Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Harmonic coordinates for real-time image cloning*

Rui WANG, Wei-feng CHEN†‡, Ming-hao PAN, Hu-jun BAO

(*State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou 310027, China*)

†E-mail: chenweifeng@cad.zju.edu.cn

Received Mar. 19, 2010; Revision accepted June 4, 2010; Crosschecked Aug. 2, 2010

Abstract: Traditional gradient domain seamless image cloning is a time consuming task, requiring the solving of Poisson's equations whenever the shape or position of the cloned region changes. Recently, a more efficient alternative, the mean-value coordinates (MVCs) based approach, was proposed to interpolate interior pixels by a weighted combination of values along the boundary. However, this approach cannot faithfully preserve the gradient in the cloning region. In this paper, we introduce harmonic cloning, which uses harmonic coordinates (HCs) instead of MVCs in image cloning. Benefiting from the non-negativity and interior locality of HCs, our interpolation generates a more accurate harmonic field across the cloned region, to preserve the results with as high a quality as with Poisson cloning. Furthermore, with optimizations and implementation on a graphic processing unit (GPU), we demonstrate that, compared with the method using MVCs, our harmonic cloning gains better quality while retaining real-time performance.

Key words: Seamless cloning, Poisson's equation, Harmonic coordinates (HCs), Mean-value coordinates (MVCs), GPU acceleration

doi:10.1631/jzus.C1000067 **Document code:** A **CLC number:** TP391.41

1 Introduction

Perceptual experiments suggest that the human visual system pays more attention to measuring local intensity differences rather than the intensity itself. This is the reason why artists usually perform editing tasks using gradient domain techniques. This leads to an important and useful application, image cloning, which clones a source image patch (e.g., an object) into a target image seamlessly (Fig. 1). Image cloning has attracted significant interest in recent years (Pérez *et al.*, 2003; Jia *et al.*, 2006; Farbman *et al.*, 2009) and has been introduced in some professional image editing software (Georgiev, 2004).

Some of the most important gradient based image cloning techniques are Poisson-based methods (Pérez *et al.*, 2003; Jia *et al.*, 2006), which solve a large linear system, the Poisson equation. However, solving the equation is very time consuming. Although fast Poisson solvers (Szeliski, 2006; Agarwala, 2007; Kazhdan and Hoppe, 2008) and implementations of the Poisson equation on GPU (McCann and Pollard, 2008) have been proposed in recent years, they are still less efficient. That is, they require solving Poisson's equations whenever the shape or position of the cloned region is changed.

A novel coordinate-based method, mean-value-coordinate cloning (MVC cloning) (Farbman *et al.*, 2009), finds a harmonic-like interpolant to values along the boundary. As coordinate-based methods do not rely on solving the Poisson equation, they perform better. Floater's construction of mean-value coordinates (MVCs) (Floater, 2003) was motivated by the mean value theorem for harmonic functions and is given by a simple closed-form formula. However, MVCs do not preserve two important properties required by gradient cloning: the non-negativity and interior locality (DeRose and Meyer, 2006; Joshi *et*

‡ Corresponding author

* Project supported in part by the National Natural Science Foundation of China (No. 60903037) and the National Basic Research Program (973) of China (No. 2009CB320803)

