Volume 0 (1981), Number 0 pp. 1–10 COMPUTER GRAPHICS *forum*

GIzMOs: Genuine Image Mosaics with Adaptive Tiling

Darko Pavić and Ulf Ceumern and Leif Kobbelt

RWTH Aachen University

Abstract

We present a method which splits an input image into a set of tiles. Each tile is then replaced by another image from a large database such that, when viewed from a distance, the original image is reproduced as well as possible. While the general concept of image mosaics is not new, we consider our results as "genuine image mosaics" (or short GIzMOs) in the sense that the images from the database are not modified in any way. This is different from previous work, where the image tiles are usually color shifted or overlaid with the high-frequency content of the input image. Besides the regular alignment of the tiles we propose a greedy approach for adaptive tiling where larger tiles are placed in homogenous image regions. By this we avoid the visual periodicity, which is induced by the equal spacing of the image tiles in the completely regular setting. Our overall system addresses also the cleaning of the image database by removing all unwanted images with no meaningful content. We apply differently sophisticated image descriptors to find the best matching image for each tile. For esthetic and artistic reasons we classify each tile as "feature" or "non-feature" and then apply a suitable image descriptor. In a user study we have verified that our descriptors lead to mosaics that are significantly better recognizable than just taking, e.g., average color values.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation;

1. Introduction

The term *mosaic* is generally used in fine arts and refers to a decoration technique. For example in the interiors of ancient buildings like churches, large patterns or pictures are composed of a number of small fragments or *tiles*. These tiles are usually made of pottery or colored glass.

In fact, a mosaic can be understood as a collection of tiles, where each tile can be an image. It is a kind of artistic recursion. E.g. painters of the impressionist movement like Monet or Renoir used visible brushstrokes (*tiles*) to create the final painting (*mosaic*). We can also find mosaicing technique in the modern art. Chuck Close created portraits based on photographs by first dividing the original photograph into smaller pieces (*tiles*) and then painting each piece separately in order to create the final, photorealistic painting (*mosaic*).

This paper presents a method for creating an *image mosaic*, which is a simple collection and arrangement of so-called *tile images*. When viewing an image mosaic from a close distance, the individual tile images are recognized, but when viewed from a large distance a completely different image, the image mosaic, appears. Such a technique can be used in advertisement in order to emphasize the connotation, e.g., one could imagine creating a logo of a sports brand out of tile images which show different products this brand is producing or players, which are using these products, etc.

Our work in this paper was motivated on the one hand by the versatility of the image mosaics: they can be used for artistic as well as for commercial purposes. On the other hand there is a huge number of images available over the internet, a large resource which can be used for creating high quality image mosaics of arbitrary input images without the need for any kind of image modification or retouch.

The most important properties of our algorithm which distinguish our work from previous approaches are:

- Image mosaics for arbitrary input images are generated by using a huge database of source images downloaded from the internet. Previous approaches have used databases of a few hundreds or a few thousands images. Our database consists of more than a million images. Such a huge

database requires very compact descriptors to be used for image matching.

- We sort out all so-called *void* images in a preprocessing step. These are images which do not contain sufficient visual information in order to contribute to a satisfying final mosaic and have a compromising effect on the esthetic appearance.
- The source images are used without applying any kind of image modification. The final *genuine image mosaic* (we call it short GIzMO) is simply obtained by copying the original images. Although such mosaics were also presented before, in our case the approximation quality of the output mosaics is clearly improved by using polynomial descriptors in connection with our huge database.
- We propose a simple segmentation method for classification of the tile images into "feature" and "non-feature" regions. In other words by this classification the input image is separated into regions with high and low local contrast.
- We use polynomial descriptors for image matching. By varying the polynomial degree we automatically vary the approximation level and hence the visual faithfulness of the image mosaic. When appropriately combined, the different degrees of approximation allow for capturing important image features while preserving the overall artistic, mosaicing effect (see Fig. 1(e)).
- We propose an approach for adaptive tiling which allows for creating a new kind of *irregular* image mosaics (see Fig. 1(f)). We avoid the visual periodicity, which is induced by the equal spacing of the image tiles in the regular mosaics. As a side-effect the approximation quality can be increased while keeping the same number of tile images. In contrary to the previous work, where adaptive tilings are created in a *top-down* manner we present a *bottom-up* approach, which introduces more irregular tilings while better preserving the visually dominant features.

2. Related work

Mosaicing is a popular topic in computer graphics. In computer vision the term *mosaicing* is used for stitching together images or videos [Sze96] in order to create higher resolutions or a wide field of view. Often the term *panorama images* or *panorama mosaicing* is used [BL03]. This kind of mosaicing is not directly related to our approach.

The image mosaicing methods related to our work can be split roughly in two types of approaches. A nice overview of these approaches was recently presented by Battiato et al. [BBFG07]. The one group of approaches considers the problem of how to arrange the individual tiles in the image mosaic. Hausner introduced a method for simulating decorative mosaics [Hau01]. His work aims at arranging tiles of constant color by using a centroidal Voronoi diagram and a distance field derived from user-given contours. Kim and Pellacini have introduced a general framework for creating so-called *Jigsaw image mosaics* (JIM), where arbitrarily

