Simulating Classic Mosaics with Graph Cuts

Yu Liu, Olga Veksler, and Olivier Juan

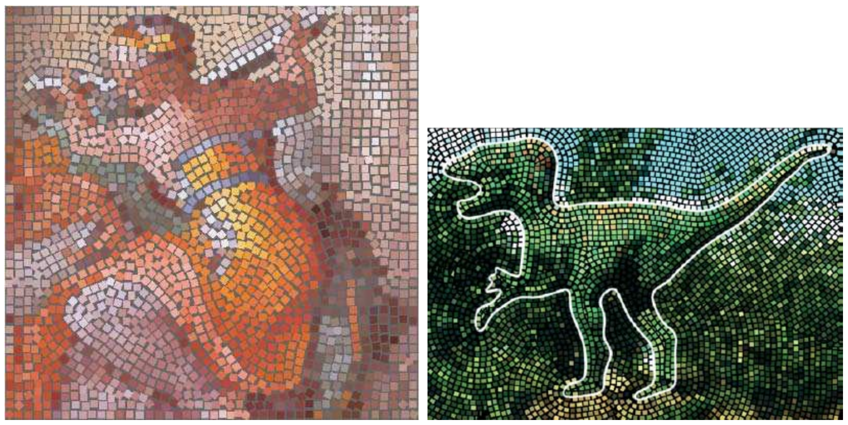
Department of Computer Science
University of Western Ontario
London, Ontario
Canada, N6A 5B7
TEL: +1-519-661-2111 ext 81417
yliu382@csd.uwo.ca, olga@csd.uwo.ca,juan@csd.uwo.ca

Abstract. Classic mosaic is one of the oldest and most durable art forms. There has been a growing interest in simulating classic mosaics from digital images recently. To be visually pleasing, a mosaic should satisfy the following constraints: tiles should be non-overlapping, tiles should align to the perceptually important edges in the underlying digital image, and orientation of the neighbouring tiles should vary smoothly across the mosaic. Most of the existing approaches operate in two steps: first they generate tile orientation field and then pack the tiles according to this field. However, previous methods perform these two steps based on heuristics or local optimisation which, in some cases, is not guaranteed to converge. Some other major disadvantages of previous approaches are: (i) either substantial user interaction or hard decision making such as edge detection is required before mosaicing starts (ii) the number of tiles per mosaic must be fixed beforehand, which may cause either undesired overlap or gap space between the tiles. In this work, we propose a novel approach by formulating the mosaic simulating problem in a global energy optimisation framework. Our algorithm also follows the two-step approach, but each step is performed with global optimisation. For the first step, we observe that the tile orientation constraints can be naturally formulated in an energy function that can be optimised with the α-expansion algorithm. For the second step of tightly packing the tiles, we develop a novel graph cuts based algorithm. Our approach does not require user interaction, explicit edge detection, or fixing the number of tiles, while producing results that are visually pleasing.

1 Introduction

Mosaic is one of the most ancient and durable art forms. Since ancient Greek and Roman times, people used beautiful, fascinating mosaics to decorate floor pavements, wall murals and ceilings. Classic mosaics are composed of a huge number of small tiles with regular shapes, such as rectangles and squares. Simulating classic mosaics automatically is one of the areas in non-photorealistic rendering that has been investigated by many researchers. In recent years, there has been a rapid growth in non-photorealistic rendering techniques, since such techniques can emphasise important aspects of a scene and create digital art.

There are two main challenges for mosaic simulating. First, tile orientations should emphasise the edges of perceptually important shapes in the image. This is achieved by placing tiles parallel to the edges to be emphasised. In addition, tiles must be packed tightly while preserving their completeness. Inspired by the artists' work, many approaches have been developed for simulating this process.



(a) (b)

Fig. 1. Previous work: (a) Result from Hausner [1]. (b) Artifacts created by Elber and Wolberg [2]. The feature curve outlined by the user is in white. Notice that even the tiles that are far away in the background are still aligned with the outline of the dinosaur, even though perceptually significant edges are present in the background.

An early and successful work on rendering classic mosaics was introduced by Hausner [1], see Fig. 1(a). In Hausner, the user is required to identify the perceptually important edges. Then the distance transform to these edges is computed, and the gradients of this distance map are adopted as the tile orientations. Finally the Centroidal Voronoi Diagram (CVD) based on Manhattan distance is used to pack the tiles, assuming that the number of tiles is known.

Elber and Wolberg [2] also require a user to draw several closed curves around the edges to be emphasised. A set of offset curves parallel to the feature curves is computed to generate the orientation guide curves. These curves are trimmed to eliminate self-intersection. Tiles are packed along these orientation guide lines under the constraint that they should not touch each other and the gap space between them is small. See Fig. 1(b) for their result.

The above approaches can produce visually pleasing results. However both approaches are based on local heuristics which lead to certain problems. In Hausner, the two main problems are: (a) the convergency of CVD algorithm is not guaranteed and (b) the tile orientations change drastically at the discontinuities of the distance transform gradient map, but these discontinuities usually do not coincide with intensity edges. In Elber and Wolberg's approach, there is big gap space where the curvature of the orientation guide curve is very large and discontinuity in tile orientations at the skeletons of the closed feature curves provided by the user. In addition, both of these approaches require large amount of user interaction to draw the feature edges. Furthermore, in [1], the number of tiles per mosaic image has to be fixed before the algorithm's start, which causes either a lot of undesired overlapping or large gap space. An additional drawback of Elber and Wolberg's approach is that the tiles in the background which are far away from the foreground objects are also aligned with the feature curves outlined by the user, which creates the artifacts shown in Fig. 1(b).

Other approaches for simulating classic mosaics include Di Blasi and Gallo [3], Battiato et. al [4], Schlechtweg et al. [5]. All of these approaches are based on local optimisation. Some of them replace the user drawn curves with the results of an edge detector. However the state of the art in edge detection does not allow yet to find perceptually important edge robustly. Some commercial image processing software also tries to simulating the visual effects of mosaics, such as Adobe Photo shop [6], but they actually perform simple resolution reducing processing which does not create effects similar to classic mosaics.

