Efficient C^2-Weighting for Image Warping

Chuhua Xian*, Shuo Jin* and Charlie C. L. Wang† *Senior Member, IEEE*

***Abstract*—Handle-driven image warping based on linear blending is widely used in many applications because of its merits on intuitiveness, efficiency and easiness of implementation. In this paper, we develop a method to compute high-quality weights within a closed domain for image warping. The property of C^2-continuity in weights is guaranteed by the carefully formulated basis functions. The efficiency of our algorithm is ensured by a closed-form formulation of the computation for weights. The cost of inserting a new handle is only the time to evaluate the distances from the new handle to all other sample points in the domain. A virtual handle insertion algorithm is developed to allow users to freely place handles within the domain while preserving the satisfaction of all expected criteria on weights for linear blending. Experimental examples for real-time applications are shown to demonstrate the effectiveness of this method.**

***Index Terms*—Image warping, closed-form formulation, weighting, linear blending**

I. Introduction

IMAGE warping based on linear blending has a wide popularity as it is intuitive, effective and easy-to-implement in many different scenarios of applications. By inserting handles within an image domain, users can bind an image Ω with the handles and then manipulate their locations and orientations to drive the deformation of Ω. Specifically, each handle H_i with $i = 1, \ldots, m$ is defined as a local frame with its origin $\mathbf{h}_i \in \Omega$. After defining an affine transformation $\mathbf{T}_i$ for each handle H_i, the deformation of Ω is realized by computing the new position of each point $\mathbf{p} \in \Omega$ via a linear blending of affine transformations $\mathbf{T}_i\mathbf{p}$. The linear blending is weighted by fields $w_i : \Omega \mapsto \Re$ associated with handles H_i. Basically, to achieve an intuitive and high-quality deformation, a few criteria on the weights are demanded, including smoothness, non-negativity, partition-of-unity, locality/sparsity, and nonlocal maxima/minima (see [1] for an analysis).

The recent advancement of technology focuses on computing weights of blending for warping with a discrete form of domain (i.e., meshes are employed to determine piecewise linear fields of weights). Weights are computed on mesh nodes via minimizing some discrete differential energies (e.g., biharmonic, triharmonic and quatraharmonic forms used in [2]). After incorporating hard constraints in equality-form or inequality-form according to different criteria, weights are determined on mesh nodes with the help of non-linear optimization solvers, which is usually time-consuming. This leads to the difficulty in the realization of real-time insertion of new handles as new routines of non-linear optimization need to be taken. Moreover, the determined weights are mesh-dependent, which indicates an implicit requirement on mesh quality. For a poorly meshed domain, the artificial distortion caused by elements in poor shape could be serious (as illustrated in Fig.1). Although the artifacts can be reduced by remeshing the domain to improve mesh density and quality, this assumes high efficiency of weight generation. Otherwise, a very slow computation process will be disappointing to users. Ideally, the distribution of weights should be affected only by the domain to be warped and the locations of handles, which indicates mesh-independence. The existing mesh-independent approaches in literature for handle-driven deformation (e.g., [3]–[5]) can only satisfy subsets of the demanded properties on weights. These factors motivate us to investigate an efficient and high-quality method to determine weights which can be applied to both meshed and meshless representations of an image domain.

In this paper, we formulate the evaluation of weights in a closed-form so that the deformation framework gains the benefit of flexibility – i.e., the response of inserting new handles is real-time. Specifically, the time cost of inserting a new handle is linear to the number of samples used to represent the domain of computation. The properly chosen basis function according to the defined criteria can guarantee the properties of smoothness, non-negativity, partition-of-unity, locality/sparsity, and nonlocal maxima/minima, all of which are necessary to ensure a deformation of high quality.

The main results of our work are as follows:

- We present an efficient method to determine linear blending weights with C^2-continuity for real-time image warping. The weights are formulated in a closed-form of basis functions centered at the handles (details are given in Section III-B). After decomposing the domain to be deformed by the Voronoi diagram of handles, aforementioned criteria for image warping are all ensured (see the analysis in Section III-C).
- A virtual handle insertion algorithm is proposed in Section IV to guarantee the locality and sparsity of weights so that a deformation interpolates all transformations defined on handles. The virtual handles are added to let the supporting region of the basis function defined on a handle not cover the origins of any other handles (see the algorithm in Section IV-A).
- After constructing the Voronoi diagram of all handles (including user-input and virtual ones), its dual-graph gives a connectivity of the handles. We compute harmonic fields on the graph to determine the transformations of virtual handles according to the ones specified on the

First manuscript submitted in December 2015; revised manuscript submitted in March 2016; final manuscript accepted in May 2015.

*Joint First Authors. †Corresponding Author. C. Xian is with the Department of Computer Science, South China University of Technology, Guangdong, China. S. Jin is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. C.C.L. Wang is with the Department of Design Engineering, Delft University of Technology, The Netherlands. E-mail: c.c.wang@tudelft.nl

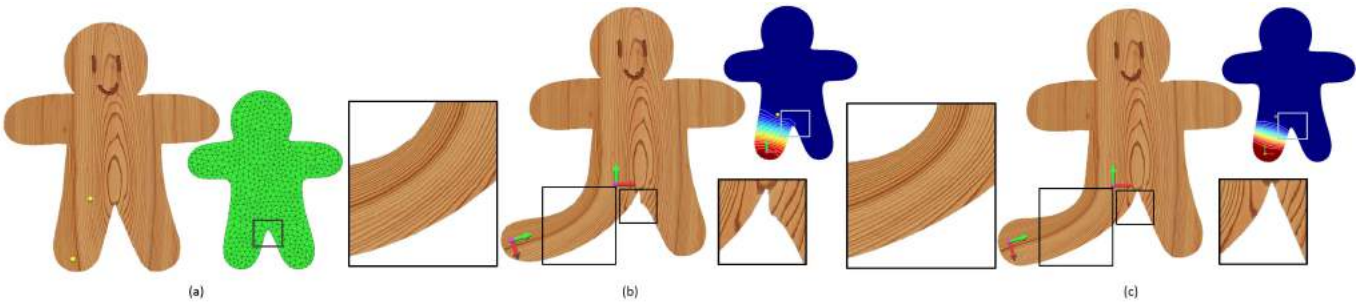

Fig. 1. Image Warping based on linear blending. (a) Handle-driven deformation based on linear blending is an intuitive method for the interactive shape manipulation. (b) Artifacts are found with the weights computed by the mesh-dependent method [1], which requires solving an equation system over the whole underlying mesh. (c) We propose a new framework to generate C^2-continuous weights for linear blending based deformation with high efficiency. Our approach inherits the merits of mesh-dependent weighting schemes while bringing the weighting to the resolution of infinity.

user-input handles (see Section IV-B). It is proved that the deformation determined in this way lead to a natural manner following the intention of user input well.

- For different discrete representations of the domain to be warped, we show that our framework is efficient and flexible for handle-driven image warping (see Section V for implementation details). Experimental results are shown in Section VI to demonstrate the performance of our approach.

II. Related Work

Shape deformation is an important research area in image manipulation and geometric modeling. There are a large amount of existing approaches in literature. The purpose of this section is not for a comprehensive review. We only focus on discussing the handle-driven deformation approaches.

