Inverse Skeletal Strokes

Dongwei Liu* and Reinhard Klette

The *.enpeda..* Project, Department of Computer Science
The University of Auckland, New Zealand
dliu697@aucklanduni.ac.nz

Abstract. The skeletal stroke method is a general brush tool which can take a straight vector artwork as "ink". It is easy to apply, but it is limited by the requirement of straight inputs. To offer additional input options, we present *inverse skeletal strokes*, a method for straightening warped vector artworks.

Our method takes a user stroke to help understanding the structure of an input artwork. The key-idea is finding a set of arcs which show the "directional trend" of the artwork, and map the artwork into a new version in which these arcs are straightened.

We propose a measure representing the degree of parallelism between two arcs. Using this measure, we select a set of arcs from the input artwork which are approximately parallel to the given user stroke. This is a condensed representation of a user's intention. Then we transform the user stroke with the goal to maximize the degree of parallelism to each of the selected approximately parallel arcs. At last, we parametrize the artwork with respect to the optimized stroke, and map it into a straight version.

Keywords: Skeletal strokes, artwork, straightening of patterns, parallelism.

1 Introduction

The *skeletal stroke method* has been suggested in [6] for the design of 2-dimensional (2D) vector graphics. The method parametrizes a given artwork along a straight line, and then maps it onto a curved path. The skeletal stroke method serves as a general brush tool in commercial vector drawing applications, such as *Microsoft Expression Design* or *Adobe Illustrator*. See Fig. 1, left, for an example.

The skeletal stroke method is based on theories which have been developed some years ago, such as procedurally generating repeated border pattern [5], or mapping geometric objects along curved paths [3]. Later on, those ideas were improved to deal with more complex cases of given artwork and curved paths [6,7,2].

The skeletal stroke method is easy to apply, but there is a limited set of possible straight inputs. Therefore, it is desirable to have a method that maps warped

* Corresponding author.

F. Huang and A. Sugimoto (Eds.): PSIVT 2013 Workshops, LNCS 8334, pp. 1–11, 2014.

artworks into straight ones. This is illustrated in Fig. 1, right. Components of existing graphic designs can thus be used by the skeletal stroke method.

Difficulties for straightening warped artwork arise because the desired geometrical transform depends on the given artwork and users intentions. Image deformation methods allow users to warp pictures, for example, by dragging handles [4], or by deforming an envelope around a pictures [8]. These methods can freely bend pictures while having invariance properties with respect to details. But these methods are not designed for straightening artwork, and straightening would require complex user interactions.

The key to straighten a warped artwork is to find out a proper "backbone path". Medial axis methods [10,1] extract skeletons of shapes, but these skeletons are not the "representative backbone" we need. First, skeletons obtained by medial axis methods have branches or non-smooth arcs which are "misleading" for defining a proper representation. Second, the skeletons only depends on the outline of a shape, and drawn textures in the interior (i.e., the artwork) is not considered for skeleton extraction.

In this paper, we present a way to generate *inverse skeletal strokes*. Our method finds the latent backbone of an artwork and maps it into a straightened version. At a general level, geometric transforms (e.g. affine, perspective) are defined by classes of invariance properties, as outlined by Felix Klein in 1872 in his *Erlangen Program*. Invariance properties for inverse skeletal strokes can be postulated by incidence invariance and that length ratios should be kept locally approximately constant. However, the second constraint is not given in a strict mathematical sense. These comments should only indicate that the presented mapping is actually in a space between geometric and topological transforms.

The "straightness" of artwork is a subjective concept. Thus we take a user stroke as input to help in understanding the structure of the given artwork. Note that such an user stroke is drawn as a sketch, not accurate, and thus not yet the possible input for a precise mapping of artwork.

We observe that for a great proportion of artwork, there is a set of potential arcs in an artwork which indicate a directional trend of it. If we parametrize such an artwork by an arc a which is approximately parallel to such a subset of arcs, we can then map the artwork into a new version in which these arcs are straightened, i.e. the chosen directional trend of the artwork is straightened. Thus, this arc a would be a proper backbone for the artwork.

