

References

System Overview:

- [1] H. Lipson and M. Shpitalni, Optimization-based Reconstruction of a 3D Object from a Single Freehand Line Drawing, CAD, vol. 28, no. 8, Aug. 1996
- [2] L. Egli, et al., Inferring 3D Models from Freehand Sketches and Constraints, CAD, vol. 29, issue 2, Feb. 1997, pp. 101-112,
- [3] Y. Guiard, Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model, The Journal of Motor Behavior, 19(4), 486-517, 1987
- [4] E. Sachs, A. Roberts, and D. Stoops. 3-Draw: A tool for designing 3D shapes. IEEE CG&A, pp. 18-25, Nov. 1991
- [5] M. Deering, The Holosketch VR Sketching System, Comm. of ACM, vol. 39, no. 5, May 1996, pp. 54-61

Prototyper (clay modeling, iso-surface of a scalar field):

- [6] R. Zeleznik, K. Herndon, and J. Hughes, SKETCH: An Interface for Sketching 3D Scenes, SIGGRAPH'96, pp. 163-170, Aug. 1996
- [7] J. Bloomenthal, An Implicit Surface Polygonizer, Graphics Gems IV, San Diego: Academic Press, Inc., 1994
- [8] W. Lorensen and H. Cline, Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm, Siggraph, Jul. 1987

Refiner (magnetic pen-tip, deformable surface):

- [9] S. Han and G. Medioni, Triangular NURBS Surface Modeling of Scattered Data, IEEE Vis'96, 295-302, Oct. 1996, San Francisco
- [10] M. Eck and H. Hoppe, Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Types, SIGGRAPH'96, pp. 325-334, Aug. 1996
- [11] V. Krishnamurthy and M. Levoy, Fitting Smooth Surfaces to Dense Polygon Meshes, SIGGRAPH'96, pp. 313-324, Aug. 1996

AutoTracer (tensor field, streamball, streamline, stream surface):

- [12] L. Forssell, Visualizing Flow Over Curvilinear Grid Surfaces Using Line Integral Convolution, IEEE Vis'94, pp. 240-247, Washington D.C., Oct. 1994
- [13] M. Brill, H. Hagen, et al., Streamball Techniques for Flow Visualization, IEEE Vis'94, pp. 225-231, Oct. 1994
- [14] A. Parkin, Some Problem of Singular Points and Boundary-conditions in the Sketching of Flow Nets, Geotechnique, vol. 44, no. 3, Sep. 1994, pp. 513-518
- [15] T. Delmarcelle and L. Hesselink, Visualizing Second Order Tensor Fields with Hyperstreamlines, IEEE CG&A, July 1993, Vol. 13, No 4, pp. 25-33
- [16] T. Delmarcelle and L. Hesselink, The Topology of Symmetric Second-order Tensor Fields, IEEE Vis'94, October 17-21, 1994, Washington, D.C., pp. 140-147
- [17] L. Hesselink, Y. Levy, and Y. Lavin, The Topology of Symmetric, Second-Order 3D Tensor Fields, IEEE Trans. V&CG, pp. 1-11, Vol. 3, No. 1, Jan.-Mar. 1997
- [18] G. Guy and G. Medioni, Inference of Surfaces, Edges and Junctions from Sparse 3D Points, IEEE Int'l Symposium on Computer Vision, Florida, Nov. 1995
- [19] G. Guy and G. Medioni, Inferring Global Perceptual Contours from Local Features, Int'l Journal of Computer Vision, vol. 20, no. 1/2, pp. 113-133, 1996
- [20] M. Lee, Using Tensor-fields As a New Framework for Several Computer Vision Problems, PhD thesis proposal, CS Dept., U. of Southern California, Jan. 1997

4 Summary

We have described a two-handed 3D “sketching” system using a 3D stylus, and we focus on modeling by digitizing an existing object. First, the user simply sketches a few strokes over the object to obtain a 3D prototype; then when the user randomly sketches more strokes over the object and specifies creases and corners, the 3D model will deform to follow the details and the edges and corners will align with the sampled creases and corners. The system can also automatically perform spatial reasoning from unstructured fragmented sketches to infer smooth surfaces and extract creases and corners, so that the tedious manual sampling of creases and corners can be avoided.

Technically, the prototyping module uses equipotential surfaces of a scalar field to produce clay-like models; for the local refinement module, scalar or vector potential fields are used to generate attractive forces to deform the shape; for the automatic inference module, tensor fields are used to trace surfaces and singular features (creases and corners).

5 Future Work

The internal surface representation is triangular splines, whose good properties in arbitrary triangulation and local subdivision makes it more flexible to model general surfaces than the rectangular splines. Spline models involve much fewer control points than the polygonal models in describing smooth surfaces, and thus are advantageous for iterative minimization and interactive editing. We have used TriBezier, TriB, TriNURBS for open and spherical surface modeling [9]. For arbitrary topology surfaces, we now use triangular Bezier surfaces, but will soon upgrade to triangular B-splines [10,11]. After the model is refined, some flat regions may have redundant triangles, and a mesh decimation is necessary to simplify the model. The decimation can also yield a multiresolution representation, which is useful for viewing and animation.

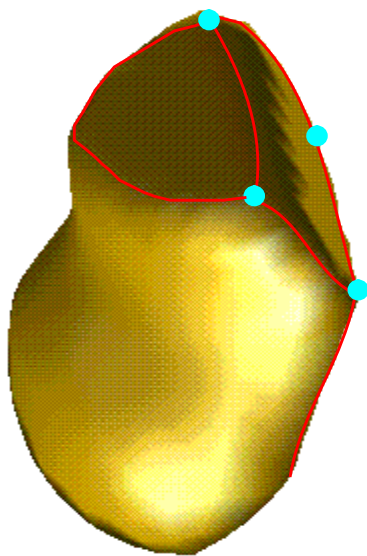
For the user interaction, we now use a 2D mouse for menu selection and 3D rotation. We have recently ordered a 6-degree-of-freedom space ball, which will provide more ease of use for the left hand operations. Some company has attached a laser beam and sensor at the pen tip, so that the object will not be damaged by the touching of the sharp pen tip during sketching. More hand gesture recognition techniques will be applied to further reduce the time necessary for looking at the screens for menus and icons. With a see-through head mounted display, the user could directly see the 3D model imposed on the real object, and find the regions where more refining strokes are needed, and a transparent display which does not block the user’s view may be helpful.



