Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)
K. Anjyo, P. Faloutsos (Editors)

Animosaics

Kaleigh Smith[†], Yunjun Liu[‡] and Allison Klein[§]

McGill University

Abstract

Animated mosaics are a traditional form of stop-motion animation created by arranging and rearranging small objects or tiles from frame to frame. While this animation style is uniquely compelling, the traditional process of manually placing and then moving tiles in each frame is time-consuming and labourious. Recent work has proposed algorithms for static mosaics, but generating temporally coherent mosaic animations has remained open. In addition, previous techniques for temporal coherence allow non-photorealistic primitives to layer, blend, deform, or scale, techniques that are unsuitable for mosaic animations. This paper presents a new approach to temporal coherence and applies this to build a method for creating mosaic animations. Specifically, we characterize temporal coherence as the coordinated movement of groups of primitives. We describe a system for achieving this coordinated movement to create temporally coherent geometric packings of 2D shapes over time. We also show how to create static mosaics comprised of different tile shapes using area-based centroidal Voronoi diagrams.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Mosaic imagery is as ancient as tiled Roman baths and Byzantine iconography, and its forms vary from traditional (e.g. Islamic patterns) to modern (e.g. photo mosaics). *Animated* mosaics are a traditional form of stop-motion animation created by arranging and rearranging small objects or tiles from frame to frame. For example, the Oscar-nominated short film "Bead Game" was painstakingly created by Ishu Patel who manually packed thousands of glass beads into a variety of configurations for each frame.

While this animation style is uniquely compelling, the traditional process of manually placing and then moving pieces from frame to frame is time-consuming and labourious. In addition, making small changes to correct or improve a completed sequence essentially means exactly reconstructing and re-shooting the sequence from the point of change onwards. Finally, creating animated mosaics of certain objects (e.g. wriggling fish) is not easily possible, even if the idea is aesthetically appealing.

Creating mosaic animations on the computer is an obvious solution. Sequences can easily be saved off, reloaded, and tweaked, helping the animator achieve better results in less time. However, the fundamental and most time-consuming activity, optimally packing objects to fill a desired shape, is a well-studied problem in computer science whose general solution is known to be hard. While recent work has addressed packing for static mosaics, the solutions presented do not address issues relevant to creating individual frames as part of a temporally coherent animation sequence. In addition, previous techniques for animated non-photorealistic rendering (NPR) have allowed NPR primitives to layer, blend, deform, or scale as part of achieving temporal coherence. Manipulating mosaic tiles in any of these fashions is not faithful to either the static decorative mosaic style or the stop-motion animated style. Therefore, new techniques for temporal coherence are needed.

This paper presents a new approach to temporal coherence and applies this to create mosaic animations. Specifically, we characterize temporal coherence as the coordinated movement of *groups* of primitives. We describe a system for

[†] e-mail:kaleigh@cs.mcgill.ca
[‡] e-mail:yliu69@cs.mcgill.ca
[§] e-mail:awklein@cs.mcgill.ca

achieving this coordinated movement to produce temporally coherent geometric packings of 2D shapes over time. We also show how to create static mosaics comprised of different tile shapes using area-based centroidal Voronoi diagrams. We believe our approach frees this uniquely beautiful animation style from its current physical limitations while still giving artists expressive power.

The remainder of this paper is organized as follows: Section 2 discusses related work, while Section 3 presents a discussion of temporal coherence and how the coordinated motion of NPR primitives can be used to achieve temporally coherent animations. Section 4 formalizes the problem of animated mosaic packing and presents a solution to this problem using the group motion of tiles. Then, Section 5 describes the details of our system implementation. Finally, Section 6 shows results of our approach, and in Section 7 we discuss some conclusions and opportunities for future work.

2. Related Work

This paper is naturally related to prior work both in NPR animation and in the creation of static mosaic imagery. However, we postpone the discussion of NPR animation until Section 3, which discusses issues related to temporal coherence in more depth.

Our work builds upon previous research regarding static NPR mosaics. Hausner [Hau01] takes as input a rectangular image and, using a point-based centroidal Voronoi diagram (CVD), generates a mosaic composed of rectangular or oval tiles, Figure 1(a). Hausner also discusses methods for tile orientation which were extended by Elber and Wolberg [EW03] to better emphasize contours. Jigsaw Image Mosaics (JIM) by Kim and Pellacini [KP02] introduced a general framework for static mosaic problems by defining a metric that measures the quality of a tile packing. Using this metric, Kim and Pellacini search for a low energy configuration of optionally deformable tiles (sampled from a library) to reproduce a target image that has been segmented into disjoint containers. Both the tiles and containers can be arbitrarily shaped. (See Figure 1(b); only contours are shown for comparative purposes.)

