Analytic Drawing of 3D Scaffolds

Ryan Schmidt[1,2] Azam Khan[1] Karan Singh[2] Gord Kurtenbach[1]
[1]Autodesk Research [2]University of Toronto

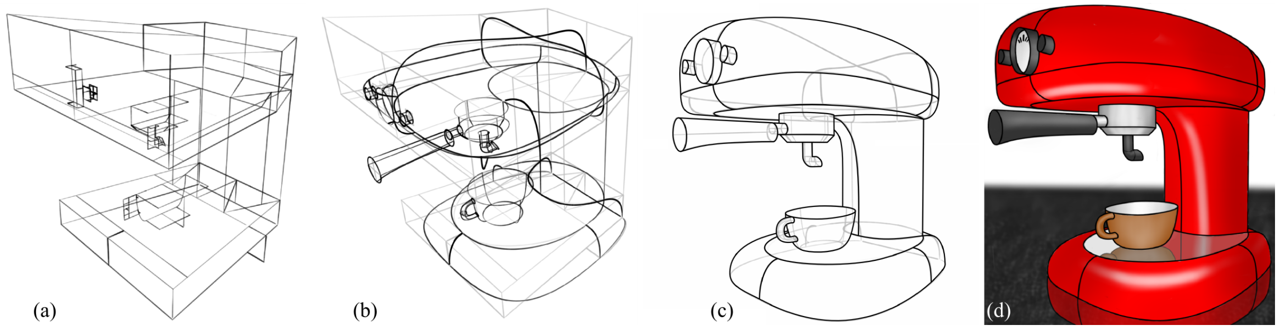

Figure 1: *Our analytic drawing tool infers 3D scaffolds of linear segments (a) from sketched strokes. 3D feature curves can then be sketched by deriving position and tangent constraints from the scaffold (b). After fixing the viewpoint and adding image-space silhouette curves (c), we apply traditional hand-rendering techniques [Robertson 2003] to create a production design drawing of the espresso machine (d).*

Abstract

We describe a novel approach to inferring 3D curves from perspective drawings in an interactive design tool. Our methods are based on a traditional design drawing style known as *analytic drawing*, which supports precise image-space construction of a linear 3D scaffold. This scaffold in turn acts as a set of visual constraints for sketching 3D curves. We implement analytic drawing techniques in a pure-inference sketching interface which supports both single- and multi-view incremental construction of complex scaffolds and curve networks. A new representation of 3D drawings is proposed, and useful interactive drawing aids are described. Novel techniques are presented for deriving constraints from single-view sketches drawn relative to the current 3D scaffold, and then inferring 3D line and curve geometry which satisfies these constraints. The resulting analytic drawing tool allows 3D drawings to be constructed using exactly the same strokes as one would make on paper.

Keywords: sketch-based interactive design, perspective drawing, geometric inference, constraints

1 Introduction

One premise underlying recent work in 3D design interfaces is that 2D drawing is more intuitive than traditional 3D modeling systems, and hence sketch interpretation will lead to more efficient and expressive tools. A frequent approach is to utilize sketches as syntax, incrementally constructing complex 3D models using a grammar of gestural shape editing operations [Zeleznik et al. 1996; Igarashi et al. 1999]. Attempts have also been made to interpret the semantics of sketches, using databases of geometric information like 3D templates [Chen et al. 2008] and junction tables [Karpenko and Hughes 2006]. To support truly freeform 3D design, projections of sketched strokes can be geometrically inverted based on two drawings of each curve [Cohen et al. 1999; Bae et al. 2008].

Although geometric inversion allows virtually any 3D curve to be sketched, Karpenko et al. [2004] and others have observed that it is challenging to "draw what one means" in 3D. This is due both to drawing skill and inherent perceptual biases in estimates of foreshortened shapes and dimensions [Taylor and Mitchell 1997; Schmidt et al. 2009]. Although curves can be corrected in additional viewpoints [Kara and Shimada 2007], a more precise alternative is to utilize *constraints*, such as position and tangent constraints at key points on a curve, guaranteeing that important relationships are satisfied. Explicit representation of constraints as 3D geometry has a long history in variational surface design [Welch and Witkin 1994; Nealen et al. 2007], but constraint specification involves 3D manipulation which, even with a sketching interface, is a difficult and tedious task [Schmidt et al. 2008].

