Combinatorial Laplacian Image Cloning

Alex Cuadros-Vargas
School of Computer Cience
San Pablo Catholic University
Arequipa, Peru
Email: alex@ucsp.edu.pe

Luis Gustavo Nonato
ICMC/SME
Universidade de São Paulo
São Carlos, SP, Brazil
Email: gnonato@icmc.usp.br

Valerio Pascucci
SCI Insitute
University of Utah
Salt Lake City, USA
Email: pascucci@sci.utah.edu

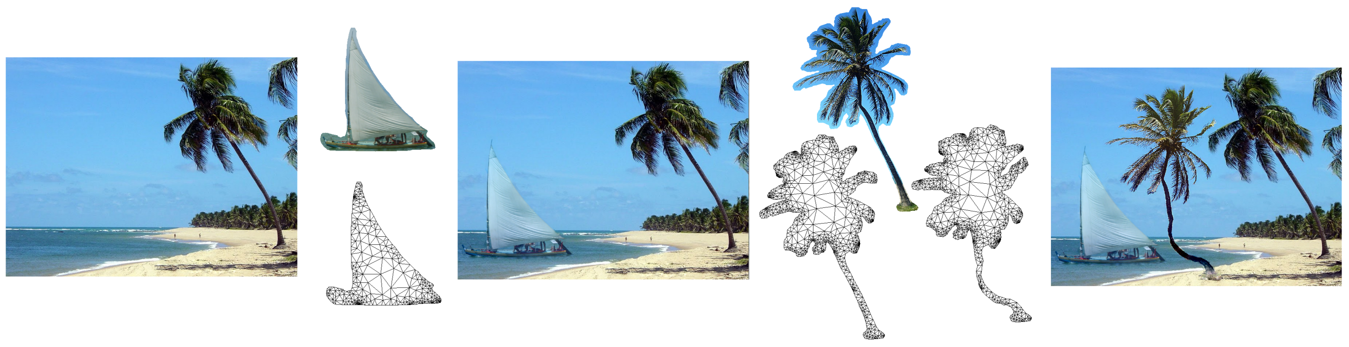


Fig. 1. Image cloning and deformation. From left to right, a "jangada" image patch is meshed and cloned to the "praia do forte". A palm tree image patch is meshed, deformed, and cloned to the "praia do forte"

***Abstract*—Seamless image cloning has become one of the most important editing operation for photomontage. Recent coordinate-based methods have lessened considerably the computational cost of image cloning, thus enabling interactive applications. However, those techniques still bear severe limitations as to concavities and dynamic shape deformation. In this paper we present novel methodology for image cloning that turns out to be highly efficient in terms of computational times while still being more flexible than existing techniques. Our approach builds on combinatorial Laplacian and fast Cholesky factorization to ensure interactive image manipulation, handling holes, concavities, and dynamic deformations during the cloning process. The provided experimental results show that the proposed technique outperforms existing methods in requisites such as accuracy and flexibility.**

***Keywords*-Image Cloning; Laplacian System; Imesh;**

I. Introduction

Editing operations based on gradient domain have emerged as one of the most effective mechanisms to perform tasks such as image compression [1], diffusion curve image representation [2], and image painting [3]. In particular, gradient based image cloning has become the most useful tool for seamlessly merging image patches together in photomontage applications [4], [5], [6], [7]

Image cloning techniques ultimately solve a large sparse Poisson system, which is a very time consuming task even for fast numerical solvers. Such a computational burden has motivated a number of alternative methods ranging from hierarchical schemes [8], [9], [10] to GPU Poisson solvers [11], [3], [2]. Greater interactivity has been recently achieved by mesh-based techniques, which dispense the Poisson equation altogether by assigning a set of coordinates to each vertex of a triangle mesh that represents the image patch to be cloned. The coordinates are used to smoothly spread information from the boundary towards the interior of the mesh, enabling to build the membrane required for seamless cloning without solving any linear system. Several different schemes have been proposed to assign coordinates to the vertices of the mesh [12], being the mean value coordinates (MVC) [13] the most well-known method.

