Meshless parameterization and surface reconstruction

Michael S. Floater and Martin Reimers

Abstract: This paper proposes a method called *meshless parameterization*, for parameterizing and triangulating "single patch" unorganized point sets. The points are mapped into a planar parameter domain by solving a sparse linear system. By making a standard triangulation of the parameter points, we obtain a corresponding triangulation of the original data set.

Key words: parameterization, triangulation, surface reconstruction, reverse engineering.

1. Introduction

One of the most critical tasks in reverse engineering is the organization of a given scattered data set, or point cloud, into some kind of topological structure, such as a triangulation; see [**17**] for an overview of reverse engineering. Several methods for triangulating unorganized points have been developed in recent years. For example, the methods of [**3**] and [**10**] are based on the idea of successively removing tetrahedra from a Delaunay tetrahedrization of the points, while the methods of [**16**] and [**1**] are based on properties of the Voronoi diagram. In [**9**] and [**2**] on the other hand, implicit methods are used to generate triangulations which approximate the data set.

This paper presents a simple new method for triangulating unorganized points which are assumed to be sampled from a single surface patch. The basic idea is to map the points into some convex parameter domain in the plane. We call this *meshless parameterization* since the mapping is independent of any given topological structure. Then, by triangulating the parameter points, we immediately obtain a corresponding triangulation of the original data set.

Many authors have proposed methods for parameterizing *organized* points, mapping them either into planar parameter domains [**4, 5, 6, 7, 8, 13, 14, 15**] or simplified mesh domains [**12**]. However, common to all these methods is the assumption that the points are structured in some kind of mesh. We propose parameterizing *unorganized* points by solving a global linear system which generalizes the method of [**5**]. The equations arise from demanding that each interior parameter point be some convex combination of some neighbouring ones. However, unlike in [**5**] where the data points are already triangulated, we use a heuristic to determine neighbourhoods.

As the figures in this paper show, our method yields good results in the numerical examples we have tested, even when there is considerable distortion or the data is noisy. Though the method is necessarily heuristic, heuristic input to the algorithm is minimal.

2. The Basic Method

Suppose we are given a sequence of distinct points $X = (x_1, \ldots, x_N)$ in $\mathbb{R}^3$, which are assumed to be sampled from a patch of some unknown surface in $\mathbb{R}^3$. By a surface patch we understand, as usual, a surface homeomorphic to a disc in $\mathbb{R}^2$. We wish to create a triangulation $\mathcal{T}$ of the point set. Our basic approach is to determine a corresponding

sequence of points $U = (u_1, \ldots, u_N)$ in $\mathbb{R}^2$ and triangulate these with a triangulation $\mathcal{S}$. Then we take $\mathcal{T}$ to be the corresponding triangulation of X, in other words, we take $\mathcal{T}$ to be the set of triangles $[x_i, x_j, x_k]$ for which $[u_i, u_j, u_k]$ is a triangle in $\mathcal{S}$. We can view the set of points in $\mathcal{T}$ as the image of the piecewise linear mapping $\psi : D \to \mathbb{R}^3$, where $D \subset \mathbb{R}^2$ is the union of the triangles in $\mathcal{S}$, ψ is linear over each triangle in $\mathcal{S}$, and $\psi(u_i) = x_i$ for all $i = 1, \ldots, N$. We can take $\mathcal{S}$ to be a standard triangulation such as a Delaunay or data-dependent one, the data here being the point set X.

The crucial task is to determine the set of parameter points U, and intuitively we would like U to mimic the geometry of the set X, in the sense that two parameter points u_i and u_j ought to be close whenever the two data points x_i and x_j are close. We propose a method for determining U which generalizes the convex combination method of [**5**], the only assumption being that one can identify a sequence of points in X to serve as the boundary of the triangulation $\mathcal{T}$. We will discuss later in the paper how one might identify such a boundary. Thus we assume now that the set X can be split into two disjoint subsets: X_I, the set of interior points, and X_B, the set of boundary points. Without loss of generality we may assume that $X_I = \{x_1 \ldots, x_n\}$ for some n, and $X_B = \{x_{n+1} \ldots, x_N\}$, where the points $x_{n+1} \ldots, x_N$ are ordered consecutively along the boundary.

The method has two steps. In the first step we map the boundary points $x_{n+1}, \ldots, x_N$ into the boundary of some convex polygon D in the plane. Thus we choose the corresponding parameter points $u_{n+1}, \ldots, u_N$ to lie around ∂D in some anticlockwise order, say. We could for example take $u_{n+1}, \ldots, u_N$ to lie on the unit circle or unit square and we could determine the distribution of $u_{n+1}, \ldots, u_N$ along ∂D by some standard polygonal parameterization, such as uniform or chord length.

In the second step, we choose for each interior point $x_i \in X_I$, a *neighbourhood* $\{x_j : j \in N_i\}$, a set of points in $X \setminus \{x_i\}$, which are in some sense close by. These could be for example the d nearest points to x_i for some suitable d. This and other choices of neighbourhood will be discussed later. We then choose a set of (strictly) positive weights λ_{ij}, for $j \in N_i$, such that

$$\sum_{j \in N_i} \lambda_{ij} = 1.$$

Then, in order to find the n parameter points $u_1, \ldots, u_n \in \mathbb{R}^2$ corresponding to the interior points $x_1, \ldots, x_n \in \mathbb{R}^3$, we solve the linear system of n equations

$$u_i = \sum_{j \in N_i} \lambda_{ij} u_j, \qquad i = 1, \ldots, n. \tag{2.1}$$

These equations demand that each interior u_i be some convex combination of its neighbours u_j, $j \in N_i$. Thus u_i will be contained in the convex hull of its neighbours.

