EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Detecting Symmetries and Curvilinear Arrangements in Vector Art

Yi-Ting Yeh[1] and Radomír Měch[2]

[1]Stanford University,
[2]Adobe Systems Incorporated

Abstract

Understanding symmetries and arrangements in existing content is the first step towards providing higher level content aware editing capabilities. Such capabilities may include edits that both preserve existing structure as well as synthesize entirely new structures based on the extracted pattern rules. In this paper we show how to detect regular symmetries and arrangement along curved segments in vector art. We determine individual elements in the art by using the transformation similarity for sequences of sample points on the input curves. Then we detect arrangements of those elements along an arbitrary curved path. We can un-warp the arrangement path to detect symmetries near the path. We introduce novel applications in form of editing elements that are arranged along a curved path. This includes their sliding along the path, changing of their spacing, or their scale. We also allow the user to brush the elements that the system recognized along new paths.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Line and Curve Generation—

1. Introduction

Symmetries are ubiquitous in natural and man-made environments. They provide an important perceptive insight for humans to understand the world. They are also essential elements in visual aesthetics. The more elements are symmetric with each other the more prominent they appear. Even if the elements are placed along a curved path, for example, leaves along a curved plant stem, humans can still perceive the regularity of such arrangement.

By a curvilinear arrangement we understand a placement of elements along a path, in which the distance between the elements is regular and each element is aligned with the curve's normal. These arrangements can loosely be viewed as symmetries warped along a curved path, although it has to be noted that the individual elements are not warped, only their position and orientation is adjusted to follow the path. Curvilinear arrangements are very common in decorative patterns where elements undergo rigid transformation locally while their arrangements form some paths or curved structures globally.

In this paper we present a robust technique for detecting symmetries and curvilinear arrangements in vector art. Vector art is defined by a set of curves and curve bounded shapes and it is used heavily in artistic design.

Our system detects the following symmetries: translations, rotations, mirror reflections, and scaling, and application of translations and mirror reflections along curved paths. We integrated this technique into an interactive design tool. This tool allows users to detect symmetries and curvilinear arrangements in existing art and to synthesize new patterns based on the extracted patterns (see an example in Figures 11, 12 and 13).

The challenge of detecting arrangements of elements along curved paths comes from the fact that we do not have a prior knowledge about the shape, size and location of elements as well as the curved paths, especially since the paths may not be explicitly drawn in the input pattern. The existing symmetry detection technique of clustering point-wise correspondences in transformation space, as presented by Mitra *et al.* [MGP06], can yield too wide distribution of points in the transformation space because the transformation vector is varying along the path. Such distribution of points would

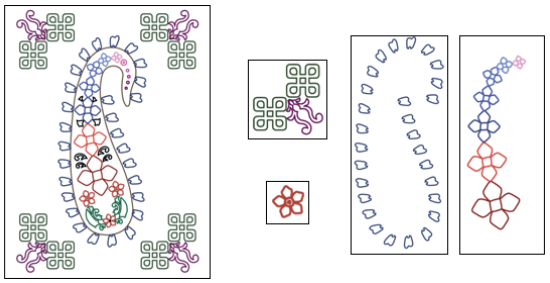

Figure 1: *A decorative pattern with various symmetries and curvilinear arrangements that our system detects.*

be difficult to cluster. Our goal was to design a method that can automatically extract such arrangements without user intervention.

Contribution. Our symmetry detection method extends the Mitra's approach [MGP06], in which the transformation between corresponding sample pairs is first mapped into the transformation space. Next, a set of candidate symmetries are found using mean-shift clustering. The candidate symmetries are verified by growing patches that satisfy the corresponding symmetry transformation in the spatial domain.

To reduce the variance in the transformation space, our algorithm performs local correspondence tests. First, we distribute sample points on the input curves based on the curvature in order to have fewer points along segments with a low curvature. Then we use neighborhood-matching to reject false symmetries that were originally included in the transformation space.

To detect curvilinear arrangements of elements along curved paths, we observe that the distance between the elements is either approximately constant or it is proportional to the size of the elements. We first recognize individual elements that are being transformed, and then define a single point, a centroid, representing each element. We parametrize transformations between pairs of centroids and cluster the points in the distance dimension of the transformation space. We extract the underlying paths of transformations by connecting elements from each cluster. In addition, we un-warp the space along the paths to detect more symmetries.

In summary, we make the following main contributions:

- We develop a novel method for detecting curvilinear arrangements of elements in vector art.
- We un-warp the arrangement paths to detect secondary symmetries skewed by these paths.
- We combine our method with an improved existing method for detecting symmetries into a single application and we introduce new concepts for editing patterns. For example, the artists can slide the group of elements placed along the detected arrangement path.

In Section 2, we review the previous literature relevant to symmetry detection. In Section 3 and 4, we give an overview of our algorithm and describe details of placing sample points, computing their signature, computing transformations of centroids, and recognizing arrangements along curved paths. In Section 5, we show detected symmetries and arrangements in various patterns and we demonstrate examples of using the arrangement detection results to assist pattern editing.

2. Related Work

Symmetry detection has been an active research area in computer vision, computer graphics, architecture, and mathematics. Mathematics behind symmetries has been studied since 19-th century [Kle93].

Early methods of symmetry detection focused on finding exact symmetries in planar figures [Ata85, WWV85]. Although they are computationally efficient, these techniques tend to break down in the presence of irregularities and noise.

