Image Vectorization With Real-Time Thin-Plate Spline

Kuo-Wei Chen, Ying-Sheng Luo, Yu-Chi Lai, *Member, IEEE*, Yan-Lin Chen, Chih-Yuan Yao, *Member, IEEE*, Hung-Kuo Chu, *Member, IEEE*, and Tong-Yee Lee, *Senior Member, IEEE*

***Abstract*—The vector graphics with gradient mesh can be attributed to their compactness and scalability; however, they tend to fall short when it comes to real-time editing due to a lack of real-time rasterization and an efficient editing tool for image details. In this paper, we encode global manipulation geometries and local image details within a *hybrid vector structure,* using parametric patches and detailed features for localized and parallelized thin-plate spline interpolation in order to achieve good compressibility, interactive expressibility, and editability. The proposed system then automatically extracts an optimal set of detailed color features while considering the compression ratio of the image as well as reconstruction error and its characteristics applicable to the preservation of structural and irregular saliency of the image. The proposed real-time vector representation makes it possible to construct *an interactive editing system* for detail-maintained image magnification and color editing as well as material replacement in cross mapping, without maintaining spatial and temporal consistency while editing in a raster space. Experiments demonstrate that our representation method is superior to several state-of-the-art methods and as good as JPEG, while providing real-time editability and preserving structural and irregular saliency information.**

***Index Terms*—Real-time vector graphics, hybrid vector representation, scalability, real-time editability.**

Manuscript received June 9, 2018; revised March 14, 2019; accepted April 17, 2019. Date of publication June 11, 2019; date of current version December 31, 2019. This work was supported in part by the National Science Council of Taiwan under Grant 107-2221-E-011-115-MY2, Grant 107-2221-E-011-112-MY2, Grant 107-2221-E-011-114-MY2, Grant 106-2221-E-006-233-MY2, and Grant 107-2221-E-006-196-MY3, and in part by the Taiwan Building Technology Center from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education in Taiwan. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sanjeev Mehrotra. *(Kuo-Wei Chen and Ying-Sheng contributed equally to this work.) (Corresponding author: Chih-Yuan Yao.)*

K.-W. Chen, Y.-S. Luo, Y.-C. Lai, Y.-L. Chen, and C.-Y. Yao are with the Department of Computer Science and Information Engineering and Taiwan Building Technology Center, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: chen51202@gmail.com; tray307969@gmail.com; cheeryuchi@gmail.com; foodgoldsoldier@gmail.com; cyuan.yao@gmail.com).

H.-K. Chu is with the Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan (e-mail: hkchu@cs.nthu.edu.tw).

T.-Y. Lee is with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan (e-mail: tonylee@mail.ncku.edu.tw).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2019.2922126

I. Introduction

VECTOR graphics with gradient mesh is used in a variety of multimedia, thanks to its compactness and scalability, however complicated image details tends to spend huge rendering costs and manipulate less-than-intuitively for low-level attributes. In this paper, we propose a hybrid vector representation in which color details are encoded locally in parametric object patches to enable parallel kernel preparation and rasterization with the aim of faithfully preserving the structural and textural content while enabling high-level editing in real time. However, the traditional gradient mesh systems require an enormous number of curves to faithfully represent photorealistic images. Within this type of system, manual creation is highly non-intuitive and editing is difficult. Numerous researchers have sought to simplify and/or automate the construction of vector graphics, such as [5], [7], [13], [15], [17]–[19], [24], [25], [27], [32]; however, those methods require global solutions to partial differential equations (PDE), boundary element problems, and global illumination, which necessitate time-consuming precomputation and memory-hungry intermediate data structures. Most of the representations listed above lack editing tools, such as cross blending, texture transfer, and color manipulation. Aim to this issue, Yi [38] proposed an editing interface via reference images, but there is a lack of direct manipulation for gradient mesh. Thin-Plate Spline (TPS) interpolation provides interpolation that is "as-harmonic-as-possible". TPS has been used in geometric modeling [29] and computer vision [35] because it provides direct control over derivative interpolation and helps to maintain their smoothness, in particular smooth local minima/maxima.

In this paper, we propose a hybrid vector representation of parametric patches and detailed color features to enable precision editing and scalability. Parametric patches (see Fig. 1(a)) are used to represent object components to facilitate editing. Color details are encoded as features (see Fig. 1(b)) to achieve faithful rasterization using TPS interpolation based on the methods proposed by Powell *et al.* [26] without the need to link them into curves or obtain global solutions to PDEs. TPS interpolation provides direct control over derivative interpolation, while maintaining smoothness and obtaining smooth local minima/maxima. Nonetheless, the problem of determining color constraints and factorizing the color constrain matrix of TPS. We therefore localize these color constraints into independent patches to enable real-time parallel computation in the

1520-9210 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Fig. 1. This illustrates our vectorization of a raster image in (a) by encoding object segments as parametric patches marked in green curves in (a) and detailed features as color constraints in (b). Rasterization in (c) employs a biharmonic interpolation of detailed features for scalability and compactness, and it can also maintain editability with parametric patches to support various editing operations including color editing in (d) and cross mapping in (e).

construction of the TPS kernel, factorization, and rasterization. Furthermore, our patch-based scheme repeatedly uses features from neighbors and applies weighted averages across samples of multiple patches in order to counter the effects of aliasing and avoid blurring and ghost artifacts.

Real-time patch-wise TPS inversion and interpolation enables several vector image editing operations, including image magnification, color editing, and cross mapping (see Fig. 1(c), (d), and (e)). This is achieved without any of the difficulties associated with vectorization, while maintaining spatial and temporal consistency in the editing of images in a raster space. We conducted a numerical comparison of rasterization results obtained using two state-of-the-art vectorization algorithms [23], [38], [40], JPEG, and the proposed system. Our results achieved superior compressibility and scalability over the two vector representations and performed well against JPEG. In a comparison of magnification results using filter interpolation and super resolution, the proposed algorithm achieved numerical performance superior to those of JPEG and more pleasing results overall.

Our main **contributions** are as follows.

- We propose a **novel hybrid vector representation** of detailed color features embedded in parametric patches for localized GPU TPS rasterization to enhance compressibility and scalability, while enabling interactive editing in real time. This representation is effective for real-time image magnification, color editing, and material replacement in cross mapping, without compromising complex structural or textural details.
- We provide an **optimal feature selection scheme** using gradient intensity histogram of an image to balance the number of features and the reconstruction error based on our proposed compression efficiency metric.

The rest of the paper is organized as follows: Section II reviews those previous research done related to this work. Section III overviews our vectorization and rendering pipeline. Section IV gives the technical details of our vectorization process. Section V describes our GPU-based TPS interpolation scheme. Section VI details our experiments to determine the used parameters. Section VII discusses possible applications of image magnification, color editing, and cross mapping using our method. Section VIII shows the results of our algorithm and comparisons to other state-of-art methods. Section IX concludes with a discussion of limitations and future works.

II. Related Work

Mesh-based vectorization methods [4], [11], [22], [34] align meshes with edges using Delaunay triangulation followed by remeshing for encoding color information with linear interpolation inside each primitive for rendering. Liao *et al.* [23] and Zhou *et al.* [40] subdivided the originally rough meshes based on the saliency of an image, wherein the color of vertices is determined using an optimization process to achieve better rendering results. Bilinear interpolation inside primitives often results in a loss of sharpness. The highly dense, complex meshes required for detailed regions, such as the eyebrow in Fig. 3, reduce compressibility and make the process of editing non-intuitive and non-interactive. Furthermore, the time-consuming nature of remeshing makes it unsuitable for real-time applications. Thus, we developed a feature selection process for the optimized extraction of structural and textural details for TPS interpolation, thereby improving compressibility, scalability, and expressibility, as well as editing in real time.

Patch-based vectorization methods [21], [31], [37], [38] encode color and geometric information in parametric patches to facilitate editing and flexibility; however, they require a global optimization process for the selection of embedded information. Sun *et al.* [31] and Lai *et al.* [21] began with an initial mesh created manually by artists, whereas Xia *et al.* [37] aligned their initial triangular meshes with edges to avoid the need for manual construction, while maintaining $C0$ continuity only across patch boundaries. Unfortunately, obtaining highly-detailed images using these methods requires a high-density collection of patches, which greatly reduces compressibility and greatly increases the cost of pre-computing colors. These methods also make it difficult to conduct object-level manipulation in an intuitive or interactive manner. Thus, we align a parametric patch with each object to enable object-based editing, while embedding detailed features in patches to enable scalability, compressibility, and high rasterization quality. Furthermore, the application of TPS interpolation in local patches maintains at least $C1$ continuity in all surface patches, and avoids the need for global optimization in preprocessing, thereby allowing editing in real time.

Curve-based vectorization methods [5], [7], [13], [15], [17]–[19], [24], [25], [27], [32] use curves and lines as color constraints to ensure smooth coloring and rasterization. Xie *et al.* [39] was able to reduce the number of curves required to achieve realistic results; however, their method still requires a

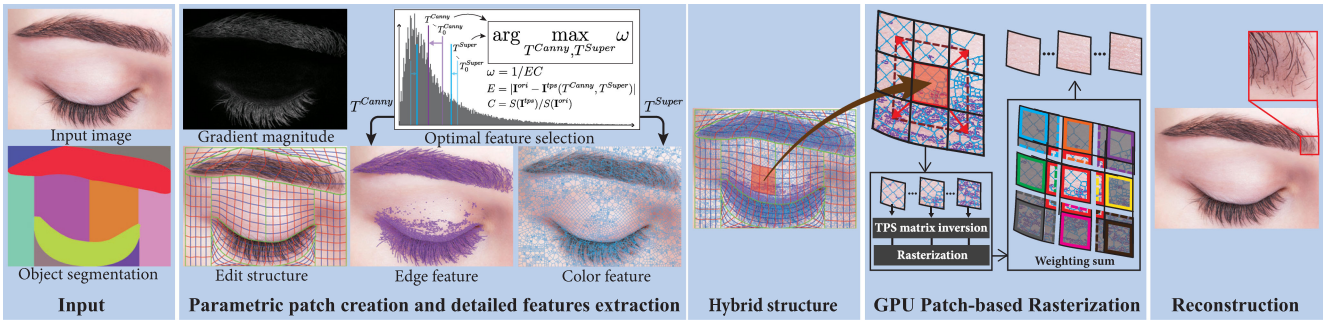

Fig. 2. Input raster image and its corresponding labeling map of object segments. Our *vectorization pipeline* involves parametric patch construction, optimal color feature extraction, patch-based feature grouping, TPS kernel construction, and rasterization using GPU-based TPS color interpolation. The red rectangle presents the eyebrow under 4X magnification.

large number of curves to render highly-detailed objects, which greatly increases memory usage. Determining the color constraints along the curve generally requires a complex filtering mechanism [39] or global optimization [19] which are in turn accelerated using multigrid PDE solvers [13], [17], [18], [24], boundary element methods [7], [15], [25], [32], [33], or ray tracing mechanisms [6], [27]. Complexity and memory usage increases with the detail of intermediate structures. Furthermore, mesh-free curve manipulation necessitates the reconstruction of intermediate structures, greatly hindering object-based editing. Conversely, our work uses color samples derived using a simple selection mechanism to enable parallel TPS interpolation in local patches which do not require global PDE solutions and decouples the complexity of the intermediate data structure to facilitate compression and scaling, while increasing the efficiency of rasterization. Furthermore, embedding within parametric patches provides a global manipulation mechanism for interesting editing applications.

