Computer Graphics International – Short Papers (2009)
Geoff Wyvill (Program Chair)

Voronoi Video Stylisation

Christian Richardt and Neil A. Dodgson

Computer Laboratory, University of Cambridge, United Kingdom

Figure 1: Examples of our video stylisation approach using 200,000 Voronoi sites (0.16% of the video's pixels): *(a)* the original video frame, *(b)* processed using our "patchwork" rendering style with average 2D cell colour, *(c)* processed using the same rendering style but now with average 3D cell colour, which causes "motion blur" artefacts.

Abstract

We introduce a novel video stylisation framework based on decomposing the video volume into 3D Voronoi cells. Colour samples are taken for each cell in a video, leading to an efficient and sparse video representation. Based on the same representation, our framework affords a wide variety of artistic rendering styles. We present two families of such styles for reconstructing stylised video frames: (1) using neighbouring Voronoi cells, and (2) a derived per-frame Delaunay-like triangulation. Our framework provides a flexible foundation for automatic video stylisation that is artistically expressive and temporally coherent.

1. Introduction

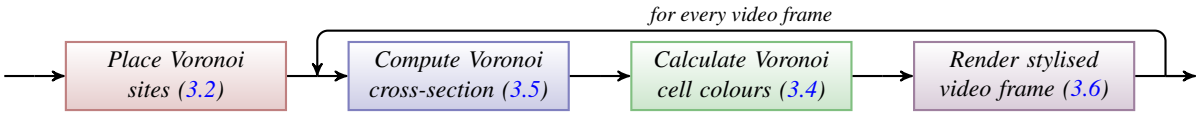
The Voronoi diagram is a versatile geometric data structure with many applications. We treat videos as spatiotemporal video volumes and tessellate them using a three-dimensional Voronoi diagram. Based on this discrete video representation, we introduce a novel video stylisation framework that supports a large variety of artistic rendering styles.

We see three main applications of our video stylisation framework. Its flexibility enables users to explore automated non-photorealistic rendering (NPR) styles for producing appealing video effects; the sparsity of our video representation makes it a viable NPR video compression scheme, although we have not look at this in detail; and our work provides useful insights into the suitability of Voronoi diagrams for video stylisation and compression.

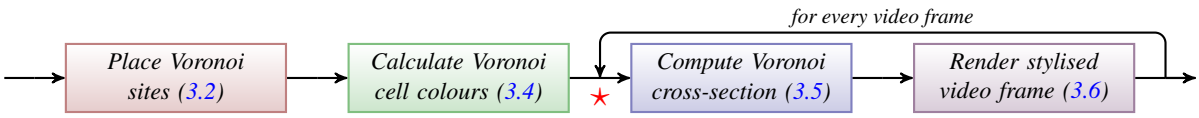
1.1. Contributions

We present a novel video stylisation framework based on spatiotemporal Voronoi diagrams, and describe two families of rendering styles that are based it. One family of styles is based directly on the Voronoi diagram and uses neighbouring cells for colouring pixels. The other family of styles colours pixels using per-frame Delaunay-like triangulations.

We also show that the spatiotemporal Voronoi diagram is a flexible video representation that can easily be modified and extended using custom distance functions, as well as site placement and colour sampling approaches. It is this flexibility that facilitates a large diversity of video rendering styles. We also make observations about how to extend image-based stylisation approaches to videos and why this can lead to "motion blur" artefacts.



(a) On-line processing: colours of all Voronoi cells are computed on-the-fly, as needed by the video stylisation step.



(b) Off-line processing: colours of all Voronoi cells are pre-computed, to produce a sparse video representation (⋆) suitable for stylisation.

Figure 2: Outline of the Voronoi video stylisation process: (a) on-line processing and (b) off-line processing.

2. Related Work

A Voronoi diagram [Vor08] is a tessellation of space defined by a set of point sites. These sites define Voronoi cells, each of which contains all points that are closer to one site than to any other site. Aurenhammer [Aur91] surveys Voronoi diagrams and the algorithms for computing them. The dual of the Voronoi diagram is the Delaunay triangulation [HD06]. It can be obtained by using the Voronoi sites as vertices and adding edges between sites of adjacent cells.

