConnectionPoint*
                end if
            end if
        end if
    end if
end for
// Construct a branched region from p in ListOfBranchedConnectionPoint.
for all $p \in$ *ListOfBranchedConnectionPoint*
  if *NumberOfNeighbours*$(p) = 26$
    construct *BranchedRegion*(p)
  end if
end for
// Find the end points and the middle points that are 26-adjacent to this branched region, and mark each as "terminator".
for all $p \in$ *BranchedRegion*
  if *degree of* $p = 1$
    $p \leftarrow$ *endpoint*
    $p \leftarrow$ *terminator*
      if *degree of* $p = 2$
        $p \leftarrow$ *middle point*
      $p \leftarrow$ *terminator*
    end if
  end if
end for
// Determine the centroids in each branched region.
// Compute the geodesic distance and find the shortest path correspond to each centroid.
for all $p \in$ *BranchedRegion*
  $CentroidOfBranchedRegion = (\sum_{i=1}^{N} x_i/N, \sum_{i=1}^{N} y_i/N, \sum_{i=1}^{N} z_i/N)$
    if $p \approx$ *CentroidOfBranchedRegion*
      $p \leftarrow$ *centroid*
        geod_dist (p, p_i) // compute the geodesic distance from centroid to all p_i, where p_i = terminator
      Dijkstra's_shortest_path (p, p_i) // apply the *Dijkstra's* algorithm
                                  // to find the shortest path from centroid to all p_i
        construct *ShortestPath*(P) where $P = (p, p_i)$
  end if
end for
// Remove the branched connection points that are not on any of the shortest paths.
// Construct the pruned skeleton.
for all $p \in$ *BranchedRegion*
  for all $p \in$ *ListOfBranchedConnectionPoint*
    if $p \notin$ *ShortestPath*
      remove p
      update *ListOfBranchedConnectionPoint*
      construct *PrunedSkeleton*(p) // construct the pruned skeleton from the remaining points in the list
  end if
 end for
end for

Algorithm 1 Skeleton pruning method.

mark each as "terminator".
(5) Determine the centroids, based on Blum's definition of a skeleton; the skeleton points must be the centers of the maximal disks contained in the curve C.
(6) Compute the geodesic distance from the particular centroids to all points in step 4.
(7) Apply the Dijkstra's algorithm, in order to find the shortest paths between the centroids and their "terminator".
(8) Remove the branched connection points that are not on any of the shortest paths.
(9) Generate the pruned skeleton from the remaining points.

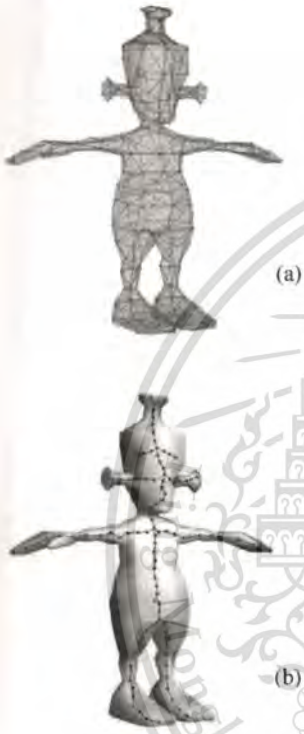


Figure 3. (a) The original 3D mesh model (b) Constructing the pruned skeleton.

B. Smoothing the skeleton

Once we can eliminate a large number of spurious skeleton joints, the next problem is raised which is the meaningless points that align unreasonably along a skeleton. These points can affect on the later creation of segments and joints, therefore the pruned skeleton is subjected to a smoothing operation. The smoothing process involves calculating the average position of consecutive points along the pruned skeleton, reassigning the new joints by splitting a skeleton into the segments [17].

i) Adjusting the skeleton

The pruned skeleton has been originated with a one-voxel thick, in a graph-like form as shown in figure 3 (b). Before we apply chord-to-point distance accumulation to the pruned skeleton, the skeleton needs to be smoothened as nearly as a curve. This can be accomplished by using the average filtering over a sliding window consisting of five consecutive points to adjust the positions of consecutive points along the skeleton.

Let $C = \{p_i = (x_i, y_i, z_i), i = 1,\ldots, N\}$ be the set of coordinate of points on the skeleton. The ith point in the set is denoted by p_i and p_{i+1} is its neighboring point. To calculate the average at the point p_i, we make an approximation of the change in the positions of two consecutive points before p_i and two consecutive points after p_i and divide it by 5 (the number of points between $p_{i-2}, \ldots, p_i, \ldots, p_{i+2}$). Thus, the average position (v_i) over the point p_i can be defined as equation (3). The filtering process is illustrated in figure 4.

$$v_i = \frac{\sum_{i-2}^{i+2} p_i}{5} \tag{3}$$

ii) Locate the smoothed joints

After the smoothed skeleton is formed, the next step is to determine which points should be split. Splitting of the skeleton is accomplished by considering chord-to-point distance accumulation value. We applied chord-to-point distance accumulation proposed by Han and Poston [6] to our method. Unlike Han and Poston's method, we use the geodesic distance instead of the Euclidean distance.

The distance feature computed by Han and Poston's method is invariant with respect to rotation and translation which is a major benefit for animating a skeleton. Chord-to-point distance accumulation is the distance measurement from the line to a point in the curve segment. The interpretation of chord-to-point distance accumulation can be defined as follows.

Let L be a fixed integer value defines a line L_i from each point p_i to p_{i+L}, where $i+L$ is taken modulo N. The perpendicular distance D_{ik} is computed from L_i to the point p_k. The distance is positive if p_k is on the left-hand side of the vector $(p_{i+L} - p_i)$, negative otherwise. Chord-to-point distance accumulation for a point p_k and a chord length L is the sum h_L of the D_{ik} as i moves from k-L to k. That is,

$$h_L(k) = \sum_{i-k-L}^{k} D_{ik} \tag{4}$$

Since $D_{kk} = 0$, Thus,

$$h_L(k) = \sum_{i-k-L}^{k} D_{ik} = \sum_{i-k-L+1}^{k-1} D_{ik}$$

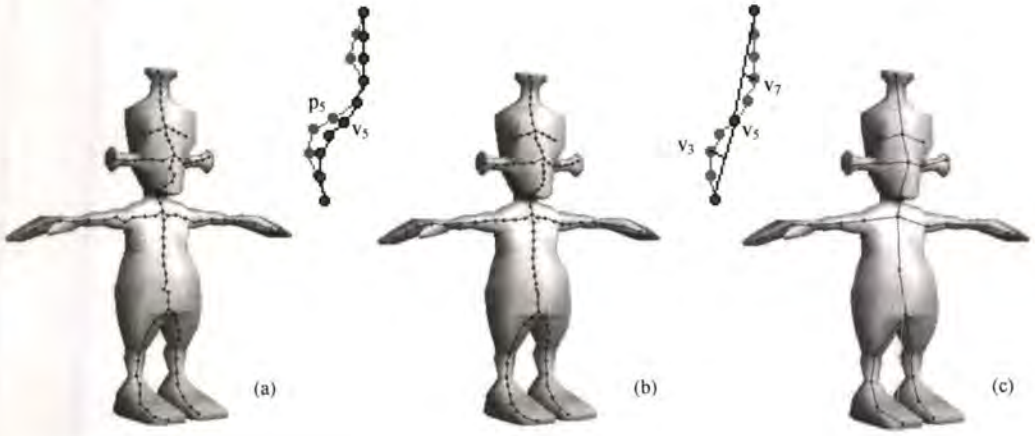


Figure 4. The method for smoothing the pruned skeleton. (a) The average position v_5 over $p_5 = (p_3+p_4+p_5+p_6+p_7)/5$, in order to obtain the smoothed skeleton, the position v_i is computed for each p_i on the pruned skeleton. (b) The result of applying the smoothing operation to the pruned skeleton from figure 4 (a), Once the smoothed skeleton is formed, chord-to-point distance accumulation is applied for splitting the smoothed skeleton into segments in order to form bones and joints. Notice that the best split is formed at v_5 since the value of chord-to-point distance accumulation at v_3 and v_7 are equal. (c) The new joints are formed as a result of the best splitting based on chord-to-point distance accumulation value.

Figure 5 illustrates chord-to-point distance accumulation, and shows an example in the case of D_{ij} is positive and D_{ik} is negative.

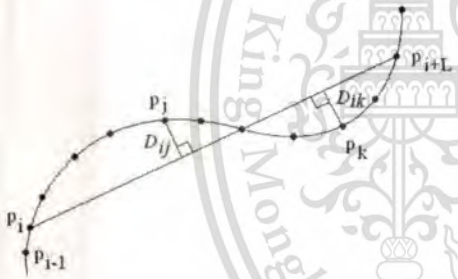


Figure 5. Chord-to-point distance accumulation.

The basic idea of locating the new joints depends on chord-to-point distance accumulation value. After the smoothed skeleton is formed, the points that align along the smoothed skeleton can be classified into three classes. The end point is a point that has only one adjacent neighbor. The junction point is a point that has three or more adjacent neighbors. The intermediate point is a point that has exactly two adjacent neighbors. The end points and the junction points split the smoothed skeleton into a set of connected segment or bones.

We use the end points and the junction points to form the initial joints of the skeleton. In each segment of the skeleton (see figure 4 (b)), we iteratively select the point from v_{i+1} to v_{N-1} and compute chord-to-point distance accumulation correspond to each selected point. The selected point becomes the splitting point if $\|D_{ij}\| = \|D_{ik}\|$ e.g. v_5 is the splitting point since chord-to-point distance accumulation value at v_3 equals to chord-to-point distance accumulation value at v_7. Then we can form two bones from the corresponding segment, the first bone whose joints are located on v_1 and v_5, another bone whose joints are located on v_5 and v_{10} respectively. In the case of more than one splitting point is created, the sub segment will be formed (this sub segment becomes a bone later on), and more joints may be created from the corresponding sub segment. Figure 4 (c) shows the final result of the smoothed skeleton generated from our method. The algorithm for smoothing the skeleton is also shown.

```
//Given skeleton V consists of N points places in a
queue Q.
Smoothen(V, N)
  If N ≤ 0
    Return V
  Repeat
    Q ← V
    N ← 1
    Repeat while Q ≠ ϕ
      v ← POP(Q)
        Compute h_L(k)
      If ||D_ij|| = ||D_ik||
        V ← V-v
```

Algorithm 2 The iterative smoothing.

IV. Experimental Results

The pruning algorithm is tested on the Princeton Shape Benchmark database [13], and a standard handmade model (from Autodesk's Maya software). A brief comparison between the results of the proposed method and their original skeleton, extracted by using the Reeb graph is provided. The

proposed method can reduce the number of the branched connection points, which is similar to the number of critical points in the original Dijkstra's algorithm therefore the spurious skeleton joints can be removed. These joints will not influence the structure of the skeleton because they are not on any of the shortest paths computed by Dijkstra's algorithm. The pruned skeleton can then be generated in the sense that it can capture the essential shape of a 3D object while preserving a compact form of its data structure. Figure 6 shows the experimental results.

Mesh	Number of vertices	
	The Reeb graph	The pruned skeleton
Cow	2904	2655
Rabbit	1238	1026
The Dragon	1166	1011
The Alien	429	303
The Femme	635	503
Boy	17342	16017

Table 1. Comparison of the number of points between the proposed method and the original model.

Object	Number of vertexes	Number of faces	Number of control segments	
			The pruned skeleton	The smoothed skeleton
M149-boy	16017	32007	860	19
M232-alien	732	1436	40	30
Femme	503	1002	50	26
Horse	535	1058	56	27
Dinopet	2039	3999	120	31

Table 2. Comparison of the number of control segments between the smoothed skeleton and the pruned skeleton.

Table 1 compares the number of points after applying the pruning algorithm with the number of points of the original 3D objects. The proposed method can reduce the number of critical points in the graph's nodes, the branched connection points in a skeleton graph. Therefore, the numbers of the spurious skeleton joints are reduced comparable to the number of unnecessary joints generated by the Reeb graph method.

We also demonstrate our smoothing alogorithm on several pruned skeletons generated from algorithm 1, as shown in figure 7. A brief comparison between the results of the smoothing algorithm and their original pruned skeletons is provided (figure 6). The smoothing algorithm can produce a useful control skeleton, the number of control segments can be significant reduced in the sense that it is still enough for use in animation. Most of the control skeletons produced from our algorithm are centralized, and run along to the ends of the main branches of the models.

Table 2 shows the results of the number of control segments after applying the smoothing algorithm with the number of control segments of the original pruned skeletons. Figure 6 (c) shows several objects of a human-like model as smoothed by our algorithm. Segments and joints relate fairly well to surface features; however there is room for improvement. For instance for the boy-M149, there is no control points for fingers or toes are produced.

The computational complexity of our approach can be derived below.

Let n be the number of points in the meshed model. The time complexity of our algorithm can be considered as follows.

The pruning algorithm: the computation of degree of each point and the point analysis takes $O(n)$ time. The computation of testing and finding the maximal disks in the neighborhood of each point takes $O(n)$ time. The computation of the single source shortest paths (in term of the geodesic distance from that source vertex) to all points on the mesh takes $O(n \log n)$ time [5]. Thus, the overall computation takes $O(n \log n)$ time.

The smoothing algorithm: the computational complexity of averaging position of v_i over the point p_i in the filtering process takes $O(n)$ steps. The computational complexity of chord-to-point distance accumulation takes $O(n)$ steps. We implement our algorithm to place n points in a queue data structure, and most of the times are spent on the *POP* operation, thus the time complexity of this process is $O(\log n)$ steps. Thus, the overall computational complexity of the smoothing algorithm takes $O(n)$ steps (this is not include the complexity analysis of the skeleton pruning algorithm which takes $O(n \log n)$ steps).

