Graph-Based Point Relaxation for 3D Stippling

Oscar Meruvia Pastor
Department of Computing Science
University College of the Cariboo
Box 3010, V2C 5N3 Kamloops, BC, Canada
meruvia@cariboo.bc.ca

Thomas Strotthote
Department of Simulation and Graphics
Otto-von-Guericke University of Magdeburg
P.O. Box 3120 39108 Magdeburg, Germany
tstr@isg.cs.uni-magdeburg.de

Abstract

Point hierarchies are suitable for creating frame-coherent animations of 3D models in non-photorealistic styles such as stippling, painterly and other artistic rendering. In this paper, we present a new approach to produce regular point distributions on the surface of a polygonal mesh. We propose a graph-based token distribution approach, where a graph that extends over the surface of the polygonal model is used as the playing field for the distribution of points. This graph-based approach eliminates the need of using geometrical operations to distribute points over the surface of a polygonal mesh. The graph exists at several levels of detail that are easily created using iterative patch fusion, and which are used to create a point hierarchy. In addition, we show how 3D stippling is used to render transparent surfaces and illustrate complex animations.

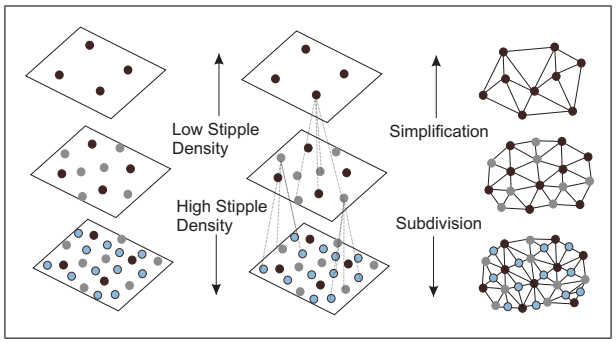

Figure 1: **Point hierarchies determine both the position and the order of appearance and removal of stipples during rendering. In addition, they are used to control the stipple density according to the rendering parameters.**

1. Introduction

Real-time animated stippling (RTA stippling, or 3D stippling) is a rendering technique where a 3D model is rendered in the stippling style by using a point hierarchy to determine the size and position of the stipple particles which are distributed on the surface of a model [11]. The point hierarchy used in RTA stippling determines both the distribution of particles on the surface of a model at different levels of detail and the order of rendering of these particles, depending on conditions such as shading, grey tone, and viewing distance (see Figure 1). The rendering function used in RTA stippling emulates the artistic rendering style called stippling, where shading is given by the stipple density when rendering. Darker tones are obtained by having points closely spaced, and lighter tones are obtained by placing a few points while taking care that the points are appropriately distributed.

Hierarchical particle systems enable both frame-coherent and view-dependant rendering. Frame-coherence is achieved by setting particles at fixed locations on the surface of the model, while adaptive rendering is achieved by assigning a radius value to each particle, according to its position in the hierarchy. The rendering system uses this radius value to produce view-dependent animations with smooth insertion and removal of points at each frame.

So far, point hierarchies have been created by applying a series of randomization, subdivision and simplification transformations on the input polygonal mesh [11]. In this paper, we present a new method for creating a point hierarchy for real-time animated stippling and for distributing particles on the surface of a model using graph-based token distribution. Our primary goal is to improve the distribution of stipples on the surface of the model, but our results indicate that this technique reduces the time required to create the point hierarchy as well.

1.1. Outline of Contents

In Section 1.2 we describe our approach at a glance. We proceed to discuss related work on computer-generated stippling and point relaxation in Section 2. After that, we describe our approach in detail: Section 3 describes the process of creating the patch hierarchy and Section 4 describes point relaxation by token displacement. Section 5 shows our experimental results and Section 6 describes new applications for 3D stippling. Finally, in Section 7 we present conclusions and future work.

1.2. Graph-Based Point Relaxation

The first stage of our approach is to produce a patch hierarchy, which is done by iterative patch fusion (described in section 3). The patch hierarchy produces several levels of surface patch subdivisions of the input model. Each patch at a given level encloses a set of patches at a lower level of the patch hierarchy.

At each level of the patch hierarchy, a connectivity graph can be derived from the set of surface patches which make up the entire model. To create the patch graph, each patch is associated with a point randomly selected from its surface, which we call the patch 'center'. The edges of the graph are formed by connecting each pair of patch centers that belong to neighboring patches (see Figure 2). This connectivity graph is used as the 'playing field' for token distribution later on.

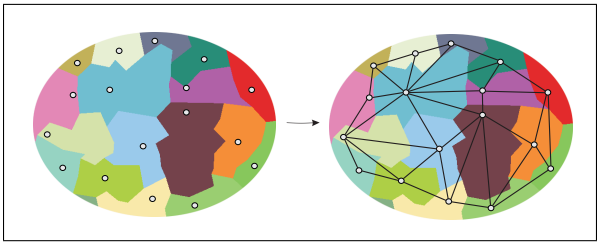

Figure 2: **On the left, we illustrate a sample patch arrangement and the points associated to the patches. On the right, we show the graph obtained from the patch arrangement based on the connectivity information contained in the patches.**

The second stage of our approach is the iterative point relaxation, where we traverse each level of the patch hierarchy, and perform graph-based token distribution. To create each level of the point hierarchy, we proceed to distribute a number of tokens among the nodes of the graph. The tokens represent point particles which are fixed to the surface of the input model once relaxation is finished. This relaxation process is explained in section 4.

