Vision, Modeling, and Visualization (2010)

3D Curve-Skeleton Extraction Algorithm Using a Pseudo-Normal Vector Field

Natapon Pantuwong[†1] and Masanori Sugimoto[‡1]

[1]University of Tokyo, Japan

Abstract

A curve skeleton is a line representation of a 3D object. It is useful in many applications, such as animation, shape matching or scientific analysis. The method described in this paper extracts a curve skeleton from the vector field which is created inside the 3D object. The topology of the vector field is analyzed to obtain the curve skeleton. In contrast with previous methods, the vector field is calculated using a pseudo-normal vector. Furthermore, by using the proposed skeleton-growing method, the vector field topology need not be computed for every voxel. Therefore, the proposed approach requires significantly less computation compared with previous vector field-based approaches, while still capturing all important parts of 3D object. The proposed method is very useful for any applications, especially real-time applications such as quick animation production and prototyping of graphical systems.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: , Computational Geometry and Object Modeling – Curve, surface, solid and object representations

1. Introduction

3D models are commonly used in many applications, including visualization [LCK10], animation [WP00, BP07], flight planning for virtual colonoscopy [HHCL01], shape representation and shape retrieval [SSGD03]. These applications sometimes need a compact representation of the original 3D object. One widely used method is a line representation, which is called a "curve skeleton". The curve skeleton captures all significant topological information of the original object. There are several properties of both the curve skeleton and curve-skeleton extraction as described below [CSM07].

- **Homotopy:** the curve skeleton should preserve the essential topology of the 3D object.
- **Robustness:** curve-skeleton extraction should be robust to small perturbations and transformations.
- **Thinness:** the curve skeleton should be thin.
- **Centeredness:** the curve skeleton should be located at the medial surface of the 3D object.
- **Connected:** for a single connected component object, the curve skeleton should be a single-connected curve.

The main contribution of this paper is a fast curve-skeleton extraction algorithm. The curve-skeleton extraction proposed in this paper is based on a vector field approach. Instead of using a repulsive force function, as in previous work [LWM*03, CSYB05], a vector field is calculated by using the pseudo-normal vector. The pseudo-normal vector field can be calculated more cheaply compared with previous methods because it uses individual voxels instead of a large set of boundary points. The curve skeleton is then obtained from the vector field by extracting and connecting the critical points in that vector field. This paper also proposes a skeleton-growing method that does not need detection of critical point at all voxels. As there may be some missing joints, the proposed method uses the divergence value to determine the additional joints.

This paper is organized as follows. Section 2 presents related work; the mathematical background of vector field

† email : na@itl.t.u-tokyo.ac.jp
‡ email : sugi@itl.t.u-tokyo.ac.jp

DOI: 10.2312/PE/VMV/VMV10/235-242

topology is explained in Section 3; an overview of the algorithm is briefly explained in Section 4; the vector field creation and critical point detection algorithm is discussed by comparing the previous method with the proposed method in Sections 5 and 6, respectively; the additional joints detection method is explained in Section 7; Section 8 contains the experimental results; and the conclusion and future work is discussed in the final section.

2. Related work

Curve-skeleton extraction from 3D objects is a well-known problem. There are several approaches that could be found in the literature. Those previous methods will be described briefly in this section. For more detail, a very good review of curve-skeleton extraction methods can be found in the paper written by Cornea et al. [CSM07].

Curve-skeleton extraction methods can be divided roughly into two categories based on the representation of 3D objects. The *volumetric approach* is used for 3D objects represented by a collection of voxels (volumetric pixels). The other approach is the *geometric approach*, used for polygonal mesh 3D objects. Due to the existence of voxelization [Lia08] and mesh generation [Zha05] algorithms, both kinds of models are convertible to each other. Therefore, we can apply both categories of methods to any kind of 3D object.

2.1. Geometric approach

The geometric approach uses the object mesh information to extract the curve skeleton. The main advantage of the geometric approach is that it is typically faster than the volumetric approach. There are two famous classes of methods: Voronoi-based methods [OI92] and Reeb graph-based methods [PP09].

Voronoi-based approaches use a Voronoi diagram to extract the medial surface of the 3D object. Then, the curve skeleton can be extracted by using a thinning or pruning method causing the extracted curve skeleton to not be thin, and located at the center of the 3D object.

