2013 IEEE Conference on Computer Vision and Pattern Recognition

Pattern-Driven Colorization of 3D Surfaces

George Leifman
Technion
gleifman@tx.technion.ac.il

Ayellet Tal
Technion
ayellet@ee.technion.ac.il

Abstract

Colorization refers to the process of adding color to black & white images or videos. This paper extends the term to handle surfaces in three dimensions. This is important for applications in which the colors of an object need to be restored and no relevant image exists for texturing it. We focus on surfaces with patterns and propose a novel algorithm for adding colors to these surfaces. The user needs only to scribble a few color strokes on one instance of each pattern, and the system proceeds to automatically colorize the whole surface. For this scheme to work, we address not only the problem of colorization, but also the problem of pattern detection on surfaces.

1. Introduction

Colorization was introduced in 1970 to describe the computer-assisted process for adding color to monochrome footage [3]. The term is now used generically to describe any technique for adding color to monochrome images, videos and surfaces [10, 11, 21]. Levin et al. [11] proposed a simple, yet effective, user-guided image colorization method. The user scribbles the desired colors in the interiors of various regions and the system spreads the colors to the rest of the image. Other scribble-based algorithms have subsequently been presented by [8, 28]. When the image contains complex textures, the above methods require massive user input. To decrease user intervention, [16] employ texture continuity and [29] add a classification step to discriminate between the appearance of scribbled pixels.

This paper addresses the colorization of 3D surfaces. The extension of colorization algorithms from images to surfaces is not straightforward. A fundamental assumption in images that the intensity (Y) is given and the algorithms estimate only two of the color channels (UV). To determine whether two neighboring pixels should be colorized using the same color, their intensities are compared. In the case of surfaces, the "intensity channel" does not exist. Therefore, a different technique is needed for determining whether neighboring points should be colorized similarly.

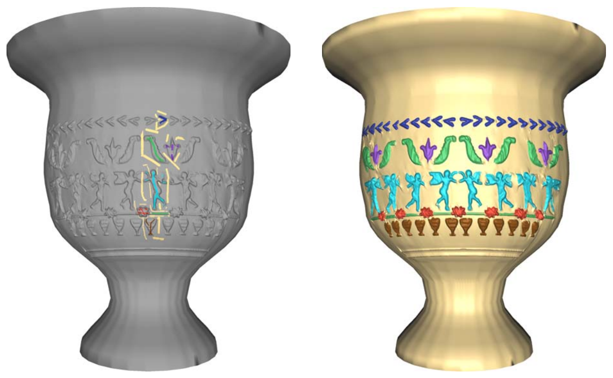


Figure 1. **Pattern-driven colorization.** Given a 3D surface, the user scribbles a handful of strokes within and around a single instance of each pattern (left). Our algorithm colorizes the whole surface accordingly (right).

Recently, [10] introduced a scribble-based colorization algorithm for 3D surfaces. The user draws several strokes and the system propagates the colors to the whole surface. This algorithm does not handle repetitive patterns. Therefore, colorization of objects with patterns requires color strokes on each pattern instance, which is time consuming.

We focus on patterned surfaces, as illustrated in Figure 1. Pattern detection is challenging since it aims at simultaneously solving two intricate problems: segmentation and correspondence across the resultant segments. For surfaces, an added difficulty is the lack of a simple parametrization: Each vertex may have a different number of neighbors and different immediate surroundings. Therefore, unlike images, patches cannot be simply compared.

Nevertheless, some methods were recently proposed for symmetry and pattern detection on surfaces [18]. In transformation voting schemes, the key idea is to find a set of candidate correspondences and vote for a transformation that accounts for them [17, 19]. Voting is limited to a fixed set of transformations. Instead of operating at the level of sample points, in [2] a graph-based approach is proposed, which works at the level of feature curves. While achieving substantial speedup, this approach is intended mostly for man-made objects, such as architectural models. In [15],

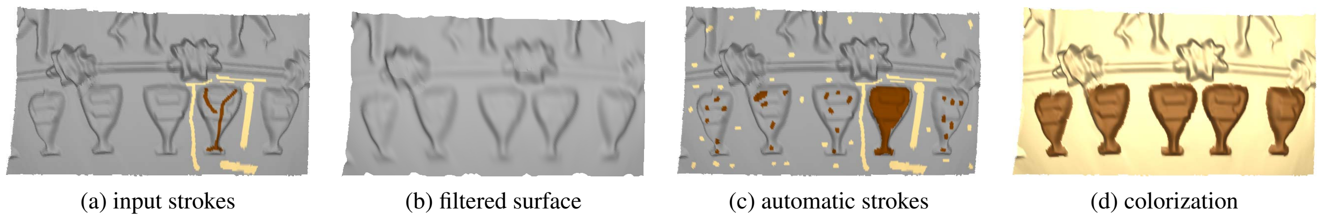


Figure 2. **General approach.** Given a 3D model, the user scribbles only on one instance of the pattern (a). A filtered surface is generated in a pattern-preserving manner (b). Each vertex is then classified in accordance with the pattern, and new strokes are automatically produced (c). Finally, the colorization is performed, utilizing this new set of strokes (d). This image is a zoom into Figure 1.

the focus is only on periodic reliefs.

