Dynamic point distribution for stroke-based rendering

David Vanderhaeghe, Pascal Barla, Joëlle Thollot, François X. Sillion

▶ **To cite this version:**

David Vanderhaeghe, Pascal Barla, Joëlle Thollot, François X. Sillion. Dynamic point distribution for stroke-based rendering. Eurographics Symposium on Rendering, Jun 2007, Grenoble, France. pp.139-146, 10.2312/EGWR/EGSR07/139-146 . inria-00300317

HAL Id: inria-00300317

https://inria.hal.science/inria-00300317v1

Submitted on 18 Jul 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Eurographics Symposium on Rendering (2007)
Jan Kautz and Sumanta Pattanaik (Editors)

Dynamic point distribution for stroke-based rendering

David Vanderhaeghe, Pascal Barla, Joëlle Thollot and François X. Sillion

ARTIS - INRIA Grenoble University

Abstract

We present a new point distribution algorithm that is well adapted to stroke-based rendering systems. Its main characteristic is to deal efficiently with three conflicting constraints: the distribution of points should retain a good repartition in 2D; their motion should tightly follow the target motion in the underlying scene; and as few points as possible should be added or deleted from frame to frame. We show that previous methods fail to meet at least one of these constraints in the general case, as opposed to our approach that is independent of scene complexity and motion. As a result, our algorithm is able to take 3D scenes as well as videos as input and create non-uniform distributions with good temporal coherence and density properties. To illustrate it, we show applications in four different styles: stippling, pointillism, hatching and painterly.

1. Introduction

Creating animations by hand requires the artist to paint or draw each frame independently. In complex styles such as painterly, watercolor, airbrush, stippling or hatching, this process does not take into account frame-to-frame coherence of "brush strokes". This can lead to vibrating, evocative, and rich visual experiences when the technique is mastered (see the work of George Schwizgebel, Alexander Petrov, Frederic Back or Bill Plimton for instance). But this noise may also become annoying for the viewer, and artists may want to avoid it while keeping the richness of a complex style. However, ensuring frame-to-frame coherence is a very tedious and difficult task to achieve by hand.

One of the goals of non-photorealistic rendering is precisely to assist the user in complicated and tedious tasks, and automate most of the non-creative process. It then found a natural application in the creation of animations in various styles. Among these, stroke-based rendering [Her03] has received much attention, since it consists in assisting the artist in creating animations where images are composed of many small 2D drawing primitives; a task that is typically time-consuming when done by hand. Previous techniques rendered animations in specific styles, either from an input animated 3D scene or from a video. In this paper, as we consider both types of input, we will refer indiscriminately to a 3D scene or a video as the *underlying scene*.

By considering only small drawing primitives, a natural approach, taken by most of previous work, is to attach them to anchor points in the underlying scene. Hence, when put in motion, these anchor points will guide the primitives in the picture and give a sense of movement. While the definition of a primitive might vary depending on the chosen style, defining the anchor point distribution from frame-to-frame is common and essential to any stroke-based system. It also raises multiple, conflicting constraints: the distribution of anchor points should retain a good repartition in 2D; their motion should tightly follow the target motion in the underlying scene; and as few points as possible should be added or deleted from frame to frame, a property often referred to as *temporal coherence*.

Naive distributions cannot satisfy all of the above requirements. For instance, if one distributes new drawing primitives at each frame, the first constraint might be easily satisfied, but since no motion is matched, and no coherence is ensured, the remaining two constraints are violated. Another approach would be to use a single fixed distribution for the whole animation (good repartition and of course perfect coherence), but it will obviously violate the motion constraint. More sophisticated approaches have been proposed in the literature, but as explained in Section 2, they fail to satisfy all the constraints in general cases. This motivates the development of a more flexible distribution method for the creation of stroke-based animations.

© The Eurographics Association 2007.

Figure 1: *Our algorithm allows to distribute points on arbitrary scenes with non-uniform density and temporal coherence.* ***(a)*** *A rigid 3D model.* ***(b)*** *a 3D scene composed of different objects.* ***(c)*** *A deformable model.*

In this paper, we present a new distribution technique that works for any kind of input, and achieves interactive frame-rates *independently* of the scene complexity. Regarding the afore-mentioned constraints, our algorithm, presented in Section 3, provides the following contributions:

- Generation of non-uniform Poisson-disk distributions with blue-noise properties, necessary for applications such as stippling or pointillism;
- Matching of any motion (rigid- or soft-body deformations, vector flows, occlusions), making the approach tractable for arbitrary 3D scenes and videos;
- Control of temporal coherence via the management of coarse-to-fine transitions when anchor points need to be added or deleted.

