EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2004)
John F. Hughes and Joaquim A. Jorge (Guest Editors)

Epipolar methods for multi-view sketching

Olga Karpenko[◇], John F. Hughes[◇] and Ramesh Raskar[†]

[◇] Department of Computer Science, Brown University, Providence, RI, USA
[†] Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA
{koa, jfh}@cs.brown.edu, raskar@merl.com

Abstract

Single view sketch-based modelers like SKETCH and Teddy can be powerful tools, but sometimes their inferences are inadequate: in Teddy, for example, if one draws an animal shape, the centerline of the tail will always lie in a plane rather than being curved in 3D. In these cases, multiple-view sketching seems like a reasonable solution: a single sketch gets viewed from a new direction, in which further sketching modifies the shape in a way that's consistent with the original sketch. This paper describes a testbed implementation of such a multi-view sketching approach, based on epipolar lines, which is used for multi-view editing of the "backbone" lines for generalized cylinders. The method works well on many objects—particularly those where precise geometry is not important, but general shape and form must be richer than those with planar symmetry— but is difficult to use in other cases; some difficulties may be related to implementation choices, but we believe that the main problems are tied to the underlying approach, which while mathematically sound proves to be cognitively difficult. We analyze the limitations, and suggest approaches for future work in multi-view sketch-based modeling.
Key words: Sketch-based interfaces, shape modeling.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Modeling packages

1. Introduction

The sketching of 3D shape is becoming a well-studied area, as evidenced by the existence of the present workshop. For smooth shapes, some of the most promising work starts from a sketch of the silhouette of a shape and guesses the 3D geometry through one algorithmic process or another. For linear forms — curves in 3-space — one starts with a projection of the form into some plane and tries to infer 3D shape. All methods must address the fundamental fact that multiple shapes may have identical projections, so no single solution will be right every time. Because a second view of an object very often resolves the ambiguities introduced by projection, it seems as if a second sketch, from a different perspective, might be a useful tool.

We've built a simple multi-view sketching system to test this idea. Our system is designed to support the construction of curves ("backbones") along which generalized cylinders are extruded. The user draws the backbone from one point of view, redraws it from another, and the constraints of epipolar geometry are used to find the curve matching both sketches as closely as possible.

The method works well for certain simple forms, but is difficult to use in more complex cases. The problems are partly algorithmic — our simple method for matching points on the two sketches cannot handle difficult cases — but we also believe that the technique itself is fundamentally difficult to work with, in part because of the unexpected appearance of intermediate views. We discuss the successes and failures of the method, and ideas for future work.

2. Related work

Epipolar geometry has been used to reconstruct scenes in which one can determine point correspondences between multiple views (typically photographs); much of the work in image-based rendering starts from this point of view [MB95, Che95].

Many authors have discussed the issue of inferring shape from 2D marks drawn by a user; approaches vary from

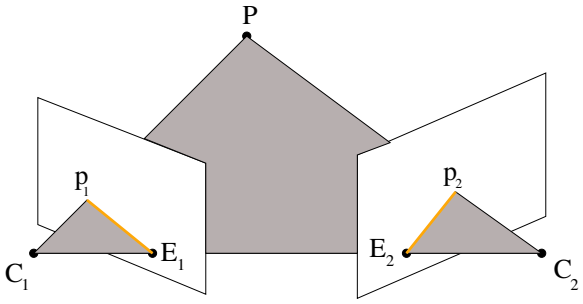

Figure 1: *Given an object point P, two pinhole cameras with the optical centers C_1 and C_2, and two image planes, we can define an epipolar plane (C_1, C_2, P), two epipoles E_1 and E_2 (the points of intersection of the line (C_1, C_2) with the image planes) and two epipolar lines (shown in orange).*