Looking to design drawing guides, we find that designers have developed elegant *analytic drawing* techniques for specifying 3D geometric constraints via lines in 2D [Ching 1997; Robertson 2003; Robertson 2004]. This approach makes extensive use of image-space construction lines to fix the relative depth of vertices, resulting in an unambiguous 3D lattice or *scaffold*. This 3D scaffold greatly simplifies the task of both human and automated sketch interpretation. For example, the only valid line segments which analytic drawing allows are those parallel or perpendicular to existing lines, or which connect known 3D points. By deriving these and other constraints from the current scaffold, we can infer an individual 3D line or freeform curve from one sketch in a single view.

We describe a "pure-inference" drawing interface which understands rules of analytic drawing, allowing designers to directly sketch complex 3D scaffolds and curve networks without recourse to modal tools (Figure 1). Our tool closely mimics the physical motions of pencil-and-paper analytic drawing, while minimizing the drudgery involving rulers and careful measurement (Section 2). Since analytic drawing restricts which lines can be drawn, rough and imprecise strokes can be interpreted with high accuracy. Our inference strategy incrementally fixes strokes in 3D, creating a scaffold which acts as a context for interpreting sketched curves (Section 3). While our tool does allow view rotation, one can draw extensively from a single view, making the experience much closer to natural pencil-and-paper sketching than previous systems. We evaluate the benefits, drawbacks, and limitations of our approach in Section 4, and demonstrate its utility in a variety of design tasks.

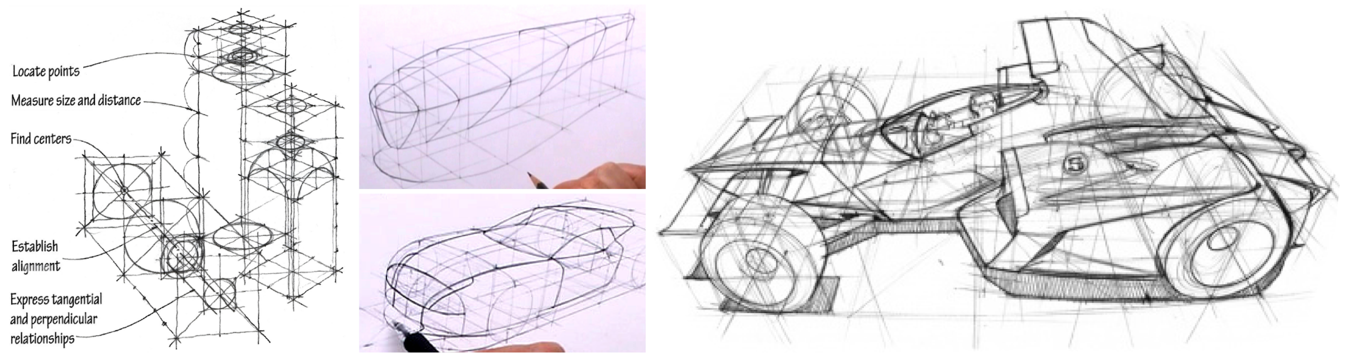
Locate points
Measure size and distance
Find centers
Establish alignment
Express tangential and perpendicular relationships

Figure 2: *Regulating lines define image-space scaffolds which express 3D relationships in a drawing (left, © Francis Ching). The other examples, created by an expert analytic drawer (© Scott Robertson), demonstrate how scaffolds reduce the ambiguity of sketched 3D curves.*

1.1 Related Work

Although constrained drawing has been studied extensively in 2D [Sutherland 1963; Gleicher and Witkin 1994; Eggli et al. 1997; Igarashi et al. 1998], these tools do not understand 3D drawing concepts. Many 3D tools do support interactive constraint-based design in the form of *snapping* [Bier 1990], such as Chateau [Igarashi and Hughes 2001], Sketchup [Google Inc. 2009], and architectural reconstruction tools [Sinha et al. 2008]. These interfaces have minimal support for curves, and hence are much more restrictive than pencil-and-paper design sketching [Do 2002; Bae et al. 2003].