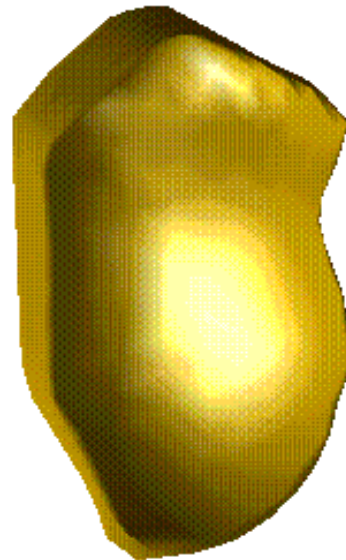
(a) 3D strokes (view1)



(b) 3D strokes (view2)



(c) result (view A)



(d) result (view B)

Fig. 14. autoTracer result on a piece of wood part

Fig. 14 (a), (b) show 3D strokes sketched over the smooth regions of a wood part. Note that we do not trace any edges or corners. The autoTracer module infers potential fields for surfaces, edges and corners. A sphere is given by the Prototyper module, and the Refiner module is called to deform the C^0 TriBezier prototype and align the edges and corners. Finally the edges and corners are automatically marked, and the TriBezier surface is upgraded to TriB surface which is C^1 everywhere except along the edges and at the corners, then the Refiner module is called once more for the TriB surface [9].

Fig. 12 (a) is the mass density after the second pass diffusion, and (b) is the stroke curves traced by searching for locally darkest positions. The intersection positions (degenerate points) are also marked.

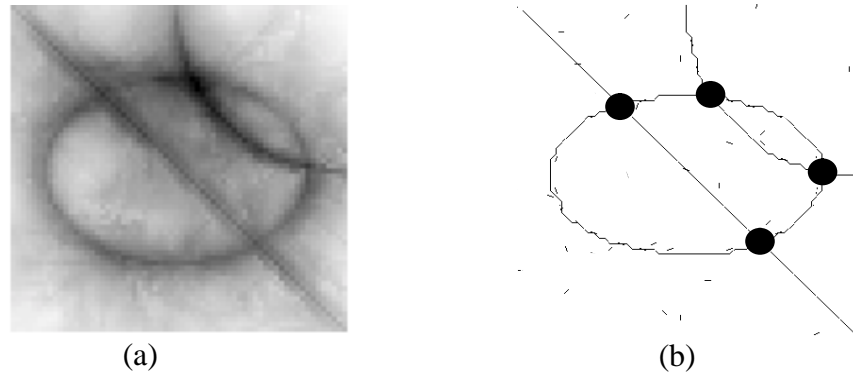


Fig. 12. (a) diffused ink; (b) traced streamlines with intersections marked

In 3D space, inkspots or ink streamlines diffuse to yield a 3D tensor field. At each position the tensor is represented by three eigenvectors which depict an ellipsoid. If the ellipsoid is stick-like, the position belongs to a surface since there exists a unique major normal direction; if it is plate-like, this position is along an edge since there are two major normal directions; for a blob, this position is a corner due to the three conflicting stream surface normal directions. See Fig. 13 for an intuitive illustration.

