

A Method for Reconstructing a 3D Surface from a 2D Sketch

Field of the invention

5 The present invention relates to a method for reconstructing a three-dimensional (3D) surface from a two-dimensional (2D) sketch. This method may be used in a sketch-based surface modeling system for various applications.

Background of the Invention

10 Although geometric models may be constructed and/or manipulated using currently available modeling systems such as Maya [1], AutoCAD [2], SolidWorks [3] and CATIA [4], these currently available modeling systems often employ the WIMP (Window, Icon, Menu, Pointer) interface paradigm whereby to form and/or manipulate a 3D model, users are required to select operations from menus and floating palettes, enter
15 parameters into dialog boxes, and move control points [5]. Such a modeling process is not intuitive and hence, requires a lot of time and effort, especially if the desired 3D model is complex. On the other hand, sketching is a natural, easy and fast way to communicate ideas. By sketching, complex shapes can be formed in a few pencil
20 strokes. Thus, in computer modeling, sketching is often used to form a design in the early prototyping stages and the depicted design is then converted into a 3D model.

Motivated by the ease of sketching, a recent research direction known as sketch-based interfaces for modeling (SBIM) aims to automate and assist a sketch-to-3D translation
25 process. SBIM systems have been explored by researchers because of their perceived efficiency and intuitiveness. These systems [5] range from early systems such as SKETCH [12] and Teddy [13] to more recent systems such as SmoothSketch [14] and FiberMesh [15]. Without lighting or shading cues (such as those in photometric stereo [18], [19], [20] or shape-from-shading [21], [22]), the task of 3D model reconstruction
30 from 2D line drawings is more challenging than image-based 3D reconstruction. Although attempts based on contours [13], hidden contours [14], and structured annotations [23] have been made to deal with this challenging task, existing SBIM systems can still only model a limited range of objects with low complexity. These objects can take only a few shapes and can contain only either rough, smooth, or

rounded structures. Moreover, in existing SBIM systems, little consideration is given to surface details such as crease structures. Even though efforts have been made in existing SBIM systems to include surface details, these efforts are mostly unsuccessful. Currently, if surface details are desired, they have to be added onto the modeled object by for example, surficial augmentation techniques [24].

Many surface reconstruction algorithms [25], [26], [27], [28] enforce the integrability condition for producing correct surface heights from a gradient field. For single-view modeling, the method in reference [9] uses a constrained optimization with many types of geometrical constraints whereas the single-view modeling system [10] uses a close form method to reconstruct curved 3D surfaces based on apparent contour, inflation constraints, and normal specification in the parameter space. However, the method in [9] requires a large amount of user interaction to provide sufficient constraints for modeling a desired 3D surface (as shown by the authors of reference [7]) and the methods in [7], [9] and [10] encounter problems in handling sparse gradients, preserving sharp features, and preventing surface distortion [16].

Based on the observation that humans are good at assigning local surface normals for specifying local shapes [29], stroke-based surface reconstruction is implemented in [6], [7] in two steps. In the first step, dense surface normals are obtained via linear vector interpolation methods from sparse user-input vectors along given strokes. In the second step, the dense surface normals are integrated for reconstructing a height map. In LUMO [6], the vectors are interpolated based on a Telegrapher's equation (the damped wave equation) whereas in ShapePalettes [7], an energy functional minimization is used and the main mechanism of interpolation is based on a parabolic type partial differential equation (PDE) with a fourth order term related to the surface curvature. In both the LUMO and ShapePalettes methods, the governing equations are linear PDEs so numerical solvers such as the multigrid method or the Fast Fourier Transform (FFT) may be applied for the vector interpolation. However, both the LUMO and ShapePalettes methods are based on the Laplace operator which enforces smoothness in the interpolated vector field. This causes the assigned vectors to be diffused into the whole domain and then eventually averaged out in the final steady state, yielding only smooth reconstructed surfaces without surface details.

Summary of the invention

This present invention aims to provide a new and useful method for reconstructing a 3D surface from a 2D sketch.

In general terms, the present invention proposes a non-linear vector interpolation for reconstructing the 3D surface and further proposes numerical solvers for this non-linear vector interpolation.

A first aspect of the present invention is a method for reconstructing a three-dimensional surface in a domain wherein the three-dimensional surface is reconstructed from a two-dimensional sketch comprising at least one stroke with a vector field assigned to the at least one stroke and the method comprises the steps of: interpolating the vector field to form a dense vector field; and reconstructing the surface using the dense vector field; wherein interpolating the vector field comprises minimizing a non-linear functional of the vector field with a curl-free constraint on the dense vector field, the non-linear functional of the vector field involving terms defined over the domain and a term defined over the at least one stroke, and wherein the non-linear functional of the vector field is minimized using an augmented Lagrangian method.

The first aspect of the present invention is advantageous as it is a fast and efficient way of reconstructing a three-dimensional surface. Furthermore, it achieves a reconstructed surface which closely reflects the assigned vector field. It also helps to preserve surface details without surface distortion in the reconstructed surface. Hence, more desirable structures may be achieved. Furthermore, the non-linear functional is solved using an augmented Lagrangian method which is capable of solving the functional more quickly.

The invention may alternatively be expressed as a computer system (for example, a sketch-based surface modeling system) for performing such a method. This computer system may be integrated with a device for recording sketches whereby the device may further comprise a user interface for generating or extracting sketches. The invention may also be expressed as a computer program product, such as one

recorded on a tangible computer medium, containing program instructions operable by a computer system to perform the steps of the method.

Brief Description of the Figures

An embodiment of the invention will now be illustrated for the sake of example only with reference to the following drawings, in which:

Fig. 1 illustrates a flow diagram of a method for reconstructing a 3D surface from a 2D sketch according to an embodiment of the present invention;

Fig. 2(a) illustrates a user's sketch and Figs. 2(b) - (d) illustrate the results of applying steps in the method of Fig. 1 on the user's sketch in Fig. 2(a);

Fig. 3(a) illustrates a user's sketch and Figs. 3(b) - (d) illustrate the results of applying steps in the method of Fig. 1 on the user's sketch in Fig. 3(a);

Figs. 4(a) - (c) illustrate examples of different stroke types used in the method of Fig. 1;

Fig. 5(a) illustrates a HSV color map and Figs. 5(b) - (d) illustrate vectors assigned along example strokes in the method of Fig. 1;

Figs. 6(a) - (c) illustrate vectors assigned to a user's sketch and Figs. 6(d) - (f) respectively illustrate surfaces reconstructed with the method of Fig. 1 using the assigned vectors in Figs. 6(a) - (c);

Figs. 7(a) - (b) respectively illustrate conceptual drawings of a ridge stroke and a valley stroke;

Figs. 8(a) - (c) illustrate base strokes, strokes of interest and a reconstructed surface for a jump stroke, a bump stroke and a dip stroke respectively;

Figs. 9(a) - (d) illustrate an example of how a weighting function used in the method of Fig. 1 affects the reconstructed surface;

Figs. 10(a) - (f) illustrate another example of how the weighting function used in the method of Fig. 1 affects the reconstructed surface;

Figs. 11(a) - (f) illustrate an example of how imposing a curl-free condition on the interpolated vectors in the method of Fig. 1 affects the reconstructed surface;

Figs. 12(a) - (c) respectively illustrate relative residues, relative errors of Lagrange multipliers and numerical energy values plotted against the number of iterations in an iterative method used in the method of Fig. 1;

Fig. 13(a) illustrates a rule for indexing variables in an augmented Lagrangian functional used in the method of Fig. 1 and Fig. 13(b) illustrates an example of implementing the rule in Fig. 13(a) in a discrete computational domain;

Figs. 14(a) and 14(b) respectively illustrate a user's sketch and a surface reconstructed from the user's sketch with the method of Fig. 1, and Figs. 14(c) and 14(d) respectively show the user's sketch with added strokes and the reconstructed surface with modifications corresponding to the added strokes;

Figs. 15(a) and (e) respectively illustrate a user's sketch and initial vectors assigned to the user's sketch, and Figs. 15(b) and (f), (c) and (g), and (d) and (h) respectively illustrate a dense vector field and a surface reconstructed from the user's sketch using the LUMO method, the ShapePalettes method and the method of Fig. 1;

Fig. 16(a) illustrates a user's sketch and Figs. 16(b) - (d) respectively illustrate a surface reconstructed from the user's sketch using the LUMO method, the ShapePalettes method and the method of Fig. 1;

Figs. 17(a) - (c) respectively illustrate a user's sketch, vectors assigned to the user's sketch and a surface reconstructed from the user's sketch using the method of Fig. 1;

Figs. 18(a) - (d) illustrate user's sketches and the respective surfaces reconstructed from the user's sketches using the method of Fig. 1;

Figs. 19(a) - (b) illustrate another example of how the weighting function used in the method of Fig. 1 affects the reconstructed surface and Figs. 19(c) - (d) illustrate an example of how different stroke types affect the reconstructed surface in the method of Fig. 1;

Figs. 20(a) - (d) illustrate frames of an example animation sequence created using the method of Fig. 1.

Detailed Description of the Embodiments

Referring to Fig. 1, the steps are illustrated of a method 100 which is an embodiment of the present invention and which reconstruct a 3D surface from a 2D sketch comprising at least one stroke.

In step 102, stroke(s) are drawn to form a user's sketch and these stroke(s) are then classified based on their stroke types. In step 104, vectors are assigned along the stroke(s) in the user's sketch. Next, in step 106, the assigned vectors are interpolated

to form a dense vector field and in step 108, a 3D surface is reconstructed using the dense vector field.

The user's sketch may comprise basic 2D shapes formed from strokes sketched in step 102. In one example, simple line drawings which reflect the characteristics of the desired 3D surface are used to form the sketch. In another example, any shading, highlighting, hatching strokes and stippling in the user's sketch are ignored. The strokes of the user's sketch may be recorded as input into method 100 using a vector format. This recording may be performed using the drawing module of a vector-based animation system CACAni. After each stroke is recorded, the user may draw further strokes, trace the recorded stroke or adjust control points on the recorded stroke to modify his or her sketch.

An example user's sketch in the form of a line drawing of an ancient Chinese phoenix is shown in Fig. 2(a) whereby the strokes are drawn in the style of silhouette drawings and are in the form of cleaned-up vectorized strokes. Figs. 2(b) – 2(d) respectively show the results after applying steps 104, 106 and 108 on the user's sketch in Fig. 2(a). In particular, Fig. 2(b) illustrates a needle map of a portion of the sketch in Fig. 2(a) whereby the needle map shows initial vectors automatically assigned to the strokes in the portion of the sketch, Fig. 2(c) illustrates a dense vector field obtained after interpolating the assigned vectors in Fig. 2(b) and Fig. 2(d) illustrates a surface reconstructed using the dense vector field in Fig. 2(c).

Another example user's sketch is shown in Fig. 3(a). Similarly, Figs. 3(b) – 3(d) respectively show the results after applying steps 104, 106 and 108 on the user's sketch in Fig. 3(a) whereby Fig. 3(b) shows vectors assigned to the strokes in the sketch, Fig. 3(c) shows a dense vector field obtained from the assigned vectors in Fig. 3(b) and Fig. 3(d) shows a surface reconstructed using the dense vector field in Fig. 3(c).