To address the noise and slight irregularities that usually appear in real-world objects and patterns, many approaches have be proposed to detect approximate symmetries. Alt *et al.* proposed a method to compute global symmetries between two point sets [AMWW88]. Zabrodsky *et al.* defined a continuous measure of distance of a shape, called symmetry distance, to detect symmetries in noisy data [ZPA95]. They detect and also recover rotation symmetries with respect to the object centroids.

In general, there are two types of approaches to detecting symmetries in 2D images or 3D models. The first type is based on analysis in the frequency domain which uses FFT or autocorrelation to find signal repetition. The second type is feature-based symmetry detection. It needs to find distinct features such as edges and corners that uniquely characterize the images or objects under test.

Keller and Shkolnisky used a pseudo-polar Fourier transform to find angular correlation to detect reflectional and rotational symmetries in 2D images [KS06]. Liu *et al.* proposed a method to detect frieze and wallpaper groups in 2D images by using autocorrelation. They define regions of dominance to find the correct set of peaks instead of using absolute values of the peak heights of the signals. They further apply the discrete Fourier transform frieze detection method to rotation symmetry detection by mapping a 2D rotation symmetry problem to a frieze detection problem using frieze expansion. Their method is also able to identify different rotation symmetry groups [LCL08].

Loy and Eklundh [LE06] developed a method that performs pair-wise matching of feature points generated by SIFT in images. Li *et al.* proposed a method to detect complete and incomplete isometric cycles that induce approximate symmetries in the set of feature points [LLM07,

LLM08]. They also introduced a method to merge incomplete cycles that are induced by the same isometry. Podolak *et al.* defined the planar reflective symmetry transform, PRST, as a continuous shape descriptor of the reflectional symmetry distance with respect to every plane reflection [PSG*06]. They accelerated the computation by applying Monte Carlo integration to perform sampling of the PRST of 3D surfaces. Kazhdan *et al.* introduced a shape descriptor using center of mass [KFR03, KFR04].

One of the main limitations of frequency-based symmetry detection is that it assumes that there exists a single repeated pattern across the whole image. On the other hand, because feature-based symmetry detection finds correspondence by matching feature points or feature clusters, it is able to detect small regions of repeated patterns within a larger image. We chose a feature-based approach because we are interested in artistic patterns which usually have features that do not occupy the whole image.

Instead of looking at the set of representative points only, Mitra *et al.* compute candidate symmetry actions of pairs of points that match in their local shape descriptor [MGP06]. In the transformation space, the most probable symmetries are extracted by using mean-shift clustering. Finally, these candidate symmetries are verified by checking the spatial consistency while false symmetries are rejected.

Another similar problem is to discover the underlying structures that describe the rules of how elements are arranged in patterns or object models. Pauly *et al.* proposed a grid fitting method in the transformation space to detect regular structures of point- or mesh-based models [PMW*08]. Liu *et al.* described a method to segment periodic reliefs by refining user-specified candidates of boundaries on the model using the iterative closest point (ICP) algorithm [LMLR07]. Simari *et al.* focused on detecting mirror symmetries and they applied the result to construct a folding tree data structure which is useful for mesh compression and repair [SKS06].

Our method is not only capable of finding a regular grid or rotation as in grid fitting method by Pauly *et al.* but it can also find elements arranged along curved paths. Instead of looking for a grid in a noisy transformation space we only cluster points in the distance dimension of a less noisy transformation space of centroids.

3. Overview

Our system takes a vector-art pattern as an input and detects symmetries and curvilinear arrangements that appear in the pattern. In Figure 1 we show examples of symmetries and arrangements we can detect in a given vector-art pattern.

Given an input pattern, we sample points along all primitive's curves (Section 4.1). For each sample point, we compute its corresponding normal vector and a signature based on curvature near the sample point (Section 4.2).

Then we compute the transformation for each pair of corresponding points p and q (Section 4.3). The transformation is defined by a scale s between the signatures, the rotation angle ϕ between the normals and the distance d and the angle θ of the translation needed to move point p to point q, after the point p has been rotated round the origin by the angle ϕ.

Valid transformations between each pair of points are then mapped into the transformation space. If we include scaling into consideration, there is a valid transformation $T = (d, \theta, \phi, s)$ for almost any pair, unless the curvature of one point is zero and of the other one is not. That is why we look at the neighbors to see if their transformation is very close. If not, we do not consider T to be a valid transformation (Section 4.3).

We keep four separate transformation spaces, one for each type of symmetry we seek. In case the scale s is close to 1, we test for a potential reflectional symmetry — the angles between normal vectors and the axes of reflection are the same — or a potential translational symmetry — their normal vectors have the same orientation. If the scale value is close to 1 but the angle ϕ is not zero, we have a rotational symmetry. Otherwise we consider the symmetry to be scaling. After classifying the symmetry we add a point to the corresponding transformation space.

Then the system performs a clustering process to find all possible symmetries. A cluster of data in the transformation space indicates that there is a high probability that a symmetry defined by the cluster appears in the spatial domain. The symmetry detection part of our algorithm is very similar to the approach described by Mitra *et al.*, except that we look at the neighborhoods to remove unnecessary points from the transformation space and we place the sample points based on the local curvature to reduce the noise in the transformation space (Section 4.3).