The goal of our work is automatic classic mosaic simulation without user interaction or explicit edge detection. We observe that we can formulate the tile orientation estimation and tile packing steps in a global energy optimisation framework, which helps to avoid the problems of local optimisation mentioned above. Like the previous approaches, we wish to make tiles align to perceptually important edges in the image while optimising the gap space between tiles. Unlike the previous approaches, we completely prohibit overlapping tiles.

Our algorithm has three major steps. In the first step, we generate a tile orientation field. Each tile is encouraged to take orientation that aligns it to the nearby strong intensity edges (if any) in the underlying image. In addition, tile orientations are encouraged to vary smoothly over the image, which forces the tiles to create a pleasing visual effect and also helps to reduce gap space between tiles, since there is less gap space between neighbouring tiles with similar orientations. We can encode the strong edge alignment and smoothness constraints in a global energy function which we then optimise with the α-expansion algorithm of [7]. The benefits of this approach to tile orientation generation is that smoothness of tile orientations is enforced globally in the whole image (unlike the distance transform approaches), and explicit edge detection (either by the user or an edge detector) is eliminated. None of the existing methods addresses the smoothness of tile orientations with global optimisation. We chose the α-expansion algorithm of [7] because it has been successfully applied to many optimisation problems in vision and graphics, such as texture synthesis [8], image stitching [9][10] and stereo correspondence [7][11][12].

After we have generated a smooth orientation field for the tiles, we can begin mosaic building. Unfortunately, if we formulate tile packing for the final mosaic as a global optimisation in the straightforward manner, the resulting energy is prohibitively expensive to optimise, since the tile packing problem is essentially a bin-packing problem. Our solution is inspired by the texture synthesis [8] and image stitching algorithms [9][10] . We generate several mosaic layers and stitch them together to form a final mosaic

Thus the second step of our algorithm is generating multiple candidate mosaic layers, using a reasonable heuristic. These layers are usually not good enough to be visually pleasing. There may be large gap space between tiles and many tiles might cross sharp intensity edges, which blurs the final mosaic since the tile colour is the average colour of the image pixels it covers. In the third and final step, we develop a novel layer stitching algorithm, that selects good parts from all the candidate layers and stitches them together to form the final mosaic. Here "a good part" means a region in a candidate mosaic layer where the tiles are packed tightly and do not cross strong intensity edges. This third step is also done in the energy optimisation framework. The constraints of gap space minimisation, edge avoidance, and prohibiting tile overlap are encoded in an energy function, which is then optimised with a novel graph cuts based optimisation algorithm.

2 Energy Optimisation with Graph Cuts

In this section, we briefly review graph cuts based optimisation. Many problems in vision and graphics can be naturally represented as labelling problems, such as image segmentation, stereo, and motion. Let $\mathcal{P}$ be the set of all image pixels and suppose for each $p \in \mathcal{P}$ we wish to assign some label $f_p \in \mathcal{L}$. Let $f = \{f_p | p \in \mathcal{P}\}$ be a labelling that assigns each pixel $p \in \mathcal{P}$ a label $f_p \in \mathcal{L}$. The following energy function is formulated to measure the quality of f:

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

Here, E_{smooth}, which is often called the smoothness term, measures the extent to which f is not smooth. E_{data}, usually called the data term, measures how pixels in $\mathcal{P}$ like the labels that f assigns them. E_{data} is often formulated as

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{2}$$

where D_p is the penalty for assigning pixel p the label f_p. A typical choice for E_{smooth} is

$$E_{smooth} = \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{3}$$

Usually, $\mathcal{N}$ consists of pairs of immediately adjacent pixels, that is the interactions are given by the standard 4-connected grid. Graph cut [7] has been proved to be very successful in optimising these types of energies [13]. We use max-flow implementation of [14] for computing minimum cut.

3 Simulating Classic Mosaics with Graph Cuts

In this section, we give a detailed description of our algorithm. In section 3.1, we give some necessary definitions, in section 3.2 we explain the first step of our algorithm, which is generating smooth tile orientations, in section 3.3 we explain the second step of our algorithm, which is generating multiple candidate mosaic layers, and finally, in section 3.4 we explain our last step, which is stitching the candidate mosaic layers together to obtain the final optimised mosaic.

3.1 Notation and Definitions

Let I be the rectangular image grid where we want to place the tiles and let $\mathcal{P}$ be the collection of all pixels inside I. We assume that all tiles are square with the side $tSize$. A tile is denoted by $t = \{p_t, \varphi_t, \mathcal{T}_t\}$. Here p_t is a pixel inside the image I such that p_t is at the centre of tile t. The angle $\varphi_t \in [0, \frac{\pi}{2})$ is the tile orientation. Since our tiles are rotationally symmetric, we need angles only in the range of $[0, \frac{\pi}{2})$. $\mathcal{T}_t$ is the set of pixels in the image that tile t covers. To find the pixels in $\mathcal{T}_t$, we build a coordinate system with origin at p_t, and the horizontal and vertical axes parallel to that of the image plane I. If the orientation of tile t is $\varphi_t \in [0, \frac{\pi}{2})$, then a pixel (x, y) is inside tile $\mathcal{T}_t$, if its coordinates (x, y) satisfy:

$$\begin{array}{l} |x \times \cos(\varphi_t) + y \times \sin(\varphi_t)| \leq \frac{tSize}{2} \\ |y \times \cos(\varphi_t) - x \times \sin(\varphi_t)| \leq \frac{tSize}{2} \end{array} \tag{4}$$

Our discrete optimisation framework requires that the set of angles is finite. Here we discretise the tile orientations into n angles. Let l_i be the ith angle, then it is defined as:

$$l_i = \frac{\pi}{2n} \times (i-1), \quad i = 1, 2, \ldots, n$$

A label set $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ represent the orientations. If p_t is assigned l_i, then tile $t = \{p_t, \varphi_t, \mathcal{T}_t\}$ has an orientation of $\varphi_t = l_i$. We set n to 16.