A long-term goal of sketch-based design interfaces is automated interpretation of a complete sketch. Much of the work in this area focuses on drawings of 3D polyhedra with restricted 2D edge-graph topologies [Pugh 1992; Lipson and Shpitalni 1996], although recent extensions support simple curved edges [Varley et al. 2004; Masry et al. 2005; Das et al. 2005; Lee et al. 2008]. Chen et al. [2008] focus on architectural sketching, relaxing prior limitations by utilizing a database of 3D template models. Karpenko and Hughes [2006] support arbitrary silhouette curves with T-junctions, but this technique does not easily generalize to other types of curves. These tools all require relatively clean input drawings, any extraneous construction lines must be removed by some other means. In Section 2 we will show that many of these construction lines implicitly define the semantics that such algorithms are trying to recover.

Interactive approaches interpret strokes as they are drawn. Raster-style 3D drawing can be realized by projecting sketched strokes onto an existing 3D surface, an approach taken in many works [Grossman et al. 2002; Tsang et al. 2004; Kallio 2005; Dorsey et al. 2007]. ILoveSketch [Bae et al. 2008] provides a carefully designed and highly efficient interface to such techniques. Similarly, [Tolba et al. 2001] exploits perspective geometry to allow limited 3D camera movement within a 2D sketch. Generally these techniques limit the space of possible 3D curves to those lying on the original surface. Deforming the surface by manipulating the embedded curves [Nealen et al. 2007] allows more complex 3D curve networks to be developed.

By considering two sketches, the projection of an arbitrary 3D space curve can be geometrically inverted, for example by applying epipolar constraints to sketches from two viewpoints [Karpenko et al. 2004]. In a single view, drawing a curve and its shadow [Cohen et al. 1999] or a symmetric pair of curves [Bae et al. 2008] fixes the 3D shape. As accurate unconstrained drawing is quite challenging, it is often necessary to edit the resulting curves by re-sketching in additional viewpoints [Kara and Shimada 2007], although this still suffers from perceptual foreshortening biases.

1.2 Analytic Drawing

The use of drawing in design processes is well-documented [Do 2002; Bae et al. 2003; Bae and Kijima 2003; Buxton 2007]. In the early *ideation* stages of design, sketching is largely an artifact of visual thinking. Often a designer will generate tens of freehand sketches per hour, most of which are discarded. Once a conceptual design has been selected, final *production* drawings are carefully constructed for presentation to clients or management, and then passed on to trained *modelers* who translate the drawings into 3D models using tools such as Alias Studio [Autodesk Inc. 2009].

Design drawing guidebooks [Ching 1997; Robertson 2003] describe in detail how *drawing systems* facilitate the translation of concept sketches into final drawings. Drawing systems such as *two-point perspective*, *isometric*, and *elevation oblique*, are sets of heuristic rules which allow viewers to accurately interpret straight lines in the drawing. Curves fall outside the rules of drawing systems, but the ambiguity of a curve can be reduced by indicating intersection or tangency constraints with lines that are well-defined.

Analytic drawing is a mechanical process for depicting a 3D form [Ching 1997]. Using a drawing system, the designer first constructs a *scaffold* of *regulating lines* which express 3D relationships in the drawing (Figure 2). These guidelines act as contextual constraints for drawing curves, allowing them to be sketched with enhanced accuracy. [Schmidt et al. 2007] discussed *visual scaffolding* in more general artistic contexts, of which analytic drawing scaffolds can be considered a subset. Figure 2 shows some examples of analytic drawings with the scaffold still intact. We rarely see these drawings because it is common practice to finish production drawings by tracing the feature curves onto another piece of paper.

2 An Analytic Drawing Interface

The scaffolds in Figure 2 act as visual constraints for the designer, allowing 3D geometry to be sketched more accurately. Analyzing such scaffolds, we find that they are composed of lines which are either parallel or perpendicular to existing lines, or connect known 3D points. If strokes are captured in real-time, these rules can be applied to infer the 3D scaffold. Similarly, the position and tangents of points on 3D feature curves can be specified in terms of scaffold vertices and edges, so if a curve is drawn over the scaffold we can attempt to infer the intended constraints. In this section we describe a tool which understands these rules of analytic drawing.

