The Visual Computer manuscript No.
(will be inserted by the editor)

Adam Runions · Faramarz Samavati · Przemyslaw Prusinkiewicz

Ribbons

A representation for point clouds

Abstract Point clouds are usually represented either globally as surfaces or locally based on a small neighbourhood. We propose an intermediate representation called *ribbons* obtained by partitioning a point cloud into one dimensional strips. This representation is well suited to the placement of strokes for non-photorealistic purposes and can be rendered efficiently using quad strips. Methods for performing hatching, cross hatching, and silhouette renderings are presented. Ribbons also allow for the application of curve based operations to the point cloud, this is demonstrated using reverse subdivision.

Keywords Point Clouds · Point Based Rendering · Non-Photorealistic Rendering · Geometric Modeling

1 Introduction

The digitization of real world data often produces point clouds. The increasing availability of point cloud data has made it necessary to develop techniques to process and manipulate unorganized point cloud data. Over the past decade point based modeling has developed into a new paradigm in computer graphics

Point cloud research initially focused on converting point samples data to better understood representations, such as signed distance functions[9], polygon meshes[2], and projective surfaces[12, 1]; allowing for the application of existing techniques in mesh and implicit models to point clouds data. An alternative approach is to work with the point cloud directly, without imposing the topological and geometric constraints of other representations. Consequently, we can categorize most of the current representations to *global* or *local*. In *global* representation, the point cloud is modeled as a single surface[2, 7] as opposed to the *local* that rely entirely on the point as individual objects (perhaps using the k nearest neighbours for determining the normal)[22]. Although, the local representation allows for greater flexibility, its lack of suitable connectivity justifies the efforts that have been put for extracting global (surface) representations for many applications. In fact the selection of the representation (local versus global) is an important task and depends on the application and practical requirements. The spectrum from points to surfaces is wide enough to include other intermediate representations. Some of these representations might be best fit to particular applications. For example, feature lines (such as silhouette, curvature, etc) play an important role for artistic drawing of the object. These feature lines require a kind of connectivity beyond that of just point samples, but on the other hand a surface seems to be an over-representation for this task. In this work, we propose a new representation, *ribbons*, that lies somewhere between the extremes point-based and surface-based representations. This representation is particularly suitable for the applications such as non-photorealistic rendering that require working on lines on the object

Our method partitioned the point cloud into one dimensional long strips called *ribbons* by connecting local parameterizations computed at each point. The resulting representation can be computed efficiently and allows for the direct application of one dimensional operations (demonstrated via the application of forward and reverse subdivision (§4.1). Additionally, the ribbons may be rendered to visualize the surface using standard OpenGL calls to obtain standard OpenGL (§3.3) and non-photorealitic (NPR) renderings (§4.2)).

(A more accurate RoadMap) The structure of the paper is as follows: Section 2 provides a discussion

Adam Runions
2500 University Dr. NW
Calgary AB, Canada
T2N 1N4
Tel.: +1-403-210-9498
E-mail: runionsa@cpsc.ucalgary.ca

Faramarz Samavati
Tel. :
Fax :
E-mail: samavati@cpsc.ucalgary.ca

Przemyslaw Prusinkiewicz Tel. :
Fax :
E-mail: pwp@cpsc.ucalgary.ca

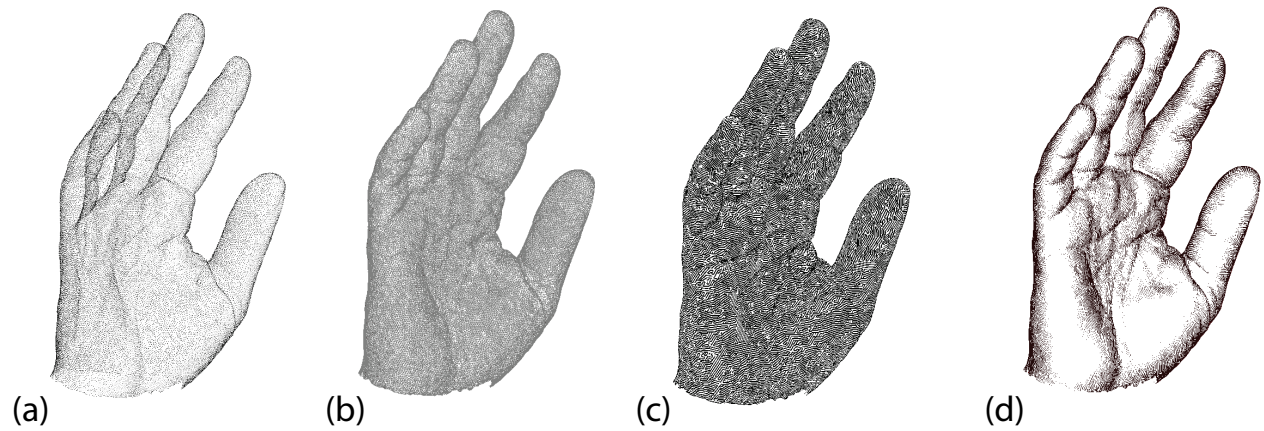

Fig. 1 Outline of system a.) Input consists of an initial point cloud b.) Local connectivity computed using WPCA c.) Ribbons generated by following local connectivity d.) Render ribbons directly

of related literature, Section 3 discuses the methodology applied to construct and render ribbons, the applications discussed above are outlined with results in Section 4 and Section 5 outlines future venues of exploration.

