Face Index Map Generation

Prateep Mukherjee, Shireen Elhabian and Ross Whitaker

January 12, 2016

1 Method

1.1 Introduction

Shape analysis in medical imaging involves analyzing shapes and properties of anatomical parts of individual patients as well as over a large population. Thus, statistical shape modeling can be classified into two groups. First, when the only input to the system is a binary segmented volume with every voxel labeled [0,1], statistical analysis involves modeling the variations of postional data. Hence, this is called **positional** analysis. The second group is more general, where every individual voxel is associated with certain significant *attributes*, like thickness, scalar field, tensile strength etc. These attributes provide clinical information regarding stucture and morphology. Therefore, such a model is called **functional** analysis, since the image data is a function of certain attributes. Functional shape analysis evidently provides better and accurate medical treatment, as it jointly models the variations in shape as well as anatomical features of shape.

1.2 Goal

The aim of this method is to generate a feature volume(F), within a narrow-band, given an image domain(Ω) and a surface(M). M is parametrized by a set of vertices($\mathcal{V}$) and faces($\mathcal{F}$). Each vertex $v \in \mathcal{V}$ is associated with d attribute values($\mathcal{S} : \Re \rightarrow \Re^d$). The nearest vertex($v^*$) to a voxel is mathematically defined as follows:

$$v^* = \underset{v \in \mathcal{V}}{\operatorname{argmin}} \, (d_{\bar{x}',v}), \;\; \forall \bar{x}' \in \mathcal{N}$$

$$\mathcal{N} : \;\; \left\{ \bar{x} \mid \left\{ \begin{array}{ll} 1, & -k \leq (\bar{x} - \bar{v}) \leq k \\ 0, & otherwise \end{array} \right\}, \forall (\bar{x}, \bar{v}) \in \Omega \right\}$$

Here, $\mathcal{N}$ denotes the narrow-band, and $d_{\bar{x},v}$ denotes the *barycentric distance* for voxel $\bar{x}$ to vertex v. Therefore, the feature volume F, at each voxel, is simply defined as the feature attribute of its nearest face. That is,

$$F(\bar{x}) = \mathcal{S}(v^*), \quad \forall \bar{x} \in \Omega, v^* \in \mathcal{V} \tag{1}$$

Nomenclature

$\mathcal{F}$ faces $\in M$

$\mathcal{N}$ narrow-band $\in \Omega$

$\mathcal{S}$ set of attributes for each vertex

Ω image domain($\Re^3$)

$\mathcal{V}$ vertices $\in M$

F feature volume

M surface mesh

1.3 KD-tree

The easiest approach to solve this is to use k-nearest neighbor(kNN) metric. KD-trees[1] are often used to get nearest neighbors for a point in k-dimensional space. KD-trees recursively split the domain Ω in two parts, and searches over each half in order to get the nearest points. This divide and conquer approach provides $O(\log n)$ query time, which is a major improvement over linear search in real-time applications.

1.4 Potential problems

However, KD-trees do not have a prior knowledge of the topology. This fails to find accurately the triangle with minimum barycentric distance(Fig 1a). To solve this problem, we generate a list of candidate face indices within a radius(R), for every voxel(v). The winning face is selected from this list, which has the minimum distance from v. Each voxel is divided into a number of sub-voxels, and for every sub-voxel a candidate face is selected. This provides a set of candidate faces for every voxel. The flowchart for this method is described in Algorithm 1.