One of our motivations was to explore the utility of a purely inferential interactive drawing tool. While we use traditional UI widgets to provide a drawing/erasing mode toggle, image-space pan/zoom, and 3D pan/tumble/field-of-view controls, the sketching interface closely mimics pencil-and-paper analytic drawing, providing assistance only where it enhances the experience. Figure 3 provides an illustrative example, in which the artist draws *exactly* the same lines as would be drawn on a piece of paper. The only difference is that we provide an initial 3D camera which determines the horizon line, and the first stroke is assumed to lie in the ground plane (to initialize inference). Alternatives such as estimating the camera from a template sketch [Kara and Shimada 2007] could also be used.

The benefit of our tool over traditional drawing is that behind the scenes, we infer 3D geometry from the input strokes to assist with accurate drawing and to allow view changes. Our inference pipeline is relatively straightforward. As each stroke is drawn, we first determine if it is an image-space guideline. If not, we try to infer 3D geometry (Section 3), and if this fails, leave the stroke as currently *free*. When inference succeeds, we add the 3D geometry to an underlying graph and associate it with the stroke (Figure 4). The 3D graph does not differentiate between segments and curves, and embodies what we believe to be the current 3D scaffold. We note that while strokes are initially fixed in temporal order, they are sometimes re-processed when additional local context is added.

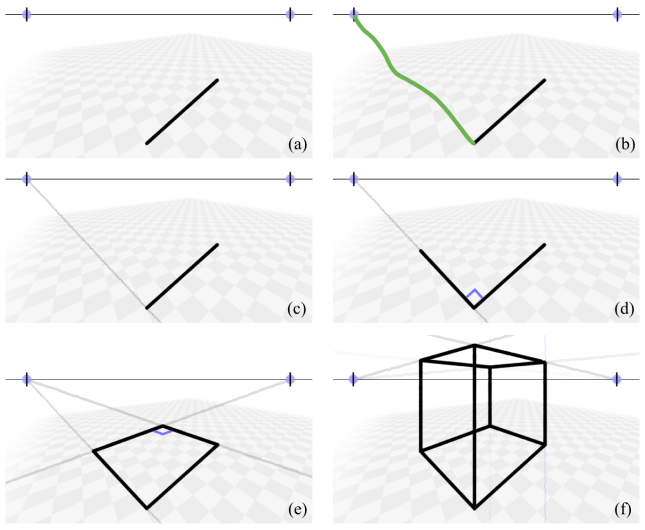

Figure 3: *Embedding the first stroke in the ground plane defines primary vanishing points (a). A stroke to a vanishing point (b) is automatically interpreted as a guideline (c). 3D segments are created by tracing along these vanishing lines (d), eliminating depth ambiguity. Intersecting vanishing lines result in precise angles (e). Guidelines fade out over time to reduce drawing clutter (f). Our inference techniques allow the same box to be drawn without guidelines, assuming the artist can draw sufficiently accurately.*

2.1 Representing Drawings

A *da Vinci Window* is a transparent surface through which a scene is viewed, allowing an artist to create an accurate perspective depiction by tracing contours on the glass. Inspired by this tool, we represent a 3D drawing as a set of virtual da Vinci Windows, which we call *sketch planes*. Each sketch plane represents a fixed 3D viewpoint, and stores all 2D strokes drawn from that viewpoint. 3D geometry inferred from the drawing is loosely coupled with the relevant strokes, but the original 2D data is maintained as the primary representation (Figure 4), allowing inference to be re-applied at any time. For example, after correcting an error, nearby strokes can be automatically re-considered. Similarly, inference techniques developed in the future can be applied to existing drawings.

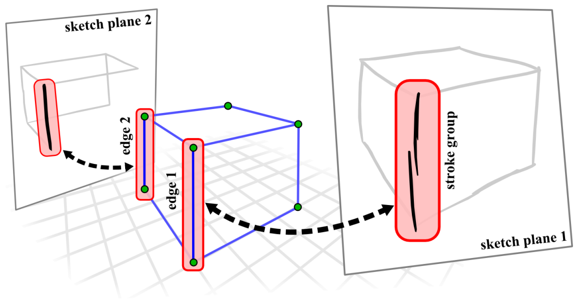

Figure 4: *In this 3D sketch, strokes drawn from two viewpoints are stored on two associated* sketch planes. *Segments of the 3D scaffold graph are dynamically inferred from the strokes, as are stroke groupings. These associations are loose couplings, as the edges and groups may be modified when the drawing changes.*

This representation has many benefits (Figure 5). Unrecognized or *free* strokes fade away when the viewpoint changes, similar to [Bourguignon et al. 2001], but re-appear if their associated viewpoint is selected from an automatic bookmark list. Free strokes can also be transformed with 2D pan/zoom operations, hence they act as view-dependent annotations. If relevant new information is provided, inference is re-applied to free strokes, possibly resolving prior uncertainty. Oversketching and continuation is also supported, but we group strokes rather than merge them, in case later evidence suggests that the grouping was incorrect.