2 Preliminaries

2.1 Weighted Principal Component Analysis

Each ribbon is constructed by stitching together local parameterizations of the point cloud. Local parameterizations take the form of coordinate systems, computed at each point using the points *k nearest neighbours* (k-NN).

To include the impact of the distance of points, we employ a weighted form of principal component analysis[1,16,10]. Weighted principal component analysis (WPCA) can be derived from principal component analysis by replacing the mean M with the weighted mean, and the correlation between two variables with the weighted equivalent:

$$M_i = \sum_{i=1}^{m} \frac{x_i w(x_i)}{\sum_{i=1}^{m} w(x_i)}$$

$$cor(x_i, x_j) = \sum_{k=1}^{m} \frac{(x_{ik} - M_i)(x_{jk} - M_j)w(x_k)}{\sum_{i=1}^{m} w(x_i) - 1}$$

A WPCA estimate is said to be *centred on* $p \in \Re^3$ if p's neighbourhood is used as the sample and p is used in the weight function w. For a local weighted function, we employ the one that employed by Levin[12]

$$w(p_i) = e^{\frac{\|p - p_i\|}{H^2}}$$

where the local density estimate H is calculated as

$$H = \sum_{p_i \in N_p} \frac{\|p_i - p\|}{k}$$

where N_p consists of p's k-nearest neighbours (k-NN) in the point cloud.

The weighted principal components can be found by determining the eigenvectors of the covariance matrix C. Let $\lambda_1 \geq \lambda_2 \geq \lambda_3$ be the sorted eigenvalues of the covariance matrix, and α_{p1}, α_{p2}, α_{p3} the associated eigenvectors. Then α_{p1} is the direction of greatest variance in a neighbourhood of p, α_{p2} the direction of second greatest variance, and α_{p3} the direction minimizing variance[10].

As the point clouds being considered approximate a 3d surface these vectors have a meaningful geometric interpretation: α_{p1} and α_{p2} estimate the tangent plane and α_{p3} the normal to the point cloud at p. This estimates the orientation at each point, from which an ordering of the local neighborhood can be inferred. As such, these three vectors provide a local parameterization of the point cloud.

2.2 Reverse Subdivision

Sub-division is a method for taking an ordered set of points, usually along a curve, or surface, and replacing the points with a finer network of points. A multiresolution system was developed in [18] by reversing the rules of a sub-division filter. A short description of the framework presented by Samavati and Bartels follows.

Assume a set of ordered points describing a surface, or curve are given; these are called *fine points* and are denoted as F. We can use F in order to obtain a smaller number of points approximating the same curve or surface, these points are called *coarse points* and denoted C.

The goal of reverse sub-division is, given F and a sub-division filter P, to find a filter A that yields C when applied to F. Inorder to reproduce F a set of details D is needed, which is created using B derived from P, such

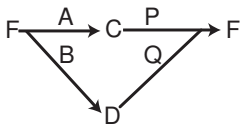

Fig. 2 Fine points F are decomposed into C and D using A and B respectively. D and C are then recombined into F using P and Q respectively.

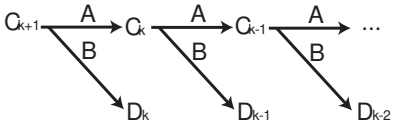

Fig. 3 Coarse points are successively decomposed into details and a new set of Coarser points to obtain a desired resolution, this process can be reversed as in fig.2 to return to a higher-resolution representation n

that

$$BF = D$$

The filter's P and Q then constitute the *reconstruction filter* and can be used to obtain F from C and D as follows

$$F = PC + QD$$

This discussion is summarized in fig.2. A multi-resolution system can be created by performing the decomposition into coarse points and details repeatedly (fig.3).

Implicit in the construction of reverse subdivision, and subsequently multi-resolution using reverse subdivision, is the assumption that some ordering of the fine points, coarse points, and details has been provided. A criteria that is not met in an unordered point cloud.

2.3 Non-Photorealistic Rendering

Placing strokes on surfaces is an important facet of *Non Photorealistic Rendering* (NPR). Strokes are used to provide visual cues to the viewer, conveying the structure of the surface in simple terms. Silhouettes, suggestive contours[6], and creases are often employed to this end.

Silhouettes are a well established NPR visualization technique[19,17]. In the context of meshes, frame coherence can be an issue. The work of Brosz et al[5] addresses this problem by varying the width and shading of silhouette lines based on the stability of each line.

The automatic generation of hatching and cross hatching on surface has also been examined outside the context of point clouds[17,8]. In the work of Pruan et al[17] and Hertzman et al[8] hatching consists of placing roughly parallel strokes on the surface whereas cross hatching is generated by placing two sets of parallel strokes perpendicular to each other.

The application of NPR to point clouds has been relatively unexplored. Kawata et al[11] proposes a technique for interactive point based painterly rendering. Xu et al [20] extract feature points but do not construct strokes, and Zakaria et al propose a method for dithering and silhouette extraction which creates strokes in image space[21]. In contrast to these works the NPR techniques proposed in this paper address the placement of strokes directly on the surface, and fit within the point cloud paradigm.