Mesh-based techniques for discrete geometry modeling and processing have been widely explored in the past decade. Typical approaches including variational surface deformation , Poisson deformation, Laplacian editing and other linear variational surface deformation approaches (see a comprehensive survey in [6]). Volumetric information and rigidity are also incorporated to enhance the shape-preservation in [7], [8]. One common drawback of these approaches is that the positions of vertices on a model need to be determined by solving a system of linear equations after every update of handles, which becomes a bottleneck of computation. A recent development in [1], [2], [9] transfers the workload from online optimization to offline. In [5], the handles are elements of a simplified mesh. Although this strategy is more efficient than the deformation methods based on online optimization, they still cannot avoid solving large linear systems, which slows down the response of deformation after inserting new handles. Moreover, the results of deformation are also suffered from the artificial distortions caused by the problems of meshes (e.g., too coarse meshes for a fine deformation, a mesh with 'needle' and 'cap' triangles, and the problem of symmetry). Our approach solves these problems by providing closed-form formulas to generate weights preserving all the demanded properties for producing deformations with high quality in real-time.

Another thread of researches for deformation focuses on mesh-independent approaches. Different handles are employed for shape manipulation. Points are used in [10], and curves are employed as handles in [3]. Cage-based deformation (e.g., [11], [12]) can be considered as a improved generalization of grid-based deformation(e.g., [13]), where weights can be found by a closed-form in terms of the handles. However, the construction of cages is usually not automatic and the manipulation on cages instead of a model itself is indirect.

Moving least squares (MLS) strategy is employed in [10] to interpolate similarity/rigid deformations at handle points. A closed-form solution is provided in their approach to determine the transformation matrix on every point in a MLS manner. The transformations in the whole domain need to be computed when any handle is moved. There is no explicit determination on the influence region of each handle. Different from this MLS approach, our approach belongs to the category of linear blending based deformation. When the property of sparsity is preserved on weights, the deformation at a point is only affected by the nearby handles that is easier to be predicted by users. Moreover, the deformation determined by our approach is resolution-independent, which is very important for image manipulation.

The work of generating weights for linear blending also relates to the research of scattered data interpolation, where *radial basis functions* (RBF) are widely used. In [14], the deformation is governed by global RBFs that lead to a dense linear system to be solved. The weights determined by the dense (or global) data interpolation approaches lack of sparsity. Therefore, every point in the domain is changed when any handle is updated even if it is far away. Although the *compactly supported radial basis functions* (CSRBF) can help on introducing the sparsity, it does not provide closed-form formulas as our approach. An improved approach with interior RBF in [15] is able to obtain natural deformation on various models, which consists of a precomputation phase and an online deformation phase. By minimizing an energy functional to determine the coefficients of the IRBFs, the shape can be deformed in a shape and volume controlled manner. When adding or removing handles during manipulation, the precomputation phase needs to be redone, which could be slow if the shape is complex.

A recent work [16] raises a new deformation framework to design linear deformation subspaces, which generally unifies

linear blend skinning and generalized barycentric coordinates into an identical variational form. The deformation following the movement of user handles is formulated as the minimization of a quadratic energy subject to constraints from the handles on discrete meshes. The computational bottleneck of this approach falls on the factorization when new handles are inserted, which is overcome by using advanced numerical schemes. Our method particularly focuses on linear blending scheme, giving a closed-form solution to compute weighting which is applicable to any types of domain representations. It shares a common problem with [1], which is the specification of affine transformations at handle points. The framework in [16] has a lower barrier on this as less manipulation needs to be taken on handles.

III. Weighting Formulation

The key idea of the linear blending is to determine the new position of a point $\mathbf{p} \in \Omega$ by a linear combination of the transformations $\mathbf{T}_i$ defined by users on handles H_i as[1]

$$\mathbf{p}' = \sum_{i=1}^{m} w_i(\mathbf{p})\mathbf{T}_i\mathbf{p} \tag{1}$$

with $w_i(\cdot)$ being a scalar field of weights to be determined. The origin of a handle H_i is denoted by $\mathbf{h}_i$. This linear blending based formulation is popular as it is fast and easy-to-implement. However, carelessness in assigning weights can lead to visible artifacts in results, i.e. the quality of warped result is much dependent on the properties of $w_i(\cdot)$ (the analysis of our weighting is given in Section.III-C). Our formulation introduced below guarantees the generation of a C^2-continuous scalar field of weights in a closed domain, leading to a high-quality deformation based on linear blending.

A. Intrinsic Distance

Definition. All points on an image form a bounded domain Ω. For any two points $\{\mathbf{p}_s, \mathbf{p}_e\} \in \Omega$, if there exists a curve (a path) $\mathcal{C} \subset \Omega$ connecting $\mathbf{p}_s$ and $\mathbf{p}_e$, we define the intrinsic-distance of $\{\mathbf{p}_s, \mathbf{p}_e\}$ along the curve $\mathcal{C}$ as

$$d(\mathbf{p}_s, \mathbf{p}_e; \mathcal{C}) = \int_{\mathcal{C}} f \, ds$$

Then the intrinsic-distance of $\{\mathbf{p}_s, \mathbf{p}_e\}$ in Ω is defined as

$$d(\mathbf{p}_s, \mathbf{p}_e) = \inf_{\mathcal{C}} \int_{\mathcal{C}} f \, ds.$$

If $\mathbf{p}_s$ and $\mathbf{p}_e$ are not located in a connected region of Ω, then there is no path connecting these two points. In this case, the intrinsic distance is defined as $d(\mathbf{p}_s, \mathbf{p}_e) = \infty$. It is obvious that the intrinsic-distance has the following properties:

- **Existence:** $d(\mathbf{p}_s, \mathbf{p}_e; \mathcal{C})$ is always calculable once $\mathcal{C}$ is determined, which is the length of a curve segment in Ω. Therefore, $d(\mathbf{p}_s, \mathbf{p}_e)$ always exists for Ω when $\mathbf{p}_s$ and $\mathbf{p}_e$ are located in the same connected region.
- **Uniqueness:** $d(\mathbf{p}_s, \mathbf{p}_e)$ is uniquely determined while the corresponding paths may be multiple.

[1] $\mathbf{T}_i$ is a homogeneous matrix and $\mathbf{p}$ is represented by homogeneous coordinate.

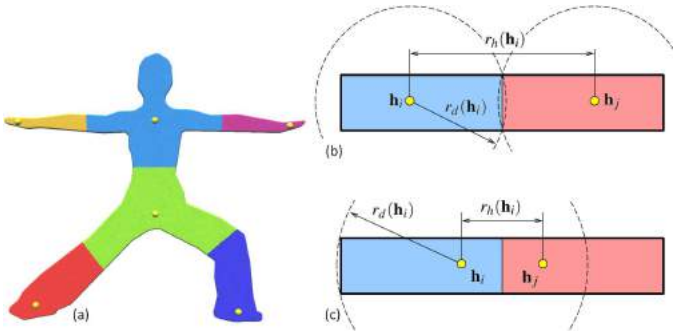

Fig. 2. Voronoi diagram based method to determine the size of local support. (a) The Voronoi diagram of handles can decompose Ω into smaller pieces. (b) The illustration of $r_h(\mathbf{h}_i)$ and $r_d(\mathbf{h}_i)$ in the Voronoi diagram. (c) Very close handles can lead to $r_h(\mathbf{h}_i) < r_d(\mathbf{h}_i)$.

Intrinsic distance approximation. For a graph $\mathcal{G}$ with a set of samples $\mathcal{S} \in \Omega$ as its nodes, we represent the shortest distance on $\mathcal{G}$ between $\mathbf{p}_s \in \mathcal{G}$ and $\mathbf{p}_e \in \mathcal{G}$ as $d_{\mathcal{G}}(\mathbf{p}_s, \mathbf{p}_e; \mathcal{S})$. For a given $\varepsilon > 0$, if for any two points $\{\mathbf{p}_s, \mathbf{p}_e\} \in \mathcal{S}$, we always have

$$|d_{\mathcal{G}}(\mathbf{p}_s, \mathbf{p}_e; \mathcal{S}) - d(\mathbf{p}_s, \mathbf{p}_e)| \leq \varepsilon,$$

then the sampling $\mathcal{S}$ is called a ε *error-bounded sampling* of Ω and $\mathcal{G}$ is its corresponding graph. $d_{\mathcal{G}}(\mathbf{p}_s, \mathbf{p}_e; \mathcal{S})$ is treated as an approximation of $d(\mathbf{p}_s, \mathbf{p}_e)$.