Fig. 1. *Left*: The skeletal stroke method maps a straight artwork onto a curved path (here: four times). *Right*: The inverse skeletal stroke method maps a curved artwork into a straight version.

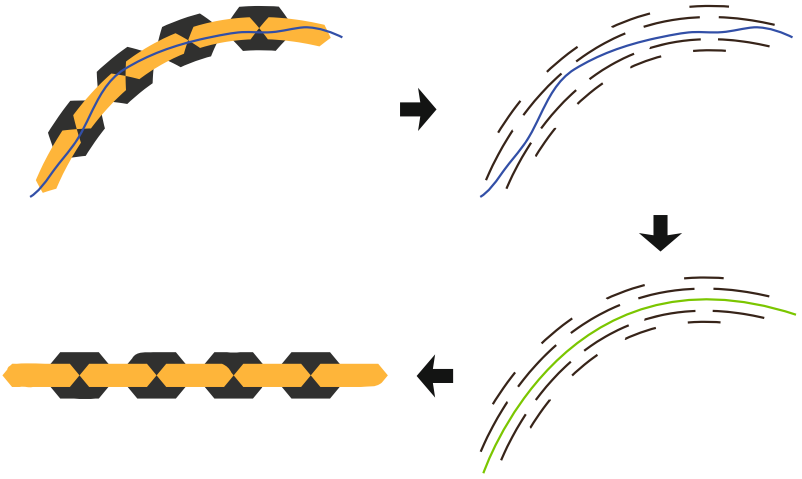

Fig. 2. The work flow of our method. From top-left following arrows: *(1)* Input an artwork and a user stroke. The user stroke is shown as a blue arc. *(2)* Extract a set of potential arcs that are approximately parallel to the user stroke. *(3)* Transform the user stroke into an arc a which is more parallel to the chosen set of arcs. *(4)* Map the artwork into a straightened version.

Therefore, we first extract a set of arcs from a given artwork, which are approximately parallel to the provided user stroke. This represents a user's intention. Then, we transform the user stroke into one which is as parallel as possible to all those selected arcs. At last, we parametrize the input artwork with respect to this optimized stroke, and map the whole artwork into a straightened version. Figure 2 shows the workflow of our method.

The paper is structured as follows. A formal description of artwork, arcs, and arc parallelism are given in Section 2. Then Section 3 provides detail process of our inverse skeletal strokes method. Experiments results are shown and discussed in Section 4. At last Section 5 concludes.

2 Basic Concepts and Notations

Before discussing the process of inverse skeletal strokes method, we first give a formal description of some related concepts.

2.1 Vector Artwork and User Stroke

We define *artwork*, denoted by U, as a vector picture in a 2D real space. An artwork $U = \{u_1, u_2, \ldots\}$ is composed of one or several individual graphic units. A *graphic unit* (or short, a *unit*) can be described by a simple curve (i.e. a Jordan curve [9]) that defines the outline, and a set of display features such as filling

style or colour. In this paper we only modify the outline (i.e. the curve defining the unit), not the display features.

We use cubic Bézier curves in a parametric form, defined as follows:

$$b(t) = p_0(1-t)^3 + 3p_1t(1-t)^2 + 3p_2t^2(1-t) + p_3t^3$$

for $t \in [0,1]$, *end points* p_0 and p_3 of the curve, and *control points* p_1 and p_2. Curves are considered in the real plane $\mathbb{R}^2$.

An outline of a unit u can be represented by a closed sequence of cubic Bézier curves $b_1, b_2, \ldots, b_m$ with end points $p_{0,i}$ and $p_{3,i}$, and control points $p_{1,i}$ and $p_{2,i}$, for $i = 1, \ldots, m$, where $p_{3,i} = p_{0,i+1}$, for $i = 1, \ldots, m-1$, and $p_{3,m} = p_{0,1}$.

A connected part of an outline of a unit is an arc. Such an arc a is a sequence of subsequent cubic Bézier curves, i.e. a subsequence of the sequence $b_1, b_2, \ldots, b_m$ for the whole outline. A *stroke* a_s of a user is also assumed to be such an arc. We also assume that a_s is smooth and of sufficient length on both side.