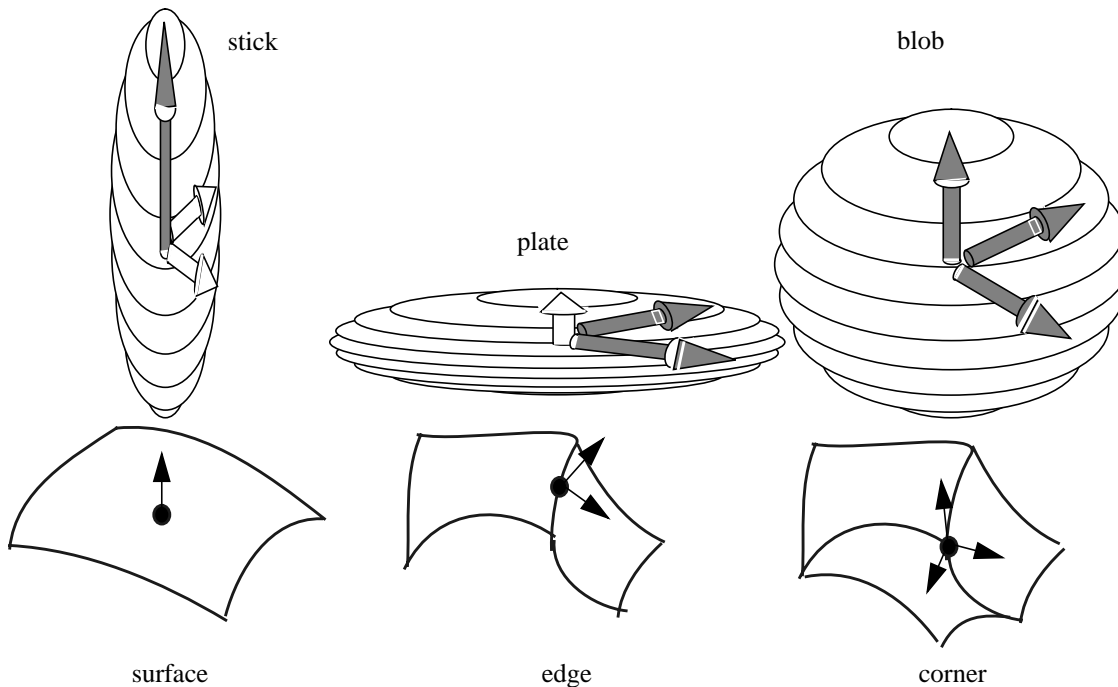


Fig. 13. Eigen analysis of the 3D tensor field produced by diffusion

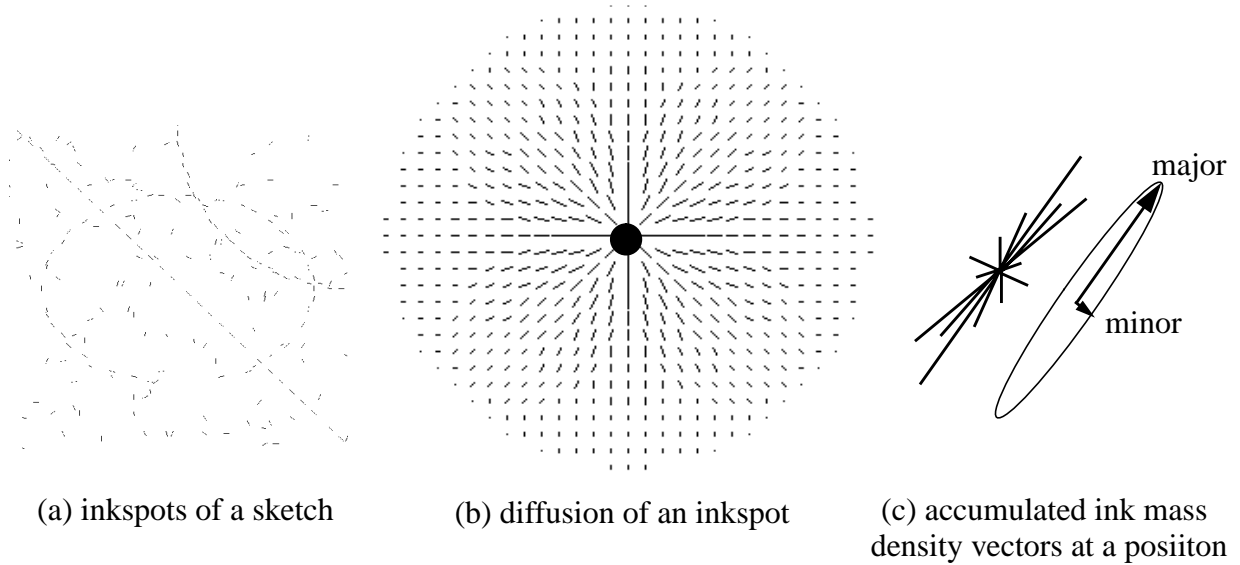


Fig. 10.. Inkspot diffusion

After the above diffusion, we obtain an estimated stroke direction (curve tangent) at each position, and we can further improve the diffusion result by a second pass of diffusion, but this time with an oriented diffusion: the mass density decays with distance and also decays with the offset angle from the tangent. If the tangent direction is very certain (the ellipse is very thin), a linear diffusion pattern in Fig. 11(a) is used. If the tangent direction is not very certain (round ellipse), the stroke at this position is either very curved or at a corner/intersection, so a curved diffusion pattern is used, see Fig. 11(b).

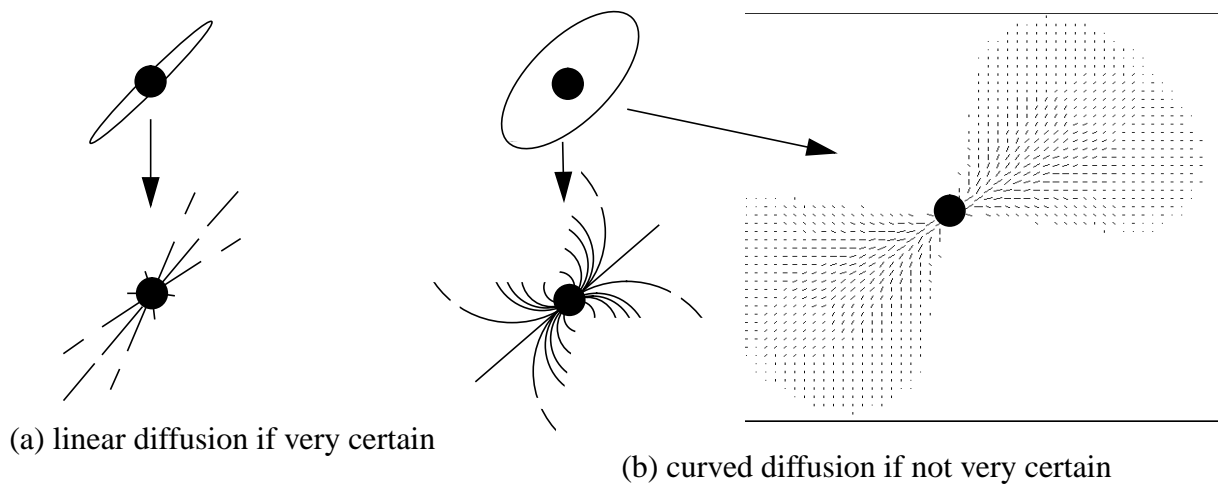


Fig. 11. Second pass inkspot diffusion

allel and the user can change stroke directions while sketching, so the recorded data are just unstructured strokes. To obtain a geometric model of the surface, the programmer's task is to diffuse these streamlines into a stream surface. In addition, the crease edges and corners between these smooth stream surfaces are to be extracted automatically. In other words, the user easily sketches streamlines of smooth steady ocean waves, and the programmer's work is to interpolate a seamless ocean wave surface, and also tell where complicated shock waves and whirlpools will appear.

To be as general as possible, we only use the sampled unstructured points and diffuse them into a stream surface. This is equivalent to diffusing rain droplets on a car's hood into a shallow water stream surface and thus obtaining the surface model of the hood. Since we are using a 3D "pen", we'd like to call each data point an inkspot (similar to "streamball" in [13]). To simplify the principles, we first explain the diffusion procedures in 2D, that is, to diffuse inkspots on a sheet of paper into streamlines and detect the corners as well. If the user sketches slowly, the streamlines are readily available by simply connecting sequential inkspots and fitting them with a B-spline curve, and then we can skip the inkspot diffusion step and directly diffuse the streamlines into stream surfaces. Details on the algorithms can be found in [18, 19, 20, 9].