V. Conclusion and Future Works

In this paper we have presented our approach of geodesic-based skeleton smoothing to compute the skeleton for 3D meshed model. Using Blum's Medial Axis Transform together with the geodesic distance function, and Han and Poston's Chord-to-point Distance Accumulation, we can generate one-voxel thick, graph-like skeleton which can capture the essential shape characteristics in a compact form. Our approach also maintain the robustness against any changes of a rotation and/or a translation of the 3D meshed model which are the major properties of 3D objects. In figure 6 (a) and (d), it is easy to observe that the skeleton extracted by using the Reeb graph is very sensitive to the connectivity of the boundary representation and it also has a problem in producing the unwanted skeleton joints. The pruning algorithm can overcome these drawbacks; the results are shown in figure 6 (b) and (e). In the smoothing step, the collection of points produced from our smoothing algorithm is transformed into a collection of bones which can be used for 3D animation. The main process of this method is based on chord-to-point distance accumulation introduced by Han and Poston [6]. Unlike Han and Poston's method, we use the geodesic distance instead of the Euclidean distance. We gain a benefit of the invariance

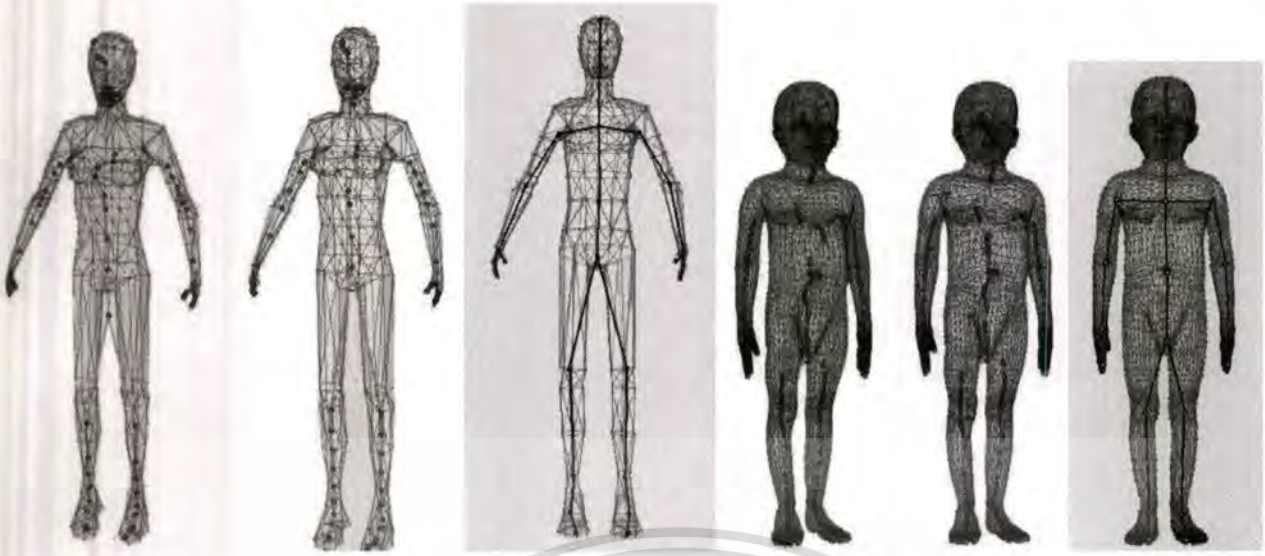


Figure 6. The pruned skeletons (b) and (e), and their skeletons extracted by using the Reeb graph (a) and (d). (c) and (f) are the results of geodesic-based skeleton smoothing.

against a rotation and a translation from the geodesic distance used in chord-to-point distance accumulation to construct the rotation and the translation invariant control skeleton. The algorithm produces the useful control skeletons with a small number of control segments. It suits for producing skeletons for more complex objects (the object which have a large number of vertex, and faces), since it can reduce the number of control segments comparable to the number of unnecessary segments generated by the skeleton pruning method. Figure 6 (c) and (f) show the meaningful characteristics of the 3D meshed model, and the generated skeletons whose joints can be associated with the 3D meshed model.

Given the skeleton construction process with the proposed approach of geodesic-based skeleton smoothing, significant improvements are desirable, either by increasing algorithm robustness or speed. Another robustness, we are taking into account is the robustness against the deformation of a model into the motion tween, which is the major technique for producing 3D character animation.

REFERENCES

[1] M. K. Agoston. Computer graphics and geometric modeling:implementation and algorithms. Springer, ISBN 1852338180, page 185, 2005.

[2] G. Aujay, F. Hetroy, F. Lazarus, and C. Depraz. Harmonic skeleton for realistic character animation. In *Symposium on Computer Animation (ACM SIGGRAPH)*, 2007.

[3] S. Baloch, H. Krim, I. Kogan, and D. Zenkov. Rotationinvariant topology coding for 2D and 3D objects usingMorse theory. In *Proceeding of IEEE International Conference on Image Processing*, ICIP 2005, pages 796-799, Sep 2005.

[4] H. Blum, R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition 10*, pages 167-180, 1978.

[5] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. In *Symposium on Computational Geometry*, pages 344-350, 2003.

[6] J.H. Han and T. Poston, Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature. *Pattern Recognition Letters*, vol. 22, pp. 1133-1144, 2001.

[7] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *Computer Graphics Annual Conference Series (Proc. Of SIGGRAPH)*, pages 203– 212, 2001.

[8] F. Lazarus, and A. Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, ACM Press, pages 130–140, 1999.

[9] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal of Computing 16(4)*, pages 647-668, 1987.

[10] T. Oda, Y. Itoh, W. Nakai, K. Nomura, Y. Kitamura, and F. Kishino. Interactive skeleton extraction using geodesic distance. In *Proceeding of the 16th International Conference on Artificial Reality and Telexistence*, 2006.

[11] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [On the singular points of a complete integral pfaff form or of a numerical function].*Comptes Rendus Acad.Science Paris 222*, pages 847–849, 1946.

[12] M. Sabry Hassouna and Aly A. Farag. On the extraction of curve skeletons using gradient vector flow. *Proc. of IEEE International Conference on Computer Vision ICCV*, Rio de Janeiro, Brazil, October 14-20, 2007.

[13] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In Shape Modeling International, pages 167-178, 2004.

[14] O. Symonova. Shape description for 3D model retrieval. Technical Report #DISI-08-010, University of Trento, Mar 2008.

[15] O. Symonova, and G. Ucelli. Using topology representation for comparison and retrieval of CAD models. *Computer Graphics topics*, pages 29-30, 5/2005, Vol. 17.

[16] P. Visutsak and K. Prachumrak. The 3D skeleton pruning for removing undesired joints. In *Proceedings of The Second IEEE International Conference on Computer, Control and Communication IEEE-IC4 2009*, Karachi, Pakistan, February 17-18, 2009.

[17] P. Visutsak and K. Prachumrak. The smoothed 3D skeleton for

animation. In *Proceedings of the 5th International Conference on INC, IDC, IMS*, Seoul, Korea, August 25-27, 2009.
[18] L. Wade and R.E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer 18*, 2002.

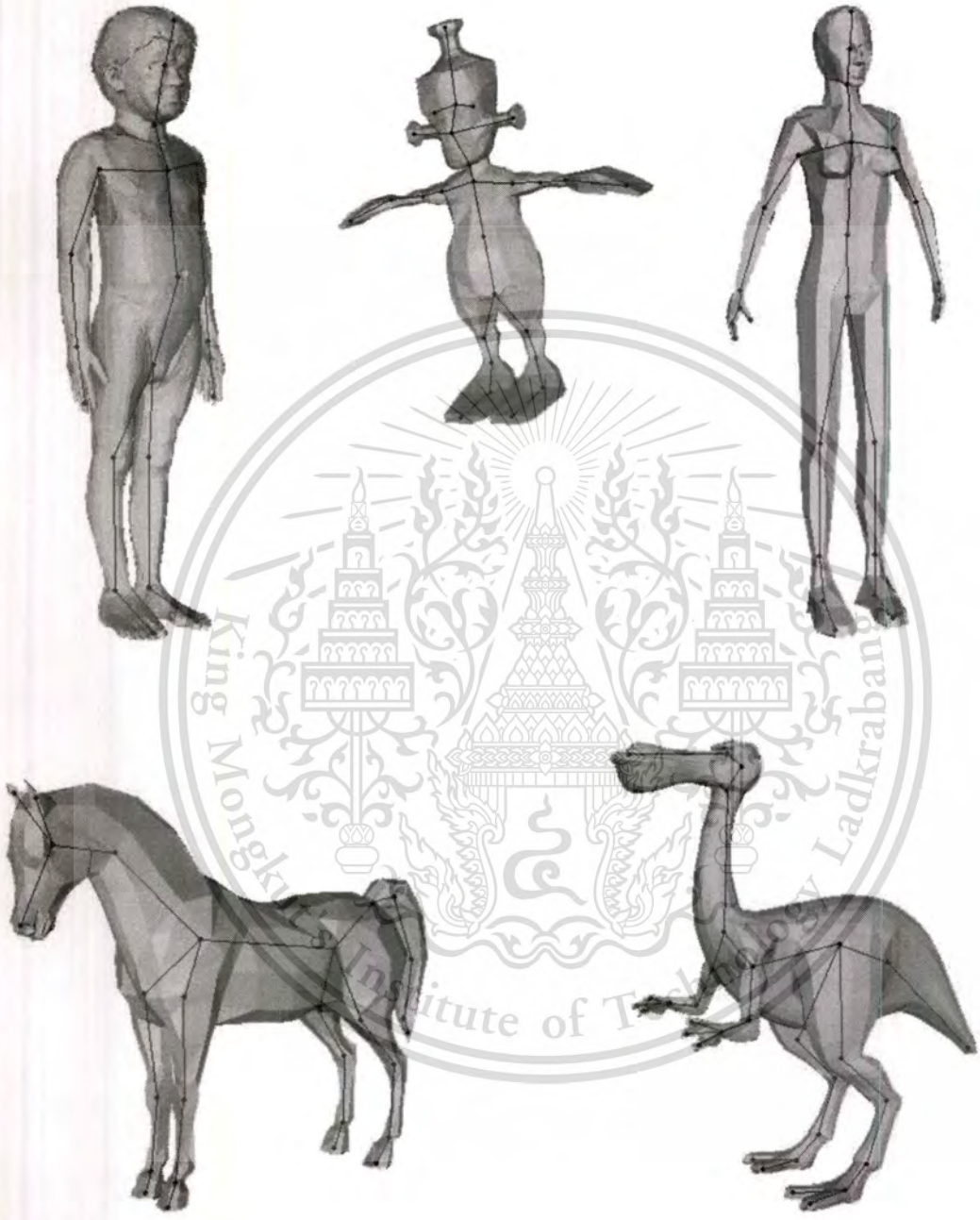


Figure 7. The smoothed skeleton of the objects: M149, M232, Femme, Horse, and Dinopet respectively.

CGiV2010

7th International Conference
Computer Graphics, Imaging & Visualisation

The Book of Abstract

University of Technology, Sydney
■ Sydney ■ Australia ■

UNIVERSITY OF TECHNOLOGY SYDNEY

Kuwait University

GraphicsLink

VG RU

Perceptually-Guided Design of Non-perspectives Through Pictorial Depth Cues
Kenichi Yoshida1, Shigeo Takahashi1, Hiroaki Ono1, Issei Fujishiro2, Masato Okada

A full-day Event: Saturday 7th August 2010, Time: 9:30 -17:00

Computer Graphics, Imaging and Visualisation
DOCTORAL RESEARCH WORKSHOP

Organised by

Visualisation & Graphics Research Unit of LSBU, UK

&

Department of Computer Systems, Faculty of Information Technology, University of Technology, Sydney, Australia

Computer Graphics, Imaging and Visualization –CGIV- Forum is an annual forum that is held for 7 year running. This year CGIV forum in collaboration with the Visualisation & Graphics Research Unit of LSBU, UK and the Department of Computer Systems , Faculty of Information Technology, University of Technology, Sydney, Australia are pleased to announce Doctoral Research Workshop within the scope of the 7th International conference on Computer Graphics, Imaging and Visualization (CGIV2010). This workshop provides an opportunity for PhD students to present their work, receive feedback and to meet other researchers working in CGIV area. The focus of this workshop will be on the pros and cons of various Computer Graphics, Imaging and Visualization ideas and solutions and its potential impact on both the research community and the industry in general.

All doctoral students involved in Computer Graphics, Imaging and Visualization research area are welcome to attend. The event is organised in four sessions with up to four PhD students per session. Presenters, which are PhD students at various stages of their PhD, will give an outline of their PhD research in order to benefit from feedback about their work and methodology from a combined industry & research panel.

Saturday 7 August 2010

10:00	< UTS - Lecture theatre Foyer >
	Registration
10:30 - 13:00	< UTS - CB10.02.240 Postgraduate Seminar Room >
	A full-day Event: saturday 7th August 2010, Time: 10:30 -16:30 **Computer Graphics, Imaging and Visualisation** **Doctoral Research Workshop** Chair: Professors Ebad Banissi, Muhammad Sarfraz, and Mao Lin Huang **10:30 An introduction from Doctoral Research Workshop chair and organiser** **11:00** Visutsak, Porawat, King Mongkut's Institute of Technology Ladkrabang, Thailand **11:45** Wahab, Dr. Abd Fatah, University Malaysia Terengganu, Malaysia **12:30 View of what examiners look for in a PhD**
13:00	< UTS - Dining Hall >
	Lunch Break
14:00	< UTS - CB10.02.240 Postgraduate Seminar Room >
	14:00 Choi, Changryoul, Hanyang University, South Korea **14:45** Oh, Hyeongchul, Hanyang University, South Korea **15:30 Break** **16:00 1-1 feedback round table discussion** **Final commentary** **Close**

ABSTRACTS

Doctoral Research Workshop

Chair: Professors Ebad Banissi, Muhammad Sarfraz, and Mao Lin Huang

Knowledge-based approach for skeleton approximation

Visutsak, Porawat; Prachumrak, Korakot

A 3D skeleton is a one-voxel thick, graph-like structure widely used in the area of a character animation. In this paper, we propose a novel method for computing the skeleton of the 3D character models based on a-priori knowledge. We present several algorithms for computing the skeleton based on Blum's Medial Axis Transform and geodesic distance algorithm. We show that the algorithm can generate the smoothed skeleton in a compact form, while providing the essential joints of the skeletal structure. Since most of skeletons of biped and quadruped character models have the same global features (spine and joints), the notion of similarity is crucial importance in comparing the unknown character models with the others stored in the database. We extend the algorithm with an approach of content-based analysis for automatic and efficient retrieval, classification, and annotation of the 3D character models. The method has several steps. First, the octree of the input model is calculated and used to compare with the octrees of the 3D models stored in the octrees database. By comparing the octrees similarity, if the result is exact match, the corresponding skeleton will be retrieved from the skeletons database. Otherwise, the method finds the list of the close match (the octree similarity ratio which the value is greater than 0.8) in order to use to estimate the new skeleton. In the worst case, if the input octree does not match with any case in the octrees database, the method computes the skeleton and stores it in the skeletons database. The method is fast and efficient because it is not necessary to compute the skeleton from every input model. Thus, the computational time of our method depends only on the time of the octrees similarity calculation, and the time of searching the similar octrees in the octrees database. Several examples show the results obtained with our approach.