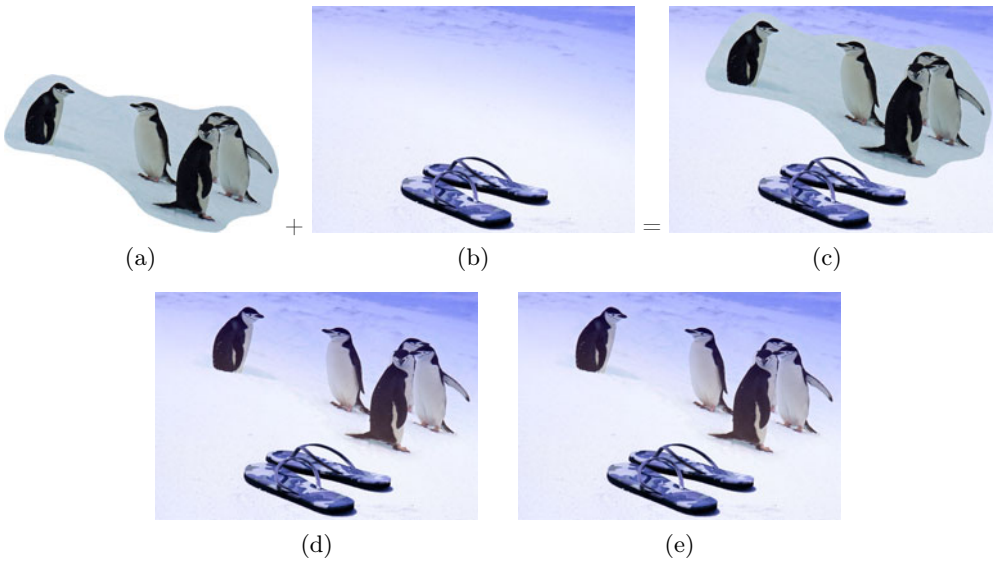


Fig. 1 Simple cloning and seamless cloning results. (a) Source patch selected from a source image; (b) Target image; (c) Simple cloning result; (d) Seamless cloning result using the Poisson image editing tool (Pérez *et al.*, 2003); (e) Seamless cloning result using our method based on harmonic coordinates interpolation

al., 2007). Hence, MVC cloning cannot clone highly concave regions (Fig. 5 in Farbman *et al.* (2009)).

In this paper, we introduce harmonic coordinates (HCs) to perform coordinate-based image cloning. As HCs retain non-negativity and interior locality, the results of harmonic cloning are much better than those of MVC cloning, especially in cloning highly concave regions.

2 Related works

In image composition, Poisson image editing (Pérez *et al.*, 2003) blends two images through Poisson's equations with a guidance field and a user-specified boundary. Image stitching in the gradient domain (Levin *et al.*, 2004) has also been proposed to reduce the artifacts caused by structure misalignment, and to correct color discrepancy. Each of these methods needs to solve Poisson's equations with a boundary condition, a computational and memory intensive task.

Agarwala (2007) constructed a reduced linear system using an adaptive quadtree subdivision of the domain, which has been shown to be significantly faster and more scalable than the general Poisson solvers for stitching large images. McCann and Pollard (2008) and Orzan *et al.* (2008) described a fast GPU implementation of the multi-grid Poisson solver with which they achieved real-time interactive performance for gradient domain image editing operations. While their methods outperform previous ones, they still involve a substantial memory footprint.

Farbman *et al.* (2009) introduced a novel approach that interpolates the membrane with coordinates assigned on each pixel rather than solving a large linear system. The coordinates they used are MVCs, and MVC cloning was successful in most examples. However, the authors reported that their system would be incompetent in cloning extremely concave regions.

Recently, Jeschke *et al.* (2009) proposed a new Laplacian solver, called the variable stencil size diffusion (VSSD) solver, with the stencil size being decided by the domain's distance map. They proved that in the 1D case the solver converges to the right solution as the standard Laplacian solver, while in the 2D case they averaged only the four values in axis aligned directions for practical use, which may cause artifacts similar to mach banding (Jeschke *et al.*, 2009). Two strategies, shrink always (SA) and shrink half (SH), were identified in their work to eliminate such artifacts. Unfortunately, the solver may take more iterations to converge, especially for the SA strategy.

3 Background

3.1 Harmonic coordinates

Generalizations of barycentric coordinates in two and higher dimensions have attracted much interest in recent years. They have been applied in several research areas, including finite element analysis, free-form deformations, and interpolation. DeRose and Meyer (2006) and Joshi *et al.* (2007) introduced HCs, which are defined as the solutions of Laplace's equation subjected to carefully chosen boundary conditions. As is mentioned, the HCs possess several properties that make them more attractive than MVCs when used to define 2D and 3D deformations. In our work, we define HCs over arbitrary planar polygons without self-intersections.

Let Ω, a closed subset of $\mathbb{R}^2$, be the image definition domain. Φ is an unknown scalar function defined over Ω. The Laplace equation assumes the following form:

$$\frac{\partial^2 \Phi}{\partial x^2}+\frac{\partial^2 \Phi}{\partial y^2}=0 \quad \text{over } \Omega. \tag{1}$$

A common Jacobi solver iterates the solution at each pixel by averaging the function value of its four connected neighbors, with previous approximation being the input to the next iteration:

$$\Phi(\boldsymbol{x})=\frac{1}{4}\sum_{i=1}^{4}\Phi(\boldsymbol{n}_i), \quad \boldsymbol{x}\in\Omega, \tag{2}$$

where $\boldsymbol{n}_i$ denotes the connected neighbors of $\boldsymbol{x}$. The iteration will terminate when the solution converges. In our framework of harmonic cloning, let $H_i(\boldsymbol{x})$ be the ith function defined over Ω with respect to the ith boundary point, and the initial value of $H_i(\boldsymbol{x})$ is

$$H_i(\boldsymbol{x})=\begin{cases}1, & \text{if } \boldsymbol{x} \text{ is the } i\text{th boundary point},\\ 0, & \text{otherwise.}\end{cases}$$