3 Generating Ribbons

Let P be a point cloud sampled from a 2d manifold surface embedded in $\Re^3$ (fig.1a). Ribbons partition the point cloud into one dimensional strips, maximizing the length of each ribbon and maintaining a relatively consistent direction improves the results of NPR visualization and operations performed on the ribbons. Partitioning P into ribbons relies upon connectivity obtained from local parameterization (Alg.1 line 2-3, §3.1). P is locally parameterized by performing WPCA centred on each point based on its neighbourhood. This provides an estimate of the local parameterization as discussed in §2.1. The connectivity of the point p is computed from the vectors α_{p1} and α_{p2} (Alg. 1 line 5, §3.1.2, fig.1b).

With the connectivity at each point defined, the point cloud is partitioned into ribbons by traversing the local connectivity (Alg. 1 line 6, §3.2). By following the principal directions of variation at each point *U-ribbons* (following the principal direction of variation) or *V-ribbons* (following the secondary principal direction of variation) can be constructed (fig.1c).

With the exception of the neighbourhood computation for each point all operations are linear, and may be implemented efficiently using simple data structures such as queues and linked lists.

```
1  For all p ∈ P:
2    Find N_p of p
3    Compute α_p0 and α_p1
4  End For
5  Construct local connectivity of P
6  Compute ribbons from local connectivity
```

Algorithm 1: High level pseudocode of algorithm for generating ribbons

3.1 Parameterizing the Point Cloud

3.1.1 Computing the local parameterization

The neighbourhood of $p \in P$ is denoted N_p and is used to estimate the tangent plane at p along with the local

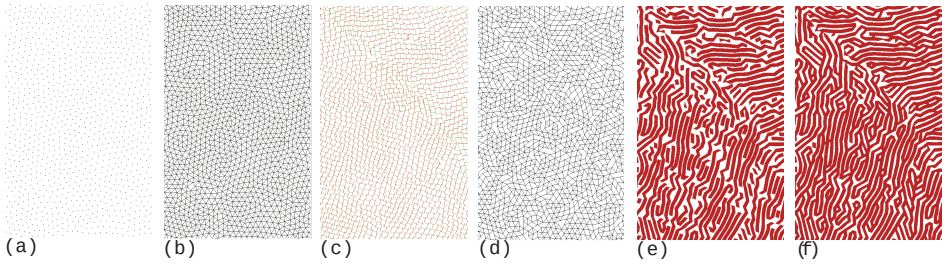

Fig. 4 a.)Initial points b.)k-NN of point set c.)Local coordinate frames (α_{p1} in orange, α_{p2} in green d.) local connectivity e.) Ribbons generated on point set f.) Result of rendering ribbons as described in text

density and connectivity of p. To this end the k-nearest neighbours (k-NN) of p are used[14, 15] (fig.4b).

N_p is used to establish a local density estimate for p. Global density estimates have also been used [12], but local estimates have been found to produce better results [14]. Let H_p be the local density estimate at p given in §2.1 This density estimate is similar to that used by Pauly et al [14], but takes into account all k-neighbours. It is possible that the euclidean distance between p and one of its k-neighbours is small even though the distance along the surface is quite large and as such should not be included in N_p. To help protect against this possibility all points in N_p further than H_p from p are removed from N_p. As H_p is a local estimate it changes adaptively across the surface.

N_p is used to perform WPCA centred on p to obtain α_{p1},α_{p2}, and α_{p3} the three eigenvectors of the weighted covariance matrix C. Eigenvalues and vectors can be computed inexpensively in constant time as C is a 3 by 3 symmetric positive definite matrix. As the formulation of WPCA uses H_p the locality of the tangent plane estimate varies adaptively across the point cloud.

Vectors α_{p1} and α_{p2} provide an estimate of the tangent plane at p. From this two perpendicular lines l_{p1} and l_{p2} are obtained, which when given in their parametric form are

$$l_{p1}(t) = p + t\alpha_{p1}$$

$$l_{p2}(t) = p + t\alpha_{p1}$$

The line l_{p1} then follows the principal direction of variation and l_{p2} the second direction of variation. Linear regressions derived from PCA usually pass through the mean of the sample, but as we are concerned with the tangent plane at p, not the mean of N_p, the point p is used in its place.

3.1.2 Orienting tangent planes consistently, and constructing the local connectivity

The coordinate frames computed for each point should be closely aligned with neighbouring points, but may differ slightly or by a sign. The problem of consistently orienting tangent planes for a point cloud is difficult in general (see [9] and [13]) but is easily addressed when ribbons are used due to their linear connectivity.

The tangent planes at each point in P are oriented by picking unprocessed points randomly from the point cloud (initially all points are unprocessed). When a point is chosen it is marked as processed and placed in a queue (Alg.2 lines 1-4). Provided the queue contains at least one point, the head of the queue will be processed. Let h be the point at the head of the queue. Using the vectors α_{h1} and α_{h2} the local connectivity of h is computed (Alg.2 lines 7-10). This process is described in detail in the following section. The connectivity of h is given by the points $Left(h)$, $Right(h)$, $Up(h)$ and $Down(h)$.

Once h's connectivity has been calculated each point connected to h is processed in order to orient their coordinate frames similarly to h. This helps to maximize the length of each ribbon.

If k is connected to h but has not yet been processed then it must be processed. To do this it is necessary to first orient α_{k1} and α_{k2} as close to α_{h1} and α_{h2} as possible while keeping both vectors in the tangent plane at k. This is done by projecting α_{h1} and α_{h2} onto the tangent plane at k to produce α_{new1} and α_{new2} respectively. Gram-Schmidt is performed on α_{new1} and α_{new2} to guarantee orthogonality. The vectors new are then used in place of α_{k1} and α_{k2} respectively. The coordinate frames are visualized for an example point cloud in fig.4c.