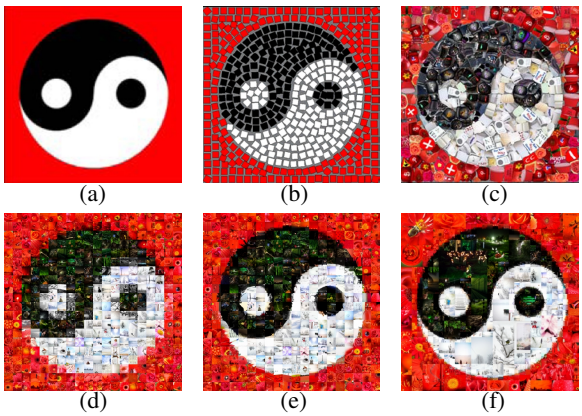


Figure 1: *Comparison of the simulated decorative mosaic [Hau01] (b), JIM [KP02] (c) and GIzMO (d)-(e) for the given input image (a). For all results roughly 400 tiles were used. The images (a)-(c) were taken from [KP02]. The solution when using matching with average colors (d) is still very aliased and blocky. When using an appropriate combination of feature and non-feature descriptors we get the result in (e) which provides sufficient approximation while still preserving the artistic mosaicing effect. Our method goes even further and allows for higher approximation by using an adaptive mosaicing approach which avoids the periodicity of the regular tiling (f).*

shaped container images are tightly packed with other also arbitrarily shaped source images [KP02]. This is achieved by minimizing an appropriate mosaicing energy function. The work of Smith et al. [SLK05] extends the two above approaches [Hau01, KP02] by additionally handling temporal coherence and allowing for creating animated mosaics. Lai et al. [LHM06] have shown how to create decorative mosaics on 3D surfaces. Kim et al. [KEA06] use mosaic representation for video navigation and they are mainly addressing the problem of packing and appropriate labeling of the given video frames in pre-segmented input images. Our method uses simple axis-aligned arrangements of the tiles. However with our simple matching descriptors the visually important feature lines are also well captured (see Figure 1).

The second group of approaches, which are closer in spirit to our work, uses a regular arrangement of tile images and is more focused on how to select them. Mostly these approaches use a kind of color correction in order to preserve the high frequency information in the final image [FR98] or video mosaic [KGFC02]. Finkelstein and Range apply simple color shifting and scaling on the final image mosaic [FR98]. The same color adjustment was used for creating video mosaics out of source videos [KGFC02]. One of the main aspects of our method is that the original colors remain unperturbed in the final image mosaic such that the look of individual tiles is not compromised.

Very recently, Orchard and Kaplan [OK08] introduced

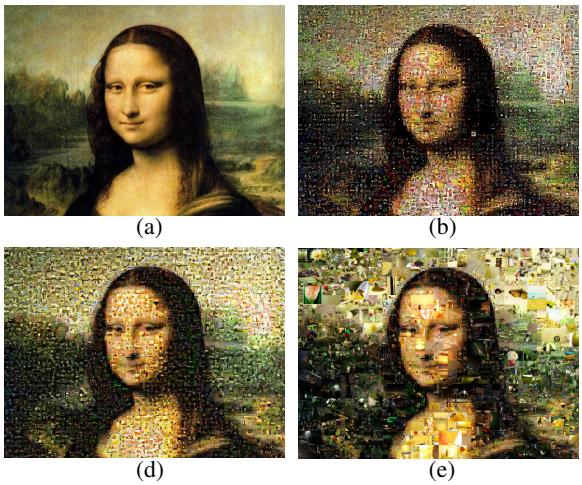


Figure 2: *Comparison of our method with a solution created with the Patchworkr tool [Yav]. (a) input image, (b) Patchworkr solution, (c) our regular solution and (d) our solution with adaptive tiling. Notice that our solutions better preserve the information from the input image. T=2700 in (b) and (c). For (d) only T=1180 tiles were used.*

Cut-Out Image Mosaics where arbitrarily shaped image parts are chosen to assemble the final mosaic. The method presented there is computationally very intensive. Already for very small image databases (several hundreds of images) several hours of computation are needed for creating the final mosaic. We use a huge image database of more than a million images and our method generates image mosaics after just a few minutes of processing time.

Our approach is probably most similar to Photomosaics of Silvers [Sil97]. However a proper comparison with his work is not possible since his technique remains proprietary. As far as we can judge from the examples available online, Silvers uses also a large database of images and creates very good mosaicing results without color correction just like our method. Di Blasi et al. [dBGP06] have also introduced a very practical mosaicing method where no color correction is applied. There each image is partitioned into a 3x3 grid and for each grid cell the average RGB color is computed. This leads to a 27-dimensional image descriptor. We use also a 27-dimensional descriptor, but instead of a piecewise constant function, our descriptor is a quadratic polynomial and therefore more faithfully approximates the image color functions. Di Blasi et al. [dBGP06] first introduced adaptive mosaics where instead of a regular tiling an adaptive quadtree cell structure was used. Our adaptive tiling method is based on a greedy procedure and allows for arbitrarily sized rectangular tiles and hence aviods the visual periodicity in the final image mosaic even more effectively.

There is also a number of tools available on the internet for creating image mosaics. Most of these tools apply color modifications to the source images in order to produce the final result. We have picked one which does not apply color correction: the Patchworkr [Yav], and show a comparison with our method in Figure 2. There we see that our method captures the overall appearance much better (e.g. the eyes or the Mona Lisa's smile).

Our search for the best matching image in a large database can be understood as a variant of content-based image retrieval [VT00], where the content we are searching for is the color distribution in the image. In our case we are using polynomial descriptors [Far02] in order to compactly approximate the color information. For each image or tile we compute a polynomial $f : [0,1]^2 \rightarrow \mathbb{R}^3$ whose components are least squares approximations to the red, green and blue image channels, respectively (see Section 4.1). Alternatively, other approximation or compression schemes to build a descriptor could be based on, e.g., wavelet-schemes [JFS95]. However our experiments showed that the simpler least squares approximant served our purpose well.

In recent years many works like ours were inspired by the fact that nowadays we have a large number of images available on the internet. Two well-known examples for such resources are Google-Images and Flickr.com where billions of images are stored. Torralba et al. [TFF07] have created a huge database of 80 millions tiny images (resolution 32×32) using it mainly for object recognition. Hays and Efros [HE07] use about 2.3 millions of images from Flickr.com for their scene completion method. Lalonde et al. [LHE*07] have created an image-based object library using the online tool LabelMe [RTMF07]. We have used Flickr.com for creating our image database, resulting in a searching space of over one million images.