The Reeb graph is a 1D structure whose nodes are critical points of a defined function on the 3D model surface. Most Reeb graph-based methods require users to set the boundary conditions explicitly. The constructed Reeb graph must be embedded into the 3D model in the postprocessing step. Therefore, the Reeb graph may not be located at the center of the object. Furthermore, this approach is not robust against transformation because the critical points depend on the defined function.

2.2. Volumetric approach

The volumetric approach analyzes information from the voxels that are located inside the object. The main advantage of this approach is that it can handle objects made of multiple overlapping parts (such as clothes over a body). There are three popular classes of methods, including thinning [SCG*09], distance field-based methods [WDK01], and vector field-based methods [LWM*03, CSYB05, JQL07].

Thinning is a process of generating a curve skeleton by iteratively removing voxels from the object boundary until the required thinness condition is satisfied. Most thinning algorithms are designed for a specific connectivity (such as 6, 8 or 26 connectivity), so this method is data dependent.

The distance field is the field of the shortest distances from each interior voxel to the boundary. This method attempts to search for a set of candidate voxels that are locally centered in the 3D object. Because the candidate set is fairly large, some postprocessing method, such as thinning and reconnecting, needs to be applied. Therefore, the curve skeleton depends on the postprocessing step.

Vector field-based approaches attempt to extract the curve skeleton from the vector field of the 3D object. The vector field can be calculated by a potential function where the potential of an interior point in the 3D object is the sum of the potentials generated by the points of the object's boundary. This approach can generate a nice curve with all the important properties mentioned previously. However, this is achieved at the cost of increased computational complexity.

3. Mathematical background of vector field topology

The topological structure of 3D vector fields has been well understood in the visualization community for many years [GLL91, HH91, TWHS03]. This section explains the important mathematical formula of vector field topology calculation, which will be referred to in a later section of this paper.

3.1. Critical points

Given a 3D vector field $\mathbf{v}$, a vector for each position *(x,y,z)* is *[u(x,y,z), v(x,y,z), w(x,y,z)]*, where *u,v* and *w* are the axial vector components for *x,y* and *z* axis, respectively. A first order critical point, $p_c = (x_c, y_c, z_c)$, is a point with $\mathbf{v}(p_c) = 0$. There are three categories of critical points based on the direction of vectors around each critical point, which are listed below.

- **Attracting node** is the point where all the vectors converge to that point.
- **Repelling node** is the point where all the vectors diverge from that point.
- **Saddle node** is the point where some vectors converge and others diverge.

The eigenvalues and eigenvectors of the transposed Jacobian matrix of the vector field at the critical point are of particular interest. The eigenvectors correspond to the directions of the vectors around that critical point. The Jacobian

matrix of the vector field at any point is defined by the following equation.

$$\mathbf{J}(p_c) = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} \quad (1)$$

Let $\lambda_1, \lambda_2, \lambda_3$ be the eigenvalues of $\mathbf{J}^{\mathbf{T}}(p_c)$ ordered according to their real parts, i.e., $Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3)$. From the real part of the eigenvalues, we can classify the critical points using the criteria as described below.

- **Attracting node**: $Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3) < 0$
- **Repelling node**: $0 < Re(\lambda_1) \leq Re(\lambda_2) \leq Re(\lambda_3)$
- **Saddle node**: $Re(\lambda_1) < 0 < Re(\lambda_2) \leq Re(\lambda_3)$ or $Re(\lambda_1) \leq Re(\lambda_2) < 0 < Re(\lambda_3)$

The eigenvectors corresponding to the eigenvalues with positive real parts describe the direction of vectors moving outward from the critical point, while the eigenvectors corresponding to the eigenvalues with negative real parts show the direction of vectors moving toward that point.

3.2. Divergence

Divergence is another interesting property of the vector field because it measures the "sinkness" of a point [BS00]. It is a scalar quantity that characterizes the rate of vectors leaving from that point. Given the 3D vector field defined as mentioned previously, the divergence at any point, p, of the vector field can be calculated using:

$$\nabla \cdot \mathbf{v}(p) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}. \quad (2)$$

A positive divergence indicates the vectors are mainly moving away from the given point, while a negative divergence describes vectors that are mainly converging to that point.