As k is now oriented consistently with the neighbouring point h its connectivity can be computed, so it is added to the queue. Once all the points connected to h have been processed the process is repeated for the next point in the queue. The preceding paragraph describes lines 11-17 of Alg.2. If any point connected to h has not yet been processed it is added to the queue.

Once all the points connected to h have been processed the process is repeated for the next point in the queue. If all the points in the point cloud are not processed in the first pass then a second point is chosen randomly from the point cloud and the process is repeated.

There is no guarantee that the local connectivity of points will connect all the points in the point cloud. This

```
1  Unprocessed := P
2  While p ∈ Unprocessed do:
3    Q.push(p)
4    Unprocessed := Unprocessed − p
5    While Q contains at least one point do:
6      h := Q.pop()
7      Order N_p along α_h1
8      Calculate Left(h) and Right(h)
9      Order N_p along α_h2
10     Calculate Up(h) and Down(h)
11     S := {Up(h), Down(h), Left(h), Right(h)}
12     For all k ∈ S do:
13       if k ∈ Unprocessed then:
14         Unprocessed := Unprocessed − k
15         Reorient α_k1 and α_k2
16         Q.push(k)
17       End if
18     End For
19   End While
20 End While
```

Algorithm 2: Psuedocode for the procedure creating local connectivity

```
1  Unprocessed = P
2  While p ∈ Unprocessed do :
3    Create ribbon C_p starting at p
4    Link C_p to existing ribbons
5    Unprocessed = Unprocessed − C_p
6  End While
```

Algorithm 3: Pseudocode for generating a single ribbon

allows point clouds representing multiple objects to be processed without any special considerations.

3.1.3 Creating local connectivity

The local connectivity at a point p is determined by finding points preceding and following p along the lines l_{p1}, l_{p2} which follow the principal directions of variation at p (illustraited in fig.4d).

Let $N_{l_{p1}}$ be the set of points in N_p which are closer to l_{p1} than l_{p2} or equidistant from both lines (fig.5(b)). Once N_p has been reduced to $N_{l_{p1}}$ it becomes possible to determine $Left(p)$ and $Right(p)$ by ordering the points along l_{p1} and selecting the points preceding and following p respectively (fig.5 c-d). If no such point exists then p is selected instead. Determining $Up(p)$ and $Down(p)$ is similar but uses l_{p2} in place of l_{p1} (fig.5e).

To traverse the local ordering, two functions are needed. The first function $next_{LR}$ takes two points in a progression and produces the next point in the same direction using the local orderings of the points along the principal direction of variation. The second function $next_{UD}$ performs the same function in the secondary direction of variation. These functions are essential to the chaining process as they provide the methods necessary to stitch local orderings together.

3.2 Partitioning the Point Cloud into Ribbons

Using the local connectivity generated in §3.1.2 it is possible to partition the point cloud into ribbons. U-ribbons are produced by following the connectivity provided by $Left(p)$ and $Right(p)$ using $Next_{LR}$, and V-ribbons by using the connectivity provided by $Up(p)$ and $Down(p)$ and $Next_{UD}$.

The process of creating a set of ribbons is summarized in Alg.3. It is important to notice that the algorithm will not terminate until all points have been placed in a ribbon. When creating ribbons the local ordering at each point is exploited to maximize the length, whilst trying to respect the original topology of the surface. The linking process then serves to connect existing ribbons in order to create longer ribbons, when possible. The result of the ribbon generation process can be seen in fig.4e and fig.1c.

3.2.1 Ribbon Preliminaries

Each ribbon is defined by an ordered set of points and their respective coordinate frames. The ribbon containing p is labelled R_p; this notation is well-defined as each set of ribbons partitions the point cloud. Using this notation Alg.4 provides the details of the ribboning process, and corresponds with line 3 of Alg 3. Linking each ribbon, as it is produced, to existing ribbons serves to create longer ribbons (described in §3.2.3). The result is a set of ribbons

$$R = \{R_1, R_2, \cdots, R_j\}$$

U-ribbons and V-ribbons are denoted R_U and R_V respectively. To maximize ribbon length the starting point of each successive ribbon should be as far from points contained by pre-existing ribbons as possible. Unfortunately, this is an expensive computation; thus the p used to create R_p is chosen randomly from the points still not contained in a ribbon, which decreases the likelihood that points near pre-existing ribbons will be selected.

3.2.2 Creating a ribbon

Each ribbon is generated by staring at p and traversing the local ordering of points. The generation of U-ribbons is described, but modifying this process to create V-ribbons is straightforward and is discussed at the end of this section.