Our work, Figure 1(c), extends [Hau01, KP02] to animation by proposing a new characterization of temporal coherence applicable to mosaic animations and then demonstrating methods for achieving this temporal coherence that converge rapidly enough to support an interactive system for creating animations. Our use of a centroidal area-based Voronoi diagram (CAVD) extends Hausner's work to support multiple, arbitrarily-shaped tiles, even within the same packing, while still achieving rapid convergence to generate packings quickly. Further, this technique supports frame-to-frame *incremental* packing optimizations to reflect container changes, allowing the artist to (optionally) fine-tune each frame's packing while still achieving temporally coherent results. Like JIM, we take as input a set of arbitrarily

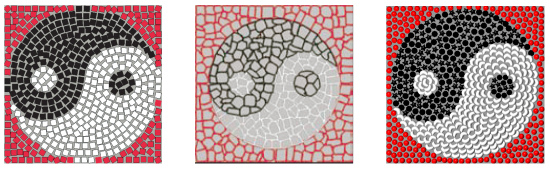

Figure 1: *A comparison of tile shapes, positions and orientations. Hausner's approach (a) packs a single tile shape well. JIM (b) achieves tight packings with irregular shapes. The CAVD approach (c) handles any variety of arbitrary shapes. (See black areas.)*

shaped containers to be packed with arbitrarily shaped tiles. However, our approach packs more quickly than JIM and with an *artist*-specified distribution of tile shapes, not an optimized subset that yields the densest packing. (JIM achieves very tight packings, but is biased against tiles that make the rest of the container difficult to pack.)

Hoff *et al.* [HKL*99, MWD97] describe the hardware-accelerated implementation of area-based Voronoi diagrams that we use. Secord [Sec02] uses weighted centroidal Voronoi diagrams to generate static stipple drawings composed of dots, while Hiller *et al.* [HHD03] uses a CAVD to distribute multiple stipple primitives (points, lines, and polygons). Hiller *et al.* proposed that CAVDs might also improve static mosaic packing algorithms. However, to our knowledge, our paper describes the first investigation of this idea. Our system for achieving temporally coherent mosaic animations can be modified to work with any packing algorithm, with potential trade-offs in tile-distribution control, speed, and per-frame fine-tuning.

Finally, as examples of other visually distinct forms of mosaiced imagery, readers are encouraged to see image and video mosaics [SH97, FR98, KGFC02] and Escherization [KS00].

3. Temporal Coherence and Group Motion

Existing techniques for achieving temporally coherent animation of NPR primitives (e.g. paint strokes or hatch marks) share a common goal of trying to minimize temporal discontinuities while having primitives appear attached to underlying scene objects. Specific examples of unwanted artifacts include frequent, noticeable appearance or disappearance of primitives ("pops"), or rapid, noisy changes in individual primitives' position, size, orientation, or colour. Common themes in creating temporally coherent NPR animations are tying the motion of NPR primitives to an underlying geometry, either directly [Mei96] or indirectly via optical flow [Lit97, HP00, KSFC02]; warping NPR imagery [LW94, HP00] in order to smoothly transition between frames; and blending in or growing in strokes, hatch marks, or other

NPR primitives [KMN*99, KLK*00, PHWF01, KDMF03]. Of specific interest, Ding [Din02] applied Hausner's static mosaic approach to animation by gradually growing in, shrinking, or merging square tiles to reduce popping. Finally, Klein *et al.* [KSFC02] also suggest smoothing attributes over a primitive's temporal lifetime.

Our observation is that even if individual primitives have temporal smoothness, uncoordinated changes among *groups* of NPR primitives will still yield distracting, incoherent animations. Imagine a rectangular arrangement of mosaic tiles. If each tile's position moves smoothly but independently of the others from frame to frame, the resulting animation suggests the uncoordinated movement of individual objects. In contrast, if all tiles move smoothly and in the same direction, this suggests a single moving rectangle. Our observation is supported by the Gestalt laws of perception [Zak97] which state that humans segment a scene by grouping individual entities according to shared qualities and behaviours.

Previous techniques for temporal coherence do not specifically target group motion or perceptual grouping of NPR primitives. Moreover, for animated mosaics, stylistic considerations preclude direct use of these previous techniques. Allowing tiles to layer, blend, deform, or scale (acceptable under previous approaches) is not faithful to either the static decorative mosaic style or the stop-motion animated style. While there may be benefits to allowing these types of operations under certain circumstances, our belief is that this should be an artistic choice, not an artifact of algorithmic limitations.