To tune the visual fidelity of the drawing, the designer can interactively interpolate between the original stroke and a projection of the 3D geometry. Again, this is only a rendering enhancement; the original stroke geometry is not modified. Rendering is done in 2D, using high-quality stamp-based raster techniques. The result is much closer to actual pencil-and-paper drawing than anti-aliased line rendering, which is important for designer acceptance of computer drawing tools. Since our strokes are 2D, implementation of an area eraser is straightforward. If within a small pixel threshold, our eraser also "snaps" to the nearest stroke. Inference is re-applied to erased strokes, possibly correcting earlier failures or errors.

2.2 Guidelines

Although every stroke in an analytic drawing can, in some sense, be considered a guideline, actual guidelines are purely functional, and only obscure the drawing once they have served their purpose. Hence, we treat certain types of guideline strokes as a special case, short-circuiting our more general stroke inference. Automatic guidelines fade out over time, limiting visual interference.

The most common guideline is the *vanishing line*, representing a 3D direction which may or may not have a 3D origin. Parallel 3D directions converge at a single 2D *vanishing point*, often lying on the *horizon line*. We locate and display these elements using 2D image-space line intersections. A vanishing guideline is created by drawing a stroke from some known point to a vanishing point, or in the direction of the vanishing point if it is off-screen (Figure 3b,c). Horizontal and vertical guidelines parallel to the image plane, which lack a vanishing point, are also supported. These are not foreshortened and can be used to make arbitrary measurements.

While this approach is very explicit, it is exactly how guidelines work on paper, and hence is easy for designers to comprehend. We experimented extensively with predictive guideline generation, but found that it actually made drawing slower, as the artist would examine the generated guidelines instead of simply drawing the desired one (the latter quickly becomes an almost subconscious act).

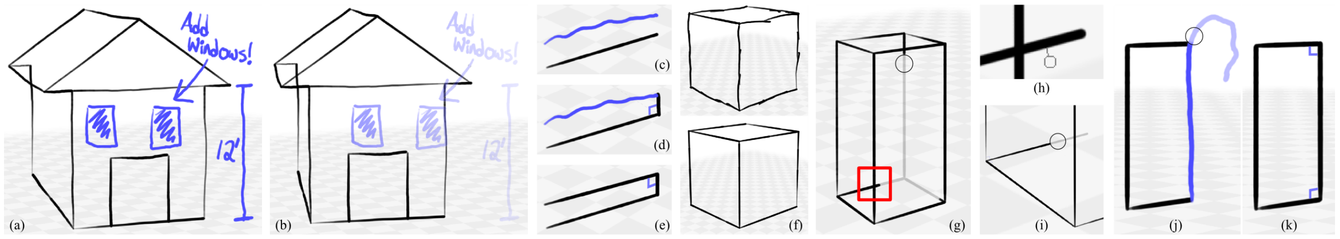

Figure 5: *Our 2D stroke representation allows unrecognized "free" strokes (blue) to be treated as view-dependent annotations (a), which fade away as the view is rotated (b), and can be recovered using automated viewpoint bookmarks. Inference is re-applied to free strokes when lines are drawn nearby, possibly leading to new 3D geometry (c-e). Overlapping strokes are automatically merged when appropriate, and the artist can tune the cleanliness of the drawing (f). An area eraser (circle) can be used to clean up construction lines (g), but is less useful near junctions (red box). Our "smart" eraser snaps to nearby edges within a distance threshold (h), allowing us to erase behind other edges (i). Inference is re-applied to erased edges, allowing drawing errors to be corrected (j,k).*

2.3 Dimension Ticks

Studies in human perception have conclusively shown that we make systematic errors when estimating foreshortened shapes and dimensions [Reith and Liu 1995; Nicholls and Kennedy 1995; Taylor and Mitchell 1997; Schmidt et al. 2009]. Ching [1997] describes a geometric technique for computing arbitrary dimensions based on special vanishing points called *measuring points*. We automated this technique, but discovered that designers were largely unfamiliar with it, and found it as confusing and unintuitive as we did.