In this work, the computation of weights in practice relies on the approximate instrinsic distance. When a domain is convex, the intrinsic distance is equal to the Euclidean distance between any two points in it. It must be clarified that an error-bounded sampling $\mathcal{S}$ does not sufficiently indicate S is a high-quality approximation of the domain (see a counter example in Fig.4(a)), which also requires the sampling should be dense to a certain extent.

B. Formulation

Each handle H_i is equipped with a compactly supported basis function with a support size r_i as $\phi_i(d(\mathbf{p}, \mathbf{h}_i)/r_i)$, where $\mathbf{h}_i$ is the location of H_i and $d(\cdot, \cdot)$ returns the intrinsic-distance between two points inside Ω. The scalar field of weights for H_i is then defined as

$$w_i(\mathbf{p}) = \frac{\phi_i(d(\mathbf{p}, \mathbf{h}_i)/r_i)}{\sum_{j=1}^{m} \phi_j(d(\mathbf{p}, \mathbf{h}_j)/r_j)}. \tag{2}$$

To ensure the C^2-continuity over the whole domain Ω, we need to properly select the basis functions according to the following criteria:

1) $\phi_i(0) = 1$ and $\phi_i(t) = 0$, $\forall t \geq 1$;
2) $\phi_i'(t) < 0$;
3) $\phi_i''(t)$ is continuous for $0 < t < 1$;
4) $\phi_i'(0) = \phi_i'(1) = \phi_i''(0) = \phi_i''(1) = 0$.

A fulfillment of the above requirements on basis functions guarantees the C^2-continuity in the domain to be warped, leading to a smooth deformation following the transformations of the handles. When the intrinsic-distance is used to generate the input parameter t for the basis functions, linear blending based deformations driven by these basis functions behave in a

shape-aware manner. Another problem to be solved is how to determine the support size r_i of each basis function. As a basic requirement of handle-driven deformation based on linear blending, every point $\mathbf{p} \in \Omega$ should be influenced by at least one handle. To be shape-aware, a point $\mathbf{p}$ should be mostly affected by its closest handle in Ω. Voronoi diagram sited at the origins of handles $\{\mathbf{h}_i\}$ provides an intrinsic decomposition of Ω according to these observations (see Fig.2(a)), where the intrinsic-distance in Ω is used as the metric for generating the Voronoi diagram. We denote the cell that corresponds to $\mathbf{h}_i$ by $\mathcal{V}(\mathbf{h}_i)$. Two metrics with regard to a handle H_i can be defined as follows (see Fig.2(b) for an illustration):

- Size of a Voronoi cell: $r_d(\mathbf{h}_i) = \sup_{\mathbf{q} \in \mathcal{V}(\mathbf{h}_i)} d(\mathbf{q}, \mathbf{h}_i)$;
- Separation to other sites: $r_h(\mathbf{h}_i) = \inf_{\mathbf{h}_j \ (j \neq i)} d(\mathbf{h}_i, \mathbf{h}_j)$.

To let the basis function $\phi_i(t)$ centered at H_i cover all points in $\mathcal{V}(\mathbf{h}_i)$ and to ensure the handle interpolation property, it should have

$$r_d(\mathbf{h}_i) \leq r_i \leq r_h(\mathbf{h}_i). \tag{3}$$

The support size can be $r_i = (1-\alpha) r_d(\mathbf{h}_i) + \alpha r_h(\mathbf{h}_i)$ with $\alpha \in [0, 1]$ being specified by users as a shape factor affecting the rigidity of deformation. For most of the examples in this paper, $\alpha = 1$ is used. It is possible to have two handles too close to each other so that $r_h(\mathbf{h}_i) < r_d(\mathbf{h}_i)$ (see Fig.2(c) for an example), i.e. the condition in (3) could be violated. To solve such cases, we will exploit a virtual handle insertion algorithm (presented in Section IV).

C. Properties of Weights

We analyze the properties of our weights for the handle-driven warping based on linear blending.

- **Smoothness:** The scalar field of weights must be smooth to avoid visual artifacts (discontinuity). The criteria we set on the basis functions ensure the C^2-continuity of our weights.
- **Interpolation:** The final transformations determined by the linear blending must interpolate the transformations at the handles. Specifically, the weight at each handle H_i is *one* at its origin while basis functions centered at other handles give *zero* at this point. This is guaranteed by the locality and sparsity in our formulation.
- **Consistency:** When applying the same transformation $\mathbf{T}$ on all handles, all points in Ω must be consistently transformed by $\mathbf{T}$. This is enforced by the partition-of-unity property in our formulation. Another consistency requirement is about direction. The region influenced by a handle should not change in the inverse direction of the transformation assigned on the handle. We ensure this by the property of non-negativity.
- **Shape-awareness:** A shape-aware warping by the transformations of handles indicates that the influence of handles should be conformal to the shape features of the image, rather than the Euclidean distances to handles. As we use intrinsic-distance to compute the weights on points by linear blending , shape-awareness is naturally guaranteed in our approach. All points will behave in a conformal manner with their relative handles.

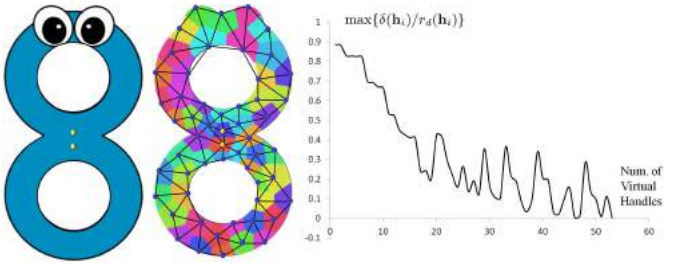

Fig. 3. When two handles are too close to each other (see left), the condition for interpolation (i.e., $r_d(\cdot) < r_h(\cdot)$) can only be satisfied after inserting virtual handles. (Middle) The newly inserted virtual handles (in blue) tessellate the deformation domain into Voronoi cells whose areas are similar to neighboring cells. The Delaunay graph, $\mathcal{DG}(\mathcal{H} \cup \mathcal{H}^v)$, of the Voronoi diagram is also shown – see the network linking the handles. (Right) The score, $\max\{\delta(\mathbf{h}_i)/r_d(\mathbf{h}_i)\}$, of our Virtual Handle Insertion algorithm drops while inserting virtual handles.

- **Non-negativity:** $\phi_i(t) \geq 0$ so that $\forall \mathbf{p} \in \Omega, w_i(\mathbf{p}) \geq 0$. Moreover, when $r_i \geq r_d(\mathbf{h}_i)$ is ensured for all handles, every point in Ω should be covered by at least one handle's support. In other words, $\sum_{j=1}^{m} \phi_j(\cdot) \neq 0$.
- **Partition-of-unity:** This has been enforced by the formulation in Eq.(2). That is,

$$\sum_{i=1}^{m} w_i(\mathbf{p}) = \sum_{i=1}^{m} \frac{\phi_i(d(\mathbf{p}, \mathbf{h}_i)/r_i)}{\sum_{j=1}^{m} \phi_j(d(\mathbf{p}, \mathbf{h}_j)/r_j)} \equiv 1.$$

- **Locality/Sparsity:** This is preserved by $\forall t \geq 1, \phi_i(t) \equiv 0$ and the condition given in Eq.(3). The transformation at a point coincident with a handle is only determined by the handle itself. $\forall i \neq j, \phi_j(\mathbf{h}_i) \equiv 0$.
- **Nonlocal maxima/minima[2]:** The global maximum of a weight w_i only occurs at the origin of handle H_i and the regions solely covered by the support of H_i. The global minima of a weight w_i only occurs in the regions not covered by the support of H_i. For the intersected regions of the supports of H_i and other handles, we observe this phenomenon in all our experiments.