The relaxation process is done first on the highest level of the patch hierarchy and proceeds in a top-down fashion. Once the relaxation of the tokens has been finished at a certain patch level, we fix the tokens on the surface of the model, go down one level in the patch hierarchy, generate additional tokens and perform relaxation on this new set of tokens. This is done iteratively until all levels of the patch hierarchy have been traversed.

2. Related Work

In the area of computer-generated stippling, both Deussen et al. [1] and Secord [13] perform Voronoi relaxation of stipples. An advantage of these techniques is that they can take as input any grey scale image to produce a stippled rendition. However, a frame-coherent animation sequence cannot be constructed using these approaches, because the point relaxation is unique to each rendered image. To address this problem, a particle rendering system for image-based frame-coherent rendering of stipples and strokes was proposed by Secord [14]. The resulting animations are coherent across frames of the animation. However, because the particles are not fixed to the model surface, the connection between the particles and the model itself is lost.

Frame-coherent animations at the particle level were presented for Real-time animated stippling [11], where a hierarchy of points fixed on the surface of a polygonal model is created by applying a series of randomization, subdivision and simplification transformations on the input mesh (see Figure 1, right). In our approach, we also create a point hierarchy fixed to the surface of the input model. However, instead of doing mesh simplification and randomization, we perform patch-based hierarchy creation, and use a graph-based relaxation approach to ensure that point distribution is regular at all levels of the point hierarchy. In our case, we need to perform mesh subdivision if the input model is very coarse or if it is not regularly tessellated. Once the model is subdivided, patches at the lowest level of the hierarchy are roughly at the same scale, and we proceed with graph-based relaxation as described before. In [12] a method is presented for halftoning on hexagonal grids which could be applied to distribute stipples on dense, regularly tessellated polygonal meshes, but noise might appear due to a lack of frame-coherence when applying the technique during animations.

frame-coherence and animation is not discussed, and dithering techniques can suffer from the screen-door effect.

A point-based rendering system to illustrate volumetric data in the stippling style has been presented by Lu et al. [6]. In their approach, a point hierarchy is produced using a spatial partition scheme (an octree). Their approach consists in controlling the size of the stipples according to their level at the octree and information computed from the volumetric data. This system has several advantages. First, their an-

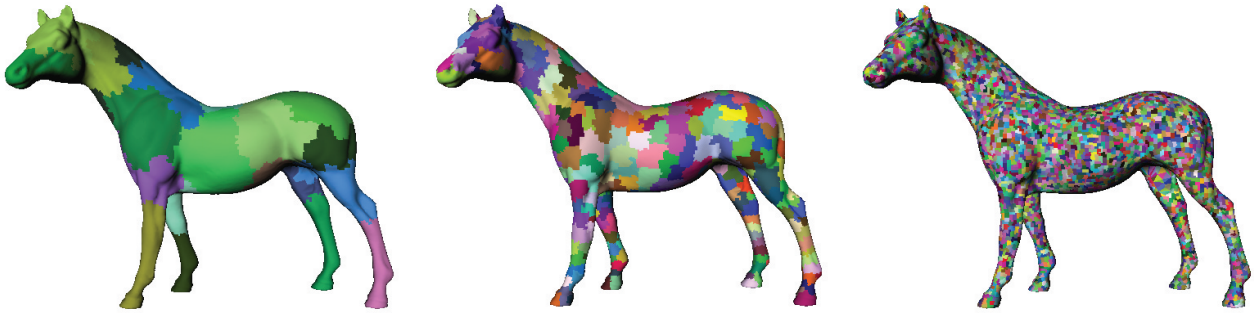

Figure 3: **Patch hierarchies on the horse model. The first image (from left to right) shows the model with 25 patches, the second and thirds models have their surface subdivided in 350 and 6144 patches, respectively.**

imations are view-dependant and frame-coherent. Second, the system takes advantage of hardware acceleration to render models at interactive rates. A drawback of their approach is that it is difficult to distinguish among different layers of the input model, because no mechanism is provided to perform occlusion culling. A solution to this problem is given by Lum and Ma [7], who mix photorealistic and non-photorealistic styles to illustrate different elements of the volume data. In addition, Dong et al. use stroke-based rendering [2] to illustrate medical volume data while providing occlusion culling. Our approach mixes polygons and stipples and uses the Z-buffer to determine occluded areas, so we can use stipples to illustrate transparent surfaces on top of opaque polygonal surfaces, while in addition, we produce frame-coherent animations of static and animated models (as shown in Section 6).

Graph-based token relaxation is based on the work on point relaxation for polygonal re-tiling [17] and for texture synthesis using reaction-diffusion [16]. An important difference between our approach and Turk's method for relaxation is that while in [16] points are displaced from polygon to polygon using geometric operators that "unfold" the surface as the point is being displaced by the repulsion forces of their neighbors, our approach eliminates the task of displacing points over the polygonal surface and thus, the need of unfolding the surface along the displacement paths. In addition, the graph where the points are displaced is a simplified graph with respect to the graph that corresponds to the input mesh. This results in a fast relaxation process which is linear to the number of points distributed at each level.