There are two main approaches for representing and computing Voronoi diagrams. The continuous representation treats the boundaries of Voronoi cells as geometric curve segments; the discrete representation maps discrete pixels or voxels to their nearest Voronoi seeds. Images and videos are pixelated, so we use a discrete representation. While discrete Voronoi diagrams can be efficiently computed using z-buffering on a GPU [HKL*99], for simplicity, we instead use an extension to 3D of Danielsson's algorithm [Dan80].

Our work is inspired by Grundland and Klein, who use Voronoi diagrams for stylising images and videos respectively. Grundland et al. [Gru07, GGD08] describe an image stylisation framework based on Voronoi diagrams and Delaunay triangulations. We extend this to video, using 3D Voronoi diagrams. Klein et al. [KSC*01, KSFC02] were the first to use a 3D Voronoi diagram for stylising videos using flat shaded or textured rendering. Their earlier work [KSC*01], on which we build, constructs the 3D Voronoi diagram one 2D slice at a time. Their later work [KSFC02] employs Voronoi diagrams induced by the intersection of point trajectory with the rendering plane.

Voronoi diagrams have been used for artistic effects from mosaics to stippling. For example, Haeberli [Hae90] describes a system that allows the user to place drawing primitives on a virtual canvas and colour them according to an underlying image. When using z-buffered cones as drawing primitives, the result is a coloured-in Voronoi diagram.

Litwinowicz [Lit97] was the first to attempt video stylisation with virtual brushstrokes that move with the underlying video. To ensure a roughly uniform brush density, he used a Delaunay triangulation to insert strokes in gaps.

3. The Voronoi Video Framework

3.1. Background

Naïvely applying Grundland et al.'s stylisation framework [GGD08] to every frame in a video would cause flickering, because the sites are different in all frames. Static Voronoi sites, on the other hand, would lead to the "shower door" effect, because the sites does not adapt to the video content. This can be mitigated by moving the sites between frames, for example using optical flow [KSFC02].

Our approach, outlined in figure 2, extends Grundland et al.'s approach to 3D Voronoi diagrams in the video volume. The Voronoi sites are distributed statically over the video's volume, while the "current frame" plane sweeps through it along the time axis. This representation produces coherent results over time, as the cell cross-sections grow and shrink smoothly over time.

The density, shape and size of Voronoi cells is determined by the number of sites, their distribution and the distance function used. Once the Voronoi cells are computed, they can be used to sample the colour from the underlying video, and to render it in a variety of styles based on the Voronoi diagram or the corresponding Delaunay-like triangulation.

3.2. Placing Voronoi Sites

The density of Voronoi cells mostly depends on the placement of the Voronoi sites. Ideally, cells should be more dense near strong contrasts, to represent those more accurately. This requires video analysis, which remains future work. In the interest of real-time stylisation, we only considered site placement schemes that do not depend on the video content.

Placing sites in a regular Cartesian lattice produces a pixel-like look. This regularity can be avoided using pseudo-random placement. However, the non-uniform site density leads to visual artefacts in the Voronoi diagram. A more uniform site density is achieved using jittered placement, which randomly perturbs the position of sites placed in a regular lattice. This is what we chose to use in this paper.

Figure 4: Comparison of colour sampling approaches (500,000 seeds, 0.2%): *(a)* original frame, *(b)* point-sampled colours (notice the "motion blur"), *(c)* average 2D cell colour, *(d)* average 3D cell colour ("motion blur" again).

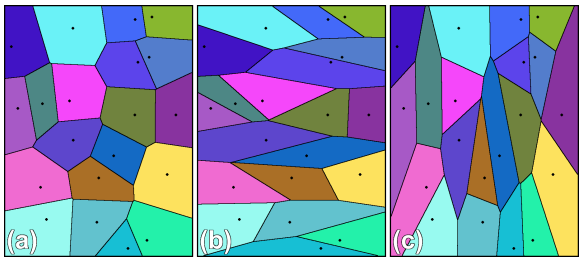


Figure 3: Comparison of distance functions for 20 Voronoi sites: *(a)* Euclidean distance, *(b)* and *(c)* anisotropic Euclidean distance with Δx weighted by *(b)* $k = \frac{1}{10}$, *(c)* $k = 10$.