Therefore, we present a new solution for temporal coherence that respects the constraints of the animated mosaic style while achieving group motion of tiles. Our system achieves the desired perceptual grouping of primitives through cohesive motion that maintains tile orientation and spacing, promotes group tile movement, and encourages the perceptual completion of container boundaries. This new solution most closely resembles that of Kalnins *et al.* [KDMF03]. Their paper outlines an important tradeoff: tying strokes directly to 3D geometry prevents swimming along silhouettes and creases, but does yield uniform spacing and size in screen space. Conversely, a uniform parameterization in screen space does not yield the desired coherence along 1D silhouettes. Our work also seeks to maintain strokes (in this case tiles) of constant size and density, but on 2D container shapes, as discussed in the next section.

4. Construction of a Mosaic Animation

Our goal is to create an animated mosaic – a temporally coherent sequence of mosaic images over time. Specifically, given a container C (a closed polygon over time) and a collection of tile shapes, our system should choose a set of tiles T (called a *packing* of C), while addressing the following three challenges:

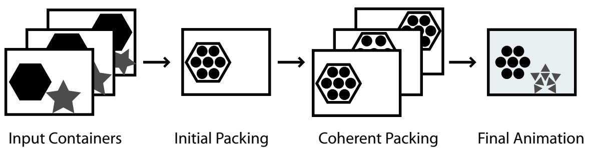

Figure 2: *Process overview: after each container is coherently packed over a sequence of frames, all packings are composited together for the final animation.*

- Temporal coherence: tiles should move smoothly over time, appear attached to their underlying object, and tile appearances or disappearances ("pops") should be minimized.
- Stylistic coherence (i.e. per-frame mosaic quality): at any time, C's packing should be composed of tiles that are evenly distributed, tightly packed with minimal overlaps, and whose orientations reflect the edges of the container shape [Hau01, KP02].
- Performance: in order to support input from the animator, the first two properties should be achieved as interactively as possible.

Note the conflict between all three goals. Independently packing each frame will lead to high per-frame mosaic quality, but at the cost of distracting temporal artifacts such as tile popping, jostling, or jitter. Conversely, very smooth, coherent tile movements may not yield pleasing individual mosaics. Additionally, performance requirements limit the amount of time that can be spent optimizing either for packing quality or temporal coherence.

We present a solution that resolves these conflicts. At a high level, our system proceeds in the steps shown in Figure 2. We take as input an animated scene represented as a collection of 2D containers (i.e. polygons). For a given container, the animator picks the desired tile shapes and sizes, and then packs the container's first frame using our system. Next, we generate the remaining frames of the container's packing in a sequential, alternating two-step process: first, our system automatically advects the container's tiles from the current frame to the next in a manner that promotes temporal coherence. Then, in the new frame, the animator optionally inserts new tiles and refines the current packing to reflect container changes. After all containers are packed through the entire sequence, the final frames can be rendered with either 2D polygonal tiles or 3D tiles using commercial modeling and rendering software.

We now explain packing and tile movement in more detail.

4.1. Mosaic Packing

The packing procedure begins with the random seeding of a user-specified set of tile shapes into the first frame of the container. Tiles are then appropriately oriented and repositioned into an even distribution over this container area.

Tile Orientation

In a static mosaic, the orientation field causes tiles to reinforce features and edges and create pleasing patterns. For mosaic animations, similar container shapes should lead to similar orientation fields, and the orientation field must be robust to the small container shape changes introduced as the container deforms over time.

JIM [KP02] preserves container edges by fitting tiles against boundaries and previously placed tiles. The best orientation for any tile is the one resulting in the tightest fit, which visually reinforces container edges but does not preserve an internal orientation field. Hausner [Hau01] aligns tiles to a continuous orientation field based on feature lines, while Elber and Wolberg [EW03] align tiles along concentric contour lines to emphasize container shape. Finally, a recent method for generating stipple drawings aligns primitives according to a variety of different orientation fields, including a field generated from image feature lines [HHD03].

We preserve the container shape boundary by aligning each tile with its closest container edge. This approach strongly reinforces the container boundary properties, maintaining both sharp changes in orientation (i.e. the corner tiles in Figure 3(b)) and the appearance of continuous orientation change (i.e. the white jellybean tiles in Figure 1(c)). In addition, to enable tighter packings and reduce image regularity, each tile shape may have a set of *equivalent* orientations, i.e. specific tile rotations which are considered to be equally valid alignments with the orientation field. (See Figure 3.) Each tile's equivalent orientation stays fixed over the tile's lifetime.

