Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2013.mssvm16

Vol. **6**, No. 1, pp. 297-324
February 2013

Stroke-Based Surface Reconstruction

Jooyoung Hahn[1,*], Jie Qiu[2], Eiji Sugisaki[3], Lei Jia[2],
Xue-Cheng Tai[4] and Hock Soon Seah[2]

[1] *Institute of Mathematics and Scientific Computing, University of Graz, Austria.*
[2] *School of Computer Engineering, Nanyang Technological University, Singapore.*
[3] *N-Design Inc., Japan.*
[4] *Mathematics Institute, University of Bergen, Norway.*

Received 13 December 2011; Accepted (in revised version) 13 June 2012

Available online 11 January 2013

Abstract. In this paper, we present a surface reconstruction via 2D strokes and a vector field on the strokes based on a two-step method. In the first step, from sparse strokes drawn by artists and a given vector field on the strokes, we propose a nonlinear vector interpolation combining total variation (TV) and H^1 regularization with a curl-free constraint for obtaining a dense vector field. In the second step, a height map is obtained by integrating the dense vector field in the first step. Jump discontinuities in surface and discontinuities of surface gradients can be well reconstructed without any surface distortion. We also provide a fast and efficient algorithm for solving the proposed functionals. Since vectors on the strokes are interpreted as a projection of surface gradients onto the plane, different types of strokes are easily devised to generate geometrically crucial structures such as ridge, valley, jump, bump, and dip on the surface. The stroke types help users to create a surface which they intuitively imagine from 2D strokes. We compare our results with conventional methods via many examples.

AMS subject classifications: 65K10, 65D18, 65D17

Key words: Surface reconstruction from a sparse vector field, augmented Lagrangian method, two-step method, curl-free constraint, total variation regularization, preservation of discontinuities in surface normal vectors.

1. Introduction

Sketch-based interfaces for modeling (SBIM) has been substantially explored by many researchers because of efficiency and intuitiveness, from the early systems like SKETCH [1] and Teddy [2] to recent SmoothSketch [3] and FiberMesh [4]. A thorough review of SBIM systems can be found in [5]. Without lighting or shading cues in photometric stereo [6–8]

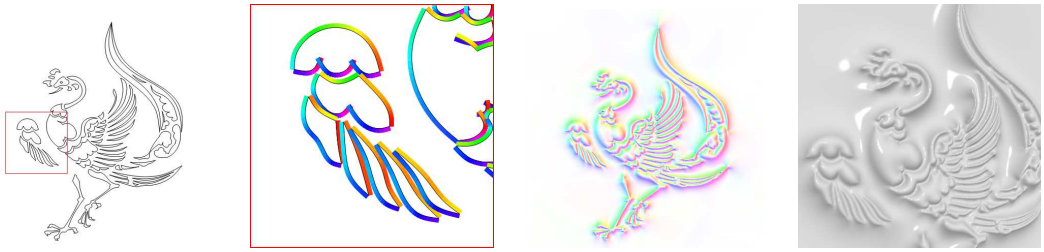
*Corresponding author. *Email addresses:* JooyoungHahn@gmail.com (J. Hahn), JQiu@ntu.edu.sg (J. Qiu), eiji@ndesign.co.jp (E. Sugisaki), JIALEI@ntu.edu.sg (L. Jia), tai@math.uib.no (X.-C. Tai), ASHSSEAH@ntu.edu.sg (H. S. Seah)

or shape-from-shading [9,10], or geometric constraints defined by 3D curve network [11], the task of 3D model reconstruction from 2D line drawings is more challenging than image-based 3D reconstruction. A lot of research results have been achieved towards this challenging task based on contours [2], hidden contours [3], symmetric sketch pairs [12], and structured annotations [13]. The models created by these systems are limited in structures of shape. Moreover, surface details such as the crease structure are not considered in these systems. Note that the crease structure can be added by surficial augmentation techniques [14]. Motivated by the above mentioned works, we are targeting at a sketch-based modeling system which can model complex 3D objects with simple sketches. As the first attempt to this direction, we present our research achievement on surface height reconstruction via 2D strokes and a vector field on the strokes.

Many surface reconstruction algorithms from surface gradients [15–18] enforce the integrability for producing correct surface heights. For single-view modeling, the authors [19] used a constrained optimization with many types of geometrical constraints. The authors [20] showed that the method in [19] requires a lot of user interactions to provide enough constraints for modeling a desirable surface. Another single-view modeling system [21] used a close form method to reconstruct curved 3D surfaces based on apparent contour, inflation constraints, and normal specification in the parameter space. Recently, the authors [22] highlighted that the algorithms in [19–21] have difficulties in solving three problems in surface-from-gradients: handling sparse gradients, preserving sharp features, and preventing surface distortion. To solve these problems, Gaussian kernel approach without discrete integrability enforcement is used in [22]. Our proposed method is also capable of handling these problems because of TV regularization and curl-free constraints imposed in a nonlinear vector interpolation. On the top of this, we provide a very fast and efficient algorithm to solve the proposed energy functionals via augmented Lagrangian method [23].

Based on the observation that humans are good at assigning local surface normals for specifying local shape [24], the authors in [20, 25] achieved stroke-based surface reconstruction in two steps. In the first step, dense surface normals are obtained from sparse vectors on given strokes via linear vector interpolation methods. In the second step, the dense normals are integrated for reconstructing a height map. In LUMO method [25], a method of interpolating vectors is based on so-called Telegrapher's equation (the damped wave equation). In ShapePalettes [20], an energy functional minimization is used and the main mechanism of interpolation is based on a parabolic type partial differential equation (PDE) with the fourth order term related to a surface curvature. The advantage of LUMO and ShapePalettes methods is that the governing equations are linear PDEs so that standard efficient numerical solvers such as the multigrid method or the fast Fourier transform (FFT) can be applied. However, the disadvantage is that both models are based on the Laplace operator which strongly enforces the smoothness in interpolated vector field. That is, the assigned vectors on strokes are diffused into the whole domain and then eventually averaged out in the final steady state. Therefore, interpolated vector fields from LUMO and ShapePalettes simply yield smooth surfaces; see Figs. 12 and 13.

The pioneering work for surface reconstruction from sparse information based on an



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

Figure 1: Phoenix example: (a) is a line drawing by an artist, referring to an ancient Chinese phoenix shape. (b) is the automatically assigned default initial vectors (needle map). (c) is the dense vectors obtained by the proposed nonlinear vector interpolation from the sparse vectors in (b). (d) is the reconstructed surface.

energy minimization approach is the visible-surface representations in [26]. An weighted H^1 norm of data and its gradient with discrete fidelities control the local smoothness of reconstructed surface. However, it is still difficult to approximate a proper weight function. More advanced method to deal with local smoothness is to use TV norm of data [27]. As jump discontinuities under TV norm of data are easily preserved without weight functions, discontinuities of surface gradient such as valleys or ridges are also recovered by TV norm of data gradient. We explore even further to devise a method for stroke-based surface reconstruction via a combination of TV and H^1 norm. The combination gives artists more freedom to create various types of surface geometries.

We use the same procedure proposed in [20] which generates a surface height map from sparse strokes and a vector field on the strokes. The procedure with the results from our method is illustrated in Fig. 1. In the drawing stage (a), an artist first draws basic 2D shapes with strokes. In [20], vectors on the strokes are manually assigned based on a 3D reference model. However, we provide automatic assignment for initial vectors orthogonal to the strokes tangents with a constant magnitude; see Fig. 1-(b). The vectors can be easily adjusted by the artist according to a more detailed surface shape. Considering assigned vectors as a projection of surface normal vectors onto the plane, different types of strokes are easily devised to generate geometrically crucial structures such as ridge, valley, jump, bump, and dip on the surface.

In this paper, the proposed method for surface reconstruction is divided into two steps: vector interpolation and height map reconstruction. In the interpolation stage in Fig. 1-(c), we propose a nonlinear vector interpolation to obtain the dense normal vectors from sparsely distributed vectors in (b). The TV regularization in the vector interpolation can preserve discontinuities in the vector field. While the assigned vectors are interpolated in the domain, the integrability constraint is well imposed almost everywhere. Moreover, this constraint does not introduce any distortion on a reconstructed surface. In the reconstruction stage in Fig. 1-(d), a height map is reconstructed via integrating the dense normal vectors obtained in the interpolation stage. The TV regularization in height map reconstruction can preserve jump discontinuities on a reconstructed surface without overshooting or undershooting.

The main contributions of our research are summarized as follows:

- Strokes are categorized to create crucial structures such as ridge, valley, jump, bump, and dip on the surface. The stroke types help users to create a surface which they intuitively imagine from 2D strokes; see Figs. 10-(d) and 14.
- A nonlinear vector interpolation and height map reconstruction method are proposed based on minimization of energy function. The interpolated vector field satisfies the integrability condition almost everywhere, which makes more desirable results corresponding to given strokes and vector settings; see Figs. 7 and 6.
- A combination of TV and H^1 regularization can present geometrically crucial structures; see Figs. 8 and 10-(b).
- Even though the nonlinear process is used in both vector interpolation and height map reconstruction, very fast and efficient numerical solvers are proposed based on augmented Lagrangian method [23, 28].

The paper is organized as follows. In Section 2, we classify strokes into several types and describe the meaning of vectors on the strokes. In Section 3, our proposed functionals for nonlinear vector interpolation and height map reconstruction are introduced. Then, we explain the proposed numerical solvers which are fast and practically easy to implement in Section 4. In Section 5, we compare with the existing state-of-the-art applications in terms of quality and demonstrate many examples. The conclusion and future work are described in Section 6.

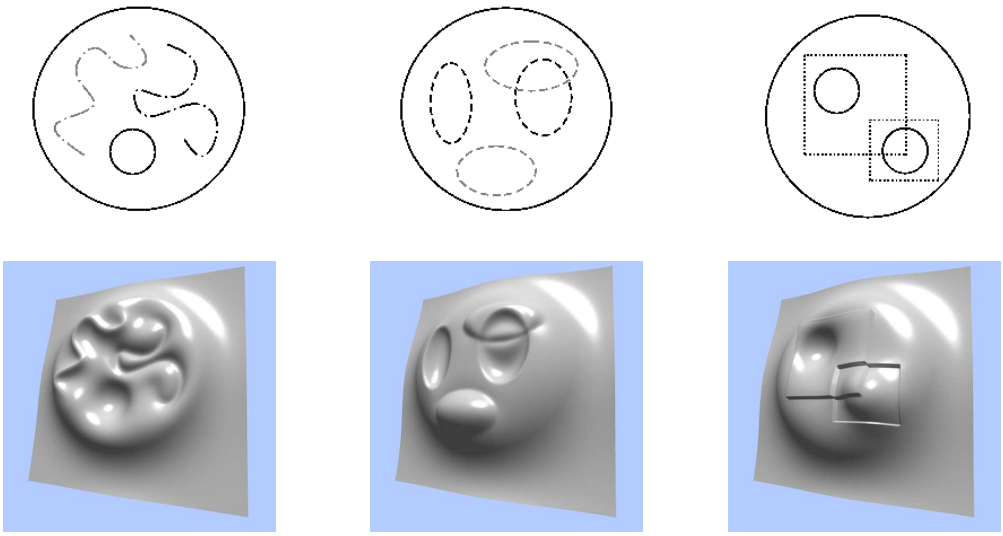
2. Vector settings on strokes

2.1. Stroke definition

In terms of sketch-based modeling for surface reconstruction, it would be better to use simple line drawings which capture the characteristics of surface. The strokes in our paper have the style of silhouette drawing as cleaned-up vectorized strokes in Fig. 1-(a). Shading, highlighting, hatching strokes, and stippling are not considered.