3.2 Generating Tile Orientations

In this section, we describe how we compute the tile orientation field. That is for each pixel p, we compute the appropriate tile orientation, assuming that a tile will be placed with its centre at pixel p in the final mosaic. Of course, in the final mosaic only a fraction of all pixels will, in fact, become tile centres. Most of the pixels will be covered by some tile, but will not be in the centre of the tile that covers them, and some pixels will be in the gap space, that is they will not be covered by any tile. Selecting pixels that will become tile centres is addressed in the second and third step of the algorithm, described in sections 3.3 and 3.4.

Our energy function for smooth tile orientation field is given by equations (1), (2), and (3), where f_p is the orientation label assigned to pixel p. This energy encodes the constraints of edge alignment and strong edge avoidance in the data term $D_p(f_p)$, and the smoothness of orientation between neighbouring pixels in the smoothness term $V_{pq}(f_p, f_q)$. The α-expansion method based on graph cuts [7] is applied to minimise this energy.

Data Term. We first discuss our data term, which has the following form:

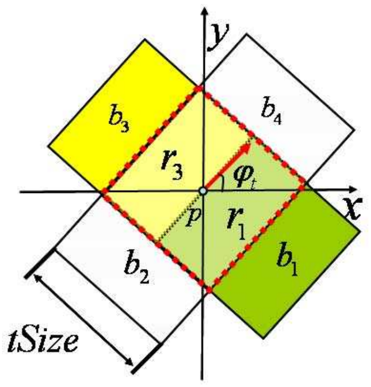
$$D_p(f_p) = E_p^{align}(f_p) + E_p^{avoid}(f_p), \tag{5}$$

where $E_p^{align}(f_p)$ encodes edge alignment constraints, $E_p^{avoid}(f_p)$ encodes edge avoidance constraints. Assuming that a pixel p will become a tile centre, and knowing the tile size, it is fairly easy to estimate which orientation is appropriate for a square tile of fixed size with the centre at pixel p, so that it aligns with a strong intensity edge in its neighbourhood, if any. It is defined as:

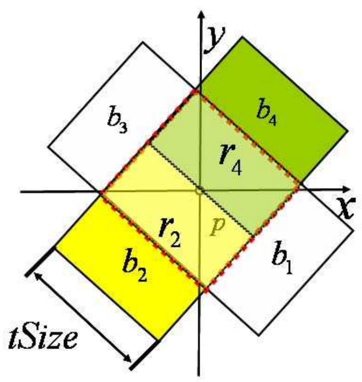
$$E_p^{align}(f_p) = w_e \times \max_{i=1\ldots4} \|C_{r_i}(p) - C_{b_i}(p)\| \tag{6}$$

Let t be the tile $t = \{p, f_p, \mathcal{T}_t\}$. The darker yellow and green rectangles in Fig. 2 show how regions b_i are defined, and the light yellow and green rectangles in Fig. 2 illustrate the r_i regions. $C_{r_i}(p)$ is the average colour vector in region r_i and $C_{b_i}(p)$ is the average colour vector in region b_i. $\|C_{r_i}(p) - C_{b_i}(p)\|$ is the magnitude of the average colour difference between regions r_i and b_i.

This term encourages pixel p which is close to an edge to be assigned the



(a)



(b)

Fig. 2. Definition of $E_p^{align}(f_p)$. A tile $t = \{p_t, \varphi_t, \mathcal{T}_t\}$ is shown by the red dotted squares in (a) and (b).Region r_i is inside tile t and it shares one edge with tile t, for example, in (a), r_1 and r_3 are shown in lighter green and yellow rectangles inside the tile t. In (b), regions r_2 and r_4 are illustrated by the lighter yellow and lighter green rectangles. Region b_i is a rectangle on the border of tile t and b_i shares the same edges with tile t and region r_i. The darker green and darker yellow rectangles in (a) are regions b_1 and b_3 when (b) shows regions b_2 and b_4. The average colour differences between r_i and b_i is measured in E_p^{align}.

orientation that aligns the tile centred at p with the edge. The weight of the colour difference term w_e is set to be negative. Thus when there is a high response on the colour difference between the pixels inside the tile and that outside the tile, the term $E_p^{align}(f_p)$ is negative, encouraging high contrast.

Strong edge avoidance is encoded in $E_p^{avoid}(f_p)$, which is defined as:

$$E_p^{avoid}(f_p) = w_v \times \sum_{q \in \mathcal{T}_t} \|g(q)\| \tag{7}$$

Let $I(p)$ be the intensity at pixel p and $\|g(p)\|$ be the magnitude of the gradient at pixel p. Let $p = (x, y)$, then $g(p) = g(x, y) = \langle g_x(x, y), g_y(x, y)\rangle$. We approximate gradient by the standard Sobel operator. This term measures the intensity variance inside the tile, therefore we call it the variance term. If the label f_p makes the tile overlap a strong intensity edge, then the variance term will penalise the overlap between the edge and the tile. This term is particularly important for pixels close to the edges of an object. The weight of w_v is set to be positive, since we want high gradient to be penalised.

Notice that the $D_p(f_p)$ term involves summation over a potentially large group of pixels (if $tSize$ is large). To compute the data term $D_p(f_p)$ efficiently, the "integral image" [15] is used in our approach. The integral image approach allows computing $D_p(f_p)$ in constant time, independent of the tile size $tSize$.