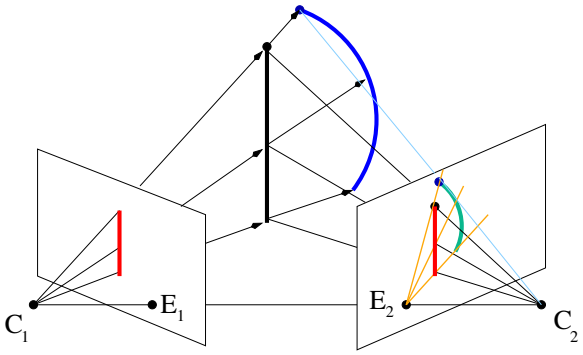

Figure 2: *Sketching a 3D curve using epipolar constraints: a user draws a red line on the left image plane to create a black 3D curve; then she can modify the shape of the black curve by drawing its green projection from a different point of view (on the right image plane). The green curve is projected onto epipolar lines (that are shown in orange and intersect at the epipole E_2). As a result, a blue 3D curve is created. The projection of this final blue 3D curve onto the left image plane remained unchanged because of the epipolar constraints.*

gesture-based systems like Sketch [ZHH96], in which 3D information is part of the gesture's interpretation, to ad hoc inflation algorithms like the "elliptical cross-sections over the chordal axis" of Teddy [IMT99], to constraint-based inference engines in which patterns in the marks (such as parallel lines or perpendicular lines) are used as constraints on the 3D reconstruction process, to systems in which external information like "shadows" can be used to resolve some ambiguities in a shape [CMZ*99].

Describing generalized cylinders by a shape that varies along a backbone curve is by now a standard topic in computer-aided design, well-covered in any textbook; the only (slight) novelty in our formulation is the use of Bishop's framing of a curve [Bis75].

3. Multi-view 3D curve sketching with epipolar constraints

We now describe our simple multi-view editing system, and the typical user-interaction with the system, but we begin with a quick review of the relevant epipolar geometry.

3.1. Epipolar geometry

Consider two images of the same object taken by two different pinhole cameras with centers at C_1 and C_2 . For each point P on the object, there is an *epipolar plane* passing through the centers C_1, C_2 and P (see figure 1).

The lines formed by the intersection of this plane with two image planes are called *epipolar lines*. Each epipolar line is a projection of the ray connecting the object point with the other camera position onto the current image plane. Intuitively, one can think of an epipolar line as a line along which we can move the projection point in the second image while preserving its projection on the first image.

We can determine the $3D$ position of a point P by sketching it from two points of view. Only by making the restriction that the second "sketch" of the point P lies on the epipolar line defined by the first camera position can we guarantee that the resulting $3D$ position of P will be consistent with both views. This idea forms the basis for our multi-view sketching system.

3.2. Multi-view sketching: the user perspective

These simple observations about epipolar geometry above lead us to an interface for $3D$ curve sketching, where a user is given epipolar lines as a guide for modifying the curve from a different point of view (see figure 2). We added this interface to our previous system [KHR02] for drawing free-form animals, and found it to be particularly useful for creating a variety of animal tails and other objects whose shape can be represented by generalized cylinders.

A user typically carries out the following steps (see figure 3):

1. The user rotates the virtual camera with the right mouse button and draws a curve (by dragging the mouse) from any point of view. The user stroke is projected on the plane passing through the world origin and perpendicular to the look vector of the virtual camera.
2. When the user starts rotating the virtual camera to find another view from which to redraw the curve, the system displays the segments of epipolar lines — yellow segments passing through each point of the curve — that serve as a sketching guide.
3. As the user redraws the curve from a new point of view, the new stroke gets projected on the epipolar lines.