Steps 102 – 108 will now be described in more detail.

Step 102: Draw stroke(s) to form a user's sketch and classify these strokes

In step 102, strokes are drawn to form a user's sketch. Different types of strokes may be provided to the user for forming the user's sketch.

The types of strokes drawn by the user in step 102 determine the surface geometry (or geometrical structures) to be recreated from the strokes. In step 102, the strokes in the user's sketch are then classified according to their stroke types. In one example, the users are able to intuitively imagine from the stroke types provided to them the surface geometries that will be recreated from the stroke types. A recent study in which lines are drawn by artists to convey 3D shapes [33] shows that humans can easily detect jumps, ridges, valleys, bumps and dips on 3D surfaces. Hence, in one example, the stroke types provided to the users comprise jump strokes, ridge strokes, valley strokes, bump strokes and dip strokes for reconstructing these distinctive local structures. The stroke types may further comprise regular strokes which serve to determine an overall shape of the reconstructed surface. In other words, each stroke in the user's sketch may be classified as one of the following: a regular stroke, a jump stroke, a ridge stroke, a valley stroke, a bump stroke or a dip stroke. In one example, the strokes are drawn in the style of silhouette drawings (for example, as shown in Fig. 2(a)) and are set as regular strokes by default. In this example, the user may edit the sketch by selecting one or more of the regular strokes and changing the stroke types of these selected strokes. This helps to form and edit the sketch in a fast and straightforward manner.

Figs. 4(a) – (c) illustrates examples of different stroke types (regular strokes 402 (black solid lines), ridge strokes 404 (gray dash-dot lines), valley strokes 406 (black dash-dot lines), jump strokes 408 (black dotted lines), bump strokes 410 (gray dashed lines), dip strokes 412 (black dashed lines)) provided in step 102 for forming the user's sketch and the respective surface geometries reconstructed from these stroke types. As shown in Figs. 4(a) – (c), the surface geometries formed from bump and dip strokes comprise discontinuities in the surface normals whereas the regular strokes translate to smooth changes in the height map. Note that although regular strokes appear to result in different structures in Figs. 4(a) – (c), this is due to the ambiguity in the view angles and the surface geometries recreated by regular strokes are in fact the same.

Step 104: Assign vectors along stroke(s) in the user's sketch

In step 104, a plurality of vectors is assigned along each stroke in the user's sketch. The assigned vectors (except those along jump strokes) represent the projection of 3D surface normals onto the 2D plane on which the sketch lies.

In one example, a plurality of initial vectors is first assigned automatically along each stroke in the user's sketch. The term "automatic" is used in this document to mean that, although human interaction may initiate the process, human interaction is not required while the process is being carried out. This assignment of initial vectors may be performed using a predefined scheme. In this example, the initial vectors are orthogonal to the tangents of the stroke (i.e. stroke tangents) and may have equal or varying magnitudes (for example, in the case of a T stroke as will be elaborated with reference to Fig. 5(d)). The initial vectors may then be modified (by modifying their magnitudes and/or directions) to more accurately indicate the desired surface geometries. The modification may be performed manually by a user using for example, a keyboard or a mouse based on the user's imagination of the desired surface. This is more intuitive than inputting numerical values indicating the desired surface heights. Furthermore, users may reconstruct a 3D surface from the initial vectors before deciding how the vectors should be modified. Alternatively, the modification may be performed automatically based on preloaded user settings or predefined schemes. Note that the modification of the initial vectors is optional.

Fig. 5(a) shows a black-and-white print out of a HSV (Hue-Saturation-Value) color map with $V = 1$. This map may be used to illustrate vectors assigned along strokes in a user's sketch as shown in Figs. 5(b) – (d). Fig. 5(b) shows the initial assigned vectors perpendicular to the stroke tangents and having equal magnitudes whereas Fig. 5(c) shows the initial vectors with their magnitudes and directions adjusted based on a particular user-defined setting. Fig. 5(d) shows how strokes joined at a T-junction may be handled. For a T-junction, the initial vectors may be assigned such that the magnitudes for the vectors near the T-junction along the visible part of the occluded stroke 502 are reduced to avoid affecting the vectors near the T-junction along the other stroke 504. Such a T junction may be automatically detected for the assignment of these initial vectors with varying magnitudes.

The vectors assigned to each stroke type in step 102 are geometrically meaningful. For example, a jump stroke indicates that a jump in the form of a step-like structure (i.e. a discontinuity or a sudden decrease in surface height) is to be formed in the reconstructed surface, and the directions and magnitudes of the vectors assigned along the jump stroke respectively represent the direction (i.e. on which side of the stroke the surface height should decrease) and magnitude of the jump to be formed. Furthermore, strokes belonging to the remaining stroke types indicate that a slope (i.e. gradual decrease in surface height) is to be formed in the reconstructed surface, and the directions and magnitudes of the vectors assigned along each of these strokes respectively represent the direction (i.e. on which side of the stroke the surface height should decrease) and gradient of the slope to be formed. Larger magnitudes indicate that a steeper slope, in other words, a higher rate of decrease in the surface height, is desired.

Regular strokes are strokes determining the overall shape of the reconstructed surface. In one example, initial vectors are assigned to regular strokes based on a default height map. Fig. 6(a) shows a sketch with initial vectors assigned along its regular strokes whereas Fig. 6(d) shows the 3D surface reconstructed using the vectors in Fig. 6(a). Fig. 6(b) shows only the vectors assigned along the strokes representing the nose 602 in the sketch. As compared to the corresponding vectors in Fig. 6(a), the vectors in Fig. 6(b) have increased magnitudes. Fig. 6(e) shows the 3D surface reconstructed using the vectors in Fig. 6(b) whereby it can be seen that the increased magnitudes of the vectors have resulted in a higher nose. Fig. 6(c) shows only the vectors assigned to strokes representing the left ear 604. These vectors are rotated 15 degrees anti-clockwise as compared to the corresponding vectors in Fig. 6(a). Fig. 6(f) shows the 3D surface reconstructed using the vectors in Fig. 6(c) whereby it can be seen that the rotation in the vectors have resulted in a twisted shape of the surface, leading to a twisted face.

A ridge (or valley) stroke indicates that a sharp ridge (or valley) on the reconstructed surface is desired. In one example, a ridge (or valley) stroke is in the form of two adjacent parallel regular strokes as shown in Figs. 7(a) – (b). Note that Figs. 7(a) – (b) are conceptual drawings and the space between the two parallel regular strokes is usually not visible. In fact, the ridge (or valley) stroke is usually not in this form in the

user's sketch. The stroke to be drawn for forming a ridge (or valley) stroke in the user's sketch is preferably in the form of a single line as this is more user-friendly (for example, as shown in Fig. 4, the ridge (or valley) stroke is in the form of a gray dash-dot line (or a black dash-dot line). As shown in Figs. 7(a) – (b), the initial vectors assigned to the two parallel regular strokes of the ridge (or valley) stroke are orthogonal to the stroke tangents and are set either in repulsive directions to form a ridge or in attractive directions to form a valley. In other words, the direction of the initial vectors assigned along a ridge stroke differs from the direction of the initial vectors assigned along a valley stroke by 180 degrees. In one example, the user can change only the magnitudes of the initial vectors assigned to the ridge (or valley) strokes and not the directions of these vectors. If the magnitudes of the vectors are increased by the user, steeper ridges (or deeper valleys) would be created in the reconstructed surface.

Jump, bump and dip strokes are useful in generating exquisite structures in the reconstructed surface. The structures may be formed on top of a given surface as shown in Figs. 8(a) – 8(c). This given surface may be an initial surface reconstructed from regular strokes, ridge strokes and/or valley strokes in the user's sketch.

Figs. 8(a) - (c) illustrate a base stroke, Strokes Of Interest(s) (SOIs) and a reconstructed surface for a jump stroke, a bump stroke and a dip stroke respectively. Each of a jump stroke, a bump stroke and a dip stroke comprises a base stroke, and either one adjacent SOI (in the case of a bump stroke or a dip stroke) or two adjacent SOIs (in the case of a jump stroke). The first, second, and third rows in Figs. 8(a) – (c) are conceptual drawings. In particular, the first row shows the given surfaces without jump, bump, and dip structures, the second row shows the changes in surface normals along the base strokes to form the respective structures and the third row shows the surfaces with the jump, bump and dip structures. The fourth row of Figs. 8(a) – (c) shows the reconstructed 3D surfaces with jump, bump and dip structures. The dashed, solid and dotted arrows in Figs. 8(a) – 8(c) respectively indicate the normal directions for the regular stroke in the user's sketch, the given surface reconstructed from the regular stroke and the base strokes of the jump, bump and dip strokes.

As shown in Fig. 8(a), a jump stroke comprises two SOIs which are parallel and adjacent to its base stroke. To form a desired jump, the surface height at the start of the vector assigned to the base stroke is decreased to form a right angled slope. Note that in this case, the magnitude of the vector assigned to the base stroke does not indicate the gradient of a slope. Rather, it indicates the size of the desired jump (i.e. the amount of decrease in surface height) in the reconstructed surface. If there is a given surface on which the jump is to be built on, vectors identical to the surface normals of the given surface are assigned to the SOIs, indicating the geometry of the given surface. If such a given surface is not present, the vectors assigned along the SOIs are initialized to zero.

As shown in Figs. 8(b) and 8(c), a bump (or dip) stroke comprises one SOI which is parallel and adjacent to its base stroke. The bump (or dip) stroke indicates that a slope is to be formed in the reconstructed surface and the magnitude of the vector assigned to the base stroke indicates the desired gradient of the slope. Similarly, if there is a given surface on which the bump (or dip) is to be built on, vectors identical to the surface normals of the given surface are assigned to the SOIs. If no such given surface is present, the vectors along the SOIs are initialized to zero.

An example procedure of using different strokes for reconstructing a 3D surface is as follows. Firstly, the user draws a rough shape of the desired surface using regular strokes, ridge strokes and/or valley strokes. Initial vectors are then assigned to these strokes and a surface is reconstructed from these assigned vectors. Based on this reconstructed surface, the user then modifies the magnitudes and/or directions of the initial vectors to obtain a more desired reconstructed surface. Jump, bump and dip strokes are then drawn to form detailed geometrical structures on the reconstructed surface. Vectors are then assigned to the base strokes and the SOIs of these jump, bump and dip strokes as shown in Figs. 8(a) – 8(c) whereby the vectors assigned to the SOIs represent the projected surface normals from the initial reconstructed surface (i.e. the given surface).

In one example, only the initial vectors assigned along regular strokes can be rotated by the user. This is because even though it is possible to adjust the geometry of the reconstructed surface by rotating the initial vectors assigned along other stroke types, it

is not intuitive to know the outcome of such rotations especially for users who do not understand the mathematical meaning of the vectors. Thus, to make method 100 more intuitive and user-friendly, it is preferable to disallow rotations of the initial vectors assigned along the other stroke types.

Step 106: Interpolate the assigned vectors to form a dense vector field

In step 106, the vectors assigned in step 104 are interpolated to obtain a dense vector field.