To achieve these goals, we sample stroke positions in the image plane and project them to the underlying scene to apply the scene motion. At each frame, previous samples points are reprojected to the image plane and updated with respect to an importance function. New points are inserted if needed. Our main results, presented in Section 4, show the advantages of our distribution over previous approaches to primitive distribution. However, for the purpose of illustration, we also show in Section 5 how our distribution algorithm can be used to render animations in various, albeit simple, styles.

2. Previous work

As our goal here is to compare with other distribution techniques, we first present classic methods employed in the sampling literature, before reviewing previous approaches encountered in stroke-based rendering.

Sampling techniques usually rely on an importance map to create non-uniform point distributions (with more points distributed in areas of higher importance). When applied to time-varying importance maps, either for video environment map sampling [HSK*05] or for primitive distribution [SHS02], the samples follow the motion of the peaks of the time-varying importance. This approach is well-suited for environment map sampling for instance, but in the case of primitive distribution, it results in points floating over the surface as the viewpoint or importance change.

In contrast, we are interested in having points follow a given motion, while their distribution matches a distinct importance map: for instance in stippling, points should follow object motion, with more points in dark regions of the image. The recent tile-based distribution method of Kopf et al. [KCODL06] shows such a behavior: they propose a real-time blue-noise point distribution technique that matches very accurately any importance map. However, because their method relies on pre-computed tileable point distributions, it can only directly deal with 2D rigid motions (panning and zooming inside an image). Adapting the method to more general motions, for instance via non-uniform warping, is not trivial. In this paper, we thus take an alternative approach that does not rely on any precomputation.

Unlike sampling techniques, stroke-based rendering systems precisely focused on ways to convey motion. Different techniques have been proposed, depending on whether a 3D scene or a video is given as input. Because of the amount of strokes we seek to distribute, we only focus here on *automatic* distributions. In comparison, in the work of Kalnins *et al.* [KMM*02] or Daniels [Dan99], the user has to manually place each stroke onto 3D surfaces and provide levels-of-detail for each of them.

For *3D scenes*, anchor points are usually directly distributed onto 3D object surfaces and projected in the picture plane to guide primitives motion. Introduced by Meier [Mei96] for painterly rendering using a static 3D distribution (hence limited in the range of possible viewpoints), the method has been extended by various authors to adapt to viewpoint changes [CRL01, PFS03, NS04, HS04]. They all first construct a hierarchy of 3D anchor points over an object's surface in pre-process. Then they choose a "cut" in the hierarchy at runtime so that the resulting projected points best match the required distribution (e.g. specified via an importance map). While this approach has the advantage of accurately following objects motion with a very good temporal coherence, it suffers from limitations that make it impractical for general scenes. First, the hierarchy construction, built in pre-process, might be time-consuming for large models and some knowledge of "how close" any object can come to the point of view is needed to control its precision. Second, as each object is considered individually, distribu-

© The Eurographics Association 2007.

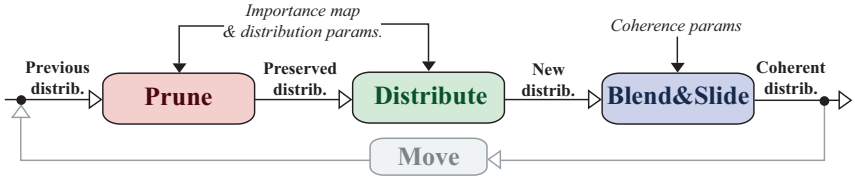


Figure 2: *The different steps of our dynamic distribution algorithm: pruning, distribution, blending&sliding. A coherent non-uniform distribution is dynamically updated from frame to frame regardless of how the scene moved: 3D objects can be deformed, viewpoint motion is not restricted, and motion in video can be provided via its optical flow.*

tions cannot overlap multiple objects (although it might be necessary, for instance with distant objects); they can only follow individual object motion with restricted deformations (no topology change, nor vector flows over surfaces for instance); and they become increasingly time- and memory-consuming with scene complexity. Finally, the repartition constraint cannot be ensured since the points are only guaranteed to appear roughly in the middle of neighbor points when projected to the picture plane [Pas03]. Moreover, distributions obtained this way sometimes present artifacts such as patterns, accumulation or grouping motifs (especially at grazing angles). As a result, 3D-based techniques, while very efficient at ensuring temporal coherence, fail to fully satisfy motion and distribution constraints.