The second part of our algorithm detects arrangements along curved paths. Pauly *et al.* identifies a sequence of translations by detecting a regular grid in the transformation space. Unfortunately, the patterns created by a transformation of objects along curved paths are not regular. In our method, we group every largest possible set of points with respect to a rigid transformation as an element (Section 4.4). The centroid of each element is assigned a weight which equals the size of the element. Then, we can perform transformation analysis between pairs of centroids and fill a centroid-specific transformation spaces. We detect transformations with the same distance component (or the same ratio of the distance and scale components) and we use the centroid locations to extract the underlying arrangement path.

After detecting curvilinear arrangements, we can further un-warp the space around the detected paths to find more symmetries (Section 4.8).

We can use symmetries and curvilinear arrangements for

various editing purposes. We chose to implement an editing mode, in which the user can scale elements associated with a certain arrangement or slide the elements along the detected path. We also implemented a brush that can brush the elements along a path drawn by the user.

4. Symmetry and Arrangement Detection

In this section, we discuss details of our algorithm.

4.1. Placing Sample Points

It is important to properly place sample points on the input curves. First, if sample points on two symmetric shapes are not placed in similar positions in high curvature areas, the values in the transformation space are skewed and noisy. On the other hand, if there are too many sample points along the curve, the algorithm can be slow.

To address this problem, we initially distribute finely spaced sample points along the curves, yet when placing points into the transformation space, we skip points in areas with low curvature (see Section 4.3 for more details).

The sample points $\mathbf{x}_i$ are placed at a distance d_s from each other. We set d_s to be approximately $1/100$ of the scene size. Once we have seed points $\mathbf{x}_i$ we compute a normal and a signature for each of them.

4.2. Normals and Signatures

Normal Vector: For each sample point $\mathbf{x_i}$ on the pattern, we store its normal $\mathbf{n_i}$. We determine the normal directly from the parametric curve.

Signature: The signature s_i is equal to the curvature value that for a parametric curve defined as a function of t can be computed as [DoC76]:

$$s_i = \frac{x'(t_i)y''(t_i) - x''(t_i)y'(t_i)}{\left(x'(t_i)^2 + y'(t_i)^2\right)^{3/2}} \tag{1}$$

where t_i are such that $\mathbf{x_i} = (x(t_i), y(t_i))$.

Two sample points, $\mathbf{x}_i$ and $\mathbf{x}_j$, match each other in their signatures if $|\, s_i - s_j \,| < s_\varepsilon$. We use a one-dimensional binary tree to speed up searches of points with similar signatures.

Because straight line-segments do not define a unique transformation, we set the *LineFlag* of sampling points with zero curvature to true in the sampling stage. They are discarded during the correspondence pairing stage.

4.3. Matching Sample Points

Adding a point to the transformation space for each pair of sample points results in very dense and noisy data. In addition, invalid points in the transformation space may cause false results in the clustering stage. Thus we want to reject those points that do not correspond to meaningful symmetries.

We take advantage of the fact that the samples are placed on continuous curves. Thus in fact we are matching pairs of point neighborhoods, not just pairs of points. In addition, we skip sample points in areas of low curvature.

Skipping sample points: The matching proceeds as follows. We pick a part of the input art represented by a sequence of curves that are continuous. We go through the sample points $\mathbf{x}_i$ on the curves. At each point we generate a random value r_i. If the random value is above the curvature k_i associated with the point $\mathbf{x_i}$ (and normalized to be in (0,1)) we skip the point, unless last n points have not been also skipped (we use $n = 10$).

Checking neighborhood: For the points that we do not skip we search a point with matching signature. For those points $\mathbf{p}_i$ and $\mathbf{p}_j$, whose signature match, we compute the transformation T_{ij} and we check if all points in the neighborhood $\mathbf{p}_i$ and $\mathbf{p}_j$ can be transformed using the transformation T_{ij}. If they do then we add a point to a transformation space for mirror, translational, or rotational symmetry, based on the normal vector (see Section 3).

We do not require an exact match when testing the transformation T_{ij} in the neighborhood of $\mathbf{p}_i$ and $\mathbf{q}_j$. We define a distance metric of a transformation T_{ij} as

$$dist(\mathbf{p}, \mathbf{q}, T_{ij}) = ||T_{ij}(\mathbf{p}), \mathbf{q}|| \tag{2}$$

and accept those neighboring points $\mathbf{p}_{i+k}$ and $\mathbf{q}_{j+k}$, for which $dist(\mathbf{p}_{i+k}, \mathbf{q}_{j+k}, T_{ij}) < d_\varepsilon$ and $|k| \leq N_{nb}/2$. We define d_ε as the distance tolerance of candidate transformations.

Figure 2 shows that by increasing the size of the examining neighborhood, we can reject most false matching. We used N_{nb} (*NBsize*) of 10 in our examples.

Scaling: In the case of scaling, we determine the scaling parameter s based on the ratio of signatures — if we scale an object up by s the curvature is scaled by $1/s$. In this case we also compare the neighborhood of size n, but we need to compute new sample points on the second curve, since we need n points placed not at distance d_s from each other as on the first curve, but at the distance $d_s s$. If the points in the neighborhood satisfy the Eq 2, using $d_\varepsilon s$, then we add a point to the scaling transformation space.

After all valid symmetries are added to the transformations spaces we use mean-shift clustering to detect clusters. Each cluster represents a symmetry applied to the points associated with the cluster.

4.4. Detecting Arrangements Along Curved Paths

In many patterns elements are placed along curved paths. Humans can easily identify them due to the regular spacing or spacing proportional to the uniformly changing size of the elements.