The combination of triangle meshes and mean value coordinates results in a computationally efficient image cloning methodology, since the bulk of the computation (the mesh generation and the MVC construction) may be performed in a pre-processing stage. Although cost effective, MVC-based image cloning bears weaknesses that restrict its use to simply connected and slightly concave image patches. Moreover, the whole set of coordinates have to be rebuilt if the boundary shape changes, which impairs interactive applications where cloning and shape deformation should be carried out simultaneously.

This paper presents a novel methodology for image cloning that is highly efficient in terms of computational times while still being more flexible than the MVC approach, allowing for handling holes, concavities, and interactive deformation during the cloning process. The proposed method relies on the combinatorial Laplacian [14] and a fast supernodal sparse Cholesky factorization [15] to carry out the image cloning as well as shape deformation in interactive rates. The rationale behind

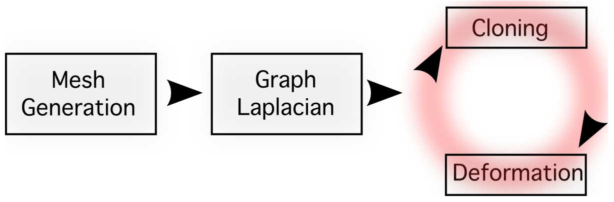
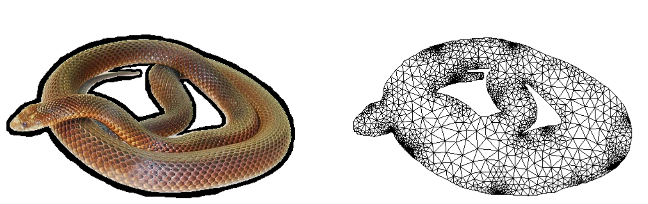


Fig. 2. Image cloning/deformation pipeline.



(a) Image patch (b) Triangle mesh

Fig. 3. Image patch (a) and the corresponding mesh produced by *Imesh* (b).

our approach is supported by the fact that a true Laplace membrane is not mandatory for seamless image cloning and could be replaced by a surface with correct boundary and smooth interior. The surface resulting from the combinatorial Laplacian fulfils those requirements, enabling visually pleasing results while avoiding to restart the system due to changes in the boundary of the image patch. Furthermore, our method is able to clone image patches that become self-intersecting during deformation, a trait not present in other techniques.

Similar to other mesh-based methods, our approach requires a triangle mesh representation for the image patch undergoing the cloning process. The triangle mesh must hold specific properties such as adaptivity and boundary conformity in order to avoid unnecessary processing and low quality stitching results. Adaptivity and boundary conformity are naturally ensured by the image-based mesh generation technique called *Imesh* [16], motivating us to adopt it in our pipeline. The good performance of the proposed methodology is confirmed in a set of comparisons and experimental results.

Contributions We can summarize the contributions presented in this paper as:

- A novel technique that combines combinatorial Laplacian and fast Cholesky factorization so as to accomplish image cloning in interactive rates.
- A mesh deformation scheme integrated into the image cloning process.
- A scheme to set Dirichlet boundary conditions that allows for handling self intersection of the mesh while performing the seamless cloning.

To the best of our knowledge, mesh deformation and image cloning have never been performed simultaneously. Moreover, this is the first time the problem of self intersecting image patches is handled in the context of image cloning.

II. Combinatorial Laplacian-based Seamless Cloning

The proposed image cloning technique follows the pipeline presented in Fig. 2, which comprises three main steps: mesh generation, combinatorial Laplacian, and interactive cloning/deformation. The mesh generation step decomposes the source image patch in a triangle mesh (Section II-A) from which the combinatorial Laplacian is built (Section II-B). Cholesky factorization is then applied to the resulting symmetric matrix, thus allowing for fast system solves. Cloning (Section II-C) and deformation (Section III) are accomplished by simply changing boundary conditions in the factored matrix, which can be done efficiently using the penalty method. Details on each step are provided in the following subsections.

A. Mesh Generation

Our approach relies on triangle meshes to perform the image cloning as well as the deformation. More specifically, given an image patch $\mathscr{P}$ picked out from a source image $\mathscr{S}$, we must build a triangle mesh $T_{\mathscr{P}}$ from $\mathscr{P}$ such that vertices in $T_{\mathscr{P}}$ lies on pixels of $\mathscr{P}$. The triangle mesh $T_{\mathscr{P}}$ must also ensure that boundaries of $\mathscr{P}$ are preserved accurately. Moreover, the triangle mesh should not be overly dense in order to lessen the computational load. Those properties are ensured by the image-based mesh generation technique called *Imesh*, an algorithm for directly generating triangle meshes from 2D digital images (see [16] and [17] for a complete and detailed description of *Imesh*).