The sparse strokes in the user's sketch may be expressed as $\Gamma \subseteq \Omega$ where $\Omega \subset \mathbf{R}^2$ is the computational domain in which the 3D surface is to be reconstructed. The vectors assigned along the strokes Γ may be expressed in the form of a vector field $\mathbf{n}^* = (n_1^*, n_2^*)^T$. In step 106, this vector field $\mathbf{n}^*$ is interpolated to obtain a dense vector field $\mathbf{n} = (n_1, n_2)^T$ on the computational domain Ω . The 3D surface to be reconstructed is associated with a height map $z = I(x_1, x_2)$ whereby z is the desired height at the point (x_1, x_2) in the 2D plane on which the sketch lies. The assigned vectors $\mathbf{n}^*$ along the strokes (except jump strokes) may be regarded as the projection of surface normal vectors of the 3D surface (with the desired height map z) onto the 2D plane. Since the direction of the surface normal vectors of the 3D surface can be expressed as $(-\partial_1 I, -\partial_2 I, 1)$, the assigned vectors $\mathbf{n}^*$ along the strokes (except jump strokes) can be expressed as $-\nabla I$.

In vector calculus, if a computational domain $\Omega \subset \mathbf{R}^2$ is simply connected, then Equation (1) holds if and only if $\exists I : \Omega \rightarrow \mathbf{R}$ such that $\nabla I = \mathbf{n}$. Equation (1) represents the curl-free condition typically known as the integrability condition in visible surface reconstruction.

$$\nabla \times \mathbf{n} \equiv \partial_2 n_1 - \partial_1 n_2 = 0 \quad (1)$$

Since a surface associated with the height map $z = I(x_1, x_2)$ whose gradient fits the interpolated vectors $\mathbf{n}$ is to be reconstructed, the curl-free condition on the interpolated (or dense) vector field $\mathbf{n}$ is preferably satisfied.

- 5 Step 106 is performed using a weighted combination of a Total Variation (TV) regularization method for preserving sharpness of the reconstructed 3D surface and a H^1 regularization method for preserving smoothness of the reconstructed 3D surface. In particular, this may be performed using an energy minimization functional with a curl-free constraint as shown in Equation (2). Equation (2) shows minimizing a non-linear
- 10 functional of the vector field $\mathbf{n}^*$ involving terms defined over the domain Ω and a term defined over the strokes Γ . In other words, Equation (2) implements non-linear vector interpolation. Note that Equation (2) is to be solved with a curl-free constraint on the dense vector field $\mathbf{n}$.

15
$$\min_{\mathbf{n}} \left\{ \mathcal{E}_v(\mathbf{n}) \equiv \int_{\Omega} (1-g) |\nabla \mathbf{n}|_F + \int_{\Omega} g |\nabla \mathbf{n}|_F^2 + \eta \int_{\Gamma} |\mathbf{n} - \mathbf{n}^*| \right\}, \text{ with } \nabla \times \mathbf{n} = 0 \quad (2)$$

- The energy minimization functional in Equation (2) is a weighted combination of the TV regularization method and the H^1 regularization method. In Equation (2), a dense curl-free vector field $\mathbf{n}$ is obtained from the vector field $\mathbf{n}^*$ assigned along the strokes Γ
- 20 by minimizing an energy function $\mathcal{E}_v(\mathbf{n})$. $|\cdot|$ is the Euclidean norm in $\mathbf{R}^2$ and $|\cdot|_F$ is the Frobenius norm as shown in Equation (3) below. Note that the integration domain Γ includes both closed and open strokes. In other words, the integration domain Γ includes all the SOIs and base strokes in jump, bump and dip strokes.

25
$$|\nabla \mathbf{n}|_F^2 = \left\| \begin{pmatrix} \nabla n_1 \\ \nabla n_2 \end{pmatrix} \right\|_F^2 \equiv \sum_{j=1}^2 \sum_{i=1}^2 (\partial_j n_i)^2 \quad (3)$$

In Equation (2), the first term $\int_{\Omega} (1-g) |\nabla \mathbf{n}|_F$ represents the TV regularization method whereas the second term $\int_{\Omega} g |\nabla \mathbf{n}|_F^2$ represents the H^1 regularization

method. The third term $\eta \int_{\Gamma} |\mathbf{n} - \mathbf{n}^*|$ (where η is a positive constant) is a penalization parameter to preserve $\mathbf{n}^*$ on Γ in the interpolated vector field $\mathbf{n}$.

Generally, the H^1 regularization method reconstructs a smoother surface while the TV regularization method reconstructs a sharper surface. This is because discontinuities in the vector field assigned along the strokes are usually well interpolated into the domain Ω under the TV regularization method whereas they are smeared into the domain Ω under the H^1 regularization method. Thus, the first term representing the TV regularization method in Equation (2) serves to generate nonlinear diffusion and preserve discontinuities for example, discontinuities in creases or jumps in the interpolated vector field whereas the second term representing the H^1 regularization method in Equation (2) serves to generate linear diffusion and smear out discontinuities in the interpolated vector field.

Using the TV regularization method to preserve discontinuities is advantageous. This is because while other algorithms (for example, the algorithm in [20]) require information indicating the locations of the discontinuities in the domain Ω , the TV regularization method does not. Since it is usually very difficult to find the locations of the discontinuities from just the vectors assigned along the strokes, using the TV regularization method in Equation (2) allows an easier implementation of the method.

In Equation (2), the TV regularization method and the H^1 regularization method are weighted using a weighting function (or indicator function) g which may take on any value between 0 and 1. The value of g (i.e. the weight) determines how smooth or sharp the reconstructed 3D surface would be i.e. it determines the texture of the reconstructed 3D surface. Thus, by combining the TV regularization method and the H^1 regularization method as shown in Equation (2) with different values of g , various surface geometries can be generated in the reconstructed surface. Note that when $g = 1$, only the H^1 regularization method is implemented whereas when $g = 0$, only the TV regularization method is implemented. In one example, the value of g for the sketch is set based on a desired texture of the surface to be reconstructed and the

assigned vectors are interpolated by weighting the combination of the TV and H^1 regularization methods using this value of g .

The value of g for the sketch (including each stroke and the spaces between the strokes in the sketch) may be set by the user using for example, a painting tool. In one example, the painting tool is used to paint the sketch in a color associated with a variable alpha value and the value of g for the sketch may be set by varying the alpha value. To perform this, the value of g ($[0, 1]$) may be mapped with the alpha value (either $[0, 1]$ to $[0, 255]$ or $[0, 1]$ to $[255, 0]$).

In one example, after setting the value of g for the sketch, the value of g is modified for each stroke in the sketch based on the classification of the stroke. Considering the differences between the TV regularization method and the H^1 regularization method as mentioned above, g may be modified to 0 for jump, ridge, valley, bump, and dip strokes (to preserve the discontinuities in the resulting structures) and may be modified to 1 for regular strokes (to generate a smooth surface).

Alternatively, the user's sketch may be divided into a plurality of portions and a different weight may be set for each portion based on a desired texture of the surface to be reconstructed for the portion. In one example, g is set to 0 in a first portion to preserve discontinuities of surface gradients in the first portion and is set to 1 in a second portion to generate a smooth surface in the second portion. Each portion may be painted with a color associated with an alpha value which may be varied to set the weight for the portion. Similarly, the value of g may be mapped with the alpha value (either $[0, 1]$ to $[0, 255]$ or $[0, 1]$ to $[255, 0]$) and the value of g for each stroke in each portion may be modified according to the classification of the stroke.

Figs. 9(a) – 9(d) and 10(a) – 10(f) show how the value of the weighting function g affects the reconstructed 3D surface.

Figs. 9(a) and 9(b) respectively show the reconstructed surface from a user's sketch when $g = 1$ and when $g = 0$. Fig. 9(c) shows the division of the user's sketch into two

portions: a left half and a right shaded half. Setting $g=1$ on the left half of the sketch and $g=0$ on the right shaded half of the sketch results in the reconstructed surface in Fig. 9(d). As shown in Fig. 9(d), the reconstructed surface is a natural combination of the respective portions of the reconstructed surfaces in Figs. 9(a) and 9(b).

5

Figs. 10(a) and 10(b) respectively show a user's sketch comprising regular strokes and initial vectors assigned to the strokes. Note that the vectors assigned to the cross in the rectangle are parallel to the stroke tangents in the cross. In the initial vector field, four discontinuities are present at the four corners of the outer rectangle and three discontinuities are present on the cross. These seven discontinuous points are indicated by the arrows in Fig. 10(b). Fig. 10(d) shows the reconstructed surface by setting g to 0 (i.e. by using the TV regularization method alone) for the entire sketch.

10

As shown in Fig. 10(d), the discontinuities in the initial vector field are well interpolated and are in the form of sharp ridges. Fig. 10(e) shows the reconstructed surface by setting g to 1 (i.e. by using the H^1 regularization method alone) for the entire sketch.

15

As shown in Fig. 10(e), smooth surfaces are obtained in the regions corresponding to the interpolated vectors. Fig. 10(c) shows the division of the user's sketch into two portions: a textured portion and a non-textured portion. Setting $g=0$ for the textured portion and $g=1$ for the non-textured portion results in the reconstructed surface of

20

Fig. 10(f). As shown in Fig. 10(f), the reconstructed surface is a natural combination of the respective portions of the reconstructed surfaces in Figs. 10(d) and 10(e).

The model in Equation (2) is implemented with a curl-free constraint imposed on the dense vector field $\mathbf{n}$. This helps to reconstruct the height map I with $\nabla I = \mathbf{n}$. It also helps to interpolate sparse vectors over a large domain. Figs. 11(a) – 11(f) illustrate the advantages of having the curl-free constraint imposed on the interpolated vectors. Fig. 11(a) illustrates a user's sketch comprising regular strokes whereas Fig. 11(b) illustrates the initial vectors assigned along strokes of the user's sketch. Figs. 11(c) and Fig. 11(d) illustrate reconstructed surfaces obtained without the curl-free constraint (with Fig. 11(c) illustrating a reconstructed surface obtained using the H^1 regularization method and Fig. 11(d) illustrating a reconstructed surface obtained using the TV regularization method). Figs. 11(e) and Fig. 11(f) illustrate reconstructed surfaces obtained with the curl-free constraint (with Fig. 11(e) illustrating a reconstructed surface

25

30

obtained using the H^1 regularization method and Fig. 11(f) illustrating a reconstructed surface obtained using the TV regularization method). Comparing Figs. 11(c) and 11(d) against Figs. 11(e) and 11(f) respectively, it can be seen that with the curl-free constraint, the weighting function has a greater effect on the reconstructed surface. Furthermore, with the curl-free constraint, the reconstructed surfaces reflect the initial vector settings more accurately and sharper surface details can be obtained.

However, it is difficult to apply the curl-free constraint such that the constraint is satisfied within the entire domain Ω because this constraint is usually violated at discontinuities in the interpolated vector field. The following describes an example method to overcome this difficulty.

Augmented Lagrangian method to minimize the energy function in Equation (2)

In one example, an augmented Lagrangian method is used to minimize the energy functional (i.e. non-linear functional of the vector field) in Equation (2). This overcomes the difficulty in applying the curl-free constraint and allows the curl-free constraint to be satisfied almost everywhere in the domain Ω . This method is elaborated below.

