Animated Classic Mosaics from Video

Yu Liu and Olga Veksler

Department of Computer Science, University of Western Ontario
London, Ontario, Canada, N6A 5B7
yliu382@csd.uwo.ca, olga@csd.uwo.ca

Abstract. Generating artificial classic mosaics from digital images is an area of NPR rendering that has recently seen several successful approaches. A sequence of mosaic images creates a unique and compelling animation style, however, there has been little work in this area. We address the problem of creating animated mosaics directly from real video sequences. As with any animation, the main challenge is to enforce temporal coherency between the frames. For this purpose, we develop a new motion segmentation algorithm. Our algorithm requires only a minimal help from the user. We pack the tiles into the discovered coherent motion layers, using color information in all the frames in a global manner. Occlusions and dis-occlusions are handled gracefully. We produce colorful, temporally coherent and uniquely appealing mosaic animations. We believe that our method is the first one to animate classic mosaics directly from video.

1 Introduction

Mosaics are composed of a large number of regularly shaped tiles, such as rectangles and squares, artfully arranged. Simulating classic mosaics from digital images is one of the areas in non-photorealistic rendering that has been widely investigated [1,2,3,4,5].

One of the reasons for popularity of NPR rendering in computer graphics is that a stylized image can have a more profound impact on the user than the original. This is perhaps even more true of an NPR animation [6,7,8,9]. Creating animated mosaics manually is very labor intensive. However, there has been little work on creating mosaic animations automatically or interactively [10,4].

We develop a system for creating animated mosaics directly from video sequences. Our approach is inspired by [10], who were the first to realize the unique set of challenges for mosaic animation. In many NPR animation methods, in order to facilitate temporal coherence, the primitives are allowed to deform, scale, blend, etc. However, to stay faithful to the classic mosaic style, the tile primitives cannot undergo any such transformations. Each individual frame must be a convincing mosaic, while the whole sequence exhibits a convincing motion.

One way to achieve temporal coherency is to displace groups of tiles in a consistent manner. For this purpose we develop a new motion segmentation algorithm with occlusion reasoning. Our algorithm requires minimal help from

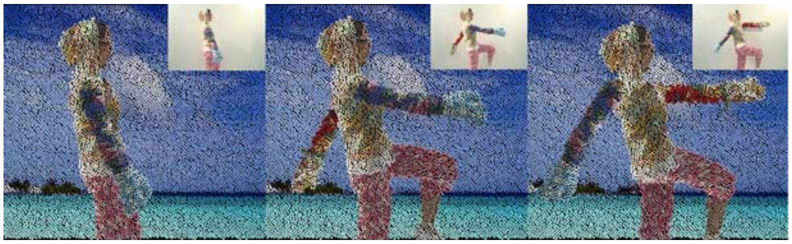


Fig. 1. Several frames from "Walking" sequence, and the corresponding mosaic

the user. We pack the tiles into the discovered coherent motion layers, using color information in all the frames in a global manner. Our tile packing algorithm is based on the one for still mosaic [5], with several modifications to address video input. Occlusions are handled gracefully. We produce colorful, temporally coherent and uniquely appealing mosaic animations, see Fig. 1. We believe that our method is the first one to animate classic mosaics directly from video.

2 Related Work

There are several approaches to still classic mosaic rendering from a digital image. In order to obtain a visually pleasing mosaic, most methods agree on the following basic principles. First, mosaic tiles should be placed at orientations that emphasize perceptually important curves in an image. This is usually done by placing the tile sides parallel to the important curves. Which curves are important is often decided through user interaction or edge detection. The second principle is to maximize the number of tiles, while avoiding overlap as much as possible. This, combined with the first principle, implies that the tile orientations should align with important boundaries and vary smoothly in the image, since smoothly varying orientations allow a tighter packing. The last principle is that the tile color distribution should reflect that of the underlying image.

There is a variety of techniques for classic mosaics [1,2,11]. All of the above have a number of heuristics steps, and their behavior may be hard to predict and control. We base our animated mosaics on the still mosaic method in [5], which is based on principled global optimization. They formulate an explicit objective function that incorporates the desired mosaic properties. User interaction and explicit edge detection are not required.