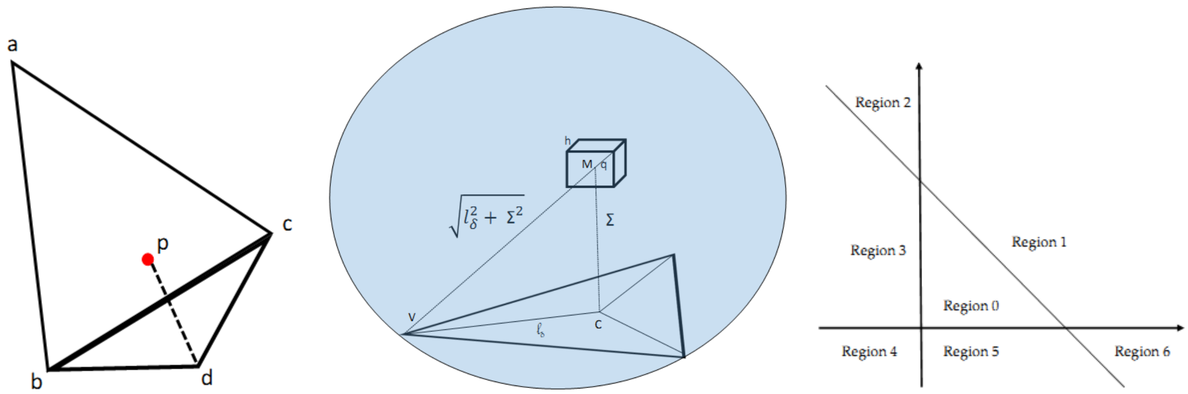
1.5 Choosing the search diameter

The maximum radius of search for a voxel is computed based on its distance to the isosurface($\S ==$ $igma$), the number of sub-voxels(h) and the maximum edge-length(s) of all faces($\mathcal{F}$). This is illustrated in Fig. 1b. In fig. 1b, q is the physical distance from the center of the voxel(M) to its corner: $q \sim h \times \sqrt{3}$. l_δ for a triangle is approximately equal to $\frac{s}{\sqrt{3}}$, assuming an equilateral $\triangle$. Also, $\triangle$ MVC, formed by the center of the voxel(M), centroid of a face(C) and one of its corner vertex(V), is a right-angled triangle. Therefore, using Pythagoras' theorem, the distance from M to V is $\sqrt{{l_\delta}^2 + \Sigma^2}$. Therefore, the maximum search radius $B = q + (\sqrt{{l_\delta}^2 + \Sigma^2})$.

1.6 Barycentric Distance from Point to $\triangle$

The problem is to compute the minimum distance between a point $\mathbf{P}$ and a triangle $\triangle(\alpha, \beta) = \alpha\mathbf{B} + \beta\mathbf{E}_0 + (1 - \alpha - \beta)\mathbf{E}_1$, for $\alpha, \beta\{\alpha, \beta : \alpha \in [0, 1], \beta \in [0, 1], \alpha + \beta \leq 1\}$. The minimum distance is computed by locating the values $(\bar{\alpha}, \bar{\beta})$ corresponding to the point on the triangle closest to $\mathbf{P}$. [2]

The squared-distance function for any point on the triangle to $\mathbf{P}$ is $Q(\alpha, \beta) = |\triangle(\alpha, \beta) - \mathbf{P}|^2$. Expanding the quadratic term, $Q(\alpha, \beta)$ can be written as:



(a) (b) (c)

Figure 1: **(a)** Bad case for KD-tree. Point p is nearest to vertex(d), while it lies inside △(abc); **(b)** Illustrates the computation of the search radius; **(c)** Partitioning the $\alpha\beta$-plane by triangle domain

$$Q(\alpha, \beta) = a\alpha^2 + 2b\alpha\beta + c\beta^2 + 2d\alpha + 2e\beta + f \tag{2}$$

where, $a = \mathbf{E}_0 \cdot \mathbf{E}_0$, $b = \mathbf{E}_0 \cdot \mathbf{E}_1$, $c = \mathbf{E}_1 \cdot \mathbf{E}_1$, $d = \mathbf{E}_0 \cdot (\mathbf{B} - \mathbf{P})$, $e = \mathbf{E}_1 \cdot (\mathbf{B} - \mathbf{P})$ and $f = (\mathbf{B} - \mathbf{P}) \cdot (\mathbf{B} - \mathbf{P})$.

The optimal values $\bar{\alpha}, \bar{\beta}$ are obtained using calculus of variations on Q. The gradients are $\nabla_\alpha Q = 2(a\alpha + b\beta + d)$, and $\nabla_\beta Q = 2(b\alpha + c\beta + e)$. To solve eq. 2, we divide the $\alpha\beta$ plane into 7 regions(Fig. 1c).

Suppose $(\bar{\alpha}, \bar{\beta})$ is in region 1. The level curves of Q are those curves in the $\alpha\beta$-plane for which Q is a constant. The graph of Q is a paraboloid. Hence, the level curves are ellipses. At the point where $\nabla Q = (0, 0)$, the level curve degenerates to a single point $(\bar{\alpha}, \bar{\beta})$, which is the global minimum. As the iso-values increase from here, the corresponding ellipses increase further away from $(\bar{\alpha}, \bar{\beta})$. There is a smallest iso-value V_0 for which the corresponding ellipse just touches the triangle domain edge $\alpha + \beta = 1$ at a value $\alpha = \alpha_0, \beta = 1 - \alpha_0$. For isovalues $V < V_0$, the corresponding ellipses do not intersect the triangle domain. For level values $V > V_0$, portions of the triangle domain lie inside the corresponding ellipses. In particular any points of intersection of such an ellipse with the edge must have a level value $V > V_0$. Therefore, $Q(\alpha, 1 - \alpha) > Q(\alpha_0, \beta_0)$, for $\alpha \in [0, 1]$, and $\alpha \neq \alpha_0$. The point α_0, β_0 provides the minimum squared distance between $\mathbf{P}$ and the triangle.