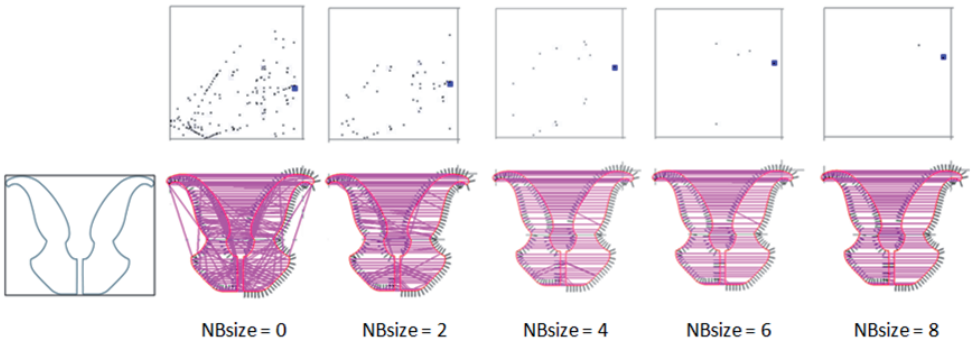

Figure 2: *The effects of the neighborhood size on the level of noise in the transformation space (top). Pink lines indicate candidate reflection symmetries.*

One approach for detecting such arrangements would be to parameterize the translational vector between pairs of points by both the distance and the angle. Figure 3 shows the transformation space after clustering. The dots in the first column show the transformation between pairs of points that correspond to adjacent elements. We can use transformations between adjacent elements to extract the arrangement path.

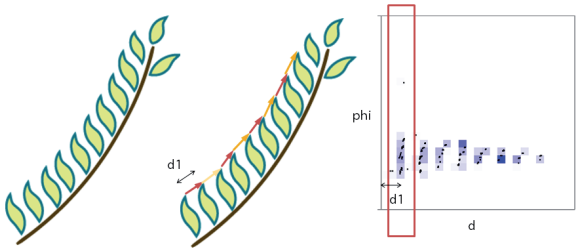

Figure 3: *Naive clustering: an example of arrangement along a curved path and associated transformation space, when all sample points are considered.*

However, in most cases, the elements are not purely translated along the path. They are usually rotated to be aligned with the path's normal. For example, each pair of mirror symmetries in Figure 6 has a different axis of reflection, thus they form separate small clusters in the transformation space. The points representing transformation between pairs of elements along the curve spread too widely and they are hard to cluster. In addition, we should not make any assumption about whether the curved paths are actually present in the art or not since both cases are common in real patterns.

We observe the fact that a transformation between two repeated elements can be viewed as a rigid body motion. Therefore, we find individual elements by grouping continuous sequences of sampling points, that can be mapped to another sequence of points using the same transformation. The centroid of each patch now represents each element.

Once we have centroids, we could use curve fitting or try to connect a sequence of centroids that are closest to each other. There are cases, though, where these approaches will not work well, for example, when there are more elements of the same type near those placed along the path or when the elements of the same type are being placed along more than one path and the paths are close to each other. The extra nearby elements would throw the curve fitting off. If we have elements mirrored along the path, for example, and we greedily connect nearest elements, we would create a zig-zag path.

Since we are interested in elements that are placed regularly along a given path, we use the transformation space of distances or distances divided by the element size and cluster points in these spaces.

Defining elements: In computer vision the common technique used to find correspondence between elements in an image is called RANSAC (RANdom SAmple Consensus) [FB81]. In this technique random points in the input sets are selected and a model is fit into them. By perturbing the selected points the algorithm can find the right correspondences.

With many input points RANSAC is slow. We take advantage of the fact that we have a continuous vector act and we use a greedy approach. We define an element as follows. Let $S(\mathbf{p}_i)$ be the longest sequence of N sample points placed on continuous segments of the input art that pass through the sample point $\mathbf{p}_i$. Let $S(\mathbf{p}_i)$ and $S(\mathbf{q}_j)$ be two sequences on different parts of the input art, and T_{ij} the transformation between points $\mathbf{p}_i$ and $\mathbf{q}_j$. An element is a subset of sequence $S(\mathbf{p}_i)$ defined as:

$$E(\mathbf{p}_i) = \{\mathbf{p}_{i+k}; \quad \mathbf{p}_{i+k} \in S(\mathbf{p}_i), \mathbf{q}_{j+k} \in S(\mathbf{q}_j), \; dist(\mathbf{p}_{i+k}, \mathbf{q}_{j+k}, T_{ij}) < d_\varepsilon\} \tag{3}$$

such that size of $|E(\mathbf{p}_i)| > N/3$. For a pair of points $\mathbf{p}_i$ and $\mathbf{q}_j$ it is easy to find the sequences $S(\mathbf{p}_i)$ and $S(\mathbf{q}_j)$ and test if enough points in them can be transformed by T_{ij}.