In the following we will first give an overview of our mosaicing algorithm for a regular tiling of the input image in Section 3. Then we will explain our matching and segmentation descriptors in more detail in Section 4. Finally in Section 5 we will explain how to extend our regular approach to an adaptive one.

3. Algorithm Overview

The input to our method is a *background image I* of arbitrary resolution $H \times W$ and the total number of tiles T to be used. We are using square tiles, a restriction which is done only for convenience and which can be easily relaxed. The tiles are arranged in a $T_h \times T_w$ grid where T_h defines the number of tiles to be placed in each column and T_w defines the number of tiles to be placed in each row of the final mosaic. In order to compute T_h and T_w from T we use the aspect-ratio of the background image I. From $H/W \stackrel{!}{=} T_h/T_w$ we get:

$$T_h = \left\lfloor \sqrt{\frac{T \cdot H}{W}} \right\rfloor \qquad T_w = \left\lfloor \sqrt{\frac{T \cdot W}{H}} \right\rfloor$$

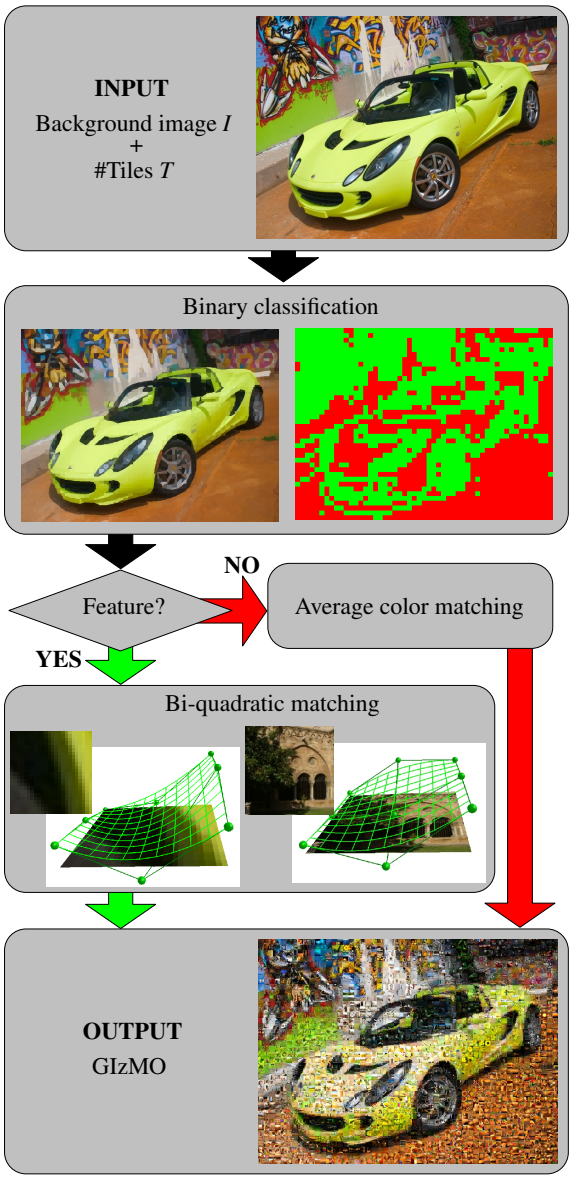


Figure 3: *Workflow of our mosaicing method. For a given input image I and a tiling size T we first apply feature classification based on our segmentation metaphor. Then for each tile either average color matching or bi-quadratic descriptor matching is performed in order to create the final result (here only the polynomial descriptor of the green color channel is shown). The used tiling here was* $T = 2028$.

Now we can either rescale I or simply crop the inner part of I in order to exactly match the tiling resolution. The final image mosaic of I is created by replacing each of the $T_h \cdot T_w$ tile images $I_{i,j}$ (where $i \in \{0,\ldots,T_h-1\}$ and $j \in \{0,\ldots,T_w-1\}$) with the best matching source image from a large database.

Our image database contains over one million square RGB-images. The source images were downloaded from random groups on Flickr.com without specific focus, e.g., landscape, building, portraits, animals, snow, Japan, Niagara falls, etc. All the images were scaled while preserving the original aspect ratio and then cropped to a square of 600×600 pixels. For images having aspect ratio $\frac{W}{H}$ of almost one we took only the central square part of the image. Otherwise, additionally left and right ($\frac{W}{H} > 1$) or top and bottom ($\frac{W}{H} < 1$) square image parts were taken. Besides the RGB color information for each image the polynomial descriptors are also precomputed and stored (see Section 4.1).

The workflow of our mosaicing algorithm is depicted in Figure 3. For each tile, we first compute a feature classification, where intuitively each tile image is classified as a "feature" or as a "non-feature" tile (see Section 4.2). Then for each tile image we are searching for the best matching source image in our image database. Depending on the classification different descriptors are used for the matching.

4. Matching

The core of our algorithm is the searching and matching procedure applied for each tile image $I_{i,j}$. Since we are using a huge database of images, we need simple descriptors which on the one hand have a very compact representation, but on the other hand are sufficiently well capturing the characteristics of the tile image.

4.1. Image Descriptor

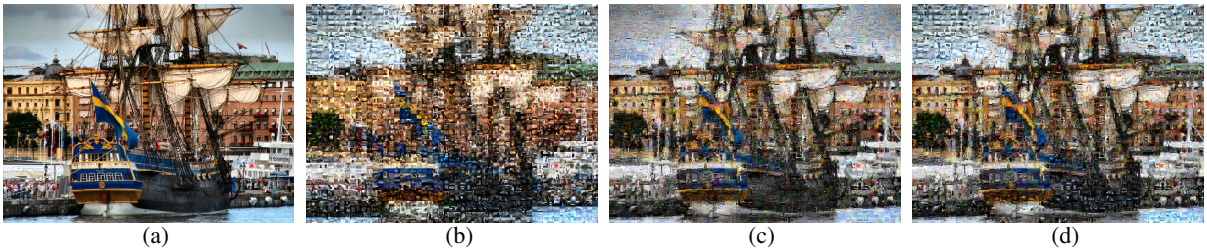
An image descriptor is a k-dimensional vector which describes the content of an image as faithfully as possible. In the k-dimensional descriptor space, we can then define a norm which allows us to measure the similarity of images. In our setup we always use the Euclidian norm in $\mathbb{R}^k$.