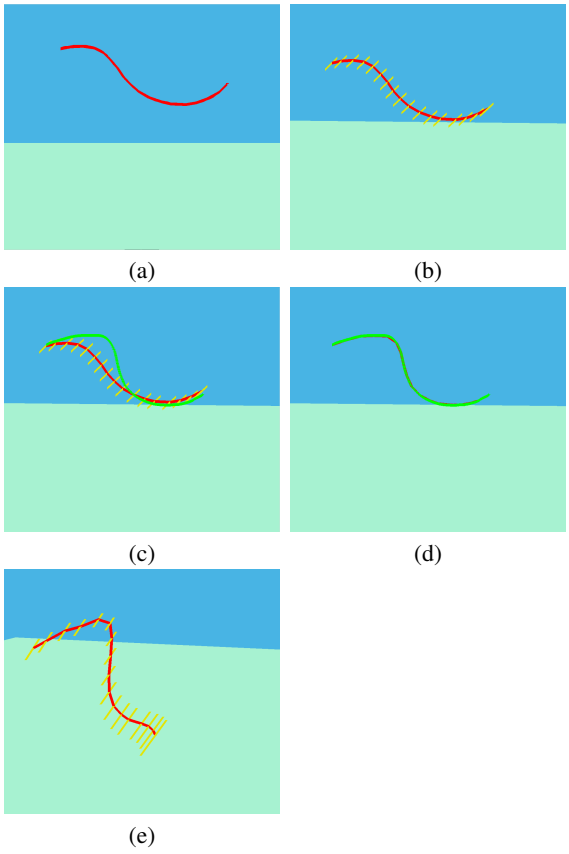

Figure 3: *(a) A user draws a curve on the image plane. (b) When the virtual camera is rotated, epipolar lines appear as yellow segments at each point of the stroke. (c) The user redraws the selected curve from a new point of view by inputing a new green stroke. (d) A new stroke is projected on the epipolar lines and the coordinates of the points of the original curve are updated to satisfy new constraints. (e) The resulting curve from a different angle - notice that new epipolar lines are calculated for each point of the new curve.*

4. Once the redraw operation is complete and the user rotates the virtual camera again, the epipolar lines are updated — they become projections of the rays connecting the previous eye point with the stroke points onto the current image plane. We only keep track of the most recent previous camera position, so the modification of the curve in the current view will only be consistent with the curve projection in the view last used for sketching. The user repeats steps 3 and 4 till a satisfactory result is achieved.

In addition to redrawing, the user can perform the following operations:

- Select and move individual points or segments of the curve along the epipolar lines.
- Select a stroke and translate it along the epipolar lines.
- Place the curves in a hierarchy by attaching them to one another (useful if the user wants to create a wireframe chair, for example).
- "Shear" the stroke by dragging just one of the points. A stroke with the end points A and B can be modified by dragging any intermediate point C along the corresponding epipolar line. The end points A and B will remain fixed and positions of all the points in between will be calculated as the intersections of the plane defined by (A, B, C) and the corresponding epipolar lines.

In practice, we found the "redraw" operation to be the most useful and intuitive.

For each segment of the newly redrawn stroke we need to decide which epipolar line to project it to, because normally it would intersect more than one epipolar line. The algorithm we are currently using is the following:

1. Consider the first segment of the stroke and check whether it intersects the epipolar line corresponding to the first point of the old stroke. If it does not, check intersection with the epipolar line corresponding to the second point of the old stroke, then third, etc.
2. Assume that the segment with the index $i-1$ was projected onto epipolar line with the index j. Then, for a segment with the index i, search for the first epipolar line it intersects starting with epipolar line indexed $j+1$.
3. Skip the point if the corresponding segment does not intersect any epipolar lines satisfying the criteria above.

This simple algorithm seems to work moderately well for curves with relatively low curvature; its drawbacks are discussed below.

After the desired backbone curve is obtained, a user can select two planes perpendicular to the curve and draw cross-section strokes on them; these are then interpolated along the axis to construct the generalized cylinder. To do that, the user clicks near some point on the backbone curve, and a small gray semi-transparent square shows up at this point perpendicular to the curve segment (figure 4). This is the "plane" on which the user can draw a cross-section stroke. The virtual camera (or the curve) can be rotated to adjust the view in such a way that the "plane" is approximately parallel to the view plane. It is the most convenient view to draw a cross-section stroke as there are no distortions as the stroke gets projected onto the "plane" (as the user draws it).

