COMPUTER GRAPHICS forum

DOI: 10.1111/j.1467-8659.2010.01752.x

COMPUTER GRAPHICS forum
Volume 29 (2010), number 8 pp. 2387–2399

Generating Classic Mosaics with Graph Cuts

Y. Liu[1], O. Veksler[1] and O. Juan[2]

[1]University of Western Ontario, Canada
{yliu382, olga}@csd.uwo.ca
[2]Ecole Centrale de Paris, France
Olivier.juan@gmail.com

Abstract

Classic mosaic is an old and durable art form. Generating artificial classic mosaics from digital images is an interesting problem that has attracted attention in recent years. Previous approaches to mosaic generation are largely based on heuristics, and therefore it is harder to analyse, predict and improve their performance. In addition, previous methods have a number of disadvantages, such as requiring that the number of tiles in a mosaic is known a priori, or relying on extensive user interaction, or using heuristics for tile placement that lead to visible artefacts. We propose a classic mosaic generation algorithm that is based on a principled global optimization. Our approach is fully automatic. We design and optimize an objective function that incorporates the desired mosaic properties, such as tile alignment to significant image edges, prohibiting tile overlap, etc. Our optimization method is based on graph cuts, which proved to be a powerful optimization tool in graphics and computer vision. Experimental comparison to previous work demonstrate the advantages of our approach.

Keywords: non-photorealistic rendering, artificial mosaic, graph cuts, optimization

ACM CCS: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

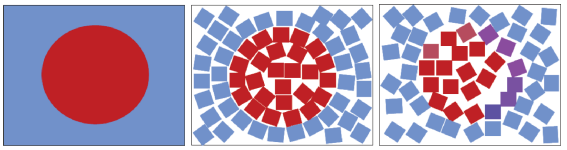
1. Introduction

Classic mosaic is an ancient art form. For thousands of years durable mosaics were used for decoration. A classic mosaic is composed of a large number of small tiles with regular shapes, such as rectangles. Recently there has been interest in generating artificial classic mosaics from digital images.

Artificial mosaic is an example of a non-photorealistic rendering technique. Unlike traditional rendering, which emphasizes faithfulness of the image to a real scene, non-photorealistic rendering focuses on art-like effects, such as engraving [Ost99, PB94], pen-and-ink illustrations [SWHS97, WS96], digital paintings [Her01, Mei96], line-art drawings [Elb95, Elb98], stipple drawing [DHvOS00, HHD03, PFS03, Sec02] and others [KMN*99]. Non-photorealistic rendering is used for drawing attention to the important parts of a scene, abstracting away irrelevant details, creating digital art, etc.

Studying the work of mosaic artists, two main properties of a visually appealing mosaic emerge. 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. For example, the circle in Figure 1(a) is strongly emphasized in Figure 1(b) by placing the tile sides parallel to the circle boundary. If the tiles are placed at random orientations, the circle is emphasized much less, Figure 1(c). Parallel tile placement is by far the most popular way to emphasize the boundaries, although other techniques for boundary emphasis are also possible.

Deciding which curves are perceptually important and are to be highlighted is frequently done with user interaction [Hau01, EW03]. While a human is the ultimate expert, it may be desirable to produce mosaics automatically. Since perceptually important curves tend to coincide with strong colour edges in an image, some methods take advantage of the



(a) (b) (c)

Figure 1: *Principles for an appealing mosaic. (a) Original image. (b) Appealing mosaic: tiles align to strong edges and their orientation changes smoothly; no tiles overlap the boundary between the red circle and the blue background, therefore this boundary appears to be as sharp as in the original image. (c) Unappealing mosaic: tiles do not align to intensity edges and their orientation changes randomly; any tile that overlaps the boundary between the red circle and blue background has a colour that is a blend of red and blue, blurring the boundary.*

information provided by the gradient magnitude to automate the mosaic generation process. Explicit methods [DBG05, BDBFG06] label the boundaries returned by an edge detection or an image segmentation algorithm as perceptually important. Such approach is simple but brittle, since edge detection and image segmentation frequently produce unappealing boundaries. Implicit methods [LVJ07, BDBG*08] use the gradient magnitude only as a soft cue for a possible boundary to emphasize. Whether a pixel with a high gradient magnitude gets emphasized or not depends on other factors, such as preferred tile orientations at other pixels, etc.

The second important mosaic principle is to maximize the number of tiles, while avoiding tile overlap as much as possible. This, combined with the first principle, means that tile orientations should align with important boundaries and vary smoothly in the image, since smoothly varying orientations allow a tighter packing of tiles. In Figure 1(b), tile orientations vary much more smoothly compared to that of Figure 1(c). Therefore, the mosaic in Figure 1(b) has less gap space and is visually more appealing than that in Figure 1(c).

The goal of this paper is to develop a principled global optimization based method for generating classic mosaics. We design and optimize an objective function that incorporates the desired mosaic properties, such as tile alignment to significant image edges, prohibiting tile overlap, etc. We also avoid both user interaction and explicit edge detection that are currently required by most mosaic generation algorithms. Furthermore, unlike many existing approaches, we completely prohibit overlap between tiles.

Preliminary version of this paper appeared in [LVJ07]. Compared to [LVJ07], we have made significant improvements to the algorithm, including a formulation of a better energy function and a better optimization algorithm. We now also provide a comparison to prior work.

This paper is organized as follows. Section 2 discusses previous work, Section 3 is a brief overview of our algorithm, Section 4 explains optimization with graph cuts. Section 5 describes our energy function for mosaic generation, Section 6 gives a detailed description of our optimization strategy. Section 7 presents our experimental results and comparison with previous work. Section 8 is a summary.

2. Previous Work

Many artificial mosaic methods have been proposed based on the principles illustrated in Figure 1 [Hau01, EW03, DBG05, BDBFG06, SGS05, BDBG*08]. Most of these methods have two main steps: generating a tile orientation field and packing the tiles. The details of how these steps are implemented vary greatly between the algorithms, however all of them are based on heuristics, and therefore it is hard to predict, analyse, and improve their performance.

Hausner [Hau01] develops one of the earliest successful algorithms. The approach is interactive, the user outlines perceptually important 'feature edges' in the image to be emphasized by the mosaic. The tiles close to the feature edges should have sides parallel to them, and the other tiles should have smoothly varying orientation. To achieve this, Hausner takes the gradient of the Euclidean distance transform applied to the feature edges as the tile orientation field. The second step of the algorithm is packing the tiles into the mosaic image. For this purpose, a Centroidal Voronoi Diagram (CVD) is used with N randomly placed seeds. By computing CVD, the image plane will be segmented into fair regions. To make the resulting regions almost rectangular, Manhattan distance, rather than Euclidean distance, is used. One tile is placed centred at the seed inside each CVD cell, with its edges parallel to the orientation vector at the cell centre. Therefore, N gives the number of tiles in the mosaic.

Hausner's algorithm produces good results, but it has a number of disadvantages. The biggest disadvantage is that their tile orientation field has discontinuities. Consider Figure 2. The original image with the user drawn edges is in Figure 2(a). Hausner's mosaic is in Figure 2(b). In Figure 2(c), the computed tile orientation field is shown with black and yellow triangular arrows. The yellow arrows highlight some discontinuities in the computed field, they are at locations equidistant from the feature edges. Such discontinuity curves appear emphasized due to tile orientations changing rapidly around them. However, these were not the boundaries that the user meant to emphasize. Another disadvantage of the Hausner's method is that the number of tiles has to be fixed beforehand. If the number of tiles is underestimated, the results could be sparse. If the number of tiles is overestimated, many tiles could overlap. Other disadvantages are that the algorithm does not guarantee that there will be no tile overlap, the CVD algorithm may not converge, and user interaction, which may not be readily available, is required.