Using an operator splitting method with two variables as shown in Equation (4), Equation (2) is transformed into a constraint minimization problem as shown in Equation (5).

$$\mathbf{P} = \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} \text{ and } \mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \quad (4)$$

$$\min_{\mathbf{n}} \int_{\Omega} (1-g) |\mathbf{P}|_F + \int_{\Omega} g |\mathbf{P}|_F^2 + \eta \int_{\Gamma} |\mathbf{s} - \mathbf{n}^*|$$

with $\mathbf{P} = \nabla \mathbf{n}$, $\mathbf{s} = \mathbf{n}$, and $\nabla \times \mathbf{n} = 0$ (5)

To solve the constraint minimization in Equation (5), an augmented Lagrangian functional $L_v(\mathbf{s}, \mathbf{n}, \mathbf{P}; \lambda_f, \lambda_c, \Lambda_r)$ as shown in Equation (6) is formed where c_f, c_c and c_r are positive penalty parameters, and λ_f, λ_c and Λ_r are Lagrangian multipliers. If

the weighting function g in Equation (2) is set to 1 on Ω , it is not necessary to use c_r , Λ_r and $\mathbf{P}$.

$$\begin{aligned}
 L_v(.) \equiv & \int_{\Omega} (1-g) |\mathbf{P}|_F + g |\mathbf{P}|_F^2 + \eta \int_{\Gamma} |\mathbf{s} - \mathbf{n}^*| \\
 & + \int_{\Omega} \Lambda_r \cdot (\mathbf{P} - \nabla \mathbf{n}) + \frac{c_r}{2} \int_{\Omega} |\mathbf{P} - \nabla \mathbf{n}|_F^2 \\
 & + \int_{\Omega} \lambda_f \cdot (\mathbf{s} - \mathbf{n}) + \frac{c_f}{2} \int_{\Omega} |\mathbf{s} - \mathbf{n}|^2 \\
 & + \int_{\Omega} \omega_c \lambda_c (\nabla \times \mathbf{n}) + \frac{c_c}{2} \int_{\Omega} (\nabla \times \mathbf{n})^2
 \end{aligned} \tag{6}$$

ω_c in Equation (6) is a weighting function. ω_c is adaptively chosen depending on the geometrical structure to be reconstructed which in turn depends on the stroke types. In one example, ω_c is set to 0 for ridge, valley, bump, dip and jump strokes and is set to 1 otherwise because the interpolated vector field $\mathbf{n}$ violates the curl-free constraint on discontinuities in surface height and gradient. In one example, the weighting function ω_c is determined automatically from prior knowledge of the stroke types on the user's sketch. Note that although crease structures which violate the curl-free constraint may arise when $\omega_c=1$, surface distortion is reduced because of the presence of the TV regularization method in Equation (2) (see Fig. 9(d)). In one example, since ω_c is a smooth function, ω_c is set as the convolution of a 2D Gaussian kernel with a smooth factor 2. In other words, Gaussian linear filtering with a scale parameter 2 is performed to obtain ω_c . The same kernel may also be used as the weighting function g in Equation (2). Having ω_c in Equation (6) is useful as without this variable, it may be necessary to choose a substantially small penalty parameter c_c for strokes which violate the curl-free constraint. A small c_c leads to a slow convergence for the iterative procedure described later on and thus, is not practical.

The following is an example algorithm incorporating an iterative procedure to find the saddle point of the augmented Lagrangian functional in Equation (6) via maximizing the Lagrangian multipliers and minimizing variables $\mathbf{s}$, $\mathbf{n}$ and $\mathbf{P}$.

5 Algorithm 1: Augmented Lagrangian method for Equation (6)

Step 1: Initialize $\mathbf{s}^0, \mathbf{n}^0, \mathbf{P}^0, \lambda_f^0, \lambda_c^0$ and Λ_r^0 to 0. A variable k (representing the iteration the algorithm is at) is also initialized to 0.

10 Step 2: Increase the value of k by 1 and if $k \geq 1$, compute an approximate minimizer $(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k)$ of the augmented Lagrangian functional $L_v(\cdot; \lambda_f^{k-1}, \lambda_c^{k-1}, \Lambda_r^{k-1})$ using an alternating minimization method. This is performed according to Equation (7) whereby $\mathbf{s}, \mathbf{n}, \mathbf{P}$ are the variables to be minimized:

15
$$(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k) \approx \operatorname{argmin} L_v(\mathbf{s}, \mathbf{n}, \mathbf{P}; \lambda_f^{k-1}, \lambda_c^{k-1}, \Lambda_r^{k-1}) \quad (7)$$

Step 3: Update Lagrange multipliers using the following equations (8) – (10)

$$\lambda_f^k = \lambda_f^{k-1} + c_f(\mathbf{s}^k - \mathbf{n}^k) \quad (8)$$

20
$$\lambda_c^k = \lambda_c^{k-1} + \frac{c_c}{\omega_c}(\nabla \times \mathbf{n}^k) \quad (9)$$

$$\Lambda_r^k = \Lambda_r^{k-1} + c_r(\mathbf{P}^k - \nabla \mathbf{n}^k) \quad (10)$$

Note that if $\omega_c = 0$ at a point, then $\lambda_c^k = \lambda_c^{k-1}$ at the point

25 Step 4: Calculate relative residuals R_{vi}^k using Equation (11). If the relative residues R_{vi}^k are larger than an error bound ε , repeat steps 2 – 4. Note that the relative residues do not depend on the size of the domain or the penalty parameters and are hence suitable to be used for determining if the iteration should stop.

$$R_{vi}^k \equiv \frac{1}{|\Omega|} \|\tilde{R}_{vi}^k\|_{L^1} \quad \forall i \in \{1,2,3\} \quad (11)$$

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

$$(\tilde{R}_{v1}^k, \tilde{R}_{v2}^k, \tilde{R}_{v3}^k) = (\mathbf{s}^k - \mathbf{n}^k, \nabla \times \mathbf{n}^k, \mathbf{P}^k - \nabla \mathbf{n}^k) \quad (12)$$

In one example, the relative errors of the Lagrange multipliers are also calculated in step 4 of Algorithm 1 to determine if the iteration should stop (i.e. if steps 2 – 4 should be repeated). These relative errors are calculated as shown in Equation (13). In a further example, the numerical energy is also calculated to determine if the iteration should stop. The numerical energy is calculated as shown in Equation (14). Similarly, the relative errors and/or numerical energy values are compared against respective error bounds to determine if the iteration should stop.

$$L_{v1}^k = \frac{\|\lambda_f^k - \lambda_f^{k-1}\|_{L^1}}{\|\lambda_f^{k-1}\|_{L^1}}, L_{v2}^k = \frac{\|\lambda_c^k - \lambda_c^{k-1}\|_{L^1}}{\|\lambda_c^{k-1}\|_{L^1}}, L_{v3}^k = \frac{\|\Lambda_r^k - \Lambda_r^{k-1}\|_{L^1}}{\|\Lambda_r^{k-1}\|_{L^1}} \quad (13)$$

$$\varepsilon_v^k = \int_{\Omega} (1-g) |\mathbf{P}^k|_F + g |\mathbf{P}^k|_F^2 + \eta \int_{\Gamma} |\mathbf{s} - \mathbf{n}^*| \quad (14)$$

Fig. 12(a) shows plots of the relative residues (calculated using Equation (11)) against the number of iterations (the plots in Fig. 12(a) show a total of 10,000 iterations (in the log scale, this is $\log 10000 = 4$), Fig. 12(b) shows plots of the relative errors of the Lagrange multipliers (calculated using Equation (13)) against the number of iterations and Fig. 12(c) shows plots of the numerical energy values (calculated using Equation (14)) against the number of iterations for the examples in Figs. 2(d) (top) and 18(b) (bottom). The axes of Figs. 12(a) and (b) are in the log scale whereas in Fig. 12(c), the y-axis is in the log scale and the x-axis is in the decimal scale. In Fig. 12(a), R1, R2 and R3 respectively correspond to $\tilde{R}_{v1}^k, \tilde{R}_{v2}^k, \tilde{R}_{v3}^k$ in Equation (12) whereas in Fig. 12(b), L_i, L_c, L_r respectively correspond to $L_{v1}^k, L_{v2}^k, L_{v3}^k$ in Equation (13).

The plots in Figs. 12(a) – (c) numerically show that the above numerical algorithms are convergent. The relative residues and the relative errors of Lagrange multipliers have the same convergence order. Furthermore, the numerical energy reaches a steady state. Note that the results shown in the figures in this application are obtained in much less than 1000 iterations; see Table 1. In example numerical experiments, it was found that hardly any visual difference in the reconstructed surface can be detected after the condition for the relative residues to be less than an error bound of 0.001 is satisfied.

As shown in Figs. 12(a) – 12(c), the relative residuals and the relative errors of the Lagrange multipliers have the same convergence order whereas the plot for the numerical energy values reaches a steady state with less iteration. Note that the speed at which the plots converge is affected by the values of the penalty parameters. Larger penalty parameter values will result in the numerical energy values decreasing more slowly across the iterations (i.e. the numerical energy plot takes a longer time to converge) whereas smaller penalty parameter values will result in the relative errors in Lagrange multipliers and the relative residues decreasing more slowly across the iterations (i.e. the relative errors plot and the relative residues plot take a longer time to converge). It is preferable to tune the penalty parameters such that the relative errors plot and the relative residues plot converge faster.

The following is an example algorithm for an alternating minimization method to solve Equation (7) in Algorithm 1 as described above.

Algorithm 2: Alternating minimization method to solve Equation (7)

Step 1: Initialize the variables: $\tilde{\mathbf{s}}^0 = \mathbf{s}^{k-1}$, $\tilde{\mathbf{n}}^0 = \mathbf{n}^{k-1}$ and $\tilde{\mathbf{P}}^0 = \mathbf{P}^{k-1}$

Step 2: For $l = 1, \dots, L$ and fixed Lagrange multipliers $\lambda_f = \lambda_f^{k-1}$, $\lambda_c = \lambda_c^{k-1}$ and $\Lambda_r = \Lambda_r^{k-1}$, solve the following minimization problems alternatively:

$$\tilde{\mathbf{s}}^l = \operatorname{argmin} L_v(\mathbf{s}, \tilde{\mathbf{n}}^{l-1}, \tilde{\mathbf{P}}^{l-1}; \lambda_f, \lambda_c, \Lambda_r) \quad (15)$$

$$\tilde{\mathbf{n}}^l = \operatorname{argmin} L_v(\tilde{\mathbf{s}}^l, \mathbf{n}, \tilde{\mathbf{P}}^{l-1}; \lambda_f, \lambda_c, \Lambda_r) \quad (16)$$

$$\tilde{\mathbf{P}}^l = \operatorname{argmin} L_v(\mathbf{s}^l, \tilde{\mathbf{n}}^l, \mathbf{P}; \lambda_f, \lambda_c, \Lambda_r) \quad (17)$$

The minimizers $\tilde{\mathbf{s}}^l$, $\tilde{\mathbf{n}}^l$ and $\tilde{\mathbf{P}}^l$ above are found by minimizing the energy functionals (18), (19) and (20).

$$\varepsilon_1(\mathbf{s}) = \int_{\Omega} \frac{c_f}{2} |\mathbf{s} - \tilde{\mathbf{n}}^{l-1}|^2 + \lambda_f \cdot \mathbf{s} + \eta \int_{\Gamma} |\mathbf{s} - \mathbf{n}^*| \quad (18)$$

$$\begin{aligned} \varepsilon_2(\mathbf{n}) = \int_{\Omega} \frac{c_r}{2} |\tilde{\mathbf{P}}^{l-1} - \nabla \mathbf{n}|_F^2 - \Lambda_r \cdot \nabla \mathbf{n} + \frac{c_c}{2} (\nabla \times \mathbf{n})^2 + \omega_c \lambda_c (\nabla \times \mathbf{n}) \\ + \frac{c_f}{2} |\tilde{\mathbf{s}}^l - \mathbf{n}|^2 - \lambda_f \cdot \mathbf{n} \end{aligned} \quad (19)$$

$$\varepsilon_3(\mathbf{P}) = \int_{\Omega} (1 - g) |\mathbf{P}|_F + g |\mathbf{P}|_F^2 + \frac{c_r}{2} |\mathbf{P} - \nabla \tilde{\mathbf{n}}^l|_F^2 + \Lambda_r \cdot \mathbf{P} \quad (20)$$

More details on how the minimization of the functionals (18) – (20) may be performed are described below.