We note that by letting $\lambda_{ij} = 0$ for $i = 1, \ldots, n$ and $j \notin N_i$, we may write the linear system (2.1) in the form

$$A\mathbf{u} = \mathbf{b} \tag{2.2}$$

where $A = (a_{ij})$ is the square $n \times n$ matrix with $a_{ii} = 1$ and $a_{ij} = -\lambda_{ij}$ for $i \neq j$, $\mathbf{u}$ is the

column vector $(u_1, \ldots, u_n)^T$, and $\mathbf{b} = (b_1, \ldots, b_n)^T$ is the column vector with

$$b_i = \sum_{j=n+1}^{N} \lambda_{ij} u_j.$$

3. The Linear System

In this section, we derive, in Proposition 3.3, a weak sufficient condition for when the linear system (2.1) is uniquely solvable. Roughly speaking, solvability occurs when the neighbourhoods N_i are large enough. On the other hand, we intuitively require that the neighbourhoods be small enough that the points in each neighbourhood are close to lying in a plane; at least they should not contain points from 'external branches' of the underlying surface. We have found in numerical examples that provided the points are sampled densely enough from the underlying surface, one can find a good compromise between the two competing requirements.

In order to study the linear system (2.1), we introduce some graph theoretic notation. The vertices u_i in (2.1) and the neighbourhoods N_i define a directed graph G. The nodes $V(G)$ of the graph are the indices $i \in I = \{1, \ldots, N\}$ and the directed edges $E(G)$ are ordered pairs (i, j) for distinct $i \in I_I = \{1, \ldots, n\}$ and $j \in I$ such that $j \in N_i$. We note that, unlike in [**5**] and [**6**], the graph G will not in general be planar. We say that j is a neighbour of i if (i, j) is a directed edge.

By a (directed) *path* from $i \in I_I$ to $j \in I$, $j \neq i$, we mean a sequence of nodes

$$i = p_1, \ldots, p_m = j$$

such that (p_k, p_{k+1}) is a directed edge for $k = 1, \ldots, m-1$. We will be interested in paths from 'interior' nodes $i \in I_I$ to 'boundary' nodes $j \in I_B = \{n+1, \ldots, N\}$. We define for a node $i \in I_I$ its *reachable boundary* $R_i \subset I_B$ as the set of nodes j in I_B that can be reached by a path from i.

Proposition 3.1. *Suppose points $u_1, \ldots, u_n$ in $\mathbb{R}^2$ satisfy equations (2.1). Then each point is contained in the convex hull of its reachable boundary,*

$$u_i \in CH(\{u_j : j \in R_i\}). \tag{3.1}$$

We will establish this inductively and to this end it helps to remove some interior point u_r from the linear system (2.1) and show that the remaining points satisfy a reduced set of equations with similar properties.

Lemma 3.2. *If $u_1, \ldots, u_n$ satisfy the equations in (2.1) then $u_1, \ldots, u_{r-1}, u_{r+1}, \ldots, u_n$ satisfy the equations*

$$u_i = \sum_{j \in \hat{N}_i} \hat{\lambda}_{ij} u_j \tag{3.2}$$

for $i \in I_I \setminus \{r\}$, *where* $\hat{\lambda}_{ij} > 0$ *for* $j \in \hat{N}_i$ *and*

$$\sum_{j \in \hat{N}_i} \hat{\lambda}_{ij} = 1 \tag{3.3}$$

and

$$\hat{N}_i = \begin{cases} N_i & r \notin N_i, \\ (N_i \cup N_r) \setminus \{i, r\} & r \in N_i, \end{cases}$$

and

$$\hat{\lambda}_{ij} = \begin{cases} \lambda_{ij} & r \notin N_i, \\ \frac{\lambda_{ij} + \lambda_{ir}\lambda_{rj}}{1 - \lambda_{ir}\lambda_{ri}} & r \in N_i. \end{cases}$$

Proof: If $r \notin N_i$ then clearly u_r does not occur in equation (2.1) and so in this case equations (3.2) and (3.3) hold with $\hat{N}_i = N_i$ and $\hat{\lambda}_{ij} = \lambda_{ij}$. Otherwise, $r \in N_i$ and we remove u_r from the right hand side of (2.1),

$$\begin{aligned} u_i &= \sum_{j \in N_i} \lambda_{ij} u_j = \sum_{j \in N_i \setminus \{r\}} \lambda_{ij} u_j + \lambda_{ir} \sum_{j \in N_r} \lambda_{rj} u_j \\ &= \lambda_{ir}\lambda_{ri} u_i + \sum_{j \in (N_i \cup N_r) \setminus \{i,r\}} (\lambda_{ij} + \lambda_{ir}\lambda_{rj}) u_j, \end{aligned}$$

giving

$$u_i = \sum_{j \in \hat{N}_i} \frac{\lambda_{ij} + \lambda_{ir}\lambda_{rj}}{1 - \lambda_{ir}\lambda_{ri}} u_j = \sum_{j \in \hat{N}_i} \hat{\lambda}_{ij} u_j,$$

where $\hat{\lambda}_{ij} > 0$ for $j \in \hat{N}_i$. We also have

$$\sum_{j \in \hat{N}_i} \hat{\lambda}_{ij} = \sum_{j \in \hat{N}_i} \frac{\lambda_{ij} + \lambda_{ir}\lambda_{rj}}{1 - \lambda_{ir}\lambda_{ri}} = \frac{1 - \lambda_{ir} + \lambda_{ir}(1 - \lambda_{ri})}{1 - \lambda_{ir}\lambda_{ri}} = 1,$$

which establishes (3.2) and (3.3). ■

In the same way as we associated the directed graph G with the linear system (2.1), we can associate a directed graph $\hat{G}$ with the reduced linear system (3.2), with nodes $V(\hat{G}) = \{1, \ldots, r-1, r+1, \ldots, N\}$. Clearly, due to the definition of $\hat{N}_i$, the set of directed edges $E(\hat{G})$ is formed by removing from $E(G)$ all its directed edges having r as an endpoint and then adding to it ordered pairs of the form (i, j), $i, j \neq r$, whenever (i, r) and (r, j) were directed edges of G.