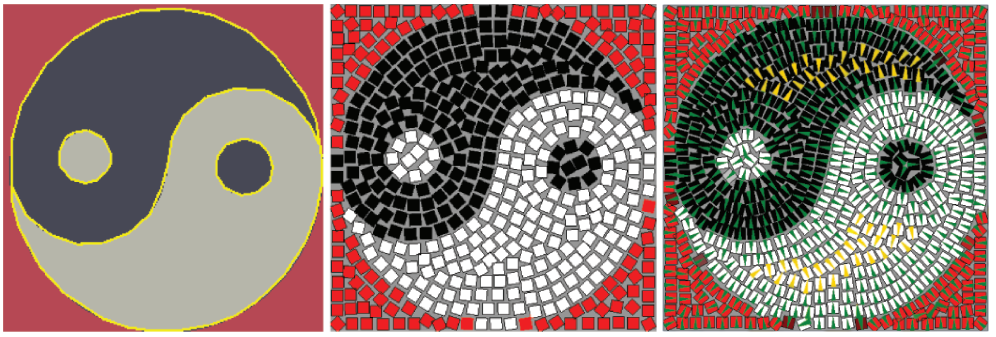


Figure 2: *Discontinuities in the tile orientation field computed by Hausner's algorithm. (a) Original image, feature edges are in yellow. (b) Mosaic produced by Hausner's algorithm. (c) Tile orientation field shown with triangular arrows. Some discontinuities are pointed out with yellow arrows. These results were produced with the code from Hausner's web site.*

An alternative, also interactive, approach to rendering classic mosaics is in [EW03]. The user is asked to draw one or several closed curves around the edges that are to be emphasized. Then a set of 'tile orientation guide' curves is produced. This is the set of curves that are parallel to the user-outlined curves, and the distance between any two neighbouring curves is the size of a square tile. This is basically the set of level lines to the user-drawn curve. Tiles are packed along these guide curves with their sides parallel to the nearest curve, and under the constraint that tiles do not overlap.

Elber and Wolberg's [EW03] tile orientation field also has discontinuities, they lie at the skeleton of the closed feature curves provided by the user. Perhaps the biggest drawback of their method is the following artefact. The tiles that are far away from a user drawn curve still align to it, even if there are other interesting objects around. For example, if the user outlines a foreground object, the background tiles far from the foreground still have orientations that align with the outlined foreground object. Another disadvantage is a large gap space where the curvature of the orientation guide curve is large.

The approach most similar to ours is in [BDBG*08]. Similar to our work, they use optimization to get tile orientation field. Although their energy function is different from ours, the idea is the same: make tiles align to strong image gradients and vary smoothly throughout the image. However, the optimization methods that they use are local and not as powerful as the graph cut based optimization. In addition, the tile packing step is still performed heuristically. The advantage of [BDBG*08] over previous work is that they prohibit tile overlap and do not require user input.

There are mosaic types other than classic. Crystallization mosaic [DHJN02, FDF05, Hae90] is a simulation of a glass-stone mosaic. Photo [DBP05, SH97, KP02, OK08, PCK09] and puzzle mosaic [DBGP05, KP02, PCK09] are two special branches that only exist in artificial mosaic. The former one places thousands of small images in a regular grid to form a bigger image which is unrelated to the small images and is better viewed from a distance. In essence, small images serve the same purpose as pixels. Puzzle mosaic packs a set of small pieces of arbitrary shapes together to generate a scene. Besides static mosaics, researchers also developed some approaches which can create videos with mosaic effect [KGFC02, SLK05].

3. Overview of Our Approach

Our goal is to develop a classic mosaic algorithm based on principled global optimization. Having a global objective function has a number of advantages. We can model the desired mosaic properties directly by including the appropriate terms into the energy function. Successful optimization of this energy guarantees that the mosaic satisfies those properties that we think are desirable. If the results are unsatisfactory, we can rethink and redesign those terms in the energy functions that are likely to be the cause of a failure. By modifying the energy function, new mosaic effects can be introduced. We can also use the value of the energy as a metric for measuring and comparing the quality of mosaics.

The objective function that we design incorporates the properties illustrated in Figure 1, such as tile alignment to significant edges, etc. We also avoid both user interaction and explicit edge detection that are currently required by most mosaic algorithms. Furthermore, unlike many existing approaches, we completely prohibit overlap between tiles.

Our objective function is too hard to optimize in all the variables simultaneously, and therefore we optimize different sets of variables sequentially. First we optimize our tile orientation variables. The constraints are similar to those in previous work: we require that tile orientations vary smoothly and align with the strong intensity edges. This step is performed with the α-expansion algorithm [BVZ01] which is

based on graph cuts. Graph cuts proved to be a powerful optimization tool [SZS*08]. The biggest benefit of using global optimization is that our approach preserves the smoothness of tile orientations as a global property. We do not have discontinuities in tile orientations like those in [Hau01, EW03] because we optimize tile orientations directly and globally. None of the existing approaches enforce the smoothness of tile orientations in a global optimization framework. Furthermore, we eliminate user interaction because explicit edge information is not needed.

In the second step, we optimize over tile visibility variables of our energy function. Intuitively, this step can be seen as stitching together multiple candidate mosaic layers. First we generate multiple mosaic layers obeying the pre-computed tile orientations. A candidate mosaic layer is heuristically generated and therefore is not a good mosaic overall. The gap space between tiles is not optimized and many tiles may be placed over sharp intensity edges, which creates blur, like in Figure 1(c). However, some parts of a candidate layer are good, i.e. the tiles are packed tightly and avoid overlapping the sharp intensity edges. By optimizing over visibility variables of the energy function, we select the good parts from all the candidate layers. This step can be seen intuitively as stitching together candidate mosaic layers. Optimizing over visibility variables is also done in the energy optimization framework with graph cuts [BVZ01].

4. Energy Optimization with Graph Cuts

We formulate the mosaic generation as a labelling problem, and address it in the energy minimization framework. In this section, we explain what a labelling problem is, and the common types of energies formulated for labelling problems. We also briefly review energy minimization with graph cuts.

In a labelling problem, one has a set of pixels $\mathcal{P}$ and a set of labels $\mathcal{L}$. The labels represent some property that one wants to assign to pixels. The task in a labelling problem is to assign some label in $l \in \mathcal{L}$ to each 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 for pixels and labels. Unary constraints, denoted by $D_p(l)$, give the penalty for assigning label l to pixel p. The smaller is $D_p(l)$, the more likely is l for pixel p. $D_p(l)$ is modelled from the observed data. Binary constraints, denoted by $V_{pq}(l_1, l_2)$ express the penalty for assigning labels l_1 and l_2 to two neighbouring pixels p and q. Binary constraints come from prior knowledge about the optimal labelling and encourage some type of smoothness on the labelling.

The following energy function is formulated:

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

$E_{\text{data}}(f)$ is called the data term, and it sums up the unary constraints

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

E_{smooth} is called the smoothness term, and it sums up the binary constraints

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

In Equation (3), $\mathcal{N}$ is a collection of neighbouring pixel pairs, it could be the standard four- or eight-connected grid, or it could include longer-range interactions. The choice of V_{pq} should reflect the a priori knowledge about the labelling one wants to get. Some typical choices are $V_{pq}(f_p, f_q) = |f_p - f_q|$, or $V_{pq}(f_p, f_q) = \min(K, |f_p - f_q|^C)$, where K, C are constants.