Three main steps make up the *Imesh*'s pipeline: mesh generation, mesh partition, and mesh improvement. It takes in a 2D digital image and a texture classifier, providing as output a quality triangle mesh that conforms to the boundary of the regions detected by the texture classifier.

Since in our context the detection of inner structures in the image patch is not so relevant, we can disregard the texture classifier, which naturally enforces *Imesh* to refine the mesh only nearby the outer boundaries of $\mathscr{P}$, as illustrated in Fig. 3. Notice that *Imesh* produces an adaptive mesh as output.

B. Combinatorial Laplacian

The *combinatorial Laplacian* associated to the triangle mesh $T_{\mathscr{P}}$ is the symmetric matrix $L = (l_{ij})_{n\times n}$ (*Laplacian matrix*) given by:

$$l_{ij} = \begin{cases} deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j, v_i \text{ and } v_j \text{ share an edge of } T_{\mathscr{P}} \\ 0 & \text{otherwise} \end{cases}$$

where v_i and $deg(v_i)$ account for the ith vertex of $T_{\mathscr{P}}$ and the degree (valence) of that vertex, respectively.

The combinatorial Laplacian holds some remarkable properties. For instance, it depends only on the topology of the triangle mesh, that is, any mesh isomorphic to $T_{\mathscr{P}}$ should have the same combinatorial Laplacian as $T_{\mathscr{P}}$. Therefore, there is no need to rebuild L if only the geometry of $T_{\mathscr{P}}$ (vertices coordinates) is changed, as for example during mesh deformation.

Moreover, the combinatorial Laplacian can be seen as a zero order approximation of the Laplace operator defined on the underlying space of $T_{\mathscr{P}}$. Therefore, it can be used to approximate the solution of $\nabla^2 f = 0$, that is,

$$\nabla^2 f = 0 \approx Lf = 0 \tag{1}$$

The linear system (1) is rank deficient, that is, it admits a non-trivial solution. In order to ensure a unique solution constraints must be imposed into the system.

Constraints handling There are many different ways to incorporate constraints into the Laplacian system. In this work we choose the so called *penalty method* [18] due to its stability, efficiency and support for factorization updates.

The penalty method can be stated as follows: let C be the indexes of vertices where constraints must be imposed and b be the vector with zero in all entries b_i such that $i \notin C$ and $b_i = c_i$ if $i \in C$, where c_i is the constraint value (Dirichlet boundary condition) to be set in v_i. The penalty method transforms the constrained problem $Lf = 0, \quad f_i = c_i$ if $i \in C$ into the unconstrained system

$$(L+P)f = Pb \tag{2}$$

where P is the diagonal penalty matrix with non-zero diagonal elements $p_{ii} = \alpha$ only if $i \in C$.

The penalty method holds several good properties. For instance, in contrast to direct elimination, the penalty method preserves the symmetry and positive semi-definiteness of the system (assuming $\alpha > 0$), thus allowing for Cholesky factorization, which can be performed very efficiently using up-to-date numerical library's such as Cholmod [19]. The Cholmod library implements supernode updating/downdating schemes [15], which allow for modifying the Cholesky factorization only locally when the matrix L is changed according to $\tilde{L} = L + MM^T$. Notice that if we set $p_{ii} = \sqrt{\alpha}$, the penalty method can be handled through supernode updates, enabling constraints to be switched in interactive rates.

C. Seamless Cloning

Let $\mathscr{P} \subset \mathscr{S}$ be the image patch to be cloned in a target image $\mathscr{T}$. Moreover, let $f_{\mathscr{S}} : \mathscr{S} \to \mathbb{R}$ and $f_{\mathscr{T}} : \mathscr{T} \to \mathbb{R}$ be the intensities of the source and target images, respectively. The image cloning problem resumes to find an intensity function $f : \mathscr{P} \to \mathbb{R}$ such that $\mathscr{P}$ can be superimposed on $\mathscr{T}$ while the seam is not noticeable. One of the first solution for this problem was formulated as a Poisson equation:

$$\nabla^2 f = \operatorname{div}\nabla f_{\mathscr{S}}, \text{ with the constraint } f|_{\partial\mathscr{P}} = f_{\mathscr{T}} \tag{3}$$

where $\partial\mathscr{P}$ is the boundary of the patch $\mathscr{P}$. In less mathematical terms, the formulation (3) seeks an intensity function whose gradient field is as close as possible to the source gradient field while agreeing with the target intensities on the boundary of the patch.