R_p is initialized to contain only a single unprocessed point p chosen randomly from P. Each point is marked during this process to indicate whether it: resides at the end or beginning of a ribbon, resides in a ribbon but not at the end or beginning, is not yet part of a ribbon. This allows for the loop conditions in line 3 and 8 of Alg. 4

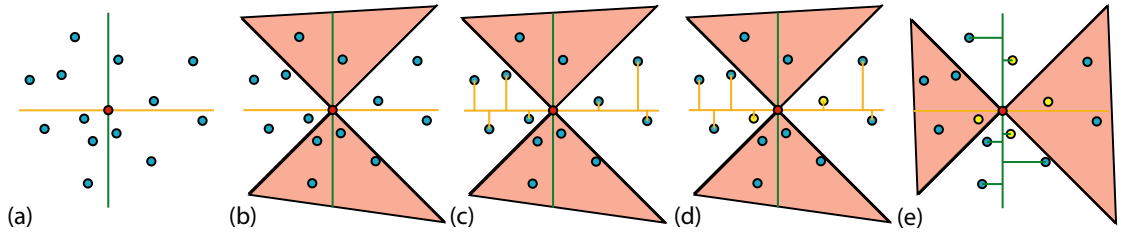

Fig. 5 a.)the point p is shown in red, its k-NN in blue and the lines l_{p1} and l_{p2} as orange and green respectively. b.) The points closer to l_{p2} then l_{p1} fall in the shaded region and are excluded from consideration c.) The remaining points are ordered along l_{p1} d.) $Left(p)$ and $Right(p)$ are determined e.) The process is then repeated for l_{p2} to establish $Up(p)$ and $Down(p)$

to be verified quickly, and makes the process of linking ribbons more efficient.

The process proceeds by adding $Left(p)$ to the front of the ribbon, $next_{LR}$ is then used to move along the local orderings and stops when $next_{LR}$ returns a point that has already been processed. The process is then repeated in lines 7-10 starting with the point $Right(p)$ with points added to the end of the ribbon, in order to respect the ribbons ordering(fig.6). When Alg. 4 terminates the ribbon R_p is ready to be linked to existing ribbons.

1	Add p to R_p
2	p := $Left(p)$
3	While p is not part of a ribbon :
4	Place p at the front of R_p
5	$p := Next_{LR}(R_{p,1}, R_{p,2})$
6	End While
7	$p := Right(R_{p,end})$
8	While p is not part of a ribbon:
9	Place p at the end of R_p
10	$p := Next_{LR}(R_{p,end}, R_{p,end-1})$
11	End While

Algorithm 4:

In order to generate V-curves it suffices to repeat the same procedure, but replace $Left(p)$, $Right(p)$, and $Next_{LR}$ with $Up(p)$, $Down(p)$, and $Next_{UD}$ respectively.

3.2.3 Linking ribbon

Once R_p is generated by Alg. 4 it is linked to the existing ribbons. This requires handling three cases: (i) R_p neither starts nor ends at the end of another ribbon, (ii) R_p starts or ends at the end of another ribbon (but not both), (iii) R_p starts and ends at the end of another ribbon. If the first case occurs nothing is done. When the second case occurs R_p is linked to the adjoining ribbon. The final case requires linking R_p to the ribbon at the start of R_p, and then linking the resulting ribbon to the ribbon at the end of R_p.

All three cases can be identified by examining $Left(R_{p,0})$ and $Right(R_{p,end})$ for U-ribbons and $Up(R_{p,0})$ and $Down(R_{p,end})$ for V-ribbons. Linking of ribbons is a local process, thus if multiple non-intersecting point clouds are present no ribbon should link the two objects.

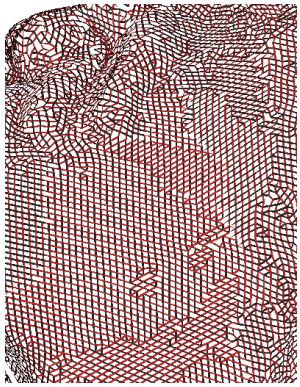

Fig. 7 U ribbons (black) and V ribbons (red) rendered with uniform width

It is interesting to note that when two sets of ribbons are produced following orthogonal vector fields their combined connectivity yields mostly quads (fig.7). As the principal directions of variation meet this criteria the unions of U and V ribbons yields mostly quads.

3.3 Rendering Ribbons

Although a set of ribbons does not reconstruct the surface they can be rendered in a manner that approximates visual continuity, this is similar to what is proposed by Boubekeur et al[4]. Each ribbon is rendered independently as three quad strips using OpenGL, the necessity of three quad strips is discussed below.

WPCA provides an estimate of the surface normal, but the normal may not be oriented correctly. Computing the correct orientation is a relatively complex and time consuming operation[9,13], to avoid this two shaded quads are rendered between consecutive points in each ribbon. The first quad is translated positively along the normal, and the second negatively along the normal. The quads are given width by moving the corners orthogonally with respect to the vector connecting the two points and the normal at each end of the quad, as is illustrated

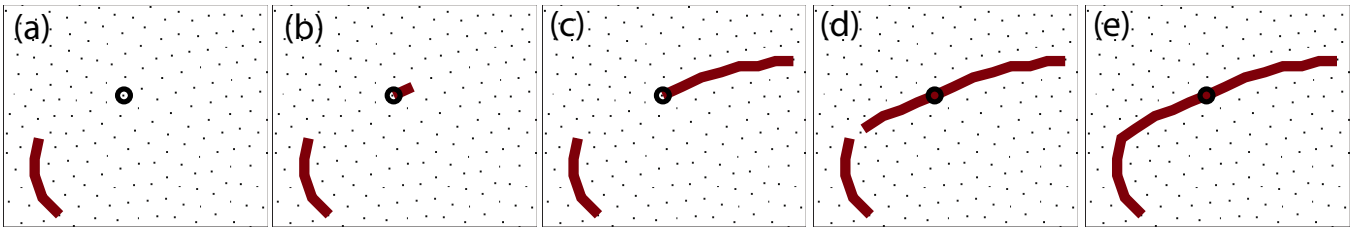

Fig. 6 a.) Point cloud with one ribbon already present, and p highlighted b.) $Right(p)$ is calculated and added to ribbon c.) process continues until $Right(p) = p$ d.) Process is repeated starting with $Leftp$ e.) The new ribbon ends at the end of an existing ribbon, the two ribbons are joined