III. Overview

Fig. 2 presents our vectorization and rendering pipeline. Users first provide a raster image and labeling map of interesting object segments. The system locates the four corners of the segments, computes the corresponding derivatives at these corners, and constructs corresponding Hermite patches. Image characteristics are analyzed based on its gradient distribution histogram to select an initial set of detailed color features using adaptive super-pixel and Canny operators. Feature selection is optimally refined using Monte Carlo searching [30] with our proposed compression efficiency heuristic aimed at balancing reconstruction error against the compression ratio. Our system embeds extracted features into Hermite patches, clusters them into localized groups for evaluation, packs these groups with neighboring features to construct TPS rasterization kernels of equal size, and applies TPS interpolation to compute the color of pixels in the group. Finally, the rasterization regions are extended to provide suitable overlap. A weighted average is applied to the overlapping groups to remove seams, i.e., improve continuity. In applying magnification, color editing, and texture transfers, the proposed system adjusts the color and location of features and then repeats the TPS kernel construction, inversion and rasterization process in order to generate results with minimal distortion while enabling editing in real time.

IV. Vectorization Using Hybrid Structure

The proposed algorithm maintains scalability and editability by vectorizing a photorealistic image and its corresponding labeling map to create a hybrid representation comprising parametric patches and detailed color features. Our vectorization process involves parametric patch construction, detail feature extraction, TPS inversion, and rasterization. The details of each process are described in the following sections.

A. Mathematical Definition and Embedment

Parametric patches enable the intuitive editing of 2D images; therefore, we opted to parameterize object segments as cubic Hermite patches in order to facilitate editability. This work defines the position vector of a Hermite patch $\mathscr{M}(s,t)$ where (s,t) are control parameters with $0 \leq s \leq M-3$ and $0 \leq t \leq N-3$, (M,N) are the number of control points specified by the users or computed automatically by our system, and each control point $\mathbf{M}_{i,j}$ has four control vectors for a given position, with derivatives along the s, t, and st directions denoted as $\overline{M}^{p}_{i,j}$, $\overline{M}^{s}_{i,j}$, $\overline{M}^{t}_{i,j}$, and $\overline{M}^{st}_{i,j}$. Given (s,t), the proposed system first locates its corresponding sub-patch index, $(\lceil s \rceil, \lceil t \rceil)$ and in-patch parameter, $(\tilde{s},\tilde{t}) = (s - \lceil s \rceil, t - \lceil t \rceil,)$. This enables us to compute the location in each sub-patch as $\mathscr{M}(\tilde{s},\tilde{t}) = \mathscr{B}(\tilde{s})\mathbf{Q}\mathscr{B}(\tilde{t})$, where $\mathscr{B}(u)$ represents the set of the basis functions as $\{2u^3 - 3u^2 + 1, -2u^3 + 3u^2, u^3 - 2u^2 + u,$ and $u^3 - u^2\}$ where $\mathbf{Q}$ refers to the four control vectors of the four control points for the sub-patch. It is easy to transform a parametric space to a pixel space; however, no analytic solution has been devised to allow transformations from a pixel space to a parametric space. We use a 2D binary search process to look for a good approximation of (s,t), when given (x,y) as follows: The proposed system first uses the locations of control points to locate the sub-patch containing the pixel. We compute the pixel location of the middle parametric point, divide the sub-patch into four quadrants and then identify the quadrant in which (x,y) lies. The above two steps are repeated until the error associated with the constructed surface is smaller than a user-selected threshold, T_{loc}, or a selected number of iterations,

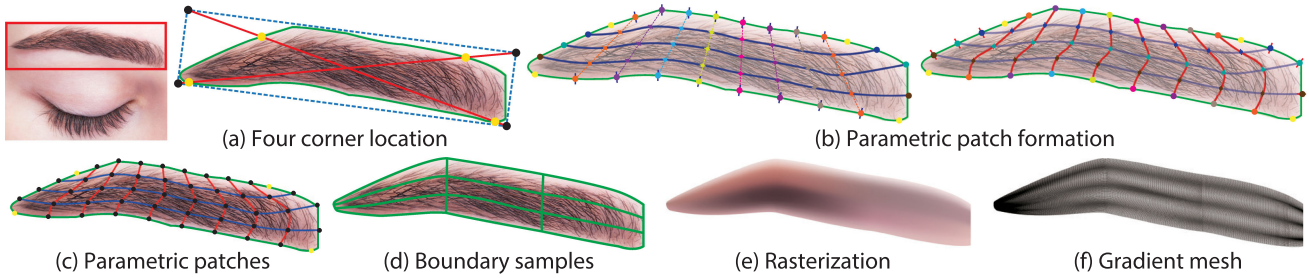

Fig. 3. A parametric patch is constructed by identifying four corners: (a) computing the location and derivatives of $M \times N$ control points; (b) create the Hermite patch $\mathscr{M}(s,t)$; (c) sample detailed features at patch boundaries; (d) and rasterize the results. (e) We also constructed a faithful gradient mesh of very dense primitives (f).

N_{itr}. All of the results in this work are based on the following: $T_{loc} = 0.001$ and $N_{itr} = 8$. The transformation process can also be accelerated using a look-up table, albeit with a slight sacrifice of memory resources.

B. Editable Parametric Patch Construction

The user identifies the desired object segments in the form of a labeling map created semi-automatically using multi-label graph-cut segmentation [10] with a few indicative multi-label strokes. In other words, while the object has multiple connected regions, the user needs to segment it properly into multiple interconnected simple regions in order to generate reasonable controlled patches. The system then generates the corresponding parametric patches, as shown in Fig. 3. The arbitrary minimum bounding box is first computed using the four intersections of the diagonal axes and patch boundaries as four corners. The intersections are used as end points to separate boundary points into 4 sets of samples by which to fit 4 cubic Hermite boundaries. A pair of boundary curves is selected from the opposite sides with the longest lengths. For each curve, we select M points with equal parameter spacing and link the corresponding two points from both of the sides to form M lines. The system then selects N points with an equal Euler spacing along each line, which are denoted as $\overline{P}_{i,j}$, where i is the index of the line and j is the point index along the line. The system constructs M Hermite curves to fit the set of samples of $\overline{P}_{i,0} \ldots \overline{P}_{i,N-1}$. Similarly, we sample N points (with an equal parameter spacing) for each newly constructed Hermite curve to be denoted as $\overline{Q}_{i,j}$, where i is the index of the curve and j is the point index along the curve. The system constructs N Hermite curves to fit the set of samples of $\overline{Q}_{i,0} \ldots \overline{Q}_{i,M-1}$. Finally, we find the intersection of all constructed Hermite curves to derive $M \times N$ control points, $\overline{M}^p_{i,j}$, of the object patch, $\mathscr{M}(s,t)$. The derivatives in the s and t directions are selected based on the derivative of the two intersected Hermite curves, and the derivative in the st direction is calculated using the Adini twist vector formulation [2] as $\overline{M}^{st}_{i,j} = 0.5(\overline{M}^s_{i+1,j} - \overline{M}^s_{i-1,j}) + 0.5(\overline{M}^t_{i,j+1} - \overline{M}^t_{i,j-1}) + 0.25(\overline{M}_{i+1,j+1} - \overline{M}_{i+1,j-1} + \overline{M}_{i-1,j-1} - \overline{M}_{i-1,j+1})$. The above process works well with convex patches, however, the intersections of non-convex patches, such as the 'C' shape, are unsuitable for manipulation. Thus, all of the border points are assembled into groups by selecting connected points using the Ramer-Douglas-Peucker (RDP) algorithm. The groups are fit into poly-lines and linked as a polygon. If the polygon is convex, then we use the above algorithm to find the four corners. Otherwise, we decompose the polygon into a set of connected convex polygons using optimal convex decomposition [8], find the four corners of each polygon, and remove the corner pairs sharing the same position. The remaining points are our four corners. This process is repeated for all object patches of interest.

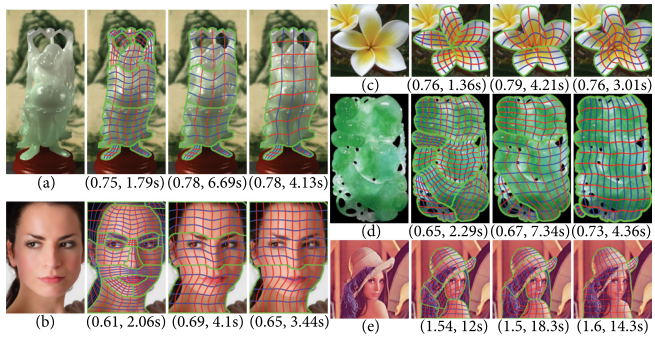

Fig. 4. Parametric patches created using various labeling maps: (a) to (e) Buddha, Face, Flower, Green jade, and Lena. The first column is the original raster image, and second to fourth columns are parametric structures created using different labeling maps along with their corresponding reconstruction mean square error and GPU-based TPS run times.

Different labelling maps result in different parametric structures, as shown in Fig. 4. The fact that the proposed color reconstruction scheme and localized patch-overlapping seam removal scheme depend only on detailed features (Sections IV-C and VI-C) means that different label maps have little impact on reconstruction error. However, the size of the patch can have a tremendous impact on the number of features that are included, which in turn has a dramatic impact on TPS reconstruction efficiency, as discussed in Section V-B.