After the Jacobi iteration (Eq. (2)) is applied to each $H_i(\boldsymbol{x})$, the converged value of $H_i(\boldsymbol{x})$ denotes the unnormalized weight to the ith boundary point. Then the HCs are defined as follows:

$$\lambda_i(\boldsymbol{x})=\frac{H_i(\boldsymbol{x})}{\sum_{j=0}^{n-1}H_j(\boldsymbol{x})}, \quad i=0,1,\ldots,n-1. \tag{3}$$

Once the HCs are computed, the function defined over Ω can then be smoothly interpolated using these coordinates:

$$\tilde{f}(\boldsymbol{x})=\sum_{i=0}^{n-1}\lambda_i(\boldsymbol{x})f(\boldsymbol{p}_i), \quad \boldsymbol{x}\in\Omega, \tag{4}$$

where f is the function defined on $\partial\Omega$ and $\tilde{f}$ is the function defined over Ω, $\partial\Omega=(\boldsymbol{p}_0,\boldsymbol{p}_1,\cdots,\boldsymbol{p}_n=\boldsymbol{p}_0)$ is a closed polygonal boundary curve of Ω.

The HCs take the geodesic distance between the points inside the cloned region and the boundary points into consideration, while MVCs involve straight-line distances instead (DeRose and Meyer, 2006). However, there is a drawback in that the iterative solver used to compute HCs costs much more time than the computation of the MVCs. For high performance, we implement the solver on GPU. Furthermore, inspired from the idea of the multigrid Poisson solver and the application in seamless cloning, the time complexity of Laplacian smoothing can be reduced by performing an adaptive triangulation over the cloned region first, which accelerates the convergence of the solution significantly.

3.2 Harmonic coordinates vs. mean-value coordinates

For comparison, we enumerate the definition of MVCs as follows.

Given a closed 2D polygonal boundary curve $\partial P=(\boldsymbol{p}_0,\boldsymbol{p}_1,...,\boldsymbol{p}_n=\boldsymbol{p}_0),\boldsymbol{p}_i\in\mathbb{R}^2$, the MVCs of a point $\boldsymbol{x}\in\mathbb{R}^2$ with respect to ∂P are given by

$$\lambda_i(\boldsymbol{x})=\frac{w_i}{\sum_{j=0}^{n-1}w_j}, \quad i=0,1,...,n-1, \tag{5}$$

where

$$w_i=\frac{\tan(\alpha_{i-1}/2)+\tan(\alpha_i/2)}{\parallel \boldsymbol{p}_i-\boldsymbol{x}\parallel},$$

and α_i is the angle $\angle\boldsymbol{p}_i\boldsymbol{x}\boldsymbol{p}_{i+1}$.

What is needed for harmonic cloning competing with MVC cloning is the following two additional properties that MVCs do not possess:

1. Non-negativity. This implies that the membrane value of an inside pixel is changed mostly in the same direction as color changed of some nearest boundary sample points, while negative coordinates mean that the membrane value is changed in the opposite direction to the nearest boundary sample points. In Fig. 2, which is an extreme example of cloning an Omega-shaped region, the distances $d_1\approx$

$d_3, d_2 \approx d_4$ indicate the MVCs $\lambda_1 \approx \lambda_3, \lambda_2 \approx \lambda_4$ (calculated using Eq. (5)). Here, the membrane value of the inside pixel $\boldsymbol{x}$ will be negative, if the signs of the values at the boundary samples $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$, and $\boldsymbol{v}_4$ are $+$, $+$, $-$, and $-$, respectively, and the absolute values at samples $\boldsymbol{v}_3$ and $\boldsymbol{v}_4$ are much larger than the ones at $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Because the solver of HCs involves a diffusion iteration, the distances of black lines (d_3 and d_4) in Fig. 2 will disappear and be measured by a diffusion path, which is an approximation of the geodesic distance across the Omega-shaped region. As a result, the membrane value of the inside pixel $\boldsymbol{x}$ will be correct and non-negativity will be retained.