2.2. Stroke classification

Inspired by a recent study on where artists draw lines to convey 3D shapes [29], human vision easily detects jump, ridge or valley, bump or dip on surfaces. Therefore, we classify strokes into different types to reconstruct distinctive structures. Note that jump discontinuity is regarded as discontinuity of surface height which appears to be a step-like structure. Fig. 2 shows examples of stroke types and corresponding surface structures. Different types of strokes are illustrated as regular (black solid line), ridge (gray dashdot line), valley (black dashdot line), jump (black dotted line), bump (gray dashed line), and dip (black dashed line).



(a) Ridge or valley strokes (b) Bump or dip strokes (c) Jump strokes

Figure 2: Effects of the different types of strokes.

2.3. Vector settings

The main reason why we can reconstruct a proper height map indicated by different types of strokes is that we assign geometrically meaningful vectors on the strokes and we adopt a nonlinear vector interpolation to obtain a dense normal vector field which satisfies the integrability condition.

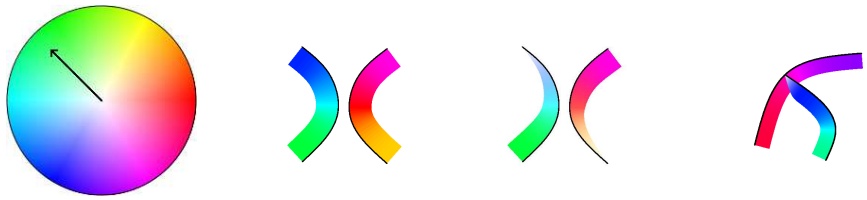
In this subsection, the meaning of vectors on different strokes is explained and we demonstrate how easy it is to slightly modify initial vectors for obtaining more desirable surfaces. The modification is very intuitive as long as one can understand the geometrical meaning of assigned vectors on each stroke, which is explained as follows.

In our method, the default initial vectors on given strokes are orthogonal to the stroke tangents with a constant magnitude. We regard the assigned vectors as the projection of 3D surface normals with desired height map $z = I(x_1, x_2)$ onto a 2D plane. That is, since the direction of surface normal vector is

$$(-\partial_1 I, -\partial_2 I, 1),$$

the assigned vectors $\mathbf{n}^*$ on all strokes (except jump strokes) are considered as $-\nabla I \equiv (-\partial_1 I, -\partial_2 I)$.

The direction of vector $\mathbf{n}^*$ indicates that the height of surface I is decreasing along the direction. The magnitude of vector $\mathbf{n}^*$ controls the rate of change in surface height. If the magnitude of the vector becomes larger, the change of the surface height becomes steeper. Note that the assigned vectors on jump strokes are not regarded as the projection of 3D surface normals but the magnitude of assigned vectors represents the size of the jump.

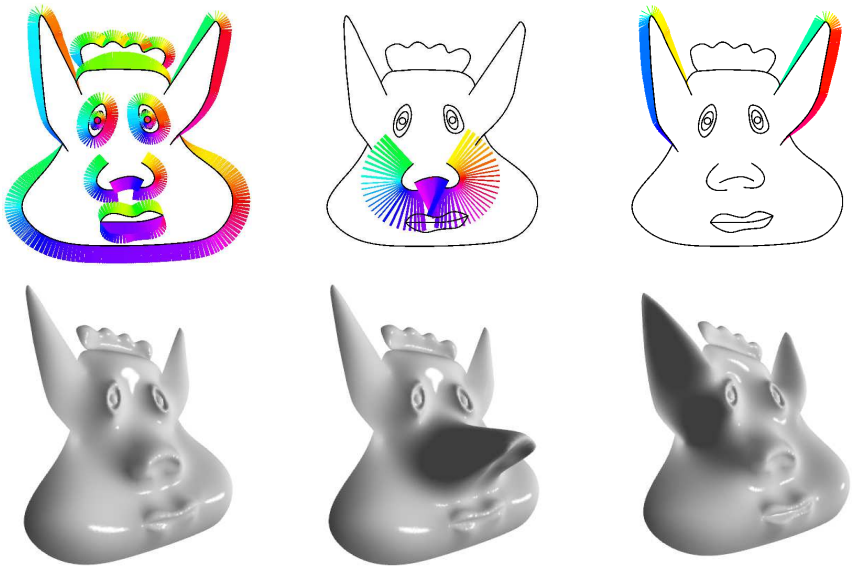


(a) (b) (c) (d)

Figure 3: HSV vector representation and vector needle map for initial vectors on strokes: (a) HSV color map, (b) default vector setting, (c) User-defined setting, and (d) T-junction handling.

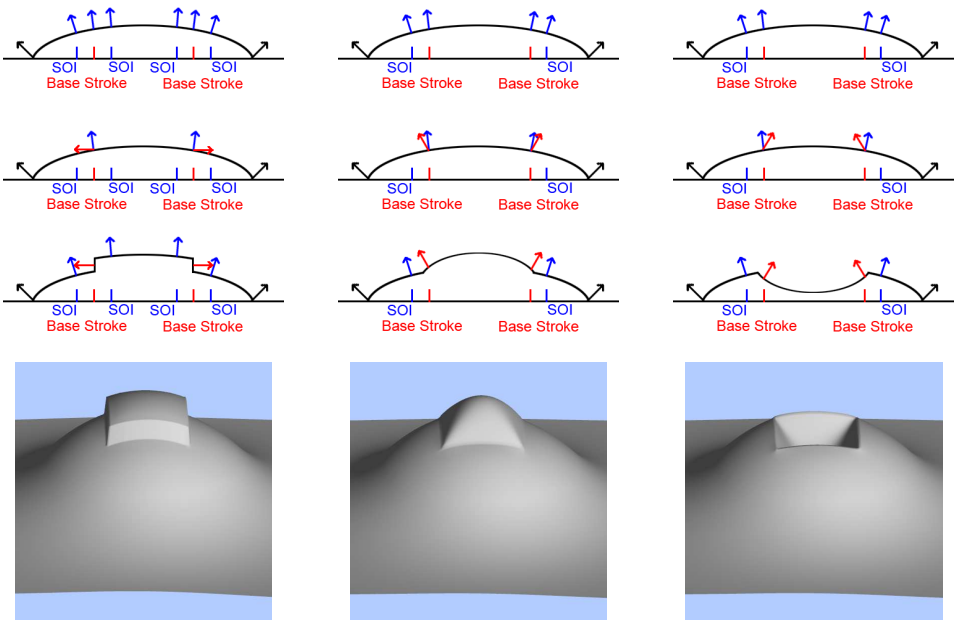
For efficiently illustrating vectors on strokes, we adopt the HSV color map with the value $V = 1$ in Fig. 3-(a). A default initial vector setting on all strokes is shown as needle map with HSV color map in Fig. 3-(b). Artists can easily change the direction and magnitude of vectors flexibly, for example in Fig. 3-(c). The magnitudes of vectors near the T-junctions along the visible part of the occluded contour are automatically reduced to avoid affecting the vectors specified for the contour occluding it, as shown in Fig. 3-(d).

Regular strokes are most commonly used to make an overall shape of reconstructed surface. Default initial vectors on regular strokes generate a default height map accordingly; see Figs. 1-(d) and 4-(a). The only user interaction is to slightly modify magnitudes or directions of vectors for reconstructing more desired surface. The effect of modification is easily expected. In Fig. 4-(b), increasing vector magnitudes on strokes of nose creates



(a) (b) (c)

Figure 4: The changes of the magnitude or direction of vector settings affect the shape of surfaces: (a) Default initial vector. (b) Increasing vector magnitudes on strokes of nose makes the nose higher. (c) Rotating the vector directions on strokes of ears twists the shape of surface.



(a) Jump (b) Bump (c) Dip

Figure 5: Base stroke and strokes of interest (SOI) on jump, bump, and dip strokes: The black, blue and red arrows indicate the normal directions for the regular stroke, surface, and base stroke, respectively. The first, second, and third rows are conceptual drawings. The first and third rows show the surfaces without and with jump, bump, and dip strokes, respectively. The second row shows the change of the surface normals on the base stroke. The vectors on the SOIs are same as the projection of the given surface normal. The fourth row shows the reconstructed surfaces. The indicated structures by jump, bump, and dip strokes are well reconstructed on the top of the given surface shape.

a higher nose than (a). In Fig. 4-(c), rotating the directions of vectors on strokes of ears (anti-clockwise for 15 degrees) creates twisted shape of surface. Note that we apply the rotation of vectors only to regular strokes. Although the rotation of vectors on other types of strokes changes the geometry of reconstructed surface, the practical effect is not very intuitive. For making the system user-friendly, we simply ignore the rotation of vectors on other types of strokes.

A ridge (or valley) stroke generates a sharp ridge (or valley) on the reconstructed height map. It is composed of two adjacent parallel regular strokes. For sharp ridge (or valley) structure, initial vector settings on two parallel regular strokes have repulsive (or attractive) directions and they are orthogonal to stroke tangents. The same idea is suggested in ShapePalettes [20]. Since we do not change the directions of vectors on ridge and valley strokes, the user interaction allowed is to change the magnitudes of vectors. The larger magnitude of assigned vectors creates steeper ridge (or deeper valley) structure in the reconstructed surface.

Jump, bump, and dip strokes are utilized to generate more exquisite structures on the top of the given surface shape. Each type of them is composed of a base stroke and strokes of interest (SOI); see Fig. 5.

Jump stroke has two SOIs which are parallel and adjacent to its base stroke in Fig. 5-(a). The magnitude of vector on the base stroke indicates the size of jump. The surface height on the base stroke is suddenly decreased along the directions of vectors on the base stroke. Note that the magnitudes of vectors on the base stroke are not projected surface gradients. The vectors on SOIs are chosen to be the projection of given surface normals at the position of SOIs to reflect the given surface structure. If there is no given surface, the vectors on SOIs are initialized as zero.

A bump (or dip) stroke has one SOI which is parallel and adjacent to its base stroke in Figs. 5-(b) and (c). The vectors on SOI are chosen as same as the projection of given surface normals at the position of SOI. If there is no given surface, the vectors on SOIs are initialized as zero. The magnitudes of vectors on the base stroke have same meaning as those of vectors on regular strokes.

A procedure of using different strokes is as follows. First of all, artists draw a rough shape of desired surface by regular strokes. Ridge or valley strokes can also be drawn. The default setting of assigned vectors generates a default surface. Artists can easily modify the magnitude or direction of vectors for obtaining more desirable surfaces. For reconstructing more detailed structures, jump, bump, or dip strokes are provided.

3. Proposed method

In this section, we present a reconstruction of height function $z = I(x_1, x_2)$ on a computational domain $\Omega \subset \mathbf{R}^2$ from sparse strokes $\Gamma \subsetneq \Omega$ and a vector field $\mathbf{n}^* = (n_1^*, n_2^*)^{\mathrm{T}}$ on Γ. From the proposed nonlinear vector interpolation in Subsection 3.1, we obtain the dense normal vector $\mathbf{n} = (n_1, n_2)^{\mathrm{T}}$ on Ω. After obtaining the vectors, we find a surface I whose gradient field fits the dense vector field $\mathbf{n}$ in Subsection 3.2.

3.1. Nonlinear vector interpolation

First of all, we review a theorem in vector calculus. If the domain $\Omega \subset \mathbf{R}^2$ is simply connected,

$$\begin{aligned} &\nabla \times \mathbf{n} \equiv \partial_1 n_2 - \partial_2 n_1 = 0 \\ &\qquad \text{if and only if} \quad \exists I : \Omega \to \mathbf{R} \ \text{ s.t. } \ \nabla I = \mathbf{n}. \end{aligned}$$

Since we reconstruct a height map I whose gradient fits the interpolated vectors $\mathbf{n}$, the curl-free condition on the vector field $\mathbf{n}$ should be guaranteed. Note that the curl-free condition is usually known as the integrability condition in visible surface reconstruction.