Since we have 3D information, our problem is not in actually calculating measurements, but in providing an inobtrusive interface. We draw our inspiration from the "tick-marks" one often makes when visually estimating dimensions. The intersection of a tick-mark and a guideline is located in 3D and snapped to any nearby 3D scaffold points. We then find any intersecting 3D guidelines, and if the 3D distance from the tick to an intersection point is A, we add ticks at distances A and -A from the intersection point along the intersecting guideline (Figure 6). This mechanism supports a wide range of 3D dimensioning tasks. For example, as image-parallel guidelines are not foreshortened, they can be used to "eyeball" arbitrary dimensions, which can then be transferred back to 3D (Figure 6g-i).

3 Inferring 3D Geometry from Strokes

Our task is the inherently ambiguous problem of inferring an intended 3D line segment l corresponding to a 2D stroke s. We are aided by the additional contextual information encoded in the current scaffold, namely geometric constraints that filter the space of admissible 3D segments. Conceptually, our algorithm is similar to that of a standard snapping technique [Bier 1990]. First, we query the current scene (i.e. scaffold) for all potential constraints that l could satisfy. Next we enumerate all possible non-conflicting combinations of constraints, producing a list of constraint sets $\{\mathbf{c}_i\}$ which define potential line segments. Finally, we select the most likely line segment by evaluating a fitness function based on the stroke, constraints, current scene, and our prior assumptions about what designers will draw.

The constraints we infer from the scaffold are hard point and direction snap constraints. Point constraints can be derived from known vertices, intersections, or the nearest point on another line. The last is a parameterized constraint, as the nearest point can vary. Direction constraints are determined from right angles, lines through fixed points, and the directions of other lines in the scene.

In our system all of these constraints may exist at the same point in 3D space. Hence, a segment l can satisfy many different combinations of constraints. This redundancy makes the segment more

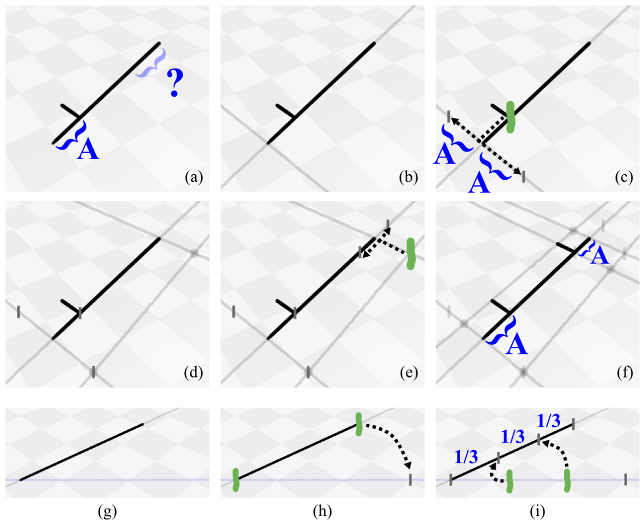

Figure 6: *To copy dimension* A *(a), we first add guidelines (b). A tick-mark over the vertex transfers the dimension* A *to the intersecting guideline, in both directions from the guideline intersection point (c). Additional guidelines (d) and a second tick-mark (e) complete the dimension transfer (f). Image-parallel guidelines (g) are not foreshortened, allowing an edge to be visually subdivided (h,i).*

likely. We also prefer certain types of constraints to others - for example, snapping to an endpoint or intersection is more likely a priori than snapping to any other point along a line. Some lines are also more likely than others, such as those the same length as other lines in the scene. Hence, we define the *fitness* of a line segment as

$$\mathbf{F}(l) = \mathbf{S}(s,l)\mathbf{G}(l)\sum_i \mathbf{C}(\mathbf{c}_i, l) \tag{1}$$