C. Selection of Detailed Color Feature

Bilinear interpolation of sparse color samples along patch boundaries can result in serious loss of image detail, as shown in Fig. 3(d) [4]. TPS interpolation can improve the results [26]; however, sparse features still have serious issues in pre-serving

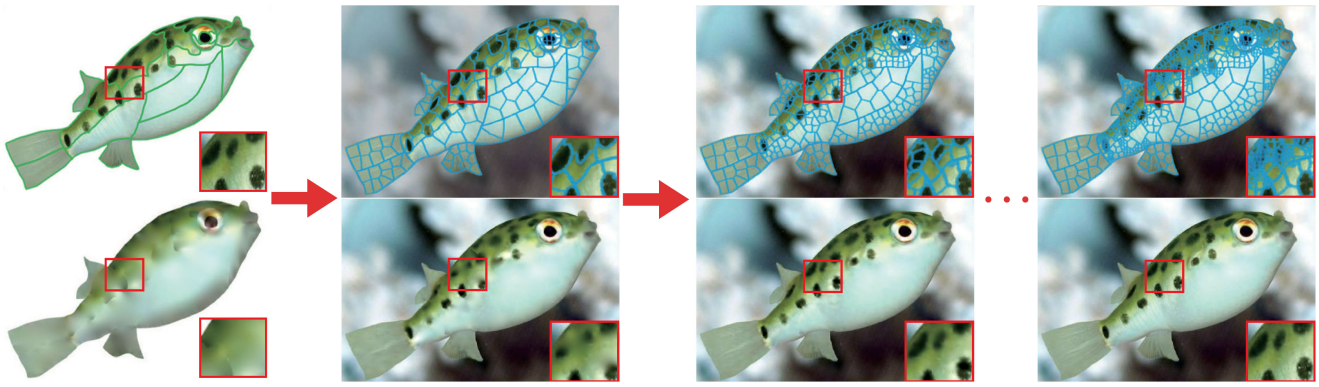

Fig. 5. Feature extraction process of proposed adaptive super pixel operator: (left to right) input, initialization, result after one adaption, and final result; (top and bottom) super pixel results and their corresponding rasterization in different stages.

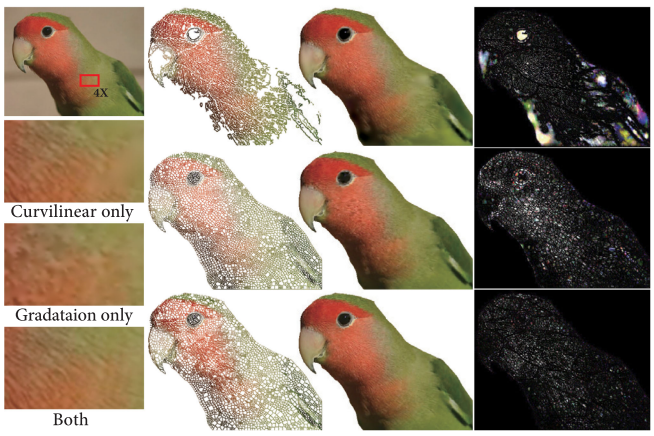

Fig. 6. Rasterization results using various types of feature but with the same number of features. The first column shows a parrot under 4-time magnification, as follows: TPS rasterization results with curvilinear features, gradation features, and both curvilinear and gradation features (from top to bottom, respectively). The second to the fourth columns show the selected features, TPS rasterization results, and error analysis with curvilinear features, gradation features, and both curvilinear and gradation features (from top to bottom, respectively).

details, as shown in Fig. 3(e). In this work, we encode fine detail as color features, $\mathscr{F}$, to overcome the problem of information sparsity. Along with scattering data interpolation of color samples, our approach to representation can preserve the compressibility of diffusion curves without the need to solve global PDEs nor the additional memory overhead imposed by intermediate structures. For the sake of clarity, we define our features based on the concept of gradient and value constraints proposed by Boyé *et al.* [7]. First, contours (gradient constraints) are features in detailed regions presenting a gradient discontinuity, i.e., the two sides present two distinct colors. In this work, contours are denoted as curvilinear features, $\mathscr{F}^C$. Second, gradations (value constraints) are features in homogeneous regions presenting a smooth color gradient, i.e., the two sides have the same color. In this work, gradations are denoted as gradation features, $\mathscr{F}^T$. In the following, we detail the processes of feature extraction and embedment.

1) Curvilinear Feature Detection: Elder *et al.* [12] reported that edges, (i.e., curvilinear features) delineate silhouettes and occlude contours, making them important cues in the capture and interpretation of scenes. Thus, these features are captured by identifying potential locations that present obvious differences in color between the two sides of a pixel. This is achieved using a Canny edge detector with threshold, T^{Canny}. To describe this distinction, we place a pair of features that present two distinct colors, $\mathbf{F}_i^C = \{\overline{p}_i, \overline{C}_i\}$, across edge points ($\overline{e}_i$) located at $\overline{e}_i \pm 0.5\vec{v}_i$ where $\vec{v}_i$ is the direction perpendicular to the pixel gradient $\vec{g}_i$, and colors are the colors of the two closest pixels. We record our features $\mathbf{F}_i^C = \{\overline{p}_i, \overline{C}_i\}$ in the patch parametric coordinate using a 2D binary search process and Lab space, respectively. The first column in Fig. 6 shows an example using these curvilinear features. There are obvious artifacts in the smooth gradient regions due to the sparse curvilinear features. We can overcome the sparsity issue by setting a very low T^{Canny} value, which increases the number of features as well as the computation and memory costs. Our scheme adds gradation features, rather than setting a small T^{Canny} to overcome the issue of sparsity, as described in the following section. In Section IV-C3, we also present a scheme by which to select an optimal set of curvilinear and gradation features based on a proposed heuristic metric.

2) Gradation Feature Identification: We overcome the issue of curvilinear sparsity associated with a large T^{canny} by adding low-frequency gradation features based on adaptive super-pixel extraction. Achanta *et al.* [1] clustered a cell of Ncell pixels that present a similar color distribution in a local area where $N^{cell} = \lfloor N^{Pixel}/4000 \rfloor$ in all our examples and N^{pixel} is the number of pixels. Although each cell is similar in size, color variation may be too great to break the smooth maintenance ability of TPS interpolation for artifact-free results. Subsequent super-pixel decomposition is enforced until the variance in cell color is below a selected threshold, T^{Super} as follows. The proposed system first estimates the color covariance of all pixels, $\sigma_{S,C}$, in a super-pixel (S, C), in the Lab color space where S is the iteration index, and C is the cell index. When $\sigma_{S,C}$ is larger than T^{Super}, we set the target number of super-pixels, as follows:

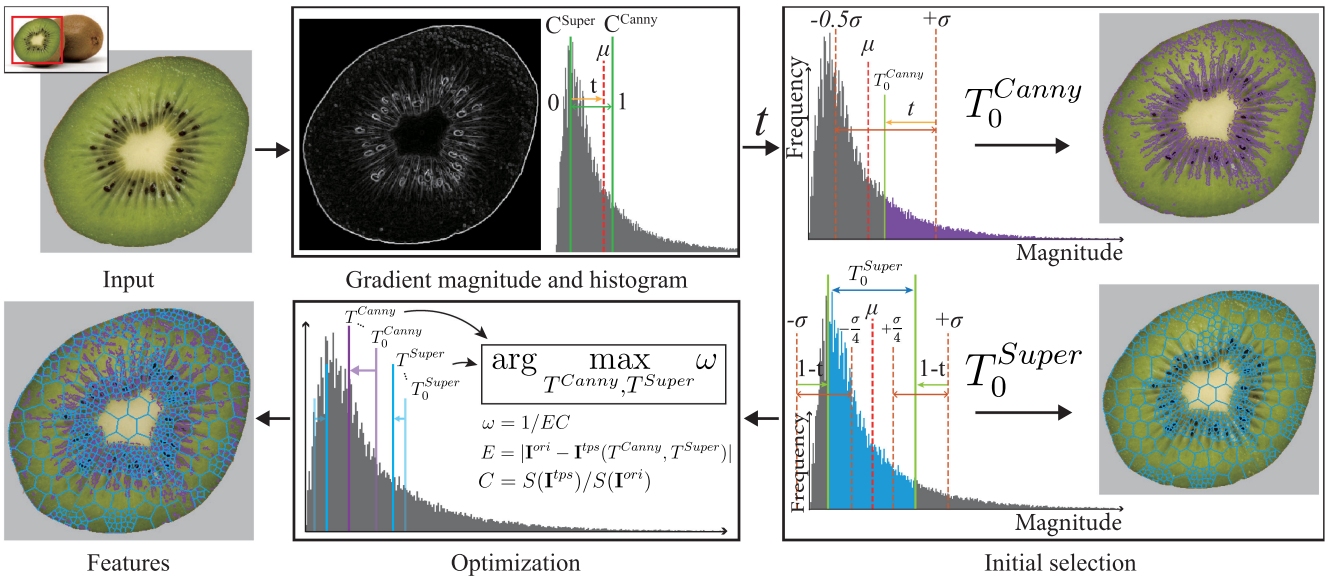

Fig. 7. Selection process of optimal features involving computation of pixel intensity gradients and their corresponding gradient histograms, selecting an initial set of parameters, $(T_0^{Canny}, T_0^{Super})$, and determining the final values of (T^{Canny}, T^{Super}) with Monte Carlo searching [30].

$N_{S+1,C}$, as $\lfloor W^p \frac{\sigma_{S,C}}{\sum_{c\in\mathscr{S}} \sigma_{S,C}} \rfloor + \lfloor W^i \frac{1}{\sigma_{S,C}^{0.5}} \rfloor$ where $\mathscr{S}$ is the cell set at this level, W^p refers to the weight for variance in the parent cell, and W^i is the weight for cell variance in subsequent super-pixel decomposition inside the (S, C)-th cell.

This process repeats until convergence or until the desired level is reached. Generally, $N_{S,C}$,C affects only the convergence speed; i.e., more gradation features leads to more iterations. Based on experiment results, we set W^p at 4, and W^i at $\frac{N_{S,C}}{40}$ to achieve the optimal balance. After constructing 40 super-pixel cells of different levels, we added all cell boundary pixels as gradation features $\mathscr{F}_i^T$ using their corresponding patch-based locations and pixel colors. Fig. 5 presents an example of iteratively selecting gradation details of different levels and their corresponding rasterization results. Our adaptive super pixel algorithm iteratively adjusts the block size to encode appropriate coloring details as constraints for the TPS operation. In other words, this process retains more of the detail as the size of the blocks becomes increasingly fine. Similarly, the second column of Fig. 6 shows that the gradation features encode low-frequency details of similar color. However, using them only blurs the high-frequency details in regions with sharp transitions. We compensate for this using curvilinear features to maintain scalability during magnification, as shown in the third column of Fig. 6.