The main challenge to our animated mosaics is ensuring temporal consistency. Stylizing each frame individually produces disturbing artifacts. Artifacts may be more tolerable in the moving parts of the scene and could be regarded as a special effect. However flickering artifacts are especially pronounced in the static regions of the frame. Therefore most NPR methods seeking to stylize a video have to deal with temporal consistency.

There are roughly two ways to approach temporal consistency. The first group of methods treats a video as a space-time 3D volume [8,4,9]. Rather than directing 2D (flat) primitives in the direction of the scene motion, temporal coherency is achieved by using volumetric (3D) rendering primitives to fill the 3D space-time volume. The advantage is that motion estimation, which is a notoriously hard computer vision problem, is avoided. This approach, however, is harder to adapt to mosaic animation, since the cross sections of the 3D primitives must be valid mosaic tiles.

The second group of methods is based on explicit computation of motion, typically based on optical flow [6,7]. The idea is to let the rendering primitive (brush strokes, etc.) follow the motion field so that the primitives appear attached to the scene objects. Our work falls into this first group.

We are aware of only two methods [10,4] for mosaic animation. In [4], a moving mosaic is created by packing 3D volumes with temporally repeating animated shapes. This work is very interesting and produces appealing animations, however, it is far from our goal of rendering a real video in a classic mosaic style.

Our work is based on [10]. They make an observation that many devices for temporal coherence in NPR animation are based on the changes of primitive renderings units (i.e. scale, blend, etc.), which is not appropriate for classic mosaic animation. They argue that for classic mosaics specifically, one should target coherent motion of group of primitives. However in their work they assume that the motion is given by the user. The input to their algorithm is an animated scene represented as a collection of 2D "containers", with known correspondences between containers in adjacent frames. We extend the work of [10] to real video sequences. Thus we must estimate the "containers" and their correspondence.

3 Energy Minimization with Graph Cuts

Many problems in vision and graphics can be stated as labeling problems. In a labeling problem, one has a set of pixels $\mathcal{P}$, which is often the set of all image pixels, and a finite set of labels $\mathcal{L}$. The task is to assign some label $l \in \mathcal{L}$ to each image pixel. Let f_p denote the label assigned to pixel p, and let f be the collection of all pixel-label assignments. Typically there are two types of constraints on pixels. Unary constraints, denoted by $D_p(l)$, reflect how likely is each label $l \in \mathcal{L}$ for pixel p. The lower the value of $D_p(l)$, the more likely is label l for pixel p. Usually $D_p(l)$ are modeled from the observed data. Binary constraints, denoted by $V_{pq}(l_1, l_2)$, express how likely it is for two neighboring pixels p and q to have labels l_1 and l_2, respectively. Binary constraints usually come from prior knowledge about the optimal labeling. An energy function is formulated to measure the quality of f:

$$E(f) = E_{smooth}(f) + E_{data}(f). \tag{1}$$

$E_{data}(f)$ is called the data term, and it sums up the unary constraints: $E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p)$. E_{smooth} is called the smoothness term, and it sums up the binary constraints:

$$E_{smooth} = \sum_{\{p,q\} \in \mathcal{N}} w_{pq} \cdot V_{pq}(f_p, f_q). \tag{2}$$

In Eq. (2), $\mathcal{N}$ is a collection of neighboring pixel pairs, often the standard 4 or 8-connected grid. The choice of V_{pq} reflects a priori knowledge about the desired labeling. A frequent choice is $V_{pq}(f_p, f_q) = \min(K, |f_p - f_q|^C)$, where K, C are constants. If $K = C = 1$, the smoothness term corresponds to the famous Potts model. Many commonly used V_{pq} are NP-hard to optimize, but there are approximations based on graph cuts [12]. We use min-cut implementation of [13].

4 Review of Still Classic Mosaic

We now review the static mosaic algorithm of [5]. They design an objective function that incorporates the desired mosaic properties, such as: (i) tiles should align with strong intensity edges; (ii) nearby tiles should have similar orientations; (iii) tiles should avoid crossing strong image edges; (iv) the gap space should be minimal; (v) tiles should not overlap. User interaction and explicit edge detection are avoided.