In short, our method preserves all the merits of prior methods for linear blending based warping (e.g., [1], [2], [9]) while introducing new benefits of flexibility and efficiency.

The current weighting formulation has a major defect. The interpolation property cannot be preserved when the distance between two handles are too close while the regions to be covered by either handle are large. Specifically, the interpolation of handles becomes an approximation when $r_d(\mathbf{h}_i) \leq r_h(\mathbf{h}_i)$ in Eq.(3) can *not* be satisfied, which is not expected. To tackle this problem, we propose a virtual handle insertion algorithm in the following section.

IV. Virtual Handle Insertion

A virtual handle insertion algorithm is developed to allow users freely set handles within the domain of an image, with the help of which the interpolation property is always guaranteed and the shape-awareness of deformation is improved.

[2]This property is proved experimentally in all our tests. More results of weight distribution with isocurves are shown in the supplementary material.

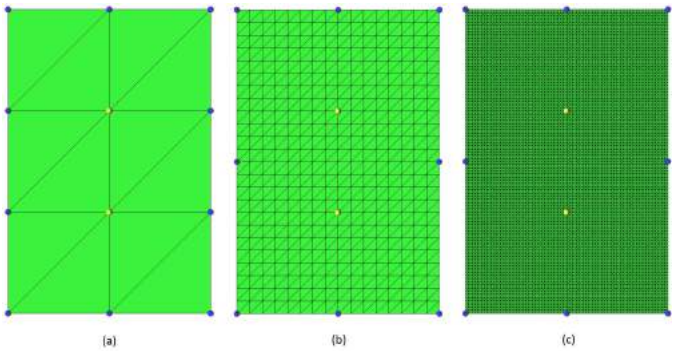

Fig. 4. The distribution of virtual handles (in blue) with two point handles (in yellow): (a) for an extremely coarse mesh, all points are selected as handles (real and virtual) and Voronoi diagram based on handles cannot be built; (b, c) for dense meshes, the virtual handles are located on the meshes by satisfying the prescribed conditions.

A. Insertion algorithm

When $r_d(\mathbf{h}_i) > r_h(\mathbf{h}_i)$, there are points in the Voronoi cell $\mathcal{V}(\mathbf{h}_i)$ whose distances to $\mathbf{h}_i$ are larger than the minimal distance from $\mathbf{h}_i$ to other handles. It is easy to observe that inserting new sites at the points $\mathbf{h}_d \in \mathcal{V}(\mathbf{h}_i)$ with $d(\mathbf{h}_d, \mathbf{h}_i) = r_d(\mathbf{h}_i)$ can reduce $r_d(\mathbf{h}_i)$ while keeping $r_h(\mathbf{h}_i)$ unchanged. Based on this observation, we develop a greedy algorithm for virtual handle insertion. Define $\mathcal{H}$ as the set of handles and $\delta(\cdot) = r_d(\cdot) - r_h(\cdot)$. When there exists $\mathbf{h}_i \in \mathcal{H}$ with $\delta(\mathbf{h}_i) > 0$, new virtual handles are inserted to resolve this problem by reducing $\max_{\mathbf{h}_i \in \mathcal{H}}\{\delta(\mathbf{h}_i)/r_d(\mathbf{h}_i)\}$. The pseudo-code for the algorithm **Virtual Handle Insertion** is described as follows:

Algorithm 1 Virtual Handle Insertion

Input: the set $\mathcal{H}$ of real handles
Output: the expanded set $\mathcal{H}$ with virtual handles
while $\exists \mathbf{h}_i \in \mathcal{H}$, $\delta(\mathbf{h}_i) > 0$ **do**
    Find the handle $\mathbf{h}_m = \arg\max_{\mathbf{h}_i \in \mathcal{H}} \delta(\mathbf{h}_i)/r_d(\mathbf{h}_i)$;
    Find a point $\mathbf{p} \in \mathcal{V}(\mathbf{h}_m)$ with $d(\mathbf{p}, \mathbf{h}_m) = r_d(\mathbf{h}_m)$;
    Insert a new virtual handle located at $\mathbf{p}$ into $\mathcal{H}$;
    Update the values of $r_d(\cdot)$ and $r_h(\cdot)$ on all handles;
end while
return $\mathcal{H}$;

We know that inserting new handles in a Voronoi cell $\mathcal{V}(\mathbf{h}_i)$ with $\delta(\mathbf{h}_i) > 0$ can reduce the value of $\delta(\mathbf{h}_i)$. However, inserting a new site $\mathbf{h}_d \in \mathcal{V}(\mathbf{h}_i)$ can also affect other handles (i.e., H_j with $j \neq i$). In an extreme case, the original $\delta(\mathbf{h}_j) \leq 0$ could be turned into $\delta(\mathbf{h}_j) > 0$. Then, more virtual handles need to be added into $\mathcal{V}(\mathbf{h}_j)$.

Our virtual handle insertion algorithm can be considered as a variant of the farthest point sampling algorithm [17], which tends to tessellate a domain into a Voronoi diagram with neighboring Voronoi cells having similar sizes. The condition of $r_d(\cdot) \leq r_h(\cdot)$ is satisfied on all handles when this is the case. Our experimental tests also verify this observation (see Fig.3 for an example).

Remarks. 1. We observe the convergence of virtual handle insertion in all our experiments, and many results provided in this paper and the supplementary material show the final distribution of handles (user-input and virtual ones). The insertion process may fail if the mesh density is not enough compared with the closeness of user-input handles, especially when there is no point between two handles on the mesh, which indicates the current sampling rate gives a very poor approximation of the domain to be warped and an up-sampling step is required to refine this approximation (see an illustration in Fig.4). We can easily detect such case and conduct a dynamic up-sampling step in our implementation. Interestingly, it is worth pointing out that the "failure" of the insertion process here means all points in the domain will be selected as virtual handles, rather than a dead loop leading to no response in our implementation, which indicates a termination guarantee of the insertion process in a discrete sense. Meanwhile, the determination of weights on virtual handles degenerates to solve a harmonic problem over the whole underlying mesh, which will be detailed in Section IV-B. 2. Noted that it lacks a complete proof on the convergence of virtual handle insertion here to provide a theoretical foundation for this algorithm. More interestingly, we find this can be generalized to a spatial partition problem in n dimensional space. Therefore, we propose a conjecture to be proved in the Appendix and would like to invite future efforts on it.

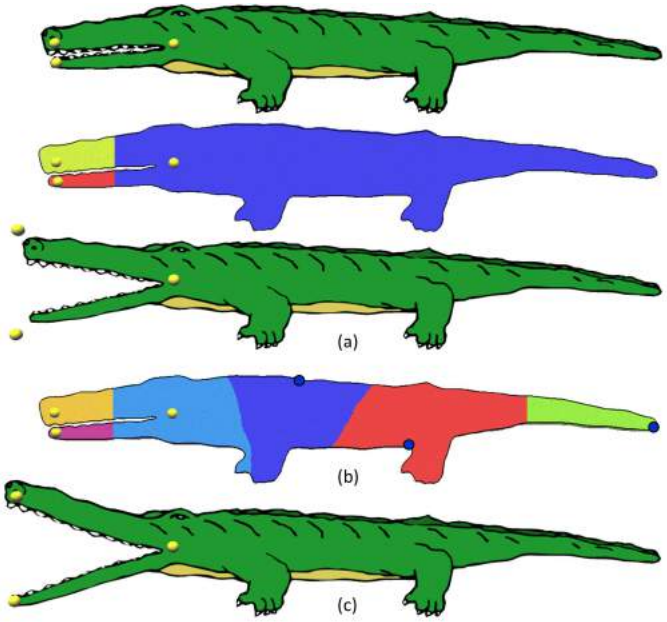

Fig. 5. A handle covering a large region can affect the interpolation on its nearby handles. (a) For the handle at the right, its Voronoi cell covers all the right part of the alligator – this leads to a value of $r_d(\cdot)$ that is much larger than $r_h(\cdot)$. In this case, transformations at the left two handles cannot be interpolated. (b) Virtual handles (in blue color) are added to resolve the problem by the *insertion* algorithm. As a result, the domain to be deformed has been decomposed into smaller Voronoi cells with handles (real and virtual) as sites. (c) The deformation result is driven by both the real and the virtual handles, where the transformations at real handles are interpolated.