2.2 Arc Parallelism

Two arcs a_1 and a_2 in 2D space are *parallel* if a_1 is an envelope of congruent circles centred on a_2. See Fig. 3, left, for an example. The figure shows on the right a translation of one arc into another one; this does not define parallelism in general. Being parallel is an equivalence relation on the set of arcs in a 2D plane.

Following this definition, we propose a measure representing the *degree of parallelism* between two arcs a_1 and a_2.

Our motivation is that a_1 is an arc in an artwork, and a_2 is a user stroke. Thus, we assume that a_2 is of sufficient length on both sides, and we only consider the parallelism between a_1 and a subarc $b_2 \subseteq a_2$.

Arc a_1 defines a subarc b_2 of a_2 as follows: Assume an orientation for arc a_1 and, accordingly, a tangential vector t_q for any point $q \in a_1$. At an endpoint q of a_1 we use a one-sided derivative along a_1 for defining t_q. We denote $\overrightarrow{qp}$ a vector from $q \in a_1$ to $p \in a_2$ which is perpendicular to t_q. Let q_b and q_e be the

Fig. 3. Arc parallelism. *Left:* An arc, congruent circles centred on this arc, and an envelope parallel to the given arc. *Right:* Two arcs defined by a translation of one arc into the other; those two arcs are not parallel.

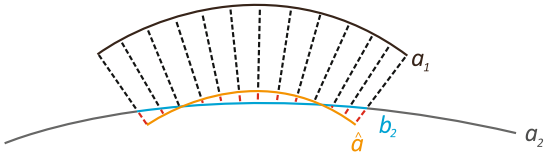

Fig. 4. Measure of arc parallelism. Blue subarc $b_2 \subseteq a_2$ is defined by two end points of a_1. A third arc $\hat{a}$, parallel to a_1, is defined by congruent circles of radius $\mu_d(b_2, a_1)$. The difference between $\hat{a}$ and b_2 is the proposed measure for the degree of parallelism between a_1 and a_2.

two endpoints of a_1, and p_b and p_e be the corresponding points on a_2. Then, p_b and p_e are the endpoints of b_2. We say b_2 is the *impacted region* of a_1 on a_2. See Fig. 4.

Let p be an arbitrary point on b_2, and $d(p, q)$ be the minimum Euclidean distance from p to a_1, which can be represented by a point $q \in a_1$. For a point $q \in a_1$, the vectors t_q and $\overrightarrow{qp}$ also define the sign of $d(p, q)$. If t_q and $\overrightarrow{qp}$ describe a left-hand orientation, we denote the sign of $d(p, q)$ by $o(q) = 1$, otherwise we have $o(q) = -1$.

We denote by $\mu_d(b_2, a_1) = E[d(p, q) \cdot o(q)]$ the *expected value* of $d(p, q)$, for all $p \in b_2$. We call this the *mean distance* between b_2 and a_1. The negative or positive of $\mu_d(b_2, a_1)$ defines whether b_2 is on the *negative* or *positive side* of a_1. (If equal to zero then we also consider it on the positive side.)

Now consider a third arc $\hat{a}_1$ that is parallel to a_1, and on the same side (i.e. negative or positive) of a_2 relatively to a_1. Further assume that the parallelism of $\hat{a}$ with a_1 is defined by congruent circles of radius $\mu_d(b_2, a_1)$. See Fig. 4. Thus, we have that

$$\mu_d(b_2, \hat{a}) = E[d(p, \hat{q}) \cdot o(\hat{q})] = 0$$

for all $p \in b_2$, where $\hat{q} \in \hat{a}$ defines the minimum distance $d(p, \hat{q})$ as above, $o(\hat{q})$ defines the sign of $d(p, \hat{q})$. Informally speaking, arc $\hat{a}$ is a modification of b_2 which follows parallel to a_1, i.e. if a_1 and b_2 are parallel, $\hat{a}$ would overlap with b_2. We call $\hat{a}$ an *auxiliary arc* of a_1.

