

Example-Based Style Synthesis

Iddo Drori

Daniel Cohen-Or

Hezy Yeshurun

School of Computer Science
Tel Aviv University
Tel Aviv, Israel, 69978

Abstract

We introduce an example-based synthesis technique that extrapolates novel styles for a given input image. The technique is based on separating the style and content of image fragments. Given an image with a new style and content, it is first adaptively partitioned into fragments. Stitching together novel fragments produces a coherent image in a new style for a given content. The aggregate of synthesized fragments approximates a globally non-linear model with a set of locally linear models. We show the result of our method for various artistic, sketch, and texture filters and painterly styles applied to different image content classes.

1. Introduction

Images and paintings can be regarded as a composition of content and style. An illustrative example is shown in Figure 1, where the four images on the top left are examples consisting of two pairs of images which have the same “content” and two pairs with the same “style”. Let \mathbf{p}_i represent the different underlying content of each row $i = 1, \dots, m$, and f_j the different style operator of each column $j = 1, \dots, n$. Then the $m \times n$ matrix of images is $\mathbf{A} = (a_{ij}) = f_j(\mathbf{p}_i)$.

Problem statement: Given the sub-matrix of images $\mathbf{A}(1 : m - 1, 1 : n - 1)$ as a small training set, and the input image $\mathbf{A}(m, n)$ shown on the lower right of Figure 1, which has a different content and a different style, the task of the algorithm is to complete the remaining images $\mathbf{A}(m, 1 : n - 1)$ and $\mathbf{A}(1 : m - 1, n)$. Loosely speaking we would like to synthesize a new image that contains the same content but with novel styles. In particular, the goal is to complete the image matrix and synthesize images with the same content as the input image, with the styles of the images of the training set.

Synthesizing images in various styles by example is related to the problem known as *image analogies* [20]: given

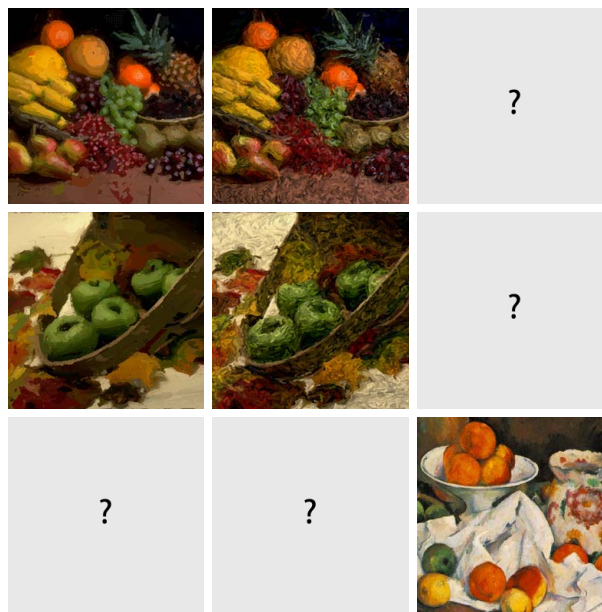


Figure 1. A training set of painterly rendered images and an input from a Still Life by Paul Cezanne form an image matrix. Each column consists of various styles of the same image, and each row results from applying the same style to different images. The goal is to synthesize the remaining images.

training images $f_1(\mathbf{p}_1), f_2(\mathbf{p}_1)$ in different styles, and another example image in one of the known styles, $f_1(\mathbf{p}_2)$, the goal is to synthesize the example in the other known style $f_2(\mathbf{p}_2)$. In contrast, we deal with a two-factor problem, since the image $f_n(\mathbf{p}_n)$ has both unknown content $\mathbf{p}_n \neq \mathbf{p}_1, \mathbf{p}_2$ and unknown style $f_n \neq f_1, f_2$. Notice that once any of the images in the image matrix is synthesized, the remaining images can then be generated by analogy. In image analogies [20], the training set is used as a look-up table, which shows explicitly the mapping between styles of a given pixel in a given local content. Here we take a dif-

ferent approach: the training set is used to generate a model that factors style and content, and then novel images are synthesized based on the computed model. Factorization of style and content has been applied successfully to aligned images [17, 29]. However, in our setting there is no natural correspondence between the training images of different content, as shown in the rows of Figure 1, and no natural correspondence with the given input image, and thus the images cannot be aligned. Moreover, the input image has a different content, and its style is not necessarily present in any of the examples in the training set.

Separating style and content in the general case is a difficult problem. It has been successfully applied to factors such as illumination, that can be approximated by a linear subspace [4]. The method introduced in this paper deals with models which are globally non-linear. The idea is to approximate a globally non-linear model by multiple locally linear models. The input image is decomposed into a set of overlapping fragments of various sizes, and fragments of the training images are considered at multiple positions, scales, and orientations. At the fragment level, similar fragments are approximately locally linear. Composing the local fragments back into a full image, approximates the globally non-linear model by locally linear models. This property is maintained by an adaptive choice of fragments, as will be described in detail below.