B. Transformation on Virtual Handles

The left problem is how to determine the transformations on virtual handles according to the user-specified transformations

on real handles. We denote the set of real handles as $\mathcal{H}$ and the set of virtual handles as $\mathcal{H}^v$. As aforementioned, the handles of $\mathcal{H} \cup \mathcal{H}^v$ can partition the given domain Ω into a Voronoi diagram $Vor(\mathcal{H} \cup \mathcal{H}^v)$. A dual graph of $Vor(\mathcal{H} \cup \mathcal{H}^v)$ can be constructed by 1) using the sites of every Voronoi cell as nodes and 2) linking the sites of every two neighboring Voronoi cells by lines as edges, which is a Delaunay graph. We denote the Delaunay graph as $\mathcal{DG}(\mathcal{H} \cup \mathcal{H}^v)$ and use the symbol H to represent nodes (real or virtual handles) in $\mathcal{DG}$. The transformations of handles in $\mathcal{H}^v$ are determined as follows:

- For each real handle H_i in $\mathcal{H}$, a harmonic field $\varpi_i(\cdot)$ is computed on $\mathcal{DG}$ to assign a field value $\varpi_i(H_g)$ to each of the other handles. Boundary conditions, $\varpi_i(H_i) = 1$ and $\varpi_i(H_{j \neq i}) = 0$, are given to compute the harmonic field $\varpi_i(\cdot)$. If there are m handles in $\mathcal{H}$, m harmonic fields are determined on $\mathcal{DG}$.
- After converting the transformation $\mathbf{T}_i$ of each real handle into a rotation quaternion $\mathbf{q}_i$ and a translation vector $\mathbf{t}_i$, the rotation and the translation on a virtual handle $H_v \in \mathcal{H}^v$ can be determined by

$$\begin{pmatrix} \mathbf{q}_v \\ \mathbf{t}_v \end{pmatrix} = \frac{1}{\varpi_{sum}(H_v)} \sum_{H_i \in \mathcal{H}} \varpi_i(H_v) \begin{pmatrix} \mathbf{q}_i \\ \mathbf{t}_i \end{pmatrix} \quad (4)$$

  with $\varpi_{sum}(\cdot) = \sum_{H_j \in \mathcal{H}} \varpi_j(\cdot)$.
- Finally, the quaternion and the translation determined on each virtual handle are converted back into a transformation matrix to be used in linear blending.

The transformations of virtual handles determined in this way improve the quality of warping. As illustrated in Figs.5 and 6, deformation of the whole image driven by the transformations on both real and virtual handles is very natural. The influence of a real handle decays when the intrinsic-distance to it increases.

V. Implementation Details

A. Basis Function

In Section III-B, we have listed the required properties for basis functions to achieve the C^2-continuity of weights. In our implementation, Bézier polynomial is employed for the function $\phi_i(t)$ so that the constraints for C^1- and C^2-continuity at the boundary of the supporting regions can be satisfied. To ease the evaluation and analysis, each $\phi_i(t)$ is represented as the y-component (i.e., $\phi_i(t) = \mathbf{b}^y(t)$, $t \in [0, 1]$) of a 2D Bézier curve with degree-n ($n \geq 5$)

$$\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_{i,n}(t), \quad (5)$$

where $B_{i,n}(t)$ are the Bernstein polynomials. From the property of Bézier curves, we have

$$\mathbf{b}'(0) = n(\mathbf{b}_1 - \mathbf{b}_0), \quad \mathbf{b}'(1) = n(\mathbf{b}_n - \mathbf{b}_{n-1})$$

$$\mathbf{b}''(0) = n(n-1)(\mathbf{b}_2 - 2\mathbf{b}_1 + \mathbf{b}_0)$$

$$\mathbf{b}''(1) = n(n-1)(\mathbf{b}_n - 2\mathbf{b}_{n-1} + \mathbf{b}_{n-2})$$

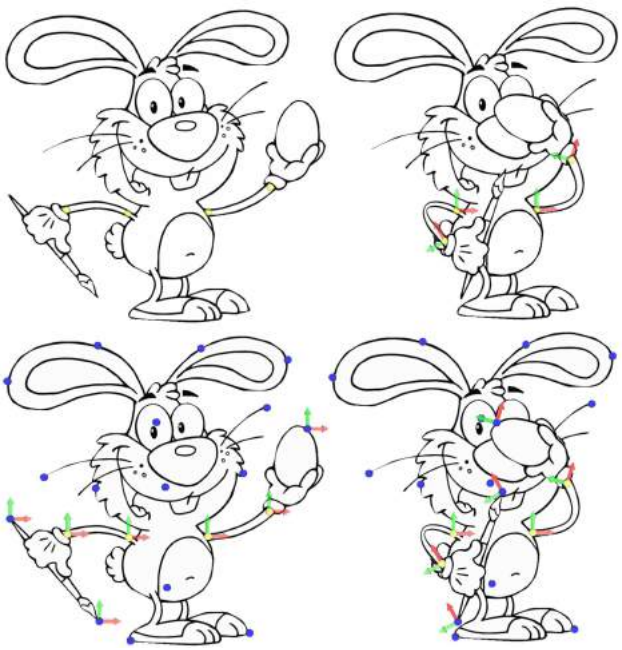

Fig. 6. The deformation of a rabbit is driven by four real handles (see the yellow dots and the frames shown in the top row). The result of deformation is determined with the help of virtual handles (shown in blue dots). The transformations at handles (both real and virtual ones) are illustrated by frames.

for a Bézier curve in n-th order. Incorporating the constraints in Section III-B, we have

$$\mathbf{b}_1 = \mathbf{b}_0, \mathbf{b}_n = \mathbf{b}_{n-1}, \mathbf{b}_1 = \frac{\mathbf{b}_0 + \mathbf{b}_2}{2}, \mathbf{b}_{n-1} = \frac{\mathbf{b}_n + \mathbf{b}_{n-2}}{2}.$$

As we already set $\mathbf{b}_i^x = i/n$ to let $x = t$, it is not difficult to find that $\mathbf{b}_0^y = \mathbf{b}_1^y = \mathbf{b}_2^y = 1$ and $\mathbf{b}_n^y = \mathbf{b}_{n-1}^y = \mathbf{b}_{n-2}^y = 0$ satisfy all these constraints. For the rest control points, we can simply assign them as 0.5 or align them along the line $\mathbf{b}_2\mathbf{b}_{n-2}$ uniformly.

