EFFICIENT RENDERING OF ANATOMICAL TREE STRUCTURES USING GEOMETRY PROXY

Hang Dou[⋆]*, Christian Bauer*[* †]*, Member, IEEE, Chris Wyman*[⋆]*, Reinhard R. Beichel*[* † ‡]*, Member, IEEE*

[⋆] Dept. of Computer Science, The University of Iowa, IA 52242
[*] Dept. of Electrical and Computer Engineering, The University of Iowa, IA 52242
[†] The Iowa Institute for Biomedical Imaging, The University of Iowa, Iowa City, IA 52242
[‡] Dept. of Internal Medicine, The University of Iowa, Iowa City, IA 52242

ABSTRACT

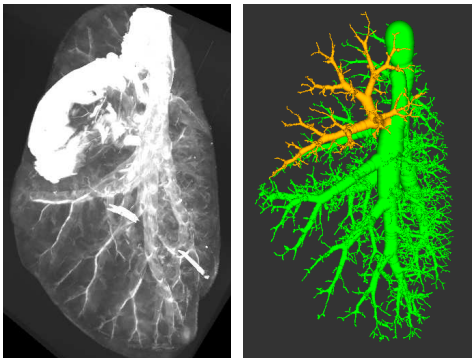
Rendering tubular structures efficiently is crucial for studying anatomical tree-like structures, such as vessels and airways. Most existing methods are based on surface reconstruction, resulting in complex meshes, which slows the rendering performance as the tree complexity increases. In this paper, we present an approach to render tubular tree structures using geometry proxy. We generate low complexity proxy meshes on the fly and shade the meshes as tubular objects consisting of truncated cones and spheres. Unlike surface reconstruction, our method requires no precomputation and produces appealing imagery with faster rendering performance.

***Index Terms*—** Geometry Proxy, Tubular Tree Structure, Truncated Cone Visualization

1. INTRODUCTION

Blood vessels, bronchial tree and nerves form dense anatomical tree-like structures. Using modern volumetric imaging techniques such as CT scans or an imaging cryomicrotome, these structures can be depicted with great detail. Understanding these structures is crucial for several medical and biomedical applications, including medical education, therapy planning, biomedical analysis and simulation tasks. Therefore, researchers have the need to interactively explore these structures, which requires efficient visualization methods. Fig. 1 shows an example of such interactive exploration.

Standard volume rendering methods (e.g. [1]) and surface reconstruction methods such as isosurface extraction [2, 3] are generally insufficient for this task, because they do not allow for a visual separation of interwoven arteries and veins, or picking arbitrary sub trees. Image analysis methods exist that allow to reconstruct such structures and to describe them on a higher abstraction level based on the tree's skeleton and associated diameter information. Such representations allow for a compact encoding of the essential structural information and are utilized in several applications, but cannot be visualized directly. To obtain visualizations from such a representation, methods have been proposed to reconstruct a surface mesh based on the vessel skeleton and radii information. For example, Oeltze [4] applied a convolution surface to visualize anatomic tree structures and Hahn [5] utilized truncated cones to visualize symbolic models of vessel structures. However, to obtain visually appealing renderings, these surface reconstruction methods result in high density meshes, which leads to slow rendering performance for complex tree structures.



(a) (b)

Fig. 1. (a) Volume rendering of a rat lung imaged with imaging cryomicrotome. (b) Segmented airway tree structure with selected subtree.

In this work, we present a novel method to efficiently render anatomical tree structures represented by their skeleton and radius information. Rather than reconstructing complex meshes for rendering, we produce one quad for each pair of adjacent centerline points and employ primitive ray intersection on the quads to accomplish a tubular appearance.