Now, for obtaining a dense curl-free vector field $\mathbf{n}$ from a given vector field $\mathbf{n}^*$ on the strokes Γ, we propose an energy minimization functional:

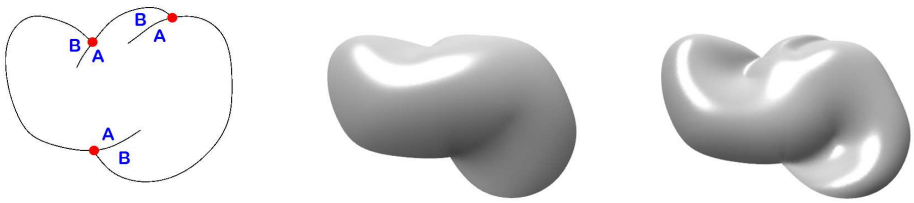
$$\min_{\mathbf{n}} \left\{ \mathscr{E}_v(\mathbf{n}) \equiv \int_\Omega (1-g)|\nabla \mathbf{n}|_F + g|\nabla \mathbf{n}|_F^2 + \eta \int_\Gamma |\mathbf{n} - \mathbf{n}^*|, \quad \text{with} \quad \nabla \times \mathbf{n} = 0 \right\}, \qquad (3.1)$$

where η is a positive constant, g is an indicator function, $|\cdot|$ is the Euclidean norm in $\mathbf{R}^2$, and $|\cdot|_F$ is the Frobenius norm:

$$|\nabla \mathbf{n}|_F^2 = \left| \begin{pmatrix} \nabla n_1 \\ \nabla n_2 \end{pmatrix} \right|_F^2 \equiv \sum_{j=1}^{2} \sum_{i=1}^{2} \left(\partial_j n_i \right)^2 .$$

Note that the integration domain Γ should include all SOIs and base strokes in jump, bump, and dip strokes. The first term (weighted TV regularization) of $\mathcal{E}_v(\mathbf{n})$ in (3.1) generates nonlinear diffusion. It preserves discontinuities in the vector field such as creases or jumps. The second term (weighted H^1 regularization) yields linear diffusion. That is, it smears out discontinuities in the vector field. The third term is the penalty to preserve $\mathbf{n}^*$ on Γ in the interpolated vector field $\mathbf{n}$.

The curl-free constraint on the interpolated vector $\mathbf{n}$ is not only a necessary and sufficient condition to find a surface I with $\nabla I = \mathbf{n}$ but also gives an extra force to interpolate sparse vectors over a large domain; see inpainting examples in [30]. In Figs. 6 and 7, we illustrate the crucial effect of curl-free constraint which makes a clear difference in practical examples. In Fig. 6-(a), an implicit meaning around T-junctions in drawn strokes on 2D usually represents a hierarchical structure of height field in 3D. That is, our vision system tends to perceive the region A is higher than the region B around T-junctions (red dots) because a part of B is occluded by the region A. It is difficult to observe such structures in (b) because H^1 regularization without curl-free constraint is simply average of vectors on strokes. However, it is easily to see the curl-free constraint reflects an implicit meaning around T-junctions in (a) into a reconstructed surface in (c). In Fig. 7, if we do not impose curl-free constraint, initial vector setting does not effectively affect reconstructed surface; see the comparison between (c) and (e) (or (d) and (f)). The difficulty in applying the curl-free constraint is that the constraint can not be satisfied within all over the domain Ω because it is violated at discontinuities in the vector field. In Subsection 4.1, we propose an algorithm to satisfy the curl-free constraint almost everywhere based on augmented Lagrangian method.



(a) (b) $\nabla \times \mathbf{n} \neq 0$ (c) $\nabla \times \mathbf{n} = 0$

Figure 6: Effect of integrability condition: (a) is strokes drawn by an user. (b) and (c) are results of H^1 regularization without and with curl-free constraint, respectively. It is clear to see that the curl-free constraint reflects an implicit meaning (occlusion) around T-junctions (red dots) in 2D sketch in (a); see details in Subsection 3.1.

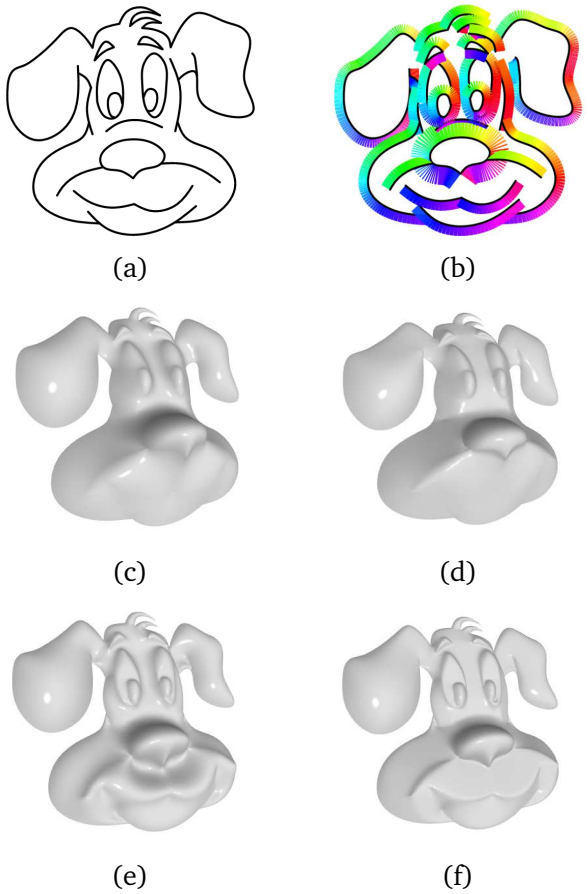
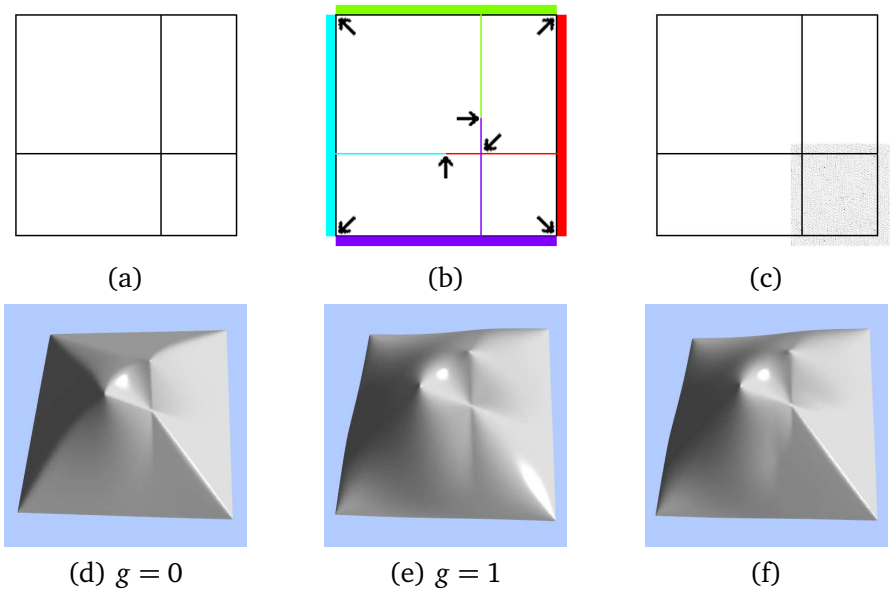


Figure 7: The effect of curl-free constraint: (a) is regular strokes and (b) is an initial vector setting. (c) and (d) are the results from H^1 and TV regularization without curl-free constraint, respectively. (e) and (f) are the results from H^1 and TV regularization with curl-free constraint, respectively. Comparing these results, we can observe that curl-free constraint yields reconstructed surfaces which are more reasonably affected by initial vector setting.

The reason why we combine TV and H^1 regularization in the proposed functional is that the combination can generate various kinds of geometries in reconstructed surfaces. The TV norm preserves discontinuities in an interpolated vector field while the H^1 norm forces smoothness in the vector field; see Figs. 7-(f) or 8-(d) (TV norm) and 7-(e) or 8-(e) (H^1 norm). That is, discontinuities in an assigned vector field on the strokes are well interpolated in the domain under the TV regularization, but they are smeared into the domain under the H^1 regularization. If there is extra information to indicate where such discontinuities are located in Ω, the algorithm in [22] can also preserve the discontinuities. However, it is a very difficult task to find possible locations of discontinuities in a whole domain just from the vectors on strokes. The TV regularization does not need to have extra information to preserve discontinuities in the vector field. Considering the difference between TV and H^1 regularization, it is reasonable to set the indicator function $g = 0$ on



(a) (b) (c)

(d) $g = 0$ (e) $g = 1$ (f)

Figure 8: The role of indicator function g in (3.1): (a) is regular strokes and (b) is the vector setting on the strokes. Direction of assigned vectors on the cross shape in the rectangle is parallel to stroke tangent. In (c), $g = 0$ in the textured region and $g = 1$ elsewhere. (f) shows a combination of (d) and (e) according to the indicator function g in (c).

jump, ridge, valley, bump, and dip strokes. Moreover, if artists want to preserve discontinuities of surface gradients in a specific region, it is simply achieved by setting $g = 0$ in the region; see Figs. 8-(c) and 10-(a).

In Fig. 8, we illustrate an example shown in [31] to show clear difference between TV and H^1 regularization. In this paper, we demonstrate that the combination of them generates reasonable surface reconstruction to merge two different surface structures. (a) is regular strokes and (b) is the initial vector setting on the strokes. Note that direction of assigned vectors on the cross shape in the rectangle is parallel to stroke tangent. The initial vector field has seven discontinuous points indicated by the arrows in (b). (d) is the result of TV regularization ($g = 0$ in (3.1)). As we expected, the discontinuities in the initial vector field are well interpolated and it is shown as sharp ridges in the reconstructed surface. (e) is the result of H^1 regularization ($g = 1$ in (3.1)) which yields a smooth surface. By setting $g = 0$ in the textured region and $g = 1$ elsewhere in (c), we can obtain a reasonable combination of (d) and (e) as shown in (f). The part of surface in (f), which corresponds to the textured region in (c), is similar to the surface in (d). The rest of the surface in (f) is similar to the shape in (e).

3.2. Height map reconstruction

After the dense normal vector field $\mathbf{n}$ is obtained by the proposed nonlinear vector interpolation (3.1), the height map is reconstructed via the minimization of energy func-

tional:

$$\min_I \left\{ \mathscr{E}_h(I) = \int_\Omega (1-h)|\nabla I - \mathbf{n}| + \int_\Omega h|\nabla I - \mathbf{n}|^2 + \xi \int_\Sigma |I - I_0| \right\}, \tag{3.2}$$

where I_0 is known height values on $\Sigma \subset \Omega$ and h is an indicator function. If there is no prior height information, we use $\xi = 0$.

The major difference between TV type norm $\int |\nabla I - \mathbf{n}|$ and H^1 type norm $\int |\nabla I - \mathbf{n}|^2$ is whether the jump discontinuities are allowed in the reconstructed height map or not. The TV type norm reconstructs the jump discontinuities without surface distortion. However, the H^1 type norm enforces the C^0 continuity of the height map because the second term in (3.2) yields the Laplace operator in the Euler-Lagrange equation. If we apply the H^1 type norm to a whole domain, a reconstructed height map easily encounter overshooting or undershooting problem on the jump strokes. Considering the different effects from TV and H^1 type norm, the value of the indicator function h is 0 on the jump strokes and 1 elsewhere. Since the jump strokes are known as prior information, the settings of h is automatically determined.

More importantly, when there is no prior height information and no given jump strokes, $h \equiv 1$ on Ω is chosen and the proposed functional in (3.2) simply has H^1 type norm. In this case, a solution of Poisson equation is a reconstructed height map whose gradient fits $\mathbf{n}$. It is justified by the curl-free constraint imposed in (3.1).