Now we calculate $\sigma_d(b_2, \hat{a})$, the *variance* of $d(p, \hat{q})$.

$$\begin{aligned}\sigma_d(b_2, \hat{a}) &= E[d(p, \hat{q})^2 \cdot o(\hat{q})^2] - \mu_d^2(b_2, \hat{a}) \\ &= E[d(p, \hat{q})^2 \cdot o(\hat{q})^2]\end{aligned}$$

If a_1 and b_2 are parallel, there would be that $d(p, \hat{q}) = \sigma_d(b_2, \hat{a}) = 0$. Otherwise, $\sigma_d(b_2, \hat{a})$ will increase as $\hat{a}$ becomes more and more different to a_2. Thus, we can use $\sigma_d(b_2, \hat{a})$ to measure the parallelism of a_1 and a_2.

According to the properties of parallel arcs, any line perpendicular to $\hat{a}$ is also a line which is perpendicular to a_1. Now assume q is the corresponding points

of $\hat{q}$ on a_1, we have that

$$\begin{aligned} d(p,\hat{q}) \cdot o(\hat{q}) &= d(p,q) \cdot o(q) - \mu_d(b_2, a_1) \\ \sigma_d(b_2,\hat{a}) &= E[d(p,\hat{q})^2 \cdot o(\hat{q})^2] \\ &= E[d(p,q)^2] - 2 \cdot E[d(p,q)] \cdot \mu_d(b_2,a_1) + \mu_d(b_2,a_1)^2 \\ &= E[d(p,q)^2] - \mu_d(b_2,a_1)^2 = \sigma_d(b_2,a_1) \end{aligned}$$

Thus, the variance $\sigma_d(b_2, a_1)$ of $d(p, q)$ is just our measurement of the degree of parallelism between a_1 and a_2.

So far we discussed continuous geometric entities in the Euclidean plane. For implementing the ideas, we apply the following *discretization*. For a set a, $\langle a \rangle$ denotes a discrete representation of this set.

We uniformly (with respect to a fixed arc length [9] increment $\triangle\mathcal{L}$) sample an arc b_2, and obtain a set of *samples* $\langle b_2 \rangle = \{p_i : i = 1, \ldots, n\} \subset b_2$. Let $d(p_i, \hat{q}_i)$ denote the minimum Euclidean distance from $p_i \in \langle b_2 \rangle$ to $\hat{a}$, where $\hat{q}_i \in \hat{a}$, for $i = 1, \ldots, n$. Then we use the mean $M[d(p_i, \hat{q}_i)^2]$ as an approximation of $E[d(p, \hat{q})^2]$, and the variance $V[d(p_i, \hat{q}_i)] = \sum_{i=1}^{n} (d(p_i, \hat{q}_i) - M[d(p_i, \hat{q}_i)])^2$ as an approximation of $\sigma_d(b_2, a_1)$.

3 Inverse Skeletal Strokes

Our system takes an artwork U_s and a user stroke a_s as input, where subscript s indicates "source". We output a straightened version of artwork U_s, denoted by U_t, where t indicates "target".

3.1 The Algorithm

Our process involves four steps. First, we extract a family A of finitely many candidate arcs from the units in U_s, which are possible to approximately parallel to a_s. Second, we measure the degree of parallelism between a_s and each arc $a \in A$, and select a set of arcs $A_p \subseteq A$ which are approximately parallel to a_s. Third, we optimize a_s according to A_p and obtain a backbone a_t, making a_t as parallel as possible to arcs in A_p. At last, we parametrize U_s with respect to a_t, and map it into a straightened version U_t.

The following subsection describe the steps in detail.

3.2 Extract Candidate Arcs

We extract candidate arcs from the outlines of units. As defined in Section 2.1, a unit $u \in U_s$ is described by a simple curve. For each unit $u \in U_s$, we cut it into several arcs by *extremum points* and *corner points* as defined following. Then we select the arcs with sufficient arc length as candidate arcs.

Let p be a point on the outline of u, and $d(p, q)$ be the minimum Euclidean distance from p to a_s, which can be represented by a point $q \in a_s$. We construct

a function l with values $l(q) \in [0, 1]$ to denote the location of q on a_s as follows:

$$l(q) = \frac{\mathcal{L}(q_b, q)}{\mathcal{L}(q_b, q_e)}$$

Here q_b and q_e are the begin point and end point of a_s, $\mathcal{L}(q_1, q_2)$ is the arc length from q_1 to q_2 along a_s. Let p be an argument, then $q = q(p)$ and $l(q) = lq(p)$ are functionally dependent on p. We call a point p an *extreme point* if $lq(p)$ is an extreme values.