2. METHODS

In our method, we decompose the skeleton into segments of adjacent centerline points with associated radius that are rendered individually. Each of these segments is rendered as a truncated cone with a sphere at the endpoint that forms the transition to the next segment as depicted in Fig. 2(e). For

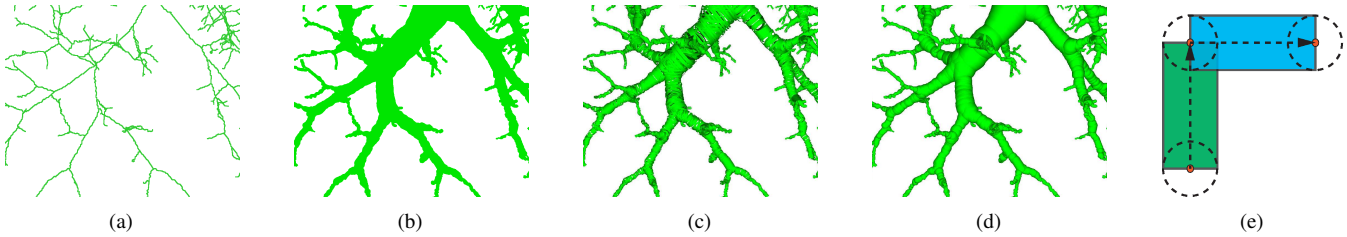


Fig. 2. Rendering process. (a) Input centerline. (b) Generated quads (c) After truncated cone ray intersection. (d) After sphere ray intersection. (e) Spheres at the joints and endpoints form the transition between the truncated cone segments.

the rendering, we utilize the programmable render pipe line of modern graphics processing units (GPUs), which allows us to generate graphics primitives from input segments on the fly rather than simply rendering of precomputed meshes. In a first step, we create quads on the fly that fully cover each segment. In a second step, by pushing quad pixels onto the truncated cone, we analytically find their position and surface normal for use in standard Lambertian shading. Details for both steps are described below. Fig. 2 shows an example with intermediate processing results.

2.1. Quad Generation

We observe that for anatomical tree structures, the projection of each segment, which is represented by a truncated cone and a sphere, on the screen can be covered by a quad (see Fig. 3(a)). Since performance decreases as the rasterized pixel count increases, we aim to quickly generate a quad whose screen space projection fully covers each truncated cone with minimal wasted area.

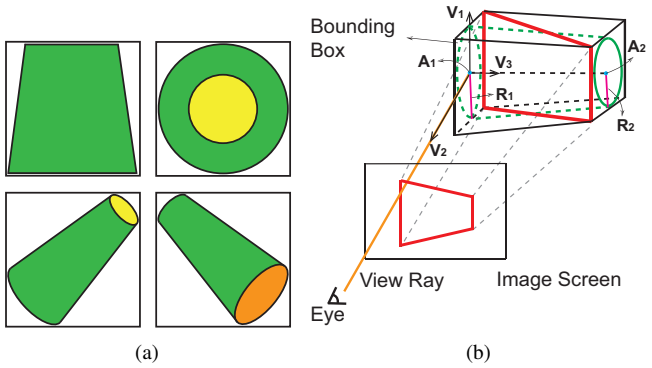


Fig. 3. Quad generation. (a) Various trancated cones projected to the image plane. All projections fall into a quad. (b) Quad generation process. We use points $\mathbf{A_1}$, $\mathbf{A_2}$ and radii R_1, R_2 to identify the red quad inside the bounding box, which minimally bounds the truncated cone's projection.

As shown in Fig. 3(b), given a tubular section with a pair of centerline points $\mathbf{A_1}$, $\mathbf{A_2}$ and vessel radii R_1, R_2, we build the truncated cone's bounding box with one of its diagonal quads facing the viewer. By applying cross product to the eye ray $\mathbf{V_2}$ and cone's axis $\mathbf{V_3}$, we get vector $\mathbf{V_1}$. ($\mathbf{V_1}$, $\mathbf{V_3}$, R_1, R_2) defines the truncated cone's bounding box. To ensure the quad fully covers the cone's projection on the image, we select the diagonal of the bounding box by choosing the one with normal closest to view ray. Additionally, to cover the projection of the sphere within the same quad, we expand the quad in length by the radius of the sphere.