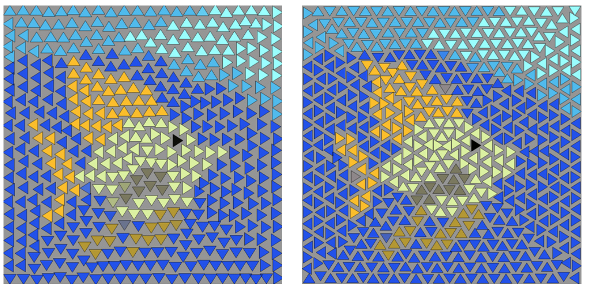
(a) Single Orientation Tiles (b) Equivalent Orientation Tiles

Figure 3: *Both mosaics show triangles aligned to the orientation field created by the image boundaries. However, in (a) only a single alignment with the orientation field is valid. In (b), the artist has specified two valid tile alignments, thereby enabling the tiles to pack more tightly while still showing alignment to boundaries.*

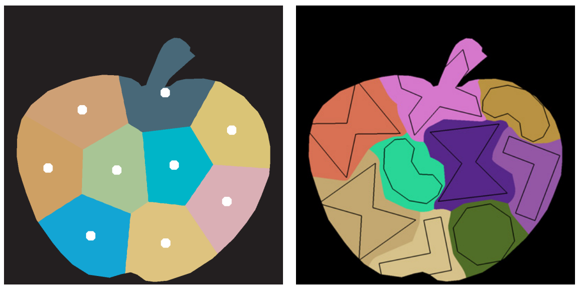
(a) Standard Voronoi Diagram (b) Area Voronoi Diagram

Figure 4: *A comparison of the standard (point-based) Voronoi diagram(a) versus an area Voronoi diagram(b). The area Voronoi diagram causes Voronoi tiles to more closely resemble generating tile shapes, particularly with concave tiles, leading to better packings.*

Tile Repositioning

After random seeding and initial orientation, we reposition the tiles into an evenly distributed arrangement characteristic of a mosaic. Our method for repositioning must be fast since repositioning is performed as part of our interactive creation process. In addition, we wish to use an artist-specified collection of arbitrary tile shapes to pack a container. Finally, in order to support fine-tuning as the container shape deforms (see Section 4.2), the packing solution we use must be amenable to small scale, frame-to-frame changes. No single previous mosaic work [Hau01, KP02, EW03] satisfies all of these constraints.

Our system employs a generalization of the centroidal Voronoi diagram approach as suggested by Hiller et al. [HHD03]. Specifically, the diagram is a constrained *centroidal area Voronoi diagram* (CAVD) constructed from a set of 2D generating *shapes* instead of generating *points*. The Voronoi region of each generating shape contains all points in space closer to that generating shape than any other under the Euclidean distance metric, and each generating shape is mass-centered within its Voronoi region. The entire diagram is constrained by the boundary of the container shape. The result of the constrained CAVD approach is that tile shapes are visually evenly distributed within the container shape as shown in Figure 4, generated with our system.

We incorporate the orientation field into the repositioning method to pack the set of tile shapes within the container shape. (The orientation field within a container is generated using the area Voronoi diagram of the container shape edges.) The method proceeds in the following steps: Given a set of k tiles placed in the container shape and oriented according to the container shape's orientation field, the discretized area Voronoi diagram (AVD) constrained to the container shape is constructed using Hoff's implementation [HKL*99]. Lloyd's algorithm [Hau01, HHD03] is then applied to construct a CAVD by iteratively translating each shape to the center of its Voronoi region, reorienting the shape according to the orientation field and then recalculating the AVD. The convergence of Lloyd's method to a stable CAVD is not proven for shapes in 2D, however in agreement with Hiller *et al.* [HHD03], our experiences show that Lloyd's converges in most cases, even when shapes re-orient according to our orientation field. During our entire project, Lloyd's failed to converge in fewer than 10 cases, all of which involved highly asymmetric, concave and unbalanced tile shapes. However, even in these cases, Lloyd's still created sufficiently even packings for our needs.

Thus the CAVD enables us to quickly pack a variety of tile shapes within a single container and provides a direct, natural connection between a tile's shape and the resulting Voronoi region. Each tile affects its local neighbourhood of tiles to result in a realistic packing and neighbouring tiles are generally oriented in similar or complementary directions. Figure 7 illustrates packings that result from our method.