Smoothness Term. We define the interaction term $V_{pq}(f_p, f_q)$ as:

$$V_{pq}(f_p, f_q) = w_s \times |f_p - f_q|_{\mathrm{mod}(\frac{\pi}{2})} \tag{8}$$

where

$$|f_p - f_q|_{\mathrm{mod}(\frac{\pi}{2})} = \begin{cases} |f_p - f_q| & \text{if } |f_p - f_q| \le \frac{\pi}{4} \\ \frac{\pi}{2} - |f_p - f_q| & \text{otherwise} \end{cases}$$

This interaction reflects the fact that the angle of $\frac{\pi}{2}$ leads to the same tile placement as the angle of 0, due to the symmetry of the square shape. The smoothness term encourages orientations to propagate smoothly over the image.

3.3 Generating Mosaic Layers

In the second step of our algorithm, we generate a set of mosaic layers. Recall that in the first step of our algorithm, for each pixel p we compute an orientation f_p s.t. if a tile t is placed with its centre at pixel p, it will have orientation f_p. Thus for each pixel, we have a "candidate" tile t for possible inclusion in the final mosaic. We build multiple mosaic layers out of these candidate tiles. To give every candidate tile a chance to be included in the final mosaic while keeping the number of candidate layers to a minimum, we insure that the candidate mosaic layers have no tiles in common, and that every candidate tile is present in one of the candidate layers. We also insure that each candidate mosaic layer does not have overlapping tiles, this is crucial for our algorithm in step 3, see section 3.4.

Mosaic layers are generated iteratively, in a region growing manner. In each iteration we build one mosaic layer and make sure that the current mosaic layer does not contain any of the tiles which are already present in a previous mosaic layer. To build one mosaic layer, we starting at some random pixel s which is not included in any candidate mosaic layer yet. We greedily choose a nearby pixel

p in such a way that the candidate tiles centred at s and p do not overlap but the gap space between the tiles s and p is small. In addition, the candidate tile centred at p must not have been selected yet for any candidate layer.

3.4 Stitching Two Mosaic Layers

After generating a set of candidate mosaic layers, our last step is to stitch them together to form the final mosaic. The stitching should minimise the gap space and and should not contain any "broken" tiles. In addition, tiles are encouraged to avoid crossing strong intensity edges. Therefore, we must take edge avoidance into account. In this section, we develop a novel graph cuts algorithm to solve this stitching problem in an energy optimisation framework.

Minimising gap spaces and prohibiting tile overlap. Suppose all the candidate tiles are indexed as $t_1, t_2, \ldots, t_n$, and $D = \{t_i : i = 1, \ldots, n\}$. A mosaic layer is just a subset of tiles in D. Let $M_1, M_2, \ldots, M_k$ be the mosaic layers. In the second step, we generated them in such a way that for any $i \neq j$, $M_i \cap M_j = \emptyset$, and $\bigcup_{i=1,\ldots,k} M_i = D$. Also, for all $i = 1, \ldots, k$, if $t \neq t' \in M_i$, then tiles t and t' do not overlap, that is for $t = \{p_t, \varphi_t, \mathcal{T}_t\}$ and $t' = \{p_{t'}, \varphi_{t'}, \mathcal{T}_{t'}\}$, $\mathcal{T}_t \cap \mathcal{T}_{t'} = \emptyset$.

Our stitching algorithm is iterative. We start from a random mosaic layer M_i, setting it to be our current solution M_c. In each iteration, we select a random layer $M = M_j$, and then find a tile preserving cut between the current solution M_c and the random layer M. Let R be the result of stitching layers $M_c \neq M$, that is $R \subset (M_c \cup M)$.

f_s	$D_s(f_s)$	Layer
$f_s = 0$	0	$s \in M_c$
$f_s = 1$	A_s	
$f_s = 1$	0	$s \in M$
$f_s = 0$	A_s	

(a): $D_s(f_s)$ without Edge Avoidance.

f_s	$D_s(f_s)$	Layer
$f_s = 0$	$w_g \times \lvert g_s \rvert$	$s \in M_c$
$f_s = 1$	A_s	
$f_s = 1$	$w_g \times \lvert g_s \rvert$	$s \in M$
$f_s = 0$	A_s	

(b): $D_s(f_s)$ with Edge Avoidance.

(f_s, f_r)	$V_{sr}(f_s, f_r)$
(0,0)	0
(0,1)	∞
(1,0)	O_{sr}
(1,1)	0

(c): $V_{sr}(f_s, f_r)$

Table 1. Energy for stitching two layers M_c, M

We formulate the problem of stitching two mosaic layers $M_c \neq M$ together as a labelling problem. Let tile set $S = M \cup M_c$. Our label set is $\{0, 1\}$. Labelling $f = \{f_s | s \in S\}$ assigns a label $f_s \in \{0, 1\}$ to every $s \in S$. Any labelling f corresponds to a stitching $R^f \subset S$ in the following way. For every $t \in M_c$, if $f_t = 0$, then tile $t \in R^f$ and if $f_t = 1$,tile $t \notin R^f$. For every $t \in M$, $f_t = 1$ means that tile $t \in R^f$ and if $f_t = 0$, then $t \notin R^f$. Notice that the meaning of labels $\{0, 1\}$ is reversed for tiles in layers M_c and M.

We define the following energy function:

$$E(f) = \sum_{s \in S} D_s(f_s) + \sum_{s,r \in \mathcal{N}} V_{sr}(f_s, f_r), \tag{9}$$

where the neighbourhood system $\mathcal{N}$ is:

$$\mathcal{N} = \{(s, r) | s \in M_c, r \in M \text{ and } \mathcal{T}_s \cap \mathcal{T}_r \neq \emptyset\}$$

The data term $D_s(f_s)$ is given in Table 1(a). In this table, A_s is the size of the region which is covered only by tile s, see Fig. 3.