A point $p \in u$ is a *corner point* if the two one-sided derivatives are not equal at this point.

We cut u into several arcs by *extreme points* and *corner points*. For an arc a with two end point p_1 and p_2, if $\mathcal{L}(p_1, p_2) > C \cdot \mathcal{L}(u)$, we extract the arc of u between p_1 and p_2 as a candidate arc. Here $\mathcal{L}(u)$ is the arc length of the outline of u, C is a threshold ratio of empiric. We use $C = 0.1$. The arcs of insufficient length are ignored, because they are not potential to show the directional trend of the given artwork.

We denoted by A all candidate arcs extracted from each unit $u \in U_s$.

3.3 Select Parallel Arcs

Now we get a set A of candidate arcs. From A, we detect the arcs that are approximately parallel to the user stroke a_s.

For each arc a_i, we calculate the parallelism measurement $\sigma_d(b_i, a_s)$ defined in Section 2.2. If $\sigma_{i,s} < w \cdot \mathcal{L}(b_i)$, we consider a_i is a *parallel arc* of a_s. Here the threshold is formed by two parts: b_i is the impacted region of a_i on a_s, and $\mathcal{L}(b_i)$ is the arc length of b_i (for standardization); w is an empirical parameter which control the accuracy, here $w = 0.02$. We denoted by A_p all parallel arcs in A. Figure 5 shows the process of extracting approximately parallel arcs.

Fig. 5. Extract approximately parallel arcs. *Left:* A given user stroke and a unit. *Middle:* We cut the outline of the unit into four candidate arcs according to extreme points and corner points. *Right:* We detect two arcs that are approximately parallel to the user stroke.

3.4 Optimize User Stroke

As we defined in Section 2.2, arc a_s would be increasingly parallel to an arc $a \in A_p$, if the corresponding subarc b_s moves more and more close to the auxiliary arc $\hat{a}$.

For each arc $a \in A_p$, we calculate the auxiliary arc $\hat{a}$ parallel to a, and the impacted region b_s on a_s. For a point $p \in a_s$, we make a line l_p that is perpendicular to a_s on p. Then, we move p to $p_t \in l_p$ which has the minimum sum squared distance $\sum d(p_t, \hat{q}_t)^2$ to all the $\hat{a}$ that affect p, where $\hat{q}_t \in \hat{a}$ indicate the minimum distance from p to $\hat{a}$. See Fig. 6 for an intuitive example.

We remove the subarcs on the two ends of a_s which are not affected by any $a \in A_p$. Then we extend the two ends of a_s with line segments following the one-sided derivative. For a subarc in the middle of a_s which is not affected by any $a \in A_p$, we apply proper transformation that make it continue to the two neighbor subarcs. Here we call the optimized stroke a_t a backbone path.

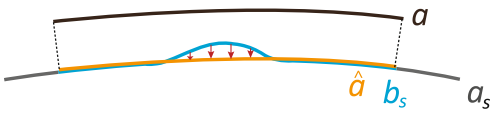

Fig. 6. Optimize the user stroke. Here a_s is a given user stroke, a is a parallel arc of a_s. We make an auxiliary arc $\hat{a}$ of a, and transform the impacted region b_s of a_s toward $\hat{a}$. Thus, the optimized stroke can be as parallel as possible to a.

3.5 Map Artwork to Straightened Version

Given an artwork $U_r = u_i, i = 1, \ldots, n$ and a backbone a_t, we use an inverse method of skeletal strokes [6] to parametrize the artwork U, and map it along a straight path. Informally speaking, we deform the 2D space around a_t, making a_t a straignt line in the deformed space. See Fig. 7 for an intuitive expression.

Suppose p_f the start point of a_t. For a point $p \in U_r$, we note point $q \in a_t$ as the point on a_t with the minim Euclidean distance $d(p, q)$ from q to p. We note $\mathcal{L}(p_f, q)$ the arc length from p_f to q along a_t. Then p is parametrize by a coordinate $(\mathcal{L}(p_f, q), d(p, q))$.