3) Optimal Feature Selection: Generally, a larger number of features results in lower reconstruction error, and vice versa. Therefore, we designed a criterion to determine the quality of the parameter selection in order to balance reconstruction error E versus the coverage ratio C (i.e., the compress rate). The criterion is referred to as efficiency ($\omega = 1/EC$). We can also express the selection as follows: where $\Omega()$ represents the feature selection and reconstruction process based on two parameters, T^{Canny} and T^{Super}. Finally, parameter selection is formulated as an optimization problem.

$$\arg \max_{T^{Canny},T^{Super}} \Omega(T^{Canny}, T^{Super}) \tag{1}$$

We do not employ a simple brute force search process for an optimal (T^{Canny}, T^{Super}) set, due to the time-consuming nature of global TPS process, as shown in Table II. Rather, we solved the optimization process of parameter selection with Monte Carlo searching [30] by selecting an initial value of T_0^{Canny} and T_0^{Super} using ω to evaluate its performance, keeping the best till now, and mutating both $T_{i+1}^{Canny} = T_i^{Canny} + \Delta T^{Canny}$ and $T_{i+1}^{Super} = T_i^{Super} + \Delta T^{Super}$ for the next iteration where ΔT^{Canny} and ΔT^{Super} are randomly selected in the range of $\pm|T^{Canny}|$ and $\pm|T^{Super}|$. The process stops when optimal solution has been found or the number of iterations exceeds N_{total}. In this study, we used the following settings: $N_{total} = 20$. Later, in Section VI-A, we have conducted an experiment to determine these parameters for our study.

D. Image Reconstruction With a Composite Structure

In order to unify parametric patches, $\mathscr{M}$, and detailed features, $\mathscr{F}$, for easy and consistent TPS operations, our system normalizes the sub-patch parametric coordinate (s, t) to uniform as (S, T). TPS interpolation creates three as-harmonic-as-possible functions, $r(S,T), g(S,T), b(S,T)$, for three color channels based on the given set of N features, $\{\ldots, (S_i, T_i, \overline{C}_i), \ldots\}$. Since our system applies the same operation to three channels independently, the following uses f to denote r, g, and b. The solution must minimize the bending energy described as

$$I(f) = \int\int_{\Omega} f_{SS}^2 + 2f_{ST}^2 + f_{TT}^2 dSdT \tag{2}$$

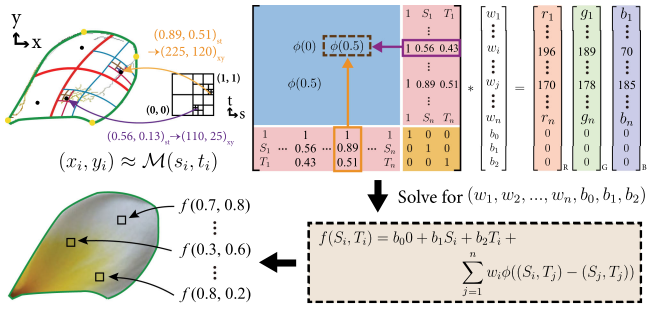

Fig. 8. Our TPS matrix construction consists of transforming the feature locations from pixel coordinate to parametric coordinate and filling $\mathbf{K}$, $\mathbf{P}$, and $\mathbf{W}$ with features' inter-relationship. After inversion, our system rasterizesthe i-th pixel by transforming to parametric coordinate and estimating its color using $f(S_i, T_i)$ listed in Eq. (3).

and f must fulfill

$$f(S,T) = b_0 + b_1 S + b_2 T + \sum_{i=1}^{N} w_i \phi(\| (S_i, T_i) - (S, T) \|) \tag{3}$$

where $\sum_{i=1}^{N} w_i = 0, \sum_{i=1}^{N} w_i S_i = 0$ and $\sum_{i=1}^{N} w_i T_i = 0$. This enables the formation of a linear system comprising all of the features $\mathscr{F}$, to determine the TPS coefficients, w_i, as

$$\begin{bmatrix} \mathbf{K} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{O} \end{bmatrix} \begin{bmatrix} \overline{W} & b_0 & b_1 & b_2 \end{bmatrix}^T = \begin{bmatrix} \overline{H} & 0 & 0 & 0 \end{bmatrix}^T \tag{4}$$

where $K_{ij} = \phi(\|(u_i, v_i) - (u_j, v_j)\|)$, the i-th row of P is $\{1, S_i, T_i\}$, $\mathbf{O}$ is a 3×3 zero matrix, $\overline{W} = \{w_1, \ldots, w_N\}$, and $\overline{H} = \{\overline{C}_{1j}, \ldots, \overline{C}_{Nj}\}$. After solving Eq. (4), the proposed system respectively utilizes $\overline{W}$, b_0, b_1, and b_2 for three color channels in estimating the color of all pixels within the patch using $r(S,T)$, $g(S,T)$, and $b(S,T)$. Fig. 8 summarizes the construction, inversion and interpolation process of the TPS kernel, and Fig. 20 illustrates the ability of the algorithm to recover the finest detail.

V. GPU-Based TPS Interpolation

Our system uses all features to faithfully rasterize a photorealistic patch using TPS interpolation. This results in an enormous, dense feature matrix with an inversion cost of $O(N^3)$, where N refers to the total number of features. As shown in Table I, inversion generally takes between minutes and hours to complete; i.e., real-time global TPS interpolation is impossible. Many-core GPUs are able to accelerate the process of global matrix inversion; however, this reduces the process to the level of several seconds, which is insufficient for real-time operations. In this section, we propose a GPU-based TPS interpolation scheme using seamless localized patches to resolve this problem.

A. Patch-Based Parallel Structure

Local divide-and-conquer curve-fitting provides better editability and lower computational costs, when features are densely packed or when a feature has only localized affection; i.e., when the distance between a feature and the estimated pixel is large, its affection is negligible. Based on the result of a simple experiment described in Section VI-A, our localized patch-wise TPS interpolation scheme directly uses the sub-patches created during the structure construction process for the clustering of all extracted features in order to decompose the inversion of a complete characteristic matrix into a set of localized characteristic matrices for interpolation in a patch-wise manner. This increases the number of inversions; however, the size of the matrices associated with each inversion actually shrinks. This means that computational costs are reduced, due to the fact that the cost grows exponentially with the size of the matrix but only linearly with the number of inversions. Nonetheless, two issues remain. First, non-overlapping sub-patches produce seams along their boundaries, due to differences in feature affections across sub-patches, as shown in Fig. 9. As shown in Section VI-C, we employ weighted averaging to remove the seams from overlapping sub-patches. Second, sub-patches vary in the number of features to have unequal-size GPU kernels for extra kernel synchronization time. We used the local distribution of features to take full advantage of the parallel computing power of the GPU for computation, as detailed in the next section.

TABLE I
Timing Statistics of Entire Matrix Construction, Inversion, and Rasterization Process (Measured as Seconds Per Patch). CPU Refers to Global CPU Scheme, GPU Denotes the Global GPU Scheme, None Refers to the Localized GPU Scheme Without Packing, and Packing Denotes the Localized GPU Scheme With Packing. All Measurements Were Obtained Using a Intel i7-4930K 3.40 GHz Computer With 64 GB of RAM, and an Nvidia GeForce GTX 1080 With 16 GB of Video RAM

	CPU	GPU	None	Packing
Lena	1.87e3	14.1	0.19	0.06
Peppers	4.55e3	14.2	0.26	0.07
Parrot	628	2.86	0.05	0.02
Duck	2.75e3	14.1	0.08	0.02
Butterfly	1.23e4	21.1	0.12	0.03
Butterfly2	3.37e3	23.7	0.22	0.02

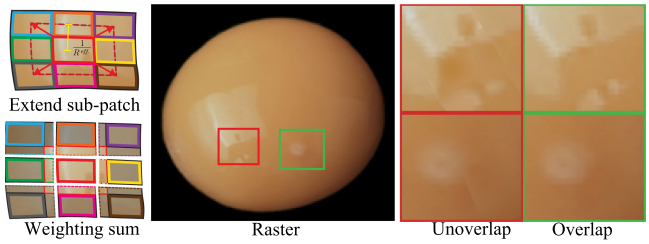

Fig. 9. Seams formed along boundaries after clustering features based on sub-patch topology. Our system overcomes this issue through formation of a set of overlapping sub-patches while blending rasterization results.

B. Maximize GPU Parallelization

We developed two schemes to enable the full utilization of all GPU cores in parallel. Our first objective was to reduce the amount of data transferred between the CPU and the GPU (a typical computational bottleneck). During manipulation, our unified parametric space allows the direct transfer of manipulated

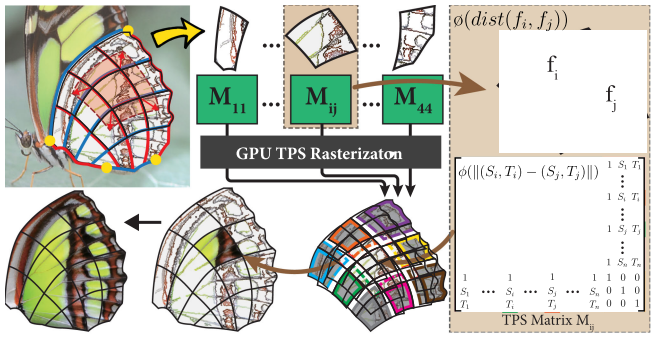

Fig. 10. Our localized patch TPS scheme consists of patch decomposition, equal-number feature patch extension, TPS kernel formation, TPS kernel inversion, rasterization, and overlapping interpolation.

control points to the GPU, whereupon the parametric information of the features is updated for TPS kernel construction in the GPU, instead of updating and constructing kernels in the CPU. This effectively reduces the amount of data that must be transmitted. Second, the cost of a TPS kernel can be expressed in terms of the number of rasterized pixels and features, which means that it should be possible to minimize GPU synchronization time by keeping both terms as similar as possible. As outlined in the previous section, M and N were assigned uniform parametric spacing, thereby ensuring that the size of the respective sub-patches is similar. It is also important to ensure that the matrices have the same size. In other words, we can ensure that each sub-patch has the same number of features by finding the sub-patch with the most features and extending feature selection outward to include all neighboring features until the largest number is reached. In other words, our system adds extra features from neighboring sub-patches according to the parametric distance to the center of the patch. In most cases, the added features have only a negligible effect on rasterization, and subsequent packing can mediate the computation costs, as shown in Table I. However, in extreme cases, when the number of features varies greatly among the patches, packing requires that an enormous number of features be located in sparse patches, for which the computation cost is not negligible.