2.2. Primitive Ray Intersection

For each pixel inside a quad, we shade it as a diffuse truncated cone surface:

$$OutputColor = (\mathbf{N} \cdot \mathbf{L}) \times \mathbf{I_m}, \tag{1}$$

where $\mathbf{L}$ denotes the light direction, $\mathbf{I_m}$ denotes the light intensity and $\mathbf{N}$ denotes the surface normal. For the ray intersection with the truncated cone, our goal is to find the hit point along the eye ray on the cone through the given pixel. As depicted in Fig. 4, given a truncated cone C($\mathbf{A_1}$,$\mathbf{A_2}$,R_1,R_2), where $\mathbf{A_1}$ and $\mathbf{A_2}$ are two centerline points and R_1 and R_2 their radii, respectively, we generate the quad as discussed in the previous section. When shading each pixel covered by the quad's projection on the screen, a ray $\mathbf{D}$ is cast from the viewer. First we convert $\mathbf{D}$ into cone space where the origin is $\mathbf{A_1}$ and the Z axis is ($\mathbf{A_1}$,$\mathbf{A_2}$). Suppose the hit point of $\mathbf{D}$ on C is $\mathbf{H}$. In cone space, $\mathbf{H}$ can be derived by solving equations(2), (3) and (4):

$$\mathbf{H} = \mathbf{E} + t \times \mathbf{D}, \tag{2}$$

$$(X_E + t \times X_D)^2 + (Y_E + t \times Y_D)^2 = r^2, \tag{3}$$

$$r = \frac{R_2 - R_1}{||\mathbf{A_1} - \mathbf{A_2}||} \times (Z_E + t \times Z_D) + R_1, \tag{4}$$

where $\mathbf{E}$ indicates eye position, $\mathbf{D}$ indicates ray direction, t indicates the distance between the hit point $\mathbf{H}$ and the viewer $\mathbf{E}$. (X_E, Y_E, Z_E) represents eye position in cone space and (X_D, Y_D, Z_D) represents the ray direction in cone space. Suppose $\mathbf{N}$ is the cone's tip. Then the surface normal at $\mathbf{H}$

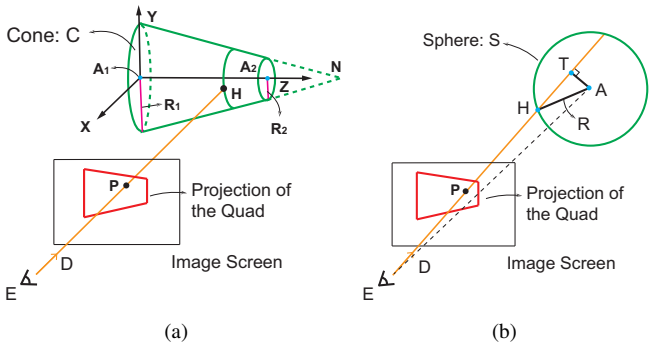


Fig. 4. Primitive ray intersection. P is the pixel we are shading. E represents the viewer and D represents the view ray direction. (a) Truncated cone C intersects with ray D (b) Sphere S intersects with ray D.

can be obtained by generating a vector perpendicular to $\overrightarrow{\mathbf{NH}}$ in the plane ($\mathbf{A_1}$,$\mathbf{A_2}$,$\mathbf{H}$).