Fuzzy Interpolation Rational Bezier Curve

Wahab, Abd Fatah; Zakaria, Rozaimi; Md Ali, Jamaluddin

Recently, interpolation methods especially in curves design are widely used for modeling data points. In this paper a new interpolation method for modeling fuzzy data using fuzzy interpolation rational Bezier curves is introduced based on fuzzy set theory. This method is able to model uncertainty data with definition of fuzzy data via fuzzy number concept. The fuzzy interpolation method is modeled using fuzzy interpolation rational cubic Bezier curve (in further will be referred as FIRCBC). For fuzzy n-data cases, segments curve constructed in order to interpolate fuzzy data piecewisely. For the illustration as hypothetical example, FIRCBC has been applied in verification of offline handwriting signature where the signatures become fuzzy cases.

Low Complexity Weighted Two-Bit Transforms Based Multiple Candidates Motion Estimation Exploiting the Redundant Computations

Choi, Changryoul; Jeong, Jechang

A low complexity weighted two-bit transforms (2BT) based multiple candidates motion estimation algorithm is proposed in this paper. By exploiting almost the identical operations in two different matching error criteria, we can efficiently determine two best motion vectors according to the respective matching criteria and can enhance the overall motion estimation accuracy. Experimental results show that the proposed algorithm achieves peak-to-peak signal-to-noise ratio (PSNR) gains about 0.47dB on average compared with the conventional 2BT-based motion estimation.

Motion Compensated Frame Interpolation using Adaptive Adjacency Pixel Information

Oh, Hyeongchul; Lee, Joohyun; Min, Changki; Jeong, Jechang

We propose an adaptive motion compensated frame interpolation scheme to develop the frame rate from a lower number into a higher

GEODESIC-BASED SKELETON EXTRACTION

Porawat Visutsak[1,2] *Korakot Prachumrak*[1]

[1]Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang
Bangkok, THAILAND
[2]Faculty of Liberal Arts and Science, Kasetsart University, Kamphaeng Saen Campus
Nakhorn Pathom, THAILAND

ABSTRACT

Skeleton is at the core of shape representation of the 3D objects, usually can be formed in a graph-like structure with a one voxel-thick. In this paper, we present a new algorithm to compute the skeleton of the 3D meshed model in a Riemannian space, based on Blum's Medial Axis Transform and geodesic distance algorithm. We gain the benefit of geodesic distance functions and parameterization that allow for efficient handling of topological changes of dynamics curves and surfaces. Thus, our algorithm can provide the robustness against any changes of a rotation and/or a translation of the 3D meshed model which are the major characteristics of the 3D objects understanding. Consequently, the generated skeleton can capture in a simple and meaningful way the essential shape of the 3D objects in a compact form.

Index Terms— Skeleton, Geodesic distance, Medial axis transform, Riemannian space

1. INTRODUCTION

We present a new method for generating a skeleton in the Riemannian space, based on the medial axis transform and the geodesic distance function associated with a 3D meshed object. The skeleton, generated by the new method captures the overall shape as well as the topology of an object. The skeleton is a basic structure of representing the 3D objects, frequently used in computer graphics, especially in the areas of 3D modeling. There are many approaches to represent 3D models in graph-like or a curve-like form as a fundamental data structure of the shapes the current method aims at reducing the 3D objects into a lower dimensional form that captures their shape characteristics (topology and/or geometry). The most common technique to represent the 3D shapes, that has been the standard for many years, is the Reeb graph [8], originally defined by Reeb.

The Reeb graph is obtained by applying the continuous function, usually a height function, to encode the topological structure. Using a height function to build a Reeb graph does not guarantee that the graph is invariant to the affine transformations, which is an essential feature of the skeletal structure of the model [9]. To overcome this drawback, the height function is replaced with the geodesic distance function [6]. By providing the invariance against a rotation and a translation [9], the geodesic distance function allows to construct the rotation and the translation invariant Reeb graph.

The extended version of the Reeb graph, called the Multi-Resolution Reeb graph (MRG) [5] has been proposed. To construct the MRG, the topological characteristics of the shape must be defined in terms of the critical points of a function on the manifold; this function is called a mapping function. The mapping function maps the points from the manifold of the shape to the domain of the function, and the configuration of the critical points of the mapping function can represent the topology of the shape. This configuration can be embedded by the Reeb graph, and becomes an essential property of the shapes. When the mapping function is defined, the model is then partitioned into regions that correspond to equal intervals of the mapping function. Each partition of the model is represented as a node in the Reeb graph, and adjacent nodes are linked by an edge that connects the corresponding nodes.

Blum et al. [3] propose to use the medial axis to define the skeleton. The algorithm for computing the skeleton in terms of the medial axis is often refers to as the "grassfire" algorithm. The main idea of the "grassfire" algorithm is lighting a fire, started from the border of the object, and let it burn into the object at a constant speed; it will then meet in the medial axis. One starts on the border of the object and strips away one layer of pixels after another until one reaches points that fire reaches from two directions [1]. The medial axis transform of the region is the set of points reached by more than one fire at the same time. Unfortunately, the medial axis transform does not provide robustness against a rotation and a translation of the objects. Therefore, a more sophisticated algorithm is needed in order to solve this problem, and that algorithm is the geodesic distance function. By using the geodesic approximation algorithm proposed by [7] to compute the "single source, all destination" shortest path on a surface of the model, any changes of a rotation and/or a translation do not affect the value of a function. Thus, this becomes the major property of the function of geodesic distance that gives the advantage of being invariant to a rotation and a translation.

Numerous Reeb graph applications have already been proposed for the freeform 3D models unfortunately, they are sensitive to the connectivity of the boundary representation, which is often violated and they cannot capture the geometry of the shapes, which is an important feature in shape analysis [9]. This paper introduces the geodesic distance function together with the medial axis transform for generating the skeleton based on the Dijkstra's shortest path. The basic idea of the new method is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region determination (see the proposed algorithm for detail). Applying the function of geodesic distance can guarantee that the new method is invariant to a rotation and a translation, and it is also robust against the changes in the connectivity on the 3D shapes. A brief review of the geodesic distance function and the medial axis transform are illustrated in sections 2 and 3. The proposed method is outlined in section 4. Finally, the experimental results and the conclusion are given in sections 5 and 6, respectively.

2. GEODESIC DISTANCE FUNCTION

The geodesic distance can be defined as the distance between two points on the surface of the model, computed by using the Dijkstra's algorithm to find the shortest path between the two points on the surface made up by n points. The following definitions characterize the geodesics based on [7].

Definition 1. Given R^3 be a surface in a Riemannian space G, and source vertex $v_s \in G$, an explicit representation of the geodesic distance function D: $G \rightarrow R^3$. For any point $p \in G$, this function $D(p)$ returns the length of the geodesic path from p back to the source v_s. The approximation of $D(p)$ can be given by

$$\|x_i - x_j\| \approx D(x_i, x_j) \text{ when } x_j \approx x_j$$

Consider the length of a curve S, represented in R^3 by equations

$$S: x^i = x^i(t), t_1 \le t \le t_2,$$

Definition 2. The distance s between two points t_1 and t_2 on a curve $x^\gamma = x^\gamma(t)$ in R^3 is given by

$$s = \int_{t_1}^{t_2} \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}\, dt, \qquad (\infty, \beta = 1, \ldots, n). \quad (1)$$

The minimum of (1) will be yielded a geodesic of the space, by using Euler's or Lagrange's equations:

$$\frac{\partial F}{\partial x^k} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{x}^k}\right) = 0$$

with

$$F = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}$$

we have

$$\frac{\partial F}{\partial x^k} = \frac{1}{2}(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta)^{-1/2} \frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta$$

$$\frac{\partial F}{\partial \dot{x}^k} = \frac{1}{2}(g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta)^{-1/2}\, 2g_{\infty k}\dot{x}^\infty$$

since

$$\frac{ds}{dt} = \sqrt{g_{\infty\beta}\dot{x}^\infty \dot{x}^\beta}$$

Euler's equations can be written in the form

$$\frac{d}{dt}\left(\frac{g_{\infty k}\dot{x}^\infty}{\dot{s}}\right) - \frac{1}{2\dot{s}}\frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta = 0,$$

and, carrying out the indicated differentiation, we obtain

$$g_{\infty k}\ddot{x}^\infty + \frac{\partial g_{\infty k}}{\partial x^\beta}\dot{x}^\infty \dot{x}^\beta - \frac{1}{2}\frac{\partial g_{\infty\beta}}{\partial x^k}\dot{x}^\infty \dot{x}^\beta = \frac{g_{\infty k}\dot{x}^\infty \ddot{s}}{\dot{s}}$$

by writing

$$\frac{\partial g_{\infty k}}{\partial x^\beta}\dot{x}^\infty \dot{x}^\beta = \frac{1}{2}\left(\frac{\partial g_{\infty k}}{\partial x^\beta} + \frac{\partial g_{\beta k}}{\partial x^\infty}\right)\dot{x}^\infty \dot{x}^\beta$$

we obtain

$$g_{\infty k}\ddot{x}^\infty + [\infty\beta, k]\,\dot{x}^\infty \dot{x}^\beta = \frac{g_{\infty k}\dot{x}^\infty \ddot{s}}{\dot{s}}$$

if we use curve length as parameter, $\dot{s} = 1, \ddot{s} = 0$, then the equation becomes

$$g_{\infty k}\ddot{x}^\infty + [\infty\beta, k]\,\dot{x}^\infty \dot{x}^\beta = 0$$

multiplying by $g^{\gamma k}$, we obtain

$$\ddot{x}^\gamma + [\infty\beta, r]\,\dot{x}^\infty \dot{x}^\beta = 0 \quad (2)$$

These are the desired equations of geodesics. In equation (2), dots denote the differentiation with respect to the length parameter of the curve S. According to [2], the geodesic distance function given by Eq. (2) is rotation and translation invariant.

3. THE MEDIAL AXIS TRANSFORM

The idea of using a skeleton of an object as an abstraction of the shape goes back to [3]. Blum et al. define a *skeleton* in

terms of the *medial axis* (*MA*), which is a set of curves that roughly run along the middle of an object, as shown in figure 1. According to [3], the medial axis of a curve S is the locus of the centers of the maximal disks contained in S. Let R^n be a symmetry set (where n is a number of dimensional space) which is defined similarly to the medial axis, except that it also includes the circles not contained in S and thus the medial axis is a subset of the symmetry set. The medial axis can then be defined as follows:

Definition 3. Let S be an arbitrary curve; Let $D^n(p, r)$ be a closed disk with a radius r centered at a point p, where $S(p, r) \subseteq R^n$. A maximal disk in S is a closed disk $D^n(p, r)$ contained in S.

Property 1. If X is a maximal disk in S, then S is not properly contained in any other closed disk in S.

Definition 4. Let $S(p, r) \subseteq R^n$. The medial axis (MA) of S is the locus of the centers of the maximal disks contained in S. The medial axis of a 3D object denoted R^3 is sometimes called the *medial surface*. The continuous function of a real-value that assigns to each center of a maximal disk in S is called the *radius function* of that medial axis.

The medial axis together with the associated radius function of the maximal disks is called the *medial axis transform* (*MAT*). Then, the medial axis can be defined:

Definition 5. The medial axis transform (MAT) of an object consists of its medial axis together with the associated radius function.

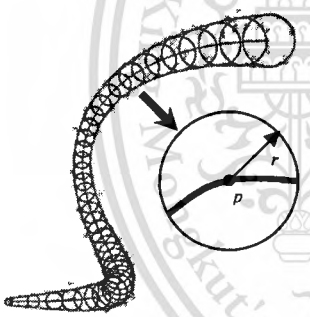


Figure 1. The medial axis of an object

4. SKELETON COMPUTATION

The following definitions are used to formally define the skeleton in our algorithm:

Definition 6. The *degree* of a point is defined as a finite number of points in its 26-neighborhood (figure 2 shows the 26-neighborhood structure).

Definition 7. By the assumption that the skeleton is one-voxel thick, the skeleton *end point* is a point that has a degree one. The *middle point* is a point that has a degree two. The *connection point* is a point that has a degree three or higher and all the other neighbors are either the end points or the middle points. The *branched connection point* is a point that has a degree three or higher and at least one of its neighbors is neither an end point nor a middle point. The 26-connected branched connection points form the *branched region*.

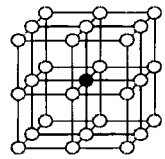


Figure 2. The 26-neighborhood structure

Algorithm: *Compute Skeleton*

Input: A set of points on the surface of model, $P = \{p_1, p_2, \ldots, p_n\}$.

Output: The skeleton of the given model.

step 1. Compute the degree of each point.

step 2. Determine which point are the end points, the middle points, the connection points, and the branched connection points. If there are no branched connection points, exit the program.

step 3. Organize the branched connection points into the branched regions. Each branched region consists of 26-adjacent branched connection points.

step 4. In each branched region, find all the end points and the middle points that are 26-adjacent to this branched region, and mark each as "terminator".

step 5. Determine the centroids, based on Blum's definition of a skeleton; the skeleton points must be the centers of the maximal disks contained in the curve S.

step 6. Compute the geodesic distance from the particular centroids to all points in step 4.

step 7. Apply the Dijkstra's algorithm, in order to find the shortest paths between the centroids and their "terminator".

step 8. Remove the branched connection points that are not on any of the shortest paths.

step 9. Generate the skeleton from the remaining points.

5. EXPERIMENTAL RESULTS

We tested the proposed algorithm on the Princeton Shape Benchmark database [10]. A brief comparison between the results of our algorithm and their original skeleton, extracted by using the Reeb graph is provided. The proposed algorithm can reduce the number of the branched connection points, which is similar to the number of critical points in the original Dijkstra's algorithm therefore the spurious skeleton joints can be removed. These joints will not influence the structure of the skeleton because they are not on any of the shortest paths computed by Dijkstra's algorithm. The skeleton can then be generated in the sense that it can capture the essential shape of a 3D object while preserving a compact form of its data structure. Figure 3 shows the experimental results.