Our system uses *polynomial descriptors*, which means we are considering an image (or tile) as a surface patch in RGB-space and then compute a polynomial patch which minimizes the approximation error in the least squares sense. This can be also understood as computing one least squares fit per channel in RGB-space resulting in three polynomial functions. We have also tested our method in other color spaces like L*a*b*, which is adapted to human color perception [KP92], but this did not result in a visually perceivable improvement of the results.

Let I be a $m \times m$ square image with RGB-pixels $I(i,j) \in R^3$. A polynomial descriptor D_n of bi-degree n is represented in tensor-product Bézier form as

$$D_n(u,v) = \sum_{p=0}^{n} \sum_{q=0}^{n} \mathbf{b}_{pq} B_p^n(u) B_q^n(v).$$

and the corresponding descriptor vector is given by the $k =$



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

Figure 4: *Solutions when using different descriptors. Given an input image (a), we generate a GIzMO either by using matching with average colors (b), which is still very aliased and blocky or by using higher order descriptors which creates much better approximation (c). Finally, an appropriate combination of these two solutions that takes into account feature information provides good approximation in high contrast regions while still generating the artistic mosaicing effect in homogenous regions as shown in (d). In all three GIzMOs we have used $T = 2120$ tiles.*

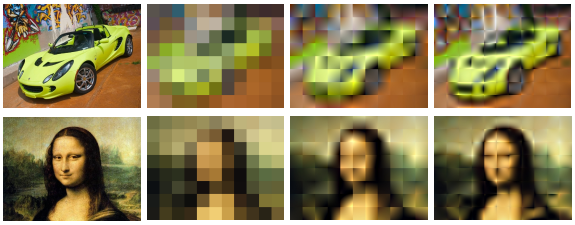


Figure 5: *The approximation quality of our image descriptors on two examples. From left to right: original image, constant, bi-linear and bi-quadratic approximation. In all cases $8 \cdot 11$ tiles were used.*

$3(n+1)^2$ coordinates of the Bézier coefficients. The descriptor for an arbitrary sub-image $[a,b] \times [c,d]$ of the image I is computed by minimizing the quadratic functional

$$E(D_n) = \sum_{i=a}^{b} \sum_{j=c}^{d} \|D_n(u,v) - I(i,j)\|^2,$$

$$u := \frac{i-a}{b-a}, \quad v := \frac{j-c}{d-c}.$$

In the case of bi-quadratic polynomials ($n = 2$), the normal equation for this least squares problem consists of three copies of the same 9×9 linear system with different right-hand sides

$$\mathbf{M}\mathbf{b} = \mathbf{r}$$

where the entries of $\mathbf{M}$ and $\mathbf{r}$ are of the form

$$m_{3p+q,3p'+q'} = \sum_{i=a}^{b} \sum_{j=c}^{d} B_p^2(u) B_q^2(v) B_{p'}^2(u) B_{q'}^2(v)$$

and

$$r_{3p+q} = \sum_{i=a}^{b} \sum_{j=c}^{d} B_p^2(u) B_q^2(v) I(i,j)$$

respectively. Notice that the entries of $\mathbf{M}$ only depend on the resolution $(b-a) \times (d-c)$ of the sub-image. Hence $\mathbf{M}$ is pre-computed once and can be reused for every tile that has the same size. Since we still have to evaluate the right hand side for every image and solve the 9×9 system, we precompute the descriptors offline for all images in the database and only generate the descriptors for the tiles on the fly.

The polynomial degree n of the descriptor D_n determines its approximation power and hence D_n can capture more and more features of an image with increasing n. In our system we compute descriptors of degree $n = 0$ (=average color, $k = 3$), $n = 1$ ($\approx$ color gradient, $k = 12$), and $n = 2$ ($\approx$ edge orientation and characteristics, $k = 27$). For higher degrees we did not observe any visual improvements. In Fig.5 we visualize the approximation power of the polynomial descriptors. The higher the polynomial degree the higher is the approximation quality. In Fig.4 we show some GIzMOs with different descriptors in comparison.

4.2. Feature classification

When creating image mosaics based on polynomial descriptors we observe that with increasing degree n the reproductions of the input images become more and more faithful. While this is a positive result from the approximation point of view, we find that from the artistic point of view it is not desired because in fairly homogeneous regions of the input image, the faithful reproduction implies that images with very little detail and contrast are picked. By this we lose the characteristic appearance of image mosaics (see Fig. 4(c)).

On the other hand, by using constant or bi-linear descriptors, we can preserve the mosaic appearance but we lose many image features due to the quite limited descriptive power of the low-degree polynomials (see Fig. 4(b)).

In order to obtain the best of both worlds, we run a classification procedure on each tile. If a tile is classified as "feature" we use bi-quadratic descriptors in order to find the best matching image which recovers the local image detail. If a tile is classified as non-feature, we switch to constant or bi-linear descriptors. The reduced approximation is acceptable

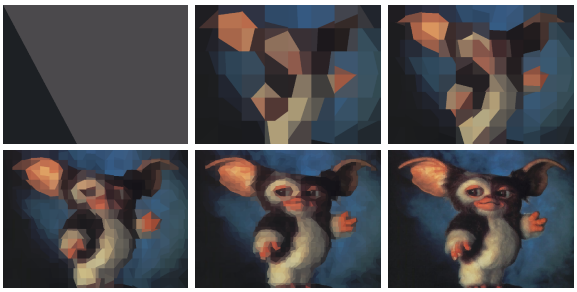


Figure 6: *From top left to bottom right visualized feature classifier for roughly 1, 50, 100, 300, 1000 and 5000 tiles. This kind of visualization provides a nice non-photorealistic effect as a side-effect.*

since the non-feature tiles do not contain significant local detail anyway (see Fig. 4(d)).