4. Algorithm overview

To simplify the remainder of this paper, we assume that the given 3D object is represented by a set of voxels. The polygonal-based 3D object could be discretized to obtain its volumetric representation using a voxelization algorithm, as mentioned in Section 2. The curve-skeleton extraction algorithm is summarized here.

1. **Vector field creation**: the first step is vector field creation. The algorithm will be described in Section 5 by comparing the previous method with our proposed pseudo-normal vector method.
2. **Curve-skeleton extraction**: the curve skeleton is extracted by connecting all critical points in the vector field together. The algorithm of this step will be explained in Section 6 for both typical techniques and for our proposed skeleton-growing method.

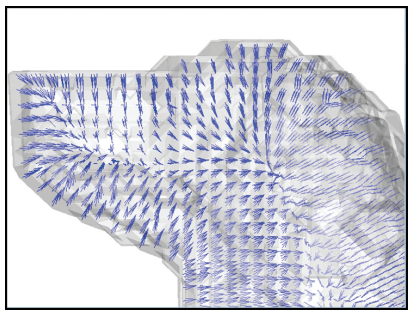


Figure 1: *Vector field created by using a force function value of 6.*

3. **Additional joint detection**: the extracted curve skeleton from the previous step may not capture all important joints. The missing joints can be determined by using the divergence. The algorithm is described in Section 7 of this paper.

5. Vector field creation

A vector field is a collection of vectors inside the 3D object. In previous work, it is usually created using a potential field function or repulsive force field function [LWM*03, CSYB05].This section explains the vector field creation algorithm for both the typical repulsive force function method and our proposed pseudo-normal vector method.

5.1. Repulsive force function

The repulsive force at a given point, P, is influenced by a specific point charge (at the boundary), B_i, and is defined as a force from that point charge that is pushing the given point away with a strength that is inversely proportional to a power of the distance between the point and the charge. The force as described above is calculated using the following equation.

$$\overrightarrow{\mathbf{F}_{PB_i}} = \frac{\overrightarrow{B_iP}}{D^m}, \quad (3)$$

where $\overrightarrow{\mathbf{F}_{PB_i}}$ is the repulsive force at a given point, P, due to the boundary voxel, B_i. $\overrightarrow{B_iP}$ is the unit vector from B_i to P, which describes the direction of the force. D is the distance between P and B_i. The value of m is called the force function value, which controls the characteristics of the created vector field. Then, the force at point P is calculated by summing all of the forces due to every boundary voxel. Figure 1 shows the vector field created by force function values of 6.

The computational complexity of the vector field creation using a repulsive force function depends on the number of interior and boundary voxels of the original 3D object. This method uses all boundary voxels for each interior voxel. Therefore, the computation complexity is $O(N_I \times$

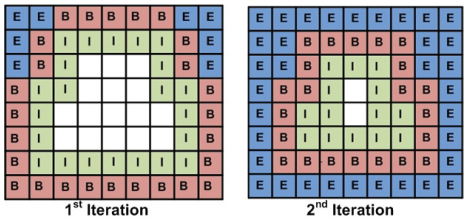


Figure 2: *The boundary points move inward during the vector field calculation. From left to right: The first and the second iteration of the calculation. The character "I", "E", and "B" indicate the interior, exterior, and boundary voxel respectively.*

N_B), where N_I is the number of interior voxels, and N_B is the number of boundary voxels.

5.2. Pseudo-normal vector

Due to the high computational complexity of the repulsive force function, it may not be applicable to real-time applications. The proposed method aims to reduce this complexity. According to Figure 1, the characteristic of the vector at each voxel is like a normal vector pointing to the medial surface of the object. That is, the repulsive force field could be approximated by calculating this normal vector at all voxels. Vector field creation using this proposed method is described in Algorithm 1 and also shown in Figure 2. The boundary points are marked as "B" at the beginning of the iteration. After completing the calculation, the interior neighboring points marked as "I" will become the new boundary points in the next iteration, and the previous boundary points will become exterior points marked as "E". The algorithm will stop when there is no further interior voxels in the object, otherwise this algorithm will calculate the normal vector at every boundary voxel. The `markVoxel` function marks all boundary voxels as exterior voxels, and also their interior neighboring voxels, so they become the new boundary voxels before going to the next iteration.