Here $\mathbf{S}(s,l)$ measures the deviation between l and s, $\mathbf{G}(l)$ expresses how well l matches with our prior beliefs about which geometry is more likely. The summation varies over all constraint sets $\mathbf{c}_i$ that are satisfied by l. Then $\mathbf{C}(\mathbf{c}_i, l)$ is defined as

$$\mathbf{C}(\mathbf{c}_i, l) = \begin{cases} \mathbf{C}(\mathbf{c}_i) & \text{if } l \text{ satisfies } \mathbf{c}_i, \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{C}(\mathbf{c}_i)$ is the "weight" for constraint set $\mathbf{c}_i$. Unfortunately this term makes $\mathbf{F}(l)$ highly discontinuous - many constraints are satisfied only at a single point in space. Hence, we must utilize a combinatorial scheme to find the optimal segment l, but our method is easy to implement, and also applies to 3D curve inference.

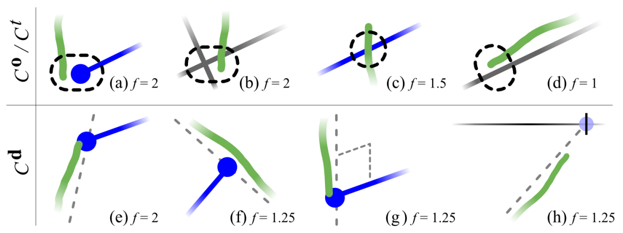

Figure 7: *Our system allows segment endpoints to be snapped to hard position constraints $\mathcal{C}^{\mathbf{o}}$, including scaffold vertices (a), guideline intersections (b), intersections with lines/curves (c), and nearest points on guidelines (d). Potential direction constraint types $\mathcal{C}^{\mathbf{d}}$ include lines through scaffold vertices (e,f), perpendicular directions (g), and directions to existing vanishing points (h). Distance constraints $\mathcal{C}^{t}$ are determined by either one of (a-d), or by the nearest point along the projected direction defined by $\mathcal{C}^{d}$.*

3.1 Inferring Line Segments

Instructing artists on how to draw a straight line, [Ching 1997] states that one should place the pen at the starting point and then focus on the desired end point, rather than track the pen tip. Hence, we consider only the endpoints when measuring the correspondence between stroke and segment. If $\{s_A, s_B\}$ and $\{l_A, l_B\}$ are the 2D stroke and (projected) line endpoints, respectively, then we define $\mathbf{S}$ as the product of Gaussians centered at each endpoint:

$$\mathbf{S}(s,l) \;=\; \mathcal{G}(|l_A\text{-}s_A|,\delta_A)\,\mathcal{G}(|l_B\text{-}s_B|,\delta_B) \tag{3}$$

Here $\mathcal{G}(d,\sigma) = \exp\left(\text{-}d^2/\sigma^2\right)$, and mechanical error is modeled by the *uncertainty radius* $\delta_i = \mathit{Lerp}(v_i/v_{max}, \delta_{min}, \delta_{max})$. The velocity $v_i = |s_{i+1}\text{-}s_i|/(t_{i+1}\text{-}t_i)$ and constants should be DPI-relative, we use $v_{max} = 40\text{in/s}$, $\delta_{min} = 0.1\text{in}$, and $\delta_{max} = 0.5\text{in}$.

We break down the $\mathbf{G}$ term into two factors, the segment length $|l|$ and direction $\mathbf{d}$, each defined as a mixture-of-gaussians based on the segments l_j that already exist in the drawing:

$$\mathbf{G}(l) = 1 + \frac{1}{N}\sum_j \mathcal{G}(|l|\text{-}|l_j|, r_l) + \frac{1}{N}\sum_j \mathcal{G}(1\text{-}\mathbf{d}\cdot\mathbf{d}_j, r_d) \tag{4}$$

where $r_l = \text{avg}(|l_j|)$ and $r_d = 1\text{-}cos(10°)$.