For *videos*, on the other hand, since there are no objects to rely on, no pre-process is needed. Rather, as in the compelling painterly renderer of Hays and Essa [HE04], anchor points follow video optical flow. Provided the optical flow is a good approximation of motion, the resulting anchor point distributions satisfy the motion and temporal coherence constraints. However, this approach needs to be adapted to work with 3D scene input, as it does not take into account occlusions: for instance, if we use Hays and Essa's system on a motion flow extracted from a 3D scene, anchor points may move from one object to the other. Moreover, their method for dealing with distributions assumes a thick, opaque paint medium: multiple uniform distributions are painted in sequence, from coarse to fine. Not only this approach makes the system very slow (around 80 seconds per frame), but it will fail to adapt to other styles such as stippling or pointillism, since there are no continuous variations of anchor points density, and distributions layers are built independently so that two anchor points from different layers may be arbitrary close to each other.

The "animosaics" method of Smith *et al.* [SLK05] provide a distribution approach similar to [HE04], even if used in another context: they take a vector based animation as input and mainly focus on packing mosaics with a uniform distribution. Therefore, regarding our goals, the approach shares the same drawbacks as Hays and Essa's work. Video-based methods hence have important limitations concerning motion and distribution constraints.

To overcome the limitations of previous work we make use of an hybrid approach that combines advantages of 3D- and video-based techniques, while having distribution properties sufficiently close to the ones obtained with sampling techniques, at least in the case of stroke-based rendering.

3. Distribution algorithm

Our goal is to distribute drawing primitives in screen-space in such a way that they follow a given motion in the depicted scene without exhibiting temporal artifacts. To permit non-uniform densities, we also allow the user to specify an importance map I, a function that assigns to any point p in screen-space an importance value $I(p) \in [0,1]$. This function, which typically evolves during the animation, can either be defined by 3D scene properties (object IDs, depth, painted textures, etc) or screen-space properties (luminance, optical flow magnitude, etc).

We choose to ground our dynamic distribution on the static Poisson-disk distribution approach of McCool and Fiume [MF92]. As argued in Section 3.1, it is best adapted to the particular constraints of stroke-based rendering. We then explain in Section 3.2 how we extend this algorithm to deal with dynamic settings. The underlying idea, summarized in Figure 2, is that at each step of our algorithm, points are moved to the next frame according to available scene motion; they are then pruned using a Poisson-disk criterion, and holes in the distribution are filled. The main contribution of this method is that through this process, the updated distribution retains the properties of the static algorithm.

3.1. Static distribution

We first present the guidelines of the Poisson-disk distribution technique of McCool and Fiume [MF92], also called relaxation dart throwing. Standard dart throwing [Coo86] randomly distributes points in the image plane and keeps a point only if no other point is found within a disk of radius r around it. McCool and Fiume extended this algorithm with a hierarchical formulation: initially, points are randomly distributed with a large radius $r = r_{\max}$; once no more space has been found after n trials the radius is multiplied by $\alpha \in (0,1)$. The algorithm stops as soon as $r \leq r_{\min}$. The quality of the distribution can be controlled by the parameters n and α, which allow us to run our algorithm at interactive frame rates for previewing ($n = 100$, $\alpha = 0.9$), or create high quality animations for off-line rendering ($n = 1000$, $\alpha = 0.99$).

© The Eurographics Association 2007.

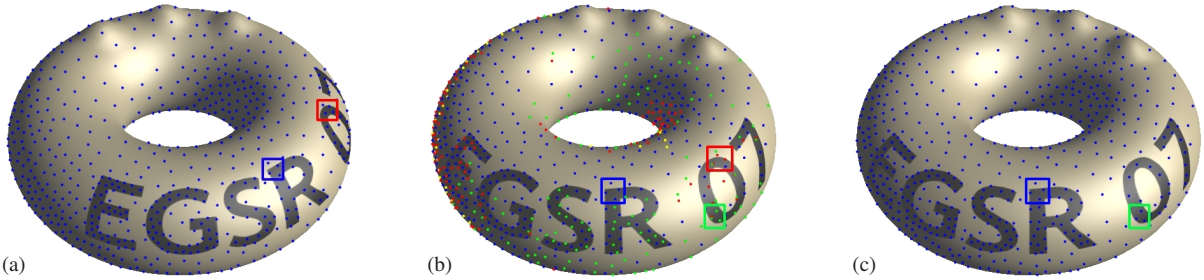


Figure 3: ***(a)*** *Points are initially distributed in screen-space, here using luminance to drive density. They are then projected onto the 3D object.* ***(b)*** *After they have been moved to the next frame, points are reprojected to screen-space. Red points are rejected based on a Poisson-disk criterion; blue points from the previous frame are still valid; green points are added to fill in the blanks; and yellow points are hidden.* ***(c)*** *The distribution after it has been updated. The red, green and blue frames locate corresponding points that have been respectively removed, added or preserved.*