Proof of Proposition 3.1: The proof is by induction on the number of interior vertices $n = |I_I|$. For $n = 1$, equation (3.1) follows trivially from (2.1). Now suppose that $n > 1$ and that the proposition holds when n is replaced by $n - 1$. Choosing any i in I_I, we will show that (3.1) holds by removing any other interior point u_r ($r \neq i$) from the equations (2.1). Applying Lemma 3.2, and since $\hat{G}$ contains only $n - 1$ interior nodes, we have from the induction hypothesis that

$$u_i \in CH(\{u_j : j \in \hat{R}_i\}),$$

where $\hat{R}_i$ is the reachable boundary of u_i in $\hat{G}$.

The proof will be complete if we can show that $\hat{R}_i \subset R_i$. Indeed, let j be a node in $\hat{R}_i$ and let $i = p_1, \ldots, p_m = j$ be a path connecting i to j in $\hat{G}$. We build a path from i to j in G as follows. For each k, if (p_k, p_{k+1}) is not a directed edge in G, then replace it by the two ordered pairs (p_k, r) and (r, p_{k+1}), which must be directed edges in G by the construction of $\hat{G}$. Thus $j \in R_i$ and so $\hat{R}_i \subset R_i$. ■

Using Proposition 3.1 we can derive a very weak sufficient condition for when the equations (2.1) are uniquely solvable. We will say that an interior vertex $i \in I_I$ is *boundary connected* if its reachable boundary R_i is nonempty.

Proposition 3.3. *Suppose all interior vertices $u_1, \ldots, u_n$ in (2.1) are boundary connected. Then the linear system (2.1) has a unique solution and moreover every u_i is contained in D.*

Proof: We first show that the matrix A in (2.2) is nonsingular. To this end we will show that $Au = 0$ implies that $u = 0$ where u is any column vector (of scalars) of length n. If we let $u_{n+1} = \ldots = u_N = 0$, then $u_1, \ldots, u_n$ satisfy equations (2.1) where each point u_i is now a value in $\mathbb{R}$. Proposition 3.1 clearly applies to points u_i of any dimension and in particular points in $\mathbb{R}^1$ and so each unknown $u_i, i \in I_I$ belongs to $CH(\{u_j : j \in R_i\})$. But since R_i is non-empty, and $u_j = 0$ for all $j \in R_i$, it follows that $u_i = 0$. Thus $u = 0$ and A is nonsingular.

Finally, since $R_i \subset I_B$, Proposition 3.1 shows that

$$u_i \in CH(\{u_j : j \in I_B\}),$$

and since D is convex, $CH(\{u_j : j \in I_B\}) = D$. ■

4. Neighbourhoods and Weights

We next describe three concrete choices of neighbourhood N_i and weights λ_{ij} for each interior point x_i which we have tested numerically.

Method 1. *Let N_i be the* `ball neighbourhood`

$$N_i = \{j : 0 < ||x_j - x_i|| < r_i\}, \tag{4.1}$$

for some radius $r_i > 0$ and let the λ_{ij} be the `uniform weights`

$$\lambda_{ij} = 1/d_i,$$

where $d_i = |N_i|$.

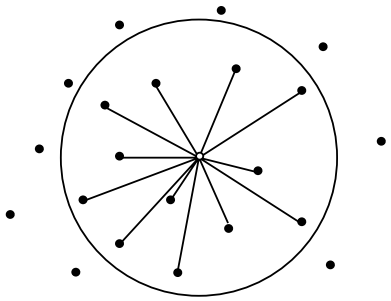


Figure 1. A ball neighbourhood

Figure 1 shows a ball neighbourhood. In our numerical examples we achieved good results by taking r_i to be constant, and we would expect this to be adequate when the distribution of the points is reasonably uniform. However, if the density of the point set X varies, it might be preferable to let r_i depend on the local density of X. Though it requires more CPU time, a good alternative would be to fix d_i; in other words, let each neighbourhood $\{x_j : j \in N_i\}$ be the set of d nearest points to x_i, for a constant d.

The danger of method 1 is that due to the uniform weights, some points x_i and x_j can be mapped to the same parameter point, as the following proposition shows.

Proposition 4.1. *Suppose that $\lambda_{ij} = 1/d_i$, where $d_i = |N_i|$. If two interior points x_i and x_k have the property that*

$$N_i \cup \{i\} = N_k \cup \{k\}, \tag{4.2}$$

then $u_i = u_k$.

Proof: Since $d_i = d_k$, we let $d = d_i = d_k$ and find from (2.1) that

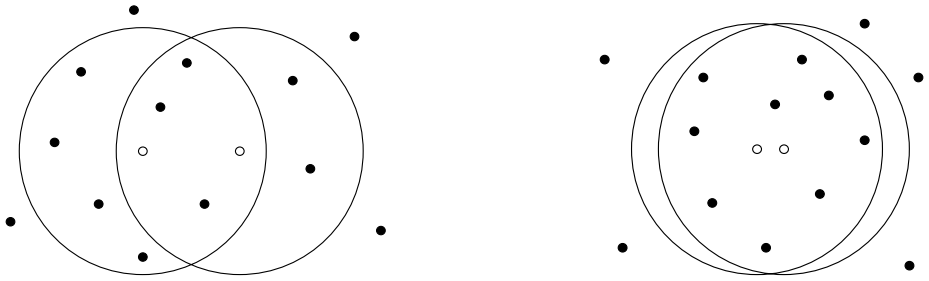
$$\left(1 + \frac{1}{d}\right) u_i = \frac{1}{d} \sum_{j \in N_i \cup \{i\}} u_j = \frac{1}{d} \sum_{j \in N_k \cup \{k\}} u_j = \left(1 + \frac{1}{d}\right) u_k.$$

■

If N_i is taken to be the ball neighbourhood (4.1), equation (4.2) is equivalent to the situation that

$$B(x_i, r_i) \cap X = B(x_k, r_k) \cap X. \tag{4.3}$$

This can occur, especially when x_i and x_k are close together. Figure 2 shows two possible pairs of ball neighbourhoods. In Figure 2a the two balls contain different sets of data points, so condition (4.3) does not hold. In contrast, in Figure 2b, the two balls share the same set of data points, so (4.3) does hold and the two points will be mapped to the same parameter point.