We start with the label set. Let I be the image to generate the mosaic from, and let $\mathcal{P}$ be the collection of all pixels of I. For each $p \in \mathcal{P}$ we wish to assign a label which is an ordered pair: (v_p, φ_p). Here $v_p \in \{0, 1\}$ is the binary "visibility" variable. If $v_p = 1$, then a tile centered at p is placed in the mosaic. If $v_p = 0$ then the mosaic does not have a tile centered at p. We assume that all tiles are squares with a fixed side $tSize$.

The second part of the label, φ_p, specifies the orientation of the tile centered at p, if there is such a tile. If $v_p = 1$ then φ_p has a meaning (i.e. tile orientation), if $v_p = 0$, the value of φ_p is not used. The set of tile orientations $\varPhi$ is discretized into m angles, at equal intervals. Since tiles are rotationally symmetric, only the angles in $[0, \frac{\pi}{2})$ are needed. We set $m = 32$.

Let $\mathcal{T}(p, \varphi_p)$ denote the set of pixels covered by a tile centered at pixel p and with orientation φ_p. The color of the tile is the average color of pixels it covers.

Let $\varphi = \{\varphi_p | p \in \mathcal{P}\}$ and $v = \{v_p | p \in \mathcal{P}\}$. A mosaic then is an ordered pair of variables (v, φ) s.t. $v \in \{0, 1\}^n$ and $\varphi \in \varPhi^n$, where n is the size of $\mathcal{P}$. The energy function for a mosaic (v, φ) is formulated as:

$$E(v, \varphi) = \sum_{p \in \mathcal{P}} (1 - v_p) + \sum_{p \in \mathcal{P}} v_p \cdot D_p(\varphi_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(v_p, v_q, \varphi_p, \varphi_q). \tag{3}$$

The first sum in Eq. (3) minimizes the gap space. The second sum in Eq. (3) is the data term. Each $D_p(\varphi_p)$ measures the quality of a tile with center at p and with orientation φ_p. $D_p(\varphi_p)$ is computed from the local area around p. Under a good orientation φ_p, the sum of image gradients under the tile is small and there is a strong intensity gradient around at least one tile edge. Multiplying $D_p(\varphi_p)$ by v_p ensures that we measure the quality only of the tiles that are visible.

The last term in Eq. (3) is the smoothness term. The neighborhood system is: $\mathcal{N} = \left\{ \{p,q\} \mid dist(p,q) \le \sqrt{2} \cdot tSize \right\}$, where $dist(p,q)$ is the Euclidian distance between pixels p and q. This $\mathcal{N}$ is large enough to contain all pairs $\{p,q\}$ s.t. if we place tiles centered at p and q, they are either adjacent or overlapping. The interaction term is:

$$V_{pq}(\varphi_p, \varphi_q, v_p, v_q) = \begin{cases} 0 & \text{if } v_p = 0 \text{ or } v_q = 0 \\ w_s \cdot |\varphi_p - \varphi_q|_{\mathrm{mod}(\frac{\pi}{2})} & \text{if } v_p = v_q = 1 \\ & \text{and } \mathcal{T}(p,\varphi_p) \cap \mathcal{T}(q,\varphi_q) = \emptyset \\ \infty & \text{if } v_p = v_q = 1 \\ & \text{and } \mathcal{T}(p,\varphi_p) \cap \mathcal{T}(q,\varphi_q) \ne \emptyset \end{cases} \quad (4)$$

where

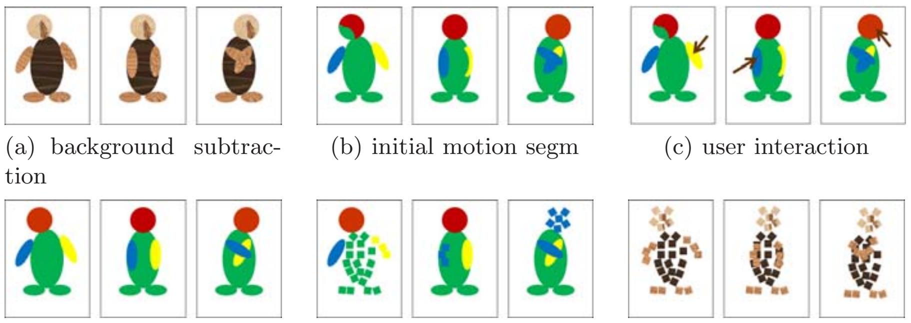
$$|\varphi_p - \varphi_q|_{\mathrm{mod}(\frac{\pi}{2})} = \begin{cases} |\varphi_p - \varphi_q| & \text{if } |\varphi_p - \varphi_q| \le \frac{\pi}{4} \\ \frac{\pi}{2} - |\varphi_p - \varphi_q| & \text{otherwise} \end{cases} . \quad (5)$$