The second cross-section stroke is drawn in the same manner. The system also allows the user to draw just one cross-section; in this case its scaled or non-scaled copy is propagated along the curve. In any case, once two cross-sections are specified, the system needs to interpolate the intermediate cross-section strokes between them.

3.3. Interpolation between the cross-sections

To interpolate between cross-sections, we need a continuous framing of the curve, i.e., a basis for the plane nor-

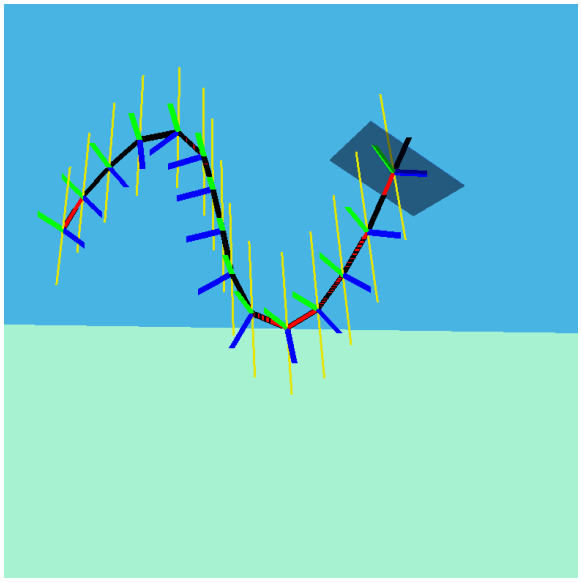

Figure 4: *The Bishop framing along the curve. The blue and green lines are the two vectors of the Bishop frame; the red lines are in the direction of the tangent vectors. Notice that the Bishop frame is continuous even at the inflection between the two bends.*

mal to the curve at each point, but one which changes continuously. The normal and binormal vectors of the Frenet frame [MP77] [DC76] almost work, but at points where the curvature is zero, the normal is undefined, and in practice often "flips" to its negative. The Bishop framing of a curve [Bis75], however, works fine (see figure 4). It is defined by first picking a single unit vector $B_1(0)$ which is orthogonal to the unit tangent vector $T(0)$ there, and then extending B_1 along the curve by the rule that $B_1'(t)$ is a linear combination of $B_1(t)$ and $T(t)$. The second vector in the frame of the tangent plane is then simply defined by $B_2(t) = T(t) \times B_1(t)$. For a polyhedral curve, the vector $B_1(t)$ changes only at vertices; at vertices it is changed by applying the rotation M that rotates the previous edge direction into the next edge direction by a rotation about an axis determined by their cross-product. When the angle between the two direction vectors is zero, the axis is undefined, but the rotation amount is zero, so one uses the identity matrix for M. The sole constraint on this process is that no two sequential edge-vectors of the polyhedral curve be in exactly opposite directions.

With this Bishop framing of the curve, we can interpolate cross-sections easily: we take the starting and ending cross-sections and represent each in the coordinate system of the Bishop frame at the corresponding point; we then interpolate between the cross-section drawings, and place the interpolated cross-sections at the corresponding points of the curve, using the Bishop frame as a coordinate system. In our testbed system, the interiors of cross-section curves are restricted to be star-shaped with respect to the origin: a ray from the origin in direction θ must intersect the curve at a single point whose distance from the origin is called $r(\theta)$. We then interpolate the curves by interpolating between the associated r-functions. More sophisticated approaches, based on 2D curve morphing ([AMZ99], [SG92]), are possible, but were not germane to the testing of the epipolar-manipulation method, so we did not implement them.