Then we map U_r to U_t along a straight path $y = 0$. For any point $p_r \in U_r$, the coordinate of the corresponding point $p_t \in U_t$ is the same with the parameter coordinate of p_r along a_b.

The shape of a vector artwork is represented by a set of finite many Bézier curves. For mapping a vector artwork, we only have to map these Bézier curves. For each Bézier curve $b_r(t) \in U_r$, first we sample four points on it at $p_{r,0} = b_r(0)$, $p_{r,1} = b_r(1/3)$, $p_{r,2} = b_r(2/3)$, and $p_{r,3} = b_r(1)$. Then, we map these points to $p_{t,0}$, $p_{t,1}$, $p_{t,2}$, and $p_{t,3}$. At last we use a new Bézier curve $b_t(t)$ to fit the four mapped points $p_{t,1}$, $p_{t,2}$, $p_{t,3}$, and $p_{t,4}$. Here $b_t(t)$ is the mapped version of $b_r(t)$.

4 Experiments

We implement our method with C++ and Qt 5.0.1. These experiments are made on a PC with Intel Core i5-3550 3.3GHz and RAM of 8GB. The run time for

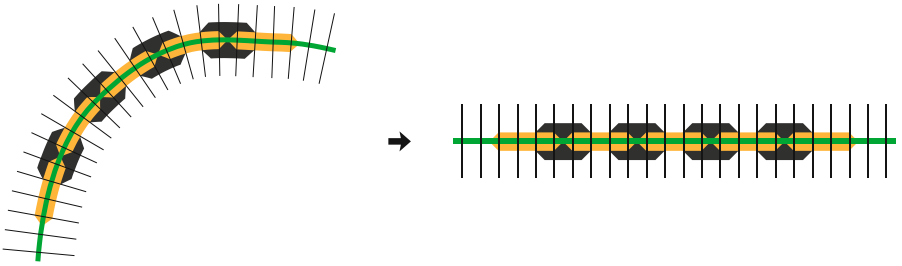

Fig. 7. Parametrize an artwork by a backbone, and map the artwork to a straight version

each experiment in Fig. 8 is less then 5 seconds, and for each experiment in Fig. 9 is less then 9 seconds.

Two examples of experimental results with simple input patterns are shown in Fig. 8. The input patterns are highly regular, thus the results show the accuracy of our method.

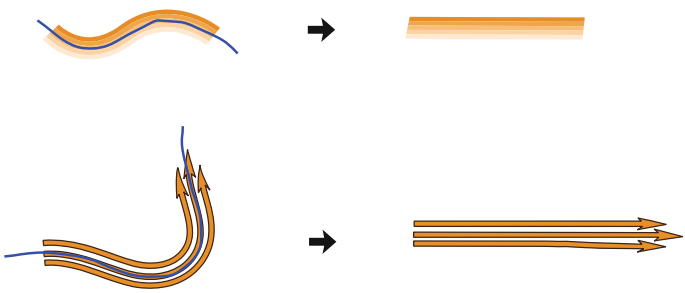

Fig. 8. Experimental results when having simple patterns as inputs. *Left*: Input artworks and user strokes. *Right*: Straightened artworks.

Experimental results of more complex cases are shown in Figs. 9 From these results we can see that our method is robust for handling complex artworks in practical application.

As we described in Section 1, an original user stroke is only drawn as a sketch, and is not accurate enough for straightening an given artwork. Figure 10 compares the the mapping results using an original user stroke and the optimized stroke. Using the same input artwork and user stroke, the output artwork mapped with the optimized stroke is straightened, but the version mapped with the original user stroke still contains undesirable curvatures.