Witkin and Heckbert [18] also presented a particle relaxation method to provide appropriate sampling and manipulation of implicit surfaces. Their work describes the use of energy functions, particle subdivision and particle fusion for adaptive sampling, and includes the use of velocity functions modeled by differential equations to rapidly achieve regular point distributions under implicit surfaces deformation. Their approach to point relaxation is similar to ours, specially in the use of repelling forces among neighboring particles to do the relaxation. However, their approach was not designed to work on polygonal models and cannot be intuitively extended to adapt to the discontinuities present in polygonal surfaces. On the other hand, our approach introduces the concept of using a graph as the playing field where point relaxation takes place.

As shown by Garland et al. [3], polygonal patch hierarchies have a number of applications such as mesh simplification, multiresolution radiosity and collision detection. Several methods exist for creating polygonal patch hierarchies, for example, face cluster hierarchies by [3], Soler et al.'s method for hierarchical texture mapping [15] and Meruvia's approach for visibility preprocessing [10]. In principle, patch-graph based relaxation can be performed using patch hierarchies generated with any of the methods mentioned above.

3. Patch Hierarchy Creation

The patch creation approach we use was originally developed for visibility preprocessing [10]. This is a bottom-up approach: we start by constructing the hierarchy from the lowest patch level and then perform iterative fusion of the patches available at a given level. The initial patch list (the lowest level in the hierarchy) is constructed from the graph connectivity information of the input model: each polygon becomes a patch, and the polygons which share an edge with a given polygon are inserted in the list of neighboring patches of this polygon.

Initially, all the patches of a given level are stored in a binary search tree (BST) ordered by the amount of surface they cover. In each fusion step, the smallest patch found in the BST is combined with a number of neighbors (four in our case), creating a patch at the next level. To select which neighbors will be combined with the patch chosen, we choose the neighbor that minimizes the span

of the bounding box of the input patch, with the restriction that only patches at the same level can be fused together. The patches that were combined are removed from the BST and the next smallest patch available is used at the next fusion step until all patches at a given level have been removed from the BST. After this, we insert the newly created patches in the BST, and proceed with patch fusion at the next level.

The patch hierarchy creation process finishes once a certain number of target patches at the highest level are reached or after a certain number of levels have been produced. In our case we stop when we have less than 10 patches at a certain level. In Figure 3 we show a series of patch hierarchies for the horse model.

A patch hierarchy is initialized with patch 'center' values by randomly assigning one point to each of the patches at the lowest level (which are the base polygons of the model). Patches of the higher levels are assigned a point randomly selected from one of its descendants 'center' values.

4. Particle Relaxation by Token Displacement

Now we are ready to carry out a token relaxation process to distribute particles on the surface of the model. The goal of the relaxation process is that the distribution of the particles does not conform with any linear or regular pattern, and that the spacing among particles is regular at each level of the point hierarchy.

We perform the process of particle relaxation as a token distribution approach. A token represents the place where a rendering particle is located. We borrow the term token from the computer networks literature, where a token is passed along the nodes of a graph and does not belong to any specific node until relaxation finishes. The graph where the tokens are distributed is defined by a list of all the patches and their connectivity information at a given level: the nodes are the points in 3D associated with each patch and the edges are formed by the list of neighboring patches of each patch in the list (see Figure 2).

The relaxation process starts by pseudo randomly distributing tokens among the nodes of the graph. After that, tokens are brought apart from each other by iteratively applying a relaxation step where each token is pushed away by the tokens in its neighborhood. At each iteration, we count how many tokens were actually displaced (changed their position in the graph). Relaxation stops when the displacement count is zero or when we detect this count has stopped decreasing. In summary, our algorithm for the relaxation process is as follows:

distribute tokens on the graph
loop until the displacement count stops decreasing
  for each token T on the graph
    1. determine neighbor tokens of T
    2. compute the repulsive forces that the neighbor tokens exert on T
    3. determine whether the token needs to be displaced, and in this case, which of the neighbor nodes should receive the token based on the repulsive forces
  for each token T on the graph
    if the token needs to be displaced, displace the token T to the new host node, and increase the displacement count.

Now we proceed to describe in more detail each step of this algorithm.

4.1. Token Distribution

Initially, we randomly select, for each patch at a given level in the hierarchy, one descendant found *n*-levels below in the patch hierarchy, and assign this descendant a token. The goal of using patches at lower levels in the hierarchy is to relax the tokens in a graph with such a number of extra nodes, that there is enough room for the tokens to be freely displaced. In our case, we have chosen *n* to be 3, so that each selected descendant is displaced within a graph containing approximately 4^3 nodes without a token.

4.2. Determining the Set of Neighboring Tokens

The first step to perform relaxation on a given token is to determine which tokens are located in its neighborhood. First, we select for each token the neighboring patches (at the higher level) that hold a token. Each token explores its neighborhood by visiting all the descendants of its ascendants, up to two levels higher in the hierarchy, and its immediate neighbors. Each node in this set is queried as to whether the node is host of a token. All the nodes which are hosting a token are inserted in the list of neighbor tokens, which is used to compute the repulsive forces exerted on the original token. An alternative approach to find the list of neighbors is to visit all nodes which are within a certain distance from the node (this can be determined for example, by using Dijkstra's algorithm to find the shortest distances to the neighbors) by traversing the graph from the token outwards, but this is more time-consuming than our approach.