We propose a novel method for colorization, which performs also pattern classification. It is based on the idea that the few strokes provided for colorization can be extremely helpful for classification. Our algorithm solves three problems. First, we propose a new descriptor that manages to characterize surface regions well (Section 3). Second, using this descriptor, a pattern classification algorithm is suggested (Sections 4). Finally, once the patterns are classified, even highly complex surfaces are automatically colorized using the user's strokes around and within a single instance of each pattern (Section 5).

Our contribution is hence threefold. First, we propose a colorization algorithm that handles patterns. Second, we introduce a new descriptor for surfaces, which is pattern-aware. Finally, we describe a pattern classification algorithm that uses a handful of strokes as input.

2. General Approach

Our aim is to colorize a 3D surface, which may include patterns. We assume that the surface is given as a triangular mesh that consists of vertices and faces. To colorize a model, the user draws a few scribbles with the desired colors on the surface (Figure 2(a)). It suffices to draw a few scribbles within and in the surroundings of a single instance of each pattern. The algorithm then automatically propagates the colors to the remaining vertices of the surface.

For each face the scribble passes through, the closest vertex gets the color of the scribble. These colored vertices are considered the user-defined constraints. Similarly to [10, 11], the underlying assumption is that nearby vertices, whose geometry is similar, should get the same color.

In addition, since our surface has patterns, we make another assumption: Vertices that belong to the same pattern should get the same color. This means that pattern colorization can be viewed as a vertex classification problem. Using the input strokes, we can learn the patterns.

Our approach consists of three steps (Figure 2), which are performed for each pattern in turn. First, we filter the surface in a pattern-preserving manner. In the filtered surface, the pattern boundaries are kept intact, whereas the details in the pattern region are filtered out. This facilities subsequent steps, where sharp pattern boundaries are important for achieving the desirable colorization. Our filtering is discussed when we elaborate on the implementation (Section 6).

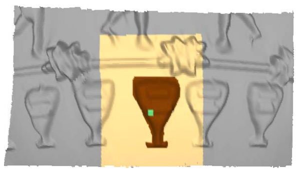
In the second step, each vertex of the filtered surface is classified, determining whether it belongs to the pattern or not. We start by associating every vertex with a descriptor, which characterizes its region in accordance with the pattern. Both the shape of the region and its boundary are considered (Section 3). Then, semi-supervised learning is applied, which combines several techniques in a manner that suits our specific problem (Section 4).

Finally, we use a subset of the classified vertices to automatically produce additional colorization strokes in accordance with the pattern (Section 5). These strokes are the input to a pattern-independent colorization algorithm. The colorization is formulated as an optimization problem, which is based on geometric similarity between neighboring vertices, where the color strokes are considered the user-defined constraints.

3. Pattern-Driven Region Descriptors

This section describes our vertex descriptor, which characterizes the neighbourhood of the vertex, in accordance with the pattern to be colorized. In particular, we first extract for each vertex its surrounding region, which is segmented into foreground (pattern) and background. Then, we consider two types of descriptors of the foreground region: region-based and boundary-based. The region-based descriptor captures the geometry of the region, whereas the boundary descriptor relies only on the shape of the region's boundary. Utilizing two different descriptors not only characterizes the region better, but is also essential for the co-training described in the next section.

Note that this procedure is applied to all vertices. If a vertex does not belong to the pattern, the resulting segmentation of its surrounding region is insignificant. This is so, since the descriptors of this region will be very different from those of foreground vertices and therefore, will be classified as background in Section 4.


(a) pattern vertex

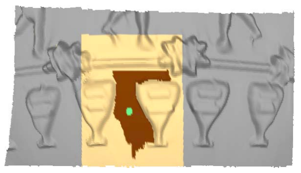

(b) non-pattern vertex

Figure 3. **Examples of pattern-driven regions.** Given a vertex (green), we first extract a sub-surface (yellow) around it and then segment it into a foreground (brown) and a background.

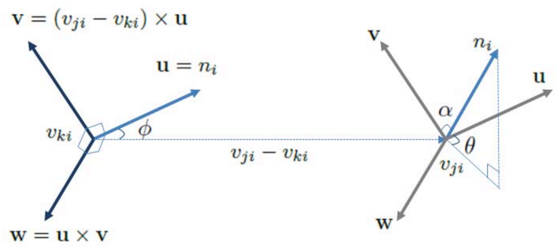


Figure 4. **Region-based vertex descriptor.** A Darboux frame definition and the computation PFH angular variations.