It can be shown that the formulation (3) is equivalent to the following Laplace equation [6]:

$$\nabla^2 \tilde{f} = 0, \text{ with the constraint } \tilde{f}|_{\partial\mathscr{P}} = f_{\mathscr{T}} - f_{\mathscr{S}}, \tag{4}$$

The final intensity function is obtained from the harmonic membrane $\tilde{f}$ by:

$$f = f_{\mathscr{S}} + \tilde{f} \tag{5}$$

If $\mathscr{P}$ is represented by a triangle mesh then the combinatorial Laplacian can be used to approximate (4), exactly as described in Eq. 1. This fact combined with the penalty method to constraint (4) allow for a highly efficient scheme to solve the image cloning problem.

III. Mesh Deformation

In the last section we showed how the combinatorial Laplacian can be used to approximate the solution of the image cloning problem. In this section we show that it can also be used to perform mesh deformation.

Following the mathematical construction presented by Yu et al. [20], let (x,y) be the Cartesian coordinates of each vertex in a triangle mesh T. The first coordinate x can be seen as a scalar field defined on T (the same is true for the second coordinate y). Therefore, assuming linear interpolation, the gradient ∇x is well defined and constant within each triangle of T. In fact, the gradient of each coordinate give rise to two vector fields $\mathbf{w}_x$ and $\mathbf{w}_y$, that is,

$$\nabla x = \mathbf{w}_x \text{ and } \nabla y = \mathbf{w}_y \tag{6}$$

By applying the divergence operator in both sides of (6) we get

$$\nabla^2 x = \operatorname{div}(\mathbf{w}_x) \text{ and } \nabla^2 y = \operatorname{div}(\mathbf{w}_y) \tag{7}$$

Equations (7) say that the coordinates (x,y) of the vertices can be recovered from their gradient fields by solving two Poisson equations. Therefore, if the gradient fields are modified, the coordinates of the vertices will change accordingly, so deformations can be performed by handling the vector fields $\mathbf{w}_x$ and $\mathbf{w}_y$. Before discussing how to modify those vector fields in order to get pleasing deformations, we need to discretize (7).

The Laplacian operator ∇^2 can be approximated by the combinatorial Laplacian, as discussed in Section II-B. The right hand side terms $\operatorname{div}(\mathbf{w}_x)$ and $\operatorname{div}(\mathbf{w}_y)$ can be approximated using the finite element method with linear elements. Some simple computation shows that the divergent $\operatorname{div}(\mathbf{w}_x)$ evaluated in a vertex v_i becomes

$$\operatorname{div}(\mathbf{w}_x)(v_i) = \frac{1}{2}\sum_{t_j \in N_i} \mathbf{e}_j \cdot \mathbf{w}_{xj}$$

where N_i is the set of triangles that share the vertex v_i, $\mathbf{w}_{xj}$ is $\mathbf{w}_x$ in the triangle t_j, and $\mathbf{e}_j = (y_j^0 - y_j^1, x_j^0 - x_j^1))$ where (x_j^0, y_j^0) and (x_j^1, y_j^1) are the (x,y) coordinates of the two vertices in t_j opposite to v_i (ccw orientation).

Handling $\mathbf{w}_x$ and $\mathbf{w}_y$ In the very beginning, the fields $\mathbf{w}_x$ and $\mathbf{w}_y$ are set equal to the gradient of the x and y coordinates. The user can then modify those vector fields interactively in order to deform the underlying triangle mesh by reconstruction the vertices coordinates from Eqs.(7).