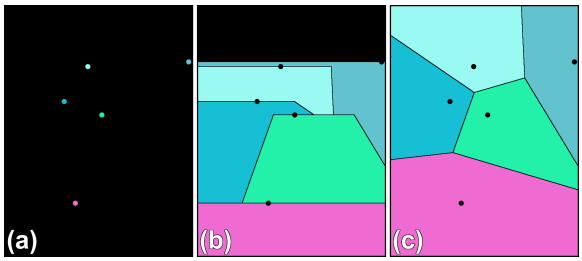


Figure 5: Extended Danielsson algorithm: *(a)* initialisation stage, *(b)* after downward scan, *(c)* after upward scan.

3.3. Distance Functions

The distance function determines the general shape of the Voronoi cells, because it determines which points belong to a cell (figure 3). In general, any scalar function of two points $\mathbf{p_1}$ and $\mathbf{p_2}$ could be used, but it is more meaningful to use metric functions, as they satisfy constraints we commonly associate with distance, such as the triangle inequality.

The Euclidean distance, $d_E = \|\mathbf{p_1} - \mathbf{p_2}\|$, is the standard distance function used in Voronoi diagrams. However, in videos, the spatial and temporal distances have different scales, because 100 pixels is not the same as 100 frames. To trade off the spatial and temporal distances, we considered a custom anisotropic Euclidean distance d_A^n, defined as

$$d_A^n(\mathbf{p_1}, \mathbf{p_2}) = \sqrt{\Delta x^2 + \Delta y^2 + n \cdot \Delta t^2}. \qquad (1)$$

For $n < 1$, spatial distance dominates temporal distance, leading to Voronoi cells that are more elongated in time (figure 3c). The smaller n, the stronger the "motion blur" or "shower door" effect. Conversely, Voronoi cells are shortened in time (figure 3d) for $n > 1$. For large values, the duration of cells is severely shortened, which causes flickering. The parameter n can be used to trade off the "shower door" and flickering effects. We found $n = 2$ to work well.

3.4. Colour Sampling

The next step is to extract the colours of Voronoi cells from the underlying video, so that they can be used for stylisation.

The simplest sampling approach is point sampling, which uses the colour at the location of the Voronoi site. This is fast, because it only accesses as many pixels as there are Voronoi cells. Unfortunately, the results are noisy (see figure 4b), as a site's colour may not be representative of all pixels in a cell. Instead, one can compute the average cell colour from all pixels in a cell (figure 4d). This is more representative than point-sampling at the expense of an increased running time.

Note that all sampling techniques that store a single colour per Voronoi cell show artefacts that resemble motion blur if there are fast motions in the underlying video. This is because the effects of motions are stretched out over the "duration" of a Voronoi cell, making them visible for longer than in the underlying video. See for example figure 4b.

These artefacts can be reduced by using a time-varying colour, such as the average colour of a cell's cross-section in the current frame. This is a reasonable representation of the underlying video at modest cost, as it be computed on-the-fly (figure 4c). This appears to be Klein's approach [KSFC02].

3.5. Slicing the Voronoi Diagram

We compute discrete 2D slices through the 3D Voronoi diagram on-the-fly. For this, we adapt Danielsson's two-pass scan-line algorithm [Dan80] for 3D Voronoi sites (figure 5):

Initialisation. Start with a Voronoi map V and a distance map D, holding the current per-pixel closest site and distance from it, which are initialised using $V(x,y) = \text{NIL}$