A very simple classifier could be based on the difference between the bi-quadratic descriptor D_2 and the (degree-elevated) bi-linear descriptor D_1. However we found that a more sophisticated edge-detection classifier leads to superior results (and in addition generates very nice intermediate non-photorealistic effect as shown in Fig. 6).

The motivation for this classifier is to be able to rate the significance of an edge (feature) in the image independently from its orientation and location within a tile. Hence our classifier computes a line which splits the tile into two regions and computes the average color in each. The difference between these two colors measures the significance and is used for the feature/non-feature classification.

Let $L : \mathbf{n}^T\mathbf{x} = d$ be a line in the index space of the image I. Then the two average colors C_+ and C_- are defined as follows:

$$C_+ = \frac{1}{n_+} \sum_{\mathbf{n}^T(i\ j)^T \geq d} I(i,j)$$

and

$$C_- = \frac{1}{n_-} \sum_{\mathbf{n}^T(i\ j)^T < d} I(i,j)$$

where n_+ and n_- are the number of pixels on both sides of the line respectively. The approximation error is defined as

$$E(I,L) = \sum_{\mathbf{n}^T(i\ j)^T \geq d} \|I(i,j) - C_+\|^2 + \sum_{\mathbf{n}^T(i\ j)^T < d} \|I(i,j) - C_-\|^2$$

Our classifier tries to find the line L for which $E(I,L)$ is minimal and then uses $\|C_+ - C_-\|^2$ as a feature measure by comparing it to some user-defined threshold θ.

Due to the non-smoothness of the function I, the optimal separating line is difficult to compute. Hence we simply define a finite set of candidate lines and explicitly check $E(I,L)$ for each candidate separately.

Let $\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}$ be a set of samples equidistantly distributed along the boundary of the image I with one sample placed at every corner. Our set of candidate lines is defined as the set of $\frac{n}{2}(n-1)$ lines connecting any pair of samples $\mathbf{p}_i$ and $\mathbf{p}_j$.

For every sample $\mathbf{p}_i$ the lines connecting it to the other samples define a triangle fan which covers the entire image. If we compute the sum of the pixel colors within each triangle plus the number of pixels then we can easily find the optimal separating line starting at $\mathbf{p}_i$. By repeating this procedure for each sample $\mathbf{p}_i$ we eventually find the optimal separaring line.

Notice that we intentionally do not take the relative image areas n_+ and n_i into account when computing the optimal separating line. This is important since we want to capture global image features, hence also small local parts of the global features should be preserved.

4.3. Image Mosaic Generation

Figure 3 depicts the workflow of our image mosaicing technique. The binary classification is based on the classifier described in Section 4.2. For each of the $T_h \cdot T_w$ tiles in the input background image I we compute the corresponding descriptor and then find the closest descriptor in the image database. In practice we can allow the user to control the feature threshold θ manually and so create the subjectively best solution. First, by computing both, feature and non-feature descriptors for each tile we generate two GIzMOs. Then the user can interactively adjust the feature threshold θ and so generate from her point of view the *best looking* final GIzMO.

To make the mosaic generation fast, we use the efficient Approximate Nearest Neighbor (ANN) library [MA06] which quickly finds the nearest descriptors in 3-, 12-, or 27-dimensional space (corresponding to constant, bi-linear, or bi-quadratic descriptors).

In order to prevent the repetition of the same image tile in homogeneous regions we have tried error diffusion techniques known from classical dithering schemes like Floyd-Steinberg. The effect of error diffusion, however, is very minor because due to our large image database the average color of the tile usually matches the retrieved image very well. This is why we decided to search for the n closest neighbors and simply pick the best candidate which has not been used yet.

All images are treated the same when downloaded from the internet although for the generation of image mosaics not all of them are appropriate. Those images which do not contain significant visual information are visually disturbing since they strongly reduce the mosaicing effect. We call these images simply *void* images and remove them from the database in a preprocessing step. We have tested a number

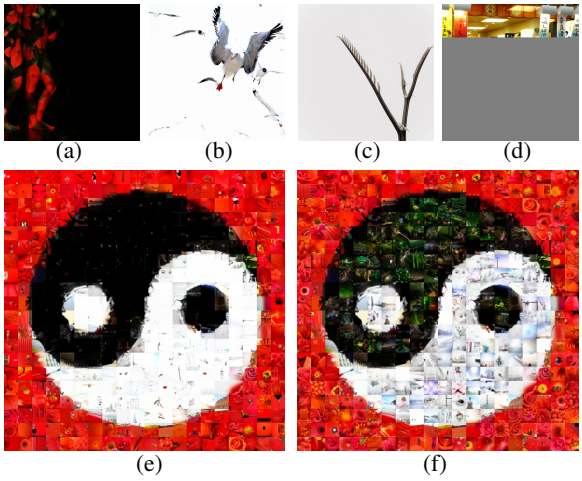


Figure 7: *Void Images. A number of images with too low entropy are shown here: (a) too dark, (b) too bright, (c) not enough details or (d) broken (canceled download). In the lower row we see the difference beween a result created with (e) and without (f) void images (for both T=400). Notice the much higher variability in the homogenous areas in (f).*

of different image statistics [GW01] like statistical moments or image uniformity, and among those finally decided to use average entropy of the gray-level histogram of each image. Since entropy is a measure of variability it is exactly the measure we need. In our current implementation all images having entropy below 6.0 were removed from the database. This threshold was chosen empirically. In Figure 7 a number of void images is shown and also a comparison between a GIzMO with (e) and without (f) void images.