3.1. Pattern-driven segmentation

Prior to computing the descriptor, each vertex is associated with a sub-surface around it, which is segmented into a foreground and a background (Figure 3). The size of the sub-surface is determined based on the user's strokes. We define a bounding box, whose size is twice the size of the bounding box that tightly encloses all the vertices of the user's scribbles. We then extract the connected sub-surface in this bounding box.

To segment the sub-surface into its foreground and background, we apply the colorization algorithm of [10]. This algorithm handles pattern-less surfaces and therefore is suitable for our sub-surface, which typically contains a single instance of the pattern.

Briefly, this algorithm first associates each vertex with the spin image descriptor [9], and then computes the diffusion distance [13] between every pair of neighboring vertices v_i and v_j. To impose the constraint that two neighboring vertices should get the same color if their geometry is similar, the following cost function is minimized:

$$\Omega(C) = \sum_{v_i \in S} \left(C(v_i) - \sum_{v_j \in N(v_i)} k_{ij} C(v_j) \right)^2 . \quad (1)$$

In this equation, $C(v_i)$ is the color of v_i of the sub-surface S, $N(v_i)$ is the neighborhood of v_i, and k_{ij} is a weight function. k_{ij} is large when the descriptor of v_i is similar to that of v_j and small otherwise, and $\sum_{v_j \in N(v_i)} k_{ij} = 1$.

The cost function in Equation 1 is optimized, adding the constraints that the vertex for which the region is computed, gets the foreground color, and the boundary vertices (of the surrounding sub-surface) get the background color. The solution to this optimization problem assigns colors to all the vertices in the sub-surface. Finally, we define the foreground (pattern) region to consist of all the vertices whose assigned colors differ from the background color.

3.2. Region-based vertex descriptor

We seek a descriptor that robustly characterizes the geometry not only of the vertex, but also of the region it resides in. We modify the *Point Feature Histogram (PFH)* descriptor of [24] to suit our problem. This descriptor, which characterizes local features at a given point v_i, is based on a combination of geometrical relations between the points' neighbors.

In particular, for every pair of v_i's neighbors v_{ji} and v_{ki}, a Darboux $\mathbf{uvw}$ frame (Figure 4) is defined as

$$\mathbf{u} = n_i, \quad \mathbf{v} = (v_{ji} - v_{ki}) \times \mathbf{u}, \quad \mathbf{w} = \mathbf{u} \times \mathbf{v}, \quad (2)$$

where n_i is the surface normal at vertex v_i. The following angular variations are then computed:

$$\begin{aligned} \alpha &= \mathbf{v} \cdot n_{ji}, \\ \phi &= (\mathbf{u} \cdot (v_{ji} - v_{ki}))/\|v_{ji} - v_{ki}\|, \\ \theta &= \arctan(\mathbf{w} \cdot n_{ji}, \mathbf{u} \cdot n_{ji}). \end{aligned} \quad (3)$$

Finally, a vertex v_i is associated with a 3D histogram of triples $< \alpha, \phi, \theta >$ for every pair of neighbors.

In our case, the neighbors of a vertex are those included in the foreground region, rather than the immediate neighbors, as common. Moreover, instead of using all the pairs of vertices, we use only the pairs in which v_i is a member. In addition to acceleration [23], the descriptor becomes more robust. This is so, since when considering all the pairs in the region, the resulting descriptors become almost identical for the vertices in the same region. Finally, to support invariance to reflection (in addition to rigid transformation), we take the absolute values of all the angles in Equation 3. In our implementation we use $6^3 = 216$ bins.

To compare our descriptors, we use the χ^2 function [25]. We experimented with other alternatives, including $L1$, $L2$, Histogram Intersection Kernel [23], and the Earth Mover's Distance (EMD) [22]. We concluded that χ^2 is the best compromise between the quality of the results and the computational efficiency. Using cross-bin distances may give better results, but are much more expensive.

3.3. Boundary-based vertex descriptor

To describe the curve that bounds the foreground region, we use 2D histograms of the curve's curvature and the torsion. Intuitively, a curve can be obtained from a straight line by bending (curvature) and twisting (torsion).

To estimate the curvature and the torsion, we utilize the *independent coordinates* [12]. Briefly, a cubic parametric curve is fitted to the coordinates of each sample point on the curve. For example, for the x coordinate we get:

$$\hat{x}(s) = x_0 + x_0' s + \frac{1}{2} x_0'' s^2 + \frac{1}{6} x_0''' s^3, \quad (4)$$

where x_0 is the position of the sample point, x_0', x_0'', x_0''' are the 1^{st}, 2^{nd}, and 3^{rd} derivatives respectively, and s is the arc-length parametrization. The derivatives are found by minimizing the following function for each sample point:

$$E_x(x_0', x_0'', x_0''') = \sum_{i=-n}^{i=n} (x_i - (x_0' s + \frac{x_0'' s^2}{2} + \frac{x_0''' s^3}{6}))^2, \quad (5)$$

where $\{\mathbf{x}_i\}_{i=-n}^{n}$ are the neighboring sample points of x_0.