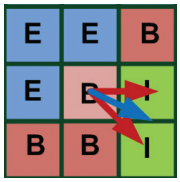


Figure 3: *Illustration of pseudo-normal vector calculation at the boundary voxel. Pseudo-normal vector is shown by blue vector. Red vectors are unit vector created during the calculation.*

To create a normal vector, a tangent plane needs to be defined at each voxel. We proposed the pseudo-normal vector, which is calculated using the position of the voxels without creating a surface or tangent plane. The pseudo-normal vector acts as a normal vector. It is approximately perpendicular to the surface around that voxel and points into the object. Figure 3 demonstrates pseudo-nomal vector calculation. From that figure, unit vectors (red vector) are created for every direction that points from an exterior (purple) to an interior (green) voxel. The pseudo-normal vector (blue vector) is then calculated by summing and normalizing over all such created unit vectors. Figure 4 shows an example of the pseudo-normal vector field. The vectors are moving mainly to the center of the object, similarly to the ones created by a repulsive force function. However, the direction of the vectors may not be as smooth as the repulsive force field.

By using this proposed method, the computation complexity of the vector field creation process is reduced. For each step of the calculation, this method uses only the neighborhood of each voxel instead of using the large set of boundary voxels, as in the repulsive force function method. Therefore, the computational complexity of this method is reduced to $O(N_I + N_B)$, which equals $O(N_{All})$, where N_{All} is the total number of object voxels. It means that as the number of object voxels is increased, the computational time will be increased linearly for this method.

Algorithm 1 : createVectorField(object)

```
if numberOfInteriorVoxel == 0 then
     stop calculation
else
     for i=1 to numberOfBoundaryVoxel do
          calculateNormalVector()
     end for
     object = markVoxel(object)
     createVectorField(object)
end if
```

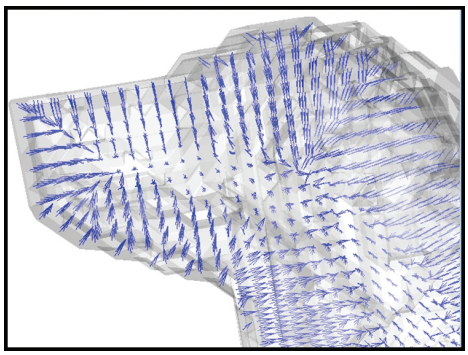


Figure 4: *The Pseudo-normal vector field.*

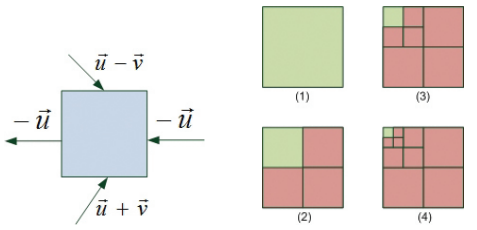


Figure 5: *Critical point detection method using the vector direction as a criterion. Left: Example of a voxel cell containing positive and negative axial vector components ($\vec{u}$, $\vec{v}$ and $\vec{w}$). Right: Subcell creation during critical point detection. Subcell that passes the candidacy test is shown by green voxel. Red voxel indicates the voxel that does not pass such test.*

6. Curve-skeleton extraction

After the vector field is created inside the 3D object, the curve skeleton is extracted from this vector field by analyzing the vector field topology. Previous work detected the critical points in the vector field and connected them to form a curve skeleton. Although our method also uses this strategy, we proposed a skeleton-growing method, which can reduce the computational time compared with the previous work.