5. Adaptive Tiling

So far we have considered only the generation of image mosaics with a regular tiling pattern. In order to avoid the visual periodicity, which is induced by the equal spacing of the image tiles, we can apply a greedy coarsification procedure, which merges neighboring tiles into one and then replaces a whole group of small tiles with one image from the database (see Fig. 1, 2,9,11 for some examples). As a side effect, we can build an image mosaic by fewer images often without a significant decrease in quality.

Our adaptive tiling idea originates from the area of incremental mesh decimation [GGK02], which is a bottom-up approach: starting from a fine input mesh a coarse output mesh is generated. During the decimation one is in general interested in preserving feature regions: in the case of meshes these are edge or corner features. The parallels to our case of tile coarsification are obvious: we want to preserve the feature tiles and to decimate the non-feature tiles. In contrast to top-down adaptive mosaicing approaches as the one proposed by Di Blasi et al. [dBGP06], our adaptive tiling introduces more irregularity by merging neighboring non-feature tiles while preserving the visually dominant features. As a consequence our coarse tiles are not restricted to dyadic locations and sizes.

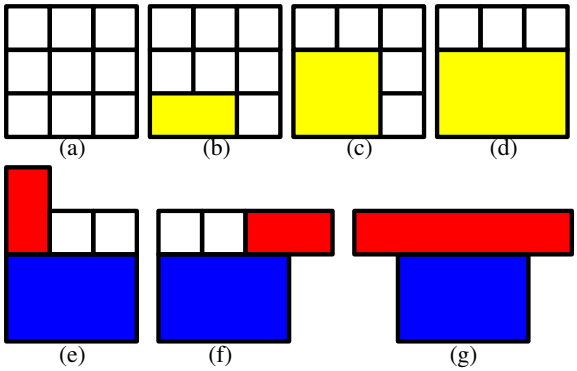


Figure 8: *Adaptive tiling. Starting from the regular grid in (a) a number of successive merging operations are shown (b)-(d) where the newly merged tiles are yellow. The lower row shows some examples where another collapse of the large blue tile in the upper direction is not possible because of the red tiles.*

We start with a regular grid of $T_h \times T_w$ square tiles and do the feature classification first. Then, each pair of horizontally or vertically neighboring non-feature tiles is an initial candidate for the next merging operation. We rate the visual quality of each candidate by computing the approximation error of the best fitting image from the database after the merge. Since only non-feature tiles are merged we use the constant descriptor as similarity metric. Finally, we run a simple greedy procedure which applies the best horizontal and vertical merge operation in each step. Notice that we possibly compare rectangular parts of the input image with square images from the database. This does not cause a large bias because in the non-feature regions the color variation is naturally very low.

During the greedy coarsification, neighboring tiles are allowed to be merged only if the resulting tile is still rectangular. This implies a number of constraints on the size and configuration of the tiles that qualify for a merge operation. Fig. 8 shows a few examples when merging is not permitted. In general, a merge between a large tile and one or several (smaller) neigboring tiles is not permitted if the smaller tiles not all have the same depth (cf. Fig. 8(e)) or the sum of their lengths does not add up to the larger tile's side lengths (cf. Fig. 8(f,g)). For aesthetical reasons we further restrict the maximum aspect ratio of the tiles to be maximally 2:1. Optionally we may also restrict the maximal tile size.

5.1. Adaptive GIzMO Generation

The image mosaic generation is now done just as in the previous regular case (see Section 4.3). The only difference is that for large tiles, which are in general rectangular and not

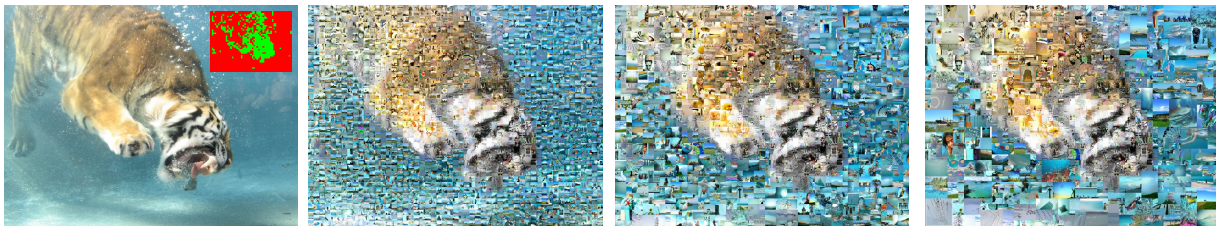


Figure 9: *Adaptive Tiling. From left to right: input image (the visualization of the binary classification shown as inset), a regular GIzMO (T=2120) and then two adaptive GIzMOs created by restricting the max. tile size to be two (T=1208) or five (T=979) times the minimal tile size, respectively. The approximation of important features remains unperturbed by the adaptive tiling process while the number of tiles is reduced considerably.*

square, we now crop the inner part of the best (square) image match from the database. The larger the ratio of the tile the larger is the error we are making because the average color of the cropped region might differ from the average color of the entire image. Since the large tiles are non-feature regions anyway this error is in general negligible.

6. Results

All our experiments were performed on a AMD64 2,2GHz PC with 2GB RAM. Computing GIzMOs never took more than a few minutes for high-resolution input images and high tiling resolutions of several thousand tiles. In order to speed-up the computation process we cache all precomputed descriptor data from the database in the main memory. This caching process takes about 1 min.

Fig. 9 and 11 show that our method is able to handle all kinds of input images. Although we restrict our method to axis-aligned tiling and do not apply any color correction, still all important features are sufficiently well captured. Since we are able to appropriately adjust the degree of noisiness by changing the feature threshold, we can easily vary between the pure artistic and pure approximative image mosaics. For more insight into this please see the accompanying video.