Table 1 compares the number of points after applying the proposed algorithm with the number of points of the original 3D objects. Our algorithm can reduce the number of critical points in the graph's nodes, the branched connection

points in a skeleton graph. Therefore, the numbers of the spurious skeleton joints are reduced comparable to the number of unnecessary joints generated by the Reeb graph method.

Table 1: Comparison of the number of points between the proposed method and the original model

Mesh	Number of points	
	The Reeb Graph	The pruned skeleton
Cow	2904	2655
Rabbit	1238	1026
The Dragon	1166	1011
The Alien	429	303
The Femme	635	503
Boy	17342	16017

Let n be the number of points in the meshed model. The computational complexity of the proposed algorithm can be considered as follows:

(1) The computation of degree of each point and the point analysis takes $O(n)$ time.

(2) The computation of testing and finding the maximal disks in the neighborhood of each point takes $O(n)$ time.

(3) The computation of the single source shortest paths (in term of the geodesic distance from that source) to all points on the mesh takes $O(n \log n)$ time [4].

Thus, the overall computation takes $O(n \log n)$ time.

6. CONCLUDING REMARKS

In figure 3 (a) and (c), it is easy to observe that the skeleton extracted by using the Reeb graph is very sensitive to the connectivity of the boundary representation and it also has a problem in producing the unwanted skeleton joints. The proposed algorithm can overcome these drawbacks. The results show the meaningful characteristics of the 3D meshed model, and the generated skeletons whose joints can be associated with the 3D meshed model.

Given the skeleton construction process with the proposed algorithm of removing the unnecessary skeleton joints, an important challenge remaining is to find a method for generating the realistic joints that are similar to the joints of the model's anatomy.

6. REFERENCES

[1] M. K. Agoston. Computer graphics and geometric modeling: implementation and algorithms. Springer, ISBN 1852338180, page 185, 2005.

[2] S. Baloch, H. Krim, I. Kogan, and D. Zenkov. Rotation invariant topology coding for 2D and 3D objects using Morse theory. In *Proceedings of IEEE International Conference on Image Processing, 2005, ICIP 2005*, pages 796-799, Sep 2005.

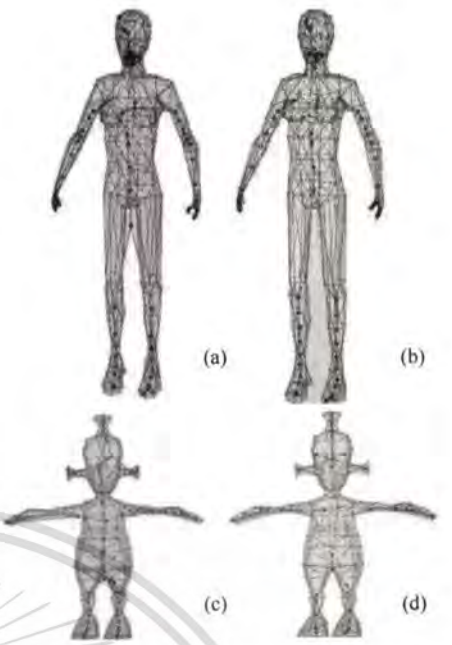


Figure 3. The skeletons from our algorithm (right) and their skeletons extracted by using the Reeb graph (left)

[3] H. Blum, R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition 10*, pages 167-180, 1978.

[4] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. In *Symposium on Computational Geometry*, pages 344-350, 2003.

[5] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *Computer Graphics Annual Conference Series (Proc. Of SIGGRAPH)*, pages 203–212, 2001.

[6] F. Lazarus, and A. Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, ACM Press, pages 130–140, 1999.

[7] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. of Computing 16(4)*, pages 647-668, 1987.

[8] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [On the singular points of a complete integral pfaff form or of a numerical function].*Comptes Rendus Acad.Science Paris 222*, pages 847–849, 1946.

[9] M. Sabry Hassouna and Aly A. Farag. On the extraction of curve skeletons using gradient vector flow. *Proc. of IEEE International Conference on Computer Vision* ICCV, Rio de Janeiro, Brazil, October 14-20, 2007.

[10] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, pages 167-178, 2004.

Knowledge-based approach for 3D skeleton extraction

Porawat Visutsak[1, 2]
faaspwv@ku.ac.th

Veera Boonjing[1]
kbveera@kmitl.ac.th

Korakot Prachumrak[1]
kpkorako@kmitl.ac.th

[1]Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang
Bangkok, THAILAND

[2]Faculty of Liberal Arts and Science, Kasetsart University, Kamphaeng Saen Campus
Nakhorn Pathom, THAILAND

ABSTRACT

A 3D skeleton is a one-voxel thick, graph-like structure, widely used in the area of a character animation. In this paper, we propose a novel method for the 3D skeleton extraction of the polyhedral models based on a priori knowledge of the learned skeleton. The method has several steps. First, the octree of the input model is calculated and used to compare with the octree of the 3D models stored in the octrees database. By comparing the octree similarities, if the searched results exactly match with the input octree, the corresponding skeleton will be retrieved from the skeletons database. Otherwise, the method finds the list of the close match (the octree similarity ratio which the value is greater than 0.8) in order to use to estimate the new skeleton. In the worst case, if the input octree does not match with any case in the octrees database, the method computes the new skeleton and stores it in the skeletons database. The method is fast and efficient because it is not necessary to extract the skeleton from every input model. Thus, the computational time of our method depends only on the time of the octree similarity calculation, and the time for searching the similar octree in the octrees database. Several examples show the results obtained with our approach.

Categories and Subject Descriptors

I.3.5 **[Computer Graphics]**: Computational Geometry and Object Modeling – *Curve, surface, solid, and object representations*

General Terms

Algorithms, Experimentation, Theory

Keywords

Skeleton; Skeleleton Extraction; Computer Animation; Octree.

ICIS 2009, November 24-26, 2009 Seoul, Korea

1. INTRODUCTION

A skeleton is an articulated figure composed of the joints and the segments linking them, used to define and drive the animation of a virtual figure such as a human, a creature, and a made-up monster. A skeleton of a 3D model can be either created manually by the professional artists or automatically calculated by a computer programming. In the case of the realistic animation of a character, the first technique seems to be the most popular option chosen by artists, even though it needs a skilled user and this is a time-consuming task. Actually, a skeleton mock-up can be initially created by artists relatively quickly, but often need to make many adjustments during the rigging process since the skin is very sensitive to the exact location of the skeleton's joints [1]. Therefore, they often have to go back and forth several times between skeleton skinning and testing animation before getting it right. The well-known methods have been developed to generate a skeleton in a graph-like or a curve-like form [5], [6], [9], [11], [12]. Unfortunately, these approaches do not suit for producing a skeleton for use in a character animation because they need the additional processes to eliminate the redundant skeleton branches that may be generated during the skeleton extraction process [15].

A recent work on the semi-automatic skeleton extraction and the fully automatic generation of a control skeleton are introduced by [1] and [17], respectively. In [1], the system allows the user select the starting point on the character model, then it generates the skeleton to match the ones that are created by hand by professionals in most biped and quadruped cases. In [17], the main task of the system involves discretizing the figure, computing its discrete medial surface (DMS), then using the discrete medial surface both to create the skeleton and to attach the vertices of the model to that structure. However, the both of [1] and [17] are often required the pre-process and the post-process methods; e.g., in [1] it needs the user to allocate the starting point of the skeleton and it often allows little control over the result, which always leads to computationally expensive algorithms.

In this paper, we present a novel method for extracting the 3D skeleton of the polyhedral models by using the knowledge of extracting the smoothed joints that have been introduced from the previous experiments [15], [16]. The proposed method has several steps. It first computes the octree of the input 3D model, in a fast and robust way (the octree is normalized by using the low resolution form of a 3D model). Then, the octree is used to

...

compare with the other octrees stored in the octrees database. By evaluating the value of the similarity ratio, if the input octree matches with the ones in the octrees database, then the corresponding skeleton will be retrieved from the skeletons database. If the input octree does not exactly match, but the value of the octree similarity ratio is greater than 0.8, the method computes the new skeleton by averaging the skeletons which the value of the similarity ratio ≥ 0.8. In the case of not match and the value of the octree similarity ratio is lower than 0.8, the method goes back to our fundamental step of extracting the smoothed joints of a skeleton that have been proposed [15], [16]. Our method is fast and efficient (that is to say, it is unnecessary to extract the skeleton from every input model). In the average case, the computational time of our method depends only on the time of the octree similarity calculation. Our method can match the octree of a model with several hundred of thousand faces in no longer than a few seconds on a low-end computer. We demonstrate the usability of our method with several input models. Finally, the conclusion and the evaluation of our approach are also provided.

2. RELATED WORK

In the following paragraph, we briefly summarize the main categories of the methods which are the precursor of this paper; the skeleton pruning method [15] and the skeleton smoothing method [16]. The first ones aim at eliminating the redundant branches of the skeleton. The second ones adjust the pruned skeleton and locate the smoothed joints for a model.

2.1 The Skeleton Pruning Method

Based on the Medial Axis Transform (MAT), the skeleton pruning method has been firstly presented in [2]. A skeleton in term of the medial axis of a 3D object can be defined as the locus of the centers of all spheres interior to the object that reach the boundary of the object. Together with the medial axis, [15] use the geodesic distance function as the distance measurement between each joint on the skeleton. Being invariant to a rotation and a translation is the major advantage of the geodesic distance function, thus, any changes of a rotation and/or a translation do not affect the value of a measured distance. The basic idea behind the skeleton pruning method is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region construction since the generation of redundant skeleton joints will seriously disturb the topology of the skeleton.

2.2 The Skeleton Smoothing Method

Once, the pruned skeleton has been originated with a one-voxel thick, in a graph-like form, the next problem is raised which is the meaningless points that align unreasonably along a skeleton. These points can affect on the later creation of segments and joints, therefore the pruned skeleton is subjected to a smoothing operation [16]. The smoothing process involves calculating the average position of consecutive points along the pruned skeleton, in order to adjust the skeleton to be smoothened as nearly as a curve. After the smoothed skeleton is formed, the next step is to determine which points should be split. Splitting of the skeleton is accomplished by considering the chord-to-point distance accumulation value. The chord-to-point distance accumulation is the distance measurement from the line to a point in the curve segment, which has been firstly presented in [3]. Unlike [3], the geodesic distance function has been used instead of the Euclidean distance. By using the Dijkstra's algorithm, the geodesic distance is computed in order to find the shortest path between two points on the skeleton path made up by n points [14]. The distance feature computed by [3] is invariant with respect to a rotation and a translation which is a major benefit for animating a skeleton. The methods of skeleton pruning and smoothing are shown in figure 1.

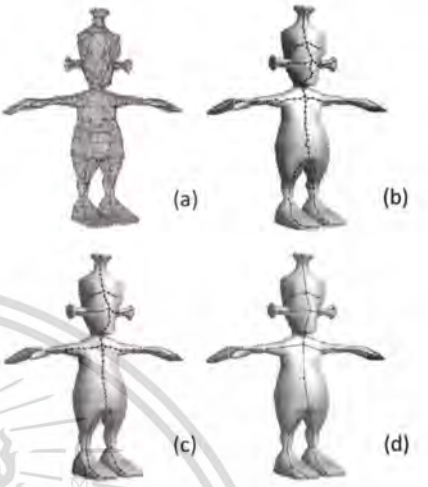


Figure 1. (a) The original 3D mesh model (b) The pruned skeleton (c) The smoothed skeleton (d) The control joints of the smoothed skeleton.

3. THE PROPOSED METHOD

As we mentioned earlier, we use the control joints of the smoothed skeleton which we already extracted and stored in the skeletons database as a priori knowledge to approximate the control joints of an unknown skeleton. In our study, we use the octree as the fundamental data structure of a 3D model, which is widely used in the application of the 3D object representation, comparison, and recognition. By comparing the input octree with the octrees stored in the octrees database, if the input octree match the ones in the octrees database, that means it is a known skeleton. Therefore, the method retrieves the control skeleton from the skeletons database, otherwise the method approximates the new skeleton from the skeleton list whose the value of the similarity ratio are greater than 0.8. In the worst case, if the input octree does not match with any case in the octrees database and the similarity ratio is below 0.8, the new skeleton will be computed by using our precursor approaches [15], [16]. In this section, we examine the octree similarity technique and discuss the major advantages of this technique to overcome an orientation and a scaling problem for comparing the 3D objects. Finally, we illustrate our method for extracting the control skeleton based on a priori knowledge of the skeleton pruning and smoothing techniques.

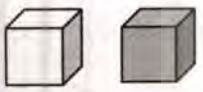
3.1 The Octree Similarity

The octree is a hierarchical volume description of the 3D objects, widely used in the field of computer graphics and image processing. In [4], [7], and [10] have been introduced the usage of

the octree in the applications of a shape analysis and an object recognition. Here, we use the octree similarity technique for finding a similar 3D object in the octrees database. Unfortunately, every object has a different spatial orientation. They might be flipped, rotated, translated, and even scaled up or down. To deal with these problems, we firstly define the similarity ratio between two 3D objects that is based only on the surfaces of the objects and their geometric attributes, then we apply a bounding box for capturing the object. Using the relative coordinates of a bounding box as the object reference can guarantee that our method is invariant to a rotation and a translation, and it is also robust against the scaling of a 3D object.

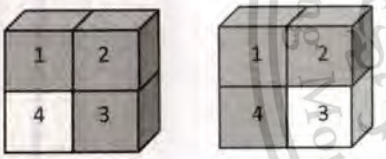
3.1.1 The similarity ratio between two 3D objects

The similarity ratio between two 3D objects is defined to be a value between 0 to 1, denoted by S: [0, 1]. To calculate the similarity ratio, we use the opposite term called the differences ratio between two 3D objects that is also a value between 0 to 1, denoted by D: [0, 1]. Thus, $S = (1\text{-}D)*P$, where P is the scalar multiplication between the two object's boxes diagonal vectors. The diagonal vector is a vector from a bounding box's centroid to one of its corners. Figure 2 (a) and (b) illustrate the similarity ratio of a full box and a partial box respectively.



(a)

A full box (gray) and an empty box (white): The space that is occupied in one box and not occupied in the other is 100%. Thus, $D = 1$, $P = 1$. Therefore $S = (1\text{-}D)*P = 0$.



(b)

Two partial full boxes: Quadrants 1 and 2 are occupied in both boxes, thus the differences ratio for them is 0. Quadrant 3 is occupied in one box and not in the other, thus the differences ratio for it is 0.25, and this is also the same as quadrant 4. Therefore $D = 0.25 + 0.25 = 0.5$. $S = (1\text{-}D)*P = 0.5$.