The smoothness term serves two purposes. First, any finite energy labeling does not have overlapping tiles. Second, adjacent tiles are encouraged to have similar orientations. We only consider the orientations of neighboring tiles that are actually placed in the mosaic. The modulo arithmetic in Eq. (5) reflects the fact that rotation by angle φ_p gives the same result as rotation by angle $\varphi_p + \frac{\pi}{2}$.

The energy in Eq. (3) is too difficult to optimize in all variables simultaneously. In [5], they use an incremental approach, based on the graph cuts [12], which first optimizes the orientation and then the visibility variables.

5 Overview of Mosaic Animation

Fig. 2 gives a schematic overview. We start with a sequence of m frames, $I_1, I_2, ..., I_m$. We assume that the scene background is known and stationary. However, we are interested in background replacement, since typical office scenes



(a) background subtraction (b) initial motion segm (c) user interaction

(d) corrected motion segmentation (e) still mosaic in key segments (f) mosaic propagated

Fig. 2. Summary of the approach

have boring backgrounds that do not produce appealing mosaics. Thus the first step is background subtraction, Fig. 2(a).

To ensure temporal coherence, we find groups of pixels that have common motion. This is the task of motion segmentation. A group of pixels with coherent motion is called a *layer*.

We develop motion segmentation algorithm for the whole sequence in a global optimization framework, Fig. 2(b). Let L be a layer of pixels with common motion throughout the whole sequence. If general motions are allowed, the "containers" corresponding to L in two different frames may undergo drastic changes in scale, shear, etc. One has to come up with non-trivial strategies for filling these corresponding "containers" with tiles such that each container is a valid mosaic and the apparent motion between the frames is acceptable. In [10], they explore two such strategies with different visual effects. Unlike [10], we are already facing a formidable task of motion segmentation of a real video, so we chose leave exploring the strategies from [10] as well as developing the new ones for future work.

We assume that the motion of layer L between frame I_l and I_{l+1} is well approximated by rotation and translation. Notice that between each individual pair of frames, the translation and rotation parameters of L can be different. With this restriction, the "containers" corresponding to layer L in frames I_l and I_{l+1} have identical shape, except if there is an occlusion or out of frame motion. Therefore, we also need to include occlusion detection as a part of our motion segmentation. Under restriction to rigid layer motion, packing two corresponding "containers" between two frames then becomes basically equivalent to moving tiles from one container to another, following the computed motion, except parts of a container may become occluded by another layer.

Automatic motion segmentation rarely produces error-free results. Therefore we ask the user for corrections, Fig. 2(c).We sample and present a portion of frames to the user. If a part of an object was not segmented correctly, the user finds a nearby frame where the same part was correctly segmented, and clicks on this part. These user indicated correct segmentations are then propagated to the rest of the sequence, Fig. 2(d). During propagation, we also handle occlusions.

Finally we pack the tiles using the algorithm in Sec. 4, with some adjustments to take advantage of the full video sequence. First the tiles are placed into the "key" segments indicated by user interaction, Fig. 2(e). These segments are likely to correspond to regions with higher image quality. Next the mosaics of the key segments are propagated to the rest of the sequence, taking occlusions into consideration. Lastly mosaic is placed in any segments that have not been tiled yet, and we render the tiles with the corresponding image colors, Fig. 2(f).

6 Detailed Description

6.1 Background Subtraction

Most indoor office scenes are dull in color, resulting in unimpressive mosaic backgrounds. Thus we remove background and render the moving object in front of a lively scene, rendered as a classic mosaic using [5]. Background subtraction is a well studied area in computer vision [14]. To get accurate background subtraction, we use global optimization, similar to that of [15].