6.1. Typical approach

Critical points could be detected by examining all object voxels. As mentioned in Section 3, a critical point is a point where the magnitude of the vector is zero. However, detecting a critical point is difficult using only individual voxels, because it is usually located between voxels. Therefore, this process must use the voxel cell (the given voxel and its neighborhood). The critical point detection method described in [CSM07] uses the direction of vectors around a voxel cell to be a criterion. Figure 5 demonstrates this idea. If the voxel cell contains the vectors with all positive and negative values for every axial vector component ($\vec{u}$, $\vec{v}$ and $\vec{w}$), as shown in Figure 5(left), this cell is a candidate to contain the critical point. This method then divides the voxel cell into 8 subcells, as shown in Figure 5(right), and retests in the same manner for each subcell. The process ends when a cell or subcell fails the test (red subcell in Figure 5(right)), or the subcell is too small, and still a candidate. The center of a candidate subcell that passes the candidacy test (green subcell in Figure 5(right)) is taken to be a critical point. The curve skeleton is then extracted by connecting all detected critical points. The critical points need to be classified using the eigenvalues and eigenvectors of the transposed Jacobian matrix of the vector at that point, as mentioned in Section 3. Because saddle nodes are always located between attracting and repelling nodes, the curve skeleton can be extracted by starting at a saddle point, and then moving to the direction of outward pointing eigenvectors until another critical point or a previously visited location is met.

6.2. Skeleton-growing method

As mentioned previously, the critical point detection method is an iterative calculation for each voxel cell. Therefore, detecting a critical point for every voxel is also a time-consuming task. We propose the skeleton-growing method, which does not need to detect the critical points for every voxel, as in the previous method. The skeleton-growing method starts with a single seed point, and then uses that seed point to find the other critical points. The seed point must be one of the critical points that form the curve skeleton, which must be either a saddle or an attracting node. Therefore, if we can find one such point, we can use it as a seed point to search for the other saddle and attracting nodes. This method uses the divergence calculated by Equation (2) at each voxel to determine the seed point. Calculating the divergence is not an additional task, because we will use this value in a later step. A negative divergence characterizes the rate of the flow converging to that point. Unfortunately, the maximum negative divergence point may not be an attracting node or saddle node. However, it can guarantee that a vector moving away from that point will reach the nearest attracting or saddle node in the vector field. Therefore, we can find the seed point using this strategy. Then, the curve skeleton is extracted by using the method described in Algorithm 2. The variable, `p` , represents a voxel in the 3D object. In the first iteration, function `calculateNextPoint` decides the next position using the direction of eigenvectors and their inverse which are calculated by function `calculateEigenVectorsAndInverse`. After that, Algorithm 2 recursively calls itself with the newly determined point. It determines whether that point is critical or not. We also use the method which described in [CSM07] to detect a critical point in the vector field. If it is a critical point, this method stores the skeleton segment using the function `savePath` and repeats the process to search for other critical points; otherwise, the function `calculateNextPoint` uses the direction of the vector at that point to calculate the next position. For attracting point, the vectors are moving mainly toward to such point, so we have to reverse the direction of vector to move further from that point. Each search path will stop when a boundary voxel is found, which means that this path is not a skeleton segment, or previously constructed skeleton is found, which means that this path is determined as a skeleton segment. If there is no available search path, this algorithm will terminate. To avoid redundancy in critical point detection, Algorithm 2 marks the voxels that have already been determined as critical or general by using the function `flagVoxel`.

Using the skeleton-growing method reduces the number of critical point detections needed. Although it cannot guarantee the rate of this reduction because it depends on

the direction of vectors in the vector field, we can ensure that, in the worst case, the number of calculations is not greater than the typical method. The step-size of vector field tracing process is an important factor. It should be less than a voxel size to ensure that there is no missing critical point. Figure 6 shows the resulting curve skeleton obtained using the skeleton-growing method. The proposed method delivers a correct curve skeleton which captures all important parts, although the shape of the curve skeleton may not be as smooth as the one extracted from a repulsive force field.

Algorithm 2 : skeletonGrowing(p)

```
if p is boundary voxel then
 stop growing
else if p is skeleton segment then
 savePath()
 stop growing
else
 if p has not been flaged
  detectCriticalPoint()
 if p is critical point then
  flagVoxel()
  savePath()
  calculateEigenVectorsAndInverse()
  for each eigenvectors and their inverse
    pn = calculateNextPoint()
    skeletonGrowing(pn)
  end for
 else
  flagVoxel()
  pn = calculateNextPoint()
  skeletonGrowing(pn)
 end if
end if
```

7. Additional joints detection

According to Figure 6, the resulting curve skeletons do not capture all important parts of the dog model. From our observation, the missing joints are usually located in the high curvature area. The vectors around the high curvature area tend to converge in that region and move to the nearest critical point. We use the negative divergence to determine the convergence rate. The threshold value, *thres*, is calculated by using Equation (4). However, it can be controlled by the *percentage* value specified by the user:

$$thres = Div_{min} + [(Div_{max} - Div_{min}) \times percentage], \quad (4)$$

where Div_{max} and Div_{min} are maximum and minimum divergences of the vector field, respectively.