Fig. 10 (a) shows some inkspots with added noise for testing. At each inkspot, the mass density is the largest and the grayscale the darkest (black). Then as an inkspot diffuses, its mass density decays with distance and the ink darkness decays also, see (b). For a position on the paper, since the ink reaches the position from a specific direction, physical measures such as mass density, velocity, and kinetic energy density are all vectors. For each position, all nearby inkspots can diffuse and reach this position with different mass density along different directions. That means that each position accumulates many mass density vectors, as depicted in (c), and thus each position contains a tensor and the paper becomes a dense 2D tensor field. This is similar to the stress tensor field inside a solid, where each position receives stress forces of different strength and directions from all nearby positions [15,16,17].

Mathematically, we compute at each position a 2x2 covariance matrix (or called second-order moments, or scatter matrix) of all accumulated vectors, and the two eigenvectors defining the major and minor axes of an ellipse, as seen in (c). A large thin ellipse indicates an ink stroke passing through this position, and the major eigenvector gives the tangent direction of the stroke; by contrast, a large round ellipse have two salient directions, indicating an intersection of two or more strokes. More specifically, the major eigenvalue yields an absolute measure of ink mass density, and the ratio (eccentricity) tells whether this position is a stroke curve position or an corner/intersection position, whereas small eigenvalues indicate positions far from the inkspots.

3.3 autoTracer: Automatic Inference of Creases and Corners

3.3.1 User’s View: Sketching Over Smooth Regions Only

In the “Prototyper” module, very few strokes are needed to define a crude clay model. In the “Refiner” module, the user quickly and randomly sketches more strokes to improve the model. But, tracing and aligning the creases and corners is still tedious. Valley creases are easier to trace, but the ridge creases are difficult to trace. Just try to trace the edge of a desk with a pencil and feel how difficult it is. The pencil will slide off the desk very often. Touching a few points on the ridges is easier but much slower if the ridges are curved. With the “autoTracer” module, the user casually and quickly sketches over the smooth surface regions, then the computer will automatically infer creases and corners between these smooth regions, and the user does not need to manually trace the edges or mark the corners. Then the Refiner module is used to perform the energy minimization which deforms the smooth surface and better aligns the creases and corners. Fig. 9 (a) shows the data points produced by sketching over a banana, and (b) shows the surface with automatically detected and aligned crease edges. The user does not have to trace the edges, making sketching much easier. The inference, however, runs for 3 minutes and we are thinking about improvements to make it faster. Researchers in flow visualization realized that drawing the steady flow is easy, but the singular points and boundaries are problematic [14]. The autoTracer module infers singular features from regular data points, and reduces the difficulty.

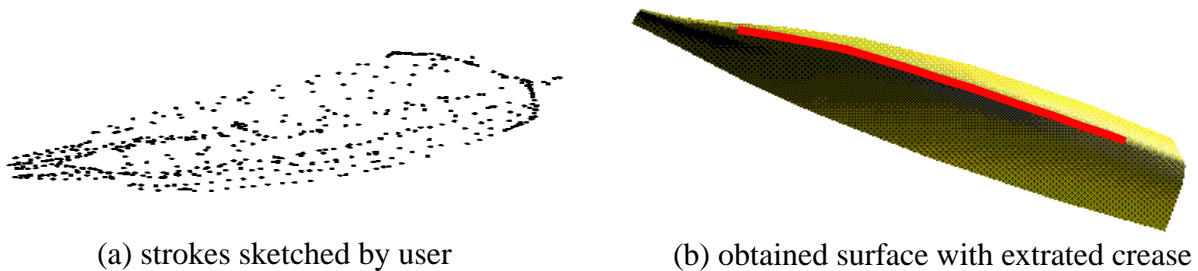


Fig. 9. Strokes over a banana and the result

3.3.2 Programmer’s View: Inkspot and Streamline Diffusion

From the programmer’s view, the problem is to reconstruct smooth surfaces from the scattered points or curve segments, and also to localize creases and corners, if any. If the user sketches slowly, the line segment between two sequential points may provide a good estimation of the tangent of the surface, and a stroke (i.e., a polyline or a fitted B-spline curve) provides a “streamline” over the object (imagine that the user is sketching over a streamlined car hood). However, if the user sketches very quickly, the sampled data points are far apart, and connecting them into a polyline, or fitting them to a spline curve may not yield truthful streamlines of the surface, so more strokes are to be added to obtain enough samples. To make the sketching easy, these strokes do not have to be par-

3.2.2 Programmer’s View: Potential Fields and Energy Minimization

Since the clay parts are solid models (blobs, plates, sticks) and lack local details, we now treat the prototype as a surface model for local refinement by adjusting the positions of control points. The triangle mesh is updated to a quadratic triangular Bezier surface. Each data point contributes a negative spherical potential field which decays with distance, and thus three potential fields are generated by surface points, crease points and corner points, respectively. Energy minimization is performed to deform the surface and align its edges and corners. Using potential fields makes surface fitting fast enough for interactive modeling. In Hoppe et al.’s work [10][11], the distances from data points to the triangle mesh is computed many times during the energy minimization iterations, and the program runs for hours. By contrast, the potential field is only computed once for each data point, and is incrementally updated with new sketching strokes.

The energy to be minimized is defined as follows:

$$E = E_{smoothness} + E_{surface} + E_{creases} + E_{corners}$$

At first, the surface is a quadratic Bezier spline surface which is at least C^0 by sharing control points between adjacent triangles, but we still need the smoothness energy to minimize the mesh roughness. The smoothness energy is defined in terms of the first-order derivatives [9]. The surface energy is for surface fitting. Minimizing the crease/corner energy aligns the creases/corners and significantly reduces the total energy, or cooperatively improve the surface fitting precision, and we thus do not have to subdivide the triangles into many tiny ones to obtain a good fitting if misaligned edges and corners are present. After the TriBezier surface is refined, creases and corners are marked and the model is upgraded to C^1 triangular B-spline (TriB) surface with preserved edges/corners (TriBezier is a special case of TriB), one more time of energy minimization is performed to obtain the final surface [9].

We use the Levenberg-Marquardt algorithm with numerically estimated gradients of the energy and take the steepest descent direction in iterations (in physics, the gradient of a potential field is a force field). To maintain interactive speed, the active triangles should be kept as few as possible, so the user must move the pen locally, instead of traversing a long stroke over the object. After a region is refined, the pen can be moved to another local region.