6.2 Initial Motion Segmentation

Temporal coherency of the final animation depends most of all on the accuracy of motion segmentation, which is a widely studied problem in computer vision [16]. Methods based on global optimization [17,18] produce more accurate results. Our algorithm,particularly suitable for our application, is most closely related to that of [17].

In [17], motion segmentation is performed on a pair of frames at a time. First a sparse set of feature points is matched across two frames. Then using RANSAC [19], several motion models are fitted to the matched points. Next, dense assignment of image pixels to motion layers is performed. The algorithm is further iterated, refining motion models and reassigning pixels to motion layers.

For our application, we need motion layers for the whole sequence, not a pair of frames. One solution is to track feature points throughout the whole sequence, as in [18], but the number of features that appear in all frames is limited.

Our solution is to first estimate pairwise motion models between two adjacent frames, but then find correspondences between adjacent motion models. Global motion models (i.e. those describing a motion from the first frame to the last) are formed from the correspondences. Finally global motion segmentation is performed for all frames at the same time, using the global motion models.

Let $I_1, I_2, ..., I_m$ be the m input frames. We match feature points between pairs of frames I_d and I_{d+1}, for $d = 1, ..m - 1$. Next we fit k motion models using RANSAC between each pair of adjacent frames.

Let M_d for $d = 1, ...m - 1$ be the set of motion models estimated between frames I_d and I_{d+1}. The initial number of models in each M_d is k. Let M_d^i stand for the ith motion model in M_d, i.e. M_d^i is the ith estimated motion model between frame d and $d + 1$. Fig. 3 is an oversimplified illustration for 3 frames and $k = 4$.

We first perform dense motion segmentation between each adjacent pair of frames independently, using the estimated motion models M_d's. Given a pair of frames I_d and I_{d+1}, the label set $\mathcal{L}$ consists of the k estimated motion models in M_d, with one label per motion model. To densely assign labels to pixels in frame I_d, we perform graph cut optimization with the energy as in Eq. (1). The data terms for pixel p and label l measure how likely is p to have motion M_d^l from frame d to $d + 1$. The data term is based on the color difference between p in I_d and the pixel in I_{d+1} it corresponds to according to motion model M_d^l. We use the Potts smoothness term. Let $S^1, S^2, ..., S^{m-1}$ be the resulting segmentations. Here S^d corresponds to segmentation in the frame number d, and $S_p^d \in M_d$, i.e. S_p^d is the motion model label assigned to pixel p, out of the possible set of motions M_d. Fig. 3 illustrates a hypothetical result of pairwise motion segmentation.

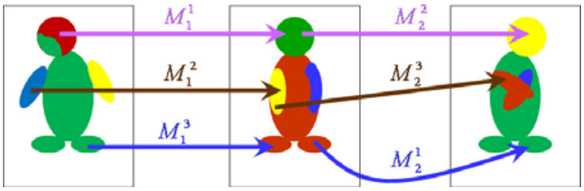


Fig. 3. Oversimplified example of pairwise motion segmentation. Four models are extracted between each pair of frames, i.e. $k = 4$. Different labels are illustrated by different colors. Notice that after motion segmentation on frame pairs, we do not know that the "red" model in frame 1 should correspond to the "green" model to in frame 2 and to the "yellow" model in frame 3. In practice, motion correspondences are not as easy to resolve as in this picture. Three global motion models extracted: purple (combines M_1^1 and M_2^2) brown (combines M_1^2 and M_2^3), and blue (combines M_1^3 and M_2^1).