For the ray intersection with the sphere at the end of the segment, we solve equations (2), (5), (6) and (7) for each pixel:

$$t = |\mathbf{TE}| - |\mathbf{TH}|, \tag{5}$$

$$|\mathbf{TE}| = (\mathbf{E} - \mathbf{A}) \cdot \mathbf{D}, \tag{6}$$

$$|\mathbf{TH}| = \sqrt{R^2 - (|\mathbf{AE}|^2 - |\mathbf{TE}|^2)}, \tag{7}$$

where A indicates the endpoint, R indicates the radius at the endpoint and H is the hit point along the view ray on the sphere. The pseudocode used for pixel shading is as follows:

Shoot a ray from eye through the pixel
Apply truncated cone intersection and get valid t_{cone}
Apply sphere intersection and get valid t_{sphere}
if t_{cone} is valid or t_{sphere} is valid **then**
  t := Min(t_{cone}, t_{sphere})
  $\mathbf{H} := \mathbf{E} + t \times \mathbf{D}$
  Compute $\mathbf{H}$'s surface normal $\mathbf{N}$
  Shade the pixel by equation 1 and store t as depth value
else
  Do nothing
end if

We observed, that in case the centerline points are very close compared to their radius, the sphere contains most of the truncated cones body and the truncated cone itself does not need to be rendered. Therefore, we introduce an optimization stragety (OPTS), where we render the truncated cone only if the distance between two points is smaller than the smaller of the two segments radii.

3. EVALUATION METHOD AND RESULTS

Our method is implemented with OpenGL/GLSL in C++ and is tested on a machine with an Intel Xeon X5450 CPU @3.00GHz and an NVIDIA GeForce GTX550 Ti graphics card. Two sets of data are used for evaluation: (a) a vascular structure from human lung CT scan and (b) a rat lung airway tree reconstructed from cryomicrotome images. We compare our method with three other approaches: standard cone mesh generation (SCM)[1], raw isosurfaces [2] extracted from binary segmentations and isosurface meshes simplified with QSlim [6]. SCM produces a set of N points along a circle around each centerline point and connects them into a cone.

Fig. 5 shows the quality of our rendering result compared with other approaches. Table 1 and 2 depict the resulting mesh complexity and the corresponding rendering speed. All the images are generated with an image size of 1680×974 pixels.

	FPS	Triangle Number
Our method with OPTS	771	63542
Our method without OPTS	572	63542
SCM with 8 Points	563	569648
SCM with 16 Points	381	1139296
SCM with 32 Points	210	2278592
Isosurface with QSlim	430	481248
Isourface	176	2568428

Table 1. Airway tree of rat lung with 35603 centerline points.

	FPS	Triangle Number
Our method with OPTS	269	216566
Our method without OPTS	251	216566
SCM with 8 Points	86	2100704
SCM with 16 Points	77	4201148
SCM with 32 Points	60	8402816
Isosurface with QSlim	120	1000000
Isosurface	53	3398188

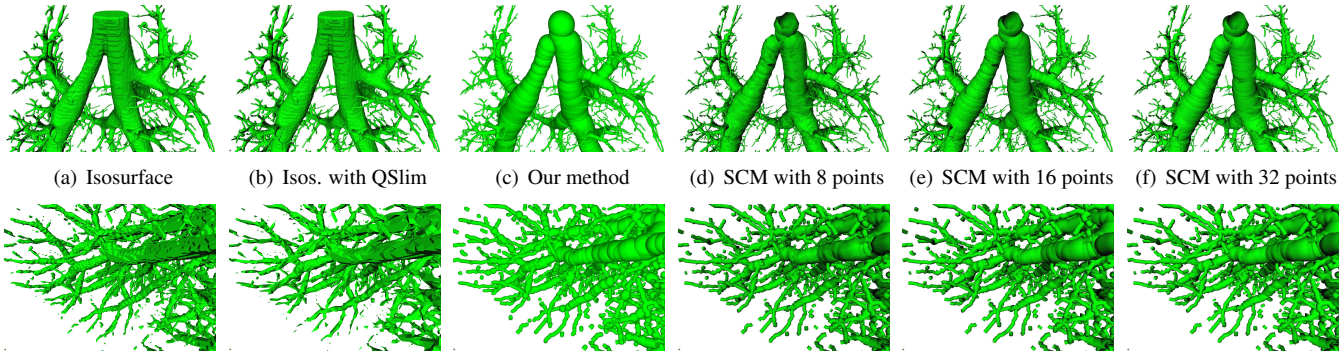
Table 2. Vascular structure from human lung CT scan with 353875 centerline points.