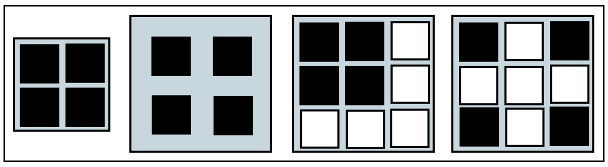
(a) Initial Packing (b) Continuous (c) Anchor-Point (d) Nearest-Edge

Figure 5: *Different flow field inference methods for tile advection illustrated with a uniform container scale. (a) The initial container packing. (b) shows the effects of continuous interpolation. Even with the increased grout, there is no place to insert new tiles. In contrast, anchor point mapping, (c) creates space for new tiles (white) around the boundary of existing tiles. (The container's upper left corner was chosen as the anchor point.) Nearest-edge mapping, (d), also concentrates new space, but at the container center.*

4.2. Temporally Coherent Tile Advection

At this point, given a container packing in one frame, tiles must be advected to the next frame in order to create our animated sequence. Because moving tiles as a coherent group is easy in the cases of container translation and rotation, our system decomposes a container C's transformation from time t to $t+1$ into translation, rotation, and deformation steps, and we concentrate our discussion on handling deformations.

In our system, a tile centered at point (x,y) is mapped to a new point $(x+\Delta x, y+\Delta y)$ using a flow field inferred from the container's behaviour. The inference of this flow field is a data interpolation problem since C's behaviour from time t to $t+1$ specifies the flow field at C's boundary, but not at the interior.

Data interpolation using weighted contributions from all container vertices is an obvious solution, but it results in a flow field unsuited to our application. Consider a uniformly expanding container: from one frame to the next, a continuously interpolating flow field would cause tiles to move away from each other, preventing group movement and increasing grout space between *all* tiles. This increase in grout space both decreases the mosaic quality and makes it impossible to add new tiles without displacing current ones until grout patches afford enough space. (See Figure 5(b).) At this point, *many* new tiles would pop in at multiple locations dispersed through the container, a distracting artifact. (Ding [Din02] addressed this problem by allowing tiles to shrink and grow.)

From our example, we can make another important observation regarding coherence: popping in a single tile is less noticeable than popping in multiple tiles, especially at multiple locations. This observation, like that regarding group motion in Section 3, is consistent with the Gestalt laws of perception. Similar location and close proximity of inserted tiles encourages perceptual grouping so that the many insertions are perceived as one single insertion. Therefore, in addition to group movement of tiles, a goal for our system is group insertion of tiles. With these motivations, we propose two approaches for handling tile movement during container deformations.

Our first technique is *anchor-point* mapping. The existing arrangement of tiles is relocated based on the movement of a single point, either a container vertex or the container centroid. This relocation is equivalent to a group translation and/or rotation, resulting in cohesive, smooth movement for all existing tiles. Anchor-point mapping will generally cause insertions or deletions to happen only at the boundary of the existing body of tiles because the relative spacing of the tiles is fixed and thus cannot cause internal gaps or overlaps to develop. (See Figure 5(c).) Therefore, anchor-point mapping is useful when preserving a packing's interior organization from frame to frame is of primary importance to the animator. Our system automatically chooses the anchor point to be the container vertex or container centroid with the smallest displacement between the two frames, however the animator may choose any specific anchor point if desired.

Our second technique is *nearest-edge* mapping, a stylistic alternative when an animator wishes to preserve the packing boundary instead of the interior. For each tile, we note its x and y offsets (in pixels) from the closest container edge at time t, and map it to the same offsets at time $t+1$. For our application, nearest-edge mapping (Figure 5(d)) sacrifices flow field continuity for improved temporal coherence in two significant ways. First, tiles along container edges move together, promoting group movement along features where we are likely to look. Second, observe that as a container's perimeter increases linearly, the internal area increases quadratically. Nearest edge mapping ensures that interior tiles move away from the center as a container in-

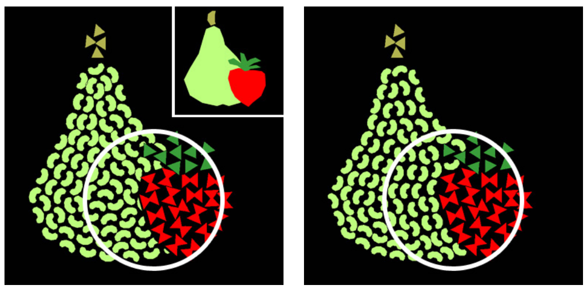
(a) compositing after packing (b) compositing before packing

Figure 6: *Container compositing. (The original containers are inset in (a).) Circled areas show how compositing before packing (b) allows background tiles to align themselves with the border of the foreground container.*

creases in size. This causes new space to accrete in the center, enabling new tile insertion without rearranging existing tiles. Since absolute positioning is used, during container contractions neighbouring tiles will overlap (and thus be removed) at the center of the container. In extreme container contractions, the tiles furthest from the edges (i.e. at the center) are mapped outside of the container and therefore deleted.