Initial motion segmentation finds groups of pixels with consistent motion between pairs of frames, but we need such pixel groups across the whole sequence. Thus we perform global optimization across the whole sequence. Let $1, 2, ...c$ be the c hypothetical global motion labels. Each individual global motion label l should describe how pixels obeying global motion l move from the first sequence to the second, from second to the third, etc. We have motion models M_d^i, that describe how pixels move from frame d to $d+1$, but we do not know how these same pixels move from frame $d+1$ to $d+2$. That is, for a motion model in M_d, we do not know the "corresponding" motion model in M_{d+1}. We use the following heuristic but simple procedure for determining which motion model in M_{d+1}^i corresponds to motion model M_d^j. Consider the motion segmentations $S^1, ...S^{m-1}$, performed between pairs of frames individually. Let $R_i^d = \{p \in \mathcal{P} | S_p^d = M_d^i\}$, that is R_i^d is the set of pixels that are labeled with motion i in frame d. Let us warp pixels in R_i^d to frame $d+1$ using motion model M_d^i, let $W(R_i^d)$ the set of warped pixels. If at least 80% of pixels in $W(R_i^d)$ are assigned to the same motion model, say model M_{d+1}^j, and if the size of $W(R_i^d)$ is equal to at least 80% of all pixels in S^{d+1} that are assigned motion model M_{d+1}^j, then we say that motion model M_d^i corresponds to motion model M_{d+1}^j. In Fig. 3 the corresponding motion models are indicated by the arrows with the same color. Occasionally we need to combine two or three motion models to satisfy this condition, i.e. we need to take several models in frame d so that pixels assigned to either of these models make 80% of pixels assigned to the same motion in frame $d+1$. This happens because motion segmentation occurs at different levels of precision. For example, between frames d and $d+1$, an arm could be fitted with two motions, but between the next pair of frames, $d+1$ and $d+2$, the whole arm is fitted with one motion. In such cases, we add new motion models to sets M^d

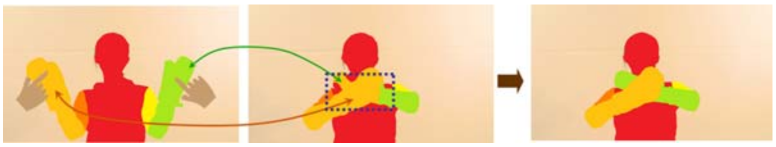


Fig. 4. User Interaction and motion segmentation correction

and M^{d+1}, the model allowing for no arm splitting in M^d (the added model is simply a combination computed from the two motion models allowing the split), and the motion model with arm split to M^{d+1} (the new model is based on warping the two "split" models from the previous pair of frames).

The procedure described in the previous paragraph creates many global motion labels by linking labels between pairs of frames into a single chain, see Fig. 3. Notice that chains can start after the first frame and end before the last frame, allowing for appearance of new layers and disappearance of old layers.

With global motion labels, we can perform global layer segmentation. However, performed on the pixel level, the whole sequence does not fit into the memory on 32-bit architecture. Therefore, we oversegment each frame into "superpixels" using the segmentation algorithm of [20]. Optimization is performed by assigning labels to superpixels, resulting in huge memory savings. The neighborhood system is three-dimensional, with superpixels between the frames also connected. Specifically, we connect a superpixel p in frame d to the closest superpixel q in frame $d+1$. This is justified because we expect the motions to be relatively slow. Data terms are still based on color similarity. For a superpixel, the data term is computed as the average of data terms for its pixels.

6.3 User Interaction

The initial results of motion segmentation are not likely to be accurate for all frames. Therefore we ask the user for guidance.

We sample one fifth of the frames and show their motion segmentation to the user. To correct segmentation, starting with the first frame that is not accurately segmented, the user has to point out its correct segmentation in a nearby frame, by a single click. Consider Fig. 4. The middle pictures shows segmentation results with gross errors due to occlusion, highlighted with a rectangle. The hands are correctly segmented in the frame on the left.

6.4 Correction of Motion Segmentation

Let $F^1, F^2, ..., F^{m-1}$ be the motion segmentation with global labels. Suppose the user clicks on a group of pixels assigned a global label l in frame i. Let G^i_l be this group of pixels, i.e. G^i_l is spatially contiguous, contains the pixel the user clicked on and $G^i_l = \{p \in \mathcal{P} | F^i_p = l\}$. We fix the labels of pixels in G^i_l to strongly prefer label l in the ith frame. That is we set the data penalties to be infinite for all labels other than l for pixels in G^i_l in the ith frame. Furthermore, we warp