4. Proposed numerical solvers

We propose an algorithm for efficiently solving minimization of energy function (3.1) in Subsection 4.1 and (3.2) in Subsection 4.2. Since the algorithms for (3.1) and (3.2) are very similar, it is enough to explain the proposed algorithm for (3.1) in detail.

4.1. Minimization of (3.1)

First of all, using a variable splitting method with two variables $\mathbf{P}$ and $\mathbf{s}$, we change (3.1) into a constraint minimization problem:

$$\min_{\mathbf{n}} \int_\Omega (1-g)|\mathbf{P}|_F + \int_\Omega g|\mathbf{P}|_F^2 + \eta \int_\Gamma |\mathbf{s} - \mathbf{n}^*|,$$
$$\text{with } \mathbf{P} = \nabla \mathbf{n}, \quad \mathbf{s} = \mathbf{n}, \quad \text{and } \nabla \times \mathbf{n} = 0.$$

In order to solve the above constraint minimization, we introduce the augmented Lagrangian functional $\mathscr{L}_v(\mathbf{s}, \mathbf{n}, \mathbf{P}; \boldsymbol{\lambda}_f, \lambda_c, \boldsymbol{\Lambda}_r)$:

$$\begin{aligned}\mathscr{L}_v(\cdot) \equiv \int_\Omega (1-g)|\mathbf{P}|_F + g|\mathbf{P}|_F^2 + \eta \int_\Gamma |\mathbf{s} - \mathbf{n}^*| + \int_\Omega \boldsymbol{\Lambda}_r \cdot (\mathbf{P} - \nabla \mathbf{n}) + \frac{c_r}{2} \int_\Omega |\mathbf{P} - \nabla \mathbf{n}|_F^2 \\ + \int_\Omega \boldsymbol{\lambda}_f \cdot (\mathbf{s} - \mathbf{n}) + \frac{c_f}{2} \int_\Omega |\mathbf{s} - \mathbf{n}|^2 + \int_\Omega \omega_c \lambda_c (\nabla \times \mathbf{n}) + \frac{c_c}{2} \int_\Omega (\nabla \times \mathbf{n})^2, \end{aligned} \tag{4.1}$$

where c_f, c_c, and c_r are positive penalty parameters, $\boldsymbol{\lambda}_f$, λ_c, and $\boldsymbol{\Lambda}_r$ are the Lagrangian multipliers. If $g \equiv 1$ on Ω, it is not necessary to use c_r, $\boldsymbol{\Lambda}_r$, and $\mathbf{P}$ and the algorithm becomes much simpler.

The weight function ω_c is automatically chosen depending on the geometrical structure on given strokes. That is, $\omega_c = 0$ is used on ridge, valley, bump, dip, and jump strokes and $\omega_c = 1$ elsewhere because the interpolated vector field $\mathbf{n}$ violates the curl-free constraint at known discontinuous points in surface height and its gradient. If crease structures (which violate the curl-free constraint) are generated in the region with $\omega_c = 1$, TV regularization prevents surface distortion; see Fig. 8-(d). One may think that weight function ω_c may not be crucially necessary because of TV regularization. However, the augmented Lagrangian method (4.1) without ω_c makes a problem of choosing substantially small penalty parameter c_c for examples with strokes which violate the curl-free constraint. The small c_c causes slow convergence and it is not practically useful. Note that we apply the Gaussian linear filtering with scale parameter 2 to obtain smooth ω_c, g in (3.1), and h in (3.2).

Algorithm 4.1: Augmented Lagrangian method for (4.1)

---

1. Initialization: $\mathbf{s}^0$, $\mathbf{n}^0$, $\mathbf{P}^0$, $\boldsymbol{\lambda}_f^0$, λ_c^0, and $\boldsymbol{\Lambda}_r^0$.
2. For $k \geq 1$, compute an approximate minimizer $\left(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k\right)$ of the augmented Lagrangian functional with the fixed Lagrange multipliers $\boldsymbol{\lambda}_f^{k-1}$, λ_c^{k-1}, and $\boldsymbol{\Lambda}_r^{k-1}$:

$$\left(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k\right) \approx \arg\min \mathscr{L}_v\left(\mathbf{s}, \mathbf{n}, \mathbf{P}; \boldsymbol{\lambda}_f^{k-1}, \lambda_c^{k-1}, \boldsymbol{\Lambda}_r^{k-1}\right). \tag{4.2}$$

3. Update Lagrange multipliers

$$\boldsymbol{\lambda}_f^k = \boldsymbol{\lambda}_f^{k-1} + c_f\left(\mathbf{s}^k - \mathbf{n}^k\right), \tag{4.3}$$

$$\lambda_c^k = \lambda_c^{k-1} + \frac{c_c}{\omega_c}\left(\nabla \times \mathbf{n}^k\right), \tag{4.4}$$

$$\boldsymbol{\Lambda}_r^k = \boldsymbol{\Lambda}_r^{k-1} + c_r\left(\mathbf{P}^k - \nabla \mathbf{n}^k\right). \tag{4.5}$$

   Note that we use $\lambda_c^k = \lambda_c^{k-1}$ if ω_c is zero.

4. Measure the relative residuals and go to Step 2 unless they are larger than an error bound ϵ_1.

An iterative procedure in Algorithm 4.1 is used to find the saddle point of the augmented Lagrangian functional (4.1) via maximizing the Lagrangian multipliers and minimizing variables $\mathbf{s}$, $\mathbf{n}$, and $\mathbf{P}$. We initialize $\mathbf{s}^0$, $\mathbf{n}^0$, $\mathbf{P}^0$, $\boldsymbol{\lambda}_f^0$, λ_c^0, and $\boldsymbol{\Lambda}_r^0$ as zero. For $k \geq 1$, an alternating minimization method is used to approximately find a minimizer $(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k)$ of the functional $\mathscr{L}_v(\cdot : \boldsymbol{\lambda}_f^{k-1}, \lambda_c^{k-1}, \boldsymbol{\Lambda}_r^{k-1})$ with the previous variables $\mathbf{s}^{k-1}$, $\mathbf{n}^{k-1}$, and $\mathbf{P}^{k-1}$.

The detailed algorithm for alternating minimization is given in Algorithm 4.2. First of all, we initialize the variables: $\tilde{\mathbf{s}}^0 = \mathbf{s}^{k-1}$, $\tilde{\mathbf{n}}^0 = \mathbf{n}^{k-1}$, and $\tilde{\mathbf{P}}^0 = \mathbf{P}^{k-1}$. For $l = 1, \cdots, L$, we

Algorithm 4.2: Alternating minimization method to solve Eq. (4.2) in Algorithm 4.1

1. Initialization: $\tilde{\mathbf{s}}^0 = \mathbf{s}^{k-1}$, $\tilde{\mathbf{n}}^0 = \mathbf{n}^{k-1}$, and $\tilde{\mathbf{P}}^0 = \mathbf{P}^{k-1}$.
2. For $l = 1, \cdots, L$ and fixed Lagrange multipliers $\boldsymbol{\lambda}_f = \boldsymbol{\lambda}_f^{k-1}$, $\lambda_c = \lambda_c^{k-1}$, and $\boldsymbol{\Lambda}_r = \boldsymbol{\Lambda}_r^{k-1}$, solve the following minimization problems alternatively:

$$\tilde{\mathbf{s}}^l = \arg\min \mathscr{L}_v \left(\mathbf{s}, \tilde{\mathbf{n}}^{l-1}, \tilde{\mathbf{P}}^{l-1}; \boldsymbol{\lambda}_f, \lambda_c, \boldsymbol{\Lambda}_r\right), \tag{4.6}$$

$$\tilde{\mathbf{n}}^l = \arg\min \mathscr{L}_v \left(\tilde{\mathbf{s}}^l, \mathbf{n}, \tilde{\mathbf{P}}^{l-1}; \boldsymbol{\lambda}_f, \lambda_c, \boldsymbol{\Lambda}_r\right), \tag{4.7}$$

$$\tilde{\mathbf{P}}^l = \arg\min \mathscr{L}_v \left(\mathbf{s}^l, \tilde{\mathbf{n}}^l, \mathbf{P}; \boldsymbol{\lambda}_f, \lambda_c, \boldsymbol{\Lambda}_r\right). \tag{4.8}$$

3. $\left(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k\right) = \left(\tilde{\mathbf{s}}^L, \tilde{\mathbf{n}}^L, \tilde{\mathbf{P}}^L\right)$.

find minimizers $\tilde{\mathbf{s}}^l$, $\tilde{\mathbf{n}}^l$, and $\tilde{\mathbf{P}}^l$ in the subproblems from (4.6) to (4.8) by minimizing the energy functionals

$$\mathscr{E}_1(\mathbf{s}) = \int_\Omega \frac{c_f}{2}|\mathbf{s} - \tilde{\mathbf{n}}^{l-1}|^2 + \boldsymbol{\lambda}_f \cdot \mathbf{s} + \eta \int_\Gamma |\mathbf{s} - \mathbf{n}^*|, \tag{4.9}$$

$$\begin{aligned}\mathscr{E}_2(\mathbf{n}) = \int_\Omega \frac{c_r}{2}|\tilde{\mathbf{P}}^{l-1} - \nabla\mathbf{n}|_F^2 - \boldsymbol{\Lambda}_r \cdot \nabla\mathbf{n} + \frac{c_c}{2}(\nabla \times \mathbf{n})^2 \\ + \omega_c \lambda_c (\nabla \times \mathbf{n}) + \frac{c_f}{2}|\tilde{\mathbf{s}}^l - \mathbf{n}|^2 - \boldsymbol{\lambda}_f \cdot \mathbf{n},\end{aligned} \tag{4.10}$$

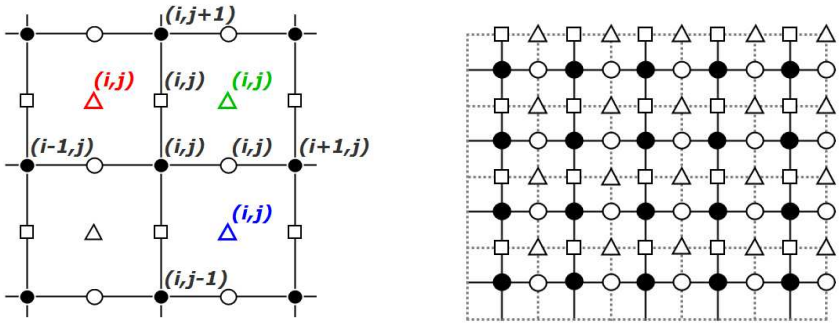
$$\mathscr{E}_3(\mathbf{P}) = \int_\Omega (1-g)|\mathbf{P}|_F + g|\mathbf{P}|_F^2 + \frac{c_r}{2}|\mathbf{P} - \nabla\tilde{\mathbf{n}}^l|_F^2 + \boldsymbol{\Lambda}_r \cdot \mathbf{P}. \tag{4.11}$$

After L^{th} iteration, $\mathbf{s}^k = \tilde{\mathbf{s}}^L$, $\mathbf{n}^k = \tilde{\mathbf{n}}^L$, and $\mathbf{P}^k = \tilde{\mathbf{P}}^L$ are updated. Note that we numerically observe that $L = 1$ is enough to obtain desirable results. Before we explain more details of each minimization, we would like to add the following comments:

- The minimization of functionals (4.9) and (4.11), c.f. (4.6) and (4.8) in Algorithm 4.2, can be solved by soft thresholding method [28] which requires a simple arithmetic computation.
- The minimization of functional (4.10), c.f. (4.7) in Algorithm 4.2, is terminated by solving a system of linear equations over the whole domain Ω. Since a grid is uniform, FFT can be used to solve equations with a very low computational cost.

Note that there are a lot of fast and efficient methods to solve energy minimization with TV regularization. For interested readers, please refer to [32–35] and references therein.