Given these derivatives, the curvature and the torsion are computed according to their definition [5] (where $v = (x, y, z)$):

$$\kappa(s) = \frac{|v' \times v''|}{\|v'\|^3}, \quad \tau(s) = -\frac{(v' \times v'') \cdot v'''}{\|v' \times v''\|^2}. \quad (6)$$

Finally, a curve is represented by a 2D histogram of the curvature and the torsion of its sample points. In our implementation, we use 256 bins. The range of the bins depends on the surface. Let μ be the average curvature/torsion and σ be the standard deviation, then the range is $[\mu - 3\sigma, \mu + 3\sigma]$.

As a similarity measure between the above histograms, we use the *diffusion distance* [13]. This distance models the difference between two histograms as a temperature field and considers the diffusion process on the field. The integration of a norm on the diffusion field over time is used as a dissimilarity measure. Though it is a cross-bin distance, as will be evident in the next section, the number of comparisons between boundary descriptors is much smaller than the number of comparisons between region descriptors. Therefore, we can afford using this cross-bin distance.

4. Pattern Classification

This section describes our classification technique, whose aim is to determine which vertices of the surface belong to the pattern and which do not. Recall that to colorize a surface, the user draws a few scribbles on a pattern's instance and around it. For each face the scribble passes through, the closest vertex gets the color of the scribble. As a result, a few vertices are colored with the foreground color, and these are the *positive examples.* A few other vertices get the background color and these are the *negative examples.* Altogether, we have a very small number of *training examples*, which is a challenge for classification.

A common way to handle shortage in training examples is to enrich the set of examples. While the set of positive examples can be easily enriched by using the foreground vertices found by our segmentation (Section 6), there is no easy way to enrich the set of the negative examples automatically. This is so, since we cannot simply extend the neighborhood of a vertex, which might intersect pattern instances. Therefore, we face a two-class classification problem, for which we have a large number of positive examples and a small number of negative examples.

Moreover, the set of positive examples is likely to represent the true distribution, since the class of interest has a compact support. Conversely, the negative examples might be too sparse to represent their true distribution. These might also be heterogeneous and reside far from each other in the descriptor space. Thus, any attempt to cluster them is not only difficult, but could also be potentially damaging.

Fortunately, we can avoid the classification of all the vertices. Recall that our final goal is to classify only a representative subset of vertices with very high confidence, which will suffice for colorization. We propose a new technique, which combines several methods in a manner that suits our purpose. In particular, to overcome the problem of having a small set of negative examples, we use semi-supervised learning (Section 4.1). However, even after enriching the set of the negative examples, it is still too sparse to represent the true distribution. Therefore, we use supervised feature extraction to improve the separability between the positive and the negative examples (Section 4.2). Finally, to get only a representative subset with high confidence, we train an SVM classifier using the enriched set of the negative examples, and choose the vertices that are far from the separation plane (Section 4.3).

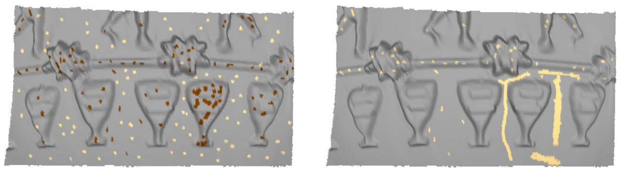
4.1. Enriching the set of examples

We propose to bootstrap the set of negative examples, using a unique combination of *self-training* [27] and *co-training* [1]. Self-training repetitively trains a classifier, uses it to classify unlabeled data, and then adds the most confident data to the training set. Co-training splits the features into two sets and trains two classifiers. Each classifier then "teaches" the other classifier, using a small set of unlabeled examples with the highest confidence.

In our case, self-training cannot be used as is, since even the most confident results might be false positives, as demonstrated in Figure 5(a). Co-training does not work sufficiently well either, since it requires two descriptors, which should each be adequate to train a "strong" classifier.

Instead, since we cannot trust the small non-representative set of negative examples, we train a one-class SVM classifier only on the positive examples, using the region descriptor. Then, the second descriptor (boundary) is utilized to add negative training examples, benefiting from the advantages of co-training.

This is illustrated in Figure 5. Though we are looking for positive examples that lie on the goblet, we find also



(a) the most confident results (b) enriched negative set

Figure 5. **Enriching the set of negative examples.** Given the input from Fig. 2(a), the most confident results of a one-class SVM are shown in (a), where positive results are in brown and negative results in yellow. It can be seen that points on different patterns were erroneously found to be positive. In (b), the enriched set of negative examples is shown, where these are correctly classified.

points which fall on the leaf. This happens because their region descriptors are similar. But, their boundary descriptors differ. We utilize this to disqualify the vertices that were incorrectly classified as positive. We add the top 10% of these vertices as negative examples. This lets us enjoy the advantages of self-training.