Building upon recent example-based image synthesis techniques [13, 15, 20, 22, 28, 29], our approach is to adaptively decompose the input image into fragments, and then synthesize a coherent image by stitching together synthesized fragments that follow multiple constraints. Taking an adaptive approach captures features at multiple scales and avoids drastic blocking artifacts. It should be emphasized that this work takes into account image content that includes both stationary and local regions such as textures, as well as non-stationary and global information present in general images. However, our approach is limited to capturing styles that are local, and at the fragment level, the model is simple and fixed.

2. Previous work

Many operations ranging from low-level vision tasks to high-end graphics applications have been efficiently performed based on examples [3, 6, 8, 9, 10, 15, 16, 20, 21, 27, 29].

The idea of defining and factoring image style and content using a bilinear model was introduced to computer vision by Freeman and Tenenbaum [17]. Given a training set of aligned face images of different people under varying illumination, Tenenbaum and Freeman [29] refer to the identity of a face as content whereas the illumination is considered as style. Style and content are factored pixel-wise by

fitting a symmetric bilinear model to the training images. Combinations of style and content are synthesized from a new face image under novel illumination by changing the estimated parameters of the new image.

Given a set of parameters describing facial expressions such as degree of happiness or surprise, along with a relative motion field of the face, Beymer *et al.* [5, 6] use a regression function that estimates a mapping from parameters to motion for synthesizing novel facial expressions from a set of labelled training images.

Rather than defining individual filters by hand, Hertzmann *et al.* [20] automatically studied a mapping of spatially local filters from image pairs in pixel-wise correspondence, one a filtered version of the other. A new target image is then filtered analogously to the training images. Pixels are rendered by comparing their neighborhood, and those of a coarser level in a multi-resolution pyramid, to neighborhoods of a training image pair. Considering the original image as output and taking its segmentation map as input allows the texture to be painted by numbers [18]. A new image that is composed of the various textures is synthesized by painting a new segmentation map. Swapping the output segmentation map with the original input image results in example-based segmentation [8]. In our case, we cannot only approximate a common representation for images by blurring and median filtering and then proceed by analogy, as this results in the loss of image detail.

Freeman *et al.* [15, 16] derive a model used for performing super-resolution by example. The technique is based on examining many pairs of high-resolution and low-resolution versions of image patches from several training images. Baker and Kanade [3] apply this technique restrictively, to a class of face images. Given a new low-resolution face image its corresponding high-resolution image is inferred by re-using the existing mapping between individual low-resolution and high-resolution face patches.

The term *image quilting* [13] was coined for describing the process of synthesizing a new image by stitching together blocks of existing images. The technique focuses on handling boundaries between blocks and is applied to texture synthesis. It enforces a constraint that the error between overlapping blocks is minimal, and blocks are stitched along a minimum cost path [11] that is computed by dynamic programming. The results depend on the size of a block, which varies according to the texture properties. Hierarchical pattern mapping [28] takes into account that patterns in a texture are often in many different sizes. A 3D surface is progressively covered by texture patches of various shapes and sizes. Starting from a large patch, it is split into smaller ones based on the texture fitting error with already textured neighborhoods. By adding the constraint that the synthesized texture match an example image, Ashikhmin [2] presented texture transfer, in which an exam-

ple image is rendered with a training texture. This technique was extended to color transfer [30], a special case of image analogies, by matching local image statistics. In our setting, we cannot simply transfer color and texture from one style to another, as applying properties of one style over the other could result in artifacts.

3. Image manifolds

A general mathematical perspective for describing the variability of images is to regard an image, which is represented by a vector of pixel intensities, as a point in an abstract image space, and then to characterize a set of images by a manifold in this high-dimensional image space [26]. The statistics of natural images are correlated [14] and far from random. Recently, several researchers [4, 23] have independently shown that irradiance environment maps, for rendering Lambertian objects under distant illumination, are well approximated by a nine-dimensional linear subspace. Thus, from any given view, the appearance of an arbitrary Lambertian shape, under arbitrary distributed distant illumination, is approximately near a ten-dimensional manifold. By adding more parameters, the dimensionality of the manifold increases, and it becomes highly complex and non-linear, although its dimensionality remains far less than that of the image space.

In our work, we would like to be able to extrapolate between images in parameter space by moving along such a manifold and extrapolating novel points. This requires addressing three issues: (i) The manifold is complex and has a highly non-linear global structure, (ii) The “curse of dimensionality”; the dimensionality of the image space is too high, and (iii) There is no natural correspondence between the images. Classical approaches such as Principle Component Analysis or Multiple Discriminant Analysis, that find projections that best represent or separate data in a least squares sense, are unsuitable for our purposes since they require correspondence.