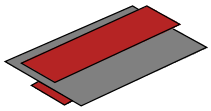

Fig. 8 The three quads connecting two points: the top and bottom quads are shaded, whereas the middle quad is not shaded and coloured the same as the background

in fig.8. A third quad is rendered for the purpose of hidden surface removal. This quad is not displaced along the normal and is wider then other quads. By colouring the quad the same colour as the background, and turning off lighting, it serves to occlude quads behind the current quad. The quads width can be set to a constant value, or vary between quads as described in §4.2 . Rendering each ribbon as three quad strips allows for interactive rates to be achieved, and the user can adjust the maximum quad width W_{max} in order to visualize the point cloud. Although ribbons do not branch or form loops the visual quality of results is increased when they are allowed to do so. This is achieved by extending the beginning and end of each ribbon along their local connectivity when rendering and is shown in fig.4f.

It should be noted that one quad strip may be omitted when oriented normals are provided, and the middle quad when hidden line removal is not needed.

4 Applications

4.1 Reverse Subdivision and Subdivision

To perform reverse subdivision on the point cloud it suffices to perform reverse subdivision along the set of ribbons R_U or R_V. This is roughly equivalent to performing reverse subdivision along one direction of a tensor surface.

Any set of ribbons can be handled similarly, so let us consider an arbitrary set of ribbons R. To this end let R_{k+1} be the $k+1^{st}$ set of coarse ribbons then

$$R_{k+1} = \{R_{k+1,p_1}, R_{k+1,p_2}, \cdots, R_{k+1,p_r}\}$$

Now let A, B, Q, P be the usual reverse subdivision filters derived from the sub-division filter P. Then

$$R_k = \{R_{k,p_1}, R_{k,p_2}, \cdots, R_{k,p_r}\}$$

D_k, the k^{th} set of details can be obtained similarly by applying B to each ribbon in R_{k+1}. Thus multiple applications of reverse subdivision can be reconstructed, provided that the set of ribbons is not recomputed.

From the k^{th} set of coarse points we can return to R_{k+1} by applying P and Q to the ribbons of R_k and D_k respectively as follows:

$$R_{k+1} = \{PR_{k,p_1} + QD_{k,p_1}, \cdots, PR_{k,p_r} + QD_{k,p_r}\}$$

Thus if we consider only one set of ribbons it is possible to obtain a multiresolution system. Additionally, by utilizing the connectivity of the coarse ribbons it is possible to avoid recomputing the k-NN making this a very efficient operation. Performing reverse subdivision without re-parameterizing is visualized in fig.14 using one of the NPR techniques presented in the following section. Unfortunately, repeated applications of reverse subdivision may not produce a nice approximation of the original point cloud as only one direction of curvature is taken into account.

It is not possible, presently, to perform reverse subdivision along the U-Ribbons followed by the V-ribbons as it is not clear how the V-ribbons must be modified to respect the reverse subdivision along the U-ribbons in a way that allows for subdivision connectivity to be maintained.

The process of parameterizing, and then reverse subdividing the point cloud can then be repeated to decrease the resolution further. Re-parameterizing the point cloud after reverse subdivision involves discarding the details needed to reconstruct the original point cloud, as such this does not provide a multiresolution system for point clouds.

The performance of reverse sub-division on ribbons was tested using Chaikin subdivision for open curves and

the corresponding length 4 reverse sub-division filter derived in [3].

The initial parameterization is fairly regular and consists mostly of quads. After reverse subdivision is applied the parameterization appears much less regular, but the points it produces still provide a reasonable approximation to the initial point cloud.

4.2 Non-Photorealistic Rendering

Ribbons are constructed to maximize length and follow the principal variation of the surface, properties which are desirable when strokes are used to visualize a surface as a line drawing. To demonstrate the effectiveness of ribbons for this purpose three NPR styles have been considered: silhouette rendering, hatching, and cross-hatching.

All three techniques are implemented by varying the width of each ribbon segment based on the view angle. Given a point p with normal n on the surface and the eye position e the view angle is

$$cos(\theta) = \frac{p-e}{|p-e|} \cdot n$$

The width of each shaded quad starting or ending at p is calculated as

$$W = W_{max}(1 - cos(\theta) - \tau)$$

where τ is a user defined threshold. When $W \leq 0$ all quads containing p are ignored. This formulation is similar to that employed for stable silhouettes by Brosz et al[5] for use in polygon meshes and offers the same advantages in this setting. Continuous changes in the view angles result in continuous variation in width, as such stable and continuous strokes are produced.

Varying width based on view angle, and rendering ribbons as described in §4.2 suffices to render the point cloud in a hatched NPR style. The unshaded quad serves to occlude strokes that should not be displayed.

Cross-hatching can be generated similarly by rendering two sets of chains following orthogonal vector fields in a hatched NPR style. As orthogonal vector fields will intersect at right angles at each point displaying the U-ribbons and V-ribbons simultaneously allows for such an effect. Examples of hatching and cross hatching using these techniques are given in fig.9-12.