3.4. Typical results

Figure 5 shows how our system can be used to model a shape like that of the wire that trails behind the Luxo lamp in "Luxo, Jr.". The curve has a wavy shape in the xy-plane and a bump in the xz-plane; constructing such a curve in Teddy or similar systems would be cumbersome. Figure 6 demonstrates the various tails (with different backbones and cross-sections) that can be created. The mouse and the monkey were generated using our previously presented free-form sketching system [KHR02]. The new interface for sketching with epipolar constraints fits well in the existing framework. By default, a "blob" (a node in the scenegraph containing a blob-shaped object) is created whenever a user draws a stroke, but the user can press a key to activate "curve sketching with epipolar constraints" mode. After the editing of the curve is complete and cross-sections are inputted, the resulting generalized cylinder becomes a "blob" and can be selected, transformed, inserted into a hierarchy and merged with the other blobs as described in the paper [KHR02]. The system is interactive and the interface allows simple shapes to be created in a couple of minutes.

3.5. Problems/ limitations

Figure 7 demonstrates the case on which "redraw" operation fails. Assume that a flat spiral curve was drawn in the xz-plane and a user wants to add torsion to this flat curve in y, to create a three-dimensional spiral curve. When a virtual camera is rotated to any intuitive view for the "redraw" operation, the epipolar lines obstruct the view of each other and confuse the user rather than help. When the user attempts the "redraw" operation, the system does not project a newly drawn stroke on epipolar lines correctly, because it is not clear which point matches which epipolar line. A better algorithm for matching two sketches drawn from different points of view might make the modeling of curves with high curvature easier.

3.6. Simple geometry editing

Suppose that a user wants to draw a simple shape, like the back and seat of a chair; both the back and seat are rectangular, and they are at 90 degrees to one another. Figure 8 shows the process the user employs in the epipolar editing system.

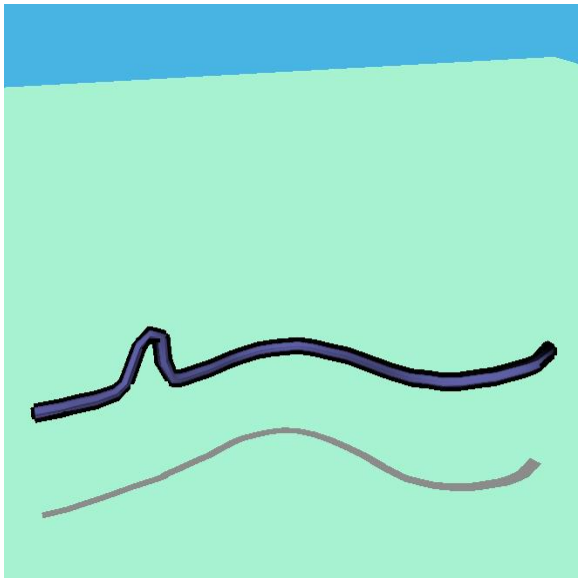

Figure 5: *A generalized cylinder in a shape like that of Luxo, Jr.'s power cord; the small vertical bump is in a different plane from the main curvature of the cord.*

First the user draws the back and seat, but as she rotates the view, it becomes clear that the seat, rather than sticking out towards the user, is actually a parallelogram lying in the first view-plane. It proves to be difficult to get the seat to be perpendicular to the back by re-sketching the sides of the seat. The only view in which this would be easy is one that looks along the edge joining the seat and back . . . but in that view, one cannot properly sketch the new position of the seat! The user tries to do it by using the "shear" operation: the stroke corresponding to the seat of the chair is selected and one of the corners is dragged along the epipolar line in a new view to shear the seat and make it perpendicular to the back of the chair. Unfortunately, shearing is not very intuitive and it is hard to adjust the seat and the back of the chair from just one attempt.

From situations like this, it becomes clear that although a pair of views may completely determine the shape of an object, that determination depends critically on the relative view directions and the orientation of the object itself; a user with a casual camera-positioning tool may not be able to determine how these are related and what the eventual consequences of her actions will be.