Figure 2. (a) The full and empty box (b) The two partial full boxes.

3.1.2 The bounding box

In order to overcome a rotation, a translation, and a scaling problem of a 3D object. We use a bounding box to capture the object into it. The bounding box's dimensions and spatial orientation is calculated according to the geometry of the 3D object's surface. The bounding box size is the minimal box size that can capture the object. The box's height, width, and length are each considered as one length unit. Then, all references to the objects are in the bounding box's coordinates. Thus, the relative coordinates are used instead of using the absolute coordinates, therefore we can gain the properties of being invariant to a rotation and a translation as well as a scaling problem of a 3D object. Figure 3 shows the usage of a bounding box.

As you can see from figure 3 (a), two identical objects in an identical spatial orientation, will both have identical bounding boxes. If one object is translated N units along the X axis with respect to the second object, this resolves in a translation of the first object's bounding box N units along the X axis. Since all references to the objects are done in the bounding boxes – relative coordinates, the two objects still have identical relative coordinates. E.g., if an object is scaled (stretched) to twice its original length (Figure 3 (b) and (c)). This causes its bounding box to stretch to twice original length, and the front end of the object is still in bounding box – relative coordinate '1' and the "middle" of the object is still in relative coordinate '0.5'. The same principle is applied when rotating an object around any axis.

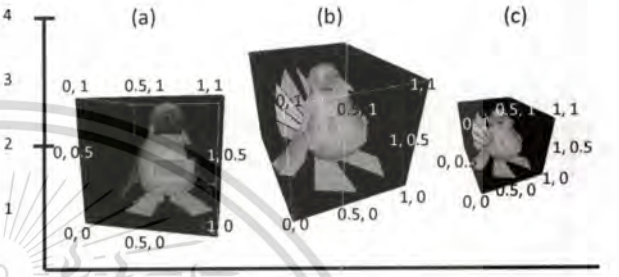


Figure 3. The usage of the bounding boxes.

3.2 Algorithm

Once the input octree is identified, our method evaluates the similarity ratio of the input octree in order to find the similar ones in the octrees database. The evaluation results can be fallen into three cases. The first one is the case that the value of the similarity ratio equals to 1.0; our method retrieves the corresponding skeleton from the skeletons database. Next, if 1.0 > the similarity ratio ≥ 0.8; our method approximates the new skeleton from the similarity list of the control skeletons from the skeletons database. Finally, if the value of the similar ratio is lower than 0.8; the method goes back to consult the precursor of this work (the skeleton pruning and smoothing methods) in order to compute the new skeleton. When the new skeleton is calculated, the method stores the result skeleton into the skeletons database and it also stores the octree into the octrees database for a later use. Our algorithm is illustrated in figure 7.

4. RESULTS

We perform our method to search for an object from among all 256 models of bipeds and quadrupeds. Our proposed method is tested on the Princton Shape Benchmark database [13] and a standard handmade model (from Autodesk's Maya software). The search results are shown in figure 4. In conducting the search, one model is selected from the 256 models, the octree similarities between it and the other models are calculated and the searched result is shown with the value of the similarity ratio equals to 1.0 (figure 4 (a) and (b)). As you can see from our experiment, the searched result matches correctly with the input model even if the input model is rotated and translated with respect to the original. The other experimental results are shown in figure 4 (c) to (f).

Notice that only the searched objects which have the value of the similarity ratio ≥ 0.8 are shown.

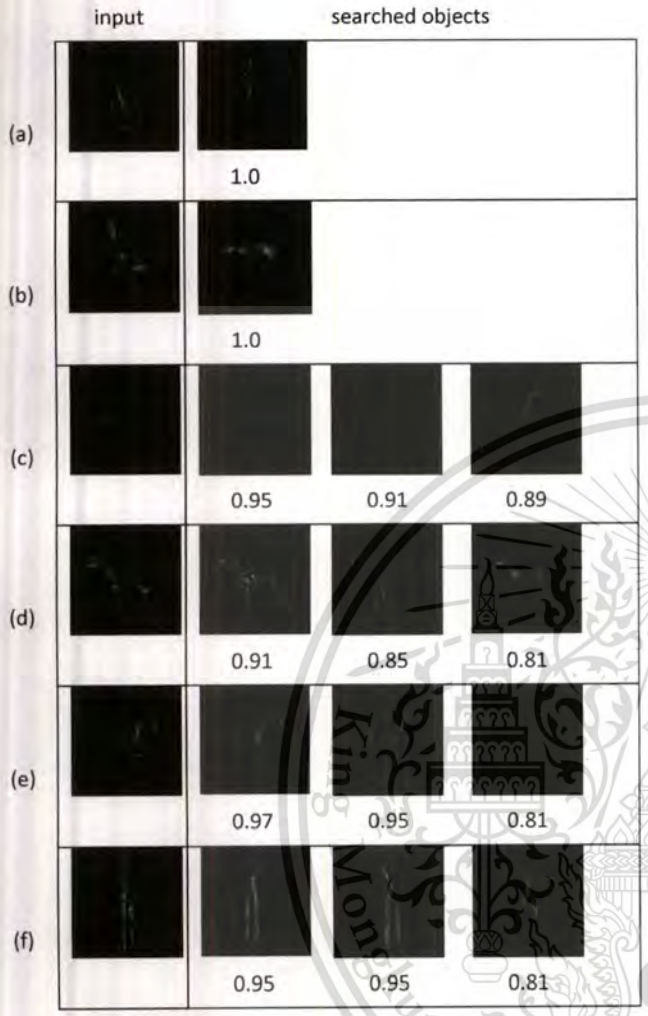


Figure 4. Results of the search experiment.

It can be shown [8] that the average execution time of the octree computation is given by $O(H)$, where H is the height of octree. Thus, the time for searching M most similar octree in the N octrees database can be given by $N*M\,[O(H)]$ (assuming that $M < \log N$; since the execution time of the searching algorithm for N objects is $\log N$, therefore M is always $< \log N$). In figure 5, the graphical user interface of the control skeleton extraction (the skeleton pruning and the skeleton smoothing) is shown, by clicking over a segment will cause the corresponding label to be printed on the screen. The control joints are shown in a hierarchical view.

5. CONCLUSIONS

In this paper, we have proposed our approach for extracting the control skeleton based on a priori knowledge of our previous experiments called the skeleton pruning and the skeleton smoothing methods. By using the octree comparisons, we can reduce the computational time of extracting the redundant control skeletons, since our method will first finds the most similar objects whose theirs control skeletons have been extracted and stored in the skeletons database. As we mentioned in the previous section, our time complexity of our algorithm depends only on the execution time of the octree calculation and the time of searching the similar octree in the octrees database. By using the similarity ratio together with the bounding box for capturing the 3D object, we can gain the major benefit of being invariant to a rotation and a translation, as well as the robustness against the scaling of a 3D object.

With respect to the current implementation, significant improvements are desirable, either by increasing algorithm robustness or speed. Another robustness, we are taking into account is the robustness against the deformation of a model. E.g., in order to perform the character animation, we need to deform the model into the motion tween as shown in figure 6. Our approach fails to recognize figure 6 (b) and (c), even though they are deformed from figure 6 (a) which is the original model of them. We envision and encourage the extension of the robustness against the deformation of a model to the motion tween.

Figure 5. The GUI of control skeleton extraction.

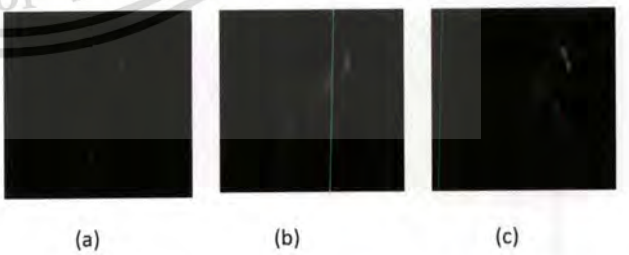


Figure 6. The original model and its deformations.

6. REFERENCES

[1] G. Aujay, F. Hetroy, F. Lazarus, and C. Depraz. Harmonic skeleton for realistic character animation. In *Symposium on Computer Animation (ACM SIGGRAPH)*, 2007.

[2] H. Blum, R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition 10*, pages 167-180, 1978.

[3] J.H. Han and T. Poston, Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature. *Pattern Recognition Letters*, vol. 22, pp. 1133-1144, 2001.

[4] B. B. Chaudhuri. Applications of quadtree, octree, and binary tree decomposition techniques to shape analysis and pattern recognition. *Transactions on Pattern Analysis and Machine Intelligence*, pages 652-661, 1985.

[5] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *Computer Graphics Annual Conference Series (Proc. Of SIGGRAPH)*, pages 203–212, 2001.

[6] F. Lazarus, and A. Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, ACM Press, pages 130–140, 1999.

[7] Y. J. Lee and Y. T. Park. Occluded 3D object recognition using partial shape and octree model. In *Proceedings of International Conference on Intelligent Computing*, pages 839-848, 2005.

[8] P. Minovic, S. Ishikawa, and K. Kato. Three-dimensional sysmmetry identification. *Memoirs Kyushu Inst. Tech. (Eng.)*, no. 21, pages 1-38, 1992.

[9] T. Oda, Y. Itoh, W. Nakai, K. Nomura, Y. Kitamura, and F. Kishino. Interactive skeleton extraction using geodesic distance. In *Proceeding of the 16th International Conference on Artificial Reality and Telexistence*, 2006.

[10] G. K. Rambally and R. S. Rambally. Octrees and their applications in image processing. In *Proceeding of IEEE Southeastcon*, pages 1116-1120, 1990.

[11] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [On the singular points of a complete integral pfaff form or of a numerical function]. *Comptes Rendus Acad.Science Paris 222*, pages 847–849, 1946.

[12] M. Sabry Hassouna and Aly A. Farag. On the extraction of curve skeletons using gradient vector flow. *Proc. of IEEE International Conference on Computer Vision ICCV*, Rio de Janeiro, Brazil, October 14-20, 2007.

[13] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In Shape Modeling International, pages 167-178, 2004.

[14] O. Symonova. Shape description for 3D model retrieval. Technical Report #DISI-08-010, University of Trento, Mar 2008.

[15] P. Visutsak and K. Prachumrak. The 3D skeleton pruning for removing undesired joints. In *Proceedings of The Second IEEE International Conference on Computer, Control and Communication IEEE-IC4 2009*, Karachi, Pakistan, February 17-18, 2009.

[16] P. Visutsak and K. Prachumrak. The smoothed 3D skeleton for animation. In *Proceedings of the 5th International Conference on INC, IDC, IMS*, Seoul, Korea, August 25-27, 2009.

[17] L. Wade and R.E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer 18*, 2002.

...

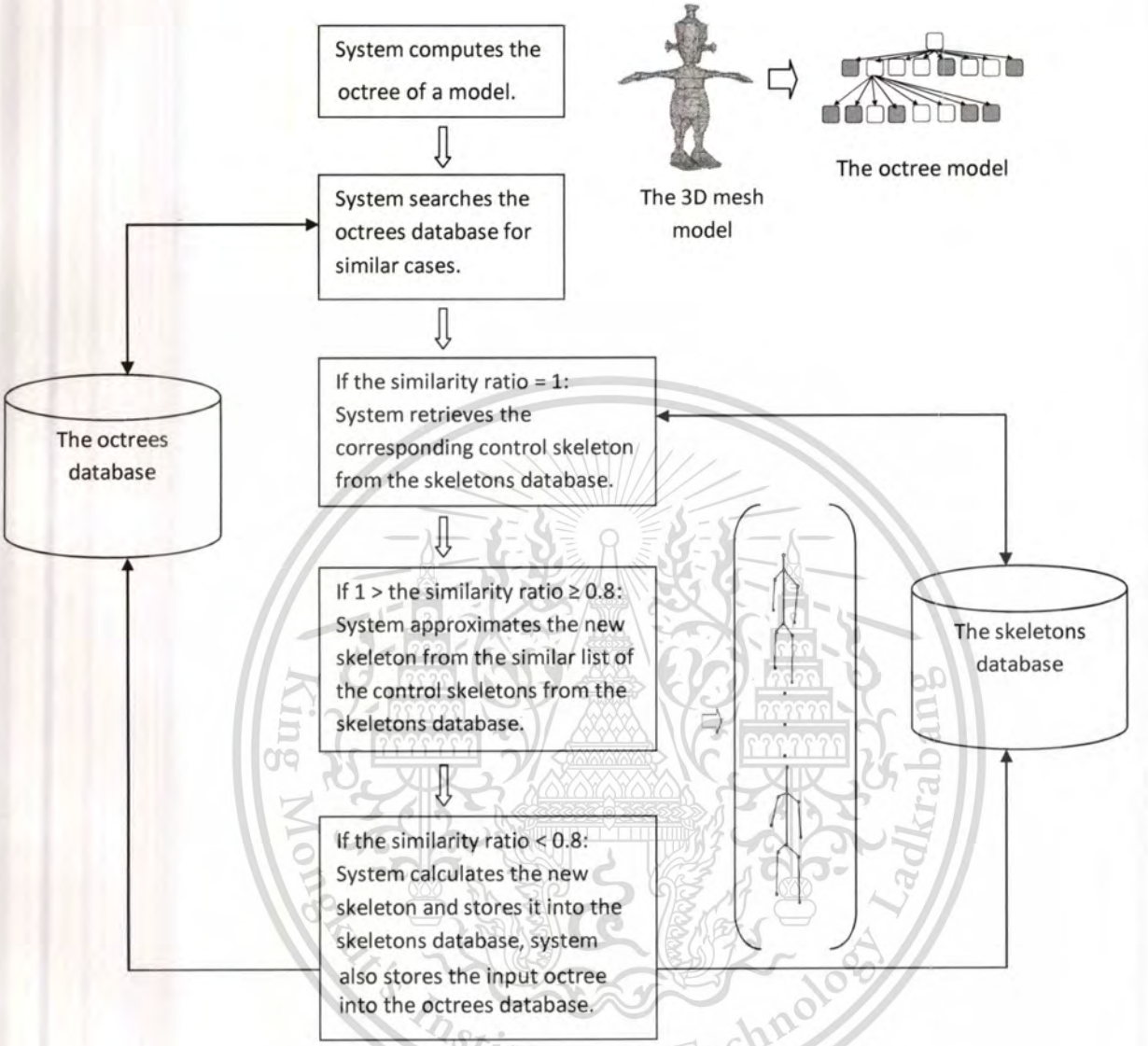


Figure 7. The proposed method.