In our implementation we rotate $\mathbf{w}_x$ and $\mathbf{w}_y$ according to the user manipulation. More specifically, denoting by o the origin

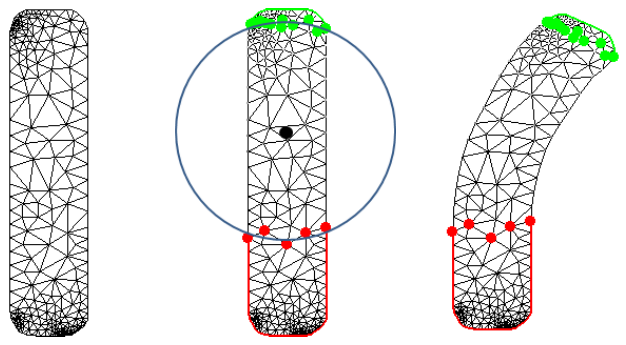

(a) Patch (b) Deformation region (c) Deformation

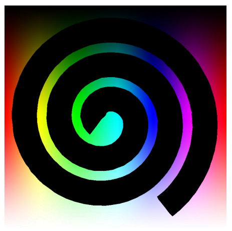
Fig. 4. Deformation is applied the triangle patch (a). The region where the deformation takes place is defined by the user using a circle (b). Dragging a point of the mesh deforms the region.

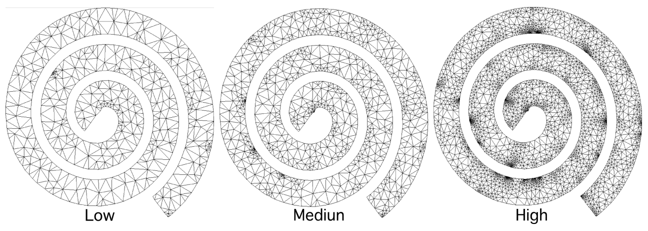
of the coordinate system, the user clicks on a point p_0 and drag the cursor to a new position p_1, allowing to compute the angle θ between the vectors $\overrightarrow{op}_0$ and $\overrightarrow{op}_1$. The vector field $\mathbf{w}_x$ ($\mathbf{w}_y$) is then rotated by θ cw or ccw depending on the direction the user moved to cursor from p_0 to p_1.

In order to restrict the deformation to a region of the image patch, we multiply the angle θ by a coefficient that tends to zero when one moves away from p_0. In practice, the coefficient is defined from a Gaussian function centered in p_0 with variance specified the by diameter of circle defined interactively by user, as illustrated in Fig. Fig. 4.