Figures 2a and 2b. Two neighbourhoods can coincide.

This drawback motivates a choice of weights which depends on the distances of the neighbours to x_i.

Method 2. *Let*

$$N_i = \{j : 0 < ||x_j - x_i|| < r_i\},$$

for some radius $r_i > 0$ *and let the* λ_{ij} *be the* `reciprocal distance weights`

$$\lambda_{ij} = \frac{1}{||x_j - x_i||} \Big/ \sum_{k \in N_i} \frac{1}{||x_k - x_i||}.$$

These weights generalize those used in chord length parameterization of polygonal curves (where every vertex has two neighbours); see [**5**]. Though we are not yet able to give a theoretical justification, Method 2 has yielded distinct parameter points u_i and a well-behaved triangulation in all the numerical exmaples we have run so far.

Our third method attempts to optimize the choice of neighbourhood by gathering a small number of points which 'surround' the point x_i, rather like the given neighbourhood one would have if the points were already triangulated. This method may be more appropriate when the points are unevenly distributed locally. For example, when the data is 'track-like', a ball will capture too many neighbouring points in one direction and not enough in the other.

Method 3. *Collect all points* x_j *in some large ball around* x_i*, fit a least squares plane, and project the points onto that plane, yielding new points* $P(x_j)$*. Triangulate the projected points* $P(x_j)$ *with a Delaunay triangulation* $\mathcal{T}_i$*. Then let* N_i *be the* `Delaunay neighbourhood`

$$N_i = \{j : P(x_i) \text{ and } P(x_j) \text{ are neighbours in } \mathcal{T}_i\}, \tag{4.4}$$

and let the λ_{ij} *be the* `shape-preserving weights` *of* [**5**].

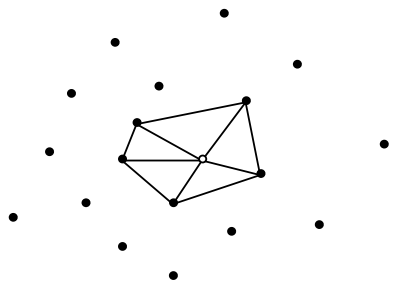


Figure 3. A Delaunay neighbourhood

Figure 3 shows a Delaunay neighbourhood. As long as the ball is large enough and the least square fit is good, we would expect $P(x_i)$ to be an interior point of the triangulation $\mathcal{T}_i$. If not, then x_i should be a boundary point of the data set X. This suggests an idea for locating boundary points (prior to parameterizing the interior points) which is discussed further in Section 5.

We choose the shape-preserving weights in Method 3 because the neighbours are now ordered, and, generalizing an argument in [**5**], the parameterization will have the reproduction property: if the points x_i all lie in a plane, the whole parameterization will be an affine mapping, provided the boundary is mapped affinely. We note that in [**6**] it was shown that the harmonic map of [**4**] also has the reproduction property, and that in many numerical examples the result is very similar. However, since the weights of the harmonic map can be negative, the latter map is not guaranteed to avoid foldover; see [**6**].

Finally, we remark that a special case of the linear system (2.1) is obtained from minimizing a function. Suppose that the neighbourhoods have the property that

$$j \in N_i \qquad \text{if and only if} \qquad i \in N_j, \tag{4.5}$$

for $i, j \in \{1, \ldots, n\}$. This implies that G can be treated as an undirected graph. If we let $w_{ij} = w_{ji}$ be some (strictly) positive value for each pair $(i, j) \in E(G)$, and we minimize the function

$$F(u_1, \ldots, u_n) = \sum_{(i,j) \in E(G)} w_{ij} ||u_i - u_j||^2,$$

keeping $u_{n+1}, \ldots, u_N$ fixed, then the solutions $u_1, \ldots, u_n$ satisfy the system (2.1) with

$$\lambda_{ij} = \frac{w_{ij}}{\sum_{k \in N_i} w_{ik}}.$$

For example, if N_i is the ball neighbourhood of *constant* radius r, then condition (4.5) holds and solving (2.1) with Methods 1 and 2 is equivalent to minimizing the functions

$$\sum_{0 < ||x_i - x_j|| < r} ||u_i - u_j||^2 \qquad \text{and} \qquad \sum_{0 < ||x_i - x_j|| < r} \frac{||u_i - u_j||^2}{||x_i - x_j||},$$

respectively.

5. Numerical Examples

We have applied Methods 1, 2, and 3 to three example data sets: (1) a foot, (2) Spock's head, and (3) a bunny. All three examples are quite severe tests as there is necessarily considerable distortion when mapping them into a plane.

In each example we needed to first identify a natural boundary and order the boundary points. We chose three different approaches in the three examples. In the foot example, we chose a suitable boundary by hand and in the Spock example, a natural ordered boundary was already part of the given data set.

In the bunny example, we applied a more sophisticated method, consisting of two distinct steps. In the first step we identified each point of the data set as being either interior or boundary. We identified x_i as a boundary point if the projected point $P(x_i)$ in (4.4) lay on the boundary of the local triangulation $\mathcal{T}_i$. Otherwise we regarded x_i as an interior point.