4.2. Improving the separability

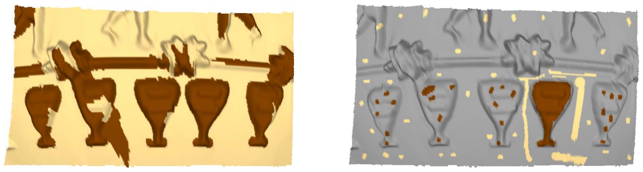
Even after enriching the set of the negative examples, this set is still too sparse to represent the true distribution. Yet, transforming the data to another descriptor space, where the positive examples can be more easily separated from the negative examples, can improve the classification. We seek such a transformation.

We utilize *supervised feature extraction* in the region descriptor space. We are given N observations in the d-dimensional descriptor space, divided into a subset $\{\mathbf{x}_i\}_{i=1}^{N_x}$ of positive examples and a subset $\{\mathbf{z}_i\}_{i=1}^{N_z}$ of negative examples. We aim at finding a projection W onto some r-dimensional subspace ($r \leq d$), $\mathbf{y} = W^\top \mathbf{x}$, where $\mathbf{y} \in \mathcal{R}^r$ is a transformed data point and W is a $d \times r$ transformation matrix. The transformation W should be one whose image can be divided into the subsets $\{\mathcal{Y}_i\}_{i=1}^{N_x}$ and $\{\mathcal{Y}_i\}_{i=1}^{N_z}$, so as to achieve the maximal separation between them.

We use *Biased Discriminant Analysis (BDA)* [30]. Let $\mathbf{m}_x$ be the mean vector of the positive examples. We denote S_z and S_x as follows:

$$S_z = \sum_{i=1}^{N_z} (\mathbf{z}_i - \mathbf{m}_x)(\mathbf{z}_i - \mathbf{m}_x)^\top,$$
$$S_x = \sum_{i=1}^{N_x} (\mathbf{x}_i - \mathbf{m}_x)(\mathbf{x}_i - \mathbf{m}_x)^\top, \quad (7)$$

where the mean vector $\mathbf{m}_x$ is subtracted from the observations. The goal is to achieve maximal separation between the positive and the negative examples, while avoiding the clustering of the negative examples, which reside far from each other in the descriptor space.



(a) all vertices (b) vertices having high confidence

Figure 6. **High-confidence classification.** (a) The result of an SVM classifier has numerous misclassifications. (b) The classification of vertices having high confidence is correct and suffices as input for the final colorization.

To obtain it, the optimal transformation matrix W is defined as

$$W_{opt} = \underset{W}{\text{argmax}} \left\{ \frac{W^\top S_z W}{W^\top S_x W} \right\}. \quad (8)$$

The solution of Equation 8 is the solution to a generalized eigen-analysis [30]. Transforming the region descriptors using W_{opt} not only improves the separability, but also reduces the complexity of the classifier described next.

4.3. High-confidence classification

Thus far, we enriched the set of the training examples and improved the separability. Training an SVM classifier in the transformed space, using the enriched set of the negative examples, produces some wrong results (Figure 6(a)). Since we only need enough correctly-classified vertices to perform accurate colorization, we use only the SVM classifications that are far from the separation plane. In practice, the top 10% of the positive and the top 10% of the negative vertices suffice, as illustrated in Figure 6(b). This result, which is also shown in Figure 2(c), leads to the eye-pleasing colorization in Figure 2(d).

Implementation: The distance measure used both for SVM and for BDA is χ^2. We utilize the generalized form of the RBF kernels [4]: $K_{d-RBF}(\mathbf{x}, \mathbf{y}) = e^{-\rho d(\mathbf{x}, \mathbf{y})}$, where $d(\mathbf{x}, \mathbf{y})$ is a symmetrized approximation of the χ^2 function:

$$d_{\chi^2}(\mathbf{x}, \mathbf{y}) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}. \quad (9)$$

Comparison to other classification methods: We compared our classification to several off-the-shelf methods on our running example in Figure 2. After tuning the parameters of each method to achieve the best results, we got the following misclassifications: 9% using our method, 15% with SVM (RBF kernel), 19% with GMM (5 Gaussians), and more than 30% with EM, when disregarding the high-confidence classification. When applying high-confidence classification, our method went down to 0% and SVM to 6%. Similar ratios are obtained for the other objects.

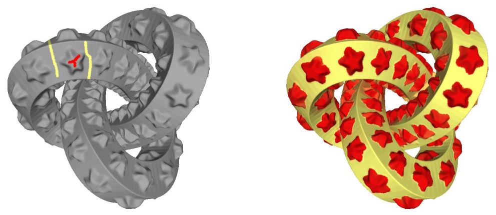


Figure 7. **Pattern-driven colorization of a relief object.** All the stars are colorized by scribbling only four color strokes on one star instance and its surroundings.