Step 3: After L iterations, $(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k)$ is updated by setting $(\mathbf{s}^k, \mathbf{n}^k, \mathbf{P}^k) = (\tilde{\mathbf{s}}^L, \tilde{\mathbf{n}}^L, \tilde{\mathbf{P}}^L)$. In one example, only one iteration (i.e. L = 1) is performed and this is sufficient to achieve desirable results.

Minimization of functionals (18) – (20)

The minimization of functionals (18) and (20) (c.f. Equations (15) and (16)) may be solved by a soft thresholding method [25] involving simple arithmetic computation whereas the minimization of functional (19) (c.f. Equation (17)) is terminated by solving a system of linear equations over the whole domain Ω . Since the grid for this system of linear equations is uniform, Fast Fourier Transform (FFT) can be used to solve these equations at a low computational cost.

The variables $\mathbf{s}, \mathbf{n}, \mathbf{P}, \lambda_f, \lambda_c, \Lambda_r$ in the augmented Lagrangian functional in Equation (6) may be defined on a discrete computational domain $\Omega = [1, N_1] \times [1, N_2]$. Figs. 13(a) – 13(b) show a staggered grid system. Physically different locations in the staggered grid system are used to evaluate the values of the variables.

Fig. 13(a) shows the rule of indexing variables $\mathbf{s}, \mathbf{n}, \mathbf{P}, \boldsymbol{\lambda}_f, \lambda_c, \boldsymbol{\Lambda}_r$ in the augmented Lagrangian functional in one example. In this example, the first and second components of $\mathbf{n}$, $\mathbf{s}$ and $\boldsymbol{\lambda}_f$ are defined at squares $\square$ and hollow dots $\circ$ respectively whereas p_{12} , p_{21} , Λ_{r12} and Λ_{r21} are defined at solid dots $\bullet$. The other variables are defined at triangles Δ whereby $\lambda_c(i, j)$ is at the triangle 1302, $p_{11}(i, j)$ and $\Lambda_{r11}(i, j)$ are at the triangle 1304, and $p_{22}(i, j)$ and $\Lambda_{r22}(i, j)$ are at the triangle 1306. These rules of indexing may be used in the discretization of the Euler-Lagrange equations for (19). Fig. 13 (b) shows an example of implementing the rule in Fig. 13(a) in a discrete computational domain of size 5×4 (i.e. $[1, 5] \times [1, 4]$).

Details of examples for minimizing the functionals (18) – (20) in step 2 of Algorithm 2 are as follows.

Minimization of $\mathcal{E}_1(\mathbf{s})$ in Equation (18)

Denoting $\mathbf{n} = \tilde{\mathbf{n}}^{j-1}$, the functional in $\mathcal{E}_1(\mathbf{s})$ can be represented in two parts as shown in Equations (21) – (23).

$$\mathcal{E}_1(\mathbf{s}) = \mathcal{E}_{\Omega\Gamma}(\mathbf{s}) + \mathcal{E}_\Gamma(\mathbf{s}) \quad (21)$$

where

$$\mathcal{E}_{\Omega\Gamma}(\mathbf{s}) \equiv \int_{\Omega\Gamma} \boldsymbol{\lambda}_f \cdot \mathbf{s} + \frac{c_f}{2} |\mathbf{s} - \mathbf{n}|^2 \quad (22)$$

$$\mathcal{E}_\Gamma(\mathbf{s}) \equiv \int_\Gamma \eta |\mathbf{s} - \mathbf{n}^*| + \boldsymbol{\lambda}_f \cdot \mathbf{s} + \frac{c_f}{2} |\mathbf{s} - \mathbf{n}^*|^2 \quad (23)$$

The minimizer in the first energy functional $\mathcal{E}_{\Omega\Gamma}(\mathbf{s})$ is easily obtained since the integrant is a quadratic polynomial in terms of $\mathbf{s}$. To obtain the minimizer in the second energy functional $\mathcal{E}_\Gamma(\mathbf{s})$, Equation (23) is reformulated as Equation (24). In Equation (24), the value of C is independent of the minimization.

$$\varepsilon_r(\mathbf{s}) \equiv \int_{\Gamma} \eta |\mathbf{s} - \mathbf{n}^*| + \frac{c_f}{2} \left| \mathbf{s} - \mathbf{n}^* + \frac{\boldsymbol{\lambda}_f}{c_f} \right|^2 + C \quad (24)$$

For each coordinate $(i, j) \in \Gamma$, a soft thresholding method [32] is used and the minimizer $\tilde{\mathbf{s}}^l$ in Equation (15) is obtained by the following Equations (25) – (26) where

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

$$(i, j) \notin \Gamma \Rightarrow \tilde{\mathbf{s}}^l(i, j) = \mathbf{n}(i, j) - \frac{1}{c_f} \boldsymbol{\lambda}_f(i, j) \quad (25)$$

$$(i, j) \in \Gamma \Rightarrow \tilde{\mathbf{s}}^l(i, j) = \mathbf{n}^*(i, j) + \alpha(i, j) \mathbf{x}_0(i, j) \quad (26)$$

10 Minimization of ε_2 in Equation (19)

For a fixed $\mathbf{P} = \tilde{\mathbf{P}}^{l-1}$ and $\mathbf{s} = \tilde{\mathbf{s}}^l$, the Euler-Lagrange equation of (19) yields a system of linear partial differential equations (PDEs) as shown in Equation (27) where

$\mathbf{n} = (n_1, n_2)^T$, $\nabla^\perp = (-\partial_2, \partial_1)^T$, $\nabla \cdot \mathbf{P} = (\nabla \cdot \mathbf{p}_1, \nabla \cdot \mathbf{p}_2)^T$, I is a 2x2 identity matrix,

$$15 \quad D_r = (\partial_1^2 + \partial_2^2)I \text{ and } D_c = \nabla^\perp \nabla^{\perp T}.$$

$$-(c_r D_r + c_c D_c - c_f I) \mathbf{n} = \nabla \cdot \boldsymbol{\Lambda}_r - \nabla^\perp (\omega_c \boldsymbol{\lambda}_c) + \boldsymbol{\lambda}_f - \nabla \cdot \mathbf{P} + c_f \mathbf{s} \quad (27)$$

20 Since D_r is a Laplacian operator and operates on each component of $\mathbf{n}$, its presence in Equation (27) makes Equation (27) a coupled equation and results in the diffusion of $\mathbf{n}^*$ on the strokes. However, the differential operator D_c in Equation (27) (present due to the enforcement of the curl-free constraint on the dense vector field) generates extra force to interpolate $\mathbf{n}^*$ onto the whole domain.

Using the standard finite difference scheme with the rule of indexing variables as shown in Fig. 13(a), n_1 in Equation (27) is discretized at a square $\square$ whereas n_2 in Equation (27) is discretized at a hollow dot $\circ$. Applying the periodic boundary condition to all variables, the discrete Fourier Transform F may be used to achieve the discretization of Equation (27). The shifting operators comprising the finite differences in Equation (27) essentially represent a discrete convolution whose discrete Fourier Transform is a component-wise multiplication in the frequency domain.

For discrete frequencies u_i, u_j , a system of linear equations as shown in (28) may be formed.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} F n_1(u_i, u_j) \\ F n_2(u_i, u_j) \end{pmatrix} = \begin{pmatrix} f_1(u_i, u_j) \\ f_2(u_i, u_j) \end{pmatrix} \quad (28)$$

Denoting

$$\varsigma_i = 2 \sin^2 \frac{v_i}{2} + \sqrt{-1} \sin v_i \quad (29)$$

$$\varsigma_j = 2 \sin^2 \frac{v_j}{2} + \sqrt{-1} \sin v_j, \quad (30)$$

where $v_i = \frac{2\pi}{N_1} u_i$ ($u_i = 1, \dots, N_1$) and $v_j = \frac{2\pi}{N_2} u_j$ ($u_j = 1, \dots, N_2$), the coefficients

in the system of linear equations in (28) are

$$\begin{aligned} a_{11} &= 4c_r \sin^2 \frac{v_i}{2} + 4(c_r + c_c) \sin^2 \frac{v_j}{2} + c_f \\ a_{12} &= c_c (-\bar{\varsigma}_j) \varsigma_i \quad a_{21} = c_c (-\bar{\varsigma}_i) \varsigma_j \\ a_{22} &= 4(c_r + c_c) \sin^2 \frac{v_i}{2} + 4c_r \sin^2 \frac{v_j}{2} + c_f \end{aligned} \quad (31)$$

Furthermore, f_1 and f_2 on the right hand side of Equation (28) can be expressed as follows.

$$f_1 = \bar{\zeta}_j F \alpha_{12}^1 + \bar{\zeta}_i F \alpha_{11}^2 - F \alpha_1^3 \quad (32)$$

$$f_2 = \bar{\zeta}_j F \alpha_{21}^1 + \bar{\zeta}_i F \alpha_{22}^2 - F \alpha_2^3 \quad (33)$$

where

$$\alpha_{mn}^1 \equiv \Lambda_{rmn} + c_r (p_{mn} - p_{nm}) + (-1)^m \omega_c \lambda_c \quad (34)$$

$$\alpha_{mn}^2 \equiv \Lambda_{rmn} + c_r p_{mn} \quad (35)$$

$$\alpha_m^3 \equiv \lambda_{fm} + c_f s_m \quad (36)$$

There are $N_1 N_2$ 2x2 systems and the determinant of the coefficients matrix in the above equations for all discrete frequencies is given by Equation (37) where

$$\beta_{ij} = \sin^2 \frac{\nu_i}{2} + \sin^2 \frac{\nu_j}{2}. \quad (37)$$

This determinant is not zero since the penalty parameters c_r , c_c , and c_f are positive. After the system of linear equations is solved for each frequency, the discrete inverse Fourier transform is used to obtain $\tilde{\mathbf{n}}^l$.

$$(4c_r \beta_{ij} + c_f)(4(c_r + c_c) \beta_{ij} + c_f) \quad (37)$$

Minimization of $\varepsilon_3(\mathbf{P})$ in Equation (20)

Denoting $\mathbf{n} = \tilde{\mathbf{n}}^l$, the functional in Equation (20) can be reformulated as follows. In Equation (38), the value of C is independent of the minimization.

$$\varepsilon_3(\mathbf{P}) = \int_{\Omega} (1-g) |\mathbf{P}|_F + \frac{c_r + 2g}{2} \left| \mathbf{P} - \frac{1}{c_r + 2g} (c_r \nabla \mathbf{n} - \Lambda_r) \right|_F^2 + C \quad (38)$$

The following approach [8] may be applied to find the closed form of the minimizer at each point $(i, j) \in \Omega$:

$$\mathbf{p}(i, j) = \max \left\{ 0, 1 - \frac{1 - g(i, j)}{c_r \|\mathbf{W}(i, j)\|_F} \right\} \mathbf{W}(i, j) \quad (39)$$

where

$$\mathbf{W} = \frac{c_r}{c_r + 2g} (c_r \nabla \mathbf{n} - \lambda_r) \quad (40)$$

Step 108: Reconstruct a 3D surface from the dense vector field

In step 108, a 3D surface associated with a height map $z = I(x_1, x_2)$ is reconstructed on the computational domain Ω from the dense vector field $\mathbf{n}$ obtained in step 106. This is performed by locating a height map I whose gradient field fits the dense vector field $\mathbf{n}$.