...

2009 Fifth International Joint Conference on INC, IMS and IDC

The Smoothed 3D Skeleton for Animation

Porawat Visutsak, Korakot Prachumrak

Department of Mathematics and Computer Science, Faculty of Science,
King Mongkut's Institute of Technology Ladkrabang, Chalongkrung Rd., Ladkrabang,
Bangkok 10520 THAILAND
{s0067152, kpkorako}@kmitl.ac.th

Abstract—**Skeleton is at the main interest of 3D character animation. In this paper, we use a recent skeleton pruning method based on the medial axis transform and the geodesic distance function that originates one-voxel thick, graph-like skeleton. Unfortunately, the location of skeleton joints does not usually match the anatomical joint of the model. We introduce a novel method for smoothing the pruned skeleton for realistic character animation by using the filtering process for adjusting the locations of consecutive points along the skeleton. Chord-to-point distance accumulation then be applied, and the smoothed skeleton is split in order to create segments and joints corresponding to its shape. The new skeleton can be regenerated later on. Therefore, the new skeleton produced from the proposed method can capture the essential shape characteristics in a compact form, while preserving the meaningful anatomical information of the 3D character models. The demonstration of the approach with several examples is also provided.**

Keywords: Skeleton; Skeleton smoothing; Skeleton pruning; Chord-to-point distance accumulation

1) Introduction

The skeleton is a basic structure of representing the 3D objects, frequently used in computer graphics, especially in the areas of character animation and 3D modeling. Using a skeleton as an abstraction of an object has two major benefits. First, it can contain both shape features and topological structures of an original object. Another benefit depends on its characteristic to capture the essential shape of a 3D object in a low-dimension form. Numerous algorithms have been developed to generate the skeleton in graph-like or a curve-like form [4], [5], [6], [7], [8]. Unfortunately, these approaches do not suit for producing a skeleton for use in animation since they need the additional processes to eliminate the redundant skeleton branches that may generate during the skeleton extraction process.

To apply skeleton for use in character animation and 3D modeling, skeleton animation is a common technique for animating a 3D model. Controlling the movement of a skeleton in a way that is designed to appear naturally is accomplished using a control skeleton (sometimes called an Inverse Kinematics or IK skeleton). IK skeleton is an articulated structure of segments and joints combined with information detailing how the surface geometry of the figure is anchored to that structure. Recent work on semi-automatic skeleton extraction is introduced by Aujay *et al.* [1]. This system allows users select the starting point on the character model, and then it generates a skeleton to match the ones that are created by hand by professionals in most biped and quadruped cases. A method for fully automatic generation of a control skeleton is proposed by Wade and Parent [12]. The main task of the system involves discretizing the figure, computing its discrete medial surface (DMS), and then using the discrete medial surface both to create the skeleton and to attach the vertices of the model to that structure. However, a major drawback of Wade and Parent's method is only features with a size greater than the voxel size can be taken into account. This often leads to computationally expensive algorithms.

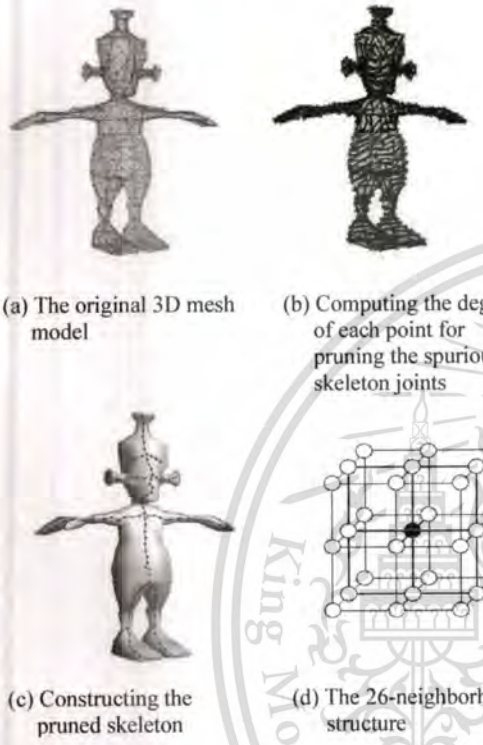
In this paper, we present a novel method for smoothing the pruned skeleton for realistic character animation. We use the skeleton pruning method introduced by Visutsak and Prachumrak [11] as a fundamental process to generate the pruned skeleton. Unfortunately, this raises the problem of producing meaningless joints since the location of skeleton joints does not match the real bone structure of the model. Thus, the additional operation is needed, by using the filtering process we can obtain the smoothed skeleton. Chord-to-point distance accumulation [3] then be applied in order to split the smoothed skeleton into the segments, and then the new joints are located correspond to chord-to-point distance accumulation values. We demonstrate the usability of the smoothed 3D skeleton with several figures. Finally, the conclusion and the evaluation of our approach are also provided.

2) The Skeleton Pruning Method

Visutsak and Prachumrak [11], in a precursor of this paper, used the skeleton pruning method based on the medial axis (MA) as a skeletonization algorithm. A skeleton in terms of the medial axis is a set of curves that roughly run along the middle of an object. According to Blum and Nagel [2], the medial axis of a 3D object can be defined as the locus of the centers of all spheres interior to the object that reach the boundary of the object. Together with the medial axis, Visutsak and Prachumrak [11] use the geodesic distance function as the distance measurement between each joint on the skeleton. Being invariant to a

978-0-7695-3769-6/09 $26.00 © 2009 IEEE
DOI 10.1109/NCM.2009.36

rotation and a translation is the major advantage of the geodesic distance function, thus, any changes of a rotation and/or a translation do not affect the value of a measured distance. The basic idea of the skeleton pruning method is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region construction since the generation of redundant skeleton joints will seriously disturb the topology of the skeleton.



(a) The original 3D mesh model

(b) Computing the degree of each point for pruning the spurious skeleton joints

(c) Constructing the pruned skeleton

(d) The 26-neighborhood structure

Figure 1. The skeleton pruning method.

Figure 1 shows an overview of the skeleton pruning method. The first step of the method is to compute the degree of each point from the 26-neighborhood structure, and determine which point are the end points, the middle points, the connection points, and the branched connection points. By the assumption that the pruned skeleton is one-voxel thick, the skeleton end point is a point that has a degree one. The middle point is a point that has a degree two. The connection point is a point that has a degree three or higher and all the other neighbors are either the end points or the middle points. The branched connection point is a point that has a degree three or higher and at least one of its neighbors is neither an end point nor a middle point. The 26-connected branched connection points will form the branched region. The second step is to determine the centroids based on Blum and Nagel's definition of a skeleton, and find the terminator points in each branched region (the terminator points are all the end points and the middle points that are 26-adjacent to the branched region). The next step is the computation of the geodesic distance from the particular centroids to all the terminator points. The Dijkstra's algorithm is applied in this step in order to find the shortest paths between the centroids and the terminator points. The branched connection points that are not on any of the shortest paths have been removed, and the pruned skeleton can be generated from the remaining points.

3) SMOOTHING THE SKELETON

Although the skeleton pruning method can eliminate a large number of spurious skeleton joints, the next problem is raised which is the meaningless points that align unreasonably along a skeleton. These points can affect on the later creation of segments and joints, therefore the pruned skeleton is subjected to a smoothing operation. The smoothing process involves calculating the average position of consecutive points along the pruned skeleton, reassigning the new joints by splitting a skeleton into the segments.

3.1 Adjusting the skeleton

Once, the pruned skeleton has been originated with a one-voxel thick, in a graph-like form as shown in figure 1(c). Before we apply chord-to-point distance accumulation to the pruned skeleton, the skeleton needs to be smoothened as nearly as a curve. This can be accomplished by using the average filtering over a sliding window consisting of five consecutive points to adjust the positions of consecutive points along the skeleton.

Let $C = \{p_i = (x_i, y_i, z_i),\ i = 1,\ldots, N\}$ be the set of coordinate of points on the skeleton. The ith point in the set is denoted by p_i and p_{i+1} is its neighboring point. To calculate the average at the point p_i, we make an approximation of the change in the positions of two consecutive points before p_i and two consecutive points after p_i and divide it by 5 (the number of points between $p_{i-2}, \ldots, p_i, \ldots, p_{i+2}$). Thus, the average position (v_i) over the point p_i can be defined as equation (1). The filtering process is illustrated in figure 2(a).

$$v_i = \frac{\sum_{i-2}^{i+2} p_i}{5} \tag{1}$$

3.2 Locating the Smoothed Joints

After the smoothed skeleton is formed, the next step is to determine which points should be split. Splitting of the skeleton is accomplished by considering chord-to-point distance accumulation value. We applied chord-to-point distance accumulation proposed by Han and Poston [3] to our method. Unlike Han and Poston's method, we use the geodesic distance instead of the Euclidean distance. The

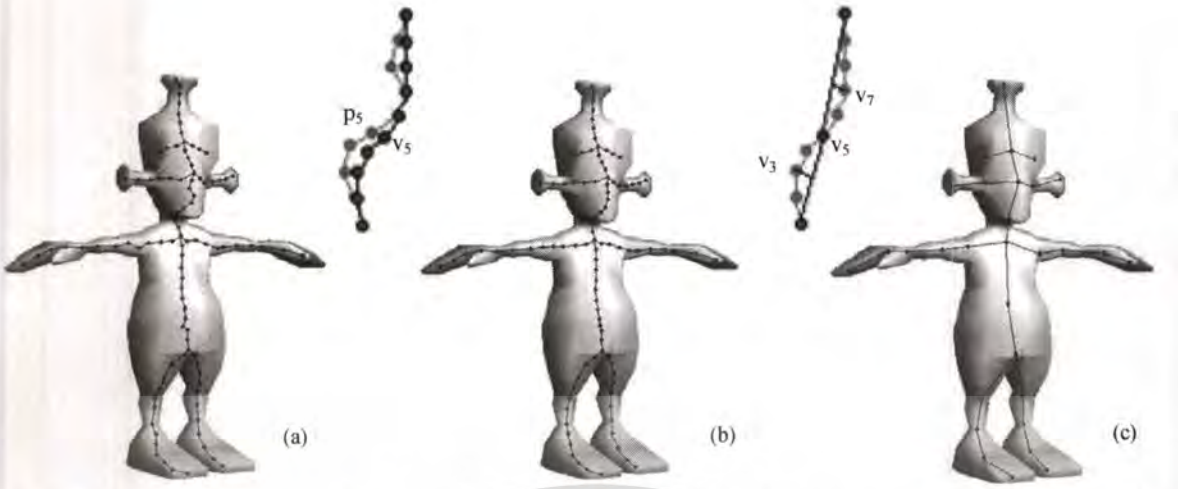


Figure 2. The method for smoothing the pruned skeleton. (a) The average position v_5 over $p_5 = (p_3+p_4+p_5+p_6+p_7)/5$, in order to obtain the smoothed skeleton, the average position v_i is computed for each p_i on the pruned skeleton. (b) The result of applying the smoothing operation to the pruned skeleton from figure 2 (a), Once the smoothed skeleton is formed, chord-to-point distance accumulation is applied for splitting the smoothed skeleton into segments in order to form bones and joints. Notice that the best split is formed at v_5 since the value of chord-to-point distance accumulation at v_3 and v_7 are equal. (c) The new joints are formed as a result of the best splitting based on chord-to-point distance accumulation.

geodesic distance is the distance between two points on the surface of the model, computed by using the Dijkstra's algorithm to find the shortest path between the two points on the surface made up by n points [10].

The distance feature computed by Han and Poston's method is invariant with respect to rotation and translation which is a major benefit for animating a skeleton. Chord-to-point distance accumulation is the distance measurement from the line to a point in the curve segment. The interpretation of chord-to-point distance accumulation can be defined as follows.

Let L be a fixed integer value defines a line L_i from each point p_i to p_{i+L}, where $i+L$ is taken modulo N. The perpendicular distance D_{ik} is computed from L_i to the point p_k. The distance is positive if p_k is on the left-hand side of the vector $(p_{i+L}- p_i)$, negative otherwise. Chord-to-point distance accumulation for a point p_k and a chord length L is the sum h_L of the D_{ik} as i moves from k-L to k. That is,

$$h_L(k) = \sum_{i=k-L}^{k} D_{ik} \qquad (2)$$

Since $D_{kk} = 0$, Thus,

$$h_L(k) = \sum_{i=k-L}^{k} D_{ik} = \sum_{i=k-L+1}^{k-1} D_{ik}$$

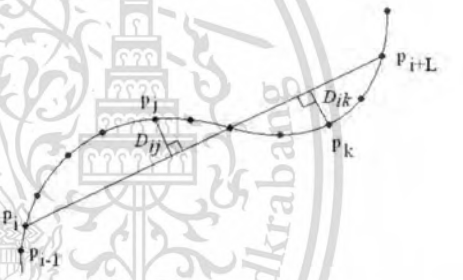


Figure 3. Chord-to-point distance accumulation.

Figure 3 illustrates chord-to-point distance accumulation, and shows an example in the case of D_{ij} is positive and D_{ik} is negative.

The basic idea of locating the new joints depends on chord-to-point distance accumulation value. After the smoothed skeleton is formed, the points that align along the smoothed skeleton can be classified into three classes. The end point is a point that has only one adjacent neighbor. The junction point is a point that has three or more adjacent neighbors. The intermediate point is a point that has exactly two adjacent neighbors. The end points and the junction points split the smoothed skeleton into a set of connected segment or bones. We use the end points and the junction points to form the initial joints of the skeleton. In each segment of the skeleton (see figure 2 (b)), we iteratively select the point from v_{i+1} to v_{N-1} and compute chord-to-point distance accumulation correspond to each selected

point. The selected point becomes the splitting point if $\|D_{ij}\| = \|D_{ik}\|$ e.g. v_5 is the splitting point since chord-to-point distance accumulation value at v_3 equals to chord-to-point distance accumulation value at v_7. Then we can form two bones from the corresponding segment, the first bone whose joints are located on v_1 and v_5, another bone whose joints are located on v_5 and v_{10} respectively. In the case of more than one splitting point is created, the sub segment will be formed (this sub segment becomes a bone later on), and more joints may be created from the corresponding sub segment. Figure 2 (c) shows the final result of the smoothed skeleton generated from our method.