Fig. 10 summarizes our localized patch-based TPS rasterization procedure. The localized TPS framework first decomposes patches into sub-patches according to their feature densities and physical size. The feature selection region of each sub-patch is extended so that each sub-patch has the same number of features for each TPS inversion kernel, thereby ensuring the same computation cost and maximizing the benefits of parallel computation by the GPU. All sub-patch features are used to construct the TPS matrix and compute its inversion for sub-patch rasterization. Finally, the final result is stitched together using our inter-patch interpolation scheme.

VI. Parametric Ablation Study

In order to cover the entire possible feature selection range and reduce the number of feature selection iterations, we have conducted a simple experiment to choose a proper initial set of parameters and determine the perturbation range. Additionally, we conduct experiments to justify adaptation of the global kernel as a local one and the overlapping ratio to remove the seam.

A. Optimal Feature Selection

The optimization process presented in Section V-A may require a large number of trials along with the possibility of finding a local maxima if the initial start-up parameters are not close to the global maximum. By observing the brute-force solutions, we were able to empirically determine the parameters for regions of high and low variance, based on the density of curvilinear features. On the histogram for a region of high variance, the optimal brute-force T^{Canny} value falls at approximately $\mu - 0.5\sigma$. Similarly, a threshold for regions of low variance generally falls at $\mu + \sigma$. When analyzing cases that fall between these two extremes, the optimal threshold moves toward $\mu - 0.5\sigma$ when the patch contains greater variation; otherwise, it moves toward $\mu + \sigma$. We applied the same procedure to determine the superpixel threshold, T^{Super}. The thresholds for regions of high and low variance are 0.8σ and $\mu + 0.25\sigma$, respectively. The optimal threshold of a patch with greater variance is close to 0.8σ; otherwise, it is close to $\mu + 0.25\sigma$. The two thresholds are linear with the curvilinear selection ratio, $t = \frac{\mu - C^{Supper}}{C^{Canny} - C^{Super}} \in [0, 1]$, where C^{Canny} and C^{Super} are two user-selected constants with values of 30 and 60, respectively. We use this ratio in the linear interpolation of two extreme thresholds for two thresholds as $T_0^{Canny} = t(\mu - 0.5\sigma) + (1 - t)(\mu + \sigma)$ and $T_0^{Super} = t(0.8\sigma) + (1 - t)0.25\sigma$, respectively.

Table II shows that our empirical approach and optimization scheme both provide results close to the brute-force solution. This table also shows that the compression efficiency of our methods is comparable to the state-of-the-art image format, JPEG, the optimal efficiency of which is determined by brute-force searching through various loss rates.

B. Localized Reconstruction

In order to determine the possible range of localized reconstruction while still introducing negligible errors, we have conducted an experiment as follows. We first computed the TPS ground truth of a patch using all of the patch features described in Section IV-D. Inside the patch, we selected a sub-patch by randomly assigning a center within the patch with a designed width of W^{sub}, creating a selection square (the generally preferred shape for manipulation and computation), and using all of the features that fall within the square for TPS rasterization. We then compared the error obtained in this exercise against the ground truth using various values for W^{sub}. We also analyzed the effective region by extending the sub-patch outward until the central PSNR exceeded a user-specified threshold ($T^{effective}$), set at 50. At the desired $T^{effective}$ level, our experiment results revealed that the $R^{sub} = N^{sub}/N^{patch}$ patch that the effective sub-patch ratio, $R^{sub} = N^{sub}/N^{patch}$, and collection ratio, $R^{collect} = N^{collect}/N^{patch}$, are relative to the R^{sub} feature density, $D = N^{features}/N^{patch}$ as $R^{sub} = 5D/(2.0 - D)$

TABLE II
STATISTICAL COMPARISON OF PROPOSED PARAMETER SELECTION SCHEME AND JPEG. INITIAL DENOTES THE RESULTS OF OUR INITIAL PARAMETER SELECTION; OPTIMAL DENOTES THE RESULTS OF OUR OPTIMAL PARAMETER ESTIMATION; BRUTE-FORCE DENOTES THE RESULTS OF THE BRUTE-FORCE PARAMETER SEARCH FOR OUR ALGORITHM, JPEG DENOTES THE RESULTS OF THE BRUTE-FORCE SEARCH FOR JPEG AT VARIOUS LOSS RATES FOR OPTIMAL COMPRESSION EFFICIENCY, PSNR DENOTES THE PEAK SIGNAL-TO-NOISE RATIO, ω DENOTES OUR DEFINED EFFICIENCY VALUE AND BPP DENOTES THE BIT-PER-PIXEL RATIO

	Initial			Optimal			Brute-force			JPEG		
	PSNR	ω	*BPP*	*PSNR*	ω	*BPP*	*PSNR*	ω	*BPP*	*PSNR*	ω	*BPP*
Lena	35.4	4.75e4	1.49	40.8	7.09e4	2.28	39.4	8.45e4	2.46	34.4	6.18e4	1.06
Peppers	52.5	4.40e5	0.97	50.7	4.66e5	0.66	47.7	4.91e5	1.44	39.9	4.67e5	0.5
Parrot	39.9	1.27e5	1.51	42.4	1.46e5	2.03	42.5	1.46e5	1.92	36.6	1.39e5	0.79
Duck	46.4	7.15e4	1.76	35.3	9.10e4	2.24	35.3	9.17e4	2.26	34.6	5.92e4	1.21
Butterfly	35.2	7.09e4	1.97	36.8	8.78e4	2.22	37.3	9.49e4	2.31	33.5	2.80e4	1.89
Butterfly 2	29.5	4.23e4	1.69	30.8	4.94e4	1.81	31.9	5.66e4	1.89	32.5	5.07e4	0.70
Monkey	38.7	9.23e4	1.04	39.9	1.02e5	1.12	42.6	1.29e5	1.26	36.1	1.54e5	0.33

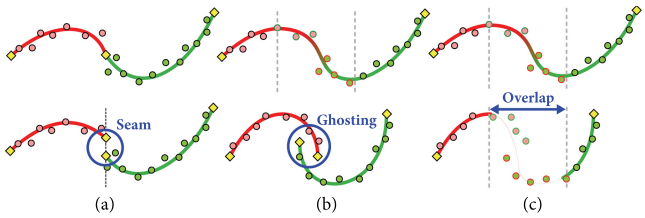

Fig. 11. We simplify TPS interpolation with simplified 2D spline fitting and interpolation using independent fitting and manipulation (a), correlation fitting and independent manipulation (b), and correlation fitting and manipulation (c) where the dots with solid red color are features for the left segment and the dots with green ring are for the right.

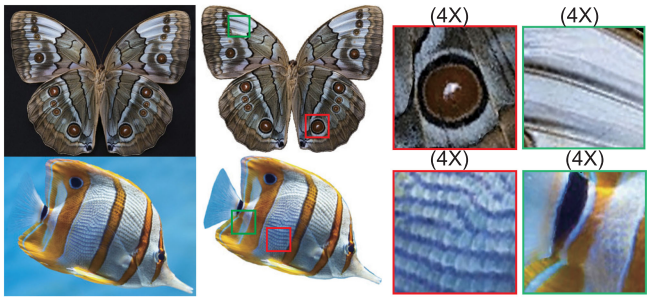

Fig. 12. The top is a butterfly of monotonic regions and complex patterns and the bottom is fish of complex fish scales. From left to right are raster images, TPS rasterization results, and 4-time magnification results.

and $R^{collect} = 5D$, where N^{sub} is the desired number of pixels in a sub-patch, N^{patch} is the number of pixel in the patch, $N^{collect}$ is the number of pixels in the extended collection region, and N features is the number of pixels in the pixel. We can estimate M and N for the patch as $M = (\frac{R^{sub}W^{patch}}{H^{patch}})^{0.5}$ and $N = (\frac{R^{patch}H^{sub}}{W^{patch}})^{0.5}$ where W^{patch} and H^{patch} are the lengths of the corresponding patch boundaries. Accordingly, our localized patch-wise TPS interpolation scheme directly uses the sub-patches created during the structure construction process for the clustering of all extracted features in order to decompose the inversion of a complete characteristic matrix into a set of localized characteristic matrices for interpolation in a patch-wise manner.

C. Patch-Based Boundary Overlapping Seam Removal

We extend the TPS kernel extension outward to include features from neighboring sub-patches and thereby minimize reconstruction error. This would cause some of the sub-patches to overlap; however, it should be possible to remove seams by weighting the sub-patches to color a pixel as follows: $\overline{C}_i = \sum_{o\in\mathscr{O}} w_o^{blend}\overline{C}_o(\overline{p}_i)$ where $\mathscr{O}$ is the set of sub-patches covering the i-th pixel, $\overline{C}_o$ denotes the TPS rasterization of the o-th sub-patches, and w^{blend} is the blending weight proportional to the distance to the sub-patch center, $\overline{p}_o^{center}$. We define the blending weight as $w_m^{blend} = \frac{|\overline{p}_i - \overline{p}_m^{center}|}{\sum_{o\in\mathscr{O}}|\overline{p}_i - \overline{p}_o^{center}|}$. Fig. 9 presents an example of the proposed overlapping scheme. Our algorithm repeatedly uses features from neighboring units to achieve high-order continuity and avoid the formation of seams, blurring, and ghost artifacts, as shown in Fig. 11. When we separated the features into two groups and fit them independently using only one link point (similar to having two independent textured meshes), any subsequent manipulation could result in the formation of a seam along the boundary as shown in Fig. 11(a). When we repeatedly used features from neighboring units to obtain two segments (similar to two overlapping textured meshes), independent manipulation could cause the results to deviate, resulting in ghosting and/or blurring, as shown in Fig. 11(b). When manipulating the features and then refitting them to obtain two segments, continuity remains (i.e., no ghosting or blurring), as shown in Fig. 11(c). This is a demonstration that the repetition of these features prevents ghosting and blurring artifacts in overlapping regions. Furthermore, seam removal averages the difference across sub-patch boundaries to provide inter-sub-patch anti-aliasing, whereas TPS interpolation provides antialiasing as an inherent feature inside a sub-patch.