6.1. User Study

In order to evaluate the quality of the proposed shape descriptors we have made a user study with 62 testing persons (31 females and 31 males). During the test each subject was shown the same 15 mosaicing examples, but with randomly chosen descriptors: either constant or bi-linear or bi-quadratic Bézier patches, in each case 5 examples. No binary classification (Section 4.2) was done.

In Fig. 2,3,4,9,11 some of the examples used in the study are shown. We have created for each example and for each metric image mosaics in 11 different tiling sizes $T = 500, 700, ..., 2500$, resulting in a total of $15 \cdot 11 \cdot 3 = 495$ GIzMOs used for the study.

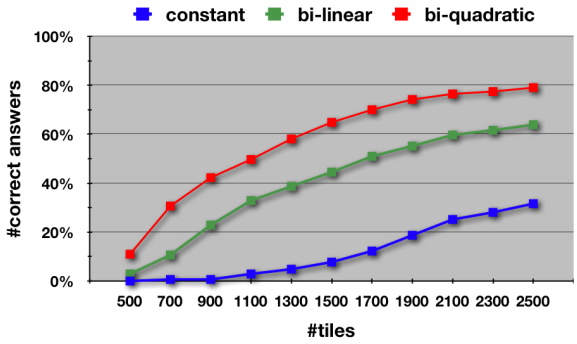


Figure 10: *This chart shows the number of correct answers given during our study. The curves show significant domination of the bi-quadratic descriptor especially for the small tiling resolutions.*

The procedure of the test was as follows: First, each subject was asked a question like "Which animal/famous person/object is shown in the mosaic?". Then the mosaics were shown with increasing resolution, each resolution for the duration of 2*s*. The used metric, the time as well as the tiling resolution the testing person needed in order to answer the given question were logged. The same question had to be answered also for the original image in order to be able to differentiate between wrong cognition and wrong knowledge.

descriptor	correct	wrong	timeout	best
constant	32%	9%	59%	6%
bi-linear	63%	15%	22%	14%
bi-quadratic	79%	8%	13%	80%

Table 1: *Statistics of our user study for the three polynomial descriptors.*

The evaluation of this study is shown in the Table 1. We have used the same number of images for each descriptor (i.e. 62 subjects * 15 examples / 3 metrics = 310 images). The columns "correct","wrong" and "timeout" show the percentage of images which were recognized correctly, wrongly or not at all, respectively. In Figure 10 we see the number

of given correct answers against the needed number of tiles. The higher degrees polynomials obviously lead to better perception, i.e. not only that more subjects were able to answer the given question, but also they needed significantly less tiles (time) in average for the answer.

A very interesting result is shown in the last column of the Table 1. After each example we have shown the original image as well as all the three final ($T = 2500$ tiles) mosaicing solutions asking for the subjectively *best* solution. The majority of the subjects (80%) preferred the bi-quadratic degree descriptors and explained that they were judging the best approximation quality, i.e., the best perceptibility. However the subjects who have chosen the lower degree descriptors (20%) explained their decision by saying that they were judging the artistic impression, i.e., the more pronounced mosaicing effect. This is the reason why we introduced our binary classification in order to satisfy both, high approximation quality as well as artistic mosaicing appearance.

7. Conclusion

In this paper we have presented a method for creating *genuine image mosaics* (GIzMOs), axis-aligned arrangements of images without applying any kind of color correction. The source images were downloaded from the internet and used to create our image database. By using only appropriate images from the database we improve the mosaicing quality of the final results. For arbitrary input background images we are able to create GIzMOs in only a few minutes even for high tiling resolutions and this is in particular due to our very simple shape descriptors which were used for the matching. In order to achieve high approximation quality and to preserve the mosaicing effect in our results, we have applied binary classification of the tiles, distinguishing between low and high contrast regions and using different strategies for both types. Our mosaicing method was evaluated in the context of a user study, which has proven the suitability of our shape descriptors as well as the fact that image mosaics are judged either with respect to the approximation quality or with respect to the artistic appearance. Finally we have introduced a simple greedy approach for creating adaptive tilings out of regular ones and hence allowing for reducing the number of tiles in the mosaic without losing the approximation power of the important parts in the scene.

References

[BBFG07] BATTIATO S., BLASSI G. D., FARINELLA G. M., GALLO G.: Digital mosaic frameworks - an overview. In *Computer Graphics Forum* (2007), pp. 794–812.

[BL03] BROWN M., LOWE D. G.: Recognising panoramas. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision* (2003), p. 1218.

[dBGP06] DI BLASI G., GALLO G., PETRALIA M. P.: Smart ideas for photomosaic rendering. In *Eurographics Italian Chapter Conference* (2006), pp. 267–272.

[Far02] FARIN G.: *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., 2002.

[FR98] FINKELSTEIN A., RANGE M.: Image mosaics. In *EP '98/RIDT '98: Proceedings of the 7th International Conference on Electronic Publishing, Held Jointly with the 4th International Conference on Raster Imaging and Digital Typography* (1998), pp. 11–22.

[GGK02] GOTSMAN C., GUMHOLD S., KOBBELT L.: Simplification and compression of 3d-meshes. In *Tutorials on Multiresolution in Geometric Modelling*. Springer, 2002.

[GW01] GONZALEZ R. C., WOODS R. E.: *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

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

[HE07] HAYS J., EFROS A. A.: Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2007).

[JFS95] JACOBS C. E., FINKELSTEIN A., SALESIN D. H.: Fast multiresolution image querying. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 277–286.

[KEA06] KIM K., ESSA I., ABOWD G. D.: Interactive mosaic generation for video navigation. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia* (New York, NY, USA, 2006), ACM, pp. 655–658.

[KGFC02] KLEIN A. W., GRANT T., FINKELSTEIN A., COHEN M. F.: Video mosaics. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002).

[KP92] KASSON J. M., PLOUFFE W.: An analysis of selected computer interchange color spaces. *ACM Trans. Graph. 11*, 4 (1992), 373–405.