In addition to specifying smoothness assumptions, the choice for V_{pq} also determines the choice of the optimization algorithm. A convex V_{pq} leads to an energy that can be optimized exactly with a graph cut [Ish03]. Other commonly used V_{pq} lead to energies that are NP-hard to optimize [BVZ01, KZ04], but there are approximation algorithms with different quality guarantees [BVZ01].

Optimization algorithms based on graph cuts [BVZ01] have proved to be successful in optimizing the energies of the type in Equation (1), see [SZS*08]. We use max-flow implementation of [BK04] for computing the minimum graph cut.

5. Energy Function for Classic Mosaics

In this section, we give a detailed motivation and description of the energy function that we use for mosaic generation. We start by explaining the label set. Any tile can be identified by its centre and orientation. Therefore, we use two labels per pixel. Let I be the colour image for which we wish to generate the mosaic, and let $\mathcal{P}$ be the collection of all pixels inside I. For each pixel $p \in \mathcal{P}$ 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 we place a tile centred at pixel p in the mosaic. If $v_p = 0$ then the mosaic does not have any tiles centred at p. We assume that all tiles are square with the side of size *tSize*.

The second part of the label, φ_p, specifies the orientation of the tile centred at pixel p, if there is such a tile in the mosaic. 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. Our discrete optimization framework requires that the set of orientations is finite. Here we discretize the orientations into n angles, at equal intervals. Because the square tiles have several angles of symmetry, we need angles only in the range of $[0, \frac{\pi}{2})$. The set of all possible orientations is

$$\Phi = \left\{ \frac{\pi}{2n} \times (i-1) | i = 1, 2, \ldots, n \right\}.$$

We set n to 32 for all the experiments.

Occasionally we need to refer to the set of all pixels covered by a tile centred at pixel p and with orientation φ_p. We will denote this set as $\mathcal{T}(p, \varphi_p)$. The colour of the tile is an average of colours over the pixels in I that this tile 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 $\varphi \in \Phi^n$, where n is the size of $\mathcal{P}$.

We are now ready to formulate the energy function for a mosaic (v, φ). Our energy function encodes the following principles for generating a visually pleasing mosaic: tiles should align with strong edges in the image I, nearby tiles should have similar orientations, tiles should avoid crossing strong edges in image I, and, finally, the gap space in the mosaic should be minimal. Our energy function is as follows:

$$E(v, \varphi) = \sum_{p \in \mathcal{P}} (1 - v_p) + \sum_{p \in \mathcal{P}} v_p \cdot D_p(\varphi_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(v_p, v_q, \varphi_p, \varphi_q). \quad (4)$$

The first sum in Equation (4) ensures that the gap space is minimized. The more tiles are placed in the mosaic, the less gap space there is. In addition, this terms ensures that the optimal solution is not the trivial one: $v_p = 0$ for all pixels p. The second sum in Equation (4) is the data term, which measures how well the tiles that we place in the mosaic align to the edges and avoid crossing the edges. The last sum is the smoothness term that encourages nearby tiles to have similar orientations and also prohibits tile overlap. We now discuss the data and the smoothness terms in greater detail.

5.1. Data term

For each pixel p, the data term is $v_p \cdot D_p(\varphi_p)$. The term $D_p(\varphi_p)$ measures the quality of a tile with centre at pixel p and with orientation φ_p. Multiplying by v_p ensures that we consider only the quality of the tiles that are actually present in the mosaic. The D_p has the following form:

$$D_p(\varphi_p) = D_p^{\text{align}}(\varphi_p) + D_p^{\text{avoid}}(\varphi_p), \quad (5)$$

where $D_p^{\text{align}}(\varphi_p)$ encodes edge alignment, encouraging tile sides to be parallel to the intensity edges of the underlying image I, and $D_p^{\text{avoid}}(\varphi_p)$ encodes edge avoidance, pushing the tiles away from intensity edges to prevent blurring.

We first explain the edge alignment term D_p^{align}. Assuming that a pixel p is a tile centre, and knowing the tile size, it is fairly easy to estimate how well a particular orientation aligns a tile to any intensity edge in the neighbourhood. Because the tiles have four sides, we check for the evidence of a strong edge for each one of them, and then choose the side with the strongest evidence. To check for an edge presence, we use the colour difference between boxes around a tile side. We split a tile in half horizontally into regions R_1 and R_3, and then, separately, we split a tile vertically into regions R_2 and R_4, as illustrated with coloured solid rectangles in Figure 3. Regions B_1, B_2, B_3, B_4, illustrated in Figure 3, are located outside the tile, and adjacent to R_1, R_2, R_3, R_4, respectively. To measure the evidence for an edge on the right-hand side of the tile, we take the difference in colour between regions R_1 and B_1. The other sides are handled similarly. Thus, D_p^{align} is

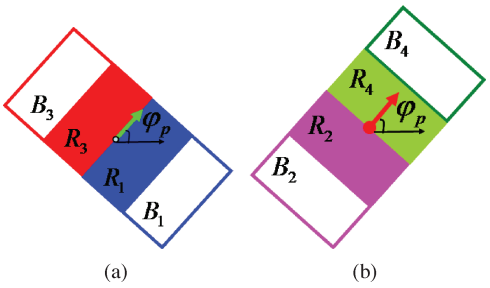


Figure 3: *Shows R and B regions used in of* $D_p^{\text{align}}(\varphi_p)$.

$$D_p^{\text{align}}(\varphi_p) = w_e \times \max_{i=1\ldots4} \left\| \sum_{p \in R_i} I(p) - \sum_{p \in B_i} I(p) \right\|, \quad (6)$$

where $I(p)$ stands for the colour vector at pixel p. The weight w_e is 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 $D_p^{\text{align}}(\varphi_p)$ is negative, making tile orientations with a higher contrast to incur less cost.

The edge avoidance term $D_p^{\text{avoid}}(\varphi_p)$ is defined as

$$D_p^{\text{avoid}}(\varphi_p) = w_v \cdot \sum_{q \in \mathcal{T}(p, \varphi_p)} \|g(q)\|, \quad (7)$$

where $\|g(p)\|$ is the magnitude of the gradient at pixel p, and $\mathcal{T}(p, \varphi_p)$ is the set of pixels covered by the tile with centre at p and orientation φ_p. 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 φ_p makes the tile overlap a strong intensity edge, then the variance term will penalize 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, because we want high gradient to be penalized.

Notice that the $D_p(\varphi_p)$ term involves summation over a potentially large group of pixels if tile size is large. To compute $D_p(\varphi_p)$ efficiently, we use the summed-area table technique [Cro84]. Summed-area tables allows computing $D_p(\varphi_p)$ in constant time, independent of the tile size *tSize*.

5.2. Smoothness term

The last term in Equation (4) is the smoothness term. First we define the neighbourhood system as $\mathcal{N} = \{\{p, q\} | dist(p, q) \leq \sqrt{2} \cdot tSize\}$, where $dist(p, q)$ is the Euclidian distance between the coordinates of pixels p and q. This neighbourhood system is large enough to contain all pairs of pixels s.t. if tiles cantered at these pixels are placed in the mosaic, then these tiles are adjacent or overlapping.