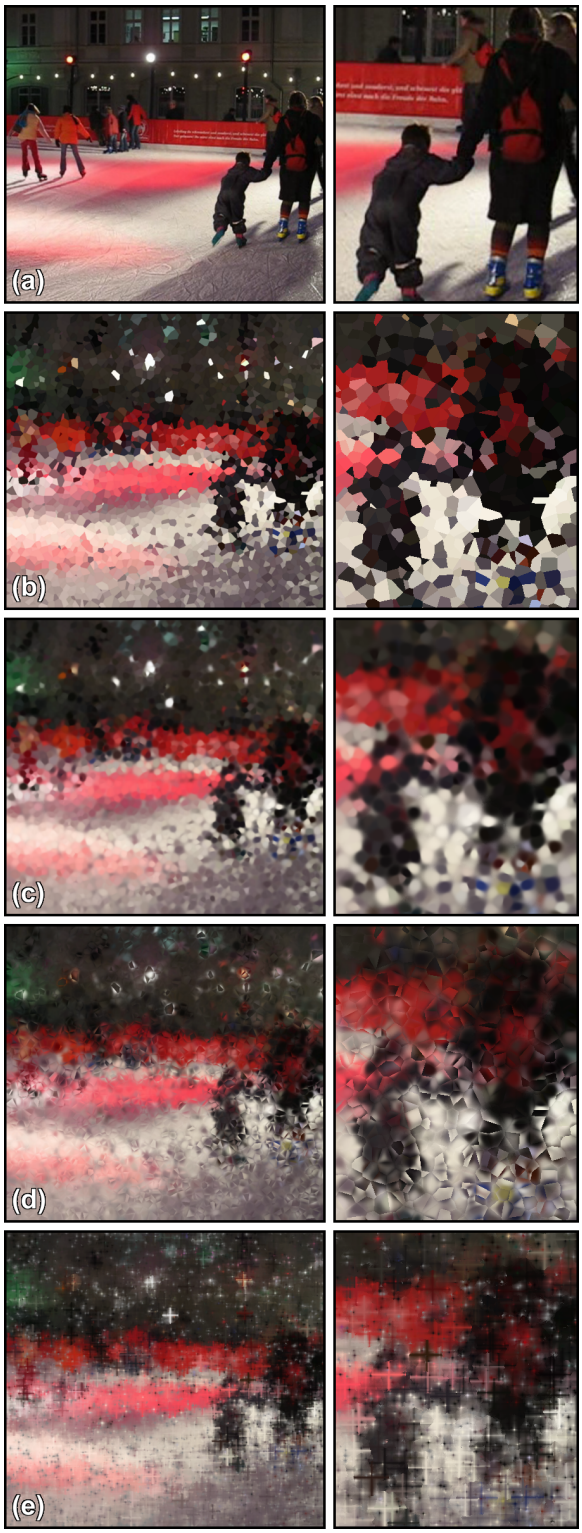


Figure 6: Voronoi rendering styles (the right image is an enlarged region from the left image): *(a)* input frame from "ice" sequence, *(b)* flat shading using 100,000 seeds (0.056%), colour sampled from sites, *(c)* "soft edges" style, *(d)* "colour facets" style, *(e)* "colour hatching" style.

and $D(x,y) = \infty$. Project all Voronoi sites onto the current frame's plane, rounding to the nearest pixel, updating V and D accordingly, only keeping the closest one for every pixel.

Downward scan. For all rows, from top to bottom, a pixel in a scan-line is updated if its upper neighbour's site is closer than its current site, setting $V(x,y) = V(x,y-1)$ and $D(x,y) = d((x,y,t), V(x,y))$, where t is the time of the current 2D slice. The same is repeated for the left and the right neighbours, all in separate passes through the scan-line.

Upward scan. For all rows, from bottom to top, pixels are updated using their lower, left and right neighbours in turn.

After the second pass, the Voronoi map will contain a slice of the 3D Voronoi diagram. This algorithm can be implemented particularly efficiently when using Euclidean distances, as the temporal component of the distance can be pre-computed when projecting the sites, and reused later on.

3.6. Rendering

We now consider how to use the sampled colour data for rendering stylised video frames. We developed extensions to the two image-based rendering approaches by Grundland et al. [GGD08], to make them work for video. The full implementation details can be found in Grundland [Gru07].