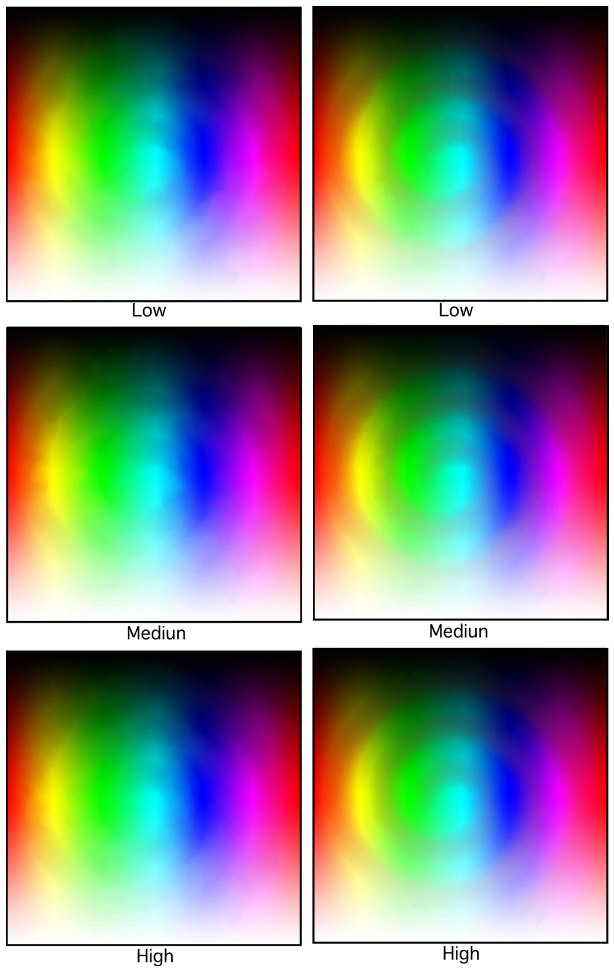
IV. Results and Comparisons

In order to confirm the quality of the proposed approach we provide comparisons against the well-known *mean value coordinates* method, which also relies on triangle meshes to perform the cloning process. Fig. 5 shows the result of cloning a spiral image patch in a colored backgroup image using meshes with three different levels of refinement (Fig. 5b). Notice from Fig. 5c) that the combinatorial Laplacian produces better cloning results in all cases. Moreover, the seam resulting from combinatorial Laplacian fades out when the triangle mesh is refined successively, a characteristic not observed in the MVC cloning. Computational times are shown in Table I. A core 2 duo 2.0Ghz processor was used to run both approaches. The time spent during deformation is not considered in the table (the formation is quite interactive). The *Pre-Processing* columns account for the Cholesky factorization and the Mean Value Coordinates computation (CPU implementation). The last two columns correspond to computational times for the membrane computation. Notice that the combinatorial Laplacian is up to two orders of magnitude faster than the CPU implementation of MVC. It is important to say that several mechanisms have been proposed to speed up MVC, including GPU implementation [13]. However, we have not implement any strategy to push MVC computational times down. The idea is to show that a straightforward implementation of the combinatorial Laplacian is already able to perform image cloning in real time. For MVC, interactive rates are only


(a) Spiral



(b) Meshes



(c) Cloning Result

Fig. 5. Comparison between the combinatorial Laplacian and the MVC method. (a) Spiral image to be cloned in colored background; (b) Meshes with different levels of refinement (.5K, 1K, and 3K vertices) used to perform the cloning; (c) Cloning results from combinatorial Laplacian (left column) and MVC (right column) using the triangle meshes in (b).

TABLE I

COMPUTATIONAL TIMES (SECONDS). FIRST COLUMN SHOWS THE NAME OF THE MODEL. SECOND AND THIRD COLUMNS SHOW THE NUMBER OF PIXELS AND VERTICES IN THE INPUT IMAGE AND TRIANGLE MESH, RESPECTIVELY. NEXT FOUR COLUMNS SHOW COMPUTATIONAL TIMES FOR THE PRE-PROCESSING AND HARMONIC MEMBRANE COMPUTATION FOR A CPU IMPLEMENTATION OF MVC AND OUR APPROACH.

			Pre-processing		Harmonic membrane	
	Pixels	Vertices	MVC	Our	MVC	Our
Mouth	121x85	53	0.000	0.000	0.000	0.000
Raft	342x353	662	0.405	0.016	0.078	0.001
Spiral	770x743	1019	1.014	0.015	0.187	0.002
Horse	392x399	1241	1.388	0.031	0.265	0.002
Palm	577x586	1254	1.341	0.031	0.265	0.002
Snake	470x280	2524	4.181	0.078	0.858	0.003

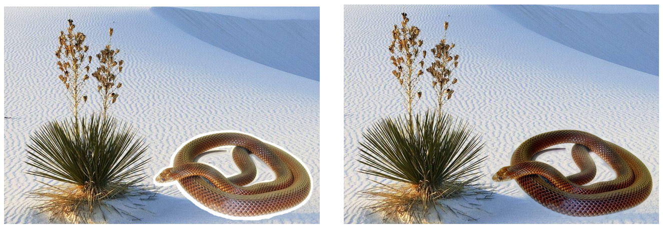


Fig. 6. Image patches with holes.

obtained with a more intricate CPU implementation.

The combinatorial Laplacian-based image cloning does not face problems when handling image patches with holes, as Fig. **??** shows. Other mesh-based schemes such as MVC cloning are not able to deal with holes, since MVC is not well defined in this kind of domain.

Combining cloning and deformation provides a powerful tool for modifying specific parts of an image. This fact is illustrated in Fig. 7, where the region surrounding the Delaunay's mouth is grabbed, meshed, deformed and cloned back onto the original image.