4.3. Computing the Repulsive Forces

To compute the repulsive forces, we use those tokens in the list of neighbor tokens which are within a region of influence determined by a *repulsion radius*. The repulsion radius ($repRadius$) is obtained as:

$$repRadius = 2\sqrt{(a/n)/\pi} = 1.128\sqrt{a/n}$$

where a is the area of the surface of the input model and n is the number of tokens being distributed.

We compute the repelling forces $ForceVec$ by computing the vectors NT_Vec which go from the position of the neighboring tokens N to the input token T, and then scaling each of these according to their distance, and in proportion of the repulsion radius, using the formula

$$ForceVec = (repRadius - |NT\_Vec|) * NT\_Vec / |NT\_Vec|$$

After that, we average all the neighboring forces and produce a vector in 3D which indicates the direction of the overall force that the neighbors within the repulsion radius exert on the input node T.

4.4. Displacement Selection

If the overall force is different from zero, we need to determine which of the empty neighbors around T is the best candidate to receive the token. This is done by computing the dot product between the computed force and the vectors leaving from the host node to its neighbors. The neighbor which maximizes this dot product is chosen and stored as the node that will hold the token. Once the displacement of each token is computed, the actual displacement of all tokens is done at once, hereby completing one relaxation iteration.

4.5. Termination Criteria

Since tokens can only occupy a limited set of positions within the surface of the model, a token can constantly jump back and forth between two nodes in search of the optimal position as several iterations take place. Since the algorithm does not necessarily converge to an stable token arrangement, we finish relaxation once the total energy (the sum of the force values of all the distributed tokens) stops decreasing.

Figure 4 illustrates the relative drop in energy as each relaxation iterations take place. The first data point shows the energy contained by the tokens measured before the first iteration took place (100%), and the following data points indicate the energy as a fraction of the initial energy value. To obtain these values, we averaged the total energy values obtained after each relaxation for a total of 7 models of different sizes (listed in Table 1). We observe that the decrease in energy reaches a plateau in less than 10 iterations, although a minimal decrease still takes place after 10 iterations. Based on this information, we stop iterating once 5 iterations have taken place without a significant decrease in the energy (where a significant decrease is any decrease larger or equal to 1% of the initial value) or after 15 iterations have taken place.

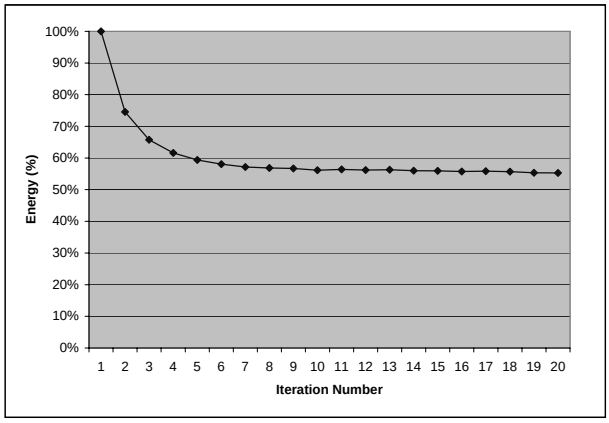

Figure 4: **Drop in total energy value as relaxation iterations take place. The data points represent the percentage of total energy remaining after each iteration, with respect to the initial energy value.**

4.6. Point Hierarchy Creation

We have so far described how we can distribute particles on the surface of a model for a specific patch level. To create a point hierarchy, we use a top-down approach as in [17]. We start the relaxation process with the patches at the highest hierarchy level, then we fix the results of the relaxation and proceed to distribute and relax tokens at the next level, until we have traversed the complete hierarchy. The algorithm of this process is the following:

Select the top level of the patch-graph hierarchy
while not all patch-graph hierarchy levels have been traversed:

1. For each patch at the current level, create a token and place it among the patch's descendants located 3 levels down the hierarchy.
2. Perform token relaxation using the graph where the tokens were placed.
3. Fix the tokens on the patch-graph hierarchy and on the model.
4. Assign the radius values for all nodes at the current patch-graph level.
5. Go down one level in the patch hierarchy.

The point hierarchy is generated as a dual of the patch hierarchy. Instead of storing the patch information and its connectivity at each level, the point hierarchy stores the point position and its radius value. For each patch, the point position is obtained from the position assumed by its token after relaxation. The point's radius value is obtained as the average of the distances to the point positions of the neighboring patches. Because the point density increases at each level down the hierarchy, the particles that are distributed at any given level of the patch hierarchy necessarily have larger radius values than the particles relaxed at lower lev-

els of the hierarchy. Finally, the information of the point hierarchy is saved in a file and used during rendering.

Figure 5 shows the distribution of points on the surface of the hand bones model at subsequent levels, once relaxation has taken place at each level.

4.7. View-dependent Rendering

Under stippling, points can only reach a maximum size. To keep shading while zooming into a model, more points are needed to maintain the tone and stippling style.