The interaction term $V_{sr}(f_s, f_r)$ is defined as table 1(c). Here O_{sr} denotes the size of the overlapping region between tiles $s \in M_c$ and $r \in M$, see Fig. 3. This interaction term insures that no tile overlap occurs in the stitching R which has finite energy $E(f^R)$, see Fig. 3. Notice that the gap space that mosaic layers

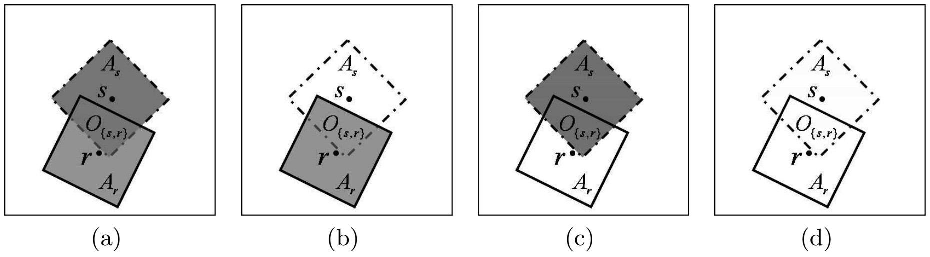


Fig. 3. A simple example to illustrate data and interaction terms. In (a), mosaic layer $M = \{r\}$ is shown by the solid square and $M_c = \{s\}$ is shown by the dotted square. A_r is the region covered only by tile r and A_s is the region covered only by tile s. O_{sr} is the size of overlapping region between s and r. Both tiles s and r presented in the result stitching, therefore, the energy is ∞. In (b), tile s is assigned label 0 and not present in the result stitching. The energy for this case is A_s. In (c), tile r is removed and the energy is A_r. In (d), both tiles are removed, then $V_{sr}(1, 0) = O_{sr}$.

M_c and M have in common can not be filled in by their stitching R. If M is a mosaic layer, let

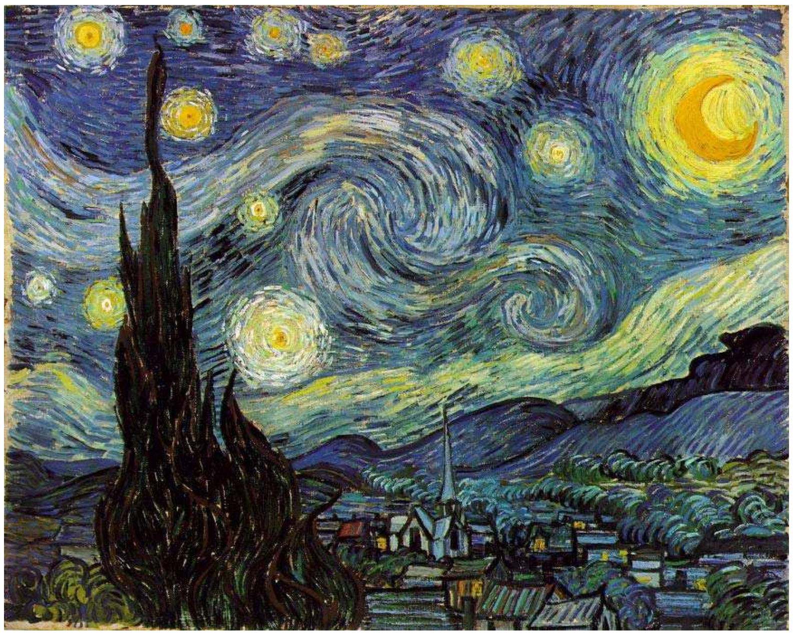
$$G(M) = \{p \in \mathcal{P} | p \notin \bigcup_{s \in M} \mathcal{T}_s\},$$

That is $G(M)$ is the gap space in mosaic layer M. It is easy to see that for any f^R such that $E(f^R) < \infty$,

$$E(f^R) = |G(R)| - |G(M) \cap G(M_c)| \tag{10}$$

That is the energy of f^R is the number of pixels in the gap space of R which were not in the gap space of either M or M_c. Since for any M_c and M, $|G(M) \cap G(M_c)|$ is a constant with respect to our optimisation problem, the optimal f^R will correspond to the stitching R which has $G(R)$, or the gap space, as small as possible. This is exactly the kind of stitching that we desire, minimising the gap space while prohibiting tile overlap.

Edge avoidance. There is an additional constraint that we wish to add to our stitching algorithm. For a visually pleasing mosaic, tiles should be placed to