3.6.1. Voronoi Styles

Grundland et al.'s "procedural" rendering styles use the neighbour relationships encoded in the Voronoi diagram to obtain a stylised reconstruction of the sampled colour data. They determine the colour of an output pixel by a weighted average of its site's K nearest Voronoi sites. However, the 2D cross-section of a 3D Voronoi diagram is not necessarily a 2D Voronoi diagram. We therefore compute the K nearest Voronoi sites on a per-pixel basis.

If a pixel's K nearest neighbouring sites are $\mathbf{s_1}$ to $\mathbf{s_K}$, ordered by distance from $\mathbf{p}$, then its colour $f(\mathbf{p})$ is given by

$$f(\mathbf{p}) = \frac{\sum_{k=1}^{K} \phi(\mathbf{p},\mathbf{s_k}) \cdot f(\mathbf{s_k})}{\sum_{k=1}^{K} \phi(\mathbf{p},\mathbf{s_k})}, \qquad (2)$$

where a particular style is encapsulated by the local filter functions $\phi(\mathbf{p},\mathbf{s_k})$. For example, a style with solidly coloured Voronoi cells (figure 6b) can be obtained with $\phi(\mathbf{p},\mathbf{s_k}) = 1$ if $k = 1$ and $\phi(\mathbf{p},\mathbf{s_k}) = 0$ otherwise, i.e. all pixels are coloured using the closest site's colour.

A smoother rendition is achieved with inverse distance weighting. Using $\phi(\mathbf{p},\mathbf{s_k}) = d_E(\mathbf{p},\mathbf{s_k})^{-\alpha}$ with $\alpha = 6$ creates Voronoi cells with soft edges (figure 6c). By setting the nearest site's contribution to zero, $\phi(\mathbf{p},\mathbf{s_1}) = 0$, a style resembling coloured facets is obtained (figure 6d). A rendering style reminiscent of colour hatching (as in figure 6e) can be created using the filter functions

$$\phi(\mathbf{p},\mathbf{s_k}) = \left|(1+\varepsilon)\, d_M(\mathbf{p},\mathbf{s_k}) - d_E(\mathbf{p},\mathbf{s_k})\right|^{-\alpha}. \qquad (3)$$

In our example, we use $\alpha = 2$ and $\varepsilon = 10^{-13}$.

3.6.2. Delaunay-like Styles

Grundland et al.'s "geometric" rendering styles are based on the Delaunay triangulation defined by the Voronoi diagram. The obvious extension to 3D is the Delaunay tetrahedralisation. However, this is hard to compute if the 3D Voronoi diagram is never explicitly computed, as in our case.

Instead, we derive a 2D Delaunay-like triangulation from the current frame's cross-section of the 3D Voronoi diagram. We position the triangle vertices at the centroids of their respective Voronoi cells in 2D. As a side-effect, the triangle vertices also smoothly move over time, reflecting the motion of the underlying Voronoi cells. Because of this change, the obtained triangulation may not be a proper Delaunay triangulation, which is why we call it "Delaunay-like".

The Delaunay vertices should, in principle, be located at the Voronoi sites, and hence use the same colour samples. A particular rendering style governs how to fill in the pixels within a triangle from its vertex colours.

Simple rendering styles can be derived from interpolation techniques. For example, taking the average colour of the vertices produces a flat-shaded look (figure 7b). Using Gouraud shading for interpolating colours improves on that (figure 7c). Grundland et al. described the "Patchwork" style which uses non-linear interpolation along triangle edges and scan-lines, for a more artistic look (figure 7d).

The "Mosaic" rendering style (figure 7e) is based on a geometric subdivision of triangles by joining the midpoints of their three edges. This forms four smaller triangles, of which the the inner one is kept black, and the outer three are coloured with their respective vertex colours.