VII. VECTOR GRAPHICS MANIPULATION

Image editing is important in a wide range of multimedia and graphics applications, such as movie post-production. It is possible to perform color editing and cross mapping in a raster space and then vectorize the results; however, it requires additional time and manual parameter adjustment for vectorization. It also tends to induce inconsistencies between the raster and vector results, due to vector information estimation. The proposed hybrid vector representation uses efficient patch-wise TPS-based inversion and interpolation, which is ideally suited to editing in real time. It provides the flexibility required for image magnification, color editing, and cross mapping with low reconstruction error in an intuitive manner. The ability to edit images directly in the

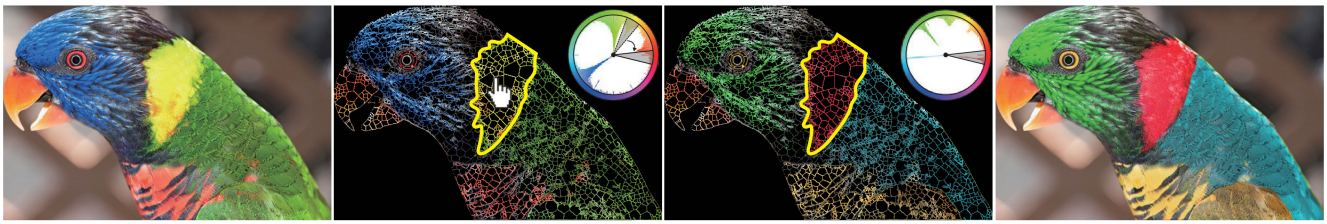

Fig. 13. Our color editing consists of the desired color selection, target color selection, and harmonization.

vector space without the need for intermediate raster representation and vectorization would be a boon to artists. Furthermore, raster-space operations generally require additional mechanisms to enforce spatial and temporal coherence among object patches and across different frames; however, the proposed representation achieves coherence directly, as shown in our supplementary videos. These applications are detailed in the following sections.

A. Image Magnification

Image magnification becomes increasingly important with the resolution of screens. Our representation enables natural magnification by directly scaling the parametric coordinate of all pixels, based on a given magnification ratio and rasterizing them based on these coordinates using the original TPS kernels. As shown in Fig. 12, the proposed system enables faithful preservation of structural and textural details, such as butterfly wing patterns and fish scales during magnification.

B. Color Editing

Color editing is generally used to manipulate the color of particular object regions while maintaining important border characteristics. Our parametric patches make it possible to limit color operations to desired object regions. It also enables the direct application of coloring operations to gradation and curvilinear features inside the desired region for the modification of appearance without altering the curvilinear features across the boundary, thereby maintaining important border characteristics, as shown in Fig. 13. This is due to the fact that two curvilinear features are used for representing both sides across the boundary. Users can select specific colors by pointing out desired features, our system propagates the selection to other features including both the curvilinear and gradation features of a similar color profile within a user-specified propagation radius. Users can then adjust the color by manipulating the ab-channel ring in the Lab space, and our system would propagate the manipulation to adjust all selected features accordingly while these color features would be used as new constraints for TPS interpolation. However, this generally requires the application of a harmonization operator [9] to obtain harmonic results. Users may also provide a 3×3 color transform matrix to transform the color of all gradation features in order to achieve interesting shading effects, such as cartoon shading.

C. Structural Color and Texture Transfer

Cross mapping is important in scenic transitions and special effects; however, it generally requires the manual construction

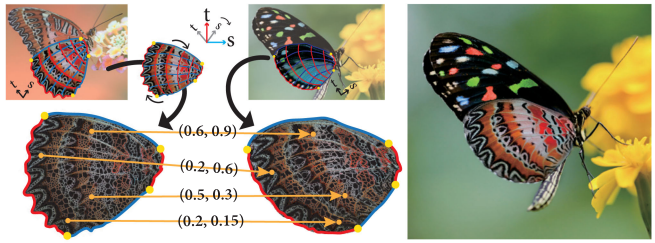

Fig. 14. Given the source and target patches, the proposed cross-mapping operator computes the source-target transformation using the source-target matching corners. Source features can be transformed onto the target patch to perform composition operations for final TPS rasterization.

of a correspondance map. As shown in Fig. 14, our parametric patches enable the direct creation of source-to-target inter-patch mapping by linking the source and target patches with the same normalized parameter space created by the four corners of the source and target patches. It is possible to forward-map a dense grid of samples to form the final result; however, this requires high resolution to avoid holes and a weighting scheme to deal with multiple samples mapped to the same destination pixel. Our GPU-based method makes it possible to rasterize the mapping results in real time without over-rendering. Our algorithm enables the seamless replacement of texture, as shown in Fig. 15.

Our editing tool also allows users to edit the colors of selected features based on their locality and color distribution. Our localized TPS interpolation scheme provides sub-patch overlapping to facilitate seamless blending; it does not provide a blending function across object patches. We therefore apply 3D remeshing operators [20], [28] to construct source-to-target correspondence and compute an alpha map across characteristic patches of the face. Users can take this one step further by adjusting the alpha map according to the effect they wish to achieve. Finally, our system applies color editing instructions to these composite features and then reconstructs the TPS kernels for rasterization.

D. Shape Manipulation

Our parametric patches align directly with object segments; therefore, we can provide intuitive high-level object-based shape manipulation rather than low-level feature-based manipulation, as shown in Fig. 16. Although raster- and object-space image-based methods can deform a butterfly as shown in Fig. 17, they might possibly deform the image structures and induce unwanted artifacts. Although a high-density mesh can relieve the deformation artifacts, the manipulation efficiency becomes an issue. However, our system provides another possibility of

Fig. 15. This shows the results of our cross-mapping operator. From left to right and top to bottom are the Banana, Kiwi, Butterfly1, and Butterfly2 scenes. The first column is the target patches, the second column is the source patches, and the third column is the cross-mapping results.

Fig. 16. The top is a duck and the bottom is a mark cup. From left to right are TPS results with parametric structures, 4-time magnifications, detailed features, TPS results with deformed parametric structures, 4-time magnifications, and relocated detailed features.

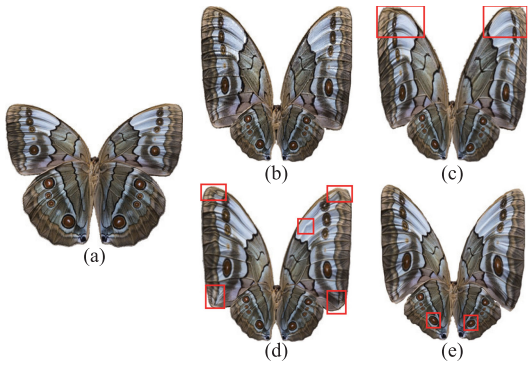

Fig. 17. This illustrates the benefit of shape manipulation in vector space to preserve the structural and textural information while comparing to other raster- and object-space editing algorithms. The input is a butterfly with beautiful and structural patterns (a). Users generally wants to have the interior patterns deformed with the exterior boundary (b) (our vector-space manipulation), but raster-space "arch warping" of Adobe Photoshop [36] crookedly bends those top boundary-perpendicular patterns due to uneven stretching of the top corner (c), object-space "as rigid as possible" of Adobe Photoshop [14] structurally distorts those top patterns because the mesh formation do not collide with the interior structure and the deformation distribution is not efficiently aligned with the structure, either (d), and object-space "Bounded Bihoarmonics Weighting" (BBW) [16] affine deform the bottom circular dot patterns (e).

real-time high-level shape editing while the results can faithfully preserve the structural and textural contents of a photorealistic image. As a result, the user can move the control points to alter the shape of the image. The system reconstructs the parametric space to relocate detailed features in order to retain consistency in the alignment of structural and textural elements, before rasterizing the final results. Furthermore, the parametric patches

Fig. 18. From left to right are the inputs, and the vectorized results and corresponding zoom-ins of Xie's [39] algorithm and ours in a lotus, butterfly, and fish.

are easily incorporated with other patch-based manipulation methods, such as [14], [16].

E. Abstraction and Stylization

Abstraction refers to image representation at various levels of detail, i.e., in different frequency bands. The proposed curvilinear features record cross-level visually important boundaries and borders with the amount of allowable variation determined by T^{Super} that mentioned in the Section IV-C. This means that the proposed system is able to select a T^{Super} value to detect fine-to-coarse color details among the curvilinear features. In other words, finer details are more faithfully represented using a smaller T^{Super}, whereas coarser details enable a higher degree of abstraction using a larger T^{Super}. Stylization refers to rendering an abstraction in a style reminiscent of painting, such as water color or oil painting. At all levels of abstraction, we are able efficiently transform and quantize gradation features to fill some areas with continuous color, while preserving more subdued variations in local regions. We can then layer and composite these stylized levels using brush strokes drawn along connected curvilinear features that are aligned with important borders in order to emphasize important elements via sharpness. As shown in Fig. 19, our vector-based approach to abstraction and stylization is better able to preserve edges, align features, and remove unwanted high-frequency detail. Furthermore, all of our manipulators work in the same global parametric space, which means

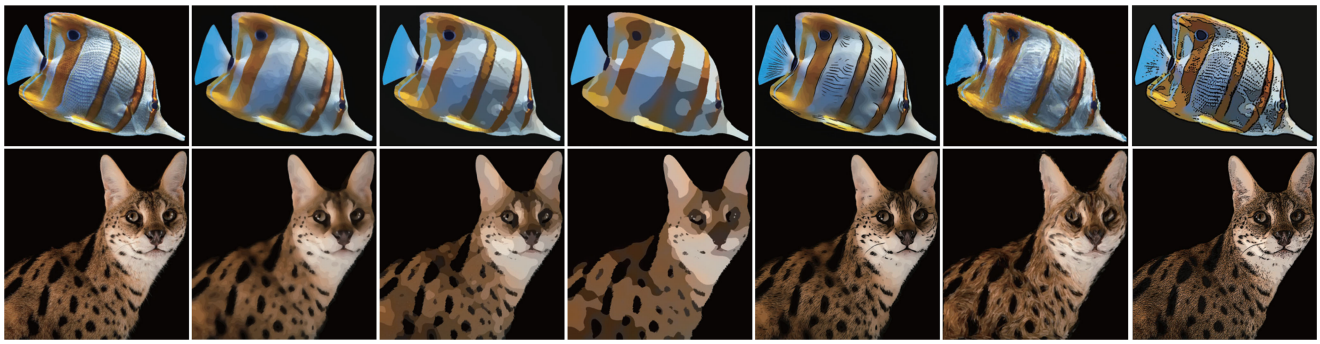

Fig. 19. From top to bottom show our abstraction and stylization results of a tropical fish and bobcat. From left to right are the input, three detail levels from fine to coarse, and three stylization results.