The rendering makes use of a function which is applied to each point tested for rendering. To reproduce a given shading tone, we compute the desired shading tone according to the viewing parameters, and compare it with the shading tone that would be produced if we decided to render the point. This shading tone is estimated dividing the maximum point size (defined by the user) over the area covered by the screen-space projection of the point as a function of its radius. If the desired shading tone is exceeded by placing the point on the image, the point is not rendered, otherwise, the point is rendered. This is the basic function, however, this is a binary test which would insert popping artifacts during interactive rendering. To avoid this, we have produced a function which interpolates the point size from zero to the maximum point size, depending on how much the estimated shading tone exceeds the desired shading tone. Using this function, new points appear between existing points, and fill-in smoothly as the animation occurs. This is the rendering function used in all stippled images shown in this paper (Figures 6, 7 and 8).

Since the point hierarchy stems from the patch hierarchy, it is possible to store and use the point hierarchy as a linear point data array ordered by radius. For rendering, we traverse the hierarchy in a top-down fashion and stop traversing once the area covered by a point is less than or equal to one pixel. Since all the descendants of a point have smaller radius than their ascendants, they occupy even a smaller section of the viewing area when projected and need not be considered for rendering. In addition, since each point belongs to a patch, and each patch is a member of a patch hierarchy, we can organize the points into groups of patches and assign a list of associated points to each patch. In this case, we use hardware acceleration (P-Buffers) to determine the set of visible patches, and then render only the points associated with the visible patches.

5. Experimental Results

Table 1 shows the completion times required for creating point hierarchies for several models using two techniques: mesh simplification and subdivision, and patch-based relaxation. The results were obtained using an SGI Octane w/1Gb Memory.

Table 1: **Statistics for the creation of point hierarchies.**

Mesh Simplification & Subdivision						
Model	**Polygons**	**Points**	**Setup Time**	**SubDiv Time**	**Simpl Time**	**Total Time**
Bunny	69,000	70,000	22”	26”	1’20”	1’59”
Horse	96,966	100,000	25”	34”	1’46”	2’11”
SavPot	172,078	86,041	38”	---	3’29”	4’7”
Brain	288,334	141,171	1’9”	---	5’41”	6’50”
Mosaic	400,000	200,002	1’30”	---	8’30”	10’

Graph-Based Point Relaxation					
Model	**Polygons/ Points**	**Setup Time**	**Patch-H Creation**	**Token Relax**	**Total Time**
Bunny	69,459	5”	14”	38”	57”
Horse	96,966	7”	19”	50”	1’15”
SavingsPot	172,078	8”	36”	1’36”	2’20”
Brain	288,334	17”	59”	2’37”	3’53”
Mosaic	400,000	19”	1’28”	3’57”	5’44”
HandBones	654,660	39”	2’31”	6’12”	9’22”
Dragon	870,877	1’15”	3’20”	8’51”	13’ 26”

The extra amount of time required for setting up mesh simplification and subdivision is due to the additional randomization stage where the vertices of the input mesh are jittered to avoid the presence of linear patterns in the renditions.

With respect to mesh subdivision, 2,400 to 3,500 polygons per second are generated depending on the size of the model. This includes the time required to update the connectivity graph which represents the polygonal mesh. An estimate of the total time required for performing graph-based relaxation in conjunction with mesh subdivision can be bone by adding mesh subdivision time (depending on the desired polygon count) to the time required for relaxation once subdivision has taken place.

Results in Table 1 show that the patch-graph based approach produces hierarchies that have a comparable amount of points within 45% to 60% of the time required by mesh simplification and subdivision. In addition, in the case of models that were processed doing only mesh-simplification, hierarchies contain in average two times more points when generated using the patch-graph relaxation approach. This is because the largest possible number of points that can be generated using mesh simplification depends on the number of vertices in the input model, while the maximum number of points that can be generated using the patch hierarchies depends on the number of polygons in the input model.

6. Applications

In this section we describe two novel applications for 3D stippling: illustration of animated transparent surfaces and virtual object reproduction for archaeology.

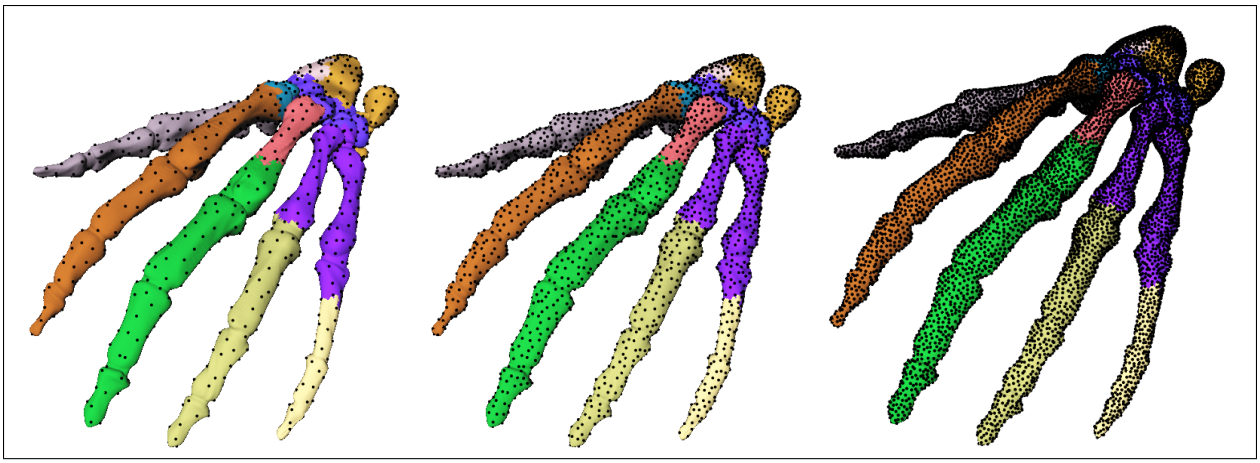

Figure 5: **Three levels of the point hierarchy after relaxation, the hand-bones model is shown with 1,192, 4,277 and 15,184 points respectively.**

6.1. Illustration of transparent surfaces

Animated stippling [11] works by mapping each stipple to a location within a face in an input model using barycentric coordinates. This approach makes it possible to produce frame-coherent animations in the stippling style.