In the rest of this subsection, we describe the details of the implementation for minimizing the given functionals in (4.9), (4.10), and (4.11). Especially, we shall present the details in the staggered grid system in Fig. 9. The variables, $\mathbf{s}$, $\mathbf{n}$, $\mathbf{P}$, $\boldsymbol{\lambda}_f$, λ_c, and $\boldsymbol{\Lambda}_r$ in the



(a) (b)

Figure 9: (a) is the rule of indexing variables, $\mathbf{s}$, $\mathbf{n}$, $\mathbf{P}$, $\boldsymbol{\lambda}_f$, λ_c, and $\boldsymbol{\Lambda}_r$ in the augmented Lagrangian functional (4.1). (b) is an example of discrete computational domain whose size is 5×4.

augmented Lagrangian functional (4.1) are defined on a discrete computational domain $\Omega = [1, N_1] \times [1, N_2]$:

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}, \quad \mathbf{n} = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}, \quad \boldsymbol{\lambda}_f = \begin{pmatrix} \lambda_{f1} \\ \lambda_{f2} \end{pmatrix},$$
$$\mathbf{P} = \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}, \quad \text{and} \quad \boldsymbol{\Lambda}_r = \begin{pmatrix} \boldsymbol{\lambda}_{r1} \\ \boldsymbol{\lambda}_{r2} \end{pmatrix} = \begin{pmatrix} \lambda_{r11} & \lambda_{r12} \\ \lambda_{r21} & \lambda_{r22} \end{pmatrix}.$$

In the staggered grid system, we use physically different locations to evaluate a value of variables. More precisely, the first and second components of $\mathbf{n}$, $\mathbf{s}$, and $\boldsymbol{\lambda}_f$ are defined at $\square$ and $\circ$, respectively, in Fig. 9-(a) and p_{12}, p_{21}, λ_{r12}, and λ_{r21} are defined at $\bullet$. The other variables are defined at $\triangle$, but the coordinate (i, j) indicates different position. More specifically, $\lambda_c(i, j)$ is at the green triangle, $p_{11}(i, j)$ and $\lambda_{r11}(i, j)$ are at the red triangle, and $p_{22}(i, j)$ and $\lambda_{r22}(i, j)$ are at the blue triangle. These rules of indexing become more reasonable when we discretize the Euler-Lagrange equations for (4.10). The periodic boundary condition is applied to all variables. An example whose discrete domain is $[1, 5] \times [1, 4]$ is shown in Fig. 9-(b).

4.1.1. Minimization of $\mathscr{E}_1(\mathbf{s})$ in (4.9)

Denoting $\mathbf{n} = \tilde{\mathbf{n}}^{l-1}$, we represent the functional in $\mathscr{E}_1(\mathbf{s})$ as two parts:

$$\mathscr{E}_1(\mathbf{s}) = \mathscr{E}_{\Omega\setminus\Gamma}(\mathbf{s}) + \mathscr{E}_\Gamma(\mathbf{s}),$$

where

$$\mathscr{E}_{\Omega\setminus\Gamma}(\mathbf{s}) \equiv \int_{\Omega\setminus\Gamma} \boldsymbol{\lambda}_f \cdot \mathbf{s} + \frac{c_f}{2}|\mathbf{s} - \mathbf{n}|^2,$$
$$\mathscr{E}_\Gamma(\mathbf{s}) \equiv \int_\Gamma \eta|\mathbf{s} - \mathbf{n}^*| + \boldsymbol{\lambda}_f \cdot \mathbf{s} + \frac{c_f}{2}|\mathbf{s} - \mathbf{n}^*|^2.$$

The minimizer in the first energy functional $\mathscr{E}_{\Omega\setminus\Gamma}(\cdot)$ is easily obtained because the integrand is a quadratic polynomial in terms of $\mathbf{s}$. For the second energy functional $\mathscr{E}_{\Gamma}(\cdot)$, we reformulate it as follows:

$$\mathscr{E}_{\Gamma}(\mathbf{s}) = \int_{\Gamma} \eta|\mathbf{s}-\mathbf{n}^*| + \frac{c_f}{2}\left|\mathbf{s}-\mathbf{n}^* + \frac{\boldsymbol{\lambda}_f}{c_f}\right|^2 + C,$$

where C does not count on the minimization. For each coordinate $(i,j)\in\Gamma$, we use the soft thresholding method in [28] and a minimizer $\tilde{\mathbf{s}}^l$ in the problem (4.6) is obtained by

$$\begin{aligned}(i,j)\notin\Gamma \quad &\Rightarrow \quad \tilde{\mathbf{s}}^l(i,j) = \mathbf{n}(i,j) - \frac{1}{c_f}\lambda_f(i,j),\\ (i,j)\in\Gamma \quad &\Rightarrow \quad \tilde{\mathbf{s}}^l(i,j) = \mathbf{n}^*(i,j) + \alpha(i,j)\mathbf{x}_0(i,j),\end{aligned}$$

where $\alpha = \max\left(0, 1-\frac{\eta}{c_f|\mathbf{x}_0|}\right)$ and $\mathbf{x}_0 = \mathbf{n}^* - \frac{\boldsymbol{\lambda}_f}{c_f}$.

4.1.2. Minimization of $\mathscr{E}_2(\mathbf{n})$ in (4.10)

For fixed $\mathbf{P} = \tilde{\mathbf{P}}^{l-1}$ and $\mathbf{s} = \tilde{\mathbf{s}}^l$, the Euler-Lagrange equation of (4.10) yields a system of linear PDEs:

$$-\left(c_r\mathscr{D}_r + c_c\mathscr{D}_c - c_f\mathscr{I}\right)\mathbf{n} = \nabla\cdot\boldsymbol{\Lambda}_r - \nabla^{\perp}(\omega_c\lambda_c) + \boldsymbol{\lambda}_f - \nabla\cdot\mathbf{P} + c_f\mathbf{s}, \tag{4.12}$$

where $\nabla^{\perp} = (-\partial_2, \partial_1)^{\mathrm{T}}$, $\nabla\cdot\mathbf{P} = (\nabla\cdot\mathbf{p}_1, \nabla\cdot\mathbf{p}_2)^{\mathrm{T}}$, $\mathscr{I}$ is a 2×2 identity matrix, $\mathscr{D}_r = (\partial_1^2+\partial_2^2)\mathscr{I}$, and $\mathscr{D}_c = \nabla^{\perp}\nabla^{\perp\mathrm{T}}$. The componentwise expression of (4.12) is as follows:

$$\begin{aligned}&-\left(c_r\begin{pmatrix}\partial_1^2+\partial_2^2 & 0\\ 0 & \partial_1^2+\partial_2^2\end{pmatrix} + c_c\begin{pmatrix}\partial_2^2 & -\partial_2\partial_1\\ -\partial_1\partial_2 & \partial_1^2\end{pmatrix} - c_f\begin{pmatrix}1&0\\0&1\end{pmatrix}\right)\begin{pmatrix}n_1\\n_2\end{pmatrix}\\ &= \begin{pmatrix}\nabla\cdot\boldsymbol{\lambda}_{r1}\\ \nabla\cdot\boldsymbol{\lambda}_{r2}\end{pmatrix} - \begin{pmatrix}-\partial_2(\omega_c\lambda_c)\\ \partial_1(\omega_c\lambda_c)\end{pmatrix} + \begin{pmatrix}\lambda_{f1}\\ \lambda_{f2}\end{pmatrix} - \begin{pmatrix}\nabla\cdot\mathbf{p}_1\\ \nabla\cdot\mathbf{p}_2\end{pmatrix} + c_f\begin{pmatrix}s_1\\ s_2\end{pmatrix}.\end{aligned} \tag{4.13}$$

The operator $\mathscr{D}_r$ is componentwise Laplacian to generates diffusion of $\mathbf{n}^*$ on the strokes. The differential operator $\mathscr{D}_c$ obtained by curl-free constraint makes (4.12) to be coupled equations. It generates an extra force to interpolate $\mathbf{n}^*$ into the whole domain.

From the standard finite difference scheme with the rule of indexing variables in Fig. 9, we discretize (4.13) at different nodes: n_1 at □ and n_2 at ∘:

$$\begin{aligned}&-c_r\partial_1^+\partial_1^- n_1 - (c_r+c_c)\partial_2^+\partial_2^- n_1 + c_c\partial_1^-\partial_2^+ n_2 + c_f n_1\\ &= -\partial_1^+\lambda_{r11} - \partial_2^+\lambda_{r12} - \partial_2^-\left(\omega_c\lambda_c\right) + \lambda_{f1} - c_r\partial_1^+ p_{11} - c_r\partial_2^+ p_{12} + c_f s_1,\end{aligned} \tag{4.14}$$

$$\begin{aligned}&-c_r\partial_2^+\partial_2^- n_2 - (c_r+c_c)\partial_1^+\partial_1^- n_2 + c_c\partial_2^-\partial_1^+ n_1 + c_f n_2\\ &= -\partial_1^+\lambda_{r21} - \partial_2^+\lambda_{r22} + \partial_1^-\left(\omega_c\lambda_c\right) + \lambda_{f2} - c_r\partial_1^+ p_{21} - c_r\partial_2^+ p_{22} + c_f s_2,\end{aligned} \tag{4.15}$$

where ∂_i^+ and ∂_i^- are a standard finite forward and backward difference operators, respectively. Note that $\partial_1^-\partial_2^+ = \partial_2^+\partial_1^-$ in (4.14) and $\partial_2^-\partial_1^+ = \partial_1^+\partial_2^-$ in (4.15). Now, introducing the identity operator $\mathscr{I}f(i,j) = f(i,j)$ and shifting operators,

$$\mathscr{S}_1^{\pm}f(i,j) = f(i\pm 1, j) \quad \text{and} \quad \mathscr{S}_2^{\pm}f(i,j) = f(i, j\pm 1),$$

the discretization of (4.14) and (4.15) is written:

$$\begin{aligned}\left(-c_r\left(\mathscr{S}_1^+ - 2\mathscr{I} + \mathscr{S}_1^-\right) - (c_r + c_c)\left(\mathscr{S}_2^+ - 2\mathscr{I} + \mathscr{S}_2^-\right) + c_f\mathscr{I}\right) n_1(i,j) \\ + c_c\left(\mathscr{S}_2^+ - \mathscr{I} - \mathscr{S}_1^-\mathscr{S}_2^+ + \mathscr{S}_1^-\right) n_2(i,j) = f_1(i,j),\end{aligned} \tag{4.16}$$

with

$$\begin{aligned}f_1(i,j) = &-\left[\left(\mathscr{S}_1^+ - \mathscr{I}\right)(\lambda_{r11} + c_r p_{11}) + \left(\mathscr{S}_2^+ - \mathscr{I}\right)(\lambda_{r12} + c_r p_{12})\right](i,j) \\ &+\left[\left(\mathscr{I} - \mathscr{S}_2^+\right)(\omega_c\lambda_c) + \lambda_{f1} + c_f s_1\right](i,j);\end{aligned}$$

and

$$\begin{aligned}\left(-c_r\left(\mathscr{S}_2^+ - 2\mathscr{I} + \mathscr{S}_2^-\right) - (c_r + c_c)\left(\mathscr{S}_1^+ - 2\mathscr{I} + \mathscr{S}_1^-\right) + c_f\mathscr{I}\right) n_2(i,j) \\ + c_c\left(\mathscr{S}_1^+ - \mathscr{S}_1^+\mathscr{S}_2^- - \mathscr{I} + \mathscr{S}_2^-\right) n_1(i,j) = f_2(i,j),\end{aligned} \tag{4.17}$$

with

$$\begin{aligned}f_2(i,j) = &-\left[\left(\mathscr{S}_2^+ - \mathscr{I}\right)(\lambda_{r22} + c_r p_{22}) + \left(\mathscr{S}_1^+ - \mathscr{I}\right)(\lambda_{r21} + c_r p_{21})\right](i,j) \\ &+\left[\left(\mathscr{S}_1^+ - \mathscr{I}\right)(\omega_c\lambda_c) + \lambda_{f2} + c_f s_1\right](i,j).\end{aligned}$$