We define the interaction term $V_{pq}(\varphi_p, \varphi_q, v_p, v_q)$ as

$$V_{pq}(\varphi_p, \varphi_q, v_p, v_q) = \begin{cases} 0 & \text{if } v_p = 0 \text{ or } v_q = 0 \\ w_s \cdot |\varphi_p - \varphi_q|_{(\frac{\pi}{2})} & \text{if } v_p = v_q = 1 \text{ and} \\ & \mathcal{T}(p, \varphi_p) \cap \mathcal{T}(q, \varphi_q) = \emptyset, \\ \infty & \text{if } v_p = v_q = 1 \text{ and} \\ & \mathcal{T}(p, \varphi_p) \cap \mathcal{T}(q, \varphi_q) \neq \emptyset, \end{cases} \tag{8}$$

where

$$|\varphi_p - \varphi_q|_{(\frac{\pi}{2})} = \begin{cases} |\varphi_p - \varphi_q| & \text{if } |\varphi_p - \varphi_q| \leq \frac{\pi}{4}, \\ \frac{\pi}{2} - |\varphi_p - \varphi_q| & \text{otherwise .} \end{cases} \tag{9}$$

The smoothness term serves two purposes. First, any finite energy labelling has no overlapping tiles. Secondly, it encourages orientations of adjacent tiles to have similar orientations. Notice that we only consider the orientations of neighbouring tiles that are actually placed in the mosaic. The modulo arithmetic in Equation (9) reflects the fact that rotation by angle φ_p gives the same result as rotation by angle $\varphi_p + \frac{\pi}{2}$, due to the symmetry of a square. Thus, the penalty for two neighbouring pixels to have different orientation labels is an absolute difference of their labels, modulo $\frac{\pi}{2}$ arithmetic.

A mosaic (v', φ') that has a low value of energy in Equation (4) is expected to be visually pleasing. Any other desired mosaic properties can also be included. However, successful optimization depends on the particular form of the energy function. There may be properties that one wishes to include that make the energy very hard to optimize. For example, we may wish to include terms that make the gap space evenly distributed throughout the mosaic. However, such terms would require higher order interactions, which are much harder to optimize. We found that the energy in Equation (4) offers a nice balance between containing the most important terms for a pleasing mosaic, and being reasonable to optimize.

6. Optimization

In this section, we describe our optimization approach. The energy in Equation (4) is too difficult to optimize in all variables simultaneously. We devise a stepwise approach for approximation. First, we ignore the tile visibility labels v_p, and optimize tile orientation variables φ_p (Section 6.1). Keeping tile orientation variables φ_p fixed, we then optimize for the visibility variables v_p (Section 6.2).

6.1. Optimizing in orientation variables

We now explain how to optimize the orientation variables φ while ignoring the visibility variables v. Intuitively, optimizing only the orientation φ generates a smooth tile orientation field, which is usually the first step in most mosaic algorithms [Hau01, BDBFG06]. However, unlike most previous algorithms (with a notable exception of [BDBG*08]), our orientation field is generated in a principled manner using a well-understood objective function. Our advantage over [BDBG*08], who also use optimization to get the orientation field, is that our objective function is optimized globally with graph cuts, not locally as in [BDBG*08]. Global optimization of non-convex functions produces better results, as shown in [SZS*08]. Our energy function is non-convex, and, in fact, NP-hard to optimize, as shown later.

Ignoring the visibility labels v_ps, our energy in Equation (4) becomes a function of orientation labels

$$E^o(\varphi) = \sum_{p \in \mathcal{P}} D_p(\varphi_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{pq}(\varphi_p, \varphi_q), \tag{10}$$

where $V_{pq}(\varphi_p, \varphi_q) = w_s \cdot |\varphi_p - \varphi_q|_{mod(\frac{\pi}{2})}$.

Another way of looking at decoupling of variables φ and v is as follows. In the energy in Equation (10), D_p terms are optimized for all pixels p, and V_{pq} terms are optimized for all neighbouring pixel pairs $\{p, q\}$. Therefore, if φ^* optimizes the energy in Equation (10), all $D_p(\varphi_p^*)$ and $V_{pq}(\varphi_p^*, \varphi_q^*)$ terms are expected to be small. In the complete energy in Equation (4), only the D_p and V_{pq} terms for pixels p, q with nonzero v_p, v_q matter. Assigning v_ps while keeping φ_ps fixed to the previously optimized values of φ_p^*s corresponds to picking out a subset of previously optimized $D_p(\varphi_p^*)$ and $V_{pq}(\varphi_p^*, \varphi_q^*)$ terms. Because all $D_p(\varphi_p^*)$ and $V_{pq}(\varphi_p^*, \varphi_q^*)$ terms were small, their subset is also expected to be small.

To minimize the energy in Equation (10), we use the α-expansion algorithm [BVZ01]. According to [BV06], to optimize our energy with α-expansion algorithm, for all $\alpha, \beta, \gamma \in \mathcal{L}$, the smoothness term V_{pq} should satisfy

$$V_{pq}(\beta, \gamma) \leq V_{pq}(\alpha, \gamma) + V_{pq}(\beta, \alpha). \tag{11}$$

We now prove that the energy in Equation (10) satisfies in Equation (11).

Proof. To simplify notation, V denotes V_{pq}. Recall that orientations are in the range of $[0, \frac{\pi}{2})$, where $\frac{\pi}{2}$ is identified with 0. By definition in Equation (9), for any orientation labels α, β,

$$V_{pq}(\alpha, \beta) = \min \left\{ |\alpha - \beta|, \frac{\pi}{2} - |\alpha - \beta| \right\} \leq \frac{\pi}{4}.$$

(a) Starry Night (b) Tiger (c) Portray

Figure 4: *Original images.*

Therefore,

$$V(\alpha,\gamma)+V(\beta,\alpha)=\min\left(|\alpha-\gamma|,\frac{\pi}{2}-|\alpha-\gamma|\right)+\min\left(|\beta-\alpha|,\frac{\pi}{2}-|\beta-\alpha|\right).$$

There are four possible cases

Case 1:

$$V(\alpha,\gamma)+V(\beta,\alpha)=|\alpha-\gamma|+|\beta-\alpha|\geq|\gamma-\beta|\geq V(\gamma,\beta).$$

Case 2:

$$V(\alpha,\gamma)+V(\beta,\alpha)=\left(\frac{\pi}{2}-|\alpha-\gamma|\right)+|\beta-\alpha|.$$

Because $|\beta-\alpha|-|\alpha-\gamma|\geq-|\gamma-\beta|$, we have that

$$V(\alpha,\gamma)+V(\beta,\alpha)\geq\frac{\pi}{2}-|\gamma-\beta|\geq V(\beta,\gamma).$$

Case 3:

$$V(\alpha,\gamma)+V(\beta,\alpha)=|\alpha-\gamma|+\left(\frac{\pi}{2}-|\beta-\alpha|\right),$$

the proof is identical to Case 2.

Case 4:

$$V(\alpha,\gamma)+V(\beta,\alpha)=\left(\frac{\pi}{2}-|\alpha-\gamma|\right)+\left(\frac{\pi}{2}-|\beta-\alpha|\right).$$

Because $\alpha,\beta,\gamma\in[0,\frac{\pi}{2})$, and $|\alpha-\gamma|\geq\frac{\pi}{4}$ and $|\beta-\alpha|\geq\frac{\pi}{4}$, we have that either α is larger than both γ and β or α is smaller than both γ and β. In the first case, $V(\alpha,\gamma)+V(\beta,\alpha)=\pi-2\alpha+\gamma+\beta\geq\gamma+\beta\geq|\gamma-\beta|\geq V(\gamma,\beta)$. In the second case, $V(\alpha,\gamma)+V(\beta,\alpha)=\pi+2\alpha-\gamma-\beta\geq\pi+\max\{\gamma,\beta\}-\max\{\gamma,\beta\}-\gamma-\beta\geq|\gamma-\beta|\geq V(\gamma,\beta)$. □

It is interesting to note that the energy in Equation (10) is NP-hard to optimize. Suppose $\Phi=\{0,\frac{\pi}{6},\frac{\pi}{3}\}$. Let $\alpha,\beta\in\Phi$. Then there are three cases for $V_{pq}(\varphi_p,\varphi_q)$: $V(0,\frac{\pi}{6})=V(\frac{\pi}{6},\frac{\pi}{3})=V(0,\frac{\pi}{3})=\frac{\pi}{6}$. Thus, this V_{pq} is the so called Potts model, which was shown to be NP-hard to optimize in [BVZ01].