Our approach is based on the observation that although the manifold is globally non-linear, *locally* it is approximately linear. This alleviates the problem of complexity of the manifold. It permits applying simple local linear models as a good approximation of a global highly non-linear model. To reduce the high dimensionality of the space and to overcome the lack of correspondence, the images are decomposed into multi-scale fragments. A fragment is defined as a connected region cut from an image tile. The tiles are overlapping and their aggregation is an over-complete representation of the image. The key idea is to apply the synthesis at the tile level, and in a way that the composition of their fragments into a full image is coherent.

4. Style synthesis

As explained in the introduction, we would like to synthesize novel images based on the training set. Our solution is based on adaptively decomposing the input image into fragments and applying the synthesis locally. We construct a feature pyramid for each image to capture various features at multiple scales. The training set constitutes an over-complete representation, considering fragments at all scales, positions, and eight orientations. The input image is adaptively subdivided by a quad-tree whose leaves consist of overlapping tiles that are traversed in Morton’s order [25]. Images are then synthesized from coarse to fine where each level is based on a coarser level. This captures features at multiple scales and accelerates the computations.

At each level, tiles are synthesized from a number of example tiles. The choice of these examples is based on multiple constraints, which include the underlying image geometry. The extrapolated tiles are then cut into fragments which together form a coherent tessellation of the novel image. In the following we elaborate on each of these stages.

4.1. Image decomposition

We adaptively decompose the input image $\mathbf{A}(m, n)$ into overlapping tiles. Our algorithm maintains a quad-tree for the input image, where each of its nodes represents an image tile (element), and its four children represent a partition of that element. The leaves of the tree correspond to elements that together cover the image. Each element is extended to include overlapping boundaries. These boundaries form the constraints among adjacent elements which are necessary to generate a coherent image. During image generation, elements are traversed in an order that preserves coherence with previously synthesized regions.

An adaptive subdivision requires a method for deciding whether an element should be split or not. Our initial subdivision of the input is based on the luminance and color values across an element. If any of the absolute differences between the maximum and minimum values is above a threshold δ , and the minimum element size ε_{\min} has not yet been reached, then the element is recursively subdivided. In all the results shown in this paper we set δ between 0.25 and 0.4, and ε_{\min} to 4 by 4 and 8 by 8.

This coarse subdivision is further refined at runtime in any of the following stages: (i) Multi-dimensional search: if a suitable set of candidate tiles is not found; (ii) Fragment synthesis: if the model fails to accurately reconstruct (validate) the input tile; and (iii) Image reconstruction: if the synthesized fragments do not agree with previously synthesized regions. There is a trade off here since finer elements meet the above conditions but do not capture large features.

4.2. Multi-dimensional search

An important part of the algorithm is the selection of a set of fragments which has a 2D (style and content) embedding that reflects the relation among the training set images in the higher dimension. At the same time, the synthesized fragments must agree with their neighbors. Moreover, in a single step, the algorithm synthesizes a set of fragments, one for each style and content. Thus, a multi-dimensional search must simultaneously take into consideration constraints from multiple tiles and boundary values. Boundaries are crucial in capturing geometric configurations and maintaining consistency with previously synthesized regions.

A *search vector* is the concatenation of components from multiple origins, as illustrated in Figure 2. Formally, let $\mathbf{A}(i, j)^l$ denote an image in row i and column j of the matrix, at scale l , where $l = 0$ is the coarsest level. Let $T_{p,\varepsilon,o}$ denote an extended tile around position p , of size ε , in orientation o , and similarly, let $B_{p,\varepsilon,o}$ denote the corresponding boundary. Each search vector is a concatenation of the boundaries B with previously synthesized fragments in styles $j = 1, \dots, n-1$, and tile T in style n , and the corresponding tiles at a coarser scale $l-1$, for $l > 0$,

$$B_{p,\varepsilon,o}(\mathbf{A}(i, 1)^l), \dots, B_{p,\varepsilon,o}(\mathbf{A}(i, n-1)^l), T_{p,\varepsilon,o}(\mathbf{A}(i, n)^l), \\ T_{\frac{p}{2}, \frac{\varepsilon}{2}, o}(\mathbf{A}(i, 1)^{l-1}), \dots, T_{\frac{p}{2}, \frac{\varepsilon}{2}, o}(\mathbf{A}(i, n)^{l-1}). \quad (1)$$

Given the vector defined in Eq. (1), with input content $i = m$ and upright orientation, we find its nearest neighbors in each content row $i = 1, \dots, m-1$, over each position, and orientation. This enforces the same position p and orientation o , for the same content across different styles $j = 1, \dots, n$, but allows different positions and orientations for different contents $i = 1, \dots, m$.