In the second step we ordered the boundary points. It was straightforward to divide the set of boundary points into two halves using a suitable plane. Then for the boundary points in each half space, two end points were identified. We then ordered each of the two sets of boundary points by applying the univariate analog of our triangulation method, that is, we solved a linear system identical to (2.1) (except the points u_i are now real values), in order to map the boundary points into the unit interval $[0, 1]$. The ordering of the parameter values was used as the ordering of the corresponding boundary points. Figure 4 shows the two half sets of boundary points in the bunny example and their

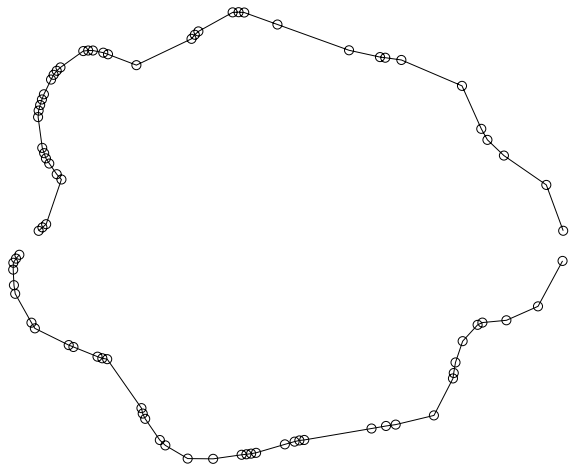


Figure 4. Ordering the bunny boundary

orderings generated by our method. Note that the two polygonal arcs are curves in $\mathbb{R}^3$ although they look planar in the figure.

For general data sets, we propose using this latter method for finding an order boundary, but it does require some user interaction.

In all the numerical examples, we took the parameter domain D to be the unit disc and we mapped the polygonal boundary X_B into the boundary of D according to chord length.

In Table 1 we indicate the dimensions of the data sets in the x, y, and z coordinates and the constant radius we chose both for the ball neighbourhood (4.1) and the ball for collecting points for projection for the Delaunay neighbourhood (4.4). A simple cubical cell structure was used to locate all points in each ball efficiently.

We found that the results of Method 1 confirmed Proposition 4.1: two data points x_i and x_k can be mapped to the same parameter point. This occurred in the foot and bunny examples: in the foot, 5092 data points were mapped to only 5091 distinct parameter points, and in the bunny the 30571 points were mapped to only 30568 distinct parameter points. Thus when using ball neighbourhoods, the uniform weights are unreliable and we recommend the reciprocal distance weights instead: Method 2.

When using either Method 2 or Method 3, we found in all three test examples that all parameter points were distinct. Even so, for the foot, Spock, and the bunny, the smallest distance between pairs of parameter points was of the order of 10^{-5}, 10^{-4}, and 10^{-7}, respectively, so there is considerable distortion in all three mappings.

Tables 2 and 3 show the results of Methods 2 and 3 respectively. The first two columns show the smallest and largest neighbourhoods N_i for each data set for the two methods (ball neighbourhood and Delaunay neighbourhood). We see that the Delaunay method is more effective at locating a small neighbourhood N_i. This means a sparser matrix A since the number of non-zero elements in the i-th row of A is $|N_i \setminus I_B| + 1$. Similar to [**5**], the

Bi-CGSTAB iterative method was applied to solve the two separate systems

$$Au^1 = b^1, \qquad Au^2 = b^2.$$

Tables 2 and 3 show the number of iterations and CPU times for each of the two component systems.

Interestingly, there is a substantial increase in the number of iterations when changing from Method 2 to Method 3, suggesting an increase in the condition number of the matrix A. On the other hand, the CPU times for Method 3 are lower than for Method 2, due to the matrix in the Delaunay case being sparser. In fact we also tested the reciprocal distance weights with the Delaunay neighbourhoods and obtained almost identical results to Table 3 so the difference in numbers of iterations seems to come from the size of the neighbourhoods.

We note that building the Delaunay neighbourhoods, and consequently the matrix A, required considerable CPU time. In fact this time was greater than that used to solve the two systems in the foot example, but somewhat less for Spock and the bunny.

Data set	N	x dim	y dim	z dim	r
Foot	5092	98.7	245	203	9.0
Spock	9508	0.69	0.80	0.68	0.03
Bunny	30571	0.16	0.15	0.12	0.003

Table 1. Dimensions of the data sets

Data set	$\min_i \lvert N_i \rvert$	$\max_i \lvert N_i \rvert$	Num. iterations	CPU time
Foot	6	27	137/150	2.08/2.30
Spock	3	29	180/170	6.81/6.36
Bunny	4	24	466/406	52.71/45.92

Table 2. Results for Method 2

Data set	$\min_i \lvert N_i \rvert$	$\max_i \lvert N_i \rvert$	Num. iterations	CPU time
Foot	3	9	230/240	0.84/0.87
Spock	3	9	303/286	4.52/4.45
Bunny	3	9	635/589	40.44/37.57

Table 3. Results for Method 3

For the three data sets, we found that both Methods 2 and 3 yielded visually pleasing triangulations $\mathcal{T}$. However, for the foot and Spock data sets, Method 3 yielded somewhat better shaped triangles, while for the bunny, Method 3 suffered from the lack of density of

points in the ears and so Method 2 yielded the best result (it has less long, thin triangles). The results are displayed in Figures 5 to 7. Figures 5a to 5d show respectively: (a) the set of parameter points U of X, using Method 2, and their Delaunay triangulation, and (b) the resulting triangulation $\mathcal{T}$ of X. Figures 5c and 5d show the corresponding results of using Method 3. Figures 6a to 6d show corresponding results for the Spock data set. Figures 7a to 7b show the corresponding results for Method 2 for the bunny and Figure 7c shows a close up of the triangulation around the ears.