Apart from having good statistical properties (see the survey on Poisson-disk distributions by Lagae and Dutre [LD06]) and being simple to implement, the algorithm is well suited to our problem: it enables non-uniform densities, provides a hierarchical distribution and can be built incrementally. In particular, the point density can be made to follow an arbitrary importance map I defined in screen-space, via the use of an importance criterion: p is kept only if $1-\frac{r(p)-r_{\min}}{r_{\max}-r_{\min}} > I(p)$, with $r(p) \in [r_{\min}, r_{\max}]$ the radius used to distribute it. This way, the user is able to finely control the distribution of points by manipulating $r_{\min}$ and $r_{\max}$.

3.2. Dynamic distribution

Up until now, we explained how points are distributed for a single frame. We start any animation with such a point distribution (Figure 3(a)). Then, we *move* points to the next frame. This produces clusters of points that we clean out by *pruning* unwanted points; but it also creates holes that we fill by *distributing* new points.

Depending on the input given to our system, motion is performed in different ways. To deal with complex 3D scene motions, we project distributed points from screen-space onto object surfaces, move objects (with any kind of motion, including skinning and deformations), and project points back to screen-space. In practice, we take as input triangle meshes, and the projection onto a surface is done by rendering at each frame buffers that hold mesh IDs, face IDs, and triangle barycentric coordinates; then for each point we obtain its corresponding IDs and coordinates simply by reading the buffers. In the next frame, the point is located in 3D and perspectively projected. This approach allows points to stick to their triangle even if they are severely distorted. For videos we simply translate points along an optical flow provided in input (we rely on a classical gradient based method available in Adobe® After Effects®).

Another important aspect of motion is occlusion: while for videos, points are considered hidden only when they fall out of the image dimensions, they can disappear behind other surfaces when dealing with 3D scenes. We perform a simple depth test to detect occlusions, and tag hidden points that will be treated specifically in the next step. Moreover, points falling onto the background in 3D scenes are considered hidden. Figure 3(b) shows points after they have been moved to a subsequent frame, with hidden points in yellow.

In both videos and 3D scenes, points can be either visible or hidden after they have been moved to the next frame. In the following, we only consider points that are still visible. In order to clean out clutters, we re-insert them in the order they have been previously distributed using a Poisson-disk criterion to reject unwanted samples. For each radius r_L corresponding to a level L of the hierarchical distribution, we check *all* the previously inserted points against the importance criterion of Section 3.1. Accepted points are re-inserted with radius r_L and we try to re-insert the remaining points at level $L-1$. The process stops as soon as all the points have been re-inserted or the finest level has been checked, usually rejecting a subset of the previous points. These rejected points are tagged as deleted (red points in Figure 3), while accepted points (in blue) are preserved so that the distribution is coherent from frame to frame.

Finally, holes still remain in the distribution due to appearing surfaces or optical flow dilation. We thus need to distribute new points to fill these holes with the same properties as in the static case of Section 3.1. Considering *all* the re-inserted points as part of the current distribution, we reuse McCool and Fiume's algorithm: each level L of the hierarchical distribution is processed in a top-down approach; points are randomly distributed with a radius r_L, until no more space has been found for n trials, discarding points that do not meet the importance criterion. The algorithm stops after the finest level, corresponding to $r_L \leq r_{\min}$, has been processed. This approach allows inserted points to appear in the highest possible level of the hierarchy, hence resulting in a better repartition.

© The Eurographics Association 2007.

3.3. Temporal coherence

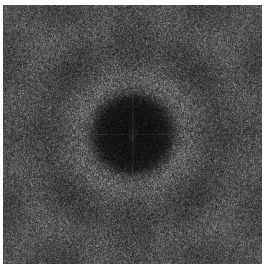
Using the algorithm described in Section 3.2 guarantees that most of the previous points are kept from frame-to-frame. However, we cannot avoid inserting or deleting *some* points, as motion and density are usually conflicting constraints. Hence, in order to further improve the temporal coherence of our approach, we propose two mechanisms: blending and sliding.

Blending: Inserting or deleting points as soon as this is requested by our distribution algorithm often results in disturbing popping artifacts. As in previous work [PFS03, HE04, KCODL06], we rather blend points in or out using smooth transitions. A blending duration is assigned to each inserted and deleted point, that is incremented or decremented at subsequent frames. Therefore, the distribution algorithm only tags points for being inserted or deleted, and leaves the task of performing the operation to the blending mechanism.