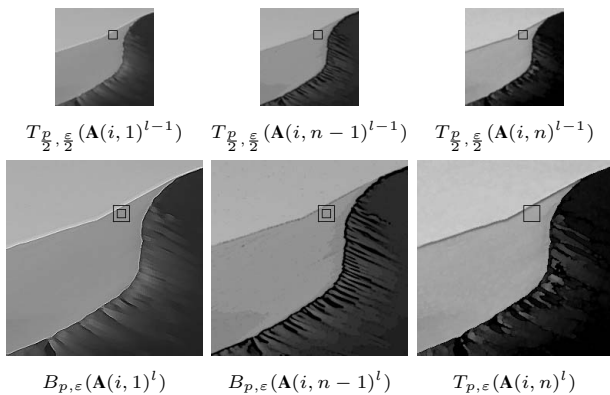


Figure 2. The components of a search vector for a single content row are highlighted. Notice that the extension of a uniform tile captures the edge.

The above multi-dimensional search is applied at multiple scales over several features. Our algorithm maintains a feature pyramid, created in a pre-processing stage, for each image in the matrix \mathbf{A} . The feature pyramid consists of color and luminance Gaussian pyramids, as well as four gradient pyramids in the horizontal, vertical and two diagonal directions, and a Laplacian pyramid, which are derived from the luminance values. Features are selected according to properties of the training set and the component type. Color is effective for a rich training set, which contains a large distribution of samples for a high dimensional search. Luminance is suitable for both the tiles (T) and boundaries (B). In addition, gradients and the Laplacian are effective features for tiles, but not used for thin boundaries. Gradients are computed by the Sobel operator and are correspondingly isotropic for horizontal, vertical and diagonal edges. The Laplacian is invariant under rotations for increments of 45 degrees. In this work we experimented with L_1 , L_2 norms, normalized correlation coefficient, and ordinal measures [7] for similarity of vectors. The normalized correlation coefficient is linearly invariant to contrast, and ordinal measure is too expensive for our purposes. Therefore, we use the L_1 and L_2 norms depending on the selected features.

The search vectors can be regarded as points in multi-dimensional space. Finding a best match is then equivalent to finding the nearest-neighbor. Three main strategies of searching for nearest neighbors are pre-structuring, partial distances, and pruning [12]. We use partial distances, which constitute a monotonic non-decreasing function, and test intermediate distances twice. In order to further improve matching performance, the search for a similar vector is performed hierarchically from coarse to fine. The position of the best matching vector is successively refined based on the best position found at a coarser level. The search is then confined to a window that decreases exponentially in size while traversing the pyramid from coarse to fine. For a full pyramid, each search is logarithmic in the number of pixels in the finest level times the number of features. We choose the coarsest level in the search for each vector such that the size is at least ε_{min} . Overall, the entire search is linear in the resolution of the input image times the logarithmic factor in the number of training images, their resolution, the number of orientations, and the number of features.

4.3. Fragment synthesis

In this section we describe the mathematical tool used to locally factor style and content of tiles and synthesize novel fragments. We begin by briefly reviewing multilinear and bilinear forms, and factoring equations for symmetric bilinear forms (the reader is referred to Tenenbaum and Freeman [29] for detailed derivations).

A multilinear form on V^n is a mapping $f : V^n \rightarrow F$ which is linear in each argument, that is, for all $\mathbf{x}_i, \mathbf{x}'_i \in V$, and all $\lambda \in F$:

$$\begin{aligned} f(\mathbf{x}_1, \dots, \mathbf{x}_i + \mathbf{x}'_i, \dots, \mathbf{x}_n) &= f(\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n) + \\ &\quad f(\mathbf{x}_1, \dots, \mathbf{x}'_i, \dots, \mathbf{x}_n) \\ f(\mathbf{x}_1, \dots, \lambda \mathbf{x}_i, \dots, \mathbf{x}_n) &= \lambda f(\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n). \end{aligned}$$

The special case of a 1-form is a linear mapping $f(\alpha \mathbf{x} + \beta \mathbf{y}) = \alpha f(\mathbf{x}) + \beta f(\mathbf{y})$. A 2-form is a bilinear form. Defined on $V \times W$, it is a mapping $f : V \times W \rightarrow F$ which is linear in each argument, that is, for all $\mathbf{x}, \mathbf{x}' \in V$, all $\mathbf{y}, \mathbf{y}' \in W$ and all $\lambda \in F$:

$$\begin{aligned} f(\mathbf{x} + \mathbf{x}', \mathbf{y}) &= f(\mathbf{x}, \mathbf{y}) + f(\mathbf{x}', \mathbf{y}) \\ f(\mathbf{x}, \mathbf{y} + \mathbf{y}') &= f(\mathbf{x}, \mathbf{y}) + f(\mathbf{x}, \mathbf{y}') \\ f(\lambda \mathbf{x}, \mathbf{y}) &= \lambda f(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}, \lambda \mathbf{y}). \end{aligned}$$

If $f : V \times V \rightarrow F$ is a bilinear form then the matrix \mathbf{M} of f relative to the basis $(\mathbf{v}_1, \dots, \mathbf{v}_n)$ is given by $m_{ij} = f(\mathbf{v}_i, \mathbf{v}_j)$. If $\mathbf{x} = \sum_{i=1}^n x_i \mathbf{v}_i$ and $\mathbf{y} = \sum_{i=1}^n y_i \mathbf{v}_i$ then the polynomial and matrix representations of the form are:

$$f(\mathbf{x}, \mathbf{y}) = \sum_{i,j=1}^n x_i y_j m_{ij} = \mathbf{x}^t \mathbf{M} \mathbf{y}. \quad (2)$$

A bilinear form is symmetric if $f(\mathbf{x}, \mathbf{y}) = f(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in V$, and in matrix form $\mathbf{M} = \mathbf{M}^t$.