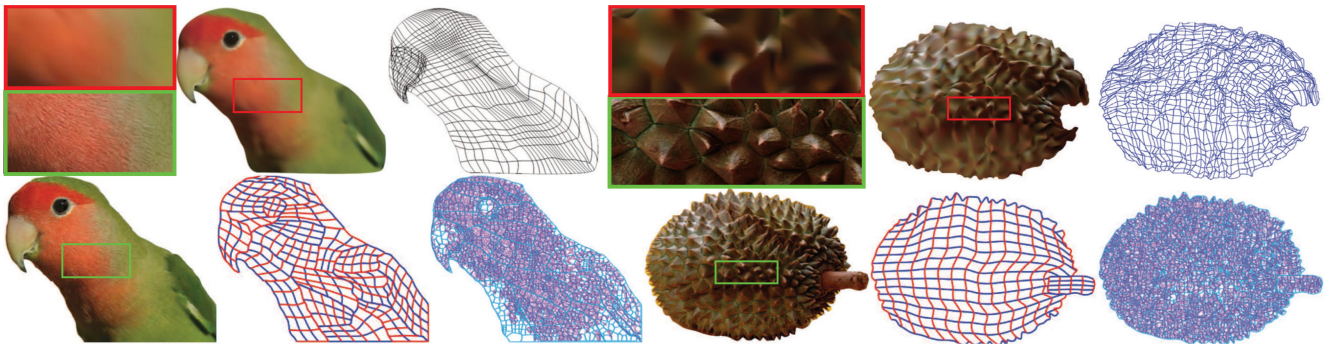

Fig. 20. The top shows the result of a parrot using optimized gradient meshes [31] and a durian using topology-preserving gradient mesh [21] along with 4-time magnification on their left top. The bottom shows our rasterized results, constructed parametric patches, and extracted details features of the parrot and durian. The left bottom in the top also shows the 4-time magnification of our rasterized results. Our algorithm can retain the regular mesh structure and fine details under the criterion of the same file size.

that our system can easily place the stylized strokes in a consistent manner across all frames to ensure temporal consistency and without flickering, as shown in our supplemental video. Our supplemental video also demonstrates the use of our abstraction, stylization operator and facial painter in producing temporally-coherent non-photorealistic effects.

VIII. Comparisons and Discussion

After inputting raster images and labeling maps, the proposed vectorization algorithm automatically creates a hybrid structure to ensure that the image remains editable and scalable. Due to length limitations, the complete results are held on supplemental website.[1] Parametric patch-based methods [21], [31], [37] provide low-level editing tools to manipulate basic patches and their color details. Ensuring that an image can be edited requires that the parametric meshes be simple; however, this can lead to the loss of fine detail, as shown in Fig. 20. In this image, only a few curvilinear features are extracted from the yellow petals and parrot, thereby hindering faithful reconstruction of the associated details. This is a clear demonstration of the trade-off between reconstruction quality, compressibility, expressibility, and editing complexity.

[1]web site address: http://graphics.csie.ntust.edu.tw/pub/RealTimeTPS/

Additionally, we also compare our reconstruction results with the curve-based vectorization algorithm proposed by Xie *et al.* [39] in Fig. 18. While visually examining the zoom-ins, our system generally can preserve more details. When matching the same file size of these methods, the proposed system makes it possible to avoid these problems by using simple parametric patches for editing and using features for detail restoration. Our representation records a feature using (x, y, R, G, B):$2 \times 2 + 3 = 7bytes$. A parametric patch requires four corner points and the s, t, and st derivatives for $4 \times (2 + 2 + 2 + 2) = 32bytes$. This means that we need a total of $(\#features \times 7bytes + \#patches \times 8bytes + \#LabelingMap)$ bytes, which are recorded as a sequence of numbers. A general compression algorithm, such as Zip, can be used to achieve compression of approximately 75%. Reports on other state-of-art algorithms provided results in raw format; therefore, we adopted the same standard. Table III shows that our representation outperformed the other vector representations [23], [37], [40] in terms of mean construction error, compression ratio, and compression efficiency. JPEG is a popular raster image format due to its efficiency and compressibility; therefore, we compared the performance of our algorithm with that of JPEG in terms of magnification. We homogenously scaled down two full HD images (beef and parrot) to one-fourth of the original size. We then vectorized the scaled results using our representation

TABLE III
THIS ILLUSTRATES THE COMPARISON OF OUR THE PROPOSED REPRESENTATION AGAINST AND OTHER STATE-OF-THE-ART METHODS. - MARKS THAT THE RELEVANT STUDY DOES NOT PROVIDE THISWE CANNOT FIND THE DATA IN THE WORK, AND. OURS, PATCH, AND SUB. RESPECTIVELY DENOTE OUR METHOD, THE PATCH-BASED METHOD [37], THE SUBDIVISION METHOD [23], AND THE CURVILINEAR METHOD [40] RESPECTIVELY

Image	Mean error				Compression ratio				Compression efficiency			
	Ours	Patch	Sub.	Curve	Ours	Patch	Sub.	Curve	Ours	Patch	Sub.	Curve
Buddha	6.00e-6	1.97e-5	-	-	0.59	0.66	-	-	2.74e5	7.67e4	-	-
Face	5.38e-6	1.18e-5	2.28e-5	1.65e-5	0.82	0.72	-	0.46	2.25e5	1.18e5	-	1.32e5
Flower	5.54e-6	1.51e-5	-	1.31e-5	0.89	0.26	-	1.13	1.99e5	2.54e5	-	6.76e4
Lena	1.38e-5	3.69e-5	-	3.23e-5	0.86	1.01	-	0.69	8.39e4	2.67e4	-	4.49e4
Goldfish	7.69e-6	2.48e-5	-	-	0.61	0.42	-	-	2.14e5	9.62e4	-	-
Green jade	1.23e-5	-	-	3.28e-5	0.53	-	-	0.53	1.52e5	-	-	5.79e4
Peppers2	1.06e-5	-	-	1.91e-5	0.62	-	-	0.62	1.50e5	-	-	8.45e4
Egg	9.13e-6	-	2.15e-5	-	0.31	-	-	-	3.56e5	-	-	-

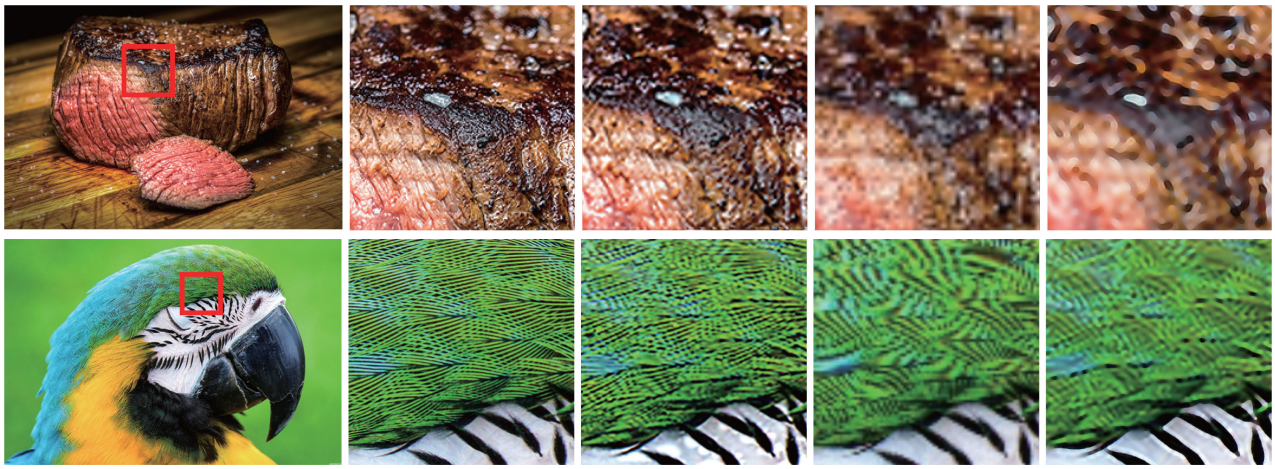

Fig. 21. This illustrates the magnification comparison between our algorithm and JPEG on a beef (top) and parrot (bottom). From left to right are the original image and the 4X magnification of the original, our reconstruction result, JPEG result with bilinear interpolation, and JPEG result with a commercial super resolution software [3].

and magnified them back to the original size. We also applied similar effects using raster-based magnification with bilinear interpolation or super resolution [3]. As shown in Fig. 21, our algorithm preserved the detail, whereas raster-based magnification tended to blur the detail and induce aliasing. Furthermore, the proposed algorithm achieved compression efficiency of 85.3 and 165.5, whereas JPEG achieved compression efficiency of 65.9 and 149.7 (bilinear interpolation) and 76.7 and 154.8 (super resolution).

IX. CONCLUSION AND FUTURE WORK

In this work, we developed a hybrid vector representation using parametric patches to enable editing and detailed color features to enable scaling and ensure compactness. Our system selects optimal thresholds for feature extraction using a genetic algorithm with a novel metric of compression efficiency while having a good start-up set estimated using a gradient histogram. Then, we register the extracted features within parametric patches for GPU-based parallel TPS rasterization in real time. Our real-time TPS kernel construction, inversion, and rasterization scheme makes it possible to perform vector-based image magnification, color editing, and cross mapping. The proposed algorithm provides compressibility, scalability, and editability superior to those of state-of-the-art algorithms. It also achieved compressibility on par with JPEG with superior scalability.