Pseudo-code for smoothing the skeleton is shown below.

```
//Given skeleton V consists of N points places in a
queue Q.
Smoothen(V, N)
  If N ≤ 0
    Return V
  Repeat
    Q ← V
    N ← 1
    Repeat while Q ≠ ϕ
      v ← POP(Q)
      Compute h_L(k)
      If ||D_ij|| = ||D_ik||
        V ← V-v
        N ← N + 1
  Return v
```

Figure 4. Pseudo-code for iterative smoothing.

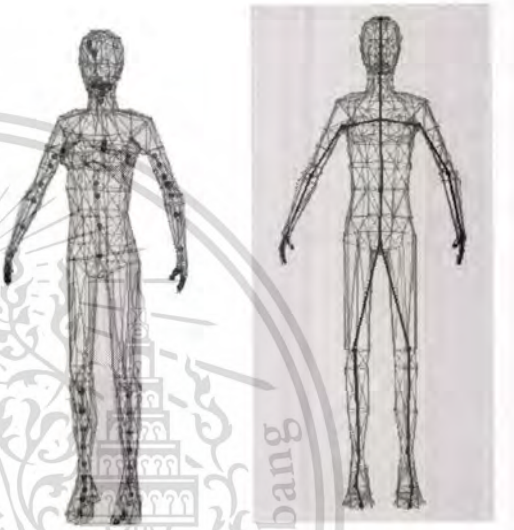
4) RESULTS

We demonstrate our smoothing method on several pruned skeletons generated from [11], as shown in figure 6. The proposed method is tested on the Princton Shape Benchmark database [9], and a standard handmade model (from Autodesk's Maya software). A brief comparison between the results of the proposed method and their original pruned skeletons is provided. The proposed method can produce a useful control skeleton, the number of control segments can be reduced in the sense that it is still enough for use in animation. Most of the control skeletons produced from our method are centralized, and run along to the ends of the main branches of the objects.

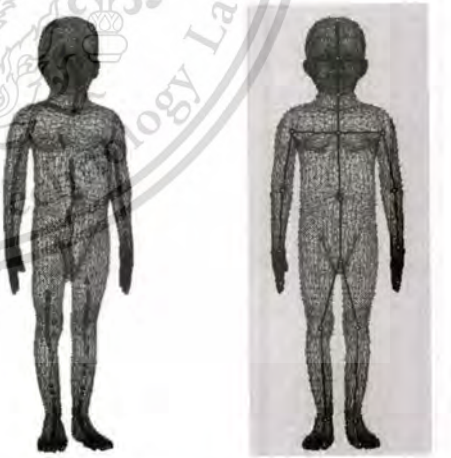
Table 1 shows the results of the number of control segments after applying the smoothing method with the number of control segments of the original pruned skeletons. Figure 5 shows several objects of a human-like model as smoothed by our algorithm (right). Segments and joints relate fairly well to surface features; however there is room for improvement. For instance for the boy-M149, there is no control points for fingers or toes are produced.

Object	Number of vertexes	Number of faces	Number of control segments	
			The pruned skeleton	The smoothed skeleton
M149-boy	16017	32007	860	19
M232-alien	732	1436	40	30
Femme	503	1002	50	26
Horse	535	1058	56	27
Dinopet	2039	3999	120	31

Table 1. Comparison of the number of control segments between the smoothed skeleton and the pruned skeleton.



(a) Femme – 50 control segments (left), and 26 control segments (right).



(b) M149-boy – 860 control segments (left), and 19 control segments (right).

Figure 5. The smoothed skeleton (right), and their pruned skeleton (left).

The computational cost of our algorithm can be derived below.

Let n be the number of points in the pruned skeleton. The time complexity of our algorithm can be considered as follows:

1) The computational complexity of averaging position of v_i over the point p_i in the filtering process takes $O(n)$ steps.

2) The computational complexity of chord-to-point distance accumulation takes $O(n)$ steps.

3) We implement our method to place n points in a queue data structure, and most of the times are spent on the *POP* operation, thus the time complexity of this process is $O(\log n)$ steps.

Thus, the overall computational complexity of our method takes $O(n)$ steps (this is not include the complexity analysis of the skeleton pruning method which takes $O(n \log n)$ steps).

5) CONCLUDING REMARKS

In this paper we have presented our approach for smoothing the pruned skeleton joints from the output of a medial-axis based skeletonization algorithm proposed by Visutsak and Prachumrak [11]. Then the collection of points produced from our smoothing algorithm is transformed into a collection of bones which can be used for 3D animation. The main process of our approach is based on chord-to-point distance accumulation introduced by Han and Poston [3]. Unlike Han and Poston's method, we use the geodesic distance instead of the Euclidean distance. We gain a benefit of the invariance against a rotation and a translation from the geodesic distance used in chord-to-point distance accumulation to construct the rotation and the translation invariant control skeleton. The algorithm produces the useful control skeletons with a small number of control segments. It suits for producing skeletons for more complex objects (the object which have a large number of vertex, and faces), since it can reduce the number of control segments comparable to the number of unnecessary segments generated by the skeleton pruning method.

6) REFERENCES

[1] G. Aujay, F. Hetroy, F. Lazarus, and C. Depraz. Harmonic skeleton for realistic character animation. In *Symposium on Computer Animation (ACM SIGGRAPH)*, 2007.

[2] H. Blum and R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition 10*, pages 167-180, 1978.

[3] J.H. Han and T. Poston, Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature. *Pattern Recognition Letters*, vol. 22, pp. 1133-1144, 2001.

[4] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *Computer Graphics Annual Conference Series (Proc. Of SIGGRAPH)*, pages 203–212, 2001.

[5] F. Lazarus, and A. Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, ACM Press, pages 130–140, 1999.

[6] T. Oda, Y. Itoh, W. Nakai, K. Nomura, Y. Kitamura, and F. Kishino. Interactive skeleton extraction using geodesic distance. In *Proceeding of the 16th International Conference on Artificial Reality and Telexistence*, 2006.

[7] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [On the singular points of a complete integral pfaff form or of a numerical function]. *Comptes Rendus Acad.Science Paris 222*, pages 847–849, 1946.

[8] M. Sabry Hassouna and Aly A. Farag. On the extraction of curve skeletons using gradient vector flow. *Proc. of IEEE International Conference on Computer Vision ICCV*, Rio de Janeiro, Brazil, October 14-20, 2007.

[9] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In Shape Modeling International, pages 167-178, 2004.

[10] O. Symonova. Shape description for 3D model retrieval. Technical Report #DISI-08-010, University of Trento, Mar 2008.

[11] P. Visutsak and K. Prachumrak. The 3D skeleton pruning for removing undesired joints. In *Proceedings of The Second IEEE International Conference on Computer, Control and Communication IEEE-IC4 2009*, Karachi, Pakistan, February 17-18, 2009.

[12] L. Wade and R.E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer 18*, 2002.

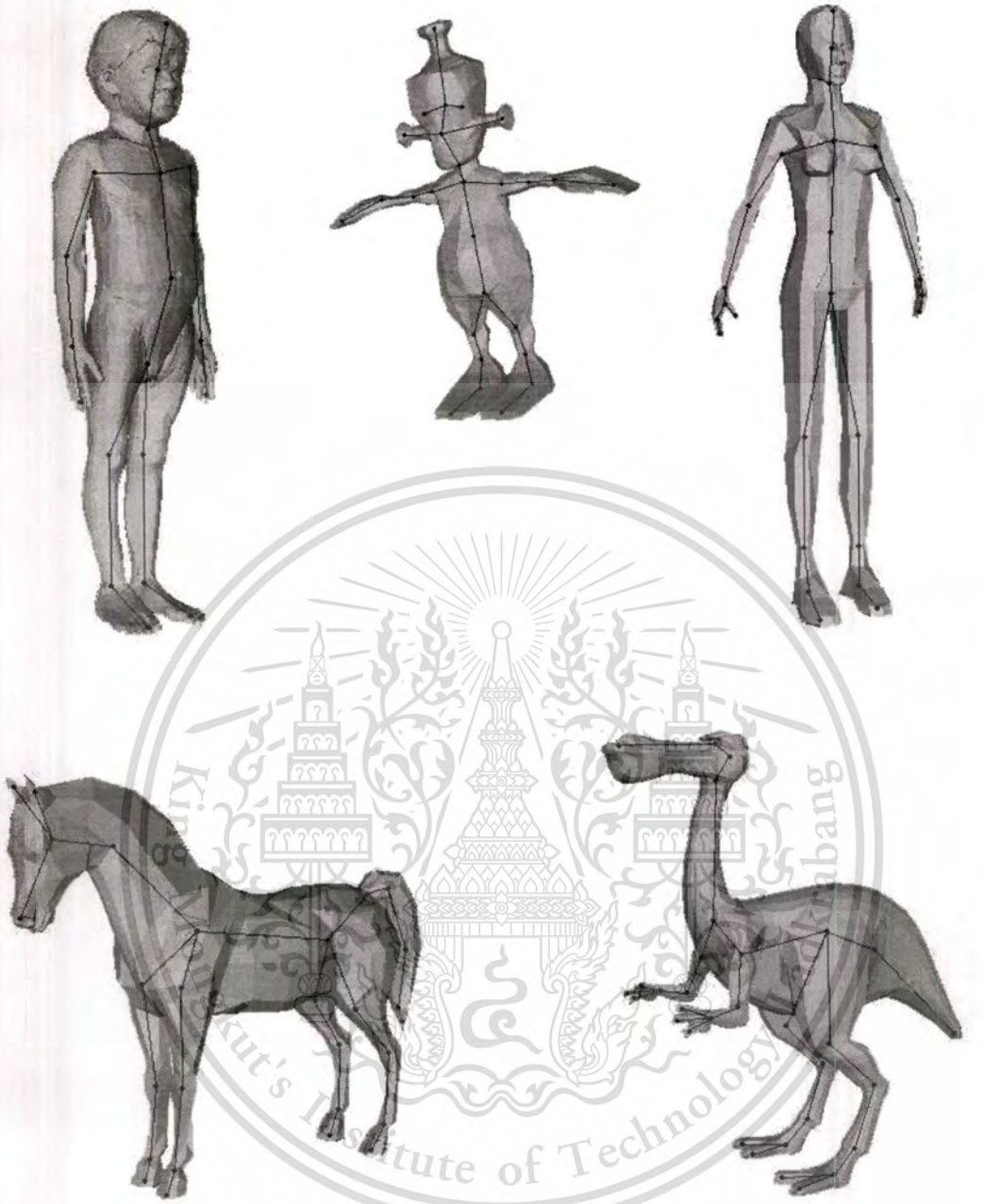
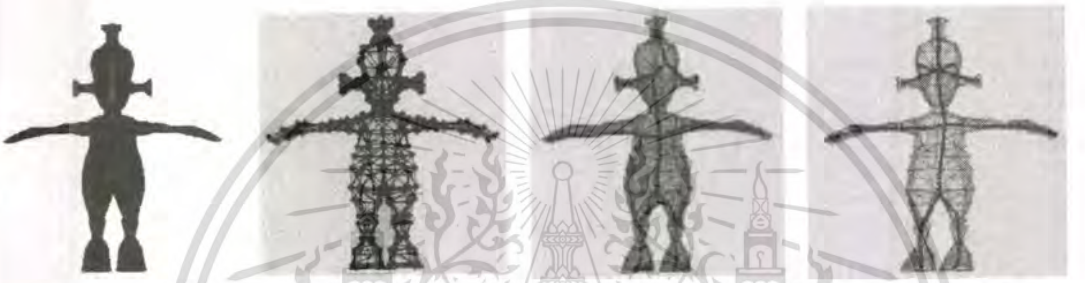


Figure 6. The smoothed skeleton of the objects: M149, M232, Femme, Horse, and Dinopet respectively.

The 3D Skeleton Pruning for Removing Undesired Joints

Porawat VISUTSAK, Korakot PRACHUMRAK

Department of Mathematics and Computer Science, Faculty of Science,
King Mongkut's Institute of Technology Ladkrabang, Chalongkrung Rd., Ladkrabang,
Bangkok 10520 THAILAND,
Tel. +66 (0)2326 4339-53 ext. 209, Fax. +66 (0)2326 4354
{s0067152, kpkorako}@kmitl.ac.th



(a) The original 3D meshed model (b) Computing the degree of points (c) Pruning the spurious skeleton points (d) Constructing the pruned skeleton

Figure 1. The skeleton pruning method

Abstract

The skeleton is a low-dimension shape representation of 3D objects useful for different areas, such as machine vision, image processing, computer graphics, character animation, and etc. The most common techniques for skeleton computing are based on the Reeb graph and the shortest path finding. Using only the shortest path algorithms for extracting the critical points and constructing the Reeb graph over the surface of the model may generate unwanted skeleton joints. A new skeleton pruning method for removing the spurious skeleton joints is introduced that is based on the medial axis transform and the geodesic distance function. Consequently, the generated skeleton can capture in a simple and meaningful way the essential shape of the 3D objects in a compact form.

Keywords: Skeleton; Skeleton pruning; Geodesic distance; Medial axis transform

1) INTRODUCTION

A new method for generating a skeleton is presented, based on the medial axis transform and the geodesic distance function associated with a 3D meshed object. The skeleton, generated by the new method captures the overall shape as well as the topology of an object. The skeleton is a basic structure of representing the 3D objects, frequently used in computer graphics, especially in the areas of character animation and 3D modeling. There are many approaches to represent 3D models in graph-like or a curve-like form as a fundamental data structure of the shapes the current method aims at reducing the 3D objects into a lower dimensional form that captures their shape characteristics (topology and/or geometry). The most common technique to represent the 3D shapes, that has been the standard for many years, is the Reeb graph [7], originally defined by Reeb.

The Reeb graph is obtained by applying the continuous function, usually a height function, to encode the topological structure. Using a height function to build a Reeb graph does not guarantee that the graph is invariant to the affine transformations, which is an essential feature of the skeletal structure of the model [8]. To overcome this drawback, the height function is replaced with the geodesic distance function [6]. By providing the invariance against a rotation and a translation [8], the geodesic distance function allows to construct the rotation and the translation invariant Reeb graph.

The extended version of the Reeb graph, called the Multi-Resolution Reeb graph (MRG) [5] has been proposed. To construct the MRG, the topological characteristics of the

shape must be defined in terms of the critical points of a function on the manifold; this function is called a mapping function. The mapping function maps the points from the manifold of the shape to the domain of the function, and the configuration of the critical points of the mapping function can represent the topology of the shape [10]. This configuration can be embedded by the Reeb graph, and becomes an essential property of the shapes. When the mapping function is defined, the model is then partitioned into regions that correspond to equal intervals of the mapping function [11]. Each partition of the model is represented as a node in the Reeb graph, and adjacent nodes are linked by an edge that connects the corresponding nodes.