As described above, a bilinear form maps two vectors \mathbf{x} and \mathbf{y} by a matrix \mathbf{M} to a scalar, and introducing subscripts for multiple vectors, matrices and scalars:

$$a_{ijk} = \mathbf{x}_i^t \mathbf{M}_k \mathbf{y}_j. \quad (3)$$

Now consider the inverse problem. Given only a set of scalars a_{ijk} , a bilinear model can be fitted by minimizing the total squared error:

$$\min_{\mathbf{x}_i, \mathbf{M}_k, \mathbf{y}_j} \sum_i \sum_j \sum_k (a_{ijk} - \mathbf{x}_i^t \mathbf{M}_k \mathbf{y}_j)^2. \quad (4)$$

A solution consists of a set of bilinear forms, defined by matrices $\{\mathbf{M}_k\}$, together with sets of vectors $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_j\}$.

We use an iterative SVD method as presented in [29] for Eq. (4). An exact solution exists when the number of vectors $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_j\}$ equals their dimension. For example, for general matrices \mathbf{M}_k , an exact solution exists when choosing each set $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_j\}$ to be a basis. For the special case of the standard basis, the matrices are simply defined by the scalar values themselves $\mathbf{M}_k(i, j) = a_{ijk}$.

Next, we would like to estimate the two vectors \mathbf{u} and \mathbf{v} , derived from the established mappings $\{\mathbf{M}_k\}$, for another vector \mathbf{b} :

$$b_k = \mathbf{u}^t \mathbf{M}_k \mathbf{v}. \quad (5)$$

The above is solved iteratively, where the initial guess for the vector \mathbf{v} is the mean of the vectors \mathbf{y}_i , and the superscript t denotes the vector transpose of a block matrix [29]:

$$\mathbf{u} = ((\mathbf{M}\mathbf{v})^t)^{-1} \mathbf{b} \quad \mathbf{v} = ((\mathbf{M}^t \mathbf{u})^t)^{-1} \mathbf{b}. \quad (6)$$

Finally, the derivation above is applied to a style and content matrix of tiles. The tile positions, orientations and scales are of the boundaries and tile in scale l in Eq. (1), for the best match in each content row $i = 1, \dots, m-1$, and the original search vector. The k th pixel value of a tile in content row i and style column j is denoted as a_{ijk} . The vectors \mathbf{x}_i and \mathbf{y}_j determine the style and content of each tile. The vector \mathbf{b} denotes the input tile, and its estimated style and content parameters are \mathbf{u} and \mathbf{v} . In case the model fails to reconstruct (validate) the input tile, Eqs. (5,6), we first extend the number of nearest neighbors considered in the search, and then refine the input subdivision.

Having estimated the style and content parameters \mathbf{u}, \mathbf{v} of the remaining row and column, the tile matrix is completed by applying $\mathbf{x}_i \mathbf{M}_k \mathbf{v}$ and $\mathbf{u} \mathbf{M}_k \mathbf{y}_j$.

As explained, the multi-dimensional search is based on a number of selected features. However, after the search finds a set of tiles, we consider only luminance, and color for a sufficiently varied training set. Synthesis is performed in the perceptually-based (l, α, β) color space [24]. This space minimizes the correlation between channels, and so tile synthesis is performed separately for each channel. The results are then transformed back to (r, g, b) values for display.

4.4. Image reconstruction

Prior to synthesis, the search for a set of tiles enforces that they match previously synthesized regions. This is not sufficient to guarantee that the synthesized tiles match previously synthesized regions as well. The error after synthesis is computed for the overlapping boundaries, over all features, across the different styles. If it is proportional to the error before synthesis, then we repeat the synthesis process, extending the number of nearest neighbors. If a small number of nearest neighbors is exhausted, then the search and synthesis process is refined by recursively subdividing tiles. Once a set of matching tiles is synthesized, we find the minimum cost path along the error surface of the overlapping boundaries by dynamic programming [13], breaking ties by taking the center values.

Traversal of the entire input image guarantees a cover of the images in the m th row, but not of the images in the n th column of \mathbf{A} . The synthesized tiles of the n th column are transformed back to their original orientation, and may appear at any image position according to the search. Thus, for each scale l , after synthesizing images $\mathbf{A}(m, j)^l$ for columns $j = 1, \dots, n-1$, the images $\mathbf{A}(i, n)^l$ for rows

$i = 1, \dots, m - 1$ are completed by extending image analogies. The roles of the training images and the inputs are then reversed. The images of each row $i < m$ serve separately as input, and the m th row serves as the training images. Additional modifications, compared with image analogies [20], consist of (i) adaptively decomposing the input, (ii) stitching together fragments instead of synthesizing pixels, and (iii) searching for a simultaneous match of previously synthesized boundaries and candidate regions, in multiple scales and across all styles as shown in Figure 2.