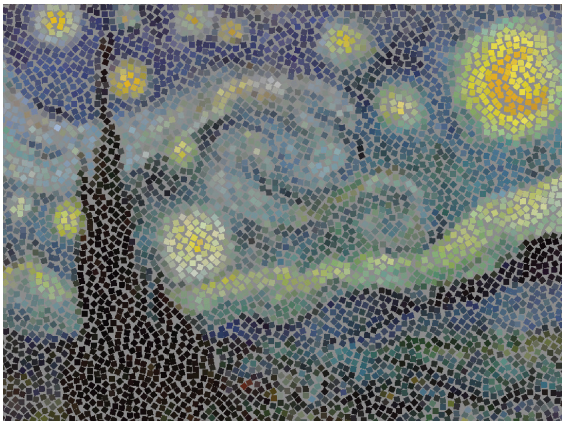
6.2. Optimizing in visibility variables

Let φ^* be the tile orientation field computed in Section 6.1. We now must optimize Equation (4) over the visibility variable v with φ fixed to φ^*. Unfortunately, optimizing v, even if φ is kept fixed, is an NP-hard bin packing problem. Our approach approximates this problem in a two-step manner. First for $i=1,\ldots,m$ we generate in a heuristic manner a number of labellings v^i s.t. $E(v^i,\varphi^*)<\infty$ for all i. Therefore, each (v^i,φ^*) corresponds to a mosaic with no overlapping tiles. Because v^i is generated heuristically, (v^i,φ^*) may not be a good mosaic overall, but might contain a few regions that are good candidates for the final mosaic. We call each (v^i,φ^*) a *candidate mosaic layer*. Orientations of the mosaic corresponding to each candidate mosaic layer are given by φ^*. The final step is to stitch all (v^i,φ^*) into a final mosaic (v^*,φ^*) in such a way that the energy in Equation (4) is minimized.

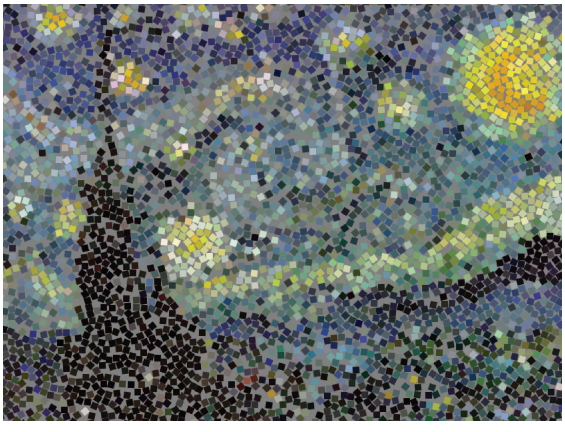
6.2.1. *Generating candidate mosaic layers*

Intuitively, building each layer candidate mosaic layer (v^i,φ^*) given a tile orientation field corresponds to using heuristics for tile placement, as done in most other mosaic generation algorithms [Hau01, BDBG*08], etc. However, instead of generating just one final mosaic, we generate m layers, that are stitched together to form a better final mosaic according to our global energy function in Equation (4).

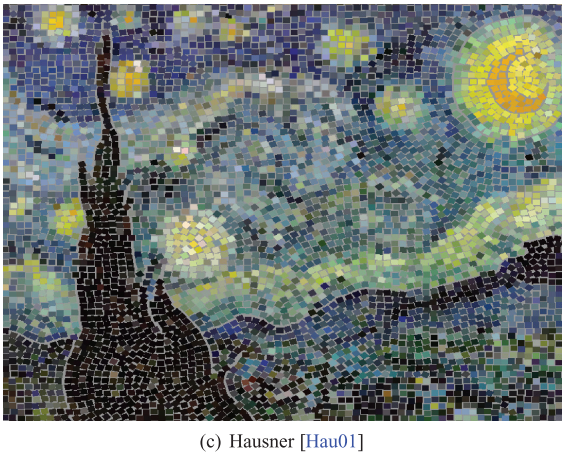
We build each v^i heuristically. Our goal is for a candidate mosaic layer (v^i,φ^*) to have no tile overlap and at the same time contain as many tiles as possible (i.e. for as many pixels as possible, variables v^i_p should be set to 1). We start with $v^i_p=0$ for all pixels p. Next we select a starting pixel s at random. The starting pixel gets assigned $v^i_s=1$. Then we put all the other pixels in $\mathcal{P}$ on an ordered list O of pixels to be processed. Pixels in list O are ordered by their distance to the starting pixel s, with pixels closer to s placed closer to the beginning of O. Let q be the next pixel to be processed. We set



(a) Our Result



(b) Battiato et.al. [BDBG*08]



(c) Hausner [Hau01]

Figure 5: *Starry night results.*

$v_q^i = 1$ if placing tile cantered at q with orientation φ_q^* does not cause tile overlap among the tiles previously placed, that is if $\mathcal{T}(p, \varphi_p^*) \cap \mathcal{T}(q, \varphi_q^*) = \emptyset$ for all pixels p with v_p^i already set to 1. The process of 'growing' v^i stops when the list O is empty. By placing pixels on the list in order of their distance to the start pixel, we are trying to ensure that the next tile placed in v^i is as close as possible to the already placed tiles, thus making the gap space in v^i minimal.

To build a candidate mosaic layer efficiently, when assigning $v_q^i = 1$ for some pixel q, we remove all the pixels in $\mathcal{T}(q, \varphi_q^*)$ from the list O. We also check for tile overlap efficiently. Whether two tiles overlap or not depends only on their relative orientations and the relative distance between the tile centres. We compute a small two-dimensional lookup table, which, based on the relative orientation and centre differences, tells us whether two tiles overlap or not. Thus, the overlap checking can be done in constant time.

We build enough candidate mosaic layers to ensure that for all pixels p, there is an i s.t. $v_p^i = 1$. This gives each pixel p a chance to have a tile cantered at p appear in the final stitching. The candidate mosaic layers built in this heuristic manner are far from an optimal mosaic. Although we try to pack the tiles tightly in each candidate layer, any layer will have regions where the packing is not as tight as possible. In addition, because the quality of tiles (i.e. the data term D_p) is not checked when building the layers, many tiles will be placed on the high intensity edges which causes blurring of the mosaic. Therefore, we need the third step of stitching all layers (v^i, φ^*) together to find a better solution.

6.2.2. *Stitching candidate mosaic layers*

After generating a set of candidate mosaic layers (v^i, φ^*), $i = 1, \ldots, m$, the last step is to stitch them together to form the final mosaic (v^*, φ^*) s.t. the energy $E(v^*, \varphi^*)$ is minimized.