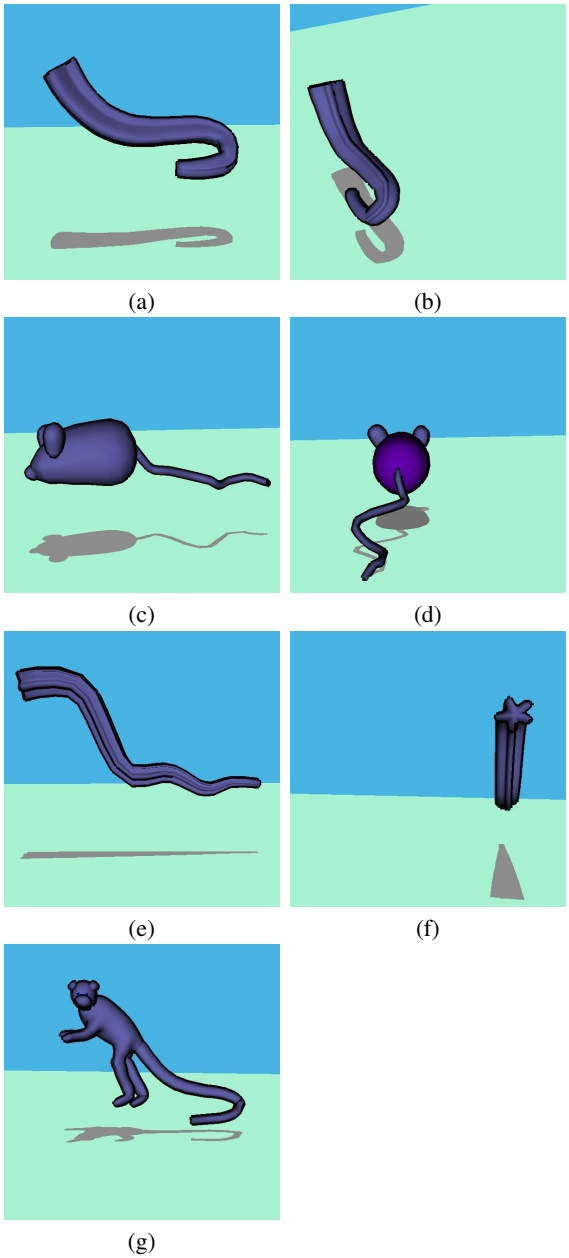

Figure 6: *Examples of the generated generalized cylinders. (a)-(b) The lizard's tail is generated from two cross-sections that are a morph between a circle and a triangle. (c)-(d) The mouse tail is created from two circular cross-sections. (e)-(f) This generalized cylinder shows off the interpolation between flower-shaped and circular cross-sections. (g) The monkey's tail has a simple circular cross-section.*

4. Discussion and future work

Epipolar-based multi-view sketching might prove, in its current form, to be a useful tool in a larger collection of methods, but one would probably not want to build a system based on this idea alone. On the other hand, there may be ways to improve the method by making cleverer guesses of the meaning of the sketch in the first view - there's no particular reason to infer that this sketch is planar, for instance. In