A novel application of animated stippling is to use stipples to represent a transparent layer in a model. To achieve this effect, stipples are placed over the surface of the 3D model and are rendered in place of the surface itself.

Using stipples in this way has a double effect: on one hand, the stipples reveal the layer or surface where they are located; on the other hand, when stipples are sparsely distributed on the represented surface, it is possible to see the inner parts of the input model through the stipple cloud.

This transparency effect has the particular advantage, with respect to the standard transparency effect achieved through alpha blending, that when the model is animated, the stipples on the surface of the transparent layer behave like an elastic texture, showing how this surface changes during the animation. If the inner parts of the model are also animated, it is possible to view the animation of these parts as well. The images in Figure 7 and Video 1 illustrate this effect in the case of a heart beat. The heart shell has been replaced with a stippled layer and it is possible to see the animation of the inner part of the heart. Notice that the video was produced using standard 3D animation software and was produced only to illustrate this effect, it does not follow the actual physics of a beating heart. Several applications can take advantage of this way of presenting transparent surfaces, for example biomechanical illustrations showing the skin as an outer stippled layer and the muscles or tendons displacing underneath, or illustrations of automotive design and mechanics. This approach can also be used in volume rendering, where transparency plays and important role, as shown by the work of Lum and Ma [7], Lu et al. [6] and Interrante et al.[4].

6.2. Object Reconstruction for Archaeology

Stippling is a common technique for scientific illustration in archaeology to document and communicate findings made during excavations. We have worked in cooperation with the Institute of Archaeology of the State of -blind-review- to apply the 3D stippling technique on some objects they have excavated. In this project, we were given two objects, a savings pot and a colored mosaic, both dated back to the middle ages, as case studies to test the possibility of using our techniques for this particular subject.

The objects were 3D scanned, and we applied the point relaxation approach to produce the stippled drawings shown in Figure 8. We presented animations and interactive demonstrations to the team of archaeologists and received feedback from them. Their feedback can be summarized as follows: the technique effectively and convincingly shows the models in the stippling style in a frame-coherent way. However, the presented approach needs to deal with several issues before it can be accepted by the archaeologists. First, the amount of darkness in some areas, even though it reflects the lighting conditions, was too dark in some cases, failing to provide useful information in some regions of the object. Second, some details of the model are lost in the renditions, such as the reliefs in the hat and in the chest of the character illustrated in the bottom of Figure 8. Third, the technique to capture and display the models is not widely available to the majority of the community of archaeologists. This however, is a matter of accessibility to technology which falls outside of the scope of this work. On the other hand, it was suggested that the presented ap-