The two techniques described above deliver coherent group movement and minimize the distraction of tile popping by spatially concentrating tile insertions and deletion. Using these techniques, we advect existing tiles from one frame to the next according to the container's flow field. At the next frame, we render the tile shapes and use pixel-space algorithms to check if tiles should be inserted or removed due to gaps or overlaps. New tiles are inserted in areas of open space, oriented, and then moved according to a CAVD, keeping all other tiles fixed. A useful property of the CAVD algorithm is that when a packing is close to a stable point, the tile movement between iterations is small. After positioning new tiles, we exploit this property in an optional fine-tuning step to adjust tile spacing by running a small number (generally 10 or fewer) of CAVD iterations on all tiles within a container. The speed of the CAVD iterations along with this fine-tuning property make the CAVD well-suited to our application.

5. System Details

Container Specification

In our system, scene containers are specified as scalable vector graphics (SVG) animations. SVG is a human-readable format for describing 2D graphics in XML and enables easy specification of vector graphic shapes and animations. (See http://www.w3.org/TR/SVG/ for more information.) Using a graphical SVG editor (we chose Corel WebDraw), an artist draws and animates a scene. Our system then reads in the SVG source, translating scene elements into coloured container shapes (i.e. collections of ordered points). Our SVG parser also automatically extracts each container's transformation matrices (for affine transformations), point-to-point perimeter correspondences (for arbitrary container deformations), and the start and end times for each animated transformation. We generally sample the SVG animations at 8 to 15 frames per second, interpolating the container values at each of these sample points.

Using SVG to specify the animation allows the artist to easily create animated scenes with readily available GUI tools, while providing our system with the necessary information regarding container behaviour (i.e. container transformations and perimeter point correspondences). However, our system can use any kind of container and flow field such as video with point correspondences around segmented regions [AHSS04, WXSC04] or video with optical flow. In the video accompanying this paper, we show a sequence created using containers from segmented video with point-to-point correspondences manually established around segment borders.

A remaining issue is handling container overlaps. Our default mode is to composite overlapping containers before packing. This changes the shape of the background container, thereby enabling tiles in the packed background container to align themselves with the border of an overlapping foreground container (Figure 6(b)). As an alternative, compositing can be performed after packing. In this case, containers are rendered back to front, either treating packed container being treated as a single, solid object with a grout background (Figure 6(a)), or letting tiles peek through (the sun in Figure 7(c)).

Packing

Given the temporally sampled containers, the artist independently packs each as follows: First, the artist chooses the set of shapes to be used for packing and the number of desired tiles. Based on these factors, the system calculates the tile sizes for each shape to randomly seed in the container to achieve maximal coverage. Each tile is assigned its container's colour. Alternatively, point- or area-sampling a reference colour image can also be used, as in Figure 7(b). As mentioned before, we use Hoff *et al.*'s [HKL*99] hardware-accelerated implementation of area-based Voronoi diagrams, and Lloyd's algorithm [Hau01, HHD03] for the CAVD. Like Hausner [Hau01], the animator iterates until satisfied with the packing. The tiles are then advected forward to the next frame based on the flow field as discussed in Section 4. Tiles that are advected outside of a container are automatically removed. In some cases, the flow field may cause tiles to overlap after advection; we also use a brute-force N^2 algorithm to automatically check for such overlapping tiles and remove them if the overlap exceeds a user specified maximum overlap threshold. The animator can specify that tiles be added

Sequence	Number of Containers	Number of Frames	Time per Second of Output (mins)	Avg. Time per Container-Frame (secs)
YinYang	9	99	9.6	48
Water Pour	5	48	11.25	45
Hula	12	60	3.3	20

Table 1: *Sample timing statistics for our sequences.*

to the container manually or automatically. In the latter case we randomly probe for areas large enough to insert new tiles. In both cases these newly inserted tiles are then re-oriented and packed while keeping the placement of the remaining tiles static. Afterward a few final, global CAVD iterations can be applied as described in Section 4. After packing the sequence, the artist can also add noise to tile positions and orientations for a more hand-crafted look.

6. Results

In this section we discuss our results (Figure 7 and accompanying video). Our algorithm is fast enough to enable interactive design sessions on a 3.2GHz Pentium 4 PC with 1GB of RAM and an ATI Radeon 9800Pro graphics card with 128MB of video memory. Table 1 shows packing times of sample sequences. Note that the Hula, Water Pour, and YinYang sequences were sampled at 10, 15, and 12 frames per second respectively. Usually, packing the first frame is the most time consuming, and because our tile advection method causes tiles to remain relatively closely packed from frame to frame, packing successive frames is faster. Our most complex sequence, the yinyang animation, took approximately 48 seconds per container-frame on average, *primarily user interaction*. The severe container deformations required the most user interaction (i.e. experimenting with anchor-point versus nearest-edge mapping or testing out automatic versus manual tile insertions). However, the user never waits more than a few seconds for any single command (i.e. click), and can easily try various choices, undo, and try something different. Therefore, we believe our system enables artists to craft animations effectively, interactively, and with satisfying results.