Finally, we added artificial noise to the Spock data set to test the robustness of Method 2. Figure 8 illustrates how Method 2 succeeded in generating a satisfactory triangulation (of a somewhat older-looking Spock).

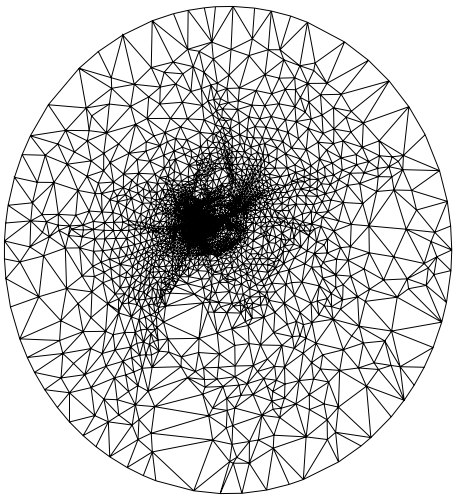


Fig. 5a Parameterization, method 2

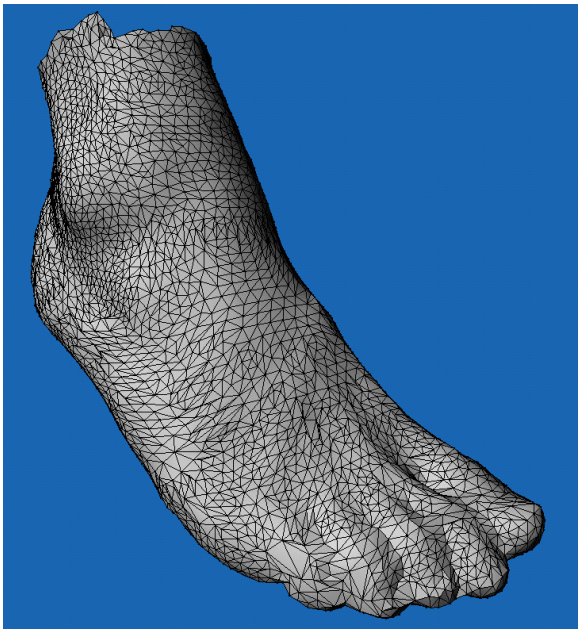


Fig 5b. 3D triangulation

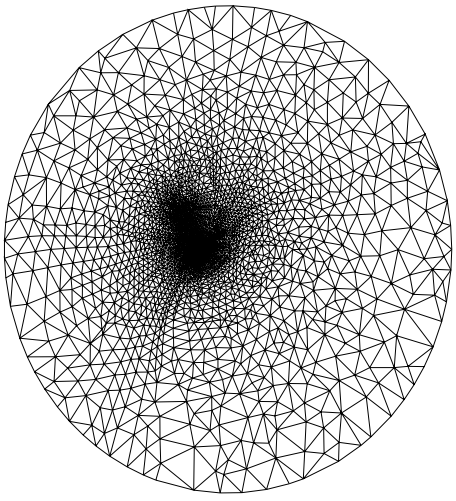


Fig 5c. Parameterization, method 3

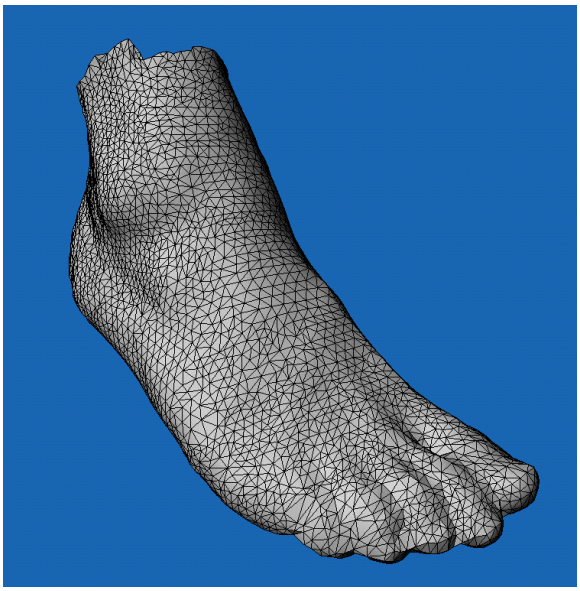


Fig 5d. 3D triangulation

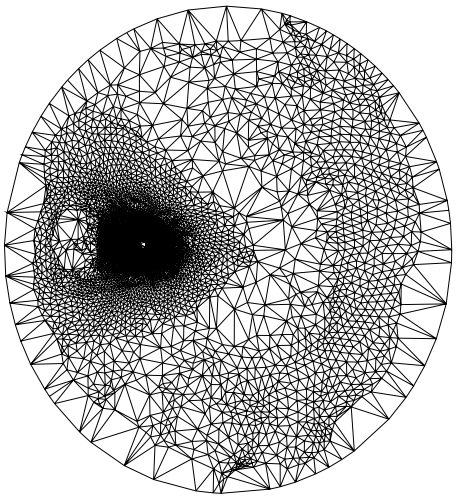


Fig 6a. Parameterization, method 2

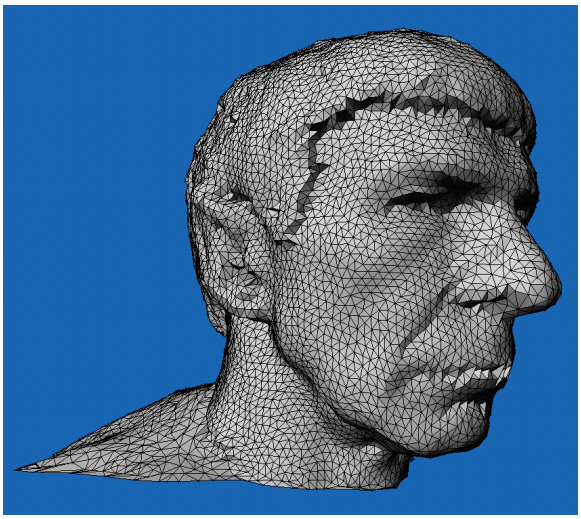


Fig 6b. 3D triangulation

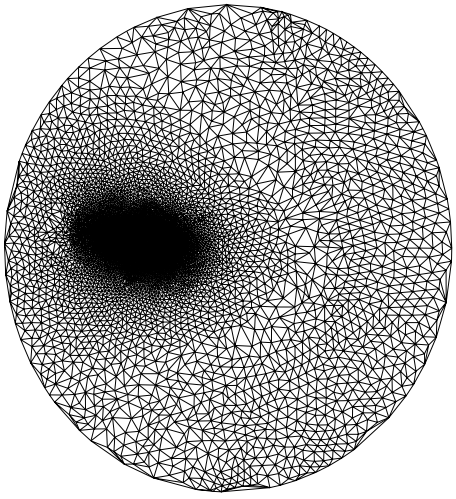


Fig 6c. Parameterization, method 3

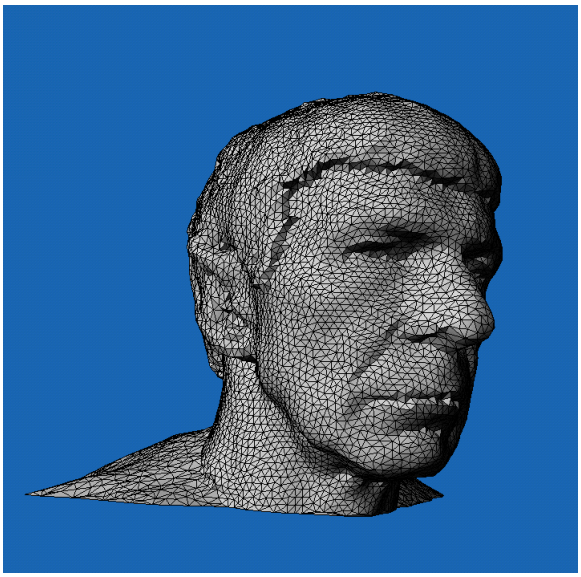


Fig 6d. 3D triangulation

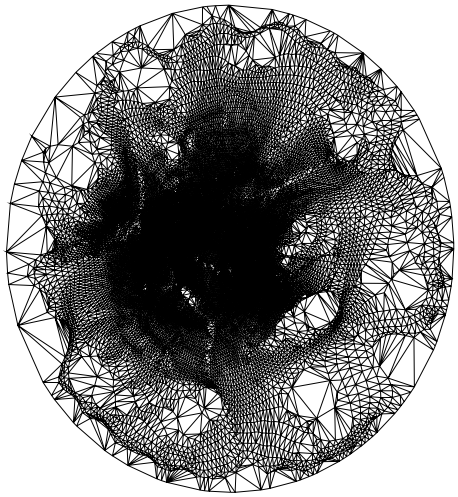


Fig 7a. Parameterization, method 2

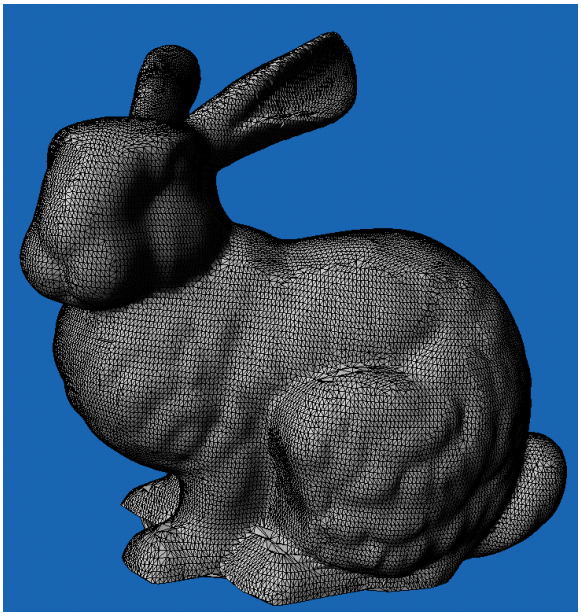


Fig 7b. 3D triangulation

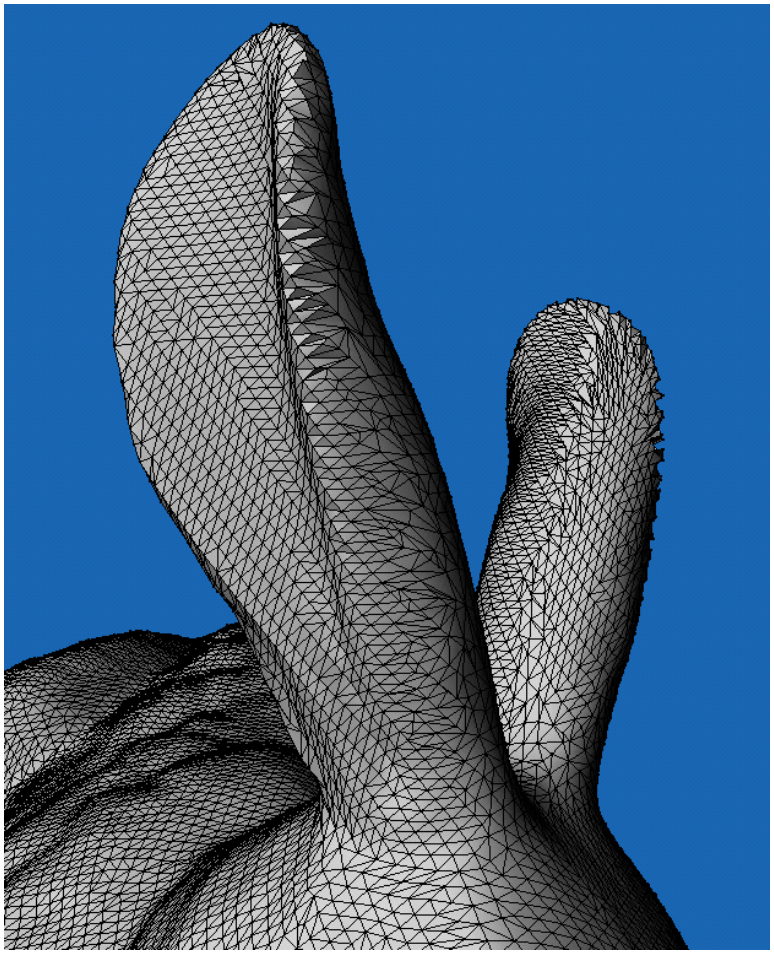


Fig 7c. Close up of 3D triangulation, method 2

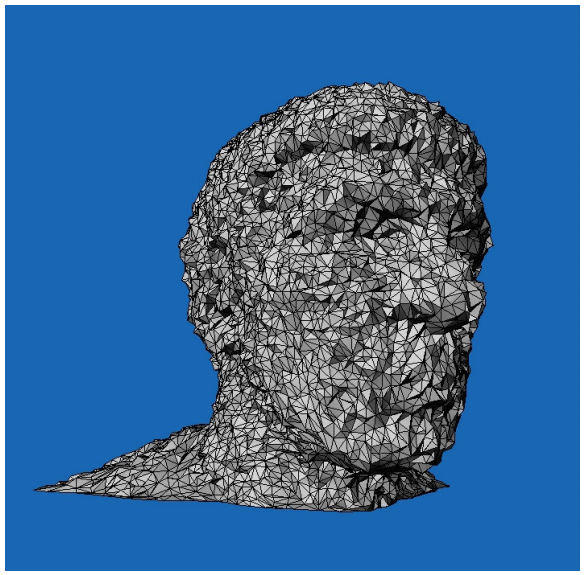


Figure 8. 3D triangulation of Old Spock, method 2

5. Conclusions

We have presented a simple method based on parameterization for triangulating unorganized points. Though further numerical tests and analysis are needed, we have found that Method 2 provides a simple, fast, and robust method. However, if the data sample is dense and free of noise, it appears that Method 3, though more expensive, yields triangulations with better shaped triangles in many cases.