The stitching is performed in a pairwise manner. Let (v^i, φ), (v^j, φ) be two mosaics with equal orientations field. Then their stitching is another mosaic (v', φ) s.t. for all $p \in \mathcal{P}$, $v'_p \in \{v_p^i, v_p^j, 0\}$. This implies that the stitching of two mosaics cannot have any tiles that were not present in either the first or the second mosaic, thus the name 'stitching'.

Our stitching algorithm is iterative. We always have the current mosaic (v^c, φ^*), and stitch it with one of the candidate mosaic layers (v^i, φ), chosen at random. The stitching is performed in such a way as to minimize the energy of the resulting mosaic. We update the current mosaic to the result of the stitching, and repeat. To initialize, v^c is set to a randomly chosen v^i. The process stops when there is no layer s.t. stitching this layer improves the current mosaic (v^c, φ), or when the maximum number of iterations is reached.

We now explain how to find the optimal stitching of the current mosaic (v^c, φ) and the candidate mosaic layer (v^i, φ). Let $\mathcal{P}_c = \{p \in \mathcal{P} | v^c = 1\}$, $\mathcal{P}_i = \{p \in \mathcal{P} | v^i = 1\}$, and let $\mathcal{S} = \mathcal{P}_i \cup \mathcal{P}_c$. Notice that only pixels $p \in \mathcal{S}$ can have their visibility variable v_p change as a result of a stitching. Therefore, optimization is performed only over the variables v_p s.t. in $p \in \mathcal{S}$. With the variables φ fixed to φ^* and optimization

(a) Our Result (b) Battiato et.al. [BDBG*08] (c) Hausner [Hau01]

Figure 6: *Portray results.*

performed only over pixels in $\mathcal{S}$, the energy in Equation (4) reduces to the energy below:

$$E^v(v) = \sum_{p\in\mathcal{S}}(1-v_p) + \sum_{p\in\mathcal{S}} v_p \cdot D_p(\varphi_p^*) + \sum_{\substack{\{p,q\}\in\mathcal{N}\\ \{p,q\}\subset\mathcal{S}}} V_{pq}(v_p, v_q, \varphi_p^*, \varphi_q^*). \quad (12)$$

The energy that we can actually optimize exactly is

$$\tilde{E}^v(v) = \sum_{p\in\mathcal{S}}(1-v_p) + \sum_{p\in\mathcal{S}} v_p \times D_p(\varphi_p^*) + \sum_{\substack{\{p,q\}\in\mathcal{N}\\ p\in\mathcal{P}_c, q\in\mathcal{P}_i}} V_{pq}(v_p, v_q, \varphi_p^*, \varphi_q^*). \quad (13)$$

The difference between the energies E^v in Equation (12) and $\tilde{E}^v$ in Equation (13) is that only the pairwise terms between pixels $p \in \mathcal{P}_c$ and $q \in \mathcal{P}_i$ are present in Equation (13). Pairwise terms between pixels inside $\mathcal{P}_i$ and inside $\mathcal{P}_c$ are missing in $\tilde{E}^v$. We omit this terms to make optimization tractable. However, the absence of these terms is not as important as may seem at first. First of all, because there is no tile overlap in either v^c or v^i, $V_{pq}(v_p, v_q, \varphi_p^*, \varphi_q^*)$ is finite when $p, q \in \mathcal{P}_c$ and when $p, q \in \mathcal{P}_i$. Furthermore, tile orientations φ_p were optimized in the first step of our algorithm. Therefore, we may assume that $V_{pq}(v_p, v_q, \varphi_p^*, \varphi_q^*)$ have low values for most neighbouring pixel pairs. Therefore, it is relatively safe to exclude the pairwise terms V_{pq} when either $p, q \in \mathcal{P}_c$ or $p, q \in \mathcal{P}_i$. However, ignoring V_{pq} when $p \in \mathcal{P}_c$ and $q \in \mathcal{P}_i$ is not safe, because for such p, q the term $V_{pq}(v_p, v_q, \varphi_p^*, \varphi_q^*)$ could be infinite due to overlap of tiles centred at p and q. We do include such 'unsafe' terms in the energy $\tilde{E}^v$. Therefore, the energy $\tilde{E}^v$ is a good approximation to the energy E^v.

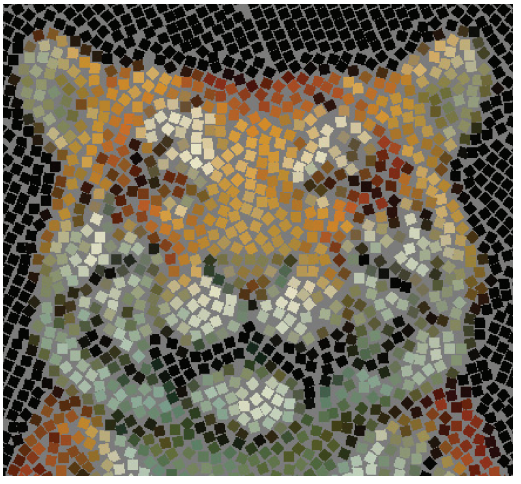
Another explanation for $\tilde{E}^v$ is that it finds a stitching of the current mosaic (v^c, φ^*) and a candidate layer (v^i, φ^*) in such a way that the gap space is optimized and the 'good' tiles (tiles with small D_p) are selected . Orientations at the seam of the stitching are accounted for, but orientations outside of the stitching seam are disregarded, because those are already assumed to be satisfactory because an optimized φ^* is used.

We now explain how to optimize the energy in Equation (13). We use the idea of roof duality from [HHS84]. Let us introduce a new variable t_p for each pixel p, with the following dependence on the visibility variables v_p. If $p \in \mathcal{P}_c$, then $t_p = v_p$. If $p \in \mathcal{P}_i$, then $t_p = 1 - v_p$. In words, for the pixels in $\mathcal{P}_c$, the meaning of variable t_p is the same as the meaning of variable v_p, and the meaning of variable t_p is reversed for $p \in \mathcal{P}_i$. Let $t = \{t_p | p \in \mathcal{P}\}$. We can rewrite the energy in Equation (13) in terms of the new variables

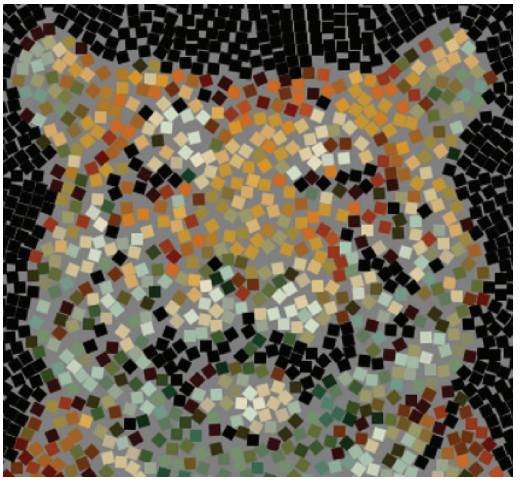
$$E^v(t) = \sum_{p\in\mathcal{S}} D'_p(t_p) + \sum_{\{p,q\}\in\mathcal{N}_{ci}} V'_{pq}(t_p, t_q), \quad (14)$$

where $\mathcal{N}_{ci} = \{\{p,q\} | \{p,q\} \in \mathcal{N}$ and $p \in \mathcal{P}_c, q \in \mathcal{P}_i\}$, and $D'_p(t_p)$, $V'_{pq}(t_p, t_q)$ are defined later:

$$D'_p(t_p) = \begin{cases} 1 + t_p(D_p(\varphi_p^*) - 1) & \text{if } p \in \mathcal{P}_c \\ t_p + (1 - t_p)D_p(\varphi_p^*) & \text{if } p \in \mathcal{P}_i \end{cases} \quad (15)$$