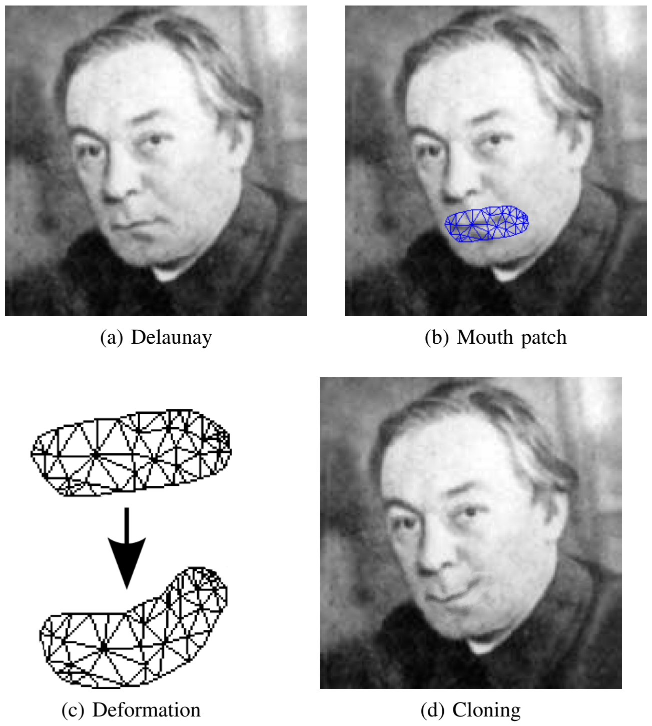
To conclude this section we show an example where the triangle mesh representing an image patch intersects itself during deformation. Fig. 8a shows the image patch of a horse to be cloned into a background image. During the cloning process the triangle mesh is interactively deformed Fig. 8b, resulting several photomontages Fig. 8c.

V. DISCUSSION AND LIMITATION

The comparisons presented in Section IV clearly show the effectiveness of the proposed cloning method, surpassing, in requisites such as accuracy and flexibility, the state-of-art mesh-based method. Moreover, the combinatorial Laplacian turns out to be quite efficient as to computational times, enabling the user to freely displace and deform the cloning patch. It is worth pointing out once again that in our context, the matrices resulting from the meshes built to perform the cloning and deformation are always very sparse and not so big. Moreover, the meshes do not directly depend on the image resolution and they do not change during manipulation, thus the Cholesky factorization is indeed computed only once.

The capability to deform image patches during the cloning process is another unique trait of our approach. The ability to handle self-intersections is also a particularity of the proposed method. In fact, no other technique described in the literature is able to accomplish image cloning and deformation in interactive rates while still handling self-intersections.

One aspect to be observed is the color of the source image can be affected during the cloning process. Mechanism such as the Matting [13] can be used to mitgate this problem and we are currently working on the front. Other point that could be improved in our methodology is that the meshes produced by *Imesh* are optimized to ensure good quality triangles. Therefore, vertices may concentrate unduly in specific parts

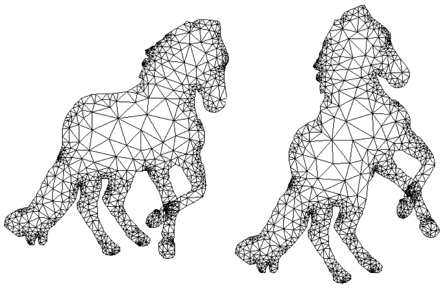


(a) Delaunay (b) Mouth patch

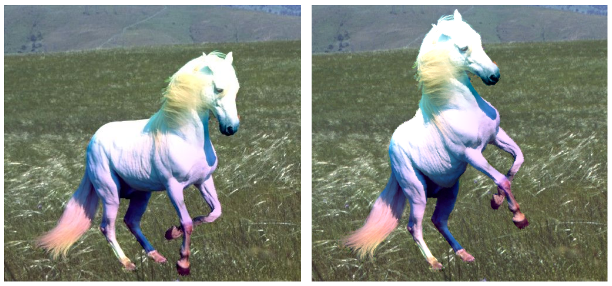
(c) Deformation (d) Cloning

Fig. 7. Combining cloning and deformation. An image patch is grabbed from the Delaunay's mouth (b), the mesh is deformed (c) and cloned back (d) in the original photo (a).

(a) Horse Patch



(b) Mesh deformation



(c) Deformation

Fig. 8. The triangle mesh of a horse image patch (a) is deformed (b). The triangle mesh self-intersects but the proposed approach is still able to produce satisfactory cloning results (c).

of the image patch, which may increase computational times. This drawback can be solved by modifying *Imesh* so as to avoid adding new vertices in regions where the lengths of the edges are smaller than a threshold.