Regarding visual quality, static images such as the goldfish and the birthday cake show results in which tiles are evenly distributed, appropriately oriented, and tightly packed with minimal overlaps, even when using multiple tile shapes within a single container (Figure 7(a)). Furthermore, our yinyang sequence demonstrates that these attractive mosaic qualities hold true even during animated sequences with severe container deformations. The pitcher-and-glass sequence shows that our system can achieve natural, compelling effects, such as pouring water. Also, notice how container compositing allows the water to cut in front of the back rim of the glass while being occluded by the front rim. The bullfrog animation uses the addition of noise to suggest a rippling pond surface. The figure descending a staircase shows how rotoscoping can also be used to create containers, leading to smooth, lifelike animations.

Our hula sequence, Figure 7(c and d), is our most complex. A vertical orientation field was used for the dancer's clothing to better suggest a grass skirt (and then mermaid scales), while orientation fields based on container edges were used everywhere else. Various sized stars were used to pack the sun/moon for a handicraft style. Different container compositing effects were chosen by the artist in different scene locations; tiles from the sun are allowed to peek through the tiles in the palm fronds, whereas container compositing before packing is used when the mermaid's tail blocks the island or the front tree trunk blocks the back trunk. After packing, the dancing portion of the sequence was repeated twice to lengthen the animation. Finally, we composited the packed containers with hand-drawn 2D elements for a mixed-media effect. The final result is a jaunty, dynamic animation.

7. Conclusion and Future Work

We have presented a novel system for animated mosaics that meets the three challenges of temporal coherence, per-frame mosaic quality, and performance that supports interactive input from the artist. We applied new approaches to temporal coherence through the coordinated movement of groups of NPR primitives to create temporally coherent geometric packings of 2D shapes over time. Additionally, we demonstrated how to create mosaics comprised of different tile shapes using area-based centroidal Voronoi diagrams.

Regarding improvements for the current system, automatically detecting tile overlaps and areas for tile insertions could be made faster. In addition, because each tile is given one of the equivalent orientations randomly (i.e. independent of neighbouring tiles), there are still sub-optimal areas. This could be corrected either manually or by having an algorithm that picks a tile's equivalent orientation based on neighbouring tiles.

Perhaps more interestingly, the novel techniques for temporal coherence presented here have potential relevance to other applications, such as reducing the need for warping and blending in other NPR animation styles. In addition, we believe the packings can be extended in many interesting ways, including the use of animated or deformable tiles (e.g. packing a container with wriggling fish over time or packing a 2D shape with flexible snakes), and packing 3D volumes with 3D objects, all based on extensions to the current CAVD approach. We also feel the CAVD approach might be used to improve simulations of packed formations and physically-based transitions between these states. Such simulations could be applied crowd or flock movements as well as pouring and flowing discrete objects.

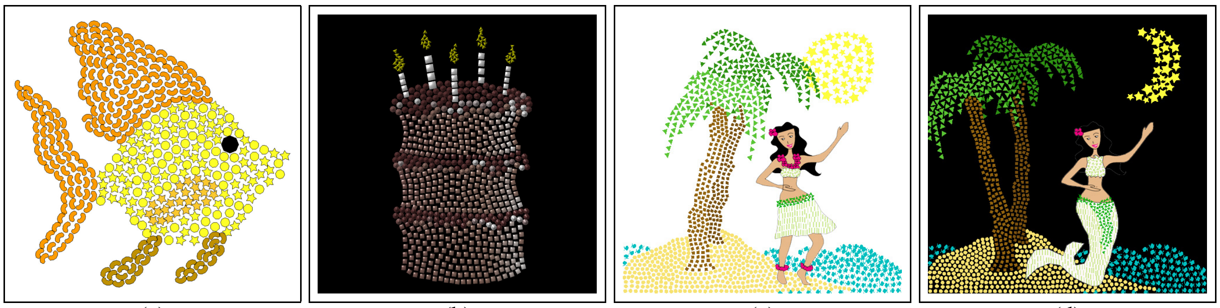
(a) (b) (c) (d)

Figure 7: *Results from our system. Images (a) and (b) are static packings, while (c) and (d) are from an animated sequence. Observe that packings remain evenly spaced and dense, even with multiple tile shapes (a) or under container deformations (c and d).*

Acknowledgements

This research was made possible by funding from NSERC and FQRNT, and through generous donations from ATI and Alias corporations. The authors wish to thank: Alejo Hausner and Junwan Kim for their source code; Jue Wang and Michael Cohen for the segmented video sequence; and Louisa Sage for the lighthouse animation.