B. Intrinsic Distance Approximation

Generally, we sample the input domain Ω (the input image) to be deformed into a dense set of points $\mathcal{P}$. By searching for k-nearest-neighbors of each point, a graph $\mathcal{G}(\mathcal{P})$ spanning Ω (in discrete form) can be established by using points in $\mathcal{P}$ as nodes and adding links between neighboring points as edges. Note that user-specified handles should also be added into $\mathcal{P}$ to construct the graph (i.e., $\mathcal{H} \subset \mathcal{P}$). The intrinsic-distance from any point $\mathbf{q} \in \mathcal{P}$ to a handle is approximated by the distance between $\mathbf{q}$ and the handle on the graph, which can be computed efficiently with Dijkstra's algorithm. Also, the Voronoi diagram $Vor(\mathcal{H})$ can be obtained by Dijkstra's algorithm with multiple sources on $\mathcal{G}(\mathcal{P})$, where each sample is assigned to a Voronoi cell. To determine the weights on a general point $\mathbf{p} \in \Omega$ that is not a sample in $\mathcal{P}$, a linear blending based on reciprocal distance weights [18] is employed to obtain the weight on $\mathbf{p}$ from its k-nearest-neighbors in $\mathcal{P}$. There are more sophisticated parameterization strategies in [18], which can also be applied here. With the help of this meshless parameterization, we can easily take an up-sampling

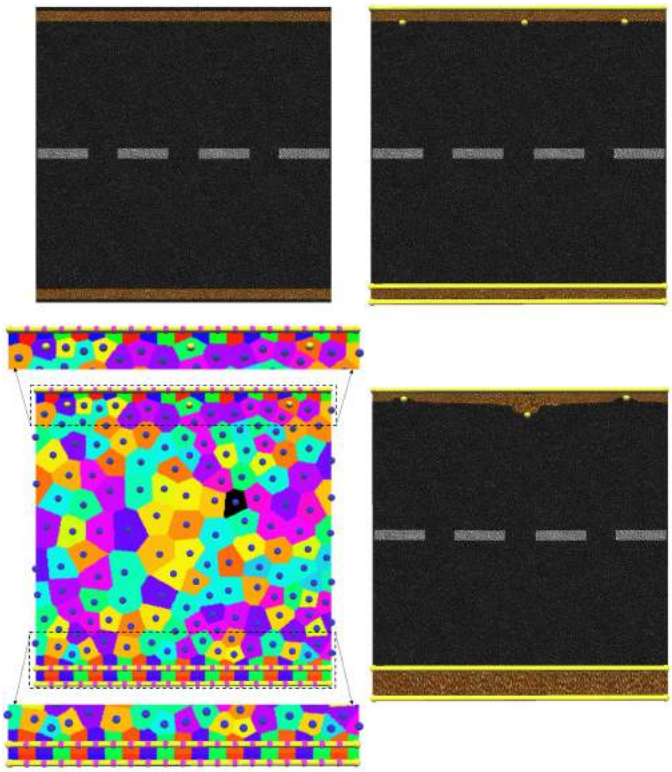

Fig. 7. An example of warping an image with point and bar handles. As shown in the Voronoi diagram based on the distribution of virtual handles and the tessellation of bars, the bar handles dominate the regions around it, leading to perfect interpolation about the movement of handles.

step in the domain Ω when the point set $\mathcal{P}$ becomes sparse when applying a drastic deformation.

As a special case, intrinsic-distance evaluation for pixelized images or convex domain is simplified into the computation of Euclidean distance between two points as the whole domain is convex, which can be achieved very fast. This also ensures the computational efficiency of our algorithm.

C. Harmonic Field

After using the virtual handle insertion algorithm to generate a set of new handles, harmonic fields are computed on a dual graph of $Vor(\mathcal{H})$ to determine the transformations on virtual handles. By our boundary condition, all field values are non-negative when uniform Laplacian is employed. In other words, the coefficients used in Eq.(4) are non-negative. Instead of solving a linear system to compute the harmonic field, we initially assign the field values on all real handles as one and the weights on all virtual handles are set as zero. Then we apply Laplacian operators to update their field values iteratively. The field values on virtual handles can be efficiently obtained after tens of iterations.

D. Interactive Handles

The point handles can be generalized to different types of bar handles (e.g. line segments and polygons). Unluckily, such extension is not straightforward as: 1) the full boundary of an image to be warped is often expected to be strictly fixed when applying a deformation inside the domain; 2) the transformations of points on any handles should respect the interpolation property introduced in Section III-C. It also needs to be processed carefully when a handle is very close to another. In our implementantion, the generalization of handles is realized by taking advantage of the properties of Voronoi diagram (see Fig.7). The bar handles are discretized into a dense set of bar samples to ensure that all points around the bar handles should belong to the Voronoi sites centered at those discrete bar samples, which leads to perfect interpolation on the bar handles. In practice, we detemine the set of bar samples for a bar handle by uniformly dividing it into $\lceil L_{bar}/d \rceil$ segments, where L_{bar} is its length and d refers to the minimum Euclidean distance between any handles (point and bar handles) over the whole domain of computation (refer to the first figure in Fig.8 (a)). The transformations of bar samples are calculated as linear combinations of the transformations at bar handles' end points.

TABLE I
EXAMPLE STATISTICS

	$\lvert\mathcal{H}_b\rvert$	$\lvert\mathcal{H}_p\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{S}\rvert$	t_{rd} (sec.)	t_w (sec.)
Tower	12	0	113	7,834	3.75	0.013
Wolf	4	2	1	6,694	0.020	0.001
Portrait	7	0	36	9,438	0.380	0.006
Alligator	0	3	3	9,456	0.409	0.002
Rabbit	0	4	16	22,972	1.700	0.434
Road	3	3	124	13,212	3.18	0.002

VI. RESULTS

Our weighting method provides a compact tool to assign continuous weights for all points in the domain to be warped. With the help of sophisticated techniques for assigning transformations on the handles (e.g. the pseudo-edge method in [1]), a natural user interface for image warping can be achieved.

We have tested this approach in a variety of examples by using both the point and bar handles. Figures 5 and 6 have already demonstrated the functionality of point handles. Especially in Fig.5, the scheme of virtual handles insertion guarantees the interpolation at real handles. Figure 6 illustrates the effectiveness of our method in determining transformations on virtual handles. The example of using bar handles to warp the shape of a Tower has been shown in Fig.8 (a). Another example is given in Fig.8 (b) to deform a portrait. Figure.8 (c) shows a comparison of image manipulation with and without fixing boundary. To obtain natural bending results, we can add rotations on handles by heuristics (e.g., the pseudo-edge [1]). The example in Fig.8 (d) demonstrates the performance of our approach with a symmetric deformation. When deforming a symmetric domain by adding symmetric transformations on the symmetric handles, it is expected to get a symmetric warped result. This property is strictly preserved by our formulation.

In prior mesh-based approaches, the numerical system must be solved once more when new handles are inserted. By using our weighting formulation, the time cost of adding new handles is very trivial as the weights are determined in a