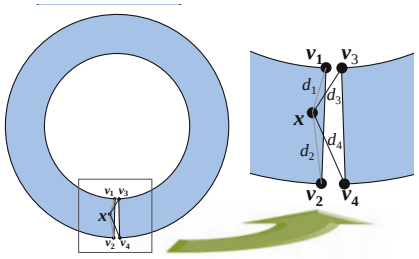


Fig. 2 Demonstration of the non-negativity property while cloning an Omega-shaped region

2. Interior locality. Inspired by DeRose and Meyer (2006), we also quantify the notion of interior locality as follows: in addition to non-negativity, the coordinate functions, which are measured within the entire region, should be monotonic to the distance between the inside pixels and the boundary points. Obviously, the distance involved in computing of MVCs is measured by a straight-line path which is irrespective of the visibility of $\boldsymbol{x}$ from $\boldsymbol{v}_3$ (or $\boldsymbol{v}_4$) in Fig. 2. It is clear that the computing of HCs involves the distance of a diffusion path which approximates the geodesic distance between $\boldsymbol{x}$ and $\boldsymbol{v}_3$ (or $\boldsymbol{v}_4$), and the HCs are proved to possess interior locality (DeRose and Meyer, 2006).

The two properties relieve the HCs from the convexity constraint of the cloned regions, especially concave regions.

4 Algorithm

In this section we explain in detail how to perform seamless image cloning using HCs.

4.1 Overview

Our work is an extension of MVC cloning. We use an adaptive triangulation over the whole cloned region, which is similar to that in Farbman *et al.* (2009), and compute HCs at each mesh vertex, rather than at each pixel, to reduce the computational cost. The algorithm consists of two steps. In the first step, an adaptive triangulation is performed over the cloned region, and we obtain the HCs of each vertex by solving a Laplace equation. In the second step, we evaluate the membrane value at each vertex and compute the color value of each pixel using linear interpolation. While the cloned region is being dragged around in the target image, only the second step is required.

4.2 Harmonic coordinates for image cloning

Let $S \subset \mathbb{R}^2$ and $T \subset \mathbb{R}^2$ be the domains of the source image and target image, respectively. We denote by $g : S \to \mathbb{R}$, $f : T \to \mathbb{R}$ the source and target image intensities over their domains, respectively. The region to be cloned from the source image to the target image is denoted by Ω. Let f^* be an unknown scalar function defined over the interior of Ω. The simplest interpolant f^* of f over Ω is the membrane interpolant defined as the solution of the minimization problem (Pérez *et al.*, 2003):

$$\min_{f^*} \iint_{\Omega} |\nabla f^*|^2 \mathrm{d}x\mathrm{d}y \quad \text{with } f^*|_{\partial\Omega} = f|_{\partial\Omega}, \qquad (6)$$

where $\nabla(\cdot) = \left[\dfrac{\partial \cdot}{\partial x}, \dfrac{\partial \cdot}{\partial y}\right]$ is the gradient operator. Poisson cloning formulates such a minimization problem into the following Laplace equation with Dirichlet boundary conditions:

$$\Delta \tilde{f} = 0 \text{ over } \Omega \quad \text{with } \tilde{f}|_{\partial\Omega} = (f - g)|_{\partial\Omega}, \qquad (7)$$

where $\Delta(\cdot) = \dfrac{\partial^2 \cdot}{\partial x^2} + \dfrac{\partial^2 \cdot}{\partial y^2}$ is the Laplacian operator. The final outcome of the unknown scalar function f^* is then simply defined as

$$f^* = g + \tilde{f}. \qquad (8)$$

This formulation reveals that Poisson cloning in fact constructs a smooth membrane (a harmonic function) $\tilde{f}$ that interpolates the difference $f - g$ between the target and source images on the boundary of Ω across the entire region (Farbman *et al.*, 2009).

As mentioned earlier, we can efficiently compute the HCs over a 2D planar polygon. Before performing the interpolant, we assume that the boundaries of the source and target patches are the same polygon. Let $\partial P = (\boldsymbol{p}_0, \boldsymbol{p}_1, ..., \boldsymbol{p}_n = \boldsymbol{p}_0) = \partial\Omega$ ($\boldsymbol{p}_i \in \mathbb{R}^2$) be the boundary of Ω (in counter-clockwise order). Now we can construct the membrane by interpolating with HCs:

$$r(\boldsymbol{x}) = \sum_{i=0}^{n-1} \lambda_i(\boldsymbol{x})(f-g)(\boldsymbol{p}_i), \quad \boldsymbol{x} \in \Omega, \tag{9}$$

where $r(\boldsymbol{x})$ is the interpolated difference value at that pixel and $\lambda_i(\boldsymbol{x})$ ($i = 0, 1, ..., n-1$) are the HCs with respect to ∂P. The harmonic cloning is then given, similar to Eq. (8), by

$$f^* = g + r. \tag{10}$$

We solve a Laplace equation in fact when we compute the HCs in the first step, so the membrane r constructed by interpolant and the $\tilde{f}$ constructed by solving a large linear system are almost the same theoretically. Therefore, the outcome of harmonic cloning will be almost the same as that of Poisson cloning.