We have recently found that vector potential fields provide better convergence than the scalar potential fields, at the cost of more storage and computation. The reason is that some position may receive conflicting forces from nearby data points. Using vector potential field with radial directions for each data point will cancel out the conflicting forces in the accumulated potential field, making it possible to attract the surface toward correct directions more efficiently.

3.1.2 Programmer's View: Scalar Fields and Iso-surface Extraction

Clay models are simulated as equipotential surfaces (or iso-surfaces) traced from a generated scalar field in 3D space. For example, if a fire is placed at $(0,0,0)$, the temperature at any position may be defined by $t(x,y,z)=f(x^2+y^2+z^2)$, where $f(d)$ is any decreasing function and we use $f(d)=2(d/R)^3-3(d/R)^2+1$ (for $d<R$), and $f(d)=0$ (for $d>R$). Then, given a specified temperature value T , the iso-surface is defined by $t(x,y,z)=T$, which is a sphere. If one more fire is put at a nearby position, the new temperature field will be the summation of the two fields, and the iso-surface $t_1+t_2=T$ will become a peanut-like shape. But for the user, it works as if two soft clay blobs blended into a peanut-like shape. Different field functions define different shapes of clay parts. We use ellipsoids, blocks, and cylinders. Furthermore, if ice (instead of a fire) is put in the space, the negative temperature field will result in a cavity, a hole, or a tunnel. Also, a clay part only blends with parts in the same group. For example, a finger is allowed to blend with the palm but will not blend with other fingers.

3.2 Refiner: Adding Details to the Prototype

3.2.1 User's View: A Magnetic Pen-tip

Since the prototype only gives a crude initial model, the user still needs to improve the surface by adding more details. This is similar to the previously mentioned ellipse sketching procedure, where more local modification strokes are added to improve the boxing-up prototype. In our system, the user randomly sketches over a region where corrections are needed, and the surface will adapt to the new strokes to yield a better shape. The user can also trace the crease edges and mark corners on the object for the model to align with. Finally a smooth surface with preserved and aligned creases and corners will be obtained. The user feels as if the 3D pen has a magnetic pen-tip which attracts the surface to desired positions. See Fig. 8 for a demonstration.

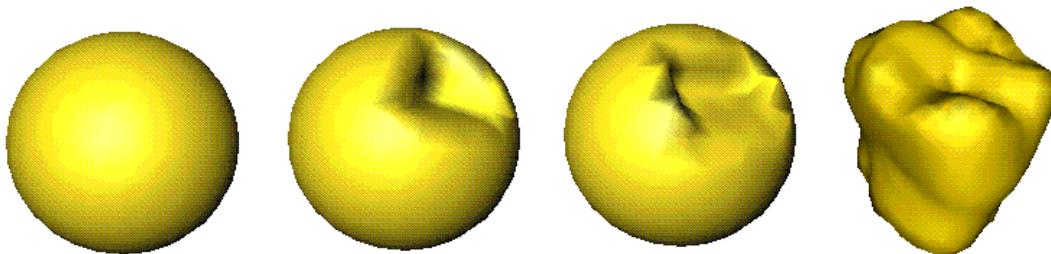
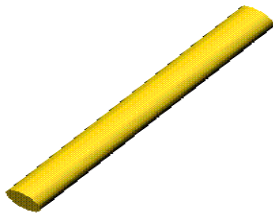
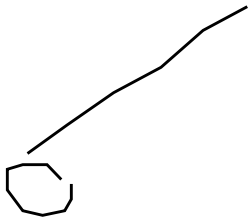
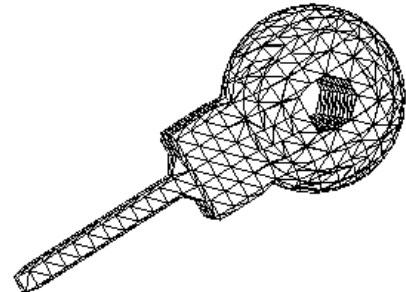
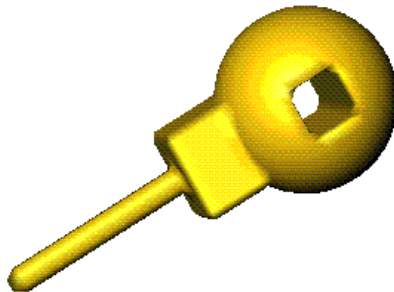
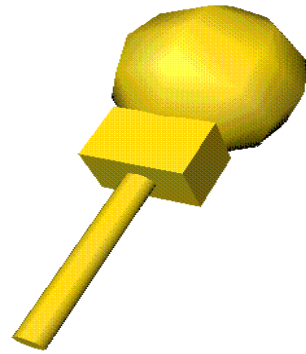
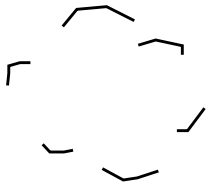
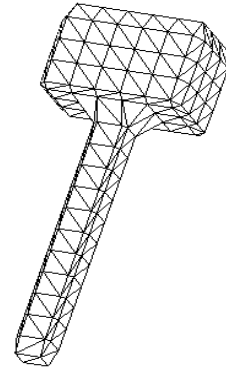
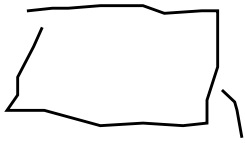


Fig. 8. Refining a blob into a tooth



The smart pen knows what the user wants to draw, so menu/icon clicking is not needed.



3D strokes

soft clay model

alternate displays

Fig. 7. Incrementally build a clay prototype by sketching a few strokes

In next Section, the principles and the implementation details are given. The work is summarized in Section 4 and the future directions are discussed in Section 5.

3 Three Modules of the System

3.1 Prototyper: Boxing-up and Outlining the Shape

3.1.1 User's View: Clay Modeling by Hand Gesture Recognition

In the prototyping module, the user rapidly builds a 3D model giving a crude shape but with the correct topology of the object. We implemented an efficient volumetric clay modeling scheme based on hand gesture recognition, and the program automatically result in a regular dense mesh from the clay pieces. Thus, the user does not need to manually trace the dense mesh as in the currently commercial packages.