Since style synthesis is performed from coarse to fine we specify the initial conditions for the search and synthesis, and the inductive step applied from each level to the next. First, if the coarsest level $l = 0$ in the pyramid consists of a single tile, then the search is empty, and the remaining tiles are synthesized. Next, for $l > 0$, the image boundaries for each style in the remaining m th row are initialized, and the coarser level $l - 1$ serves as an initial guess for the remaining n th column. In particular, for different luminance across styles, after initializing $A(m, j)$ with the target, linearly matching the histograms to $A(1 : m - 1, j)$ improves the result. In this case, $A(i, n)$ is initialized with random samples of $A(m : n)$. For the same colors across styles, initializing $A(i, n)$ with the mean of $A(i, 1 : n - 1)$ yields comparable results.

Finally, we consider two special cases: (i) A single content training row; The search is performed under multiple constraints simultaneously, finding a single fragment for each image. Therefore instead of extrapolation, the nearest neighbor is selected. Notice that if there is no underlying model across the different content rows, then images in the same style and different contents are regarded as a single image; and (ii) A single style training column; Content rows are considered as one image, and the search finds a single fragment, selecting the nearest neighbor.

5. Results

We have experimented with our method with various sources and styles: (i) Artistic, sketch, and texture filters from an image editing tool [1]; (ii) Painterly renderings generated by applying layers of curved brush strokes [19]; and (iii) Real paintings. The filters and painterly styles were applied to a variety of image classes from a database of stock photographs. The subjects of these photographs are varied and include landscapes, buildings, people, products, and more. For all of these families of styles and image content classes, our method produces visually plausible results. Computation times range from one hour (Figure 4) to ten hours (Figure 3) for synthesizing a set of 256 by 256 images on a 1.8GHz PC, running a Java implementation.

The transposed matrix in Figure 3 shows natural images in four different artistic styles, in each row from top to bot-

tom: poster edges, fresco, paint daubs and dry brush. The poster edges filter reduces the number of colors in an image and accentuates edges with black lines. The fresco filter applies short and rounded dabs creating a coarse style. The paint daubs filter paints an image with simple round brush strokes. The dry brush filter simplifies an image by painting edges with round brush strokes and reducing the range of colors.

The transposed matrix in Figure 4 shows landscape images in three different sketch styles: charcoal, reticulation, and graphic-pen. The charcoal filter draws major edges in bold, while mid-tones are sketched using diagonal strokes. The reticulation filter creates an image that appears clumped in the shadow areas and lightly grained in the highlights. The graphic pen filter uses fine linear strokes to capture image details. Only upright orientation was considered for synthesizing the images shown on the right column, to capture the directional strokes of the graphic-pen. We compare our results with applying the filter directly to the original images, as shown in the bottom row of Figure 4.

The matrix in Figure 5 shows landscape images. Each of the landscape images has a different appearance. We applied three different artistic and texture styles: craquelure, sponge, and water-color. The craquelure filter paints an image onto a relief surface, generating cracks that follow contours. The sponge filter creates images with highly textured areas of contrasting color. The water-color filter simplifies image details by saturating color at edges.

The matrix in Figure 6 shows images in two different painterly styles [19], impressionist and expressionist, and a real painting. The impressionist style applies moderately curved brush strokes without random jitter. The expressionist style paints an image with elongated brush strokes with random color jitter. The input image is from a painting in a post-impressionist style.

6. Conclusions and future work

We have introduced a method for style synthesis that utilizes a training set of images. While the specific implementation here is image synthesis, the main motivation is to explore the separability of content and style, which can be viewed as a fundamental topic in image pattern analysis. Our method is based on an adaptive scheme to synthesize local fragments of an image by extrapolating multiple fragments. Future work will focus on problems with more than two factors, using a local multilinear model for approximating highly complex factors of appearance, such as changing weather conditions.