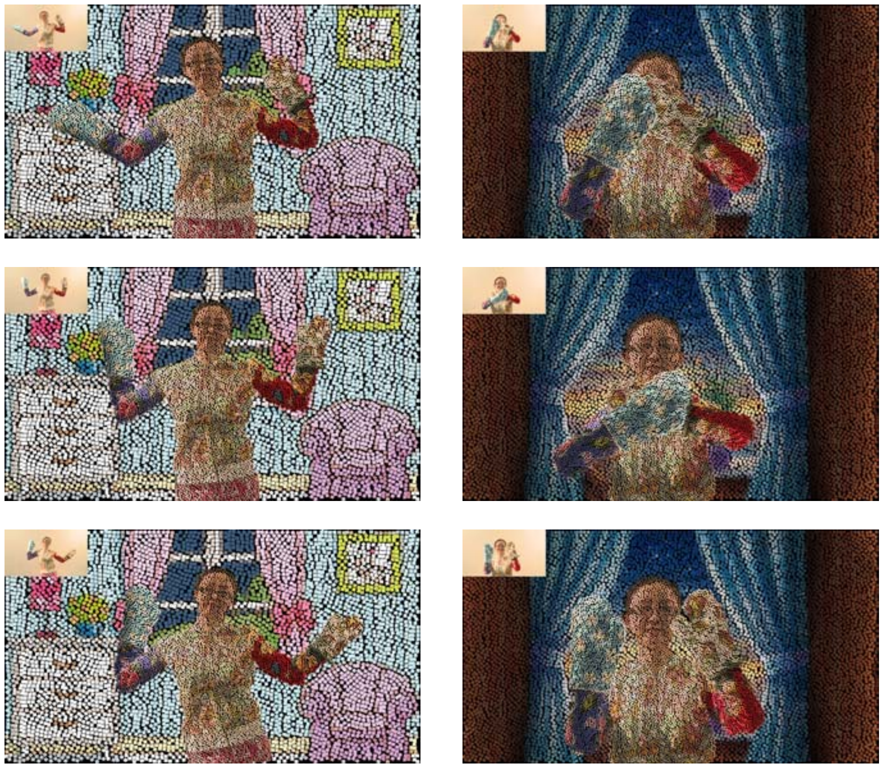


Fig. 5. Results on “Waving arms” sequence and “Overlapping arms” sequence

pixels in G_l^i to the $(i+1)$th frame according to the motion label l. Let $W(G_l^i)$ be the set of warped pixels in frame $i+1$. We set w_{pq} (see Sec. 3) between pixels in G_l^i and $W(G_l^i)$ to be large. Here p is a pixel in frame i and q is the pixel in frame $i+1$ that q gets warped to by the global motion model l.

Now we are ready to talk about occlusion handling. The coefficient w_{pq} is also set in proportion to the color similarity between pixels p and q. The more similar are the colors, the higher is w_{pq}. Weighting w_{pq} in direct proportion to color similarity helps us to handle occlusions automatically. Consider Fig. 4 again. Let O be the group of pixels in the area where the left hand occludes the right hand. Both the left hand pixels and the right hand pixels in the first frame get connected by strong links to pixels in O. However, the links from the left hand are stronger, since the left hand pixels are actually visible in the second frame and their color similarity, on average, is stronger than that between the right hand and pixels in O. Therefore pixels in O get assigned the correct label.

After the data terms and the neighborhood weights w_{pq} are updated, the motion segmentation is recomputed again, propagating user corrections throughout the whole sequence and resolving occlusions.

6.5 Mosaic Rendering

Now we are ready to pack tiles. We start with the "key" segments pointed out by the user, since these are likely to correspond to image data of high quality.

For a still mosaic, given a pixel p and orientation label φ, we need to decide on the penalty of placing tile with center at p and orientation φ. This penalty is modeled from the data around pixel p. For a video sequence, the penalty should depend not just on the current frame, but on all the other frames in the sequence. Let K be a "key" segment in frame I^d that the user clicked on. If we place a tile centered at p under orientation φ, this tile will be propagated by the global motion model throughout the whole sequence. Therefore, to model the data penalty, we propagate the tile throughout the whole sequence (notice its orientation will change in different frames) and compute the data penalty in each frame of the sequence, using the same procedure in each frame as for the still mosaic. The final data term for pixel p to have a tile centered at it with orientation φ in frame I^d is the average of all the data terms from all the frames.