An object is either a simple entity or an assembly of several parts, and a part is approximately a blob, a stick, or a plate, and each part can be globally deformed to produce a more general part. For example, a straight stick can be bent to produce a handle of a teapot, then the handle may need further local deformation to reflect the shape details of a specific handle. To digitize a human head, a single sphere prototype can be produced by just clicking 4 points on the head, then the sphere can be globally tapered, and then locally deformed in the refinement module. To digitize a hand, we can specify the palm, the thumb, and the four fingers separately then glued them together to give a single triangular mesh for later local refinement. A finger may also be prototyped as a cylinder, or as a few shorter jointed cylinders, or as a few glued blobs. Compared with surface patch method, volumetric clay modeling is more efficient and there is no gaps or holes in the parts or the objects, and it does not require Boolean operations in the composition.

Gesture recognition is used for fast modeling without clicking the menus or icons. The recognition rules are as follows: if more than three strokes are sketched, an ellipsoid is generated; if only two strokes, the closed (or almost closed) stroke is taken as cross-section and classified into a square or an ellipse in 3D space, and the other open stroke is used as the sweeping axis for the cross-section. If both strokes are closed, a negative part is recognized, i.e., a hole is produced. Of course, the user can always override the recognized properties by manually clicking the icons; for example, the user can sketch a closed curve as the cross-section of a hole, and then sweep along the hole to specify an axis, then changes the default “positive” property to “negative”. Fig. 7 shows an example. Note that the mesh is produced automatically. Using current commercial softwares, the user has to manually trace such a mesh, which is very slow and tedious.

ing an existing object. Some subtle differences may exist between these applications. For example, a sphere can be designed by clicking at the center then stretching an initial sphere to a desired radius. But in digitizing, the probe can never move inside a solid object to specify a center. For digitizing, one display view is enough, but for modeling and visualization of real data, multiple display views may be necessary for 3D alignment.

To digitize curved, especially organic shapes, current systems require that the user trace very dense and regular meshes on the surface. Such work is tedious and time-consuming (8 hours are reportedly needed for an experienced modeler to manually complete a surface mesh by digitizing a toy cat). Our system allows the user to casually specify the most important features and let the computer to fill in the regular gaps.

Textbooks on sketching teach the students to first box-up the objects to draw, outline dimensions and proportions, then refine the details. The simplest example is to sketch an ellipse. First draw a horizontal axis and a vertical axis, then draw a rectangle box, see Fig. 6(a), and small arc segments are drawn at the top, bottom, left and right locations to provide tangent constraints. From this incomplete sketch, we can already perceive a good ellipse (our eyes or brains fill the gaps based on perceptual experience). To finish the sketch, more strokes inside these key points and segments are filled in, and some local modifications may also be made, see (b).



(a) boxing-up, dimensions, proportions (b) local refinement

Fig. 6. Steps in sketching an ellipse

These techniques provide guidelines for our system design. In our system, the user first makes a very crude prototype (a blob, a plate, a stick, etc.); then adds more strokes to specify surface details, edges, and corners, and the crude surface will automatically deform to reach the sketch strokes and the surface edges and corners will also move to align with the specified edges and corners. Since it is difficult to trace the pen along ridge edges on objects (the pen will often slide off the object), we provide an automatic inference module which can infer edges and corners from the user's sketching strokes over the nearby smooth regions, and thus the user is free from the edge tracing tasks. The automatic inference module is based on our group's previous research on perceptual grouping [18,19, 20,9].

of menus/icons may be impractical since there are too many possible gestures in the sketching procedure. In the Quick-sketch system made by Eggli et al. [2], the user sketches over a pressure-sensitive screen with a pen, and the system provides icon highlighting feedback, and user correction is allowed to override the automatic gesture recognition. We use gestures as well as menus/icons in our 3DSketch system, but our sketching device is a 3D pen, and the system has more self-adaptation and spatial reasoning ability.

In other related work, 3D devices are also used to replace the 2D mice. Sachs et al. [4] described a “3-draw” system. The user holds a mirror-like plate in the left hand to rotate the whole screen display, and holds a 3D stylus in the right hand to sketch curves in 3D space. Deering [5] implemented a virtual environment called “holosketch”. Despite the existence of so many 3D modeling software packages, we find that many producers still prefer to start with clay models because clay is malleable and allows artists the full flexibility to manipulate surfaces in ways that are not easy to use or not available in off-the-shelf modeling softwares. Many methods and steps are involved in translating a clay model into a computer model. Our goal is to make the translation more efficient. Our system is special in that it allows random scratching over a region, so that rigorous curve tracing or structured patch meshing is not required. Such random scratching is much easier and faster to use than the conventional curve-based designing schemes. For the latter, the user cannot pause during sketching a curve, and must follow a restrictive sequence to specify a surface mesh, mostly a rectangular mesh which has artifacts in modeling irregular shapes and corners.

Many modeling activities, such as sketching and sculpting, involve both hands. According to Guiard’s psychophysical research [3], the left and right hands often act as elements in a kinematic chain. For right-handed people, the left hand acts as the base link of the chain. The right hand’s motions are based on this link, and the right hand finds its spatial references in the results of the motion of the left hand. Also, the right and left hands are involved in asymmetric temporal-spatial scales of motion: right hand for high frequency fine motion, left hand for low frequency coarse motion. Our 3DSketch system uses this natural division of manual labor by assigning the low-frequency coarse setting of spatial context to the left hand, and the high-frequency fine selection and manipulation operations to the right hand. The left hand provides context by doing menu/icon selection, rotating the global scene, and providing constraint mode. The right hand operates in the frame of reference set up by the left hand, and operates after the left hand’s selection of a command and picking of a tool. The right hand performs finer sampling, sketching and manipulation of the surface geometry.