5. Final Colorization & Results

The vertices, which were classified with the highest confidence, are used as input to the colorization algorithm, described in Equation (1). The positive vertices get the "foreground color" and the negative vertices get the "background color." Thanks to our pattern-aware filtering, even a single vertex in a pattern suffices to extract the pattern correctly.

We applied our algorithm to a variety of surfaces. Some of the surfaces have reliefs (Figures 1, 7, 8, 11), whereas others are general objects (Figures 9, 10, 12).

In Figure 7, all the stars are colorized easily, using only four color strokes on a single star instance and its surroundings. For comparison, to get similar results using the algorithm of [10], one would need more than 300 strokes.

Figure 8 demonstrates the colorization of multiple patterns. This is done iteratively, colorizing one pattern at a time. The user first marks one of the deers (Figure 8(a)) and our algorithm completes the colorization of all the instances of the deer (Figure 8(b)). Then, the user proceeds to scribble on one of the trees (Figure 8(c)). Our algorithm ignores the pattern found in the previous iteration and colorizes the other trees compatibly (Figure 8(d)). Figure 1 demonstrates an example with six different patterns.

Figure 9 demonstrates the result of our algorithm when applied to a very large Thai statue, consisting of a million vertices. Even though the pattern instances of the three elephants are noisy and non-identical, our algorithm achieves good results. We allow the user to mark scribbles on more than one instance of a pattern, and take all these scribbles into consideration in our classification algorithm.

Finally, Figure 10 demonstrates our result on a man-made CAD object. Contrary to scanned objects, man-made objects usually consist of many disconnected components. In this case, there is no background in each component, but rather only foreground. These objects are typically noise-free and the instances are identical. Therefore, here, our one-class SVM classification in the region descriptor space suffices to obtain the correct result.

Comparison with state-of-the-art results. There is no previous work on pattern-driven colorization. Therefore, we compare our results to those of related, though different, problems. Figure 11 demonstrates that our algorithm competes favorably with the self-similarity detection of [7]. Our algorithm detects all the suction cups and their accurate boundaries, whereas the boundaries produced by [7] are fuzzy and some suction cups are not detected.

Figure 12 compares our results with those of the symmetry-aware segmentation of [14]. Our algorithm colorizes nicely both the different parts and the reliefs on them.

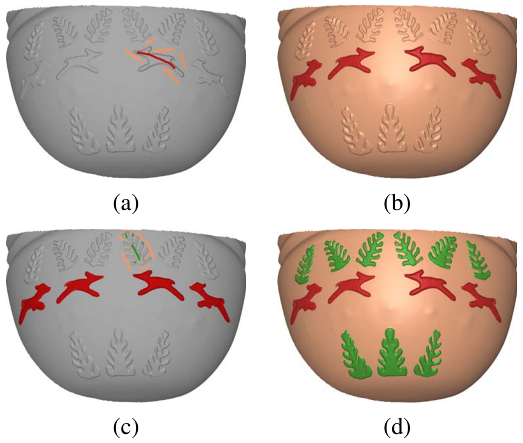


Figure 8. **Colorization with multiple patterns.** Given a surface with two distinct patterns, the user first marks one of them (the deer) (a). All the instances of the deer are colorized by our algorithm (b). Then, the user scribbles on an instance of a tree (c) and all other trees are colorized automatically (d).

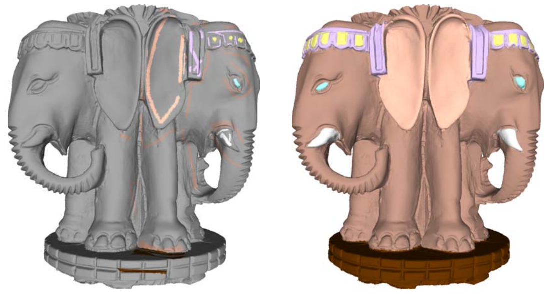


Figure 9. **Colorizing a large noisy object.** Since the pattern instances are not identical, the user marks on a few instances of the same pattern and the learning is based on all these scribbles.

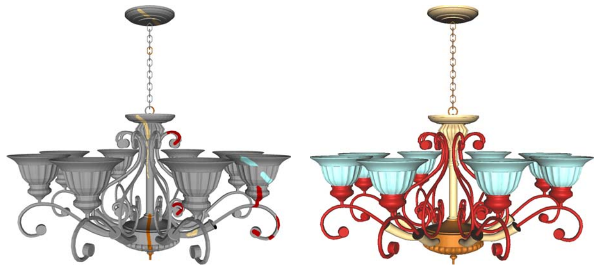


Figure 10. **Colorization of a man-made object.** Since the object consists of identical disconnected components, our one-class SVM classification suffices to obtain the correct result.