Throughout the paper we have used a Delaunay triangulation of the parameter points. We believe, however, that just as for functional triangulations, it will often be preferable to triangulate the parameter points according to some criterion which depends on the geometry of the orginal data points: a data-dependent triangulation. Nevertheless, we believe the Delaunay triangulation of the parameter points will usually act as a good initial triangulation and that the desired triangulation can be reached by swapping (relatively few) triangles.

Finally, we note that meshless parameterization, preferably with the reproduction property as in Method 3, can also be used for interpolating or approximating the points with a smoother parametric surface, such as a spline over the domain triangulation, radial basis functions, or a tensor-product B-spline surface; see [**11**].

References

1. N. Amenta, M. Bern, and M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm, Computer Graphics Proceedings, SIGGRAPH 98 (1998), 415–421.
2. C. L. Bajaj, F. Bernardini, and G. Xu, Automatic reconstruction of surfaces and scalar fields from 3d scans, Computer Graphics Proceedings, SIGGRAPH 95 (1995), 109–118.

3. J.-D. Boissonnat, Geometric structures for three-dimensional shape representation, ACM Transactions on Graphics **3(4)** (1984), 266–286.
4. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, Computer Graphics Proceedings, SIGGRAPH 95 (1995), 173–182.
5. M. S. Floater, Parametrization and smooth approximation of surface triangulations, Comp. Aided Geom. Design **14** (1997), 231–250.
6. M. S. Floater, Parametric tilings and scattered data approximation, International Journal of Shape Modeling **4** (1998), 165–182.
7. G. Greiner and K. Hormann, Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville (1997), 163–172.