Although the 3D pen can also be used for designing surfaces from imagination or fitting surfaces to real data such as CT/MRI medical data, we are now just concentrating on modeling by digitiz-

imagine and require special skills to draw [13]. Our goal is to free the sketcher from such difficult tasks and allocate them to computers also. The tasks allocated to the users are limited to some high level decisions and a few essential operations. For scientific applications, the user measures data in smooth steady zones, and the computer infers the properties in the unsteady degenerate points (it would be nice to predict the properties inside a tornado from data measured at far safer positions). Based on this philosophy, we started to develop the 3DSketch system using a 3D digitizing pen. We call the system “3DSketch” since the strokes are sparse, only giving a “sketchy” description of the object. We demonstrate how to reduce the user’s work by endowing more “intelligence” for the 3D pen. The user would feel as if he/she had a “smart pen” that can understand his/her intent from the quick sketch and shows a regularized model, with degeneracy automatically detected and marked.

Since current computer modeling tools require precise structured operations, the tools block the user’s flow of ideas and interfere with the user’s concentration on creativity. Artists and designers still prefer to make freehand sketches on paper with a pen for quick feedback in visual thinking. Some researchers tried to convert such hand sketches to computer models. Lipson and Shpitalni [1] implemented a system to reconstruct 3D models from freehand sketches. The problem with these systems is that the 3D models are inferred from a finished entire 2D sketch, so a lot of intermediate information, such as the orientation of a curve, the sequence between a few curves, the grouping of strokes for a part, and the outer boundary of an object, is already lost. The inference system must perform segmentation, recognition, reconstruction, and representation tasks to interpret the entire finished sketch, and these tasks are often coupled together, making the problem very hard to solve. To avoid such problems, an interactive system is preferred to perform incremental inference while the sketching is being done, and thus the intermediate information can be utilized. In our system, intermediate information is used and timely feedbacks are provided.

3 Interactive Sketching Techniques

Most interactive 3D design packages use curve-based sketching methods, such as sweeping, revolving, and extrusion. These methods usually produce quadrilateral meshes. To make a whole object, several patches have to be aligned, with possible gaps and holes filled by more operations. Since the input device is often a 2D mouse, the user switches between XY, YZ, and ZX planes to draw the curves, and the user operations are specified by menus, icons, keyboard shortcuts and direct manipulations. Zeleznik, Herndon and Hughes developed an interactive system SKETCH [6], which does not use any menus or icons, instead the system automatically infers hand gestures from the 2D mouse strokes to determine the solids or surfaces the user intends to input. Adding some hand gestures significantly makes the system easier and faster to use, but totally getting rid

From the programmer's view, the clay prototype modeling is implemented as iso-surfacing of a 3D scalar field. The local refinement is a procedure of deformable surface fitting and active edge alignment, using energy minimization in potential fields. The automatic tracing of discontinuity edges and corners are based on directional diffusion and topology analysis of a 3D tensor field.

We implemented the user interface using Xt/Motif and OpenGL. We do not use the 3D pen as a menu/icon selection device, since that will require extra motion of the right hand which does not always rest on the desk. The left hand is always resting on the desk, so hand fatigue is not a problem and the menu/icon selections are allocated to the left hand. Also we want to make use of the Motif utilities as much as possible, instead of writing very low level interactions from scratch using the 3D pen. However, we also make effort to reduce the left hand operations by introducing gesture recognition in the system, so that some menu/icon selections are not really necessary.

A "virtual" pen is also displayed on the screen to mimic the actual motion of the 3D pen. The user can look at the pen to check the orientation of the coordinate system, the display mode, or to test lighting and rendering parameters on the pen for fast feedback. If the left button of the foot pedal is pressed, the data digitizing begins. If it is released, the digitizing pauses. Pushing the right button will finish a session of digitizing and back to viewing and modeling mode. If the pen stays static or moves very little, the data recording program will not update the data input, so the user can pause during sketching. Fig. 4 is a color print of the screen shot of the interface.

2 Motivation

We humans can effectively perceive a smooth world and detect discontinuities from noisy sparse scattered samples. Fig. 5 show some imperfect hand sketches, but we easily perceive meaningful industrial parts. There are some very general principles in our human perception processes to

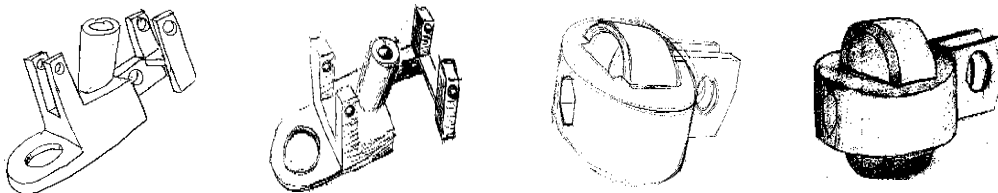


Fig. 5. Sketches made with a pencil on paper

account for the regularities in the visual world. If we can convert such perception principles into computer programs, the computer will automatically infer and reconstruct a complete (or nearly complete) model from a compact set of samples. This can remove or significantly reduce the laborious manual overhead in the dense sampling or rigorous structuring in building computer models. Sketching smooth ocean waves is easy, while the shock waves and whirlpools are quite difficult to



Fig. 2. 3DSketch system set-up

The software has three major modules: Prototyper, Refiner, and autoTracer. With the Prototyper module, the user quickly builds a “clay” prototype by sketching a few strokes. With the Refiner module, the user locally improves the prototype surface and aligns edge/corner features by adding more strokes. The autoTracer module automatically finds creases and corners from the stroke data over smooth regions, so that the user can avoid the tedious tracing of such features in the Refiner module. The automatically traced features are then fed into the Refiner module to improve the model. Fig. 3 shows an example.