Adopting the periodic boundary condition to all variables, we apply the discrete Fourier transform $\mathscr{F}$ to solve the discretization of (4.12). The shifting operators represented by finite difference are essentially a discrete convolution and then its discrete Fourier transform is the componentwise multiplication in the frequency domain. For discrete frequencies, u_i and u_j, we have

$$\mathscr{F}\mathscr{S}_1^{\pm}f(u_i,u_j) = e^{\pm\sqrt{-1}v_i}\mathscr{F}f(u_i,u_j), \qquad \mathscr{F}\mathscr{S}_2^{\pm}f(u_i,u_j) = e^{\pm\sqrt{-1}v_j}\mathscr{F}f(u_i,u_j),$$

where

$$v_i = \frac{2\pi}{N_1}u_i, \quad u_i = 1,\cdots,N_1, \qquad \text{and} \qquad v_j = \frac{2\pi}{N_2}u_j, \quad u_j = 1,\cdots,N_2.$$

It yields a system of linear equations:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} \mathscr{F}n_1(u_i,u_j) \\ \mathscr{F}n_2(u_i,u_j) \end{pmatrix} = \begin{pmatrix} f_1(u_i,u_j) \\ f_2(u_i,u_j) \end{pmatrix}.$$

The coefficients in the system of linear equations are

$$a_{11} = 4c_r \sin^2\frac{v_i}{2} + 4(c_r + c_c)\sin^2\frac{v_j}{2} + c_f,$$
$$a_{12} = c_c(-\bar{\zeta}_j)\zeta_i, \qquad a_{21} = c_c(-\bar{\zeta}_i)\zeta_j,$$
$$a_{22} = 4(c_r + c_c)\sin^2\frac{v_i}{2} + 4c_r \sin^2\frac{v_j}{2} + c_f,$$

where

$$\zeta_i = 2\sin^2\frac{v_i}{2} + \sqrt{-1}\sin v_i, \qquad \zeta_j = 2\sin^2\frac{v_j}{2} + \sqrt{-1}\sin v_j,$$

and the right hand side of linear equations are

$$f_1 = \bar{\zeta}_j \mathscr{F}\alpha_{12}^1 + \bar{\zeta}_i \mathscr{F}\alpha_{11}^2 - \mathscr{F}\alpha_1^3,$$
$$f_2 = \bar{\zeta}_j \mathscr{F}\alpha_{21}^1 + \bar{\zeta}_i \mathscr{F}\alpha_{22}^2 - \mathscr{F}\alpha_2^3,$$

where

$$\alpha_{mn}^1 \equiv \lambda_{rmn} + c_r(p_{mn} - p_{nm}) + (-1)^m \omega_c \lambda_c,$$
$$\alpha_{mn}^2 \equiv \lambda_{rmn} + c_r p_{mn}, \qquad \alpha_m^3 \equiv \lambda_{fm} + c_f s_m.$$

Now, we have $N_1 N_2$ numbers of 2×2 systems. The determinant of the coefficients matrix in the above equations for all discrete frequencies is

$$(4c_r\beta_{ij} + c_f)(4(c_r + c_c)\beta_{ij} + c_f),$$

where $\beta_{ij} = \sin^2\frac{v_i}{2} + \sin^2\frac{v_j}{2}$, not zero because the penalty parameters c_r, c_c, and c_f are positive. After the systems of linear equations are solved for each frequency, the discrete inverse Fourier transform is used to obtain $\tilde{\mathbf{n}}^l$.

4.1.3. Minimization of $\mathscr{E}_3(\mathbf{p})$ in (4.11)

Denoting $\mathbf{n} = \tilde{\mathbf{n}}^l$, the functional (4.11) is reformulated:

$$\mathscr{E}_3(\mathbf{P}) = \int_\Omega (1-g)|\mathbf{P}|_F + \frac{c_r + 2g}{2}\left|\mathbf{P} - \frac{1}{c_r + 2g}\left(c_r\nabla\mathbf{n} - \mathbf{\Lambda}_r\right)\right|_F^2 + C,$$

where C does not count on the minimization. We apply the same approach in [23] to find the closed form of the minimizer at each point $(i,j) \in \Omega$:

$$\mathbf{p}(i,j) = \max\left\{0, 1 - \frac{1 - g(i,j)}{c_r|\mathbf{W}(i,j)|_F}\right\}\mathbf{W}(i,j),$$

where

$$\mathbf{W} = \frac{c_r}{c_r + 2g}\left(c_r\nabla\mathbf{n} - \boldsymbol{\lambda}_r\right).$$

4.2. Minimization of (3.2)

Using the same method in Subsection 4.1, the minimization problem (3.2) is also efficiently solved by augmented Lagrangian method. We change (3.2) into a constraint minimization problem with introducing new variables $\mathbf{q}$ and J:

$$\min_I \int_\Omega (1-h)|\mathbf{q}-\mathbf{n}| + \int_\Omega h|\mathbf{q}-\mathbf{n}|^2 + \xi \int_\Sigma |J-I_0|, \quad \text{with} \quad \mathbf{q}=\nabla I \quad \text{and} \quad J=I.$$

Now, we cast the problem into unconstraint problem and find a saddle point of the augmented Lagrangian functional:

$$\begin{aligned}\mathscr{L}_h(\cdot) \equiv & \int_\Omega (1-h)|\mathbf{q}-\mathbf{n}| + h|\mathbf{q}-\mathbf{n}|^2 + \int_\Omega \boldsymbol{\mu}_r \cdot (\mathbf{q}-\nabla I) \\ & + \frac{d_r}{2}\int_\Omega |\mathbf{q}-\nabla I|^2 + \int_\Omega \mu_f (J-I) + \frac{d_f}{2}\int_\Omega (J-I)^2 + \xi \int_\Sigma |I-I_0|, \end{aligned} \tag{4.18}$$

where d_r and d_f are positive penalty parameters and $\boldsymbol{\mu}_r$ and μ_f are Lagrange multipliers. The iterative algorithm to find the optimality condition for (4.18) is almost similar to Algorithms 4.1 and 4.2; see [31] for discretization in detail.

In case of $\xi = 0$ and $h = 1$ on Ω, which happens in most common examples, we do not need to use an iterative algorithm to solve (3.2) since the Euler-Lagrange equation for I is simply Poisson equation:

$$\nabla \cdot (\nabla I) = \nabla \cdot \mathbf{n}. \tag{4.19}$$

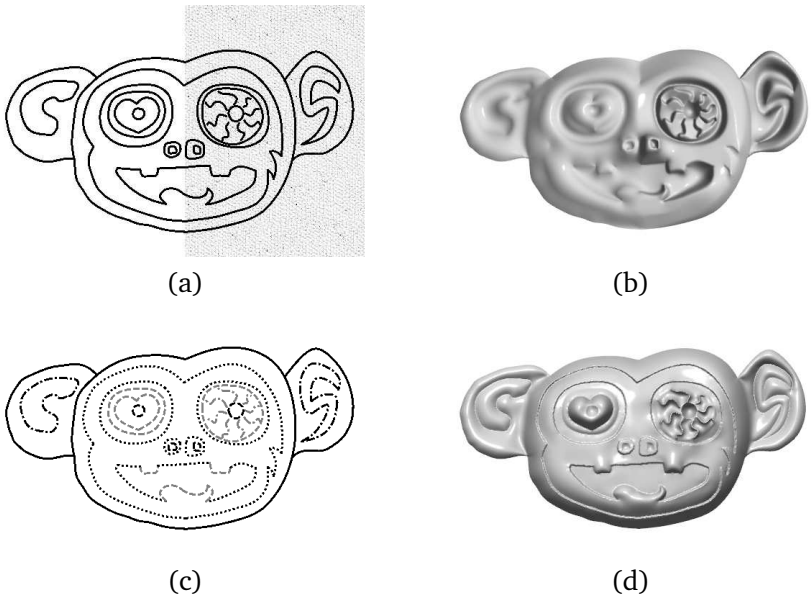
The main reason why surface distortion is eliminated in the solution of the equation (4.19) is that the interpolated vector field $\mathbf{n}$ satisfies with the integrability condition.

In case of $h \neq 1$ on Ω or $\xi \neq 0$, the iterative algorithm is necessary to efficiently solve (3.2). Moreover, if there are jump strokes, which is the case of $h \neq 1$, the TV type norm prevents a height map from having overshooting or undershooting problem on jump strokes.

5. Numerical examples

In this section, we demonstrate the advantage of our proposed method and compare with other related works. The computational cost is also shown to prove the applicability of our method for practical modeling tasks. Moreover, stopping criterion and parameter settings in the algorithm are explained.

Geometry Control: Fig. 1 shows an example created by our proposed method using only regular strokes and default initial vector settings. For more complicated surfaces, users can construct the desired geometry with various stroke types and the indicator function g in (3.1). When artists would like to create a surface which has discontinuities in its



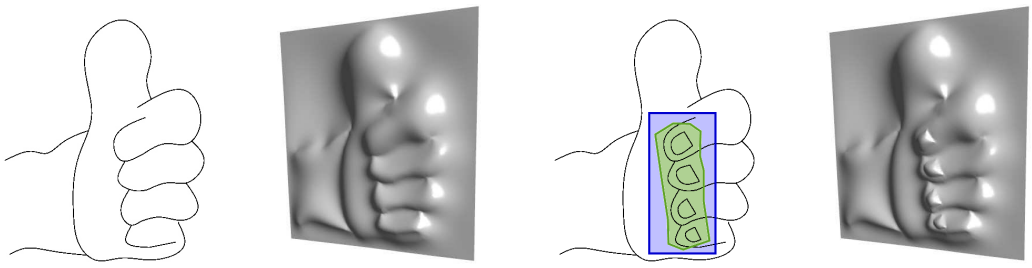
(a) (b)

(c) (d)

Figure 10: Monkey example: (a) and (c) are the same drawing by an artist. In (a), the indicator function g is set as 0 in the textured region and 1 elsewhere. In (c), the artist uses the different types of strokes. (b) and (d) are the reconstructed surfaces from (a) and (c), respectively.

gradient, they simply need to select textured regions; see the textured region in Fig. 10-(b). It is a much simpler task than exactly indicating where gradient discontinuities are located in the domain, which is described in [22]. In Fig. 10, a monkey example drawn by an artist is shown in (a). An artistic intention, which is to obtain sharp surface on the right and smooth surface on the left, can be expressed by selecting the right side (textured region) indicator function $g = 0$ while the left side (white region) $g = 1$. The reconstructed surface in (b) illustrates different geometrical structures with sharp (right) and smooth (left) appearance. An alternative way to create a surface with gradient or jump discontinuities is to use various types of strokes as in (c). The reconstructed surface in (d) preserves discontinuities without any distortion. It clearly shows the intended surface from stroke types.