The pseudocode of the unoptimized harmonic cloning procedure similar to Farbman *et al.* (2009) is given in Algorithm 1. The algorithm computes the HCs of each pixel inside the region Ω once the cloned region is selected and then repeatedly performs coordinate-based interpolation for each location of the source patch in the target image.

Algorithm 1 Harmonic-coordinate cloning

```
//perform an iteration to obtain a harmonic map
HarmonicIterations(Ω, ∂P)
//compute the harmonic coordinates of x w.r.t. ∂P
FOR each pixel x ∈ Ω DO
    [λ_i(x)]_{i=0}^{n-1} = HarmonicCoord(x, ∂P)
END FOR
FOR each new position of Ω in target image DO
//compute the intensity difference w.r.t. ∂P
    FOR each vertex p_i of ∂P DO
        diff_i = f(p_i) − g(p_i)
    END FOR
//evaluate the harmonic interpolant at x
    FOR each pixel x ∈ Ω DO
        r(x) = Σ_{i=0}^{n-1} λ_i(x) · diff_i
        f*(x) = g(x) + r(x)
    END FOR
END FOR
```

4.3 Optimizations

It is clear that the memory footprint and computation are substantial if our algorithm is not optimized. The overall storage and time complexity is $O(nm)$, where m is the number of pixels in the cloned region, n is the number of boundary pixels, and $m \gg n$. Since our framework is similar to Farbman *et al.* (2009), the optimizations, adaptive mesh, and hierarchical boundary sampling (Fig. 3) they proposed can be adopted in our method with corresponding modification.

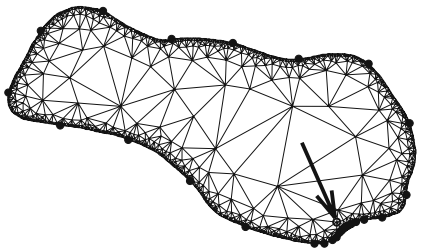


Fig. 3 Adaptive triangle mesh constructed over the cloned region (from Fig. 1a), and hierarchical boundary sampling result (dots on the boundary) of the inside vertex (pointed out by an arrow). The same strategy from Farbman *et al.* (2009)

Inspired by the idea of the multi-grid Poisson solver, we can construct a fast Poisson solver by first solving the problem at a lower resolution, which is the adaptive mesh in this optimization. But as a modification, here we do not need to perform back substitution to finer resolution, since only the HCs for each mesh vertex are used for interpolation. Once the adaptive mesh optimization has been performed on the cloned region, the construction and implementation of HCs at each pixel (Eq. (2)) inside the cloned region are no longer suitable for each mesh vertex of the covered triangle mesh. According to Iserles (1996) and Floater (2003), we use the following cotangent weights for the relaxation method of solving Laplace's equation:

$$\theta_i = \frac{\sigma_i}{\sum_{j=1}^{N} \sigma_j}, \quad \sigma_i = \cot\beta_{i-1} + \cot\gamma_i, \quad i = 1, 2, ..., N, \tag{11}$$

where N is the number of 1-ring neighbors of the center mesh vertex, and β_{i-1} and γ_i are shown in Fig. 4. The Jacobi iteration (Eq. (2)) is now modified

as follows:

$$\Phi(\boldsymbol{x}) = \sum_{i=1}^{N} \theta_i \Phi(\boldsymbol{n}_i), \quad \boldsymbol{x} \in \Omega, \tag{12}$$

where the solver uses a varying number (N) of neighborhood and varying weight (θ_i) of each connected neighbor. This optimization reduces the total complexity of both steps of our algorithm to $O(n^2)$.

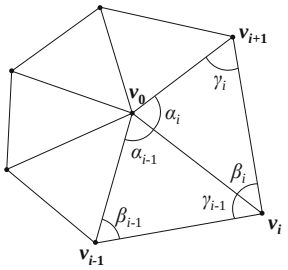


Fig. 4 1-ring neighbors of a vertex on a 2D triangle mesh

As Farbman *et al.* (2009) claimed, the complexity will be reduced to $O(n)$ if the algorithm is optimized by hierarchical boundary sampling. After the HCs have been computed, hierarchical boundary sampling is performed to sample the boundary pixels for each mesh vertex with the same condition as in Farbman *et al.* (2009). Then the HCs for each mesh vertex should be re-normalized and stored with its corresponding boundary pixel's index.