Silhouettes are produced by varying the width as described above, but with each quad rotated to coincide with the normal instead of the tangent plane at each point. This increases the visibility of quads on the silhouette, which otherwise would be almost perpendicular to the view plane. Silhouettes require higher values of τ for proper visualization.

All the NPR techniques discussed operate directly on the ribbon represented without relying on an underlying mesh like representation. Furthermore, the point cloud can be visualized at interactive rates, allowing for easy user specification of τ.

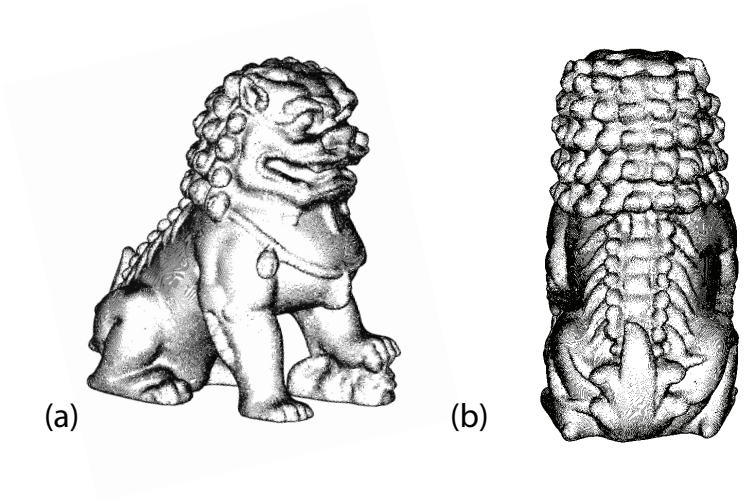

Fig. 9 Point cloud consisting of around 153k points, a.) V ribbons rendered as hatching b.) U ribbons rendered as hatching, model is provided courtesy of INIRIA by the AIM@SHAPE Shape Repository

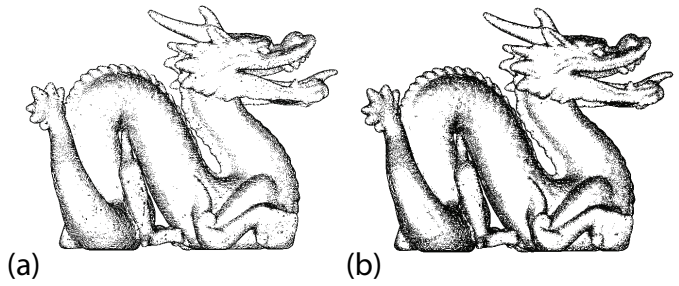

Fig. 10 Point cloud consisting of around 100k points, a.) V ribbons rendered as hatching b.) Rendered with cross hatching model is provided courtesy of Stanford Computer Graphics Laboratory

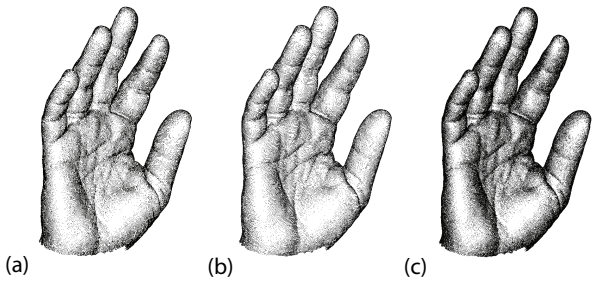

Fig. 11 Point cloud consisting of around 53k points, a.) U ribbons rendered as hatching b.) V ribbons rendered as hatching c.) Rendered with cross hatching model is provided courtesy of INIRIA by the AIM@SHAPE Shape Repository

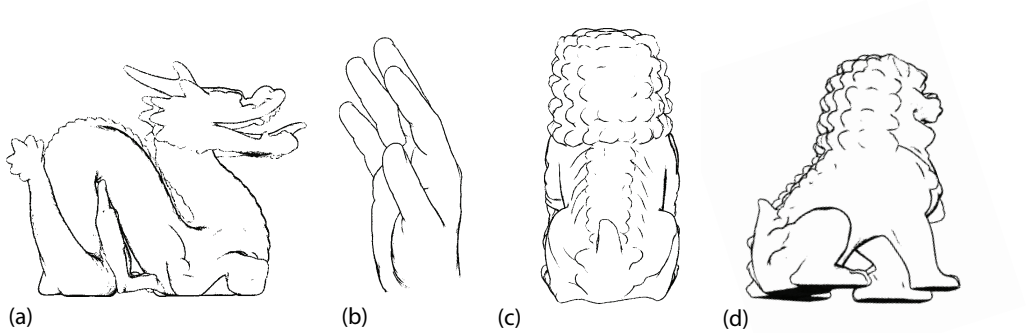

Fig. 13 Silhouette technique applied to a.)dragon b.)Olivier hand c-d) Chinese dragon

Fig. 12 Point cloud consisting of 180k points a.)U ribbons rendered as hatching (b-c) Rendered with cross hatching and silhouette, model is provided courtesy of INIRIA by the AIM@SHAPE Shape Repository

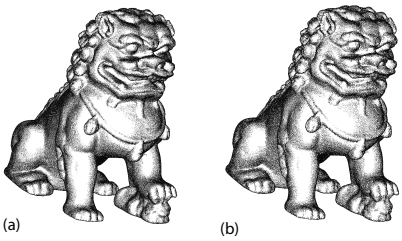

Fig. 14 (a) Initial point cloud consisting of approx. 180k points (b) After one level of subdivision, approx. 96k points

5 Conclusion

Ribbons provide an intermediate representation for point clouds, more local than surface reconstruction and less local than point based splatting. The ribbon representation of a point cloud can be computed efficiently using only simple data structures, and can be rendered directly as quad strips without additional processing.