All points having a divergence lower than the threshold will be selected as additional joints. Figure 7 shows the curve skeletons that include the additional joints found using a *percentage* value of 25%.

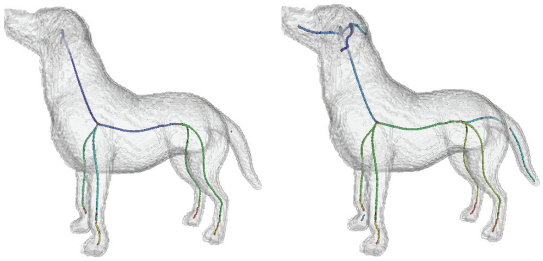


Figure 6: *Curve skeleton extracted using the skeleton-growing method from a repulsive force field (left) and a pseudo-normal vector (right).*

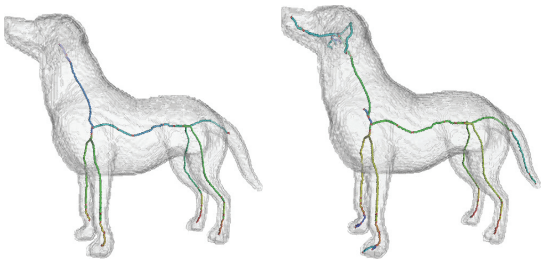
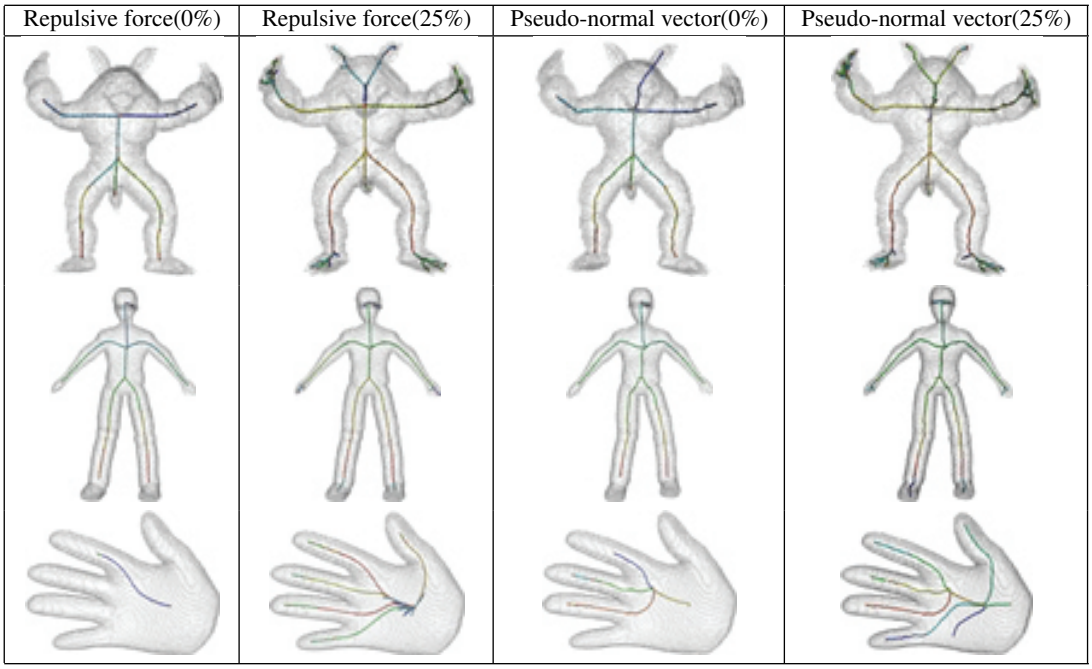


Figure 7: *Curve skeleton extracted using the skeleton-growing method from a repulsive force field (left) and a pseudo-normal vector (right). Additional joints are detected using percentage* = 25%.