In practice, only a constant number of boundary pixels are involved when we evaluate the membrane value of each mesh vertex. Because we linearly interpolate the membrane values of all m pixels, the total complexity is still $O(m)$.

5 Implementation

With the observation of the parallelization of per-pixel/vertex operations, we implemented harmonic cloning mainly on the GPU. Our implementation takes two steps: (1) Perform an adaptive triangulation and compute the HCs of each vertex once the user selects the region to be cloned; (2) Evaluate the membrane as soon as the position of the cloned region Ω changes and finally compute the color of the cloned region in the target image. The second step is similar to Farbman *et al.* (2009).

5.1 Computing harmonic coordinates

Thanks to Shewchuk (2005)'s triangle library, given appropriate parameters, the generation of an adaptive triangle mesh is very fast with satisfactory quality. Once the cloned region was selected, our system generated the adaptive triangle mesh, and a vector of HCs was computed and stored with each mesh vertex. In our experiment, we found that the Laplacian smoothing converges in about 50–80 iterations when the maximum change to all the mesh vertices dropped below a specified threshold $\tau = 10^{-5}$, since the resolution of the triangle mesh is low, especially away from the boundary (Fig. 3). For convenience, we set a fixed number of iterations s=100 in our GPU implementation, and the performance statistics are shown in Table 1. Besides, we performed hierarchical boundary sampling on host CPU and the sampling result was uploaded into GPU's device memory. According to this sampling result, the HCs at each mesh vertex were re-normalized. The final HCs were stored in GPU's device memory for use in the next step.

Table 1 Performance statistics

Number of CPs	Number of BPs	Number of MVs	Time (s)*	
			MVCs	HCs
11 557	308	511	0.005	0.010
48 560	794	1338	0.033	0.039
135 618	1263	2200	0.091	0.094
613 341	2086	3885	0.285	0.310
3 826 809	5862	11 381	2.336	2.863
7 663 209	7462	14 500	3.754	4.309

* For the computation of coordinates only. CPs: cloned pixels; BPs: boundary pixels; MVs: mesh vertices. MVCs: mean-value coordinates; HCs: harmonic coordinates

5.2 Evaluating the membrane

The membrane value at each mesh vertex was evaluated by a weighted combination of the boundary error and the HCs in parallel, and stored in the texture along with the mesh vertices as well as the source and target images. The error membrane at each pixel was obtained by a linear interpolation via hardware rasterization, that is, rasterizing the adaptive triangle mesh with the membrane value as the color of each vertex. The final colors inside the cloned region in the target image were replaced by the addi-

tion of pixel membrane and the colors of the source patch.

6 Results and extensions

We implemented the algorithm in CUDA and tested it on a PC with Intel Core2 Duo 3.0 GHz CPU, 2.0 GB memory and an NVIDIA GeForce GTX 280 graphic card.

We also implemented Poisson image editing (Pérez *et al.*, 2003) and MVC cloning (Farbman *et al.*, 2009) in our system, and tested our algorithm with several examples, using a variety of images. In the comparison (Fig. 5), due to the lack of non-negativity and interior locality, the MVCs-based approach cannot smoothly clone the patch in the image. Compared with Poisson cloning, the discontinuity of the cloned region in the final image was obvious, whereas our method can generate results with as high a quality as with Poisson cloning. More results of image cloning are shown in Fig. 6.

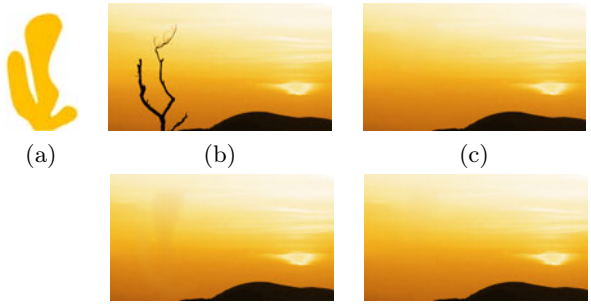


Fig. 5 Hiding the tree in the scenario of sunset with the patch image. (a) The patch image with pure color; (b) Original (target) image; (c) Poisson cloning; (d) MVC cloning; (e) Harmonic cloning. Note that the results of Poisson cloning and harmonic cloning are nearly the same. The shadow-like pattern in (d) is produced because the MVC membrane in each branch region is affected by values along the boundary of each other, despite there being no straight sight between the two branches