After packing the key segments and propagating them throughout the sequence, we pack the empty regions. We start with the first frame, pack any unprocessed regions and propagate them throughout the whole sequence using the same algorithm as for the key segments. If there are any unprocessed regions in the second frame (for example, because a new motion label appears), we repeat the procedure. We stop when the whole sequence is packed with tiles. The final step is to paint the tiles with the colors of the underlying image.

7 Experimental Results

In Fig. 1 we show the "Waking" sequences. Observe how each individual frame of animation is a pleasing mosaic. This sequence contains significant occlusions, and parts of the leg appear and disappear from the scene. Our system produces a nice coherent animation, with correctly handled occlusions. Due to our restricted motion assumption, the animated figure has a distinctive "puppet"-like effect.

Fig. 5 shows two more video sequences. The "Waving arms" sequence is relatively simple, with no significant occlusions. The motion of the torso is modeled with two layers, creating an interesting visual effect. The "Occluding arms" sequence has significant overlap between the two arms, which is handled gracefully. The torso and the head have motions very close to stationary. We decided to fix the head and the body to be stationary which visually blends them into the background, creating interesting "arms sticking out of the wall" effect. Our results are best viewed from the animations on the web.[1]

[1] see *http://www.csd.uwo.ca/faculty/olga/VideoMosaic/results.html*

References

1. Hausner, A.: Simulating decorative mosaics. In: Proceedings of SIGGRAPH 2001, pp. 573–580 (2001)
2. Elber, G., Wolberg, G.: Rendering traditional mosaics. The Visual Computer 19, 67–78 (2003)
3. Blasi, G.D., Gallo, G.: Artificial mosaics. The Visual Computer 21, 373–383 (2005)
4. Dalal, K., Klein, A.W., Liu, Y., Smith, K.: A spectral approach to npr packing. In: NPAR 2006, pp. 71–78 (2006)
5. Liu, Y., Veksler, O., Juan, O.: Simulating classic mosaics with graph cuts. In: Yuille, A.L., Zhu, S.-C., Cremers, D., Wang, Y. (eds.) EMMCVPR 2007. LNCS, vol. 4679, pp. 55–70. Springer, Heidelberg (2007)
6. Litwinowicz, P.: Processing images and video for an impressionist effect. In: SIGGRAPH 1997, pp. 407–414 (1997)
7. Hertzmann, A., Perlin, K.: Painterly rendering for video and interaction. In: NPAR 2000, pp. 7–12 (2000)
8. Klein, A.W., Sloan, P.P.J., Finkelstein, A., Cohen, M.F.: Stylized video cubes. In: SCA 2002, pp. 15–22 (2002)
9. Wang, J., Xu, Y., Shum, H.Y., Cohen, M.F.: Video tooning. In: SIGGRAPH 2004., pp. 574–583 (2004)
10. Smith, K., Liu, Y., Klein, A.: Animosaics. In: SCA 2005, pp. 201–208. ACM, New York (2005)
11. Battiato, S., Di Blasi, G., Gallo, G., Guarnera, G., Puglisi, G.: Artificial mosaics by gradient vector flow. In: Proceedings of EuroGraphics (2008)
12. Boykov, Y., Veksler, O., Zabih, R.: Efficient approximate energy minimization via graph cuts. IEEE transactions on PAMI 21, 1222–1239 (2001)
13. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transactions on PAMI 24, 137–148 (2004)
14. Elgammal, A.M., Harwood, D., Davis, L.S.: Non-parametric model for background subtraction. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1843, pp. 751–767. Springer, Heidelberg (2000)
15. Sun, J., Zhang, W., Tang, X., Shum, H.: Background cut. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 628–641. Springer, Heidelberg (2006)
16. Adelson, E., Weiss, Y.: A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In: CVPR 1996, pp. 321–326 (1996)
17. Wills, J., Agarwal, S., Belongie, S.: What went where. In: CVPR, vol. I, pp. 37–44 (2003)
18. Xiao, J., Shah, M.: Motion layer extraction in the presence of occlusion using graph cuts. PAMI 27, 1644–1659 (2005)
19. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, pp. 726–740 (1987)
20. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Int. J. Comput. Vision 59, 167–181 (2004)
21. Shi, J., Tomasi, C.: Good features to track. Technical report, Ithaca, NY (1993)