2 Algorithm

Following is a pseudo-code of the algorithm.

```
Data: Narrow Band(N_B), original volume(I), Mesh(M), s_v, S_v
Result: Map containing face indices for I
initialize Output to be same size as N_B
declare candidates
/*a map to store all candiate faces for every super-voxel */
initialize Thresh
I↓ ← downscale(I)
I↑ ← upscale(I)
for each voxel v in I↓ do
    p ← indexToPhysicalCoord (v)
    for each face f in M do
        d ← PointTriangleDistance (p,f)
        if d ≤ Thresh then
            candidates [v] ← f
declare faceIndexMap
/*a map to store all faces for every voxel */
for each voxel v in N_B do
    p↑ ← indexToPhysicalCoord (v)
    p↓ ← get position of p↑ in I↓
    v↓ ← physicalCoordToIndex (p↓)
    faceId ← PointTriangleDistance (p↓, candidates [v↓] )
    Output [v] ← faceId
    p ← get position of v in I
    V ← physicalCoordToIndex (p)
    faceIndexMap [V] ← faceId
```

Algorithm 1: getFaceIndexList

3 Results

Distance maps generated from our method and that using KD-trees are shown in Fig.2. It can be seen that distance transforms generated using our method is continuous and smooth, while that using KD-trees are more irregular.

References

[1] John Louis Bentley, 'Multidimensional binary search trees used for associative searching,' *Commun. ACM*, 18(9):509-517, 1975.

[2] David Eberly, 'Distance Between Point and Triangle in 3D,' *Geometric Tools, LLC*, 2008.

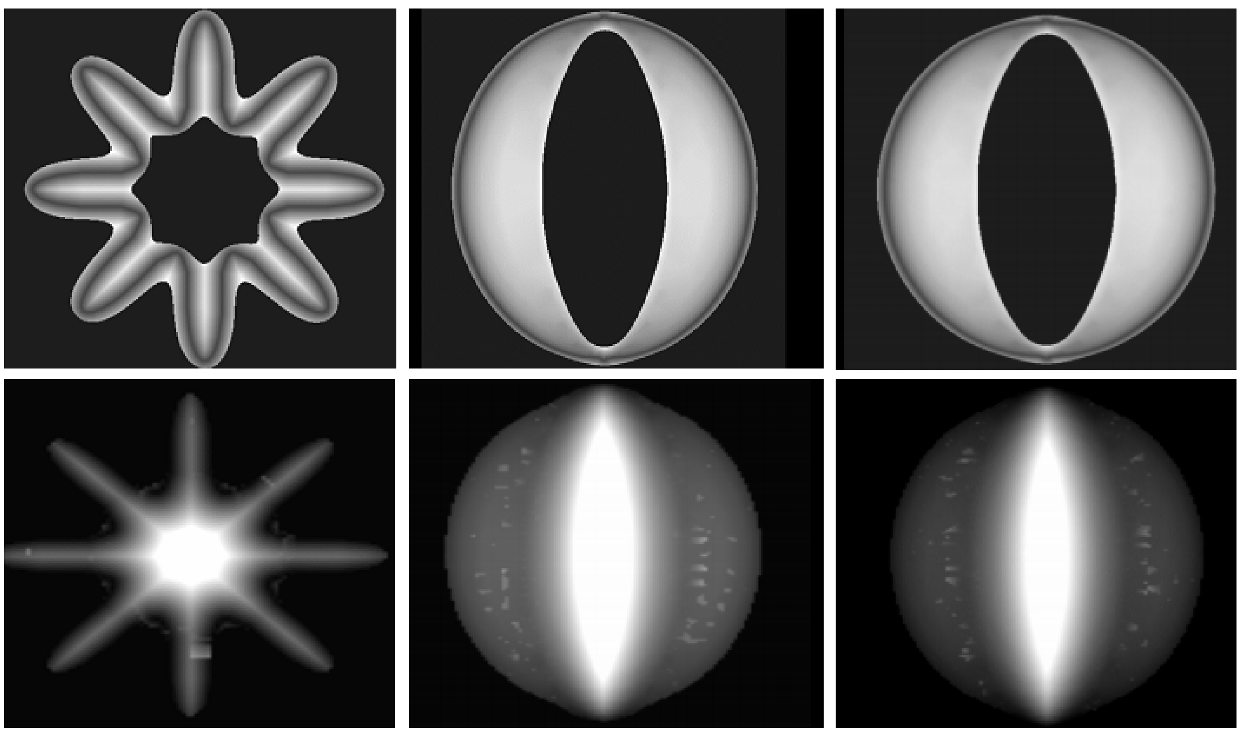


Figure 2: Distance map generated using our method(**Top**), and KD-tree(**Bottom**).