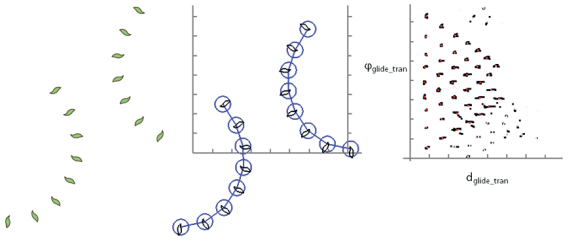

Figure 4: *Our clustering: an example of a curvilinear arrangement and associated transformation space, when centroids of elements are used. Clustering in both dimensions would not give us sufficient information, since the angle of the translation changes along the arrangement curve. Thus we cluster in the distance dimension only, collapsing the columns into strong clusters.*

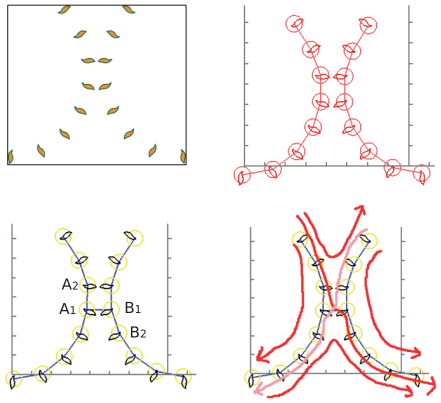

Figure 5: *Connecting centroids to form the arrangement path. If several centroids are at the same distance we choose the one resulting in a smoother path.*

Centroids: For each valid element E we define the centroid $C(E)$ as

$$C(E) = \frac{1}{TotalLength} \sum_{\mathbf{v}_j \in E} (w_j \mathbf{p}_j) \qquad (4)$$

where w_j are lengths of the curve associated with points $\mathbf{p}_j$ and $TotalLength = \sum w_j$. The signature of a centroid is a pair of minimum and maximum signature of sample points in the element and the normal is the sum of all normals computed at the sample points.

Clustering: After detecting elements and computing their centroids with signatures, we match all pairs of centroids. For each pair we insert a point representing the transformation between the two centroids into a centroid-specific transformation space. Instead of clustering in all dimensions, though, we cluster only in the distance dimension, to detect elements at a regular distance from each other.

Before extracting the underlying arrangement paths formed by the elements from the found cluster, we have to remove redundant transformation clusters in the transformation space. For example, if we have 5 elements $E_1, E_2, ..., E_5$ at a distance d_1 along a curve, there are 4 points representing pairs (E_i, E_{i+1}) in the dual distance space at value d_1. In addition, there are 3 points representing pairs (E_i, E_{i+2}) at value $d_2 < 2d_1$, 2 points for pairs (E_i, E_{i+3}) at value $d_3 < 3d_1$, and 1 point for pair $(E1, E5)$ at value $d_4 < 4d_1$. The clusters at distances d_2, d_3, and d_4 represent transformations that are a combination of the basis transformations by the distance d_1. Thus we can remove those points from clusters at d_2, d_3, and d_4 that are formed by elements E_i. Figure 4 illustrates this on a bigger example. Note that in the figure we show the transformation space before collapsing it into distance dimension only.

Connecting centroids: After removing redundant clusters, we can create the arrangement path by connecting elements that fall in the same cluster of distances d. We start with any element and we connect it with a nearby element at the distance d. We repeat the process and create the resulting path. In the case where two paths are adjacent to each other with a distance similar to the inter-centroid distance d, we use a heuristic approach by choosing the path that has smoother transitional angles. For example, in Figure 5 the distance between element A_1 and B_1 is similar to the elements between A_1, A_2 and B_1, B_2. Connecting elements with a similar spacing distance may result in redundant and unnatural paths.

4.5. Detecting Mirror Symmetries Along Curved Paths

If the elements placed along a path are reflected along the path as well, their centroids can be further away from each other at one side of the path and closer to each other at the other side in areas with a higher curvature (see Figure 6). If we detect elements A and B that are reflected, we introduce a phantom object placed at the midpoint between the element's centroids, $C(A)$ and $C(B)$. Each phantom object is also assigned an additional *width* parameter, which indicates the distance between $C(A)$ and $C(B)$. We use the *width* parameter as a signature of the phantom object and it can determine the scaling part of the transformation (Section 4.6).

This approach is a special case of detecting hierarchical symmetrical arrangements along curved paths. Similarly to the reflectional symmetry, we could create a phantom object representing a point symmetry, a rotational symmetry or their combination and use that to detect the arrangement path. In case of combination of symmetries, such as glide reflection, the selection of the right phantom object is not straightforward and we decided to leave the additional implementation for the future work.

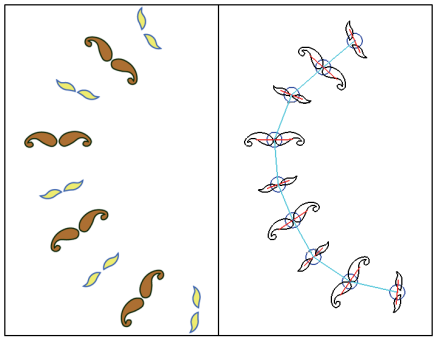

Figure 6: *Example of elements reflected along a path. Blue circles represent phantom objects in the middle of red lines formed by element centroids.*

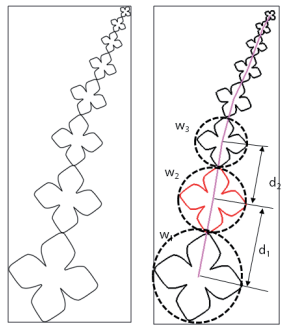

Figure 7: *Examples of elements scaled along a path, in which the ratio of d_i/w_i is constant.*

4.6. Detecting Scaling Along Curved Paths

When we include scaling of elements in our arrangement detection, we use the size of the elements — the length of the curve forming the element — or the *width* parameter of phantom objects to determine the scale factor s between a pair of elements or phantom objects.