8. Experimental results

The experiment is set up to measure the accuracy of the proposed method. We used 30 different 3D models in our experiment. The curve skeletons were extracted from all of the voxelized 3D models using the skeleton-growing method with the vector field created by both a repulsive force and the pseudo-normal vector. The step-size of vector field tracing process is 20% of voxel size. Examples of resulting curve skeletons are shown in Figure 8. Additional joints were detected with a *percentage* value of 25%. In this experiment, we found that the *percentage* value of around 20–25% is enough to capture all important parts of the original object. Although the shapes of the curve skeletons extracted using our proposed method may not be as smooth as the one extracted by the repulsive force field, they are located at the correct position. Therefore, our method is applicable for applications such as animation systems, which do not need a smooth curve skeleton. Resolution of the voxelized 3D model is an important factor affecting the accuracy of the extracted skeleton. Cleary that, the higher resolution we use , the better accuracy we get. In this expermiment, the resolution of each model is around 80^3 voxels. We also measured the computational time for both vector field calculations and the curve-skeleton extraction process by comparing the previous method with our proposed method. The

Figure 8: *Example of curve skeletons extracted using the proposed method from a repulsive force field and the pseudo-normal vector field. Additional joints are detected using a percentage value of 25%.*



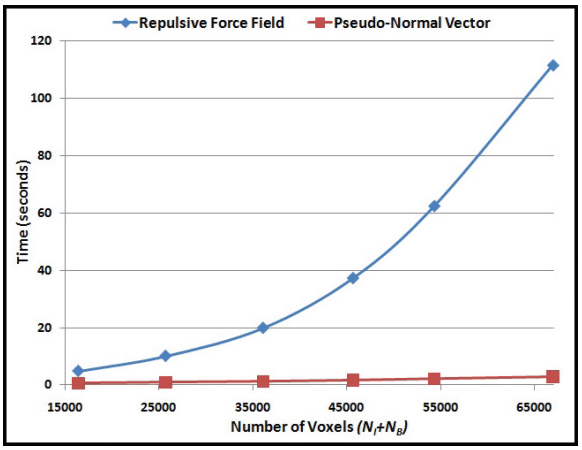


Figure 9: *Computational time (in seconds) of the vector field creation process for the repulsive force field (blue) and the pseudo-normal vector field (red).*

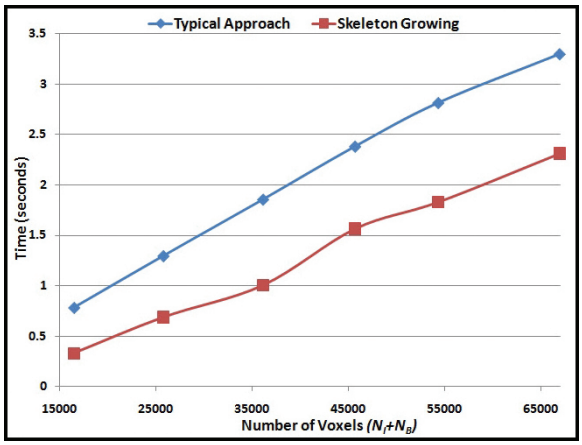


Figure 10: *Computational time (in seconds) of the critical point detection process between a typical approach (blue) and the skeleton-growing method (red).*

computational time was measured using an Intel Core2Duo 2.4 GHz machine with 1 GB of RAM. The developed software is a single-threaded application so only one core was used. Figure 9 compares the computational time of the vector field creation using the repulsive force field with our proposed method. As the number of voxels is increased, the computational time of the pseudo-normal vector field increases linearly, while for the repulsive force field, it increases exponentially. Using our proposed method, the computational time is significantly reduced. Figure 10 shows the computational time of the critical point detection process. This experimental result confirms that the skeleton-growing method can reduce the computation time compared with the traditional methods.