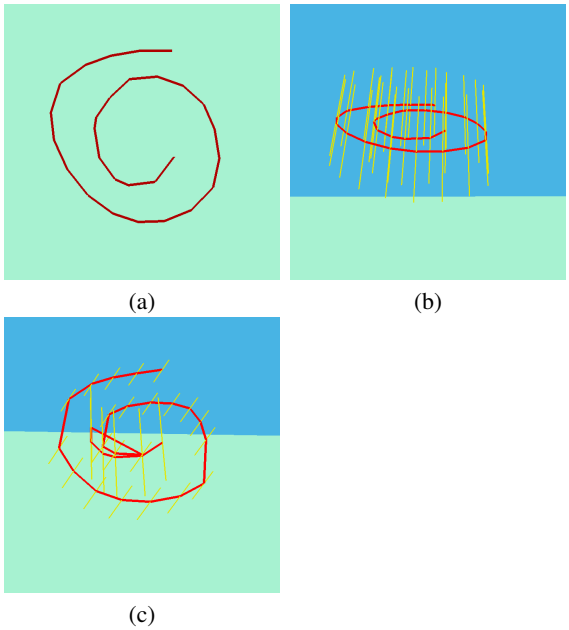

Figure 7: *(a) Assume that the following flat spiral curve was drawn in the bottom plane (xz-plane) and a user wants to add torsion to this flat curve in y. (b) When the virtual camera is rotated to the most intuitive view for "redraw" operation, the system cannot project a newly drawn stroke on the epipolar lines because it is unclear which point matches which epipolar line.*

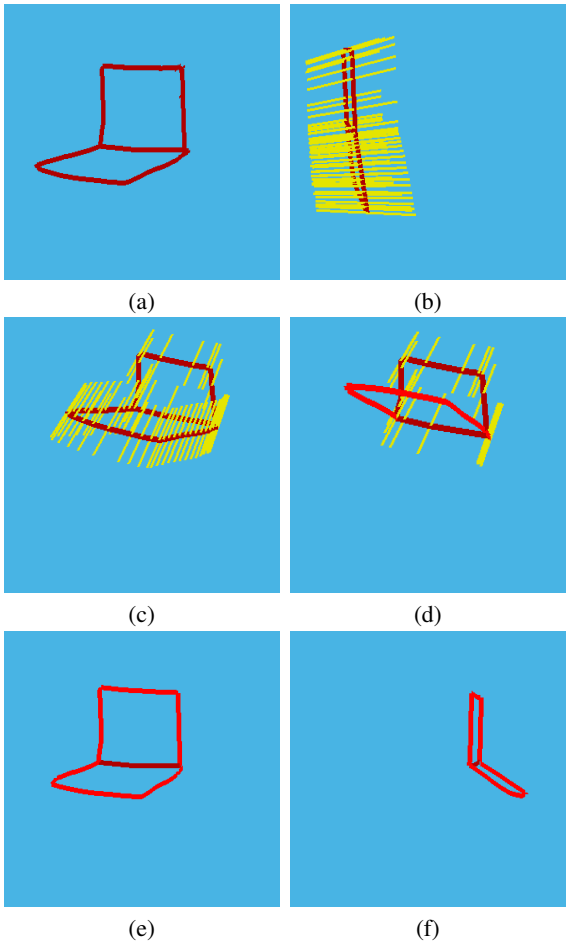

Figure 8: *(a) The user has drawn the back of a chair (as a rectangle) and its seat (as a parallelogram). (b) When the user views it from a new direction, it is clear that the back and seat are actually coplanar. (c)-(d) The user tries to modify the seat in this new view by using a shear operation to make them actually perpendicular, but (e)-(f) the results, seen from another view, show that, although the seat and the back of the chair are not coplanar anymore, they are still not at 90 degrees to one another.*

this section, we discuss the limitations and possibilities for future work on this topic.

Despite the "obviousness" of multi-view sketching, it is surprisingly hard to use in practice. Why is this? The problem, in cases like the spiral above, is partly due to the simplicity of our matching-and-reconstruction algorithm. If it were easier to indicate which points in the new sketch matched which ones in the old sketch, some of these problems would be resolved. But it is part of the nature of *sketching* that we want to make informal drawings, not spend time precisely indicating correspondences. Furthermore, for views that are nearly parallel, the sensitivity of the reconstruction to small changes in the input can be enormous — multi-view sketching works best for nearly orthogonal pairs of views.

As the "chair" example demonstrates, there's a further problem: as the user sketches the back and seat of the chair, she has a clear 3-dimensional idea of what she has drawn. But as the view is rotated, this notion becomes untenable, and the world must be readjusted to make the notion valid again. This cognitive dissonance — between what the user "knew" that she'd drawn and what she sees as the result — makes the user's task one of "fixing the world" rather than "saying what she knows." We suspect that it is this aspect of the experience that most makes our multi-view sketching awkward. Of course, the problem mentioned in describing the chair construction is relevant as well: the particulars of the relationship between two views may have a large effect on the result of a multi-view drawing, but these particulars are difficult for a user to determine, and their effects may not be at all obvious.