There are two cases that we consider. First, if the distance between scaled elements or phantom objects is the same, we use the same technique of clustering in the distance dimension as described in Section 4.4. When connecting the centroids to form the arrangement path, we connect those pairs than have the same scale factor s.

In the second case, we assume that the distance between elements or phantom objects for two consecutive pairs of elements or phantom objects is being scaled by the factor s as well. To detect such arrangements, we construct an additional one-dimensional transformation space, in which we store points at location d/w where d is the distance between a pair of elements of phantom objects and w is the size of the first one. Note that we need to add two points for each pair since the elements or phantom objects may be processed out of order. After clustering in this space we can detect sequences of elements and phantom objects that are scaled along a path.

Figure 7 illustrates the case when the distance between elements is scaled by the factor s as well.

4.7. Merging Lists Corresponding to the Same Path

Figure 8 shows an example of multiple types of elements being arranged along the same path. Sometimes we want to group all types of elements along the same path together while sometimes we want to classify them as different groups.

In the first case, we merge lists of elements that have approximately the same underlying paths. Because the paths are extracted by manifold learning on different point sets in the transformation space, we define a similarity measure between every two extracted paths, P and Q, as

$$E = \sum \left(|\, \mathbf{p}_i, \mathbf{q}_j \,| - |\, \mathbf{p}_{i+1}, \mathbf{q}_{j+1} \,| \right)^2 \qquad (5)$$

where points $\mathbf{p}_i$ are on the path P and points $\mathbf{q}_j$ are on the path Q. In other words, if the distances between elements or phantom objects in consecutive pairs are similar, the two paths can be merged together.

4.8. Inverse Warping of Space Around Detected Paths

Once we detect arrangement paths we can further un-warp the space around these paths to extract more embedded symmetries.

To straighten a curved path we compute the geodesic distance from any point on the path to the head of the path as the x coordinate in the un-warped space. Only objects within a certain distance of the path need to be un-warped. For each sample point near the path we find the closest point on the path. The x coordinate of that point becomes the un-warped x coordinate of the sample point. The un-warped y coordinate is the distance of the sample point to the path.

Figure 9 shows an example of a pattern and its un-warped version. Notice that after straightening the path we are able to detect more symmetries and arrangements.

5. Results and Discussion

For testing our algorithm we converted high resolution raster images of ornamental art into vector representation. In some simple examples we took individual elements from those converted patterns and arranged them by hand in Adobe Illustrator. More complex input patterns are shown in Figure 10 and 11. It took under 2 seconds on a Macbook Pro with 2.6 GHz Intel Core2 Duo to detect all symmetries and curvilinear arrangements.

When the patterns become much more complex than those

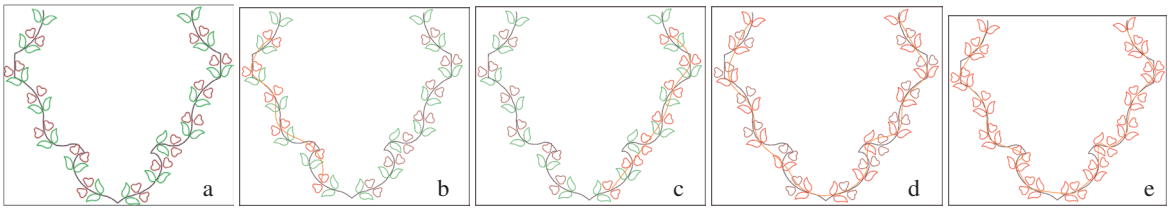

Figure 8: *Merging detected arrangement paths (b,c,d) with elements having constant distance into a single path (e).*

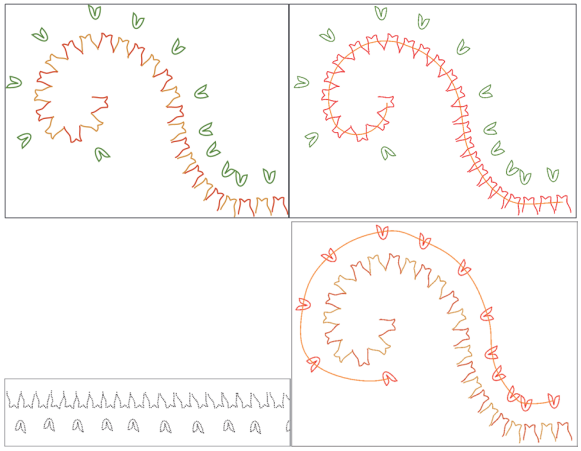

Figure 9: *Without un-warping the method detects only one arrangement path (top right). After the sample points are unwarped along the path (bottom right) the method detects an additional path (bottom right).*

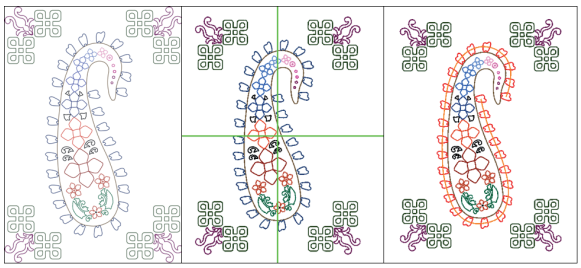

Figure 10: *Several reflectional symmetries and arrangements detected in a more complex pattern.*

shown in Figure 10 the matching of candidate sample points may be too slow for interactive editing. One possible solution is to apply the matching only to objects that are selected by the user, and to detect only symmetries and arrangements of these objects.