Blum et al. [3] propose to use the medial axis to define the skeleton. The algorithm for computing the skeleton in terms of the medial axis is often refers to as the "grassfire" algorithm. The main idea of the "grassfire" algorithm is lighting a fire, started from the border of the object, and let it burn into the object at a constant speed; it will then meet in the medial axis. One starts on the border of the object and strips away one layer of pixels after another until one reaches points that fire reaches from two directions [1]. The medial axis transform of the region is the set of points reached by more than one fire at the same time. Unfortunately, the medial axis transform does not provide robustness against a rotation and a translation of the objects. Therefore, a more sophisticated algorithm is needed in order to solve this problem, and that algorithm is the geodesic distance function. By using the ratio of distance as shown in Eq. (3) to compute the distance function, any changes of a rotation and/or a translation do not affect the value of a function. Thus, this becomes the major property of the function of geodesic distance that gives the advantage of being invariant to a rotation and a translation.

Numerous Reeb graph applications have already been proposed for the freeform 3D models unfortunately, they are sensitive to the connectivity of the boundary representation, which is often violated [11] and they can not capture the geometry of the shapes, which is an important feature in shape analysis [8]. This paper introduces the geodesic distance function together with the medial axis transform for generating the pruned skeleton based on the Dijkstra's shortest path. The basic idea of the new method is to iteratively snip off the spurious skeleton joints that have been produced from the step of the branch region determination (see the proposed pruning method for detail). Applying the function of geodesic distance can guarantee that the new method is invariant to a rotation and a translation, and it is also robust against the changes in the connectivity on the 3D shapes. A brief review of the geodesic distance function and the medial axis transform are illustrated in sections 2 and 3. The proposed method is outlined in section 4. Finally, the experimental results and the conclusion are given in sections 5 and 6, respectively.

2) THE GEODESIC DISTANCE FUNCTION

The geodesic distance can be defined as the distance between two points on the surface of the model, computed by using the Dijkstra's algorithm to find the shortest path between the two points on the surface made up by n points [10].

Definition 1. Suppose $P = \{p_1, p_2, \ldots, p_n\}$ is a path in a connected surface over points p_1 and p_n. *The geodesic distance* between two points p_1 and p_n is the length of the shortest path from p_1 to p_n where the path length $G(P)$ is defined as

$$G(P) = \sum_{i=1}^{n-1} geod\_dist\,(p_i, p_{i+1}) \tag{1}$$

Thus, *the geodesic distance function* measures the average distance between a point and all the other points over the surface of a model.

Note that the function $G(P)$ given by Eq. (1) is not invariant with respect to a translation. In order to gain the invariance, point p_1 and p_n are supposed to locate at the centroid μ of the surface of interest and set

$$G(P_\mu) = \sum_{i=1}^{n\mu_i} geod\_dist\,(p_i, \mu_i) \tag{2}$$

Then, the invariant geodesic distance function is introduced by the formula

$$\overline{G}(P_\mu) = \frac{G(P_\mu) - G(P)_{\min}}{G(P)_{\max} - G(P)_{\min}} \tag{3}$$

According to [2], the geodesic distance function given by Eq. (3) is rotation and translation invariant.

3) THE MEDIAL AXIS TRANSFORM

The idea of using a skeleton of an object as an abstraction of the shape goes back to [3]. Blum et al. define a *skeleton* in terms of the *medial axis* (*MA*), which is a set of curves that roughly run along the middle of an object, as shown in figure 2. According to [3], the medial axis of a curve S is the locus of the centers of the maximal disks contained in S. Let R^n be a symmetry set (where n is a number of dimensional space) which is defined similarly to the medial axis, except that it also includes the circles not contained in S and thus the medial axis is a subset of the symmetry set. The medial axis can then be defined as follows:

Definition 2. Let S be an arbitrary curve; Let $D^n(p, r)$ be a closed disk with a radius r centered at a point p, where $S(p, r) \subseteq R^n$. A maximal disk in S is a closed disk $D^n(p, r)$ contained in S.

Property 1. If X is a maximal disk in S, then S is not properly contained in any other closed disk in S.

Definition 3. Let $S(p, r) \subseteq R^n$. The medial axis (MA) of S is the locus of the centers of the maximal disks contained in S. The medial axis of a 3D object denoted R^3 is sometimes called the *medial surface*. The continuous function of a real-value that assigns to each center of a maximal disk in S is called the *radius function* of that medial axis.

The medial axis together with the associated radius function of the maximal disks is called the *medial axis transform* (*MAT*). Then, the medial axis can be defined:

Definition 4. The medial axis transform (MAT) of an object consists of its medial axis together with the associated radius function.

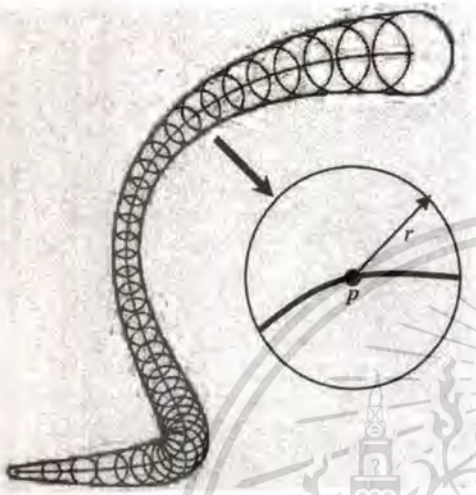


Figure 2. The medial axis of an object

4) OVERVIEW OF THE PROPOSED METHOD

According to [12], the following definitions are used to formally define the pruned skeleton in the proposed method:

Definition 5. The *degree* of a point is defined as a finite number of points in its 26-neighborhood [12] (figure 3 shows the 26-neighborhood structure).

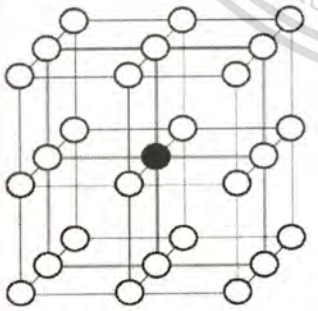


Figure 3. The 26-neighborhood structure

Definition 6. By the assumption that the pruned skeleton is one-pixel thick, the skeleton *end point* is a point that has a degree one. The *middle point* is a point that has a degree two. The *connection point* is a point that has a degree three or higher and all the other neighbors are either the end points or the middle points. The *branched connection point* is a point that has a degree three or higher and at least one of its neighbors is neither an end point nor a middle point. The 26-connected branched connection points form the *branched region*.

Figure 1 shows an overview of the proposed method the original 3D object is shown in figure 1(a). The method involves of nine different steps:

(1) Compute the degree of each point (figure 1(b)).

(2) Determine which point are the end points, the middle points, the connection points, and the branched connection points. If there are no branched connection points, exit the program.

(3) Organize the branched connection points into the branched regions. Each branched region consists of 26-adjacent branched connection points.

(4) In each branched region, find all the end points and the middle points that are 26-adjacent to this branched region, and mark each as "terminator".

(5) Determine the centroids, based on Blum's definition of a skeleton; the skeleton points must be the centers of the maximal disks contained in the curve S.

(6) Compute the geodesic distance from the particular centroids to all points in step 4.

(7) Apply the Dijkstra's algorithm, in order to find the shortest paths between the centroids and their "terminator".

(8) Remove the branched connection points that are not on any of the shortest paths (figure 1(c)).

(9) Generate the pruned skeleton from the remaining points (figure 1(d)).

The algorithm of the proposed method is illustrated in figure 4.

5) EXPERIMENTAL RESULTS

The proposed method is tested on the Princeton Shape Benchmark database [9]. A brief comparison between the results of the proposed method and their original skeleton, extracted by using the Reeb graph is provided. The proposed method can reduce the number of the branched connection points, which is similar to the number of critical points in the original Dijkstra's algorithm therefore the spurious skeleton joints can be removed. These joints will not influence the structure of the skeleton because they are not on any of the shortest paths computed by Dijkstra's algorithm. The pruned skeleton can then be generated in the sense that it can capture the essential shape of a 3D object while preserving a compact form of its data structure. Figure 5 shows the experimental results.

Table 1 compares the number of points after applying the pruning method with the number of points of the original 3D objects. The proposed method can reduce the number of critical points in the graph's nodes, the branched connection

Input: A set of points on the surface of model, P = $\{p_1, p_2, \ldots, p_n\}$.
Output: The pruned skeleton of the given model.
// Read P and determine which p is an endpoint, a middle point, a connection point, and a branched connection point.
for all $p \in$ *surface*
  compute *degree of p*
    if *degree of p* = 1
      $p \leftarrow$ *endpoint*
        if *degree of p* = 2
          $p \leftarrow$ *middle point*
            if *degree of p* ≥ 3 **and** all the other neighbors are either the end points or the middle points
              $p \leftarrow$ *connection point*
                if *degree of p* ≥ 3 **and** at least one of its neighbors is neither an end point nor a middle point
                  $p \leftarrow$ *branched connection point*
                  store *p* in *ListOfBranchedConnectionPoint*
                end if
            end if
        end if
    end if
end for
// Construct a branched region from p in ListOfBranchedConnectionPoint.
for all $p \in$ *ListOfBranchedConnectionPoint*
  if *NumberOfNeighbours*(*p*) = 26
    construct *BranchedRegion*(*p*)
  end if
end for
// Find the end points and the middle points that are 26-adjacent to this branched region, and mark each as "terminator".
for all $p \in$ *BranchedRegion*
  if *degree of p* = 1
    $p \leftarrow$ *endpoint*
    $p \leftarrow$ *terminator*
      if *degree of p* = 2
        $p \leftarrow$ *middle point*
        $p \leftarrow$ *terminator*
      end if
  end if
end for
// Determine the centroids in each branched region.
// Compute the geodesic distance and find the shortest path correspond to each centroid.
for all $p \in$ *BranchedRegion*
  $CentroidOfBranchedRegion = (\sum_{i=1}^{N} x_i/N, \sum_{i=1}^{N} y_i/N, \sum_{i=1}^{N} z_i/N)$
    if $p \approx$ *CentroidOfBranchedRegion*
      $p \leftarrow$ *centroid*
      geod_dist (p, p_i) // compute the geodesic distance from centroid to all p_i, where p_i = terminator
      Dijkstra's_shortest_path (p, p_i) // apply the *Dijkstra's* algorithm
                                            // to find the shortest path from centroid to all p_i
      construct *ShortestPath*(*P*) where $P = (p, p_i)$
    end if
end for
// Remove the branched connection points that are not on any of the shortest paths.
// Construct the pruned skeleton.
for all $p \in$ *BranchedRegion*
  for all $p \in$ *ListOfBranchedConnectionPoint*
    if $p \notin$ *ShortestPath*
      remove *p*
      update *ListOfBranchedConnectionPoint*
      construct *PrunedSkeleton*(*p*) // construct the pruned skeleton from the remaining points in the list
    end if
  end for
end for

Figure 4. Pseudo-code for the proposed method

points in a skeleton graph. Therefore, the numbers of the spurious skeleton joints are reduced comparable to the number of unnecessary joints generated by the Reeb graph method.

Mesh	Number of points	
	The Reeb graph	The pruned skeleton
Cow	2904	2655
Rabbit	1238	1026
The Dragon	1166	1011
The Alien	429	303
The Femme	635	503
Boy	17342	16017

Table 1. Comparison of the number of points between the proposed method and the original model

Let n be the number of points in the meshed model. The computational complexity of the proposed method can be considered as follows:

(1) The computation of degree of each point and the point analysis takes $O(n)$ time.

(2) The computation of testing and finding the maximal disks in the neighborhood of each point takes $O(n)$ time.

(3) The computation of the single source shortest paths (in term of the geodesic distance from that source) to all points on the mesh takes $O(n \log n)$ time [4].

Thus, the overall computation takes $O(n \log n)$ time.

6) Conclusion and Future Works

In figure 5 (i) and (j), it is easy to observe that the skeleton extracted by using the Reeb graph is very sensitive to the connectivity of the boundary representation and it also has a problem in producing the unwanted skeleton joints. The proposed method can overcome these drawbacks. The results show the meaningful characteristics of the 3D meshed model, and the generated skeletons whose joints can be associated with the 3D meshed model.

Given the skeleton construction process with the proposed method of pruning the unnecessary skeleton joints, an important challenge remaining is to find a method for generating the realistic joints that are similar to the joints of the model's anatomy.

References

[1] M. K. Agoston. Computer graphics and geometric modeling: implementation and algorithms. Springer, ISBN 1852338180, page 185, 2005.

[2] S. Baloch, H. Krim, I. Kogan, and D. Zenkov. Rotation invariant topology coding for 2D and 3D objects using Morse theory. In *Proceedings of IEEE International Conference on Image Processing, 2005, ICIP 2005*, pages 796-799, Sep 2005.

[3] H. Blum, R.N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition 10*, pages 167-180, 1978.

[4] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. In *Symposium on Computational Geometry*, pages 344-350, 2003.

[5] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. *Computer Graphics Annual Conference Series (Proc. Of SIGGRAPH)*, pages 203–212, 2001.

[6] F. Lazarus, and A. Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, ACM Press, pages 130–140, 1999.

[7] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [On the singular points of a complete integral pfaff form or of a numerical function]. *Comptes Rendus Acad.Science Paris 222*, pages 847–849, 1946.

[8] M. Sabry Hassouna and Aly A. Farag. On the extraction of curve skeletons using gradient vector flow. *Proc. of IEEE International Conference on Computer Vision* ICCV, Rio de Janeiro, Brazil, October 14-20, 2007.

[9] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, pages 167-178, 2004.

[10] O. Symonova. Shape description for 3D model retrieval. *Technical Report #DISI-08-010*, University of Trento, Mar 2008.

[11] O. Symonova, and G. Ucelli. Using topology representation for comparison and retrieval of CAD models. *Computer Graphics topics*, pages 29-30, 5/2005, Vol. 17.

[12] T. Wang. Unit-width curve skeleton generation based on 3D thinning and shortest path finding. *Technical Report TR08-11*, Department of computing science, Faculty of science, University of Alberta, Jun 2008.

Figure 5. The pruned skeletons (right) and their skeletons extracted by using the Reeb graph (left)