4. Results

We implemented our video stylisation framework using C++ for Avisynth [Avi] on an Intel 2.4 GHz quad-core CPU with 2 GB RAM. We tested the performance of our implementation using the average 2D cell colours on the "ice" video (320×240, 360 frames). Our implementation achieved a real-time frame-rate of 54 fps for the flat-shaded Voronoi style, 0.12 fps for all other Voronoi styles and the Mosaic style, and 3.6 fps for the remaining Delaunay-based styles.

We build on work by Klein et al. who first used 3D Voronoi diagrams for video stylisation [KSC*01]. Using our framework, we are able to produce similar results, as shown in figure 8. However, we can also go beyond their simple styles by extending work by Grundland et al. to videos, as illustrated in figures 6 and 7 as well as our sample videos at www.cl.cam.ac.uk/research/rainbow/projects/voronoivideo/. These styles are necessarily similar to those by Grundland et al. [GGD08], as they provide the foundation for our video framework. The extension to video, however, introduces a need for inter-frame consistency and a range of undesirable artefacts that we address.

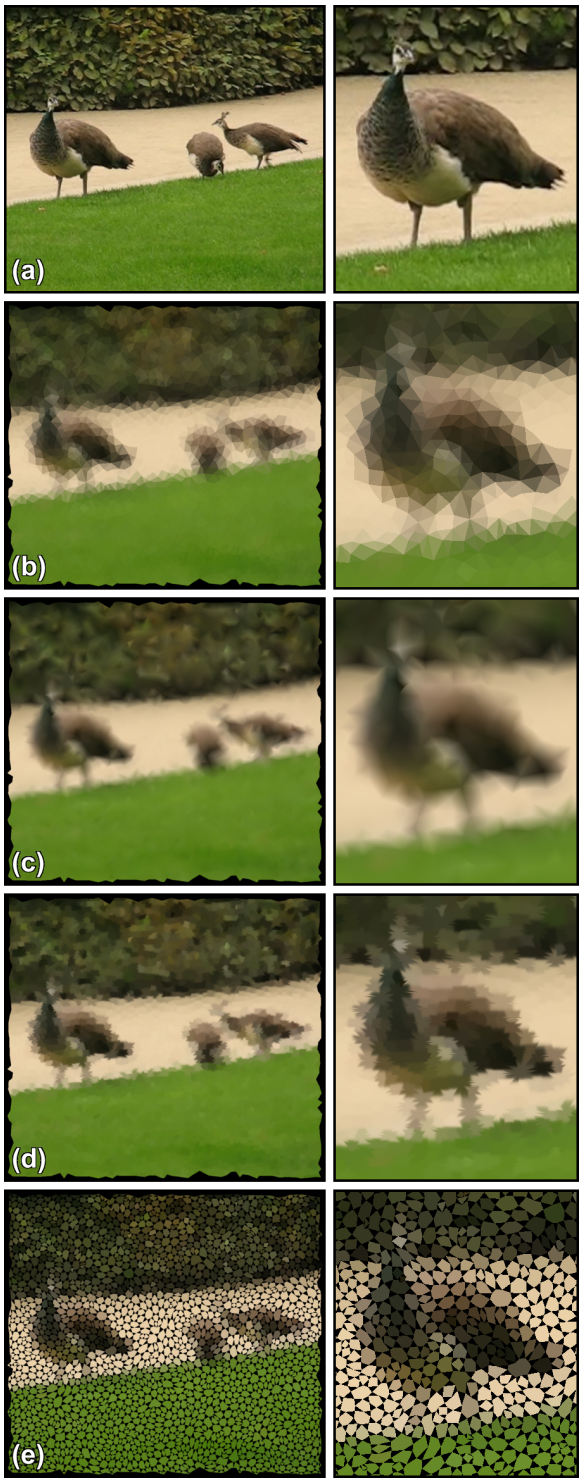


Figure 7: Delaunay-like rendering styles (full frame left, enlarged region right): *(a)* input frame from "peacocks" sequence, *(b)* flat shading using 20,000 seeds (0.033%) and mean average 2D cell colour, *(c)* Gouraud shading style, *(d)* "Patchwork" style, *(e)* "Mosaic" style.