9. Conclusion

This paper presents a curve-skeleton extraction from vector field method. The experimental results demonstrate that using the proposed pseudo-normal vector field reduces the computation time of the vector field creation process significantly compared with the repulsive force field. Furthermore, the pseudo-normal vector does not need any parameters, making this an easier algorithm to use. The skeleton-growing method can also decrease the computational time, because it reduces the number of voxels needing to be calculated. Some missing joints can be detected using the divergence of every pixel to determine a convergence area. Although the proposed method could not produce a smooth curve skeleton, it is applicable for any applications that require rapid processing in the skeletonization step. In future work, we would like to apply our proposed method to the motion retargeting method to complete an automatic animation system.

References

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM. 1

[BS00] BOUIX S., SIDDIQI K.: Divergence-based medial surfaces. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I* (London, UK, 2000), Springer-Verlag, pp. 603–618. 3

[CSM07] CORNEA N., SILVER D., MIN P.: Curve-skeleton properties, applications, and algorithms. *Visualization and Computer Graphics, IEEE Transactions on 13*, 3 (5-6 2007), 530–548. 1, 2, 5

[CSYB05] CORNEA N., SILVER D., YUAN X., BALASUBRAMANIAN R.: Computing hierarchical curve-skeletons of 3D objects. 945–955. 1, 2, 3

[GLL91] GLOBUS A., LEVIT C., LASINSKI T.: A tool for visualizing the topology of three-dimensional vector fields. In *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on* (22-25 1991), pp. 33–40. 2

[HH91] HELMAN J., HESSELINK L.: Visualizing vector field topology in fluid flows. *Computer Graphics and Applications, IEEE 11*, 3 (may 1991), 36–46. 2

[HHCL01] HE T., HONG L., CHEN D., LIANG Z.: Reliable path for virtual endoscopy: Ensuring complete examination of human organs. *IEEE Transactions on Visualization and Computer Graphics 7*, 4 (2001), 333–342. 1

[JQL07] JIA J., QIN Z., LU J.: Stratified helix information of medial-axis-points matching for 3D model retrieval. In *MIR '07: Proceedings of the international workshop on Workshop on multimedia information retrieval* (2007), ACM, pp. 169–176. 2

[LCK10] LANGEVELD V., CHRISTENSEN M., KESSLER R.: Digital visualization tools improve teaching 3d character modeling. In *SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education* (New York, NY, USA, 2010), ACM, pp. 82–86. 1

[Lia08] LIAO D.: GPU-accelerated multi-valued solid voxelization by slice functions in real time. In *VRCAI '08: Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (New York, NY, USA, 2008), ACM, pp. 1–6. 2

[LWM*03] LIU P.-C., WU F.-C., MA W.-C., LIANG R.-H., OUHYOUNG M.: Automatic animation skeleton construction using repulsive force field. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society. 1, 2, 3

[OI92] OGNIEWICZ R., ILG M.: Voronoi skeletons: theory and applications. pp. 63 –69. 2

[PP09] POIRIER M., PAQUETTE E.: Rig retargeting for 3d animation. In *GI '09: Proceedings of Graphics Interface 2009* (Toronto, Ont., Canada, Canada, 2009), Canadian Information Processing Society, pp. 103–110. 2

[SCG*09] SHE F., CHEN R., GAO W., HODGSON P., KONG L., HONG H.: Improved 3d thinning algorithms for skeleton extraction. In *Digital Image Computing: Techniques and Applications, 2009. DICTA '09.* (1-3 2009), pp. 14–18. 2

[SSGD03] SUNDAR H., SILVER D., GAGVANI N., DICKINSON S.: Skeleton based shape matching and retrieval. In *SMI '03: Proceedings of the Shape Modeling International 2003* (2003). 1

[TWHS03] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3d vector fields. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), IEEE Computer Society. 2

[WDK01] WAN M., DACHILLE F., KAUFMAN A.: Distance-field based skeletons for virtual navigation. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 239–246. 2

[WP00] WADE L., PARENT R. E.: Fast, fully-automated generation of control skeletons for use in animation. In *CA '00: Proceedings of the Computer Animation* (Washington, DC, USA, 2000), IEEE Computer Society, p. 164. 1

[Zha05] ZHANG H.: *Mesh generation for voxel-based objects*. PhD thesis, West Virginia University, Morgantown, WV, USA, 2005. Chair-Smirnov, Andrei V. 2