References

[AHSS04] AGARWALA A., HERTZMANN A., SALESIN D. H., SEITZ S. M.: Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics 23*, 3 (2004), 584–591. 6

[Din02] DING Z.: *Computer Generated Mosaic Animation*. Master's thesis, University of Toronto, 2002. 2, 5

[EW03] ELBER G., WOLBERG G.: Rendering traditional mosaics. *The Visual Computer 19*, 1 (2003), 67–78. 2, 4

[FR98] FINKELSTEIN A., RANGE M.: Image mosaics. *Lecture Notes in Computer Science 1375* (1998). 2

[Hau01] HAUSNER A.: Simulating decorative mosaics. In *Proceedings of ACM SIGGRAPH* (2001), pp. 573–580. 2, 3, 4, 6

[HHD03] HILLER S., HELLWIG H., DEUSSEN O.: Beyond stippling - methods for distributing objects on the plane. *Computer Graphics Forum 22*, 3 (2003), 515–522. 2, 4, 6

[HKL*99] HOFF III K. E., KEYSER J., LIN M., MANOCHA D., CULVER T.: Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of ACM SIGGRAPH* (1999), pp. 277–286. 2, 4, 6

[HP00] HERTZMANN A., PERLIN K.: Painterly rendering for video and interaction. In *Non-Photorealistic Animation and Rendering* (Annecy, France, 2000), pp. 7–12. 2

[KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. *ACM Transactions on Graphics 22*, 3 (2003), 856–861. 2, 3

[KGFC02] KLEIN A. W., GRANT T., FINKELSTEIN A., COHEN M. F.: Video mosaics. In *Non-Photorealistic Animation and Rendering* (2002), pp. 21–28. 2

[KLK*00] KLEIN A. W., LI W. W., KAZHDAN M. M., CORREA W. T., FINKELSTEIN A., FUNKHOUSER T. A.: Non-photorealistic virtual environments. In *Proceedings of ACM SIGGRAPH* (July 2000), pp. 527–534. 2

[KMN*99] KOWALSKI M. A., MARKOSIAN L., NORTHRUP J. D., BOURDEV L., BARZEL R., HOLDEN L. S., HUGHES J.: Art-based rendering of fur, grass, and trees. In *Proceedings of ACM SIGGRAPH 1999* (1999), pp. 433–438. 2

[KP02] KIM J., PELLACINI F.: Jigsaw image mosaics. In *Proceedings of ACM SIGGRAPH* (2002), pp. 657–664. 2, 3, 4

[KS00] KAPLAN C. S., SALESIN D. H.: Escherization. In *Proceedings of ACM SIGGRAPH* (2000), pp. 499–510. 2

[KSFC02] KLEIN A. W., SLOAN P.-P. J., FINKELSTEIN A., COHEN M. F.: Stylized video cubes. In *ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), ACM Press, pp. 15–22. 2, 3

[Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *Proceedings of ACM SIGGRAPH* (1997), pp. 407–414. 2

[LW94] LITWINOWICZ P., WILLIAMS L.: Animating images with drawings. In *Proceedings of ACM SIGGRAPH* (1994), pp. 409–412. 2

[Mei96] MEIER B. J.: Painterly rendering for animation. In *Proceedings of ACM SIGGRAPH* (1996), pp. 477–484. 2

[MWD97] MASON WOO J. N., DAVIS T.: *OpenGL Programming Guide (2nd Edition)*. Addison-Wesley, 1997. 2

[PHWF01] PRAUN E., HOPPE H., WEBB M., FINKELSTEIN A.: Real-time hatching. In *Proceedings of ACM SIGGRAPH* (2001), pp. 579–584. 2

[Sec02] SECORD A.: Weighted voronoi stippling. In *Non-Photorealistic Animation and Rendering* (2002), pp. 37–43. 2

[SH97] SILVERS R., HAWLEY M.: *Photomosaics*. Henry Holt and Co., Inc., 1997. 2

[WXSC04] WANG J., XU Y., SHUM H.-Y., COHEN M. F.: Video tooning. *ACM Transactions on Graphics 23*, 3 (2004), 574–583. 6

[Zak97] ZAKIA R. D.: *Perception and imaging*. Focal Press, Boston, MA, 1997. 3