(a) Our Result



(b) Battiato et.al. [BDBG*08]

(c) Hausner [Hau01]

Figure 7: *Tiger results.*

and

$$V'_{pq}(t_p, t_q) = \begin{cases} 0 & \text{if } t_p = 0, t_q = 0 \\ 0 & \text{if } t_p = 0, t_q = 1 \\ V_{pq}(\varphi^*_p, \varphi^*_q, 1, 1) & \text{if } t_p = 1, t_q = 0 \\ 0 & \text{if } t_p = 1, t_q = 1 \end{cases} \tag{16}$$

As shown in [KZ04], a binary energy can be optimized exactly with a graph cut if it is submodular, that is if the pairwise terms satisfy: $V_{sr}(0,1) + V_{sr}(1,0) \geq V_{sr}(0,0) + V_{sr}(1,1)$. Clearly the energy in Equation (14) is submodular, since $V'_{pq}(1,0)$ is either a positive constant or infinite, and all other V'_{pq} are 0. Therefore, the energy in Equation (13) can be optimized exactly with a graph cut. For implementation, we use the max-flow/min-cut algorithm in [BK04].

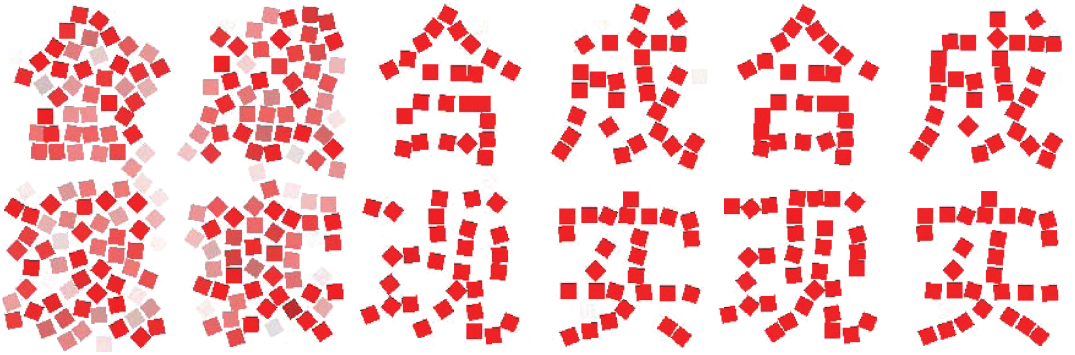
The last detail of the stitching algorithm is as follows. Let $(\tilde{v}, \varphi^*)$ be the mosaic that is the result of stitching the current mosaic (v^c, φ^*) with a candidate mosaic layer (v^i, φ^*), that is $\tilde{v}$ optimizes the energy in Equation (13). Because the energy in Equation 13 is only an approximation to the energy in Equation 12, we check if $E^v(\tilde{v}) < E^v(v^c)$, that is if the stitching using approximate energy is better according to the exact energy. If yes, then we update v^c to $\tilde{v}$. If not, we discard the results of stitching.

Note that [LRR08] use an optimization procedure similar to ours for computing the optical flow from video. They compute many flow fields and 'fuse' or stitch them together.

7. Experimental Results

We now present our experimental results. We perform comparison to Hausner [Hau01] (using code available on the author's web site), and we also compare to Battiato *et al.* [BDBG*08] (using the results provided by the authors). For all the experiments, the parameters were fixed to the following values: $w_e = -50$, $w_v = 20$ and $w_s = 20$.

Figure 4 shows the images used for mosaic generation. Figures 5–7 show the mosaics obtained with our method, Battiato *et al.* and Hausner. Compared to Battiato *et al.*, our mosaic is more spatially coherent. Figures 5(b) and 7(b) have many tiles that 'pop out', that is their colour is incoherent compared to the nearby tiles. This happens because Battiato *et al.* contains heuristic steps that do not discourage tiles to cross strong intensity edges. When a tile crosses a strong intensity edge, its colour is blended and it stands out from the surrounding tiles, as illustrated in Figure 1(c). We perform global optimization and discourage strong edge crossing as a part of the energy function. In addition, compared to that of Battiato *et al.*, the tiles in our mosaics are better aligned to object edges, and therefore they outline the object shapes more accurately. For example, in Figure 5(a), all the stars are clearly delineated, where as in Figure 5(b) only the larger stars have reasonable outlines and the other



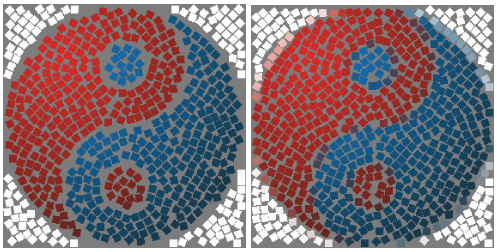
(a) Starting Layer (b) After 64 iterations (c) After 198 iterations

Figure 8: *Progression of mosaic stitching described in Section 6.2.2*

stars are blended with the background. The tiger face that we produce [Figure 7(a)] is more readily recognized as a tiger compared to that of Battiato *et al.* [Figure 7(b)]. This is, again, due to our mosaics delineating objects more clearly and have less tiles that are incoherent with the surrounding tiles.

The method of Battiato *et al.* can be viewed as a simplified version of our algorithm. They also generate tile orientation field (similar to what we do in Section 6.1, but using local optimization). Then they generate the final mosaic heuristically. Thus, their final mosaic is equivalent to our single mosaic layer, and the steps in Sections 6.2.1 and 6.2.2 are not performed. To understand the importance of stitching multiple layers together, consider Figure 8. Figure 8(a) shows our starting layer. This would be similar to the output of Battiato *et al.* The mosaic in Figure 8(a) is clearly inferior, many tiles overlap intensity edges and therefore their colour is blurred. The Chinese character can be barely recognized in this mosaic. Figure 8(b) shows our result after stitching 64 layers together. The results are significantly improved compared to the starting layer in Figure 8(a). The result of stitching 98 layers [Figure 8(c)] still shows a mild improvement over Figure 8(b).

Compared to Hausner [Hau01], our mosaics also have fewer tiles that 'pop out', due to our use of global optimization. Because Hausner needs user interaction, the extent to which the objects are outlined depends solely on the user. For example, in Figure 5(c), the user marked the boundaries around the tree, the largest star, and the border between the sky and the ground regions. These are exactly the objects that are outlined very well in the resulting mosaic. The boundaries of other objects are not emphasized. For example, the medium size star on the right of the castle has tiles with orientations following the tree boundary, not the star boundary. In addition, tile orientations of Hausner's mosaics is visibly more discontinuous compared to our mosaics.



(a) Mosaic with large w_v (b) Mosaic with small w_v

Figure 9: *Mosaics with different* w_v.

Having a global energy function allows us to control the mosaic appearance by tuning parameter values. For example, parameter w_v in Equation (7) determines the importance of the edge avoidance constraint. When w_v is set to 0, tiles are free to cross any edges and the mosaic is blurred [Figure 9(b)]. When w_v is very large, some tiles which are close to the edges will be removed and a large gap space is created in the final mosaic [Figure 9(a)]. A good choice of w_v is in the middle between the two extremes in Figure 9.

To improve the time spent on building and stitching mosaic layers, we pre-compute a table storing the area of overlap for a pair of tiles. This table is indexed by the relative positions and orientations between tiles. With these speed-ups, and considering the significant amount of layer stitching, our running times are reasonable. On the images in Figure 4, of sizes 867×691, 940×1233 and 1200×827, the running times were, respectively, 5, 10 and 9 min. Further speed improvements would help, of course.