VI. Conclusion

In this work we proposed the use of combinatorial Laplacian as basic tool for image cloning/deformation applications. The evaluation we provided shows that our approach outperforms existing techniques in terms of accuracy as well as computational times. Besides allowing for combining cloning and deformation in a unified framework, the proposed methodology handles self-intersection during deformation and cloning operations, a trait not found in other mesh-based methods. In summary, flexibility and effectiveness in terms of computational times render the proposed method one of the most attractive alternatives in the context of image editing. We are currently investigating how to combine cloning with depth images so as to overlapping objects while still accomplishing the cloning process satisfactorily.

References

[1] R. Fattal, D. Lischinski, and M. Werman, “Gradient domain high dynamic range compression,” *ACM Trans. Graph.*, vol. 21, pp. 249–256, July 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566573

[2] S. Jeschke, D. Cline, and P. Wonka, “A gpu laplacian solver for diffusion curves and poisson image editing,” *ACM Trans. Graph.*, vol. 28, pp. 116:1–116:8, December 2009. [Online]. Available: http://doi.acm.org/10.1145/1618452.1618462

[3] J. McCann and N. S. Pollard, “Real-time gradient-domain painting,” *ACM Trans. Graph.*, vol. 27, pp. 93:1–93:7, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360692

[4] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, “Interactive digital photomontage,” *ACM Trans. Graph.*, vol. 23, pp. 294–302, August 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015718

[5] J. Jia, J. Sun, C.-K. Tang, and H.-Y. Shum, “Drag-and-drop pasting,” *ACM Trans. Graph.*, vol. 25, pp. 631–637, July 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141934

[6] P. Pérez, M. Gangnet, and A. Blake, “Poisson image editing,” *ACM Trans. Graph.*, vol. 22, pp. 313–318, July 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882269

[7] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, “Seamless image stitching in the gradient domain,” in *In Proceedings of the European Conference on Computer Vision*, 2006, pp. 377–389.

[8] A. Agarwala, “Efficient gradient-domain compositing using quadtrees,” *ACM Trans. Graph.*, vol. 26, July 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276495

[9] M. Kazhdan and H. Hoppe, “Streaming multigrid for gradient-domain operations on large images,” *ACM Trans. Graph.*, vol. 27, pp. 21:1–21:10, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360620

[10] R. Szeliski, “Locally adapted hierarchical basis preconditioning,” *ACM Trans. Graph.*, vol. 25, pp. 1135–1143, July 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1142005

[11] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder, “Sparse matrix solvers on the gpu: conjugate gradients and multigrid,” *ACM Trans. Graph.*, vol. 22, pp. 917–924, July 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882364

[12] R. Wang, W. feng Chen, M. hao Pan, and H. jun Bao, “Harmonic coordinates for real-time image cloning,” *Journal of Zhejiang University - Science C*, vol. 11, pp. 690–698, 2011.

[13] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski, “Coordinates for instant image cloning,” *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–9, 2009.

[14] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P.Seidel, “Laplacian surface editing,” in *SGP ’04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. New York, NY, USA: ACM, 2004, pp. 175–184.

[15] T. Davis and W. Hager, “Dynamic supernodes in sparse cholesky update/downdate and triangular solves,” *ACM Trans. Math. Softw.*, vol. 35, no. 4, pp. 1–23, 2009.

[16] A. J. Cuadros-Vargas, M. A. S. Lizier, R. Minghim, and L. G. Nonato, “Generating segmented quality meshes from images,” *Journal of Mathematical Imaging and Vision*, vol. 33, pp. 11–23, 2009.

[17] M. Lizier, D. Martins-Jr., A. Cuadros-Vargas, R. Cesar-Jr., and L. Nonato, “Generating segmented meshes from textured color images,” *J. Vis. Comun. Image Represent.*, vol. 20, no. 3, pp. 190–203, 2009.

[18] K. Xu, H. Zhang, D. Cohen-Or, and Y. Xiong, “Dynamic harmonic fields for surface processing,” *Comput. Graph.*, vol. 33, no. 3, pp. 391–398, 2009.

[19] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, “Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate,” *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 1–14, 2008.

[20] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, “Mesh editing with poisson-based gradient field manipulation,” *ACM Trans. Graph.*, vol. 23, pp. 644–651, August 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015774