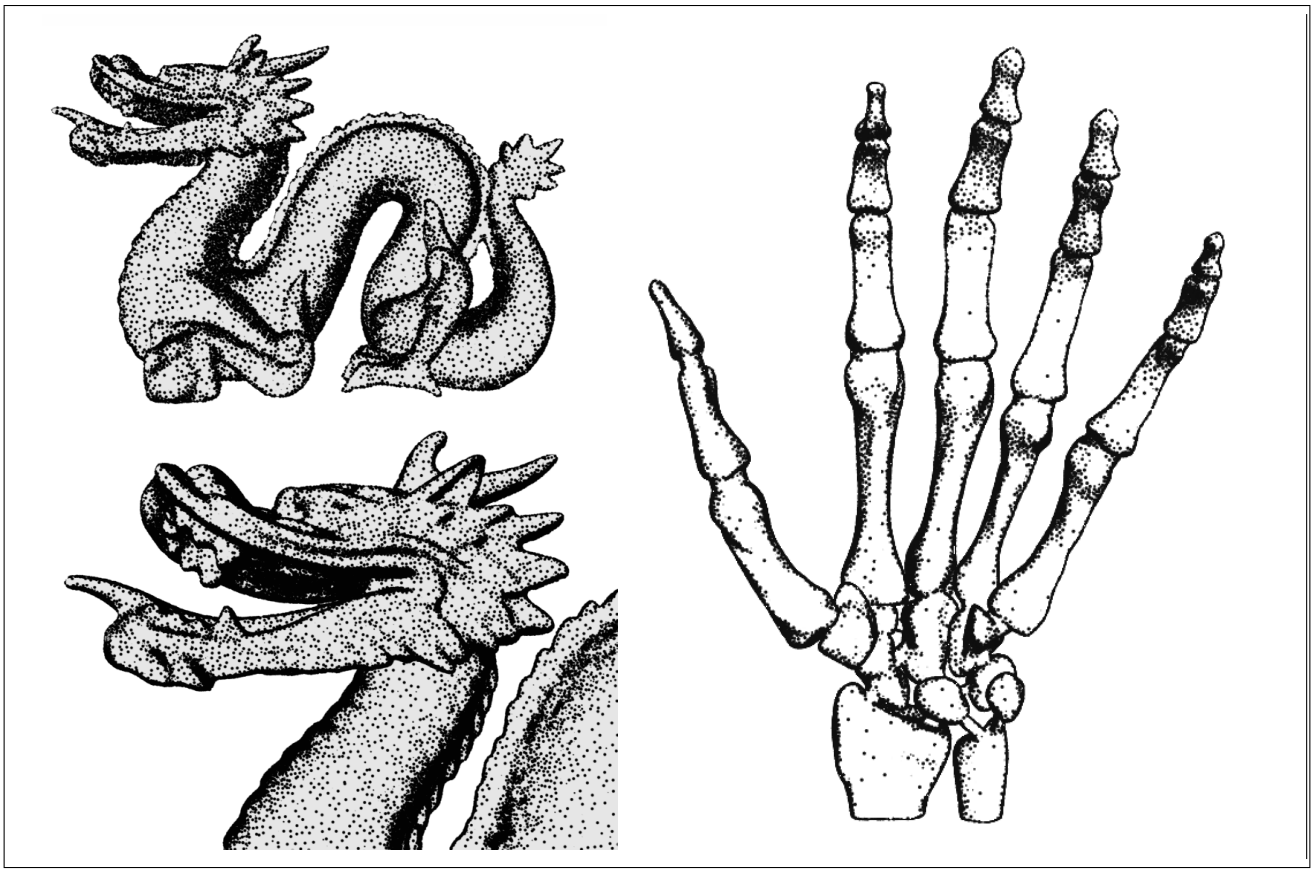

Figure 6: **Left: Stipple rendition of the dragon model. To keep shading while zooming from the view on the top left to the view on the bottom left, more points appear that maintain the tone and stippling style. Right: Illustration of the hand bones model. Both models were rendered using a point hierarchy generated by graph-based point relaxation.**

proach could be appropriate for making 3D presentations of the findings for the general public in museum exhibitions.

To address the first two (rather technical) issues we have implemented the following improvements in the rendering system: To deal with the issue of darkness, we control the maximum stipple saturation that can be shown on the renditions. This has the side effect that contrast is reduced, and calls for the adoption of multiple light sources to illuminate particularly dark regions. To deal with the loss of detail we have produced two solutions: the first one is the introduction of silhouette enhancement using surface normals, where points that belong to the silhouette are rendered at the maximum point size (Figure 8, top right). The second improvement is the rendition of sharp edges rendered on top of the polygonal model and the stipples. This feature is illustrated in the bottom middle of Figure 8, where sharp edges have been rendered using a small point size. The result is that more details of the model are presented in the renditions.

In addition, some details of the original models were lost due to the resolution at which we scanned the input models (400,000 polygons for the mosaic and ca. 172,000 polygons for the savings pot), such as the curls in the hair of the character of the mosaic and some foldings in the clothes around the neck and in the middle of the chest. This situation can be corrected by producing higher density meshes during 3D scanning.

7. Conclusions

In this paper, we have presented an alternative approach to produce a point hierarchy based on point relaxation which is faster than existing approaches for point hierarchy creation. Our method first creates a patch subdivision scheme, and uses the patch connectivity information to create a graph where particle 'tokens' are distributed among the nodes of the graph. The tokens are distributed on the surface of the model by using a point relaxation technique where tokens are only allowed to locations defined by the nodes of a graph derived from the polygonal mesh. Our approach has the advantage that no polygonal 'unfolding' and

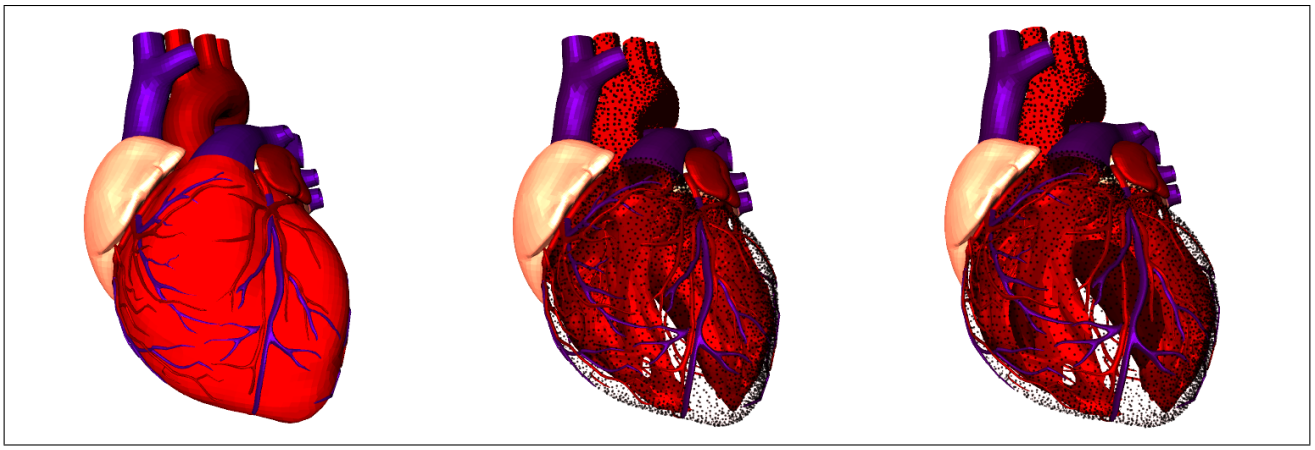

Figure 7: **Left: the heart model illustrated with an opaque surface. Following images: frames from the animation of the heart beating, where the outer heart layer has been replaced by a semi-transparent stippled layer to allow a look at the changes taking place inside the model.**

no geometrical point displacement is required to displace the tokens. In an initial step, polygons are organized into patches, and these are fused together to create a patch hierarchy. We apply a graph-based approach for point relaxation at each level of the patch hierarchy to create a point hierarchy. Finally, the point hierarchy is given as input for the stipple renderer.