8. Conclusion

We formulated the problem of classic mosaic generation in a global optimization framework, and designed an energy

function encoding the properties that lead to perceptually pleasing mosaics. Our global optimization framework offers a more principled approach than previous work, which is mostly based on heuristics.

Note that the artificial mosaics produced by our algorithm are different in appearance from the classic mosaics produced by artists. In these, the tiles are placed at successive bands around the feature edges. This creates discontinuities at the skeleton of the feature curves, as explained in the introduction. The tiles have smoothly varying orientations.

While removing user interaction is an advantage for a naive user, an artist may want more control on the edges to emphasize. If desired, it is easy to include user interaction in our framework, by fixing orientations of some tiles to user specified values. Then the algorithm can proceed as before, but optimizing only over the free variables.

In the future, we intend to extend our approach to rendering mosaics with tiles of different size and shapes. Smaller tiles are needed in image regions which have fine scale details, and larger tiles are sufficient in areas of the image which have coarse features. Therefore, we need to vary the tile size for different regions of the image. It is relatively trivial to include tile size as an additional (third) variable in our optimization framework, favouring the areas with higher spacial frequencies to have smaller tile size. In addition, we also plan to generate time-coherent video mosaics.

References

[BDBFG06] Battiato S., Di Blasi G., Fraxinella G. M., Gall G.: A novel technique for opus vermiculatum mosaic rendering. In *Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2006)* (Bory, Czech Republic, 2006), pp. 133–140.

[BDBG*08] Battiato S., Di Blasi G., Gallo G., Guarnera G. P.: Artificial mosaics by gradient vector flow. In *Proceedings of EuroGraphics 2008* (Crete, Greece, 2008).

[BK04] Boykov Y., Kolmogorov V.: An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9 (2004), 1124–1137.

[BV06] Boykov Y., Veksler O.: *Graph Cuts in Vision and Graphics: Theory and Applications, in Mathematical Models of Computer Vision: The Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, 2006.

[BVZ01] Boykov Y., Veksler O., Zabih R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (2001), 1222–1239.

[Cro84] Crow, F. C.: Summed-area tables for texture mapping. In. *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1984), ACM, pp. 207–212.

[DBG05] Di Blasi G., Gallo G.: Artificial mosaics. *The Visual Computer 21*, 6 (2005), 373–383.

[DBGP05] Di Blasi G., Gallo G., Petralia M.: Puzzle image mosaic. In *Proceedings of IASTED/VIIP 2005* (Benidorm, Spain, 2005).

[DBP05] Di Blasi G., Petralia M.: Fast photomosaic. In *Poster Proceedings of ACM/WSCG 2005* (Bory, Czech Republic, 2005).

[DHJN02] Dobashi J., Haga T., Johan H., Nishita T.: A method for creating mosaic image using voronoi diagrams. In *Proceedings of EuroGraphics 2002* (Saarbrücken, Germany, 2002), pp. 341–348.

[DHvOS00] Deussen O., Hiller S., van Overveld C. W. A. M., Strothotte T.: Floating points: A method for computing stipple drawings *Computer Graphics Forum 19*, 3 (2000), 40–51.

[Elb95] Elber G.: Line art rendering via a coverage of isoparametric curves. *IEEE Transaction on Visualization and Computer Graphics 1*, 3 (1995), 231–239.

[Elb98] Elber G.: Line art illustrations of parametric and implicit forms. *IEEE Transaction on Visualization and Computer Graphics 4*, 1 (1998), 71–81.

[EW03] Elber G., Wolberg G.: Rendering traditional mosaics. *The Visual Computer 19*, 1 (2003), 67–78.

[FDF05] Faustino G. M., De Figueiredo L. H.: Simple adaptive mosaic effects. In *SIBGRAPI '05: Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing* (Washington, DC, USA, 2005), IEEE Computer Society, p. 315.

[Hae90] Haeberli P.: Paint by numbers: Abstract image rappresentation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM, pp. 207–214.

[Hau01] Hausner A.: Simulating decorative mosaics. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 573–580.

[Her01] Hertznmann A.: Paint by relaxation. In *Proceedings of Computer Graphics International 2001* (Hong Kong, China, 2001), pp. 47–54.

[HHD03] Hiller S., Hellwig H., Deussen P.: Beyond stippling methods for distributing objects on the plane. In *Computer Graphics Forum 22*, (2003), 515–522.

[HHS84] Hammer P. L., Hansen P., Simeone B.: Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming 28*, 28 (1984), 121–155.

[Ish03] Ishikawa H.: Exact optimization for Markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 10 (2003), 1333–1336.

[KGFC02] Klein A., Grant T., Finkelstein A., Cohen M. F.: Video mosaics. In *NPAR 2002: Second International Symposium on Non Photorealistic Rendering* (2002), pp. 21–28.

[KMN*99] Kowalski M., Markosian L., Northrup J., Bourdev L., Barzel R., Holden L., Hughes J.: Art-based rendering of fur, grass, and trees. In *SIGGRAPH '99: Proceedings of the 26th annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 433–438.

[KP02] Kim J., Pellacini F.: Jigsaw image mosaics. *ACM Transactions on Graphics 21* (2002), 657–664.

[KZ04] 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.

[LRR08] Lempitsky V., Roth S., Rother C.: Fusionflow: Discrete-continuous optimization for optical flow estimation. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'08)* (Anchorage, Alaska, US, 2008), pp. 1–8.

[LVJ07] Liu Y., Veksler O., Juan O.: Simulating classic mosaics with graph cuts. In *Proceedings of 6th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition* (EZhou, Hubei, China, 2007), pp. 55–70.

[Mei96] Meier B.: Painterly rendering for animation. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 477–484.

[OK08] Orchard J., Kaplan C. S.: Cut-out image mosaics. In *NPAR '08: Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2008), ACM, pp. 79–87.

[Ost99] Ostromoukhov V.: Digital facial engraving. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 417–424.

[PB94] Pnueli Y., Bruckstein A.: Digidurer - a digital engraving system. *The Visual Computer 10* (1994), 277–292.

[PCK09] Pavic D., Ceumern U., Kobbelt L.: Gizmos: Genuine image mosaics with adaptive tiling. *Computer Graphics Forum 28*, 8 (2009), 2244–2254.

[PFS03] Pastor M. O., Freudenberg B., Strothotte T.: Real-time, animated stippling. *IEEE Computer Graphics and Applications 23*, 4 (2003), 62–68.

[Sec02] Secord A.: Weighted voronoi stippling. In *NPAR '02: Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2002), ACM, pp. 37–43.

[SGS05] Schlechtweg S., Germer T., Strothotte T.: Renderbots-multi-agent systems for direct image generation. *Computer Graphics Forum 24*, 2 (2005), 137–148.

[SH97] Silvers R., Hawley M.: *Photomosaics*. Henry Holt and Co., Inc., New York, NY, 1997.

[SLK05] Smith K., Liu Y., Klein A.: Animosaics. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), ACM, pp. 201–208.

[SWHS97] Salisbury M., Wong M., Hughes J., Salesin D.: Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 401–406.

[SZS*08] Szeliski R., Zabih R., Scharstein D., Veksler O., Kolmogorov V., Agarwala A., Tappen M., Rother C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 6 (2008), 1068–1080.

[WS96] Winkenbach G., Salesin D.: Rendering parametric surface in pen and ink. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 469–476.