Our approach is example-based, which means that its performance is dependent on the richness of the available fragments in the training set. In addition, it is an image-based 2D method and does not incorporate high-level in-

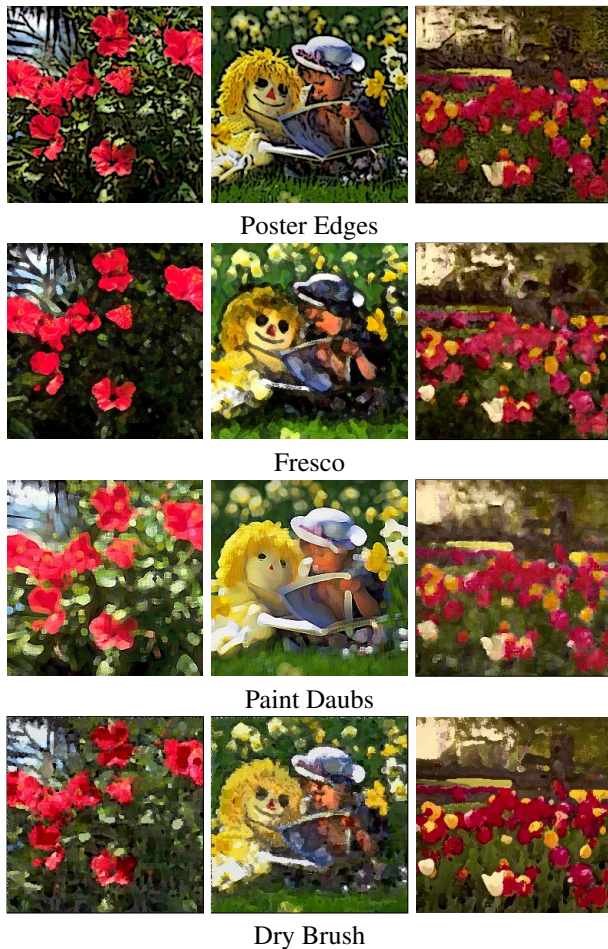


Figure 3. A set of natural images and artistic filters form an image matrix. The two images in dry-brush on the left of the lower row, and the three images in poster-edges, fresco, and paint-daubs on the right column were synthesized by our algorithm.

formation about the underlying scene. Although we have applied several techniques for accelerating the search, our algorithm is still very slow. To speedup the search we would like to apply geometric hashing, and then use larger training sets and consider tile deformations.

Finally, an interesting extension of this work is to 3D surface patches of 3D models. This will require to partition a mesh into patches and incorporate mesh coordinates in the synthesis process.

Acknowledgments

This work was supported by a grant from the Israeli Ministry of Science and by a grant from the Israeli Academy of Sciences (center of excellence).

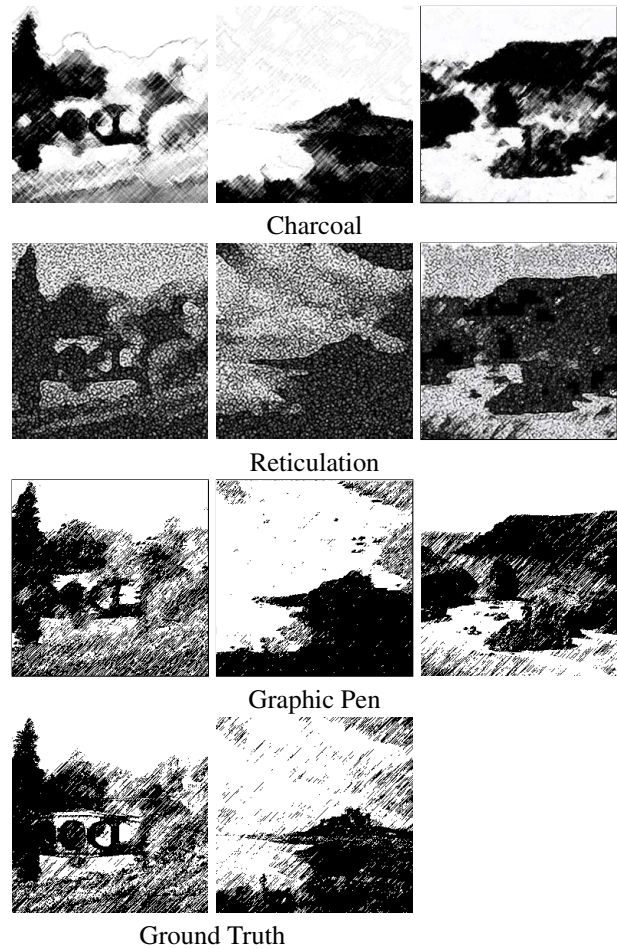
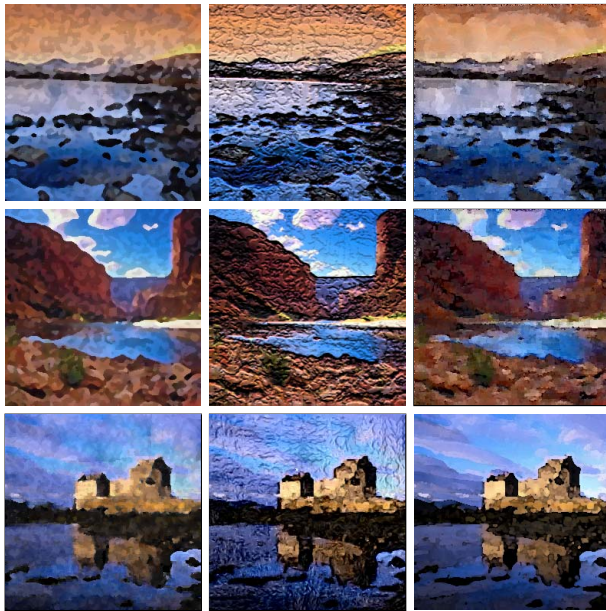


Figure 4. A set of landscape images and sketch filters form an image matrix. The two images in charcoal and reticulation on the right column were synthesized by our algorithm.