The main difference in our approach compared to previous work is that we only focus on blending *duration*, so that the way blending is actually performed can make use of this value, but vary depending on the application (see Section 5). Moreover, we propose to distinguish between two different cases where blending occurs: density and occlusion events. For occlusion and dis-occlusion events, points are deleted (resp. inserted) directly. In practice, detecting which points become occluded in the current frame is easy (using the z-buffer), but finding out what points are dis-occluded requires to store some more information (z-buffer and transformation matrices from the previous frame). For points that appear or disappear by density, we developed another mechanism that assigns blending durations proportional to the radius of the inserted or deleted point. In practice, it produces smooth transitions where "details" are added progressively, from coarse to fine.

Sliding: Another problem may appear when we remove a point due to the Poisson-disk constraint, and it leaves enough place to add another point in its neighborhood: even with blending enabled, it can result in a flickering artifact during animation. We avoid this problem by matching a pair $\{p_d, p_i\}$ of deleted and inserted points, where p_d is the closest deleted points to p_i in screen-space, using the radius of p_i as a threshold. Then instead of removing p_d, we allow it to *slide* linearly towards the position of the newly inserted one. We only perform sliding for density events though, otherwise hidden points would tend to slide across occlusion boundaries.

In practice, we track the original points p_d and p_i and linearly interpolate between their projected positions. This results in a projected distance $d_p(t)$ between the sliding point and its target position, with $t \in [0, t_s]$ where t_s is a sliding duration set by the user. While this would work for simple motions, more complex movements such as rotations can stretch out the projected pair, hence increasing $d_p(t)$; which is opposite to the goal of the sliding mechanism. Therefore, we also clamp $d_p(t)$ to the distance that would be obtained with a straightforward 2D linear interpolation, $d_p(0)(t - t_s)/t_s$. This way, the interpolated position gets progressively closer to the inserted point, while still conveying object motion.



Relative radius statistics		
α/n	RDT	ours
0.9/100	0.69	0.67
0.99/1000	0.80	0.74

Figure 4: ***Left:*** *Power spectrum of our point distribution method obtained by averaging periodograms for each frame of a test animation (central peak removed).* ***Right:*** *Relative radius statistics for static relaxation dart throwing (RDT) and our dynamic point distribution method.*

4. Results

Figure 1 shows stills from dynamic distributions created with our algorithm, either from 3D scenes or video input. The full animations can be viewed in the accompanying video (available at `http://artis.imag.fr/Publications/2007/VBTS07a/`). The point distribution tightly matches required motion and density in all cases, while exhibiting good temporal coherence.

The sampling times are independent of scene complexity, and our method requires no pre-process. The performance of our distribution algorithm thus only depends on image dimensions and the choice of the n and α parameters that drive points density. Note that as in [MF92], we use a uniform grid to speed up distance tests during distribution. For interactive manipulation ($n = 100/\alpha = 0.9$) we get frame rates between 5 and 10 fps, while for off-line rendering ($n = 1000/\alpha = 0.99$), we are able to render point distributions at approximately 1 fps. These parameter settings are the ones advocated by McCool and Fiume [MF92] and in both cases, we render images at 720×540 resolution, and get performances similar to them.

We now present statistical and visual evaluations of our distribution algorithm in comparison with previous work.

4.1. Statistical evaluation

We first want to check that our dynamic distribution has qualities similar to that of McCool and Fiume's original algorithm [MF92]. A study by Lagae and Dutré [LD06] proposed to use two measurements to evaluate Poisson-disk distributions: relative radius that measures the amount of "packing" in the distribution; and power spectrum that gives infor-

© The Eurographics Association 2007.

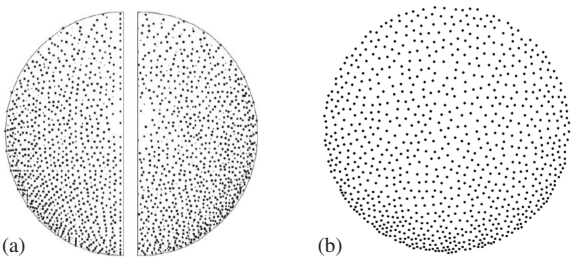


Figure 5: *Comparison with 3D-based distributions.* ***(a)*** *The technique of Pastor et al. [PFS03] (figure taken from their paper) cannot ensure a good repartition (left half), even after randomization (right half).* ***(b)*** *Our approach, in comparison, produces distributions with blue-noise properties.*