8. K. Hormann and G. Greiner, MIPS: An efficient global parametrization method, in *Curve and Surface Design*, P.-J. Laurent, P. Sablonnière, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville (2000), 153–162.
9. H. Hoppe, T. DeRose, T. DuChamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, Computer Graphics, Vol. 26, No. 2 (1992), 71–78.
10. F. Isselhard, G. Brunnett, and T. Schreiber, Polyhedral reconstruction of 3d objects by tetrahedral removal, Technical Report No. 288/97, Fachbereich Informatik, Univeristy of Kaiserslautern, Germany, 1997.
11. J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, AKPeters, Wellesley, 1994.
12. A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, MAPS: Multiresolution adaptive parameterization of surfaces, Computer Graphics Proceedings, SIGGRAPH 98 (1998), 95–104.
13. B. Lévy and J. L. Mallet, Non-distorted texture mapping for sheared triangulated meshes, Computer Graphics Proceedings, SIGGRAPH 98 (1998), 343–352.
14. W. Ma and J. P. Kruth, Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces, Computer-Aided Design **27** (1995), 663–675.
15. J. Maillot, H. Yahia, and A. Verroust, Interactive texture mapping, Computer Graphics Proceedings, SIGGRAPH 93 (1993), 27–34.
16. T. Schreiber and G. Brunnett, Approximating 3d objects from measured points, in *Proceedings of 30th ISATA*, Florence, Italy, 1997.
17. T. Varady, R. R. Martin, and J. Cox, Reverse engineering of geometric models — an introduction, CAD **29** (1997), 255–268.

Michael S. Floater and Martin Reimers
SINTEF
Postbox 124, Blindern
0314 Oslo, NORWAY
`mif@math.sintef.no`
`mre@math.sintef.no`