We also compared our method with the one based on the VSSD solver (Jeschke *et al.*, 2009) by cloning rate at runtime on the same graphic card (GeForce 8800 GTX). Since our method performed only an interpolation at runtime, we achieved a cloning rate of over 350 Mpixels/s (cloning about

Fig. 6 Application in changing the background. Top-left: the source image. Others: copying the couple into three other background images. Note that the tone of the foreground image can be changed automatically according to the background image

330×10^3 pixels), with respect to 63 Mpixels/s reported in Jeschke *et al.* (2009) by setting the default number of iterations to 8–16. However, when the user clones a region to a target image with higher frequency, the VSSD solver requires more iterations to avoid the mach banding artifacts by applying the 'shrink half' strategy (Fig. 7). Hence, the cloning rate will further decrease. In conclusion, under the scheme of coordinates interpolation, our method has about one order of magnitude faster than the VSSD solver in runtime processing.

Since our harmonic cloning is one of the coordinate-based methods as well as MVC cloning, the extensions of MVC cloning (Farbman *et al.*, 2009) can also be performed in our framework. Fig. 8 shows the result of invariant transforming and Fig. 9 shows the result of selective boundary suppression.

Our method can also be extended to seamless video cloning. The main constraint in video cloning is that the result membrane should be temporally smooth to avoid flickering and be boundary smooth to avoid boundary jumping in the guidance field. Similar to Farbman *et al.* (2009), we formulated video cloning as

$$F(\boldsymbol{x}) = \alpha(\boldsymbol{x}) * f_0(\boldsymbol{x}) + (1 - \alpha(\boldsymbol{x})) \sum_{i=1}^{k} \beta_i f_i(\boldsymbol{x}), \quad (13)$$

where $F(\boldsymbol{x})$ is the final membrane value of mesh vertex $\boldsymbol{x}$, $f_i(\boldsymbol{x})$ is the membrane value of mesh vertex $\boldsymbol{x}$ in the last ith frame ($f_0(\boldsymbol{x})$ is the membrane of the current frame), β_i is the normalized weight of the last ith frame (decaying with frame spacing count:

(a) (b) (c) (d) (e)

Fig. 7 Comparison with the variable stencil size diffusion (VSSD) solver. (a) Source patch in the target image; (b) Our result; (c) Result of using the VSSD solver with 32 iterations (note the mach banding artifacts); (d) & (e) Result of using the VSSD solver combining the shrink half (SH) strategy with 16 and 32 iterations, respectively. While the cloning patch to the target image with high frequency, more iterations as well as the SH strategy are required to produce a plausible result

$\Delta k^{-0.75}$), and $\alpha(\boldsymbol{x}) = 2^{-d(\boldsymbol{x})}$ implies the decaying of distance from boundary. Snapshots from interactive seamless video cloning are shown in Fig. 10, and it turned out that our approach achieved interactive performance in video cloning.

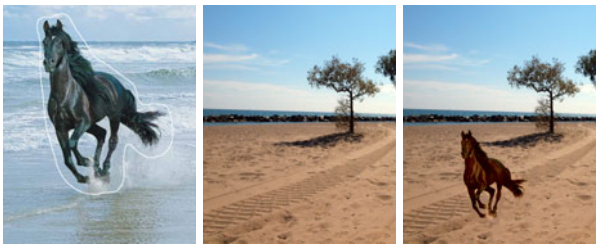
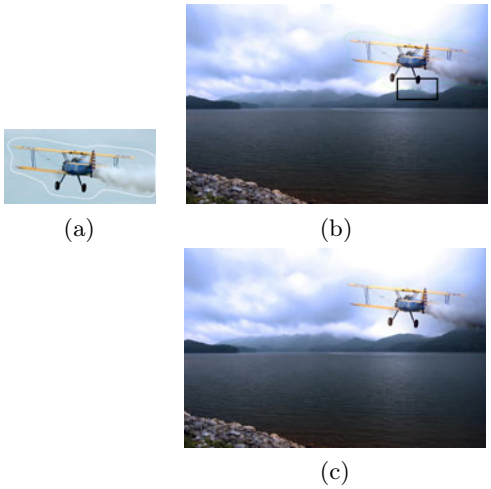
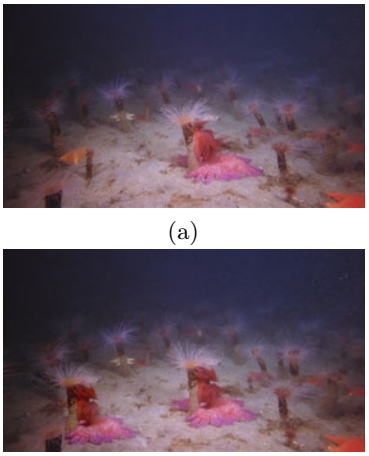