In one example, the 3D surface is reconstructed via a weighted combination of two methods, a TV regularization method for preserving the sharpness of the reconstructed surface and a H^1 regularization method for preserving the smoothness of the reconstructed surface. This may be based on the minimization of energy functional $\mathcal{E}_h(I)$ as shown in Equation (41) which implements a weighted combination of the TV regularization method and the H^1 regularization method. In Equation (41), I_0 represents desired height values on $\Sigma \subset \Omega$ and h is a weighting function (or indicator function) which may take on any value between 0 and 1. Similar to Equation (2), Equation (41) shows minimizing a non-linear functional of a height map, the non-linear functional involving terms defined over the domain Ω (the domain in which the surface is to be reconstructed) and a term defined over the domain Σ (the domain in which prior information regarding heights is present). The prior information regarding heights, in other words, the initial height information, may be specified by the user, making the modeling process similar to a morphing process from one height map to another. Furthermore, certain formats of the initial height information may comprise

vectors on a part of the domain. In one example, if no prior information regarding heights is available, ξ is set to 0. Furthermore, the convolution of a 2D Gaussian kernel with a smooth factor 2 may be used for the weighting function h .

$$\min \left\{ \varepsilon_h(I) = \int_{\Omega} (1-h) |\nabla I - \mathbf{n}| + \int_{\Omega} h |\nabla I - \mathbf{n}|^2 + \xi \int_{\Sigma} |I - I_0| \right\} \quad (41)$$

In Equation (41), the term $|\nabla I - \mathbf{n}|$ represents the TV type norm (i.e. the TV regularization method) whereas the term $|\nabla I - \mathbf{n}|^2$ represents the H^1 type norm (i.e. the H^1 regularization method). The TV type norm preserves sharpness of the reconstructed surface and reconstructs discontinuities (for example, jump discontinuities) without surface distortion whereas the H^1 type norm enforces smoothness of the reconstructed surface by enforcing the C^0 continuity of the height map (since $\int_{\Omega} h |\nabla I - \mathbf{n}|^2$ yields the Laplacian operator in the Euler-Lagrange equation).

It is preferable not to apply the H^1 type norm to the entire domain Ω (i.e. not to set $h = 1$ for the entire domain Ω) as problems of overshooting or undershooting may be encountered with the jumps in the reconstructed surface. On the other hand, the use of the TV type norm when $h \neq 1$ helps to prevent these overshooting or undershooting problems. In one example, considering the different effects from the TV and H^1 type norms, the value of the weighting function h (i.e. weight) is set individually for each stroke based on the classification of the stroke. For example, it may be set as 0 for jump strokes and 1 for other strokes. In one example, the location of the jump strokes are known beforehand and h is set automatically based on this prior knowledge.

Minimization in Equation (41)

In one example, the minimization problem of Equation (41) is solved by an augmented Lagrangian method as follows. Note that there are other methods for minimizing the energy function in Equation (41) as discussed in references [39] – [42]. The same applies for minimizing the energy function in Equation (2).

The minimization problem of Equation (41) is first transformed into a constraint minimization problem as shown in Equation (42) whereby $\mathbf{q} = \nabla I$ and $J = I$.

$$\min_I \int_{\Omega} (1-h) |\mathbf{q} - \mathbf{n}| + \int_{\Omega} h |\mathbf{q} - \mathbf{n}|^2 + \xi \int_{\Sigma} |J - I_0| \quad (42)$$

The minimization problem is then cast into an unconstrained problem as shown in Equation (43) and a saddle point of the augmented Lagrangian functional in Equation (43) is found. In Equation (43), d_r and d_f are positive penalty parameters and μ_r and μ_f are Lagrange multipliers.

$$\begin{aligned} L_h(\cdot) \equiv & \int_{\Omega} (1-h) |\mathbf{q} - \mathbf{n}| + h |\mathbf{q} - \mathbf{n}|^2 \\ & + \int_{\Omega} \mu_r \cdot (\mathbf{q} - \nabla I) + \frac{d_r}{2} \int_{\Omega} |\mathbf{q} - \nabla I|^2 \\ & + \int_{\Omega} \mu_f (J - I) + \frac{d_f}{2} \int_{\Omega} (J - I)^2 + \xi \int_{\Sigma} |I - I_0| \end{aligned} \quad (43)$$

The optimality condition for Equation (43) may be found using an iterative algorithm similar to Algorithm 1 and Algorithm 2 as described above. More details on this iterative algorithm are described in reference [11], the contents of which are incorporated by reference herein. This iterative algorithm is typically used when $\xi \neq 0$ or when $h \neq 1$ on Ω .

If no prior information about the desired heights is available and no jump strokes are present, $\xi = 0$ and h is set as $h = 1$, and Equation (41) simply comprises the term representing the H^1 type norm. In this case, the Euler-Lagrange equation for I is simply a Poisson equation as shown in Equation (44). Thus, Equation (41) need not be solved by an iterative algorithm. Instead, the minimization of the energy functional in Equation (41) can be simply achieved by solving the Poisson equation in Equation (44). In other words, a solution of the Poisson equation in Equation (44) is a reconstructed height map whose gradient fits the vector field $\mathbf{n}$ (this is due to the curl-free constraint imposed on Equation (2)).

$$\nabla \cdot (\nabla I) = \nabla \cdot \mathbf{n} \quad (44)$$

Surface distortion is eliminated in the solution of Equation (44) since the interpolated vector field $\mathbf{n}$ satisfies the integrability condition. Hence, a surface associated with a height map whose gradient fits the vector field $\mathbf{n}$ may be reconstructed without surface distortion.

Step 110: Perform local editing on the reconstructed 3D surface

Step 110 is an optional step for modifying surface details on the reconstructed surface. A real time computation may be performed in step 110 since the computational cost for step 110 is not high.

The entire domain of the reconstructed surface need not be interpolated again to modify surface details in the reconstructed surface. Instead, in one example, the user's sketch is first modified by adding at least one further stroke to the sketch and the computational domains are re-defined based on where the modifications in the user's sketch are. Equations (2) and (41) are then implemented using the re-defined computational domains.

Figs. 14(a) and 14(b) respectively show a user's sketch and a reconstructed surface from the user's sketch whereas Figs. 14(c) and 14(d) respectively show the user's sketch with added strokes (i.e. with local editing) and the reconstructed surface with additional surface details corresponding to the added strokes.

As shown in Fig. 14(c), several strokes are added to the sketch within a local domain 1402 and a bounding rectangle 1404 around the strokes in the local domain is located (either manually or automatically). A portion of the reconstructed surface corresponding to the bounding rectangle 1404 in the sketch is then modified as follows.

To incorporate the additional surface details into the reconstructed surface based on the additional strokes in Fig. 14(c), the computational domains are re-defined. In one example, Ω is re-defined as the domain enclosed by the bounding rectangle 1404

whereas Γ is re-defined as the strokes in the bounding rectangle 1404 (comprising the newly added strokes, previously present strokes, and the region 1406). The region 1406 comprises the region within the bounding rectangle 1404 minus the region within the boundary 1402. Vectors are then assigned along the strokes Γ and the assigned
5 vectors are then interpolated using Equation (2) to form a dense vector field. The portion of the reconstructed surface corresponding to the bounding rectangle in the sketch is then modified by reconstructing this portion using Equation (41). Equations (2) and (41) are computed using the re-defined domain Ω and strokes Γ . Furthermore, in Equation (41), Σ can be set as the region 1406 and the height information on Σ can
10 be used as I_0 . As shown in Fig. 14(d), the additional surface details are incorporated without changing the geometry of the previously reconstructed surface. In the example shown in Figs. 14(a) – 14(d), the domain size is 64x64 pixels and the local editing step uses less than 1 second. Note that although a single local domain 1402 is shown in Fig. 14, two or more separate local domains may be defined in a single sketch.

Comparison of method 100 with existing algorithms

As compared to existing WIMP-based systems, using method 100 to reconstruct 3D surfaces is more intuitive, less time-consuming and less labor-intensive. Furthermore,
20 the 3D surfaces reconstructed by existing SBIM systems such as SKETCH [12], Teddy [13], SmoothSketch [14] and FiberMesh [15] are usually either entirely rough or entirely smooth and rounded. In other words, the 3D surfaces reconstructed by these existing SBIM systems usually do not comprise surface details, for example the shapes of creases. Method 100 does not have these limitations.

Method 100 is also advantageous over the methods in LUMO [6] and ShapePalettes [7]. The vector interpolation methods in LUMO and ShapePalettes are linear and the reconstructed surfaces do not faithfully reflect the assigned sparse vectors. On the
25 other hand, method 100 can faithfully reflect the vector settings in the reconstructed surface due to the presence of the curl-free condition. Although this requires nonlinear vector interpolation, a fast solver based on augmented Lagrangian method [8] is used in method 100 to allow the nonlinear vector interpolation to be performed at a fast
30 speed.

In one example, the drawing as shown in Fig. 15(a) [34] is traced by an artist in the form of a sketch and Fig. 15(e) shows the assigned vectors along strokes in this sketch. The dense vector fields and reconstructed surfaces obtained for this sketch using the LUMO method, ShapePalettes method and method 100 are shown respectively in Figs. 15(b) and (f), (c) and (g), and (d) and (h). As shown in Figs 15(f) and (g), the ripples on the clothes are "smoothed out" in the reconstructed surfaces because of the linearity and the lack of appropriate enforcement of the curl-free constraint in the LUMO and ShapePalettes methods. In the ShapePalettes method, the curl-free constraint leads to surface distortion in the reconstructed surface as the constraint is not satisfied at jump discontinuities and discontinuities in the surface normals. If prior information regarding the locations of such discontinuities is available, it may be possible to prevent the surface distortion from being formed with the use of a proper weight function. However, such prior information is seldom available. On the other hand, the surface reconstructed using method 100 clearly comprises the ripples on the clothes due to the non-linear vector interpolation with the enforcement of the curl-free constraint, the use of the TV regularization method and the use of the augmented Lagrangian method.