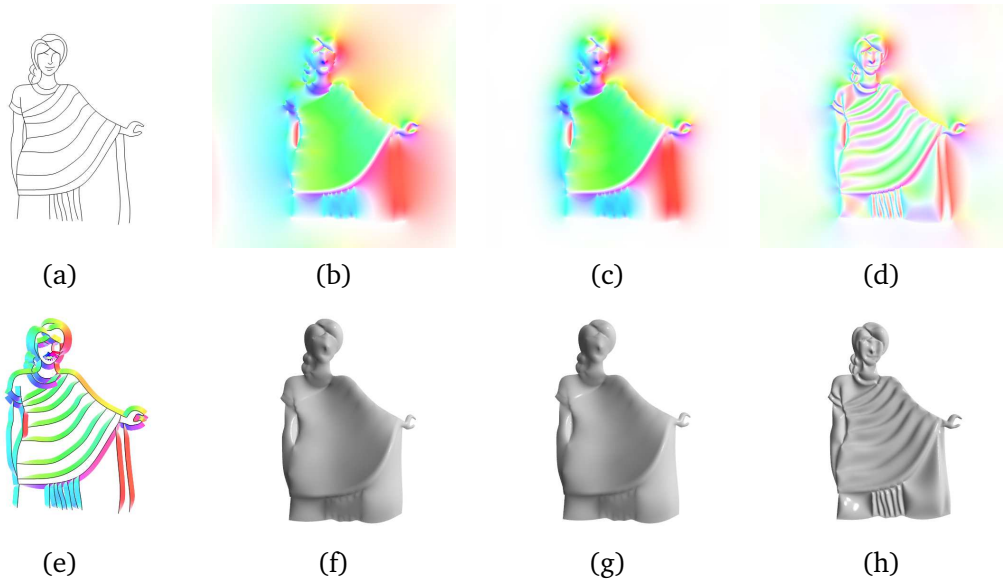
Local Editing: Local editing is necessary for an efficient and real-time modification on the surface which is already reconstructed. Although we use an efficient algorithm, the computational cost is not in real time; see Table 1. Local editing is thus supported to achieve efficient surface modification. In Fig. 11-(c), we can easily select new computational domain shown as the green region. A bounding blue rectangle is automatically detected. We set Ω as the domain enclosed by the blue rectangle and Γ as the blue region and new strokes. Then, (3.1) and (3.2) are computed based on new Ω and Γ. Note that we need to choose Σ as the blue region and utilize I_0 as the height information on Σ. In our numerical experience, if the domain size is 64^2, the whole procedure takes less than 1 second.



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

Figure 11: Local editing: If several missing strokes in (a) are added, we do not need to compute the whole procedure. The local domain is selected as green and the local changes are reflected without changing the previous geometry. (b) and (d) are the reconstructed height maps from (a) and (c), respectively.

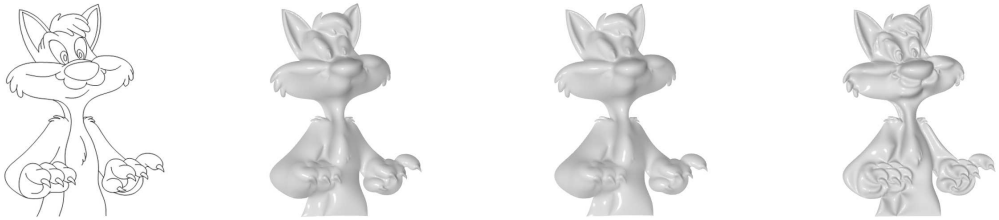
Comparisons: We demonstrate the differences between our proposed method and closely related works such as LUMO [25] and ShapePalettes [20]. To make a fair comparison, we use the results after 500 iterations for all methods in order to confirm that all results are converged. We ask an artist to trace the drawing as Fig. 12-(a), which is shown in [36], and set the initial vectors as (e). The dense vector fields and reconstructed surfaces via LUMO, ShapePalettes, and our method are shown in (b) and (f), (c) and (g), and (d) and (h), respectively. In Figs. 12-(f) and (g), the ripples on the clothes are smoothed out because of linearity and lack of curl-free constraint. On the other hand, our method



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

(e) (f) (g) (h)

Figure 12: Indian lady example: (a) is the line drawing. (e) is the assigned vectors on the strokes in (a). (b) and (f), (c) and (g), and (d) and (h) are the dense vector field and reconstructed surface using the LUMO [25], ShapePalettes [20], and our proposed method, respectively.

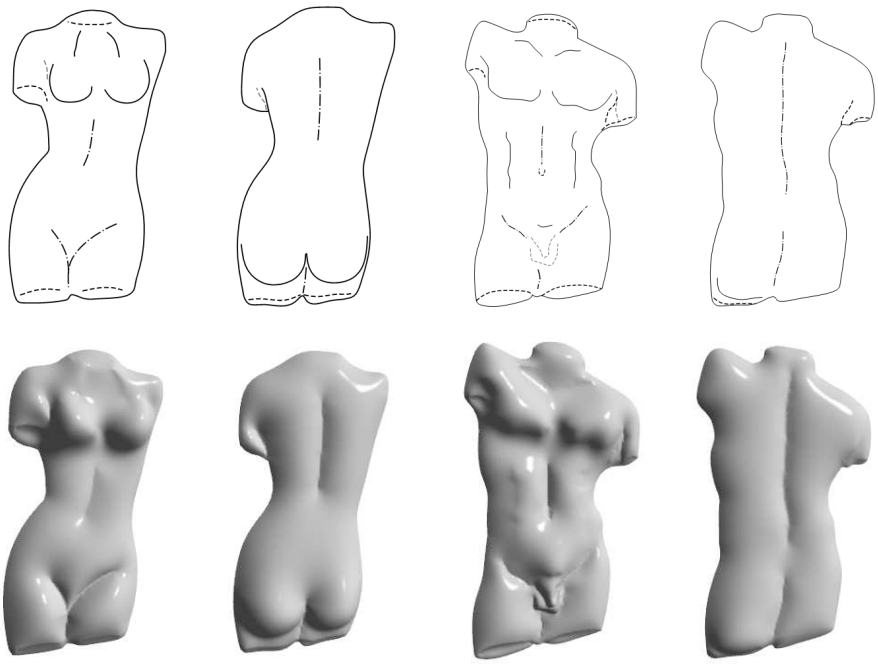


(a) Line drawing (b) LUMO method (c) ShapePalettes method (d) Our method

Figure 13: Cat Example: Comparison among LUMO [25], ShapePalettes [20], and our proposed method.

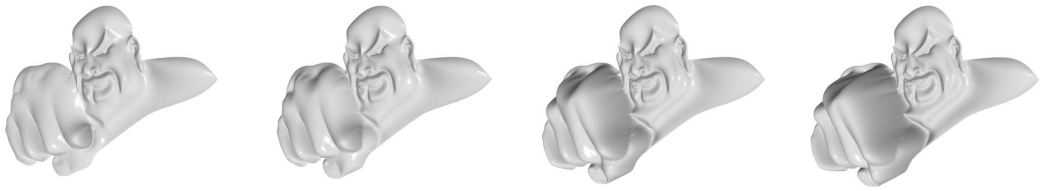
shows the ripples on the reconstructed surface clearly due to nonlinearity and curl-free constraint. Fig. 13 also shows the difference between our method and the related works. Due to the curl-free condition and non-linear interpolation, our method can reconstruct surface details more properly than other two methods. Note that LUMO and ShapePalettes also can generate similar surface structure 12-(f) if there are more strokes initially, which is clearly not intuitive by users.

Modeling & Animation: Fig. 14 shows the modeling intuitiveness using our method. Using less strokes, our method can generate similar quality of surfaces as reconstructed in [22] and these strokes are in accordance to the drawing style of artists. In practical



(a) Female front (b) Female back (c) Male front (d) Male back

Figure 14: Female & male examples.



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

Figure 15: Fisting animation sequence.

modeling process, artists draw strokes in their own styles. They may spend a little time to set stroke types or modify vectors to create more desirable surfaces if necessary.

Our method can also handle a simple animation by changing assigned vectors gradually. Fig. 15 demonstrates several frames selected from a fisting animation sequence. By changing the magnitudes of the initial vectors on the fist and reconstructing the surface, the animation is created. More complicated animation can be created by changing the magnitudes or directions of vectors.

Stopping Criterion: In order to provide a fair stopping criterion, we monitor the relative residuals defined by:

$$R_{vi}^k \equiv \frac{1}{|\Omega|}||\tilde{R}_{vi}^k||_{L^1} \leq \epsilon, \quad \forall i \in \{1,2,3\}, \tag{5.1}$$

where $||\cdot||_{L^1}$ is the L^1 norm on Ω, $|\Omega|$ is the area of domain, and

$$\left(\tilde{R}_{v1}^k, \tilde{R}_{v2}^k, \tilde{R}_{v3}^k\right) = \left(\mathbf{s}^k - \mathbf{n}^k, \nabla \times \mathbf{n}^k, \mathbf{P}^k - \nabla \mathbf{n}^k\right).$$

Since the relative residuals do not depend on the size of domain and penalty parameters, the criterion (5.1) with the given error bound ϵ is reasonable to stop iteration.

We also monitor the relative errors of Lagrange multipliers

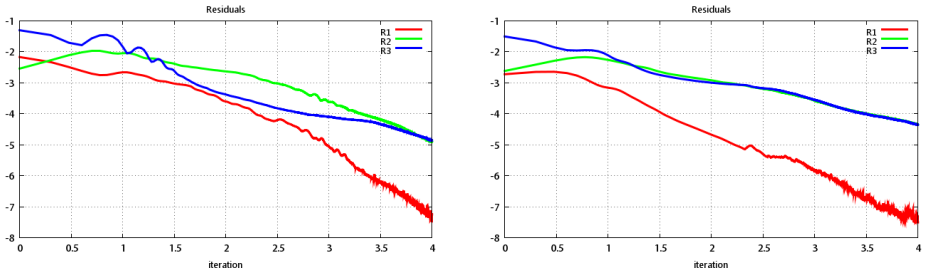
$$L_{v1}^k = \frac{||\boldsymbol{\lambda}_f^k - \boldsymbol{\lambda}_f^{k-1}||_{L^1}}{||\boldsymbol{\lambda}_f^{k-1}||_{L^1}}, \quad L_{v2}^k = \frac{||\lambda_c^k - \lambda_c^{k-1}||_{L^1}}{||\lambda_c^{k-1}||_{L^1}}, \quad L_{v3}^k = \frac{||\boldsymbol{\Lambda}_r^k - \boldsymbol{\Lambda}_r^{k-1}||_{L^1}}{||\boldsymbol{\Lambda}_r^{k-1}||_{L^1}}, \tag{5.2}$$

and the numerical energy

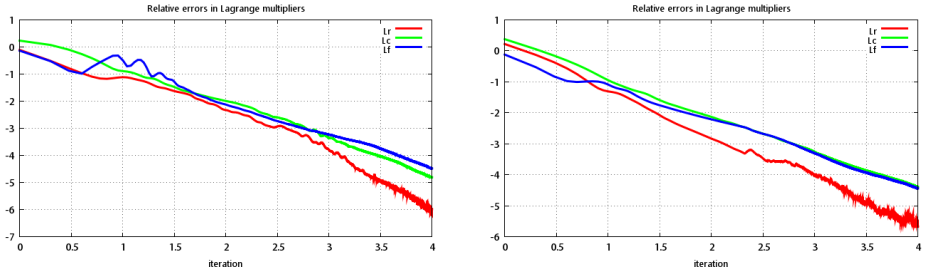
$$\mathscr{E}_v^k = \int_\Omega (1-g)|\mathbf{P}^k|_F + g|\mathbf{P}^k|_F^2 + \eta \int_\Gamma |\mathbf{s} - \mathbf{n}^*|. \tag{5.3}$$

The graphs of (5.1)-(5.3) are indicators to observe a convergence of the proposed algorithm.