Figure 8: Comparison of Klein et al.'s stylisation results (top row, images courtesy of Allison Klein [KSC*01]) and similar results obtained using our framework (bottom row).

4.1. Discussion

The 3D Voronoi diagram, our core video representation, has some interesting properties. It is essentially resolution-independent, as the stylised video can be rendered for a higher resolution than the original video, even though this does not add additional details. It is also sparse compared to the underlying video, which makes it amenable to non-photorealistic compression. The entire Voronoi diagram can be stored using only $(20 + 3 \cdot \text{sites})$ bytes: 4 `ints` for the video dimensions and the random seed, 3 bytes for every colour. The site coverage, i.e. the number of sites per pixel, roughly corresponds to the compression ratio.

The Voronoi cells also have non-zero spatiotemporal extents. This has advantages, but also disadvantages. On the upside, the 2D cross-sections of the Voronoi cells smoothly grow and shrink over time which improves the coherence of the stylised video. The downside is aliasing caused by the extent of cells which leads to "motion blur" artefacts.

4.2. Future Work

In our implementation, we only considered content-agnostic point sampling techniques. However, the location of Voronoi sites should take the video content into account, to encode it more accurately. This could for example be done by over-segmenting the video volume and placing Voronoi sites at the centroid of segments. Grundland's adaptive sampling techniques [GGD08] may also be extensible to 3D for video.

Our current framework is based on a static set of Voronoi sites. However, this quickly becomes intractable for longer videos which require a large number of seeds, exceeding a few hundred millions. In this case, a progressive sampling and rendering approach would ease the memory bottleneck.

It might also be of interest to investigate the stylisation properties of the Delaunay tetrahedralisation derived from the 3D Voronoi diagram. This would define a pixel's colour using the four vertices of its enclosing Delaunay tetrahedron.

And lastly performance: our current implementation is single-threaded and unoptimised, but our framework greatly benefits from parallelism, since all pixels can be rendered simultaneously. A dedicated GPU implementation would hence be desirable.

4.3. Acknowledgements

We would like to thank Mark Grundland for his kind support, Allison Klein for allowing us to use her figures, and the anonymous reviewers for their valuable feedback.

References

[Aur91] AURENHAMMER F.: Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput Surv 23*, 3 (1991), 345–405.

[Avi] AVISYNTH DEVELOPERS AND CONTRIBUTERS: Avisynth 2.5.7. http://www.avisynth.org.

[Dan80] DANIELSSON P.-E.: Euclidean distance mapping. *Comp Graph Imag Proc 14*, 3 (1980), 227–248.

[GGD08] GRUNDLAND M., GIBBS C., DODGSON N. A.: Stylized multiresolution image representation. *J Electron Imaging 17*, 1 (2008), 013009:1–17.

[Gru07] GRUNDLAND M.: *Color, Style and Composition in Image Processing*. PhD thesis, University of Cambridge Computer Laboratory, 2007.

[Hae90] HAEBERLI P.: Paint by numbers: abstract image representations. In *SIGGRAPH* (1990), pp. 207–214.

[HD06] HJELLE Ø., DÆHLEN M.: *Triangulations and Applications*. Springer, 2006.

[HKL*99] HOFF III K. E., KEYSER J., LIN M., MANOCHA D., CULVER T.: Fast computation of generalized Voronoi diagrams using graphics hardware. In *SIGGRAPH* (1999), pp. 277–286.

[KSC*01] KLEIN A. W., SLOAN P.-P. J., COLBURN R. A., FINKELSTEIN A., COHEN M. F.: *Video Cubism*. Tech. Rep. MSR-TR-2001-45, Microsoft Research, 2001.

[KSFC02] KLEIN A. W., SLOAN P.-P. J., FINKELSTEIN A., COHEN M. F.: Stylized video cubes. In *Proc SCA* (2002), pp. 15–22.

[Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *SIGGRAPH* (1997), pp. 407–414.

[Vor08] VORONOI G.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. *J Reine Angew Math 134* (1908), 198–287.