Fast local thickness

Dahls

***Abstract*—The abstract goes here.**

I. Background

Contemporary high-resolution scanners allow reconstruction of volumes typically containing 2048^3 voxels, and with technology development of scanners soon volume containing 4096^3 voxels will be available. With volumes containing billions of voxels even the simplest processing methods may be computationally demanding. This is especially pronounced when using a kernel which also grows cubically in size – processing large images often requires using large kernels. When even the simplest processing is cumbersome, downsizing an image is often seen as a first step in 3D analysis pipeline – effectively degrading the rich information available thanks to the advances in 3D imaging techniques.

For this reason, efficient 3D analysis requires re-visiting processing algorithms. In this paper, we consider the computation of local thickness. As we will show, time complexity of local thickness algorithm is such that if it takes 1 second to process a volume of size 512^3, processing a 2048^3 volume takes 4.5 hours and a 4096^3 volume takes 24 days.

II. Introduction

For an object in 3D, the local thickness in any point of the object is defined as the radius of the larges sphere which fits inside the object and contains the point. Despite this simple definition, computing local thickness using a direct implementation of the definition is computationally very demanding, and may for large volumetric data be computationally infeasible.

We propose an algorithm for fast computation of local thickness. The algorithm utilizes two approximations for a more efficient computation. Firstly, the dilation with the sphere of a large radius is approximated with the consecutive dilations with the sphere of radius one. This results in a dilation not being perfectly spherical, but polygonal (polyhedral). Secondly, the functions use a multi-scale approach, where the local thickness computed for an image of a halved size, and then upscaled. The default number of scales (corresponding to the number of times an image is halved) is set to 3.

III. Related Work

IV. Complexity

Consider a 3D object occupying a cube as in Fig. 1. Denote the the side length of the cube as X and the largest local thickness of the object as S. If the structure is imaged such that length X is divided in x pixels, the thickness of the object in pixels is $s = \frac{S}{X}x$. The conventional computation of local thickness includes a loop

for r = 1 to s **do**

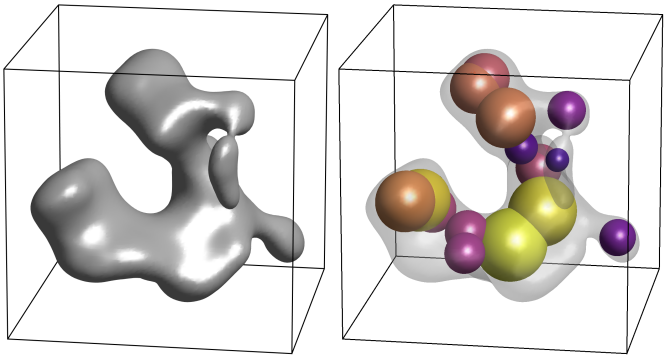

Fig. 1. *Left:* An example of a 3D structure of varying local thickness. *Right:* A few spheres fitting inside the object of changing radius, depending on the local thickness of the object.

volume size x^3	512^3	1024^3	2048^3	4096^3
processing time $\mathcal{O}(x^7)$	1 sec	2 min	4.5 h	24 days

TABLE I
The increase in processing time.

dilate(volume, sphere(r))

Computational complexity of dilation is linear in respect to the number of voxels to be processed, in this case x^3, and the size of the structuring element, here r^3. That is, dilation takes $\mathcal{O}(x^3r^3)$ per iteration. On the other hand, the number of iterations s grows linearly with x. The computational complexity of the loop is therefore

$$\mathcal{O}(x^3)\sum_{r=1}^{\mathcal{O}(x)}\mathcal{O}(r^3) = \mathcal{O}(x^3)\mathcal{O}(x^4) = \mathcal{O}(x^7)\,.$$

This leads to rapid increase in processing time as volume size grows, see Tab. I where we assume that it takes 1 second to process a volume of size 512^3.

VAND: Move after method. The computational complexity of our method is constant for the size of the structuring element 1^3, that is $\mathcal{O}(x^31^3)$ per iteration, therefore we obtain

$$\mathcal{O}(x^3)\sum_{i=1}^{\mathcal{O}(x)}1^3 = \mathcal{O}(x^3)\mathcal{O}(x) = \mathcal{O}(x^4)\,.$$

V. Method

For an object in 3D, the local thickness in any point of the object is defined as the radius of the largest sphere which fits inside the object and contains the point.

In other words, for an object $A \subset \mathbf{R}^3$ and a point $\mathbf{x} \in A$, local thickness

$$t(\mathbf{x}) = \max\{r \in \mathbf{R} \,:\, \mathbf{x} \in S(\mathbf{c}, r) \subseteq A,\, \mathbf{c} \in \mathbf{R}^3\}\,, \qquad (1)$$

where

$$S(\mathbf{c}, r) = \{\mathbf{p} \in \mathbf{R}^3 \,:\, ||\mathbf{c} - \mathbf{p}||_2^2 \leq r\}\,. \tag{2}$$

VAND: Trying to move from the definition to the computation. A more operative way of computing local thickness is via a distance field defined as

$$d(\mathbf{x}) = \max\{r \in \mathbf{R} \,:\, S(\mathbf{x}, r) \subseteq A\}\,. \tag{3}$$

For a point $\mathbf{x} \in S(\mathbf{c}, d(\mathbf{c}))$ we know that $t(\mathbf{x}) \geq d(\mathbf{c})$. So $t : \mathbf{R}^3 \to \mathbf{R}$ may be computed for all $\mathbf{x}$ as

$$t = \max\{d(\mathbf{c})\, I_{S(\mathbf{c},d(\mathbf{c}))},\, \mathbf{c} \in A\}\,. \tag{4}$$

where $I_S : \mathbf{R}^3 \to \mathbf{R}$ is an indicator function of any set $S \subset \mathbf{R}^3$.