In another example, a surface is reconstructed from a sketch of a cat as shown in Fig. 16(a) using the prior art methods and method 100. The reconstructed surfaces using the LUMO method, ShapePalettes method and method 100 are respectively shown in Figs. 16(b) – (d). As shown in these figures, the LUMO and ShapePalettes methods can only reconstruct 3D surfaces with limited shapes and are unable to preserve surface details. On the other hand, due to the curl-free condition and the non-linear interpolation used in method 100, the surface details in the reconstructed surface in Fig. 16(d) are sharper i.e. better preserved. Note that although the LUMO and ShapePalettes methods are also capable of generating a reconstructed surface similar to that in Fig. 16(d), a large amount of additional effort (for example, drawing extra strokes and implementing additional vector settings) is required. These additional steps may not be intuitive.

Furthermore, compared with most single-view modeling methods [9, 10] that require a large amount of user interaction (hence, time) for constraint specification, the vector assignment approach in method 100 is intuitive and user-friendly. Also, the algorithms

in [7], [9], [10] have difficulties handling sparse gradients, preserving sharp features, and preventing surface distortion in surface-from-gradients construction as pointed out by the authors of reference [16] (which uses a Gaussian kernel approach without discrete integrability enforcement to overcome the difficulties). On the other hand, method 100 is not faced with the same difficulties due to the use of the TV regularization method and the imposition of the curl-free condition with the nonlinear vector interpolation. On top of this, method 100 provides a very fast and efficient method to solve the energy functionals in the nonlinear vector interpolation via an augmented Lagrangian method.

Google SketchUp [17] is a sketch-based 3D modeling software and enables intuitive and fast modeling of 3D surfaces. However, to use the program in SketchUp, the height of the 3D surface needs to be manually specified by users. This is performed by using a mouse dragging operation or a keyboard input. On the other hand, the height of the 3D surface reconstructed in method 100 may be automatically estimated. Furthermore, the system in SketchUp is not a single-view modeling system whereas such a system may be implemented using method 100. Furthermore, the process of using method 100 is also more aligned with traditional design stages, as designers usually sketch in 2D space during the early design stage before modeling and manipulating a model of the sketch in 3D space. In addition, method 100 is also capable of modeling complex surfaces as illustrated in Figs. 17(a) – (c) using simple strokes and a fast vector assignment. On the other hand, it may be difficult to create such freeform surfaces using SketchUp.

Advantages of the embodiments of the present invention

The key advantages of method 100 are as follows.

Combination of TV and H^1 regularization methods in method 100

Previously, surface reconstruction from sparse information based on an energy minimization approach has been performed using visible-surface representations [30] whereby the weighted H^1 norm of data, together with its gradient with discrete fidelities, controls the local smoothness of the reconstructed surface. However, it is difficult to

approximate a proper weight function for this. Another previous method to control local smoothness is the TV norm of data [31]. As jump discontinuities under the TV norm of data are easily preserved without weighting functions, discontinuities of surface gradients can be recovered by the TV norm of data gradient.

5 In method 100, the TV regularization and H^1 regularization methods are combined in a weighted manner to implement stroke-based surface reconstruction. The texture of the reconstructed surface can be easily controlled by varying the weight used for the combination. This not only provides artists more freedom to create various types of
10 surface geometries, it also helps in the reconstruction of geometrically crucial structures (see Figs. 10 and 19) and the preservation of surface details (so that the reconstructed surface comprises sharper surface features) without causing surface distortion. A few of these advantages arise due to the use of the TV regularization method. The TV regularization method not only helps to preserve discontinuities in the
15 vector field, but also helps to preserve discontinuities in the gradient of the reconstructed surface without causing surface distortion. Thus, jump discontinuities can be preserved on the reconstructed surface without overshooting or undershooting.

Use of the nonlinear vector interpolation with the curl-free condition in method 100

20 In method 100, a nonlinear vector interpolation is performed to obtain a dense vector field from a sparse vector field assigned along strokes in the user's sketch. This interpolation is performed with a curl-free condition imposed on the dense vector field.

25 The nonlinear vector interpolation in method 100 achieves a faithful reflection of the vectors' settings and the resulting interpolated vector field in method 100 satisfies the integrability (curl-free) condition almost everywhere in the domain. This preserves surface details better without causing surface distortion in the reconstructed surface, hence achieving more desirable structures corresponding to the given strokes and
30 vector settings (see Fig. 11).

Method 100 is intuitive and user-friendly

Method 100 is capable of creating a reconstructed surface as shown in Fig. 2(d) by using only regular strokes and default initial vector settings. In other words, method 100 can reconstruct a surface using strokes in accordance with the typical drawing style of artists. It is common for artists to draw strokes in their own styles, instead of spending time to set stroke types or modify vectors to create more desirable surfaces. Thus, method 100 is intuitive and user-friendly.

Initial vectors may be automatically assigned based on for example, default initial vector settings, to strokes in the user's sketch. In one example, the assigned vectors are orthogonal to the stroke tangents and are of the same magnitude. Automatically assigning vectors to strokes in the user's sketch is advantageous over manually assigning the vectors (for example, by assigning them based on a 3D reference model as performed in [7]) since it reduces the amount of work the user has to perform, hence allowing method 100 to be more user-friendly.

However, the automatically assigned vectors can still be adjusted by the user to form a more detailed surface shape. In one example, the user can modify the magnitudes and directions of the assigned vectors using a keyboard or a mouse based on his or her imagination of the desired surface. This is more intuitive than inputting numerical values for the surface heights.

Ability to construct different types of surface geometries using different stroke types and/or the weighting function g

To reconstruct more complicated surfaces, users have the option of using various stroke types to generate different geometrically crucial structures such as ridge, valley, jump, bump and dip structures. In method 100, the strokes are classified according to the desired surface geometry to be formed. The different types of strokes help users create desired structures on the reconstructed surface. These structures can be intuitively imagined from the corresponding 2D strokes (see Figs. 18 and 19) and thus, allows artists to create surfaces which they can intuitively perceive from the 2D strokes. Geometrically meaningful vectors are assigned to each type of stroke based on the desired surface geometry. These assigned vectors may be a projection of surface

normal vectors onto the 2D plane on which the user's sketch lie. Method 100 can successfully transform these sparse assigned vectors into desired structures in the reconstructed surface.

5 More complicated surfaces may also be formed by adjusting the weighting function g in Equation (2). For example, to reconstruct a surface comprising discontinuity in its gradient, a sketch may be divided into different textured regions and a user simply has to decide which of the textured regions should contain the discontinuities. This is less tedious than existing methods (for example, the algorithm in reference [16]) whereby it is necessary to indicate where exactly gradient discontinuities are located in the domain.

Fig. 19(a) illustrates a user's sketch of a monkey. In Fig. 19(a), the user's sketch is divided into two portions: a textured portion on the right and a non-textured portion on the left. Setting $g=0$ in the textured portion on the right and $g=1$ in the non-textured portion on the left results in the reconstructed surface of Fig. 19(b). As shown in Fig. 19(b), this reconstructed surface comprises different geometrical structures which appear sharp on the right and smooth on the left.

20 Figs. 19(c) – 19(d) illustrates the use of various stroke types to reconstruct a surface with gradient or jump discontinuities. Fig. 19(c) illustrates a user's sketch of a monkey with different strokes whereas Fig. 19(d) illustrates the reconstructed surface from the user's sketch in Fig. 19(c). As shown in Fig. 19(d), the reconstructed surface preserves discontinuities without any distortion and clearly comprises the intended features according to the stroke types in the user's sketch.

Speed and low computational cost of method 100

30 Fewer strokes are required by method 100 to generate reconstructed surfaces of a particular quality as compared to for example the algorithm in reference [16]. Furthermore, local editing may be performed in method 100 whereby to add a few surface details to a previously reconstructed surface, it is not necessary to interpolate the entire domain. Rather it is simply necessary to re-define the domains based on where the modifications in the user's sketch are.

Furthermore, using the augmented Lagrangian method in method 100 helps to overcome the difficulties in solving the constraint minimization problem. Even though a nonlinear process is implemented for both the vector interpolation and the height map reconstruction in method 100, results can still be achieved quickly using very fast and efficient numerical solvers based on the augmented Lagrangian method [32]. This reduces the required computational time and increases the speed of the modeling process to reconstruct the 3D surface. The non-linear vector interpolation in method 100 can be implemented at a low computational cost and at a high computational speed using the augmented Lagrangian method and a CUDA-based GPU computation. In one example, using a nVidia Quadro FX 1800, a computational time of less than 15 and 5 seconds respectively for a domain size of 512 x 512 pixels and 256 x 256 pixels is required. A lower computational time may be achieved depending on the user's sketch.

Table 1 shows the number of iterations performed and the computational time (Intel(R) Xeon(R) CPU E5520 @ 2.27GHz, NVIDIA Quadro FX 1800) used for creating the reconstructed surfaces shown in some of the above figures.

To reconstruct the surfaces in the figures shown in Table 1, if the sketch comprises jump, bump or dip strokes, two interpolations are performed. The first interpolation is performed for reconstructing a 3D surface from regular, ridge and valley strokes and the second interpolation is performed for creating jump, bump and dip structures on top of the reconstructed surface from the first interpolation. This preserves the geometry surrounding the jump, bump or dip structures as shown in Figs. 2(d), 10, 18 and 19. In Table 1, "-1" indicates that the results in that figure are obtained after the first interpolation whereas "-2" indicates that the results in that figure are obtained after the second interpolation.

If $\xi = 0$ and $h = 1$ on Ω (i.e. the user's sketch does not comprise jump strokes), an iterative algorithm is not required to solve Equation (41) since the height map is simply obtained using Equation (44). In Table 1, a dash "-" in the "Number of iterations" column for step 108 indicates that no iteration is required to solve Equation (41) to obtain the results in the figure. A dash "-" in the "Time" column for step 108 indicates

that step 108 is not performed. To reconstruct the surfaces in the figures shown in Table 1, if jump strokes are not present in the user's sketch, the following parameters $\eta = 100, c_r = 10, c_c = 1$ and $c_f = 1$ are used in Equation (6). If jump strokes are present in the user's sketch, the following parameters $\xi = 0$ and $\mu_r = 0.01$ are used in the reconstruction step i.e. in Equation (43).

Furthermore, for computing the statistics in Table 1, the error bound ε in Algorithm 1 is set as 0.001 in step 106 and 0.05 in step 108 (if the iterative algorithm is necessary) for all the figures shown in Table 1. This maintains the coherence of the computational cost statistics for comparison using Table 1. However, note that bigger error bounds may be selected to reconstruct some of the surfaces to reduce the computation time required.

As shown in Table 1, little iteration is required to reconstruct the surfaces in the above figures. Furthermore, the reconstructed surfaces are of reasonably high quality and can be obtained in a short time. Thus, it can be concluded that method 100 can be implemented quickly and at a low computational cost.