We use the point hierarchy to improve the stipple distribution in our frame-coherent stippling animations, but the whole family of techniques in NPR which perform frame-coherent particle distribution, like artistic [8, 5] and painterly [9] rendering could benefit from this relaxation approach. For example, the point hierarchy could be used for evenly distributing paint strokes on the surface of a model, or for providing levels of detail for stroke-based rendering systems in the pen-and-ink style.

Acknowledgments

Many thanks to Oliver Deussen for proposing the use of relaxation techniques to improve the quality of the point distribution; thanks to Lourdes Peña Castillo for her editorial assistance and support; to R. Kuhn from the Office of Archaeology of Saxony-Anhalt for providing the archaeological samples, to Armin Botcher (heart animations), E. Trostmann from the Fraunhofer IFF Magdeburg (3D scanning) and to the staff at the Institute for Simulation and Graphics, University of Magdeburg.

References

[1] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(3):40–51, 2000. `http://www.eg.org/EG/CGF/volume19/issue3`.

[2] F. Dong, G. J. Clapworthy, H. Lin, and M. A. Krokos. Nonphotorealistic rendering of medical volume data. In *IEEE Computer Graphics and Applications Special Issue on Non-photorealistic Rendering*, July / August 2003.

[3] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *SIGGRAPH 2001 Conference Proceedings*, pages 49–58. ACM Press, 2001.

[4] V. Interrante, H. Fuchs, and S. M. Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *IEEE Visualization*, pages 52–, 1995.

[5] M. Kaplan, B. Gooch, and E. Cohen. Interactive artistic rendering. In *Proceedings of the First International Symposium on Non-photorealistic Animation and Rendering*, pages 67–74. ACM Press, 2000. `http://www.cs.utah.edu/npr/papers.html`.

[6] A. Lu, C. Morris, D. Ebert, P. Rheingans, and C. Hansen. Non-photorealistic volume rendering using stippling techniques. In *IEEE Visualization 2002 Conference Procedings*, 2002.

[7] E. B. Lum and K.-L. Ma. Hardware-accelerated parallel nonphotorealistic volume rendering. In *Proc. of the 2nd. International Symposium on Non-photorealistic Animation and Rendering*, pages 67–74. ACM Press, 2002.

[8] L. Markosian, B. J. Meier, M. A. Kowalski, L. S. Holden, J. D. Northrup, and J. F. Hughes. Art-based rendering with continuous levels of detail. In *Proc. of the 1st. International Symposium on Non-Photorealistic Animation and Rendering*, pages 59–66. ACM Press, 2000.

[9] B. J. Meier. Painterly rendering for animation. In *SIGGRAPH 96 Conference Proceedings*, pages 477–484. ACM Press, 1996.

[10] O. Meruvia. Visibility preprocessing using spherical sampling of polygonal patches. In *Eurographics'2002 Short Paper Proceedings*, 2002.

[11] O. Meruvia, B. Freudenberg, and T. Strothotte. Real-time, animated stippling. In *IEEE Computer Graphics and Applications Special Issue on Non-photorealistic Rendering*, July / August 2003. `http://isgwww.cs.uni-magdeburg.de/~oscar/`.

Figure 8: **The savings pot and the colored mosaic (left column) were found during archaeological excavations and scanned in 3D. Top middle: the savings pot. Top right: silhouette enhancement of the mosaic model using surface normals. Bottom middle: detail enhancement by highlighting sharp edges. Bottom right: stippling on top of the textured mosaic model.**

[12] V. Ostromoukhov. Digital halftoning over a hexagonal grid. In *Proc. of Graphics Interface 2002*. Graphics Interface, 2002.

[13] A. J. Secord. Weighted voronoi stippling. In *Proc. of the 2nd International Symposium on Non-Photorealistic Animation and Rendering*, pages 37–43. ACM Press, 2002.

[14] A. J. Secord, W. Heidrich, and L. Streit. Fast primitive distribution for illustration. In *Proc. of the 13th Eurographics Workshop on Rendering*, pages 215–226. Eurographics Association, Eurographics Association, 2002.

[15] C. Soler, M.-P. Cani, and A. Angelidis. Hierarchical pattern mapping. In *SIGGRAPH 2002 Conference Proceedings*, pages 673–680. ACM Press, 2002.

[16] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH 91 Conference Proceedings*, pages 289–298. ACM Press, 1991.

[17] G. Turk. Re-tiling polygonal surfaces. In *SIGGRAPH 92 Conference Proceedings*, pages 55–64. ACM Press, 1992.

[18] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH 94 Conference Proceedings*, pages 269–277. ACM Press, 1994.