In our implementation, the signature tolerance s_ε in Section 4.2 is 0.1, the distance tolerance d_ε in Section 4.3 and in Eq 3 is 1/100 of the scene size. The window size for clustering in the centroid-specific transformation space was about 1/10 of the scene size. We found these values to work well for various examples we used.

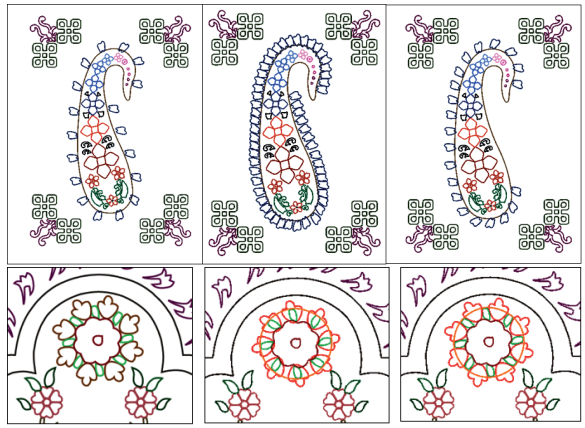

Figure 11: *Changing spacing of elements and moving elements along detected arrangement path.*

We have assumed that our scanned artwork is free of broken gaps and missing segments. Large missing segments at the end of element curves will add a considerable noise to the location of their centroids. If there is a gap in the middle of an element, the element may not be detected at all. In this paper we used vector art that originated from good quality raster images, and the vectorizing algorithm did not create such gaps. To be able to handle input of a lower quality we would have to bridge those gaps. For each segment we could search a neighborhood of its endpoints to see if there is another segment within a certain distance and a certain range of directions near the tangent at each endpoint.

5.1. Application

In this section, we demonstrate pattern editing using results obtained from our method.

Figures 11 and 12 show several input reference patterns, which contain elements arranged along curved paths. The system detects whether there are arrangements correspond-

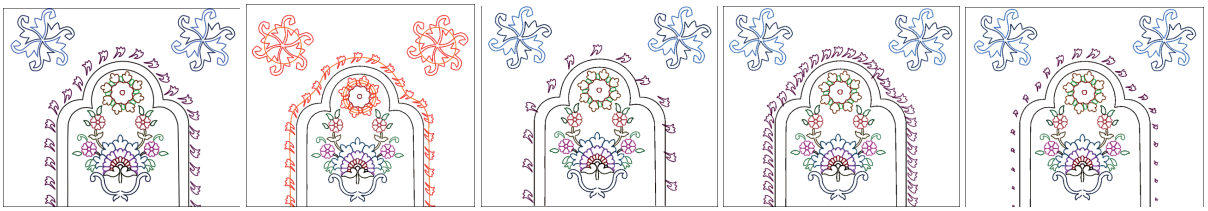

Figure 12: *The system detected arrangement along a curved path allowing the user to perform adjustment to element spacing, location, and scale.*

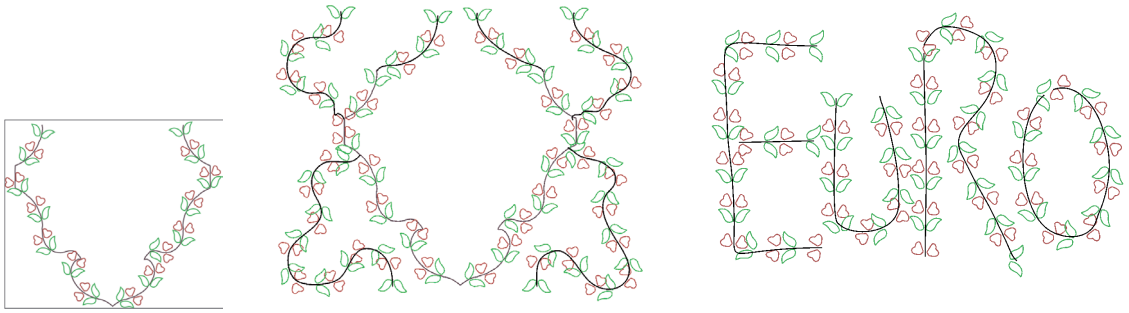

Figure 13: *The pairs of red and green elements in the input pattern (left) were recognized as a group by merging the underlying curves. They were used to brush over the input pattern and to create a new pattern.*

ing to the user selected point in the pattern. A set of possible operations are offered. The user can slide elements along the detected arrangement path as a group, change the distance between elements, or scale the elements further away from the picked one.

Notice that it is possible to generalize some basic symmetries, such as translational or rotational symmetry, as an arrangement along a line or a circle, respectively. Our system detects those cases, see the bottom of Figure 11, and allows the user to make the sliding or scaling edits on these arrangements as well.

In addition, we allow the user to extend the existing pattern or create new patterns using the elements or groups of elements learned from the input example. In the input pattern in the left of Figure 13 the system recognized several curves and merged them together, allowing the user to brush the two pairs of red and green symbols.

We view this application as a step toward inverse procedural modeling. Our system learned some rules in constructing the input pattern and it allows the user to extend the pattern using the same rules. In this case the rule is based on a group of symbols being distributed along a path in a certain pattern.