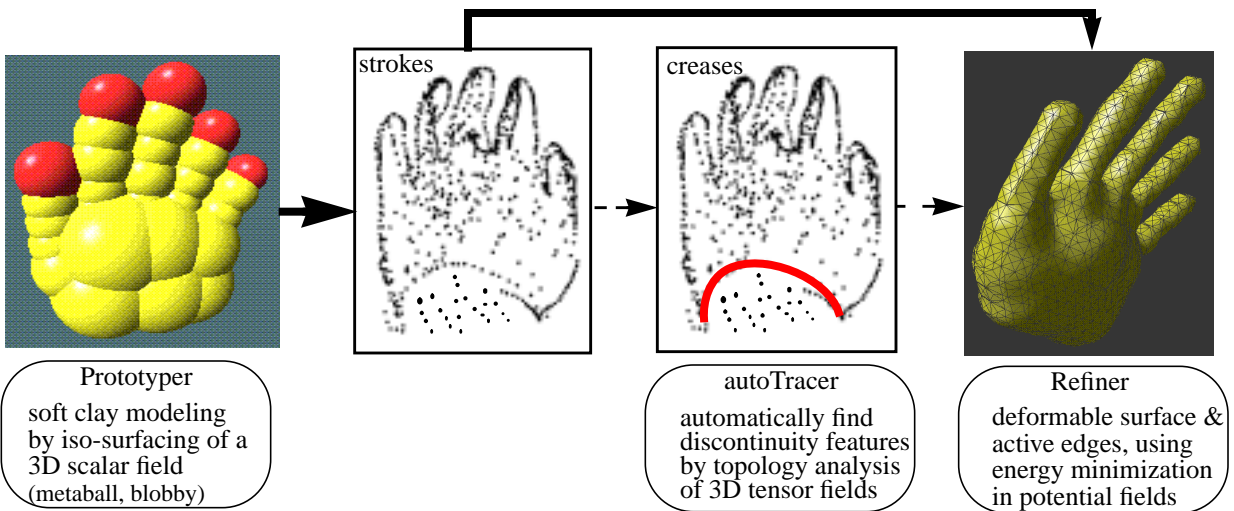
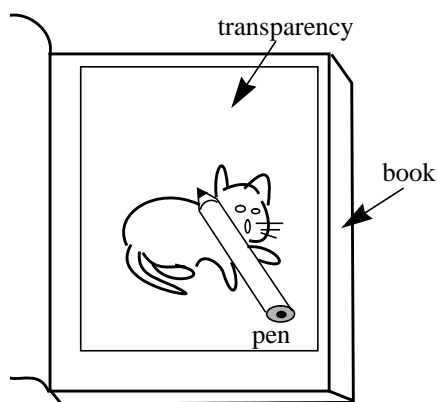


Fig. 3. Three modules of the system

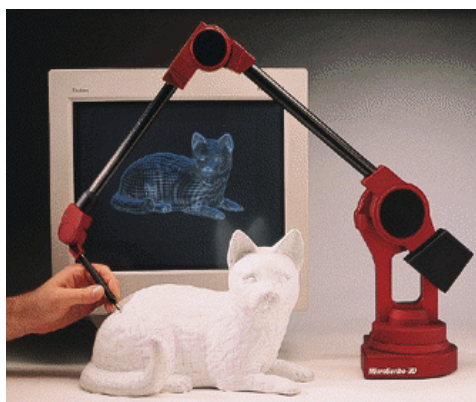
1 System Overview

3D models are fundamental for many applications such as design, analysis, visualization, animation, and multimedia presentation. However, building 3D models is not an easy task. The difficulty comes from three aspects: firstly, the input devices (mouse and keyboard) are two dimensional and lack the freedom for a user to specify and manipulate in three dimensions; secondly, the output devices (screens) are also two-dimensional and do not provide enough cues for exact alignment and depth perception; thirdly, the 3D modeling tasks usually require artistic skills and computer experience which only special groups of people may have.

To tackle the above problems, we use a 3D input device for direct 3D input and manipulations; for the output, we currently use projective display with many depth cues such as floor mesh, shading, occlusion, etc. A stereo display device is preferred and will be used later. To make the modeling easier for a common user with no artistic or computer skills, we base our system on a conceptual model of freehand sketching or copying. In our daily life, not everyone is able to draw a cat on a piece of paper, but everyone can “copy” a cat from a drawing in a book by tracing a pen along the existing strokes, as illustrated in Fig. 1(a). Similarly, to draw a 3D cat, the user can easily trace strokes on the body of a plaster or clay cat model using a 3D pen, see (b). Our system is novel in that the user only needs to draw unstructured strokes and the computer programs will infer structured meshes automatically, and thus the 3D modeling work becomes casual and easy.



(a) 2D: copy a cat from a book



(b) 3D: copy a cat from a plaster model

Fig. 1. Modeling made easier by copying

Our “3DSketch” system can be used for 3D design, digitizing, and data visualization, but we currently concentrate on 3D modeling by digitizing from a real object. The system set-up is shown in Fig. 2, in which an SGI Indy workstation is used for computation and display, a mouse is used by left-hand for menu/icon clicking and 3D rotation, a 3D pen is held in the right hand, and a 2-button foot pedal is used to pause/continue/stop the data sampling procedure.

Rapid 3D Modeling By Casual Sketching On An Object

Song Han and Gerard Medioni

Integrated Media Systems Center and
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273

han@iris.usc.edu medioni@iris.usc.edu
(213) 740-6442 (tel) (213) 740-7877 (fax)

Abstract

We describe a novel system “3DSketch” which demonstrates a two-handed 3D sketching paradigm for 3D modeling by casually digitizing an existing object. The conceptual model of the interface is based on the everyday experience in sketching with a pen on a piece of paper, but in our system, the user holds a 3D digitizing stylus as a 3D pen to sketch in 3D space. We call the pen a “smart pen”, since in the Prototyper module, the user just sketches a few strokes on the object, and immediately sees a 3D prototype made of clay lumps; then, in the Refiner module, when the user adds more random strokes over the object, the prototype surface automatically adapts to follow the pen, and the surface features (edges and corners) align with the user specified ones, as if the pen tip applied magnetic attractive force to the prototype; in the autoTracer module, the user sketches over smooth regions, and the system performs intelligent reasoning to infer smooth surfaces, and also extract discontinuity edges and corners from the user’s inaccurate and fragmented strokes. The internal surface representation is triangular splines (TriBezier, TriB, TriNURBS), whose advantages in arbitrary triangulation and local subdivision make it flexible to model general surfaces. We present encouraging results for 3D digitizing and modeling.

Keywords: 3D Modeling, 3D Digitizing, Authoring Tools, Hand Gestures, Multimodal Input, Human-Computer Interface, Clay Modeling, Deformable Surfaces, Triangular Splines