The proposed system is not without limitations. Although scattering data interpolation permits localized acceleration and manipulation, while rasterizing with a very large magnitude of magnification, these point constraints become sparse to induce aliasing and blurring artifacts. Furthermore, point-based features are not effective and efficient for encoding line and curve features, and curvilinear features represented as a feature pair requires extra memory. Therefore, we would like to encode line and curve features as curve-based features along with a discontinuity embedment scheme for TPS interpolation in order to have better compression rate and rasterization speed. While rasterizing our patches, we transform the pixel coordinate to the parametric coordinate with a 2D binary searching process. This operation may induce numerical errors and require extra computation time. We would like to use newly available tessellation shaders to accelerate the process. Our framework currently only subdivides an object segment based on its feature density, but this may induce varied rasterization areas and feature densities among patches from different segments, i.e., to have different payloads for GPU threads. In the future, we will take segmented areas into consideration in the subdivision of patches. Packing features can be used to balance kernel payloads

to improve performance; however, when the feature density of patches varies too widely, packing becomes less than optimal. It would be preferable to sort patches based on feature density and arrange inversion kernels with a similar number of features in the same computation group. This could greatly improve the efficiency of packing and TPS interpolation. Our framework currently uses the CPU to build TPS kernels for each sub-patch and send them to the GPU for inversion; however, packing the same number of features requires the storage of multiple copies of the same feature, thereby imposing undue burden on memory usage and curtailing CPU-GPU data transmission bandwidth. We would therefore also like to deploy memory indexing that includes the usage of shared memory among threads in order to resolve this issue. While applying our algorithm to a sequence of frames, independently feature extraction may induce temporal inconsistency and flickering artifacts. Therefore, we would like to develop a temporal feature extraction and TPS interpolation scheme for video vectorization in future. Finally, our algorithm currently depends on the input labelling map to create operational patches for various topologies. We would like to develop an automatic labelling mechanism based on object recognition and segmentation.

REFERENCES

[1] R. Achanta *et al.*, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[2] M. K. Agoston, *Computer Graphics And Geometric Modeling: Implementation and Algorithms*. London, U.K.: Springer, 2005.

[3] Akvis, Magnifier, Jun. 26, 2019. [Online]. Available: http://akvis.com/en/magnifier/index.php

[4] W. A. Barrett and A. S. Cheney, "Object-based image editing," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 777–784, 2002.

[5] H. Bezerra, E. Eisemann, D. DeCarlo, and J. Thollot, "Diffusion constraints for vector graphics," in *Proc. 8th Int. Symp. Non-Photorealistic Animation Rendering*, 2010, pp. 35–42.

[6] J. C. Bowers, J. Leahey, and R. Wang, "A ray tracing approach to diffusion curves," in *Proc. 22nd Eurographics Conf. Rendering*, 2011, pp. 1345–1352.

[7] S. Boyé, P. Barla, and G. Guennebaud, "A vectorial solver for free-form vector gradients," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 173:1–173:9, 2012.

[8] B. Chazelle and D. P. Dobkin, "Optimal convex decompositions," *Mach. Intell. Pattern Recognit.*, vol. 2, pp. 63–133, 1985.

[9] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, "Color harmonization," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 624–630, 2006.

[10] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," *Int. J. Comput. Vis.*, vol. 96, no. 1, pp. 1–27, 2012.

[11] L. Demaret, N. Dyn, and A. Iske, "Image compression by linear splines over adaptive triangulations," *Signal Process.*, vol. 86, pp. 1604–1616, 2006.

[12] J. H. Elder, "Are edges incomplete?" *Int. J. Comput. Vis.*, vol. 34, no. 2/3, pp. 97–122, 1999.

[13] M. Finch, J. Snyder, and H. Hoppe, "Freeform vector graphics with controlled thin-plate splines," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 166:1–166:10, 2011.

[14] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1134–1141, 2005.

[15] P. Ilbery, L. Kendall, C. Concolato, and M. McCosker, "Biharmonic diffusion curve images from boundary elements," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 219:1–219:12, 2013.

[16] A. Jacobson, I. Baran, J. Popović, and O. Sorkine-Hornung, "Bounded biharmonic weights for real-time deformation," *Commun. ACM*, vol. 57, no. 4, pp. 99–106, Apr. 2014.

[17] S. Jeschke, D. Cline, and P. Wonka, "A GPU Laplacian solver for diffusion curves and poisson image editing," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 116:1–116:8, 2009.

[18] S. Jeschke, D. Cline, and P. Wonka, "Rendering surface details with diffusion curves," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 117:1–117:8, 2009.

[19] S. Jeschke, D. Cline, and P. Wonka, "Estimating color and texture parameters for vector graphics," *Comput. Graph. Forum*, vol. 30, no. 2, pp. 523–532, 2011.

[20] V. Kraevoy and A. Sheffer, "Cross-parameterization and compatible remeshing of 3d models," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 861–869, 2004.

[21] Y.-K. Lai, S.-M. Hu, and R. R. Martin, "Automatic and topology-preserving gradient mesh generation for image vectorization," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 85:1–85:8, 2009.

[22] G. Lecot and B. Levy, "ARDECO: Automatic region detection and conversion," in *Proc. 17th Eurograph. Symp. Rendering*, 2006, pp. 349–360.

[23] Z. Liao, H. Hoppe, D. Forsyth, and Y. Yu, "A subdivision-based representation for vector image editing," *IEEE Trans. Visualization Comput. Graph.*, vol. 18, no. 11, pp. 1858–1867, Nov. 2012.

[24] A. Orzan *et al.*, "Diffusion curves: A vector representation for smooth-shaded images," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 92:1–92:8, 2008.

[25] W. M. Pang, J. Qin, M. Cohen, P. A. Heng, and K. S. Choi, "Fast rendering of diffusion curves with triangles," *IEEE Comput. Graph. Appl.*, vol. 32, no. 4, pp. 68–78, Jul./Aug. 2012.

[26] J. Powell, *A Thin Plate Spline Method for Mapping Curves Into Curves in Two Dimensions*. University, U.K.: Cambridge DAMTP. Department of Applied Mathematics and Theoretical Physics, Univ. of Cambridge, 1995.

[27] R. Prévost, W. Jarosz, and O. Sorkine-Hornung, "A vectorial framework for ray traced diffusion curves," *Comput. Graph. Forum*, vol. 34, no. 1, pp. 253–264, 2015.

[28] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, "Inter-surface mapping," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 870–877, 2004.

[29] O. Sorkine and D. Cohen-Or, "Least-squares meshes," in *Proc. Shape Model. Appl.*, 2004, pp. 191–199.

[30] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, vol. 65. Hoboken, NJ, USA: Wiley, 2005.

[31] J. Sun, L. Liang, F. Wen, and H.-Y. Shum, "Image vectorization using optimized gradient meshes," *ACM Trans. Graph.*, vol. 26, no. 3, 2007, Art. no. 11.

[32] T. Sun, P. Thamjaroenporn, and C. Zheng, "Fast multipole representation of diffusion curves and points," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 53:1–53:12, 2014.

[33] X. Sun *et al.*, "Diffusion curve textures for resolution independent texture mapping," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 74:1–74:9, 2012.

[34] S. Swaminarayan and L. Prasad, "Rapid automated polygonal image decomposition," in *Proc. 35th IEEE Appl. Imagery Pattern Recognit. Workshop*, 2006, pp. 28-1–28-6.

[35] D. Terzopoulos, "Multilevel computational processes for visual surface reconstruction," *Comput. Vis., Graph. Image Process.*, vol. 24, no. 1, pp. 52–96, 1983.

[36] D. W. Thompson, *On Growth and Form: The Complete Revised Edition*. New York, NY, USA: Dover, 1995.

[37] T. Xia, B. Liao, and Y. Yu, "Patch-based image vectorization with automatic curvilinear feature alignment," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 115:1–115:10, 2009.

[38] Y. Xiao, L. Wan, C. Leung, Y. Lai, and T. Wong, "Example-based color transfer for gradient meshes," *IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 549–560, Apr. 2013.

[39] G. Xie, X. Sun, X. Tong, and D. Nowrouzezahrai, "Hierarchical diffusion curves for accurate automatic image vectorization," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 230:1–230:11, 2014.

[40] H. Zhou, J. Zheng, and L. Wei, "Representing images using curvilinear feature driven subdivision surfaces," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3268–3280, Aug. 2014.

Kuo-Wei Chen received the B.S. degree from the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology (NTUST), Taipei, China, in 2013, and the M.S. degree from the Department of Computer Science and Information Engineering, NTUST, Taipei, Taiwan, R.O.C., in 2015. He is currently working toward the Ph.D. degree at the Department of Computer Science and Information Engineering, NTUST. His research interests include the area of graphics, vision, and multimedia.

Ying-Sheng Luo received the M.S. degree from the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C., in 2018. He is currently a Full Time Researcher with Inventec AI center, Taipei, Taiwan. His research interests include the area of graphics, vision, and multimedia.

Yu-Chi Lai received the B.S. degree from the Electrical Engineering Department, National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, R.O.C., in 1996, and the M.S. and Ph.D. degrees in electrical and computer engineering from University of Wisconsin—Madison, Madison, WI, USA, in 2003 and 2009, respectively, and the M.S. and Ph.D. degrees in Computer Science, in 2004 and 2010, respectively. He is currently an Assistant Professor with NTUST. His research interests include the area of graphics, vision, and multimedia.

Yan-Lin Chen received the B.S. degree from the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, R.O.C., in 2017. He is currently working toward the graduate degree at the Department of Computer Science and Information Engineering, NTUST. His research interests incluede the area of graphics, vision, and multimedia.

Chih-Yuan Yao received the M.S. and Ph.D. degrees in computer science and information engineering from National Cheng-Kung University, Tainan, Taiwan, in 2003 and 2010, respectively. He is an Assistant Professor with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan. His research interest include computer graphics, including mesh processing and modeling, and non-photorealistic rendering (NPR).

Hung-Kuo Chu received the B.S. and Ph.D. degrees in computer science and information engineering (CSIE) from National Cheng-Kung University (NCKU), Tainan, Taiwan, in 2003 and 2010, respectively. After the graduation, he was recruited with a summer visiting internship at Yahoo! Inc. Research Lab under the supervision of B. Tseng and S. Mittur. His major research interests include computer graphics including specific topics such as geometry processing, texture/image processing, human computer interaction and visual perception.

Tong-Yee Lee received the Ph.D. degree in computer engineering from Washington State University, Pullman, WA, USA, in May 1995. He is currently the Chair Professor with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, R.O.C. His current research interests include computer graphics, nonphotorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, medical visualization and medical systems, and distributed and collaborative virtual environments.