(a) Input painting

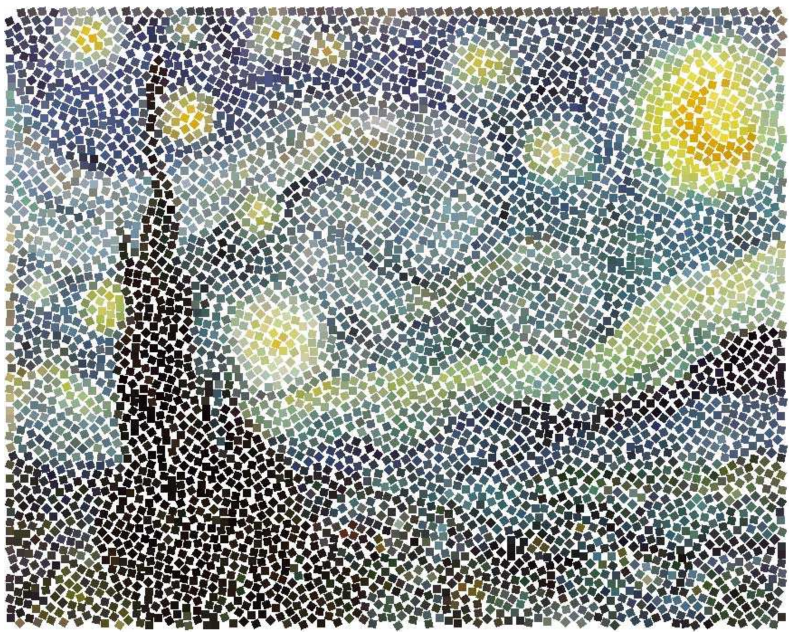

(b) Output Mosaic

Fig. 4. Starry Night

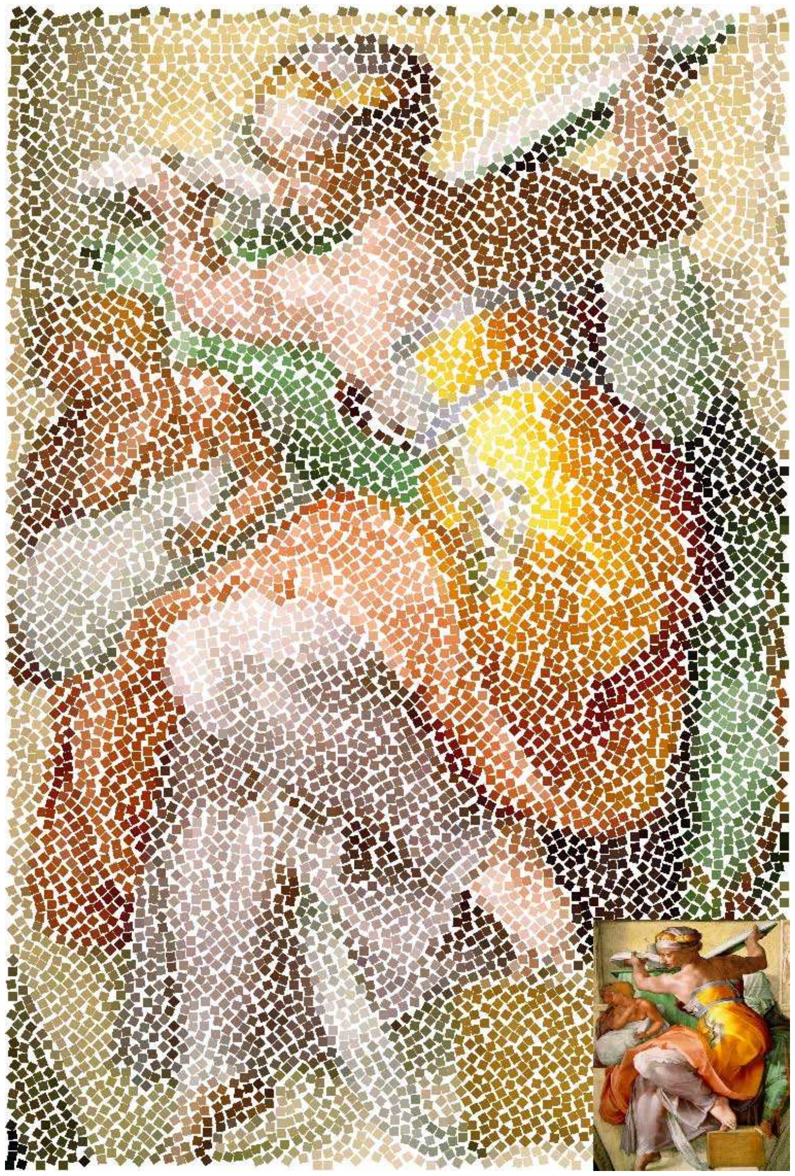


Fig. 5. Libyan Sibyl Mosaic. The input image is zoomed out and illustrated in the right bottom corner.

avoid crossing strong intensity edges inside the underlying image, since otherwise the resulting mosaic is blurred (a colour of a tile is just the average colour of the underlying image pixels that the tile covers). Let g be the intensity gradients for the given image I, where the gradients are computed by standard Sobel operator. The sum of gradient values inside tile $s = \{p_s, \varphi_s, \mathcal{T}_s\}$ is denoted by $|g_s| = \sum_{p \in \mathcal{T}_s} |g(p)|$, where $|g(p)|$ is the magnitude of gradient at pixel p. To incorporate the intensity variance with our energy function defined in Eq. (9), we redefine the data term $D_s(f_s)$ as Table 1(b). The weight of g_s, denoted by w_g, is positive.

With the introduction of the edge avoidance constraint, the energy of stitching current layer changes from that in Eq. (10) to the one below:

$$E(f^R) = |G(R)| - |G(M) \cap G(M_c)| + w_g \times \sum_{t \in R} |g(t)|. \tag{11}$$

It can be seen that the energy in Eq. (11) is submodular [16], and therefore we can optimised it exactly by computing a minimum cut in a certain graph [16]. However, this energy is not quite what we need to measure the quality of stitching R. The $\widetilde{E}(R)$ below is what we really need, since it accounts for the total gap space:

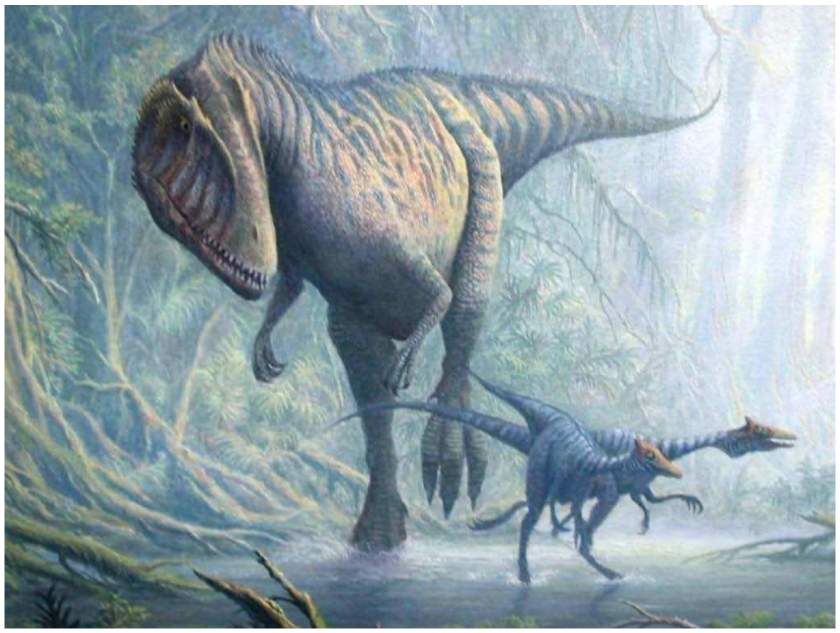
$$\widetilde{E}(f^R) = |G(R)| + w_g \times \sum_{t \in R} |g(t)| = E(f^R) + |G(M) \cap G(M_c)| \tag{12}$$