[KP02] KIM J., PELLACINI F.: Jigsaw image mosaics. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002).

[LHE*07] LALONDE J.-F., HOIEM D., EFROS A. A., ROTHER C., WINN J., CRIMINISI A.: Photo clip art. *ACM Transactions on Graphics (SIGGRAPH 2007) 26*, 3 (2007).

[LHM06] LAI Y.-K., HU S.-M., MARTIN R. R.: Surface mosaics. *Vis. Comput. 22*, 9 (2006), 604–611.

[MA06] MOUNT D. D., ARYA S.: Approximate nearest neighbor library, 2006.

[OK08] ORCHARD J., KAPLAN C. S.: Cut-out image mosaics. In *NPAR '08* (2008), pp. 79–87.

[RTMF07] RUSSELL B. C., TORRALBA A., MURPHY K. P., FREEMAN W. T.: Labelme: a database and web-based tool for image annotation. *Int. Journal of Computer Vision* (2007).

[Sil97] SILVERS R.: *Photomosaics*. Henry Holt and Co., Inc., 1997.

[SLK05] SMITH K., LIU Y., KLEIN A.: Animosaics. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 201–208.

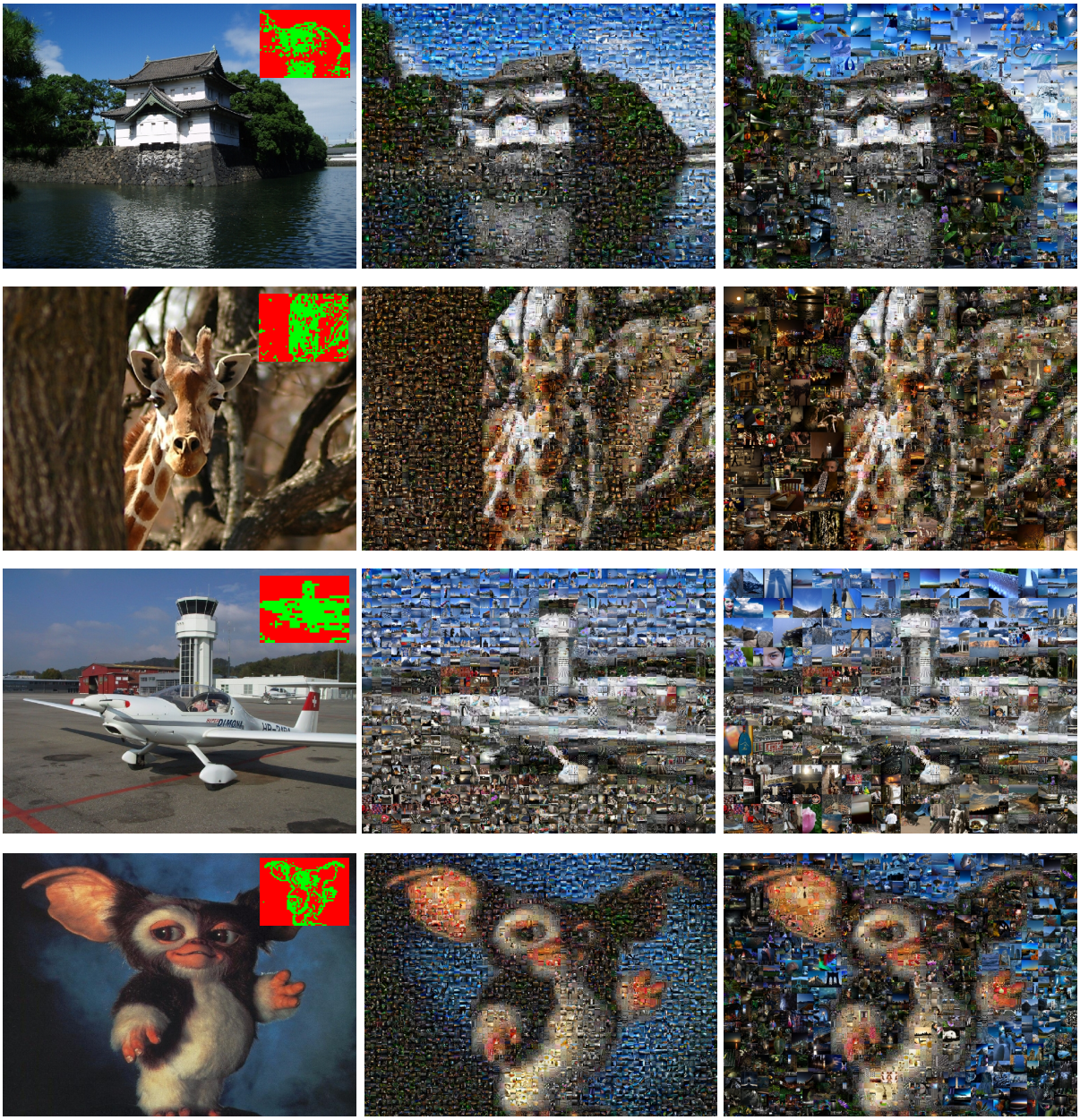


Figure 11: *More GIzMOs. Here we see in each row from left to right the original image (binary classification shown as inset), a regular GIzMO and an adaptive GIzMO. The tilings were: building regular T=2120 and adaptive T=1146, giraffe regular T=2120 and adaptive T=1274, airplane regular T=972 and adaptive T=571, gizmo regular T=2961 and adaptive T=1469.*

[Sze96] SZELISKI R.: Video mosaics for virtual environments. *IEEE Computer Graphics and Applications 16*, 2 (1996), 22–30.

[TFF07] TORRALBA A., FERGUS R., FREEMAN W. T.: *Tiny Images*. Tech. rep., Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, 2007.

[VT00] VELTKAMP R., TANASE M.: Content-based image retrieval systems: A survey, 2000.

[Yav] YAVARI A.: Patchworkr: www.fumpf.de/patchwork/.