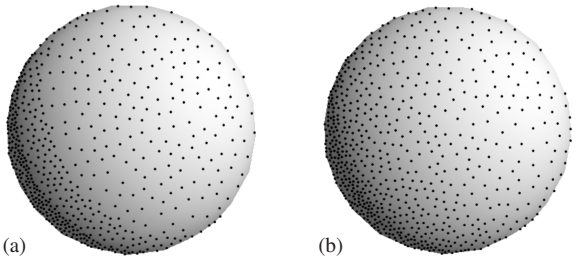


Figure 6: *Non-uniform density.* ***(a)*** *Using* $\alpha = 0.5$ *results in a quantification effect while* ***(b)*** *with* $\alpha = 0.9$ *the importance gradation is faithfully reproduced.*

mation about its frequency content. Regarding these measurements, that are only valid for *uniform* distributions, McCool and Fiume's static distribution method is one of the best available techniques. We found no similar evaluation tool for non-uniform distributions.

We thus measured the power spectrum and relative radii for the frames of a test animation using the torus model of Figure 3, and a dynamic *uniform* distribution that fills the image (using $r_{\min} = 2$, $r_{\max} = 10$, $\alpha = 0.99$ and $n = 1000$). We expected that our approach, by introducing coherence of points distribution, would result in worse relative radii, or reveal regularities in the power spectrum. However, it did not show significant influence (see Figure 4): the spectrum clearly exhibits blue-noise properties (we removed the central peak for clarity as in [LD06]) and relative radii results are only slightly better in the static case. Note that as our distribution is not periodic, horizontal and vertical lines appear in the spectrum.

While these measures do not outperform static Poisson-disk distribution methods, they show that our technique has comparable statistical qualities while it satisfies additional constraints, i.e. scene motion and temporal coherence.

4.2. Visual evaluation

The previous section compared our approach to previous techniques using uniform distribution statistics. However, such comparison is sometimes infeasible, either because no statistics are available, or because we deal with non-uniform distributions. In this section, we thus resort to visual evaluation, to compare our approach with object-based methods and illustrate its behavior with non-uniform densities.

The pioneering work of Meier [Mei96] produced remarkably stable object-based distributions: this is because her distribution is static, and done once in pre-process. A similar image-based approach, obtained using a static 2D distribution, would introduce the so-called "shower-door effect". Thus the object-based approach of Meier gives much better results, albeit for the limited range of viewpoints where the distribution is dense enough, or not too packed.

But as soon as we need to insert or remove points to allow camera and object motions to be unconstrained, we lose stability and need blending. This is a problem common to object- and image-based approaches: in the first case, points are distributed on a surface and selected/deselected depending on a refinement function; while in our case, points are added/deleted using a Poisson-disk criterion. While both approaches have similar problems, object-based techniques have the additional burden of finding a good refinement function. And as shown in Figure 5(a), there is no simple way to get a good 2D distribution starting from a 3D distribution, even after introducing randomization as in Pastor et al. [PFS03]; while our method, shown in Figure 5(b), simply avoids this problem by directly distributing points in the picture plane.

Non-uniform distributions created with our approach depend on the values of $r_{\min}$, $r_{\max}$ and α. While $r_{\min}$ and $r_{\max}$ control the density in regions where the importance map is equal to 1 and 0 respectively, the value of α influences the continuity of the non-uniform distribution. As shown in Figure 6, a low value for α has an effect similar to quantization, while a reasonable value results in smooth gradations. In practice, we found that using $\alpha > 0.9$ gives very good results in most cases.

5. Applications

We now present some applications of our distribution algorithm that work with 3D scenes or video input (with optical flow). Each of them target a different visual style: stippling, pointillism, hatching or painterly. Note that we are concerned here with a very low-level notion of style: stipples are dots of varying size, pointillism introduces color, hatching needs to deal with orientation (e.g. for cross-hatching), and painterly refers to the use of small paint strokes of varying color and orientation. There is much more research to do to characterize how a style can be specified, but this falls out of the scope of this paper.

Our stroke-based rendering prototype system is similar in spirit to previous painting systems [Mei96, DOM*01, HE04, PFS03] in that it separates low-level strokes drawing from high-level style control. At each frame, stroke parameters are

© The Eurographics Association 2007.