The technique presented allows for a number of NPR visualizations, and the application of one dimensional operations to the point cloud. The ribbons linear structure is well suited to stroke placement, and based on this initial work it appears that further exploring the application of this representation to NPR provides an interesting direction for further research.

At present the visual continuity obtained using ribbons is insufficient for purposes beyond visualization. Improving the visual continuity of ribbons is thus very desirable and is one of the aims of future research.

Although ribbons are created along the principal directions of variation it is possible to generate ribbons along any vector field defined on the point cloud. This should allow for the visualization of vector fields on the surface of the point cloud.

And finally, by exploiting both the U and V ribbons together it should be possible to allow for additional operations taking into account both directions of curvature. This would allow for a wider range of operations to be performed

References

1. Adamson, A., Alexa, M.: Approximating and intersecting surfaces from points. In: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 230–239. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
2. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications, pp. 249–266. ACM Press, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/376957.376986
3. Bartelss, R., Samavati, F.: Reversing subdivision rules: Local linear conditions and observations on inner products. Journal of Computational and Applied Mathematics **119**(1-2), 29–67 (2000)
4. Boubekeur, T., Reuter, P., Schlick, C.: Visualization of point-based surfaces with locally reconstructed subdivision surfaces. In: Shape Modeling International. IEEE Computer Society Press (2005). URL http://iparla.labri.fr/publications/2005/BRS05
5. Brosz, J., Samavati, F., Sousa, M.C.: Silhouette rendering based on stability measurement. In: SCCG '04: Proceedings of the 20th spring conference on Computer graphics, pp. 157–167. ACM Press, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/1037210.1037235
6. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. ACM Trans. Graph. **22**(3), 848–855 (2003). DOI http://doi.acm.org/10.1145/882262.882354
7. Dey, T.K., Giesen, J., Hudson, J.: Delaunay based shape reconstruction from large data. In: PVG '01: Proceedings of the IEEE 2001 symposium on parallel and large-data

visualization and graphics, pp. 19–27. IEEE Press, Piscataway, NJ, USA (2001)
8. Hertzmann, A., Zorin, D.: Illustrating smooth surfaces. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 517–526. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000). DOI http://doi.acm.org/10.1145/344779.345074
9. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 71–78. ACM Press, New York, NY, USA (1992). DOI http://doi.acm.org/10.1145/133994.134011
10. Joliffe, I.: Principal Component Analysis. Springer-Verlag (2002)
11. Kawata, H., Gouaillard, A., Kanai, T.: Interactive point-based painterly rendering. In: CW '04: Proceedings of the 2004 International Conference on Cyberworlds (CW'04), pp. 293–299. IEEE Computer Society, Washington, DC, USA (2004). DOI http://dx.doi.org/10.1109/CW.2004.42
12. Levin, D.: Mesh-independent surface interpolation. In: Geometric Modeling for Scientific Visualization, pp. 37–49 (2003)
13. cius Mello, V., Velho, L., Taubin, G.: Estimating the in/out function of a surface represented by points. In: SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications, pp. 108–114. ACM Press, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/781606.781625
14. Pauly, M., Gross, M., Kobbelt, L.P.: Efficient simplification of point-sampled surfaces. In: VIS '02: Proceedings of the conference on Visualization '02, pp. 163–170. IEEE Computer Society, Washington, DC, USA (2002)
15. Pauly, M., Keiser, R., Kobbelt, L.P., Gross, M.: Shape modeling with point-sampled geometry. ACM Trans. Graph. **22**(3), 641–650 (2003). DOI http://doi.acm.org/10.1145/882262.882319
16. Pauly, M., Kobbelt, L.P., Gross, M.: Point-based multiscale surface representation. ACM Trans. Graph. **25**(2), 177–193 (2006). DOI http://doi.acm.org/10.1145/1138450.1138451
17. Praun, E., Hoppe, H., Webb, M., Finkelstein, A.: Real-time hatching. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, p. 581. ACM Press, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/383259.383328
18. Samavati, F., Bartels, R.: Multiresolution curve and surface representation by reversing subdivision rules. Computer Graphics Forum **18**(2), 97–120 (1999)
19. Sousa, M.C., Prusinkiewicz, P.: A Few Good Lines: Suggestive Drawing of 3D Models **22**(3), 381–390 (2003). DOI 10.1111/1467-8659.00685
20. Xu, H., Chen, B.: Stylized rendering of 3d scanned real world environments. In: NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering, pp. 25–34. ACM Press, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/987657.987662
21. Zakaria, N., Seidel, H.P.: Interactive stylized silhouette for point-sampled geometry. In: GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, pp. 242–249. ACM Press, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/988834.988876
22. Zwicker, M., Pfister, H., van Baar, J., Gross, M.: Surface splatting. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 371–378. ACM Press, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/383259.383300