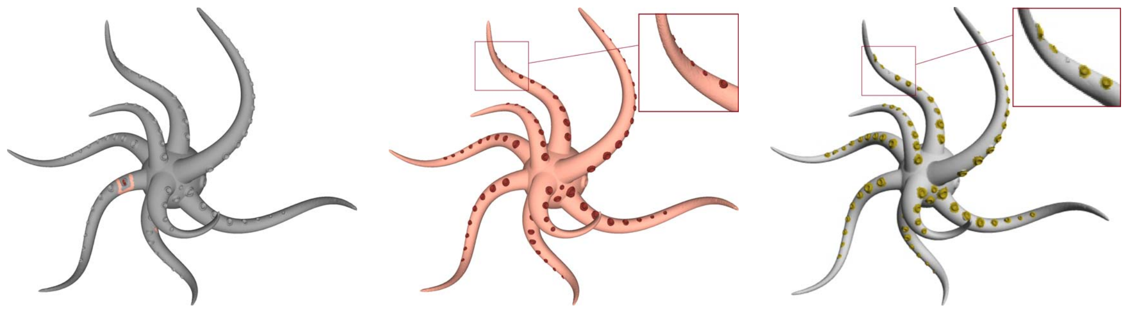


Figure 11. **Comparison to [7].** Our algorithm detects all the suction cups and colorizes them accurately, whereas the boundaries produced by [7] are less precise and not all the suction cups are detected. Note that the suction cups vary in size.

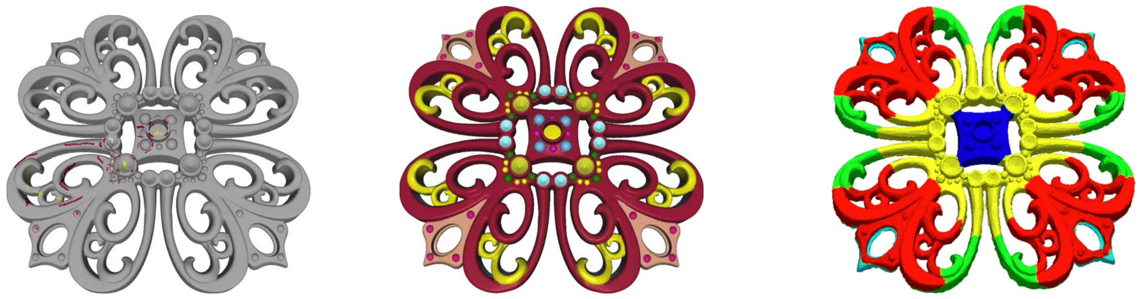


Figure 12. **Comparison to [14].** Our pattern-driven algorithm nicely colorizes both the different parts and the reliefs on them.

6. Implementation: Pattern-Aware Filtering

As mentioned before, even a single vertex inside a pattern suffices to colorize the whole pattern. This is thanks to our filtering, performed as an initial step, which smooths the surface, while keeping the pattern boundaries intact.

Our filtering algorithm proceeds in two steps (Figure 13). First, we segment the sub-surface into its foreground (a single pattern instance) and its background. This is performed by the basic colorization technique described in Section 3.1, using the user's scribbles as input for Equation 1.

Then, we find the optimal smoothing of the surface, as described hereafter. We apply the uniform Laplace smoothing, which approximates the Laplacian of the discretized surface [20]. For vertex v, it is defined as:

$$L(v) = \left(\frac{1}{|N(v)|} \sum_{v_i \in N(v)} v_i \right) - v, \tag{10}$$

where $N(v)$ is the neighborhood of v. Each vertex moves halfway along its $L(v)$ vector: $v' = v + \frac{1}{2}L(v)$.

The question is how many smoothing iterations should be applied. The more iterations, the greater the smoothing. We wish to filter out all the redundant features, yet preserve the boundaries of the pattern. We thus introduce a *pattern-separation quality* measure, as follows.

We randomly sample a set of vertices V from the foreground region. For each sample $v_j \in V$, we apply the colorization algorithm of [10], where v_j gets the foreground color and all the vertices on the sub-surface's boundary get the background. Let R_{Fij} be the foreground region obtained by applying smoothing i times and then colorization.

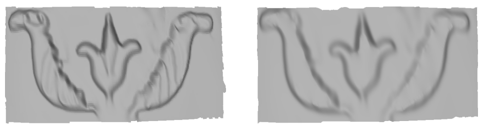


Figure 13. **Pattern-aware filtering.** The boundary of the pattern is preserved, while the details inside the pattern are filtered out.

We say that a smoothed surface has a good *separation quality* if the colorization of the pattern, using a single vertex in it, is similar to the colorization utilizing all the user's strokes. Let R_F be the foreground region, resulting from applying Equation (1) using the user's strokes. If R_{Fij} and R_F agree, then the separation quality is the best. To realize this quality measure, we define the pattern separation quality as

$$Q_i = \frac{1}{|V|} \sum_{j \le |V|} \frac{Area(R_F \bigcap R_{Fij})}{Area(R_F \bigcup R_{Fij})}. \tag{11}$$

We choose the number of smoothing iterations as the one that maximizes Equation 11. In our implementation, $|V| = 10$ and we consider 20 levels of smoothing.