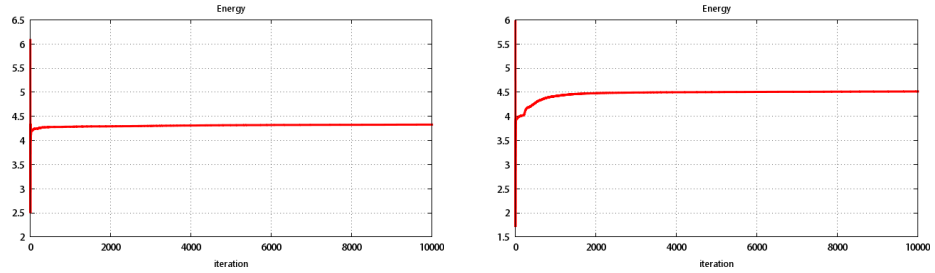
In Fig. 16, we illustrate the graph of (5.1) and (5.2) in the log scale until the outer iteration is $\log 10^4 = 4$. The y-axis and x-axis in the graph of (5.3) is the log and the



(a) Relative residuals (5.1)



(b) Relative errors of Lagrange multipliers (5.2)



(c) Numerical energy (5.3)

Figure 16: The graphs of (5.1)-(5.3) for Fig. 1-(d)(left) and Fig. 10-(b)(right). The log scale is used for y-axis in all graphs and x-axis in (a) and (b).

decimal scale, respectively. The residuals and the relative errors of Lagrange multipliers have the same convergence order. The energy reaches steady state. Note that the results in our paper are obtained in much less than 10^4 iterations; see Table 1. In our numerical experiments, no visual difference of the surface can be detected after the error bound 10^{-3} is satisfied. It is still a very difficult problem to find the most optimal number of iteration N for making visually same surfaces after N. We numerically observe that the penalty parameters affect the speed of convergence. Large values of the parameters yield slow decreasing of energy and small values make slow decreasing of relative errors in Lagrange multipliers and residuals. Therefore, it is necessary to tune the penalty parameters to obtain fast decreasing of residuals and energy. Heuristically, the same parameter setting is suitable for all examples in this paper.

Table 1: Computational cost for examples.

Example	Size	First Step		Second Step	
		Iter No.	Time (sec)	Iter No.	Time (sec)
Fig. 1-(d)	512^2	146	8.14	-	0.094
Fig. 4-(a)	256^2	90	1.77	-	0.062
Fig. 7-(e)	256^2	156	3.00	-	0.062
Fig. 7-(f)	256^2	307	5.86	-	0.062
Fig. 10-(b)	512^2	288	15.95	-	0.094
Fig. 10-(d)-1	512^2	80	4.50	-	-
Fig. 10-(d)-2	512^2	170	9.44	54	1.42
Fig. 14-(c)-1	256^2	77	1.52	-	-
Fig. 14-(c)-2	256^2	63	1.27	-	0.062
Fig. 15-(c)-1	512^2	254	13.95	-	-
Fig. 15-(c)-2	512^2	62	3.53	-	0.094

Computational Cost: Table 1 lists the iteration numbers and computational time (Intel(R) Xeon(R) CPU E5520 @ 2.27GHz, NVIDIA Quadro FX 1800) for examples in this paper. For examples with jump, bump or dip strokes, the vectors are interpolated twice, the first is to obtain a surface from regular, ridge and valley strokes, and the second is to create jump, bump and dip structures on top of the obtained dense vectors in the first interpolation. The geometry surrounding the jump, bump or dip structures are thus preserved, as shown in Figs. 2-(b), (c), 10-(d), and 14. If jump strokes are not used, iterative algorithm is not necessary in the second step because the height map is simply obtained by (4.19). For all the examples in this paper, parameters $\eta = 100$, $c_r = 10$, $c_c = 1$, and $c_f = 1$ are used in vector interpolation (4.1). For examples with jump strokes, $\xi = 0$ and $d_r = 0.01$ are used in surface reconstruction (4.18). The error bound ϵ for the first step is 0.001 to keep the coherence of the computational cost statistics in Table 1 even though a bigger value can be selected for certain examples to dramatically reduce the computation time with the same quality of reconstructed surfaces. The error bound for the second step is 0.05 if the iterative algorithm is necessary. Using augmented Lagrangian method and implementing our proposed method based on CUDA, the computational cost is quite low, which proves the proposed method is applicable for practical modeling tasks.

6. Conclusion

We have presented surface reconstruction based on a two-step method. In the first step, we proposed a nonlinear vector interpolation combining TV and H^1 regularization with the curl-free constraint for obtaining a dense vector field from given sparse vector field on strokes. In the second step, we proposed a height map reconstruction algorithm which integrates the dense vector field in the first step. The curl-free constraint in an interpolated

vector field makes a clear difference from other methods. Moreover, TV regularization allows to preserve jump discontinuities in the reconstructed surface and discontinuities of its gradient without surface distortion. We also provided different types of strokes to generate geometrically crucial structures such as ridge, valley, jump, bump, and dip on the surface, helping artists to create desirable surfaces they can intuitively imagine from 2D strokes. Comparing with other methods, the reconstructed surfaces from our proposed method are effectively affected by the assigned sparse vectors. Moreover, we can obtain fast numerical results using augmented Lagrangian method and local editing. The future direction of our research is to extend our proposed method to reconstruct and edit 3D objects. We are planning to use the proposed method to other applications such as 2D cartoon shading [37–39], geometric error fixing [40], etc.

Acknowledgments The research is supported by MOE (Ministry of Education) Tier II project T207N2202 and National Research Foundation grant, which is administered by the Media Development Authority Interactive Digital Media Programme Office, MDA (IDMPO).

References

[1] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes, "Sketch: an interface for sketching 3d scenes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, Aug. 1996, pp. 163–170.

[2] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (Siggraph 1999)*, 1999, pp. 409–416.

[3] O. A. Karpenko and J. F. Hughes, "SmoothSketch: 3D free-form shapes from complex sketches," in *Proceedings of the 33th annual conference on Computer graphics and interactive techniques (Siggraph 2006)*, 2006, pp. 589–598.

[4] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "FiberMesh: designing freeform surfaces with 3D curves," *ACM Transactions on Graphics (Siggraph 2007)*, vol. 26, no. 3, p. Article No. 41, July 2007.

[5] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, no. 1, pp. 88–103, Feb 2009.

[6] R. J. Woodham, "Photometric method for determining surface orientation from multiple images," *Shape from shading*, vol. 27, pp. 513–531, 1989.

[7] A. Hertzmann and S. M. Seitz, "Example-based photometric stereo: Shape reconstruction with general, varying brdfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1254 – 1264, 2005.

[8] T.-P. Wu and C.-K. Tang, "Dense photometric stereo by expectation maximization," *Computer Vision - ECCV 2006*, vol. 3954/2006, pp. 159–172, 2006.

[9] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape-from-shading: a survey," *IEEE Transcations on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 690–706, Aug 1999.

[10] E. Parados and O. Faugeras, *Shape from shading*. Springer, 2006.

[11] S. Morigi and M. Rucci, "Reconstructing surfaces from sketched 3d irregular curve networks," in *Proceedings of the 8th Eurographics workshop on Sketch-based interfaces and modeling (SBIM)*, 2011, pp. 39–46.

[12] C. Öztireli, U. Uyumaz, T. Popa, A. Sheffer, and M. Gross, "3d modeling with a symmetric sketch," in *Proceedings of the 8th Eurographics workshop on Sketch-based interfaces and modeling (SBIM)*, 2011, pp. 23–30.

[13] Y. Gingold, T. Igarashi, and D. Zorin, "Structured annotations for 2d-to-3d modeling," *ACM Transactions on Graphics (Siggraph Asia 2009)*, vol. 28, no. 5, p. Article No. 148, Dec 2009.

[14] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based mesh augmentation," in *Proceedings of the 2nd eurographics workshop on sketch-based interfaces and modeling (SBIM)*, 2005.

[15] A. Agrawal, R. Raskar, and R. Chellappa, "What is the Range of Surface Reconstructions from a Gradient Field?" *Computer Vision ÍC ECCV 2006*, pp. 578–591, 2006.

[16] R. T. Frankot, R. Chellappa, and S. Member, "A Method for enforcing integrability in shape from shading algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 118–128, 1987.

[17] N. Petrovic, I. Cohen, B. J. Frey, R. Koetter, and T. S. Huang, "Enforcing integrability for surface reconstruction algorithms using belief propagation in graphical models," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, p. 743, 2001.

[18] T. Simchony, R. Chellappa, and M. Shao, "Direct analytical methods for solving poisson equations in computer vision problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 435–446, 1990.

[19] L. Zhang, G. Dugas-Phocion, and J.-S. Samson, "Single-view modeling of free-form scenes," *The Journal of Visualization and Computer Animation*, vol. 13, pp. 225–235, 2002.

[20] T.-P. Wu, C.-K. Tang, M. Brown, and H.-Y. Shum, "Shapepalettes: Interactive normal transfer via sketching," *ACM Transactions on Graphics*, vol. 26, no. 307, p. Article No.44, 2007.

[21] M. Prasad and A. Fitzgibbon, "Single view reconstruction of curved surfaces," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, 2006, pp. 1345 – 1354.

[22] H.-S. Ng, T.-P. Wu, and C.-K. Tang, "Surface-From-Gradients Without Discrete Integrability Enforcement: A Gaussian Kernel Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2085–2099, 2009.

[23] X.-C. Tai and C. Wu, "Augmented lagrangian method, dual methods and split bregman iteration for rof model." in *SSVM*, ser. Lecture Notes in Computer Science, X.-C. Tai, K. Mľrken, M. Lysaker, and K.-A. Lie, Eds., vol. 5567. Springer, 2009, pp. 502–513.

[24] J. J. Koenderink, "Pictorial relief," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 256, no. 1740, pp. 1071–1086, May 1998.

[25] S. F. Johnston, "Lumo: illumination for cel animation," in *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*. New York, NY, USA: ACM, 2002, pp. 45–52.

[26] D. Terzopoulos, "The computation of visible-surface representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 4, pp. 417–438, 1988.

[27] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[28] C. Wu, J. Zhang, and X.-C. Tai, "Augmented lagrangian method for total variation restoration with non-quadratic fidelity," UCLA CAM Report 09-82, Tech. Rep., 2009.

[29] F. Cole, A. Golovinskiy, A. Limpaecher, H. S. Barros, A. Finkelstein, T. Funkhouser, and S. Rusinkiewicz, "Where do people draw lines?" in *Proceedings of the 35th annual conference on Computer graphics and interactive techniques (Siggraph 2008)*, 2008.

[30] J. Hahn, C. Wu, and X.-C. Tai, "Augmented Lagrangian method for generalized TV-Stokes model," UCLA CAM Report, Tech. Rep., 2010.

[31] ——, "Augmented Lagrangian method for generalized TV-Stokes models," *J. Sci. Comput*, vol. 50, no. 2, pp. 235–264, 2012.
[32] A. Chambolle, "An algorithm for total variational minimization and applications," *J. Math. Imaging Vis.*, vol. 20, pp. 89–97, 2004.
[33] T. Chan, G. Golub, and P. Mulet, "A nonlinear primal-dual method for total variation-based image restoration," *SIAM J. Sci. Comput.*, vol. 20, pp. 1964–1977, 1999.
[34] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM J. Img. Sci.*, vol. 2, pp. 323–343, 2009.
[35] G. Steidl and T. Teuber, "Removing multiplicative noise by douglas-rachford splitting methods," *J. Math. Imaging Vision*, vol. 36, no. 2, pp. 168–184, 2010.
[36] H. Winnemoller, A. Orzan, L. Boissieux, and J. Thollot, "Texture design and draping in 2d images," in *Eurographics Symposium on Rendering*, no. 4, 2009, pp. 1091–1099.
[37] K.-I. Anjyo and K. Hiramitsu, "Stylized highlights for cartoon rendering and animation," *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 54–61, 2003.
[38] K.-I. Anjyo, S. Wemler, and W. Baxter, "Tweakable light and shade for cartoon animation," in *Proceedings of NPAR 2006*, 2006, pp. 133–139.
[39] H. Todo, K.-I. Anjyo, W. Baxter, and T. Igarashi, "Locally controllable stylized shading," *ACM Transactions on Graphics (Siggraph 2007)*, vol. 26, no. 3, p. Article No. 17, August 2007.
[40] J. Tao, "Fixing geometric errors on polygonal models: A survey," *Journal of Computer Science and Technology*, vol. 24, pp. 19–29, 2009.