$\widetilde{E}(f^R)$ simply adds the area of the gap space and the intensity variance inside the mosaic tiles. The problem is that even though for a given layer M, optimising the energy in Eq. (11) gives us the best stitching $R \subset (M_c \cup M)$ of current layer M_c and the new layer M, the common gap space $|G(M) \cap G(M_c)|$ that we cannot optimise for may be too large to give the actual decrease of the desired energy in Eq. (12). The way we solve the problem is as follows. After optimising the energy in Eq. (11), we only switch to the stitching R if $\widetilde{E}(f^R)$ goes down. The steps generating a new layer M, stitching it with the current layer M_c, and updating the current solution in case of decrease in $\widetilde{E}(f^R)$ are performed until the maximum number of iterations.

4 Experimental Results

In this section, we show some results generated by our algorithm. We used the following settings of parameters for all the experiments in this paper: the edge alignment $w_e = -50$, the edge avoidance $w_v = 20$, the smoothness $w_s = 20$, and, finally, the layer stitching $w_g = 0.015$. These values were determined experimentally to give good results for all the images.

In Figs. 5 and 4 we show the original images and our simulated mosaics for the Libyan Sibyl and the Starry Night paintings. The mosaics are visually pleasing, possessing the desired effects. The tile orientations emphasise the shapes of the objects and vary smoothly across the image. For the dinosaur image, see Fig. 6,


(a) Input Image

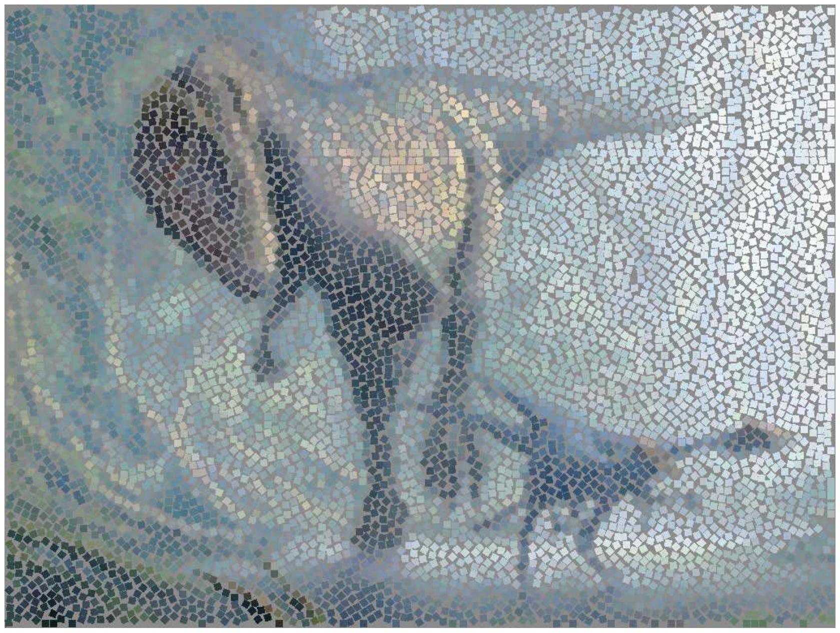
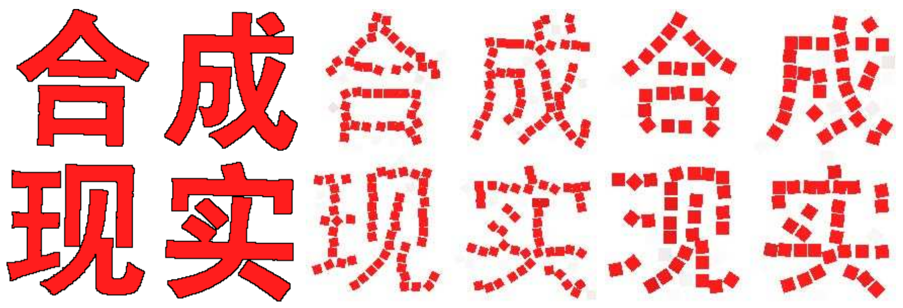

(b) Output Mosaic

Fig. 6. Dinosaur



(a) Input Image (b) Mosaic with small tiles (c) Mosaic with large tiles

Fig. 7. Mosaic images for characters.

the tiles in the mosaic are aligned with the edges of the dinosaurs inside the image, and the tiles in the background are aligned with significant background shapes, unlike the results of Elber and Wolberg [2] in Fig. 1(b). This is because we do not make hard decisions about the boundaries to be emphasised, rather we let our algorithm to discover those boundaries automatically.

To make sure every candidate tile has a chance to appear in the final mosaic, the number of candidate mosaic layers and the number of iterations when stitching the layers together must be large enough. Figs. 8 and 9 show how these two parameters affect the final result. We increase these two parameters gradually until the final result is visually pleasing.