7. Conclusion

This paper introduced a colorization algorithm for surfaces with patterns. After the user scribbles a few color strokes on one instance of every pattern, the system successfully colorizes the whole surface.

In particular, our algorithm addresses several subproblems. The first is surface descriptors for characterizing objects with patterns. We propose two such descriptors—a region-based descriptor and a boundary-based descriptor, both take pattern information into account. Second, we discuss pattern classification of surfaces. We suggest an algorithm that combines the co-training and the self-training approaches. Finally, we show that our classification produces results that allow us to colorize surfaces of varying types and complexities.

We plan to test the suitability of our technique for recoloring surfaces, similarly to image recoloring [6, 26].

Limitations: We assume that the pattern instances are separated by some background area between them. For relief surfaces, we also assume that the curvature of the base is smaller than that of the relief.

Acknowledgements: This research was supported in part by the Israel Science Foundation (ISF) 1420/12 and the Ollendorff Foundation.

References

[1] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Computational learning theory*, pages 92–100, 1998. 4

[2] M. Bokeloh, A. Berner, M. Wand, H. P. Seidel, and A. Schilling. Symmetry detection using line features. *Computer Graphics Forum*, 28(2):697–706, 2009. 1

[3] G. Burns. Colorization. *Museum of Broadcast Communications: Encyclopedia of Television*. 1

[4] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999. 5

[5] M. Do Carmo. *Differential Geometry of Curves and Surfaces*, chapter 1, pages 16–23. Prentice-Hall, 1976. 4

[6] D. Freedman and P. Kisilev. Object-to-object color transfer: optimal flows and SMSP transformations. In *CVPR*, pages 287–294, 2010. 8

[7] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):150, 2006. 6, 7

[8] Y. Huang, Y. Tung, J. Chen, S. Wang, and J. Wu. An adaptive edge detection based colorization algorithm and its applications. In *ACM Multimedia*, pages 351–354, 2005. 1

[9] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *PAMI*, 21(5):433–449, 1999. 3

[10] G. Leifman and A. Tal. Mesh Colorization. *Computer Graphics Forum*, 31(2):421–430, 2012. 1, 2, 3, 6, 7

[11] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694, 2004. 1, 2

[12] T. Lewiner, J. Gomes, H. Lopes, and M. Craizer. Curvature and torsion estimators based on parametric curve fitting. *Computers & Graphics*, 29(5):641–655, 2005. 3

[13] H. Ling and K. Okada. Diffusion distance for histogram comparison. In *CVPR*, volume 1, pages 246–253, 2006. 3, 4

[14] Y. Lipman, X. Chen, I. Daubechies, and T. Funkhouser. Symmetry factored embedding and distance. *ACM Transactions on Graphics*, 2010. 6, 7

[15] S. Liu, R. R. Martin, F. C. Langbein, and P. L. Rosin. Segmenting periodic reliefs on triangle meshes. In *IMA int. conf. on Mathematics of surfaces*, pages 290–306, 2007. 1

[16] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum. Natural image colorization. In *Eurographics Symposium on Rendering*, pages 309–320, 2007. 1

[17] N. Mitra, L. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics*, 25(3):560–568, 2006. 1

[18] N. Mitra, M. Pauly, M. Wand, and D. Ceylan. Symmetry in 3D geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*, 2012. 1

[19] M. Pauly, N. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics*, 27(3):#43, 1–11, 2008. 1

[20] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993. 7

[21] Y. Qu, T.-T. Wong, and P.-A. Heng. Manga colorization. *ACM Transactions on Graphics*, 25(3):1214–1220, 2006. 1

[22] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000. 3

[23] R. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *ICRA*, 2009. 3

[24] R. Rusu, N. Blodow, Z. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IROS*, pages 3384–3391, 2008. 3

[25] B. Schiele and J. Crowley. Object recognition using multidimensional receptive field histograms and its robustness to view point changes. In *ECCV*, pages 610–619, 1996. 3

[26] B. Shida and J. van de Weijer. Object recoloring based on intrinsic image estimation. In *ICCV*, pages 327–334, 2011. 8

[27] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Computational Linguistics*, pages 189–196, 1995. 4

[28] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing*, 15:1120–1129, 2006. 1

[29] L. Yuanzhen, E. Adelson, and A. Agarwala. Scribbleboost: Adding classification to edge-aware interpolation of local image and video adjustments. *Computer Graphics Forum*, 27(4):1255–1264, 2008. 1

[30] X. Zhou and T. Hunag. Small sample learning during multimedia retrieval using BiasMap. In *CVPR*, volume 1, pages I–11 – I–17, 2001. 5