VAND: Trying to get to the dilation. We also have

$$I_{S(\mathbf{c},d(\mathbf{c}))} = I_{\mathbf{c}} \oplus S_{d(\mathbf{c})}\,, \tag{5}$$

where $S_{d(\mathbf{c})}$ is a sphere of radius $d(\mathbf{c})$ centered at origin. VAND: Check whether dilation may be used on functions AND sets. Otherwise use indicator function for sphere too. So now we have

$$t = \max\{d(\mathbf{c})\, I_{\mathbf{c}} \oplus S_{d(\mathbf{c})},\, \mathbf{c} \in A\}\,. \tag{6}$$

Computation may now be performed by discretizing the distance field. If we consider only parts VAND: by saying 'parts', I'm again mixing functions and sets of d containing values larger than h, i.e.

$$d_h(\mathbf{x}) = \begin{cases} d(\mathbf{x}) & d(\mathbf{x}) \geq h \\ 0 & \text{elsewhere} \end{cases} \tag{7}$$

we know that

$$t \geq d_h \oplus S_h, \text{ for every } h \in \mathbf{R} \tag{8}$$

actually

$$t = \max\{d_h \oplus S_h,\, h \in \mathbf{R}\}\,. \tag{9}$$

This is an operative approach to local thickness leading to conventional local thickness algorithm, which operates on discretized distance field, Alg. 1.

VI. Algorithms

A. Idea

Normally, we dilate with binary discrete sphere of a large radius.

In discrete setting, a dilation with a large radius cannot be achieved by many dilations using smaller radii.

We approximate a disk using a non binary convex shape. And we utilize the property of distance field.

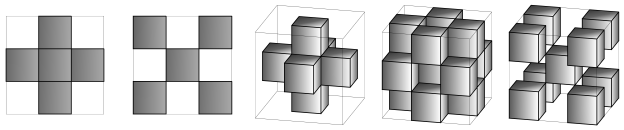

Fig. 2. Structuring elements for fast local thickness in 2D and 3D.

B. Structuring elements

VAND: Check, check, check!!! The structuring elements used for our algorithm have kernel size length of 3, so smallest possible kernel. The structuring elements are a discrete disc and annuli of growing radii in 2D. In 3D, it is spheres and spherical shells. We make sure that the central pixel (voxel) is always set to one. Algorithms are 3 and 4. Here disc(r) is structuring element with ones in all pixels with distance to kernel center being smaller or equal to r. VAND: Maybe rather in text? The resulting structuring elements may be seen in the Fig. 2.

The weights w_i determine the shape of the non-binary discrete approximation of the circle. We weight the structuring elements by the inverse of their radius. So disk($\sqrt{3}$) would be weighted proportional to $1/\sqrt{3}$. Since we normalize the weights to sum to 1, we arrive to expressions in 3 and 4. Our results show that this weighting yields a good approximation. Still, it is worth noticing that this weighing is slightly arbitrary. It turns out that weights affect the shape of the resulting approximation in nontrivial way, and the quality of the approximation is difficult to assess.

Algorithm 1 Conventional local thickness

```
out = sdf
for r = 1 to floor(max(sdf)) do
    selemdisk = disk(r)
    temp = dilate(sdf ⊙ (sdf >= r), selemdisk)
    change = temp > 0
    out[change] = temp[change]
```

Algorithm 2 Fast local thickness

```
out = sdf
selem_i = selems for 2D or 3D
w_i = weights for 2D or 3D
for r = 0 to floor(max(sdf)) do
    temp = sum_i(w_i dilate(sdf, selem_i))
    change = out > r
    out[change] = temp[change]
```

Algorithm 3 Selems and weights 2D

```
selem_1 = disk(1)
selem_2 = disk(√2) - disk(1) + disk(0)
w_1 = √2/(1+√2)
w_2 = 1/(1+√2)
```

Algorithm 4 Selems and weights 3D

1: selem_1 = disk(1)
2: selem_2 = disk($\sqrt{2}$) - disk(1) + disk(0)
3: selem_3 = disk($\sqrt{3}$) - disk($\sqrt{2}$) + disk(0)
4: $w_1 = \frac{\sqrt{6}}{\sqrt{6}+\sqrt{3}+\sqrt{2}}$
5: $w_2 = \frac{\sqrt{3}}{\sqrt{6}+\sqrt{3}+\sqrt{2}}$
6: $w_3 = \frac{\sqrt{2}}{\sqrt{6}+\sqrt{3}+\sqrt{2}}$

VII. Results

VIII. Conclusion

The conclusion goes here.