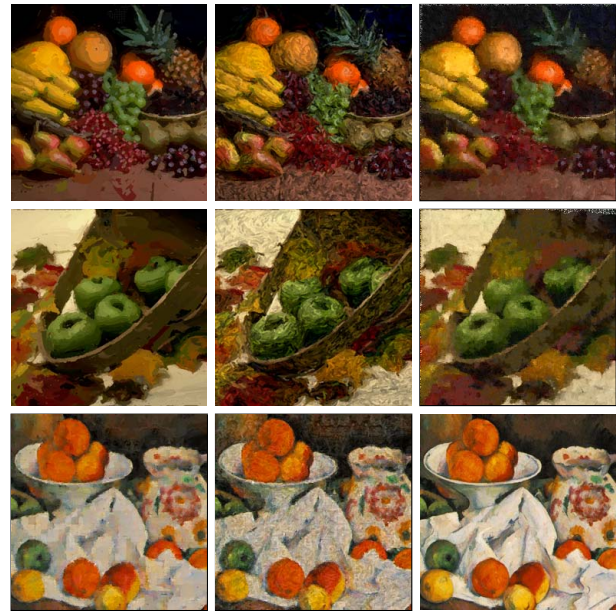
References

- [1] Adobe Systems Incorporated. <http://www.adobe.com>.
- [2] M. Ashikhmin. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, pages 217–226, 2001.
- [3] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002.
- [4] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. In *IEEE International Conference on Computer Vision*, volume 2, pages 383–390, 2001.
- [5] D. Beymer and T. Poggio. Image representation for visual learning. *Science*, 272:1905–1909, 1996.
- [6] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. MIT AI Lab Technical Memo 1431, 1993.
- [7] D. N. Bhat and S. K. Nayar. Ordinal measures for image correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:415–423, 1998.



Sponge Craquelure Water Color

Figure 5. A set of landscape images and artistic, texture filters form an image matrix. The images in sponge and craquelure on the lower row, and the two images in water color on the top right column were synthesized by our algorithm.



Impressionist Expressionist Cezanne

Figure 6. A set of painterly rendered images and an input from a Still Life by Paul Cezanne form an image matrix. The images in impressionist and expressionist style on the lower row, and the two images on the top right column were synthesized by our algorithm.

- [8] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *European Conference on Computer Vision*, pages 109–124, 2002.
- [9] M. Brand and A. Hertzmann. Style machines. In *ACM Siggraph*, pages 183–192, 2000.
- [10] C. Bregler, M. Covell, and M. Slaney. Video rewrite: driving visual speech with audio. In *ACM Siggraph*, pages 353–360, 1997.
- [11] J. Davis. Mosaics of scenes with moving objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 354–360, 1998.
- [12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd ed., 2001.
- [13] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *ACM Siggraph*, pages 341–346, 2001.
- [14] D. J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America*, 4(12):2379–2394, 1987.
- [15] W. T. Freeman, T. R. Jones, and E. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, pages 56–65, 2002.
- [16] W. T. Freeman and E. Pasztor. Learning low-level vision. In *IEEE International Conference on Computer Vision*, pages 182–189, 1998.
- [17] W. T. Freeman and J. B. Tenenbaum. Learning bilinear models for two-factor problems in vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 554–560, 1997.
- [18] P. Haeberli. Paint by numbers: abstract image representations. In *ACM Siggraph*, pages 207–214, 1990.
- [19] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *ACM Siggraph*, pages 453–460, 1998.
- [20] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *ACM Siggraph*, pages 327–340, 2001.
- [21] A. Hertzmann, N. Oliver, B. Curless, and S. M. Seitz. Curve analogies. In *Proceedings of 13th Eurographics Workshop on Rendering*, pages 233–245, 2002.
- [22] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, 2001.
- [23] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *ACM Siggraph*, pages 117–128, 2001.
- [24] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, pages 34–40, 2001.
- [25] H. Samet. *Applications of Spatial Data Structures*. Addison-Wesley Publishing, 1989.
- [26] S. H. Seung and D. D. Lee. The manifold ways of perception. *Science*, 290:2268–2269, 2000.
- [27] P.-P. Sloan, C. Rose, and M. Cohen. Shape by example. In *ACM Symposium on Interactive 3D Graphics*, pages 135–143, 2001.
- [28] C. Soler, M.-P. Cani, and A. Angelidis. Hierarchical pattern mapping. In *ACM Siggraph*, pages 673–680, 2002.
- [29] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.
- [30] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. In *ACM Siggraph*, pages 277–280, 2002.