4. DISCUSSION

In our experiments, we compared our method to the results of three other methods.

Standard isosurface mesh extraction from a binary segmentation allows to represent the tree structure very accurately. However, these meshes are of very high density and only allow for low rendering performance. Simplifying these isosurfaces using methods like QSlim (Fig. 5(b)) allows reducing the mesh complexity and increasing the performance.

[1]available in VTK (http://www.vtk.org) in class vtkTubeFilter.



(a) Isosurface (b) Isos. with QSlim (c) Our method (d) SCM with 8 points (e) SCM with 16 points (f) SCM with 32 points

Fig. 5. Rendering results. The first row shows images of a rat lung airway tree extracted from cryomicrotome images and the second row shows images of vascular structures from a lung CT scan. For both cases, only subparts of the whole dataset are shown.

However, the resulting meshes appear visually less appealing and artifacts may occur in branching areas or for thin tree branches, if the meshes are simplified too much. Even after simplifying the isosurfaces, rendering performance is still much slower compared to our proposed method.

Similar to our method, the SCM method utilizes centerline and radius information as input besides an additional parameter n specifying the number of surfaces that are used to approximate the individual cylinder segments between two adjacent centerline points. This parameter represents a trade-off between quality and rendering performance. While the method requires at least 3 surfaces around 16 to 32 surfaces are needed to provide visually appealing results (Fig. 5(e) and Fig. 5(f)). Our method performs about 2-3 times faster than SCM with 16 or 32 points. SCM with 8 points shows comparable performance to our method without the optimization strategy (OPTS), but to the cost of obvious artifacts (Fig. 5(d)). Additionally, the introduced OPTS approach allows increasing the performance by about 10% without any noticable quality degradation. Contrary to the SCM method, our approach (Fig. 5(c)) utilizes only a small number of triangles and does not just provide an approximation of the cones but renders them accurately utilizing analytical ray-intersection. This allows for fast and visually appealing rendering results.

5. CONCLUSION

We present a novel way to render anatomical tree structures. Using centerlines with radius information as input, quads are generated as geometry proxies for the tree. By exploiting primitive ray intersection in OpenGL fragment shader, we shade these proxies with a tubular exterior. Thus, our method gives a real truncated cone look of the branches rather than a triangulated mesh approximation and runs in high frame rates compared to mesh-based approaches.

6. ACKNOWLEDGEMENTS

This work was supported in part by NIH grant R21HL110000. The authors thank Prof. Robb Glenny from the University of Washington for providing the cryomicrotome images.

7. REFERENCES

[1] E. Bullitt and S.R. Aylward, "Volume rendering of segmented image objects," *Medical Imaging, IEEE Transactions on*, vol. 21, no. 8, pp. 998–1002, 2002.

[2] Y. Livnat and C. Hansen, "View dependent isosurface extraction," in *Visualization'98. Proceedings*. IEEE, 1998, pp. 175–180.

[3] Will Schroeder, Kenneth M. Martin, and William E. Lorensen, *The visualization toolkit (3rd ed.): an object-oriented approach to 3D graphics*, Kitware, Inc., Corporate Park Drive, Clifton Park, NY 12065, USA, 2001.

[4] S. Oeltze and B. Preim, "Visualization of anatomic tree structures with convolution surfaces," in *IEEE/Eurographics Symposium on Visualization, Informatik aktuell, S*, 2004, pp. 311–320.

[5] H.K. Hahn, B. Preim, D. Selle, and H.O. Peitgen, "Visualization and interaction techniques for the exploration of vascular structures," in *Visualization, 2001. VIS'01. Proceedings*. IEEE, 2001, pp. 395–578.

[6] M. Garland and P.S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216.