Fig. 8 Invariant transforming: transforming and copying a horse running on the beach to sandy road. No extra operations, except transforming the position of boundary samples, are needed for performing invariant transforming



(a) (b) (c)

Fig. 9 The source image (a), the smudge result (surrounded by a rectangle) (b), and the result after performing a selective boundary suppression (c)



(a) (b)

Fig. 10 Snapshots from interactive seamless video cloning. The size of the source video is 1280×720 pixels. (a) Source video; (b) Cloning the halobios to the left-bottom corner

7 Discussion

Real-time seamless image cloning is an interesting task in image composition. We have tested a number of examples to compare harmonic cloning with previous methods. Our conclusion is that harmonic cloning produces satisfactory results.

Our method involves a highly parallel algorithm, however, the performance will decline violently if the program runs on CPU only. Although we can compute HCs in a step-by-step manner for large scale cloning, the performance is constrained by memory exchange between the host and the device because of limited capacity of GPU memory.

In conclusion, the harmonic cloning proposed in this paper can generate as high a quality of results as with Poisson cloning, and achieve as high a perfor-

mance as with MVC cloning. As an extension from 2D to 3D space, it will be of interest to apply HCs to mesh editing in the future.

References

Agarwala, A., 2007. Efficient gradient-domain compositing using quadtrees. *ACM Trans. Graph.*, **26**(3), Article No. 94, p.1-5. [doi:10.1145/1276377.1276495]

DeRose, T., Meyer, M., 2006. Harmonic Coordinates. Pixar Technical Memo 06–02. Pixar Animation Studios. Available from http://graphics.pixar.com/HarmonicCoordinates/

Farbman, Z., Hoffer, G., Lipman, Y., Cohen-Or, D., Lischinski, D., 2009. Coordinates for instant image cloning. *ACM Trans. Graph.*, **28**(3), Article No. 67, p.1-9. [doi:10.1145/1531326.1531373]

Floater, M.S., 2003. Mean value coordinates. *Comput. Aided Geom. Des.*, **20**(1):19-27. [doi:10.1016/S0167-8396(03)00002-5]

Georgiev, T., 2004. Photoshop Healing Brush: a Tool for Seamless Cloning. Workshop on Applications of Computer Vision, p.1-8.

Iserles, A., 1996. A First Course in Numerical Analysis of Differential Equations. Cambridge University Press, New York, USA.

Jeschke, S., Cline, D., Wonka, P., 2009. A GPU Laplacian solver for diffusion curves and Poisson image editing. *ACM Trans. Graph.*, **28**(5), Article No. 116, p.1-8. [doi:10.1145/1618452.1618462]

Jia, J., Sun, J., Tang, C.K., Shum, H.Y., 2006. Drag-and-drop pasting. *ACM Trans. Graph.*, **25**(3):631-637. [doi: 10.1145/1141911.1141934]

Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T., 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, **26**(3), Article No. 71, p.1-9. [doi:10.1145/1276377.1276466]

Kazhdan, M., Hoppe, H., 2008. Streaming multigrid for gradient-domain operations on large images. *ACM Trans. Graph.*, **27**(3), Article No. 21, p.1-10. [doi:10.1145/1360612.1360620]

Levin, A., Zomet, A., Peleg, S., Weiss, Y., 2004. Seamless Image Stitching in the Gradient Domain. 8th European Conf. on Computer Vision, p.377-389. [doi:10.1145/1276377.1276495]

McCann, J., Pollard, N.S., 2008. Real-time gradient-domain painting. *ACM Trans. Graph.*, **27**(3), Article No. 93, p.1-7. [doi:10.1145/1360612.1360692]

Orzan, A., Bousseau, A., Winnemöller, H., Barla, P., Thollot, J., Salesin, D., 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph.*, **27**(3), Article No. 92, p.1-8. [doi:10.1145/1360612.1360691]

Pérez, P., Gangnet, M., Blake, A., 2003. Poisson image editing. *ACM Trans. Graph.*, **22**(3):313-318. [doi:10.1145/882262.882269]

Shewchuk, J.R., 2005. A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. Available from http://www.cs.cmu.edu/~quake/triangle.html [Accessed on Mar. 22, 2010].

Szeliski, R., 2006. Locally adapted hierarchical basis preconditioning. *ACM Trans. Graph.*, **25**:1135-1143. [doi:10.1145/1141911.1142005]