Figure	Size of domain	Step 106		Step 108	
		Number of Iterations	Time (sec)	Number of iterations	Time (sec)
Fig. 2(d)	512^2	146	8.14	-	0.094
Fig. 6(a)	256^2	90	1.77	-	0.062
Fig. 10(e)	256^2	156	3.00	-	0.062
Fig. 10(f)	256^2	307	5.86	-	0.062
Fig. 18(b)	512^2	288	15.95	-	0.094
Fig. 18(d)-1	512^2	80	4.50	-	-
Fig. 18(d)-2	512^2	170	9.44	54	1.42
Fig. 17(c)-1	256^2	77	1.52	-	-
Fig. 17(c)-2	256^2	63	1.27	-	0.062
Fig. 19(d)-1	512^2	254	13.95	-	-
Fig. 19(d)-2	512^2	62	3.53	-	0.094

Table 1

Multiple applications of method 100

Method 100 is not only capable of reconstructing a static 3D surface. It is also capable of creating an animation sequence made up of a series of static 3D surfaces by gradually changing the vectors assigned to a user's sketch. For example, a plurality of frames for an animation sequence may be created by iteratively reconstructing a 3D surface from a user's sketch whereby the assigned vectors along the strokes in the user's sketch for each iteration are different. Thus, method 100 can be used in an animation tool for creating short animation clips. Figs. 20(a) – (d) show various frames selected from an example animation sequence formed using method 100. This animation sequence is generated by changing the magnitudes of the assigned vectors along the strokes forming the fist and reconstructing the 3D surface each time the magnitudes of the vectors are changed. Each reconstructed 3D surface is a frame of the animation sequence. A more complicated animation sequence may be formed by changing not only the magnitudes of the assigned vectors but also their directions.

Furthermore, method 100 may be used in other applications, for example, 2D cartoon shading [35], [36], [37] or geometric error fixing [38], etc. A sketch-based surface modeling software may also be developed using method 100 and this software can be used in various design applications for reconstructing complex 3D surfaces based on user's sketches easily, quickly and accurately.

REFERENCES

[1] AutoDesk Maya.

<http://usa.autodesk.com/adsk/servlet/pc/index?id=13577897&siteID=123112>.

[2] AutoCAD.

<http://usa.autodesk.com/adsk/servlet/pc/index?id=13779270&siteID=123112>.

[3] SolidWorks. <http://www.solidworks.com/>.

[4] CATIA. <http://www.3ds.com/products/catia>.

[5] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):88–103, Feb 2009.

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

[7] T.-P. Wu, C.-K. Tang, M. Brown, and H.-Y. Shum. 2007. Shapepalettes: Interactive normal transfer via sketching. In *ACM Transactions on Graphics* volume 26 Issue 3 07, Article No.44.

[8] X.-C. Tai and C. Wu, Augmented Lagrangian method, dual methods and split Bregman iteration for ROF model, in *SSVM '09: Proceedings of the Second International Conference on Scale Space and Variational Methods in Computer Vision*, Berlin, Heidelberg, 2009, Springer-Verlag, pp. 502-513.

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

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

[11] J. Hahn, C. Wu, and X.-C. Tai. 2010. Augmented Lagrangian method for generalized TV-Stokes models. In *UCLA CAM Report 10-30*, 2010.

[12] 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*, pages 163–170, August 1996.

[13] 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)*, pages 409–416, 1999.

- [14] 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)*, pages 589–598, 2006.
- [15] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. FiberMesh: designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (Siggraph 2007)*, 26(3):Article No. 41, July 2007.
- [16] H.-S. Ng, T.-P. Wu, and C.-K. Tang. 2009. Surface-From-Gradients Without Discrete Integrability Enforcement: A Gaussian Kernel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- 10 [17] Google SketchUp. <http://sketchup.google.com/>.
- [18] R. J. Woodham, "Photometric method for determining surface orientation from multiple images," *Shape from shading*, vol. 27, pp. 513–531, 1989.
- [19] 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.
- 15 [20] T.-P. Wu and C.-K. Tang, "Dense photometric stereo by expectation maximization," *Computer Vision - ECCV 2006*, vol. 3954/2006, pp. 159–172, 2006.
- [21] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape-from shading:a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 690–706, Aug 1999.
- 20 [22] E. Parados and O. Faugeras, *Shape from shading*. Springer, 2006.
- [23] 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.
- 25 [24] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketchbased mesh augmentation," in *Proceedings of the 2nd eurographics workshop on sketch-based interfaces and modeling (SBIM)*, 2005.
- [25] A. Agrawal, R. Raskar, and R. Chellappa, "What is the Range of Surface Reconstructions from a Gradient Field?" *Computer Vision C ECCV 2006*, pp. 578–591, 30 2006.
- [26] 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.

- [27] 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.
- 5 [28] 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.
- [29] J. J. Koenderink, "Pictorial relief," *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 256, no. 1740, pp. 1071–1086, May 1998.
- 10 [30] D. Terzopoulos, "The computation of visible-surface representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 4, pp. 417–438, 1988.
- [31] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.
- [32] 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.
- 15 [33] 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.
- [34] H. Winnemoller, A. Orzan, L. Boissieux, and J. Thollot, "Texture design and draping in 2d images," no. 4, pp. 1091–1099, 2009.
- 20 [35] K. ichi Anjyo and K. Hiramitsu, "Stylized highlights for cartoon rendering and animation," *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 54–61, 2003.
- [36] K. ichi Anjyo, S. Wemler, and W. Baxter, "Tweakable light and shade for cartoon animation," in *Proceedings of NPAR 2006*, 2006, pp. 133–139.
- 25 [37] H. Todo, K. ichi 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.
- [38] J. Tao, "Fixing geometric errors on polygonal models: A survey," *Journal of Computer Science and Technology*, vol. 24, pp. 19–29, 2009.
- 30 [39] A. Chambolle, "An algorithm for total variational minimization and applications," *J. Math. Imaging Vis.*, vol. 20, pp. 89–97, 2004.
- [40] 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.

- [41] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," SIAM J. Img. Sci., vol. 2, pp. 323–343, 2009.
- [42] G. Steidl and T. Teuber, "Removing multiplicative noise by douglas-rachford splitting methods," J. Math. Imaging Vision (accepted).

Claims

1. A method for reconstructing a three-dimensional surface in a domain wherein the three-dimensional surface is reconstructed from a two-dimensional sketch comprising at least one stroke with a vector field assigned to the at least one stroke and the method comprises the steps of:

interpolating the vector field to form a dense vector field; and

reconstructing the surface using the dense vector field;

wherein interpolating the vector field comprises minimizing a non-linear functional of the vector field with a curl-free constraint on the dense vector field, the non-linear functional of the vector field involving terms defined over the domain and a term defined over the at least one stroke, and wherein the non-linear functional of the vector field is minimized using an augmented Lagrangian method.

2. A method according to claim 1, wherein interpolating the vector field further comprises using a weighted combination of a TV regularization method for preserving sharpness of the reconstructed surface and a H^1 regularization method for preserving smoothness of the reconstructed surface.

3. A method according to claim 2, wherein interpolating the vector field further comprises:

setting a weight for the sketch based on a desired texture of the surface to be reconstructed; and

interpolating the vector field by weighting the combination of the TV regularization method and the H^1 regularization method using the weight.

4. A method according to claim 3 wherein the step of setting the weight for the sketch further comprises the sub-steps of:

painting the sketch in a color associated with a variable alpha value; and

varying the alpha value to set the weight for the sketch.

5. A method according to claim 2, wherein the method further comprises the step of dividing the sketch into a plurality of portions and interpolating the vector field further comprises the following sub-steps for each of the plurality of portions:

setting a weight for the portion based on a desired texture of the surface to be reconstructed for the portion; and

interpolating the vector field in the portion by weighting the combination of the TV regularization method and H^1 regularization method using the weight set for the portion.

6. A method according to any of the preceding claims, wherein reconstructing the surface comprises using a weighted combination of a TV regularization method for preserving sharpness of the reconstructed surface and a H^1 regularization method for preserving smoothness of the reconstructed surface.

7. A method according to any of the preceding claims, wherein reconstructing the surface comprises minimizing a non-linear functional of a height map using an augmented Lagrangian method, the non-linear functional of the height map involving terms defined over the domain and a term defined over a further domain in which prior information regarding heights is present.

8. A method for reconstructing a three-dimensional surface in a domain wherein the three-dimensional surface is reconstructed from a two-dimensional sketch comprising at least one stroke and the method comprises the steps of:

assigning a plurality of vectors along the at least one stroke to form the vector field; and

reconstructing the three-dimensional surface in the domain by a method according to any of the preceding claims.

9. A method according to claim 8, wherein prior to assigning the plurality of vectors along the at least one stroke, the method further comprises the step of classifying the at least one stroke according to a surface geometry to be reconstructed from the at least one stroke.

10. A method according to claim 9, wherein the step of classifying the at least one stroke further comprises classifying the at least one stroke as one of a regular stroke, a ridge stroke, a valley stroke, a jump stroke, a dip stroke and a bump stroke.

11. A method according to any of claims 8 – 10, wherein assigning the plurality of vectors along the at least one stroke further comprises the sub-steps of:

assigning a plurality of initial vectors along the at least one stroke wherein the plurality of initial vectors are orthogonal to the tangents of the at least one stroke; and

5 modifying the plurality of initial vectors to form the plurality of vectors assigned along the at least one stroke.

12. A method according to claim 11, wherein the sub-step of modifying the plurality of initial vectors further comprises modifying one or both of the magnitudes and directions
10 of the plurality of initial vectors.

13. A method according to any of claims 8 – 12, wherein the at least one stroke indicates that a jump is to be formed in the reconstructed surface, and the directions and magnitudes of the plurality of vectors assigned along the at least one stroke
15 respectively represent a direction and magnitude of the jump to be formed.

14. A method according to any of claims 8 – 12, wherein the at least one stroke indicates that a slope is to be formed in the reconstructed surface, and the directions and magnitudes of the plurality of vectors assigned along the at least one stroke
20 respectively represent a direction and gradient of the slope to be formed.

15. A method according to claim 9 when dependent on claim 3, wherein the step of setting the weight for the sketch comprises setting a weight for the at least one stroke and the method further comprises the following step prior to the step of interpolating
25 the vector field:

modifying the weight for the at least one stroke based on the classification of the at least one stroke.

16. A method according to any of the preceding claims, further comprising the steps of:

30 modifying the sketch by adding at least one further stroke to the sketch;

locating a bounding rectangle around the at least one further stroke in the sketch; and

modifying a portion of the reconstructed surface corresponding to the bounding rectangle in the sketch.

17. A method according to claim 16, wherein the at least one stroke further comprises a stroke lying within the bounding rectangle and modifying the portion of the reconstructed surface corresponding to the bounding rectangle in the sketch further comprises the sub-steps of:

5 assigning a further plurality of vectors along the at least one further stroke and the stroke lying within the bounding rectangle;

 interpolating the further plurality of vectors to form a further dense vector field;
and

 modifying the portion of the reconstructed surface by reconstructing it from the
10 further dense vector field.

18. A method for generating an animation sequence from a two-dimensional sketch comprising at least one stroke, the method comprising the step of:

 creating a plurality of frames for the animation sequence by iteratively
15 reconstructing a three-dimensional surface from the two-dimensional sketch by a method according to any of claims 8 – 17 wherein the assigned plurality of vectors along the at least one stroke are different for each iteration.

19. A computer system having a processor arranged to perform a method according to
20 any of claims 1 – 18.

20. A computer program product such as a tangible data storage device, readable by a computer and containing instructions operable by a processor of a computer system to cause the processor to perform a method according to any of claims 1 – 18.