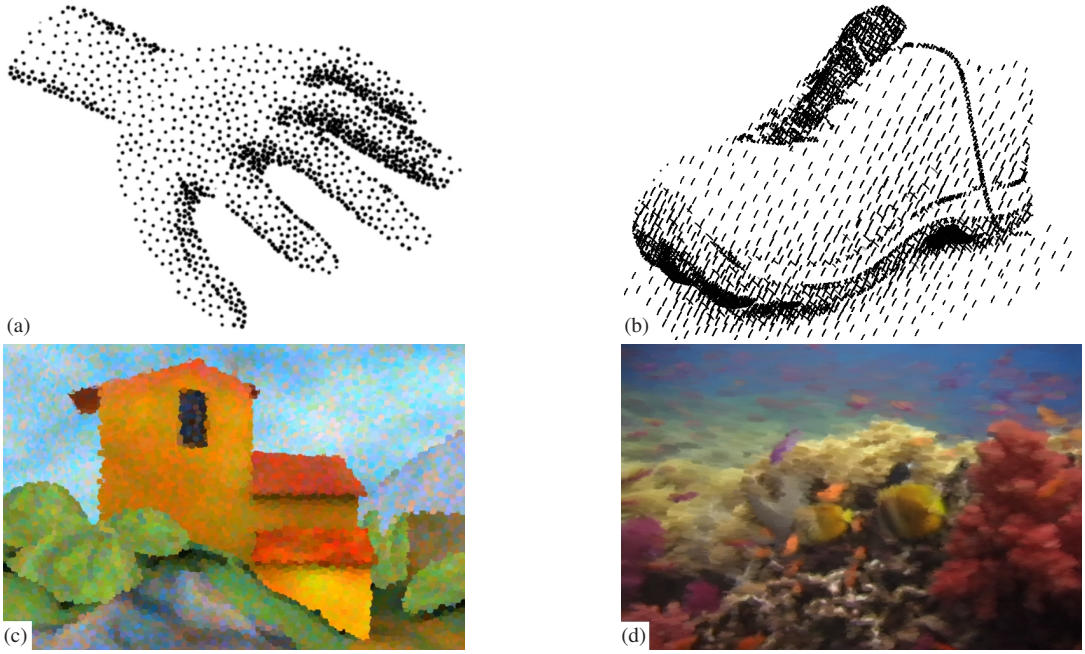


Figure 7: *Our distribution can be used as a basis for various styles:* ***(a)*** *Stippling rendition of a deformable hand model,* ***(b)*** *a hatching style applied to a rigid shoe model,* ***(c)*** *pointillist rendering of a scene consisting in multiple objects and* ***(d)*** *a painterly rendering of a video that follows optical flow.*

computed for the whole image and stored in a style buffer. Stroke anchor points are then distributed using our dynamic algorithm. And finally, primitives are drawn by taking their parameters at their anchor point location in the style buffer. Note however that nothing is done to make stroke parameters evolve coherently, hence stroke orientation, length or thickness may change suddenly in the animations. Such temporal artifacts are specific to every new style and we plan to study temporal coherence for individual styles in future work.

Stroke density is a special parameter that deserves more attention since it is common to all styles. In our system, it is controlled by a time-varying importance map that is input to the distribution algorithm at each frame. As it plays an important role in anchor points distribution, we implemented a variety of importance mappings in the form of GPU shaders, so that new mappings can be added easily. Each style uses a different importance mapping, as shown in Figure 7 and in the accompanying video.

Stippling: For stippling, we use an approach similar to Pastor et al. [PFS03]: stipples are simply black dots of varying size; their density follows the luminance information coming from the scene (our importance map); and stipples are blended in or out by making them grow or shrink.

Pointillism: Pointillism is a variant of stippling where we make use of color information to color the stipples, and the point distribution is close to uniform. The points are added or removed by alpha blending. Moreover, as in [Mei96], we jitter each point's color in the Lab color space.

Hatching: Hatching introduces two new parameters: strokes length and orientation. In order to make the distribution visible in our results, we intentionally chose a simple style: stroke length and orientation are uniform across the image. To create cross-hatching, we simply create two independent distributions of strokes with different orientations. Moreover, as in previous work [HP00, HE04], we cut strokes at object boundaries, that we measure as depth discontinuities.

Painterly: Finally, painterly rendering combines all the characteristics of previous styles at once (color, length and orientation) and adds another parameter: stroke thickness. We again set the parameter values to uniform to better show the underlying distribution. Paint strokes are blended using the half-toning technique of Durand et al. [DOM*01].

6. Discussion and Future Work

The work presented in this paper proposes an new approach to the dynamic distribution of anchor points for stroke-based rendering: it consists in a hybrid method that combines the advantages of distributing points in image space and the richness of object-space motions.

In the near future we plan to adapt our algorithm to work with more complex motions, such as specular reflexions motion, or natural phenomena (e.g. fluid, smoke, etc). Indeed, the only requirement in our case is to be able to project a point to the scene, and after it has been moved, to project it back to the picture plane.

© The Eurographics Association 2007.