The depth-inference used in our system is the simplest possible: all strokes are assumed to be in the view plane until they're redrawn from a new view. If we used some

more sophisticated inference like Lipson's geometric constraints [LS97], the rotated view might be less jarring, as it would be only *slightly* unexpected rather than totally unexpected. Of course, if the user intended to draw a square and diamond in the same plane, the rotated view would be *more* jarring in the latter case. This naturally leads one to consider the question "Which is more *likely*?" It may be that a probabilistic formulation of the problem could form the basis for a fruitful approach. For 3D curve sketching, for instance, it is conceivable that one could create a reasonable prior on curves and then ask for a maximum likelihood estimate of the curve drawn, given that its projection must match the drawing.

The sensitivity of the curve reconstruction when the views are near-parallel hints at a more general problem: since it is difficult to draw the same object from two different views and have the two drawings be consistent, is there a way to treat the two drawings as each providing *evidence* about the shape rather than absolute constraints?

When we consider a related problem — how to perform multi-view sketching of blobby models like those supported by Teddy — we see the issue arise again: if we draw the shape viewed along the z- and x-axes, the y-extent in both views must be the same. And yet a human has no problem in resolving two apparently inconsistent drawings to infer a 3D form, even if the heights of the drawings are slightly different. Once again, a probabilistic formulation may provide a solution: if one treats the drawings as noisy evidence about the shape, some sort of maximum likelihood estimate of the shape (with respect to a prior distribution) may be a good way to infer the shape. Of course, the correct prior is unknown, and depends on the class of shapes being sketched: the shapes of man-made and natural objects evidently have quite different distributions!

References

[AMZ99] AGUADO A. S., MONTIEL E., ZALUSKA E.: Modeling generalized cylinders via fourier morphing. In *ACM Transactions on Graphics* (1999), vol. 14, pp. 293–315. 4

[Bis75] BISHOP R.: There is more than one way to frame a curve. *American Mathematical Monthly 82* (1975), 246–251. 2, 4

[Che95] CHEN S. E.: QuickTime VR — an image-based approach to virtual environment navigation. *Computer Graphics 29*, Annual Conference Series (1995), 29–38. 1

[CMZ*99] COHEN J. M., MARKOSIAN L., ZELEZNIK R. C., HUGHES J. F., BARZEL R.: An interface for sketching 3d curves. In *1999 ACM Symposium on Interactive 3D Graphics* (April 1999), pp. 17–22. 2

[DC76] DO CARMO M. P.: *Differential geometry of curves and surfaces*. Prentice-Hall, Englewood Cliffs, NJ, 1976. 4

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 409–416. 2

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. In *Proceedings of EuroGraphics'02* (2002). 2, 4

[LS97] LIPSON H., SHPITALNI M.: Conceptual design and analysis by sketching. In *Journal of AI in Design and Manufacturing* (1997), vol. 14, pp. 391–401. 7

[MB95] MCMILLAN L., BISHOP G.: Plenoptic modeling: An image-based rendering system. In *Proceedings of SIGGRAPH 95* (August 1995), pp. 39–46. 1

[MP77] MILLMAN R. S., PARKER G. D.: *Elements of Differential Geometry*. Prentice-Hall, Inc., 1977. 4

[SG92] SEDERBERG T. W., GREENWOOD E.: A physically based approach to 2d shape blending. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (July 1992), vol. 26, pp. 25–34. 4

[ZHH96] ZELEZNIK R. C., HERNDON K., HUGHES J. F.: Sketch: An interface for sketching 3d scenes. In *Computer Graphics Proceedings, SIGGRAPH'96* (New Orleans, Louisiana, August 1996), Annual Conference Series, ACM. 2