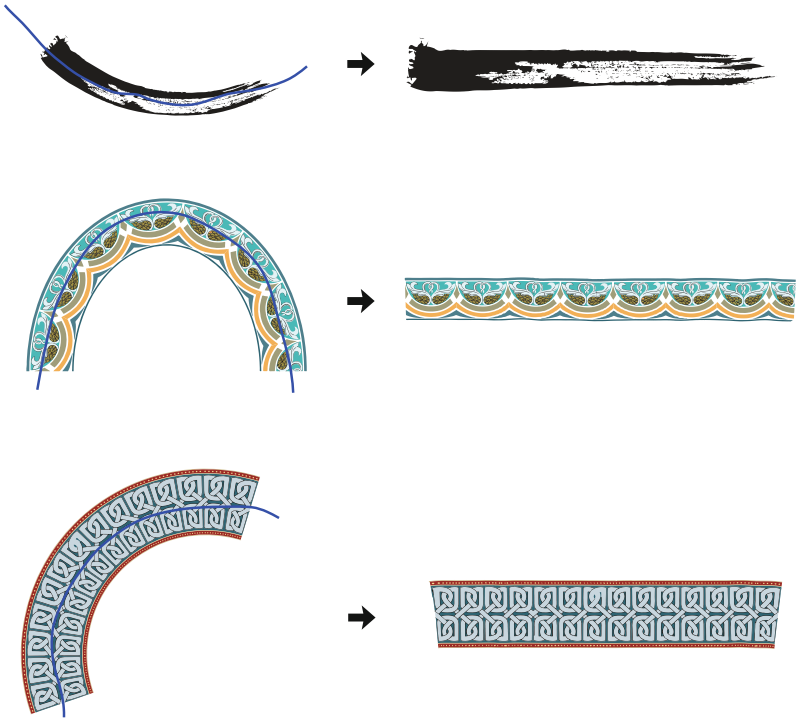

Fig. 9. Experimental results when having more complex artwork as inputs

Fig. 10. Comparison of mapping a given artwork with an original user stroke and the optimized stroke. *Left:* An artwork, an original user stroke (blue arc), and the optimized stroke (green arc). *Middle:* A version of the given artwork mapped along the original user stroke. *Right:* Another version of the given artwork mapped along the the optimized stroke.

As we discussed in section 1, our method assume that there are some arcs in the given artwork which indicate a directional trend of the artwork. This assumption is an limitation of our method.

5 Conclusions

The paper proposed a novel theoretical framework (e.g. discussion of parallel arcs) and a novel algorithm for straightening vector artworks. This is the inverse process to the known skeletal stroke method.

Guided by a user stroke, the provided method extract a set of potential arcs in a given artwork which indicate a directional trend of it. Then the user stroke is transformed into one which is as parallel as possible to all those selected arcs. At last the given artwork is parameterized with the optimized stroke, and mapped into a straightened version.

The provided method can be used as a technique for generating proper inputs for the skeletal stroke method. Thus, it can broaden the range of application of the skeletal stroke method.

Acknowledgment. This paper is supported by China Scholarship council.

References

1. Aichholzer, O., Aigner, W., Aurenhammer, F., Hackl, T., Jüttler, B., Rabl, M.: Medial axis computation for planar free–form shapes. Computer-Aided Design 41, 339–349 (2009)
2. Asente, P.: Folding avoidance in skeletal strokes. In: Proc. Sketch-Based Interfaces and Modeling Symposium, pp. 33–40 (2010)
3. Barr, A.: Global and local deformations of solid primitives. In: Proc. ACM SIGGRAPH, pp. 21–30 (1984)
4. Barrett, W., Cheney, A.: Object-based image editing. In: Proc. ACM SIGGRAPH, pp. 777–784 (2002)
5. Beach, R., Stone, M.: Graphical style towards high quality illustrations. In: Proc. ACM SIGGRAPH, pp. 127–135 (1983)
6. Hsu, S., Lee, I., Wiseman, N.: Skeletal strokes. In: Proc. ACM Symposium on User Interface Software and Technology, pp. 197–206 (1993)
7. Hsu, S., Lee, I.: Drawing and animation using skeletal strokes. In: Proc. ACM SIGGRAPH, pp. 109–118 (1994)
8. Karni, Z., Freedman, D., Gotsman, C.: Energy based image deformation. Computer Graphics Forum 28, 1257–1268 (2009)
9. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Picture Analysis. Morgan Kaufmann, San Francisco (2004)
10. Lee, D.: Medial axis transformation of a planar shape. IEEE Trans. Pattern Analysis and Machine Intelligence 4, 363–369 (1982)