Another area of research resides in using more complex strokes, such as long paint strokes, or mosaics. Already, when dealing with stippling, we cannot ensure an accurate tone reproduction, even if we get satisfying results in practice. While with hatching or painterly styles, one interesting question would be to adapt the distribution to primitives direction (e.g. using "Poisson-ellipses"). But this will not be enough as soon as long strokes will be involved. In this case, one possibility would be to use our approach as an underlying structure, and attach long strokes to multiple anchor points. And in the case of mosaics, another issue is raised: we need to meet boundary constraints between primitives. Our approach already offers good packing properties, but additional mechanisms are needed to make mosaic primitives "touch" each other without inter-penetrating.

Finally, while our distribution, by construction, makes use of the coherence of points from frame to frame, we additionally proposed two temporal coherence mechanisms: blending and sliding. Our simple assumption being that by reducing the number of insertions and deletions, and making the remaining ones less noticeable, we effectively enhance coherence from frame to frame. Interestingly, while temporal coherence is a recurrent problem in non-photorealistic rendering, there is no standard way to measure it to our knowledge, making a formal study an interesting avenue of future work.

Acknowledgments

We would like to thank Florent Moulin and Laurence Boissieux for the 3D models; and Lionel Baboud, Cyril Soler, Kartic Subr and Kaleigh Smith for their suggestions.

References

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1 (1986), 51–72.

[CRL01] CORNISH D., ROWAN A., LUEBKE D.: View-dependent particles for interactive non-photorealistic rendering. In *Proceedings of Graphics Interface 2001* (2001), pp. 151–158.

[Dan99] DANIELS E.: Deep Canvas in Disney's Tarzan. In *SIGGRAPH 99 Conference Abstracts and Applications* (1999), p. 200.

[DOM*01] DURAND F., OSTROMOUKHOV V., MILLER M., DURANLEAU F., DORSEY J.: Decoupling strokes and high-level attributes for interactive traditional drawing. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (2001), pp. 71–82.

[HE04] HAYS J., ESSA I.: Image and video based painterly animation. In *NPAR'2004: international symposium on non-photorealistic animation and rendering* (2004), pp. 113–120.

[Her03] HERTZMANN A.: A survey of stroke-based rendering. *IEEE Computer Graphics and Applications 23*, 4 (July/August 2003), 70–81. Special Issue on Non-Photorealistic Rendering.

[HP00] HERTZMANN A., PERLIN K.: Painterly rendering for video and interaction. In *NPAR'2000: international symposium on non-photorealistic animation and rendering* (2000), pp. 7–12.

[HS04] HALLER M., SPERL D.: Real-time painterly rendering for MR applications. In *GRAPHITE '04* (2004), pp. 30–38.

[HSK*05] HAVRAN V., SMYK M., KRAWCZYK G., MYSZKOWSKI K., SEIDEL H.-P.: Importance Sampling for Video Environment Maps. In *Rendering Techniques 2005 (Proceedings of the Eurographics Symposium on Rendering)* (2005), pp. 31–42,311.

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive wang tiles for real-time blue noise. *ACM Transactions on Graphics 25*, 3 (2006).

[KMM*02] KALNINS R. D., MARKOSIAN L., MEIER B. J., KOWALSKI M. A., LEE J. C., DAVIDSON P. L., WEBB M., HUGHES J. F., FINKELSTEIN A.: WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002) 21*, 3 (July 2002), 755–762.

[LD06] LAGAE A., DUTRÉ P.: *A Comparison of Methods for Generating Poisson Disk Distributions*. Report CW 459, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2006.

[Mei96] MEIER B. J.: Painterly rendering for animation. In *SIGGRAPH'96* (1996), pp. 477–484.

[MF92] MCCOOL M., FIUME E.: Hierarchical poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface '92* (1992), pp. 94–105.

[NS04] NEHAB D., SHILANE P.: Stratified point sampling of 3d models. In *Eurographics Symposium on Point-Based Graphics* (June 2004), pp. 49–56.

[Pas03] PASTOR O. E. M.: *Frame-Coherent 3D Stippling for Non-photorealistic Computer Graphics*. PhD thesis, Otto-von-Guericke University of Magdeburg, Germany, October 2003.

[PFS03] PASTOR O. M., FREUDENBERG B., STROTHOTTE T.: Real-time animated stippling. *IEEE Computer Graphics and Applications 23*, 4 (2003).

[SHS02] SECORD A., HEIDRICH W., STREIT L.: Fast primitive distribution for illustration. In *Rendering Techniques 2002 (Proceedings of the Eurographics Symposium on Rendering)* (2002), pp. 215–226.

[SLK05] SMITH K., LIU Y., KLEIN A.: Animosaics. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 201–208.

© The Eurographics Association 2007.