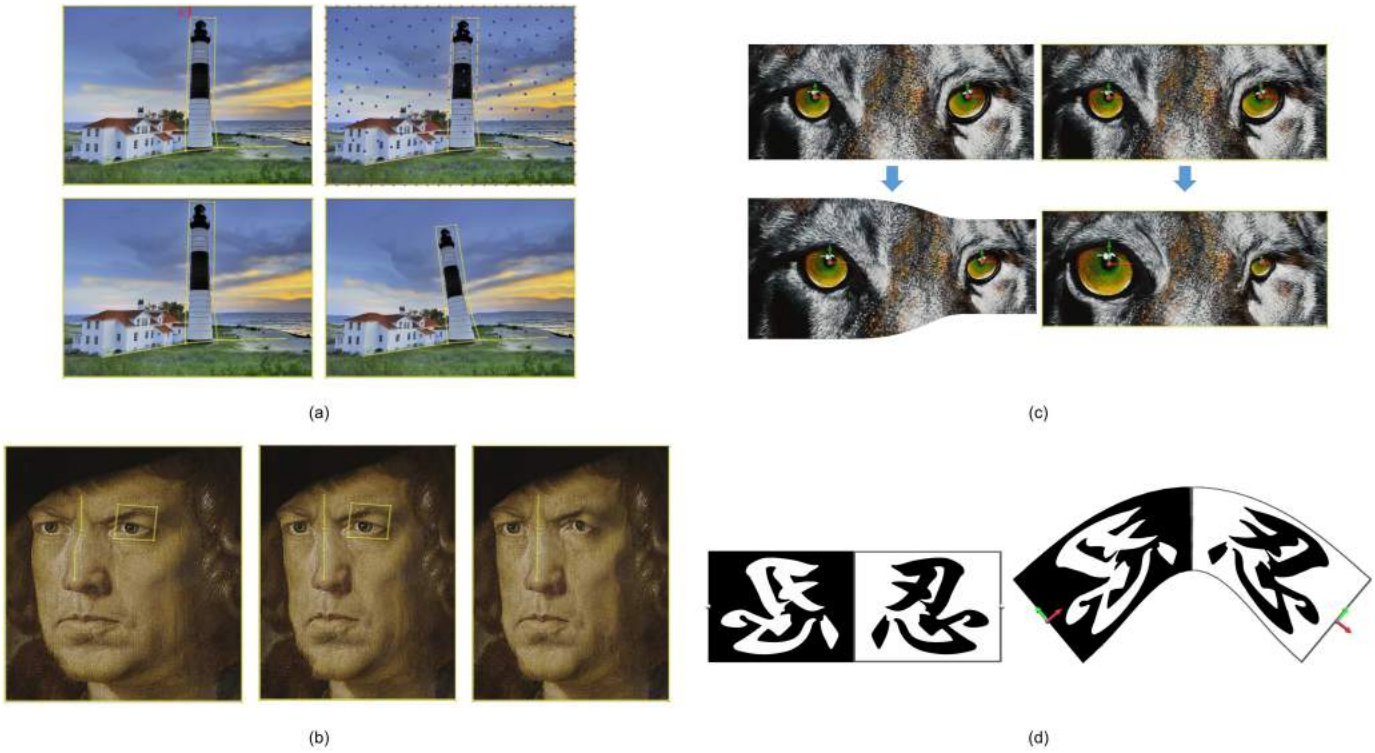

Fig. 8. Experimental Results: (a) an example of manipulating the tower while fixing the image boundary; (b) a portrait edited by bar handles with salient feature inside the closed loop of bar handles at the left eye preserved; (c) a comparison of image editing with and without fixing its boundary; (d) an example with symmetric transformations on two symmetric handles to deform a symmetric domain.

TABLE II
COMPARISON WITH DIFFERENT RESOLUTIONS

Number of Vertices	Time of Computation (sec.)	
	Biharmonic	Our method
627	0.818	0.024
31,325	8.691	0.110
619,649	N/A	20.796

closed-form. Table I lists the statistics of our approach on different examples, where $|\mathcal{H}_b|$ and $|\mathcal{H}_p|$ denote the number of bar and point handles, $|\mathcal{V}|$ stands for the number of virtual handles added in the domain and $|\mathcal{S}|$ represents the number of points used in the computation. The columns under t_{rd} and t_w state the time used in the computation of region decomposition (including both Voronoi diagram computation and intrinsic distance evaluation) and weights respectively. All the tests are conducted on a laptop with Intel Core i7-3740QM CPU at 2.70GHz and 8GB memory. Our current version of implementation only uses a single core. As shown in the table, the computational bottleneck is the computation of region decomposition that is closely related with the number of points used in the domain and the type and placement of handles. But compared with prior approaches, our method has achieved significant improvement in computational efficiency. Table II shows an comparison with [1] using the same setup for the Gingerman model in Fig.1. We fail to test [1] on a mesh having $617k$ vertices with the public program provided by the authors as there are too many vertices. To compare our technique with another approach [15], both approaches require reinitialization when inserting a new handle in the computational domain. As mentioned in [15], it roughly takes seconds (using Euclidean distance) to minutes (using geodesic distance) to compute interior distances. In our experiments with a $3D$ bar and armadillo model shown in the accompany video, the time costs for precomputation are about 0.1 second (using Euclidean distance) and 1.5 minutes (using geodesic distance) respectively.

Limitations & Discussion. When using the weighting formulation presented in the paper to deform images, sample points are adopted as the medium for realizing the computation. The error-bound of computation on this discrete representation is controlled by the density of samples. However, during the process of a sequence of deformations, the density of points could be changed dramatically. In this sense, a dynamic up-sampling step should be integrated in the framework to preserve the error-bound of intrinsic-distance computation. Super-sampling techniques or texture mapping on a mesh can be exploited in image editing applications. In our framework, the time cost of weight evaluation is trivial after resampling. The bottleneck is the computation of intrinsic-distances on the sample points. Our current implementation is based on the Dijkstra's algorithm. This shortest path problem with multiple sources can be computed in parallel by using graphics accelerated hardware, which can result in a significant speed-up.

Our formulation gives global maxima at the positions of handles according to the interpolation property. For a shape-aware deformation, it is also demanded having nonlocal max-

ima/minima anywhere else in the domain. This has been verified in our experimental tests. We check the topology of isocurves on the fields of weights (see Fig.9 for an example). If there is a closed loop formed by isocurves of $w_i(\cdot)$ at one place except the center of the handle $\mathbf{h}_i(\cdot)$, a local maximum/minimum is generated there. No such case is found in all our examples. More examples for verification can be found in the supplementary material. It should be admitted that a theoretical proof for nonlocal maxima/minima property is lacking here as a technical guarantee.

The deformations driven by linear blending are not always injective and therefore can generate the results with foldovers and self-intersections. Recently, some researches have been conducted in this thread to produce injective mappings (e.g., [19]), which are mainly mesh-based. In a function based formulation, the injectivity of a mapping can be checked by the positiveness of Jacobian. But the interpolation property is not preserved any more in those methods trying to hold the injectivity of mappings. We argue that a simultaneous satisfaction of both the interpolation and injectivity property is paradoxical. Other method preserving the interpolation property also violate the injectivity as our approach (see an comparison in [20]). We will unavoidably see foldovers in some warped results if the handles are dragged too much (see Fig.10 for an example).

As analyzed in Section III-B, when inserting a new handle, the distribution of virtual handles needs to be recomputed to ensure no violation of condition 3 happens in the whole domain, which is the current computational bottleneck of this proposed method. A dynamic mechanism can be realized in practice to perform a resampling of virtual handles only if unsatisfaction of condition 3 is detected as a trick to reduce computational overhead. As indicated in the Table I and II, the time cost for determination of virtual handles and region decomposition is negligible for small-scale meshes and trivial for large-scale ones.

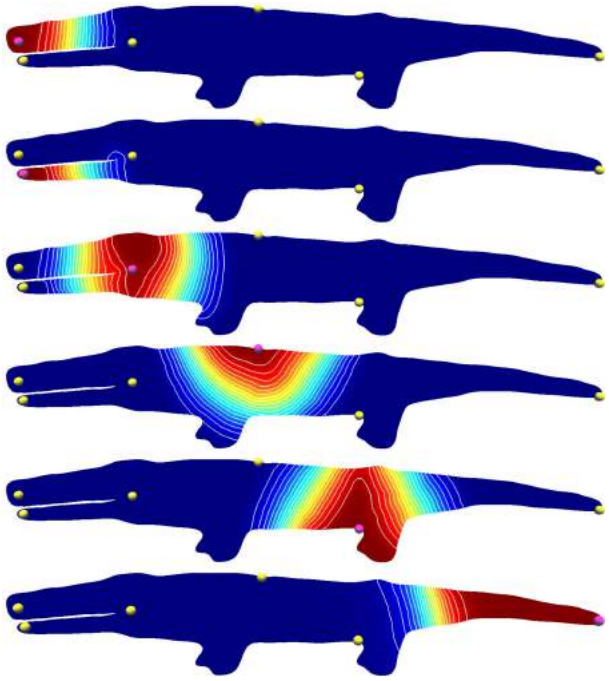

Fig. 9. The verification of nonlocal maxima/minima is taken by analyzing the topology of isocurves on the scalar fields of weights. The handles (real and virtual) in this example are the ones shown in Fig.5.