6. Conclusion and Future Work

In this paper we presented a technique for detecting symmetries and curvilinear arrangements in vector art. The technique can detect not only regular symmetries, such as the reflectional, translational or rotational symmetry, or scaling and their combinations, but also various arrangements along arbitrary curved paths. The technique also detects individual elements that are being transformed along curved paths.

Our method uses transformation spaces that represent potential symmetries to find the suitable candidate symmetries, similarly to existing techniques. We use a different approach when populating transformation spaces, though. We adaptively distribute sample points on the input art, based on the curvature, and we test neighborhood sample points before inserting a point into the transformation space.

To detect arrangements along a curved path, we group points that can be transformed to another group of points and mark them as elements. We also create phantom objects between mirror symmetric elements. We use centroids of elements or the phantom objects to find clusters in centroid-specific transformation space.

We un-warp the arrangement paths to detect elements that may be further away from the curve and whose spacing is too irregular for direct path detection. We can also detect elements that have similar paths and group them.

To demonstrate the applicability of the framework to facilitate easy editing of complex patterns with curvilinear arrangements we introduce several editing operations that could not be possible without the element and arrange-

ment detection. The system automatically learns the rules for placement of symbols along curved paths and it allows the user to recreate parts of the input patterns using a brushing tool.

One limitation of the technique is its speed for very complex patterns. Another limitation is the dependency on the quality of the input art. If the scanned image is poor, for example, with many discontinuities for thin lines, we may fail to detect elements with our greedy algorithm. One possibility of handling more challenging examples and addressing the issue of gaps in the input art would be to let the user select the initial element, possibly containing also disjoint parts. First, it would allow us to expand the definition of an element to include gaps between segments and second, the algorithm would not have to pair as many matching element centroids. Another approach, suggested in Section 5, would try to bridge short gaps.

Our technique for detecting arrangement paths can fail when the elements follow a curved path but their centroids are too far from the path, for example for a T shape, and there are no mirror elements along the path. We would like to investigate some optimization techniques that would allow us to estimate a path that is at a certain fixed distance away from the centroid.

Another interesting area of future research is to extend the idea of phantom objects to a hierarchies of symmetries along curved paths, as mentioned in Section 4.5.

In addition to improvements in speed and robustness we would like to extend the technique to 3D and apply the resulting symmetry and arrangement graph for creation of a similar or bigger version of the pattern, similarly to work by Ijiri *et al.* [IMIM08].

Acknowledgments

We thank Gavin Miller, Qi-xing Huang, and Nathan Carr for insightful discussion and advice. We would also like to thank the anonymous reviewers for their useful comments. Artistic elements used in the examples of this paper are a courtesy of Honesty Publishers & Distributors, Mumbai, India.

References

[AMWW88] ALT H., MEHLHORN K., WAGENER H., WELZL E.: Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom. 3*, 3 (1988), 237–256.

[Ata85] ATALLAH M. J.: On symmetry detection. *IEEE Trans. Computers 34*, 7 (1985), 663–666.

[DoC76] DOCARMO M.: *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (1981), 381–395.

[IMIM08] IJIRI T., MECH R., IGARASHI T., MILLER G.: An example-based procedural system for element arrangement. *Computer Graphics Forum 27*, 2 (2008), 429–436.

[KFR03] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 156–164.

[KFR04] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S.: Symmetry descriptors and 3d shape matching. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM, pp. 115–123.

[Kle93] KLEIN F.: Vergleichende betrachtungen ber neuere geometrische forschungen. *Mathematische Annalen 43* (1893).

[KS06] KELLER Y., SHKOLNISKY Y.: A signal processing approach to symmetry detection. *IEEE Transactions on Image Processing 15*, 8 (2006), 2198–2207.

[LCL08] LEE S., COLLINS R., LIU Y.: Rotation symmetry group detection via frequency analysis of frieze-expansions. In *Proceedings of CVPR 2008* (June 2008). (to appear).

[LE06] LOY G., EKLUNDH J.-O.: Detecting symmetry and symmetric constellations of features. In *ECCV (2)* (2006), pp. 508–521.

[LLM07] LI M., LANGBEIN F. C., MARTIN R. R.: Detecting approximate incomplete symmetries in discrete point sets. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2007), ACM, pp. 335–340.

[LLM08] LI M., LANGBEIN F. C., MARTIN R. R.: Detecting approximate symmetries of discrete point subsets. *Comput. Aided Des. 40*, 1 (2008), 76–93.

[LMLR07] LIU S., MARTIN R. R., LANGBEIN F. C., ROSIN P. L.: Segmenting periodic reliefs on triangle meshes. In *IMA Conference on the Mathematics of Surfaces* (2007), pp. 290–306.

[MGP06] MITRA N. J., GUIBAS L. J., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph. 25*, 3 (2006), 560–568.

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L. J.: Discovering structural regularity in 3d geometry. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–11.

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3d shapes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 549–559.

[SKS06] SIMARI P., KALOGERAKIS E., SINGH K.: Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 111–119.

[WWV85] WOLTER J., WOO T., VOLZ R.: Optimal algorithms for symmetry detection in two and three dimensions. 37–48.

[ZPA95] ZABRODSKY H., PELEG S., AVNIR D.: Symmetry as a continuous feature. *IEEE Trans. Pattern Anal. Mach. Intell. 17*, 12 (1995), 1154–1166.