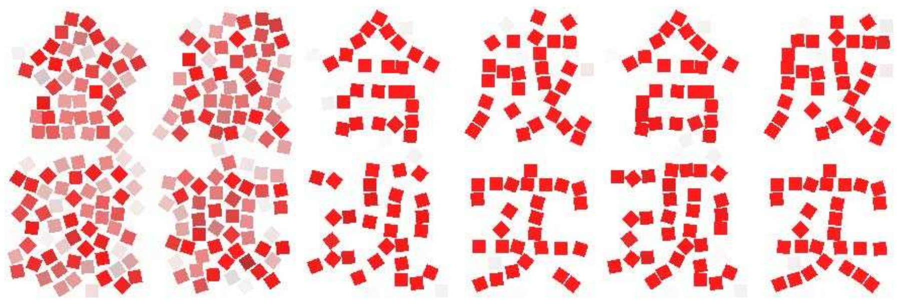
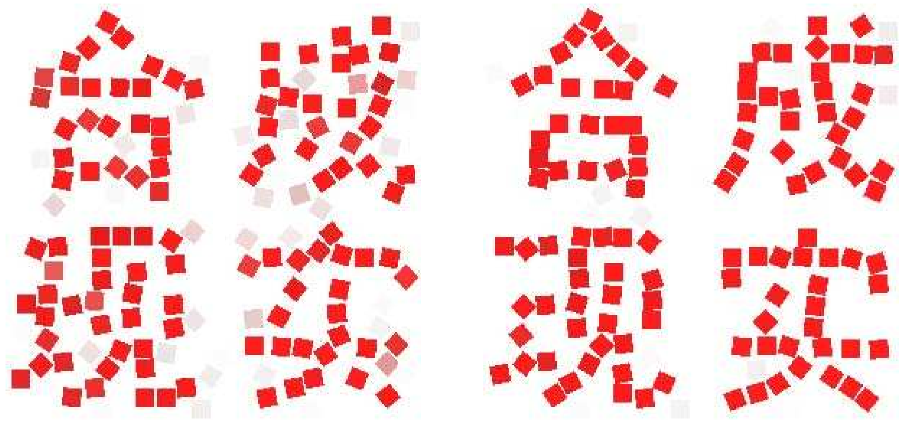

(a) Starting Layer (b) Mosaic after 64 iterations (c) Mosaic after 198 iterations

Fig. 8. Mosaics generated with different numbers of iterations

In our work, we set all the tiles to have the same size and same square shape. It is important to choose an appropriate *tSize* for synthetic images. In



(a) Mosaic with 10 candidate layers (b) Mosaic with 100 candidate layers

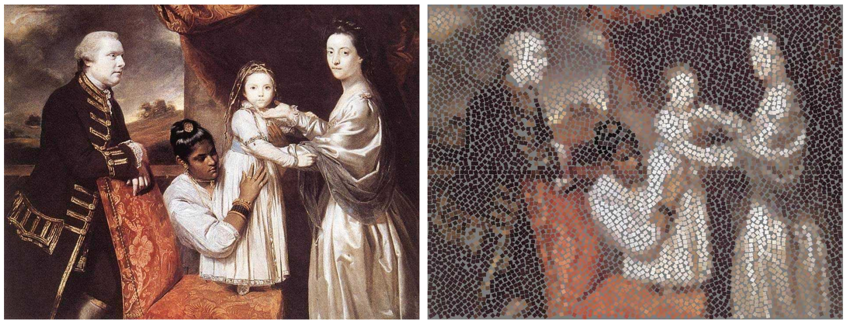
Fig. 9. Mosaics generated with different numbers of candidate layers

Fig. 7(b), the tile size is too small compared to the strokes of the characters. To keep the resulting mosaic from being blurred, the tiles located at the edges of the characters are removed, therefore, the characters in the mosaic look much thinner than those in the input image. In Fig. 7(c), the previous problem is solved by increasing the tiles size, such that $tSize$ is almost the thickness of the character strokes, and the result is much improved.

For some images, we need to vary the tile size for different regions of the image. Small tiles are needed in the image regions which have many fine details, and large tiles work well in areas with larger details. For example, in Fig. 10(b), the background is visually pleasing with the tile size of 10×10 pixels, however, the facial details of the people inside the image are blurred, because the tile size is too large to represent the fine facial details. We plan to incorporate different tile sizes and shapes to our future work.

References

1. Hausner, A.: Simulating decorative mosaics. In: Proceedings of SIGGRAPH2001. (2001) 573–580
2. Elber, G., Wolberg, G.: Rendering traditional mosaics. The Visual Computer **19** (2003) 67–78
3. Blasi, G.D., Gallo, G.: Artificial mosaics. The Visual Computer **21** (2005) 373–383
4. Battiato, S., Blasi, G.D., Farinella, G.M., Gallo, G.: A novel technique for opus vermiculatum mosaic rendering. In: proceedings of ACM/WSCG2006. (2004) 3247–3259
5. Schlechtweg, S., Germer, T., Strothotte, T.: Renderbots-multi-agent systems for direct image generation. Computer Graphics Forum **24**(2) (2005) 137–148
6. PhotoShop: Adobe photoshop (2006)
7. Boykov, Y., Veksler, O., Zabih, R.: Efficient approximate energy minimization via graph cuts. IEEE transactions on PAMI **21**(12) (2001) 1222–1239



(a) Input Image (b) Output Mosaic

Fig. 10. Family

8. Kwatra, V., Schodl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. ACM Transactions on Graphics, SIGGRAPH 2003 **22**(3) (2003) 277–286
9. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. ACM Transaction on Graphics (Proceedings of SIGGRAPH2004) **23**(3) (2004) 294–302
10. Eden, A., Uyttendaele, M., Szeliski, R.: Seamless image stitching of scenes with large motions and exposure differences. In: Proceedings of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Volume 2. (2006) 2498–2505
11. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusion via graph cuts. In: Proceedings of IEEE International Conference on Computer Vision. (2001) 508–515
12. Kim, J., Kolmogorov, V., Zabih, R.: Visual correspondence using energy minimization and mutual information. In: Proceedings of IEEE International Conference on Computer Vision. Volume 2. (2003) 1033–1040
13. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M.: Comparative study of energy minimization methods for markov random fields. In: European Conference on Computer Vision, ECCV 2006,. Volume 2. (2006) 16–29
14. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI) **24** (2004) 137–148
15. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE CVPR2001. Volume 1. (2001) 511–518
16. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(2) (2004) 147–159