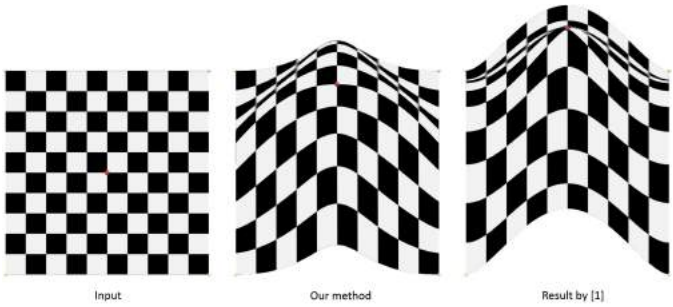

Fig. 10. Self-intersection can be found for the methods respecting handle interpolation.

VII. Conclusion & Future Work

In this work, we present a method to efficiently determine weights of linear blending for image warping. Our formulation is in a closed-form and can be easily used in a variety of applications. Equipped with the virtual handle insertion algorithm, good properties of weights generated by prior mesh-based methods can all be preserved in this approach. A variety of examples have been shown to demonstrate the effectiveness of our approach.

There are some potential improvements for our technique. First, the tessellation of bar handles is simple and naïve in the current version. As it will affect the result of warping, we are curious about finding a locally adaptive and non-uniform strategy to further strengthen our result. A similar scenario is about the density of underlying mesh. Although the validity of our method about mesh density applies to infinity, the distribution of real and virtual handles will converge when the mesh is fine enough, indicating the possible existence of a density threshold which can help reduce the computational overhead of this approach (see Fig.4(b)(c)). Second, it is interesting to investigate how to resolve the problem when self-intersection is detected, which will be one of our future work. We may restrict the movement of handles to avoid the occurrence of foldovers. Besides, only linear blending scheme in 2D is tested in the paper. Actually our method can be extended directly to 3D models for shape deformation without any modifications (see two examples in the accompany video). Lastly, we plan to further apply the weights generated in this approach to more advanced skinning applications, such as the blending of two rigid motions will result in a rigid motion, which is a very important property when deformation of articulated characters is computed by the skinning methods.

Acknowledgments

We would like to thank Dr. Yang Liu with Microsoft Research for his insightful suggestions to this work. We also feel thankful for the constructive comments from all reviewers on the improvement of this manuscript. This work is

partially supported by NSFC (No. 61300136), NSF of Guangdong Province (No. 2015A030313220, S2013020012795) and Doctoral Fund of Ministry of Education of China (No. 20130172120010).

REFERENCES

[1] A. Jacobson, I. Baran, J. Popović, and O. Sorkine, "Bounded biharmonic weights for real-time deformation," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 78:1–78:8, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964973

[2] A. Jacobson, T. Weinkauf, and O. Sorkine, "Smooth shape-aware functions with controlled extrema," *Comp. Graph. Forum*, vol. 31, no. 5, pp. 1577–1586, Aug. 2012.

[3] K. Singh and E. Fiume, "Wires: A geometric deformation technique," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. ACM, 1998, pp. 405–414.

[4] W. von Funck, H. Theisel, and H.-P. Seidel, "Vector field based shape deformations," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1118–1125, Jul. 2006.

[5] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.

[6] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213–230, Jan. 2008.

[7] M. Botsch, M. Pauly, M. Wicke, and M. H. Gross, "Adaptive space deformations based on rigid cells," *Comput. Graph. Forum*, vol. 26, no. 3, pp. 339–347, 2007.

[8] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, ser. SGP '07. Eurographics Association, 2007, pp. 109–116.

[9] A. Jacobson and O. Sorkine, "Stretchable and twistable bones for skeletal shape deformation," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 165:1–165:8, Dec. 2011.

[10] S. Schaefer, T. McPhail, and J. Warren, "Image deformation using moving least squares," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 533–540, Jul. 2006.

[11] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki, "Harmonic coordinates for character articulation," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.

[12] M. Ben-Chen, O. Weber, and C. Gotsman, "Variational harmonic maps for space deformation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 34:1–34:11, Jul. 2009.

[13] S.-M. Hu, H. Zhang, C.-L. Tai, and J.-G. Sun, "Direct manipulation of ffd: efficient explicit solutions and decomposible multiple point constraints," *The Visual Computer*, vol. 17, no. 6, pp. 370–379, 2001.

[14] M. Botsch and L. Kobbelt, "Real-time shape editing using radial basis functions," *Comput. Graph. Forum*, vol. 24, no. 3, pp. 611–621, 2005.

[15] Z. Levi and D. Levin, "Shape deformation via interior rbf," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 7, pp. 1062–1075, 2014.

[16] Y. Wang, A. Jacobson, J. Barbič, and L. Kavan, "Linear subspace design for real-time shape deformation," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 57:1–57:11, Jul. 2015. [Online]. Available: http://doi.acm.org/10.1145/2766952

[17] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *Image Processing, IEEE Transactions on*, vol. 6, no. 9, pp. 1305–1315, 1997.

[18] M. S. Floater and M. Reimers, "Meshless parameterization and surface reconstruction," *Computer Aided Geometric Design*, vol. 18, no. 2, pp. 77–92, 2001.

[19] C. Schüller, L. Kavan, D. Panozzo, and O. Sorkine-Hornung, "Locally injective mappings," *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)*, vol. 32, no. 5, pp. 125–135, 2013.

[20] Y. Lipman, "Bounded distortion mapping spaces for triangular meshes," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 108, 2012.

APPENDIX - CONJECTURE

Following the definition of intrinsic distance $d(\cdot,\cdot)$ in Section III-A, we consider the partition of a connected closed domain $\mathcal{D} \subset \mathcal{R}^n$ into cells. We first give some fundamental definitions:

- *Anchor points* $\mathcal{A} = \{\mathbf{h} \mid \mathbf{h} \in \mathcal{D}\}$ are a set of points that are predefined in the domain $\mathcal{D}$.
- *Size of a cell* $\mathcal{V}$ centerd at $\mathbf{h}_i$ is defined as $r_d(\mathbf{h}_i) = \sup_{\mathbf{p}\in\mathcal{V}(\mathbf{h}_i)} d(\mathbf{p}, \mathbf{h}_i)$;
- *Separation of* $\mathbf{h}_i$ to other cells is defined as $r_h(\mathbf{h}_i) = \inf_{\mathbf{h}_j\ (j\neq i)} d(\mathbf{h}_i, \mathbf{h}_j)$.

The domain $\mathcal{D}$ is partitioned with regard to a set of anchor points in a Voronoi-diagram-like manner. Specificly, a point $\mathbf{p}$ belongs to a cell centered at $\mathbf{h}_i$ if $d(\mathbf{p}, \mathbf{h}_i) \leq d(\mathbf{p}, \mathbf{h}_j), j \neq i$.

Conjecture: For any given set of anchor points $\mathcal{A}$, there exists at least an expanded set $\mathcal{H}$ based on $\mathcal{A}$ (i.e. $\mathcal{A} \subseteq \mathcal{H}$), with regard to which the partition of $\mathcal{D}$ into cells satisfies the following non-intrusive condition

$$r_d(\mathbf{h}_i) \leq r_h(\mathbf{h}_i), \mathbf{h}_i \in \mathcal{H}.$$