As mentioned, a given constraint set restricts which line segments are admissible. While a segment could satisfy an arbitrary constraint set, we would then need to somehow normalize for the number of constraints. Instead, since $l = \mathbf{o} + t\mathbf{d}$, we limit $\mathbf{c}$ to three constraints $\mathbf{c}^l = \left\{\mathcal{C}^{\mathbf{o}}, \mathcal{C}^{\mathbf{d}}, \mathcal{C}^{t}\right\}$ and define $\mathbf{C}$ as a product of hand-tuned constants f for each constraint type (Figure 7):

$$\mathbf{C}(\mathbf{c}^l) = f(\mathcal{C}^{\mathbf{o}})f(\mathcal{C}^{\mathbf{d}})f(\mathcal{C}^{t}) \tag{5}$$

Having defined each term in Equation 1, we now consider how to find the optimal segment l for a given stroke s. Since constraints exist at well-defined points on the scaffold, we can safely assume that only constraints whose 2D projections are near to s are relevant. Hence, the first step is to collect all possible position constraints $\mathcal{C}^{\mathbf{o}}$ which could be applied to either endpoint of s, based on image-space proximity. Then for each endpoint, we find all possible direction constraints $\mathcal{C}^{\mathbf{d}}$, and for each of those, find all possible $\mathcal{C}^{t}$. The result of this exhaustive enumeration is a list of constraint triplets $\mathbf{c}^l_k$, each of which defines a segment l_k. If the constraints are parameterized, we fix these parameters using nearest points on the stroke. Since many $\mathbf{c}^l_k$ define the same line, we gather the unique segments and evaluate Equation 1, summing over the constraint sets, to find the best-fitting 3D geometry.

We have thus far assumed that the stroke s is exactly the stroke the designer intended to draw. In practice strokes are inaccurate, moreso as one becomes comfortable with the system and draws more quickly. We prefer to think of s as a sample from some distribution of strokes which could have been sketched. We define this distribution independently for each endpoint, as a uniform Gaussian with a deviation defined by our uncertainty radius δ. Then we can sample other possible strokes, find the best segment for each, collect duplicates, and, falling back on our redundancy argument, select the most frequent as the inferred segment. For efficiency, we collect all potential constraints within δ as a pre-computation, then hold the constraint combinations fixed, allowing hundreds of sampled strokes to be considered in a fraction of a second.

We can express the confidence in our choice of line based on an uncertainty metric $\mathbf{F}(l_1)/\mathbf{F}(l_0)$, where l_0 and l_1 are the best and second-best lines, respectively. In Figure 8 we map line uncertainty to red, showing (a-c) that the inference of an otherwise highly ambiguous stroke is resolved by adding 3D guidelines. Another possibility would be to automatically re-apply inference to the highly uncertain stroke when more context was added, such as the other box edges. We have experimented with this, but during interactive drawing it can be confusing. An interface for indicating such automatic corrections to the artist is left for future work.

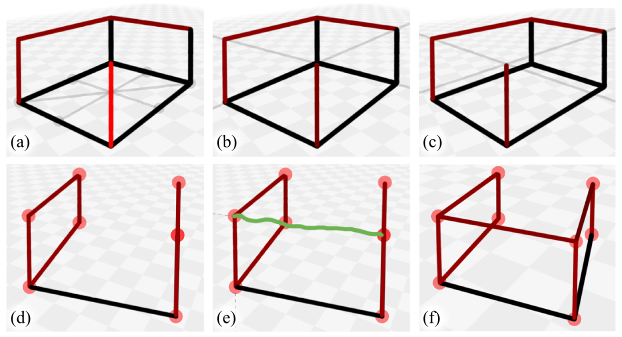

Figure 8: *The diagonal segment in (a) has been embedded in the ground plane but the bright red color indicates high uncertainty. Two guidelines clarify our intent, ensuring that the same stroke becomes a vertical segment (b,c). Often this context already exists, and guidelines are unecessary. For example, the box corners overlap in (d), but our line inference finds the correct result (e,f).*

3.2 Inferring Curves

The principle behind our curve inference is that the strokes designers draw represent imagined 3D curves which satisfy various constraints. In analytic drawing, the scaffold visually depicts many of these constraints. Hence, given a stroke we first infer 3D position and tangency constraints from the scaffold, then construct a curve c which explicitly satisfies these constraints while minimizing reprojection error (Figure 9).

Bae et al. [2003] observed that designers generally constructed complex curves out of inflection-free segments. Hence, we represent c by a multi-segment C^2 cubic Bezier spline, with a segment between each pair of position constraints. Given a set of fixed rays through stroke points in the image plane, we find the best-fit 3D Bezier curve by solving the least-squares optimization problem

$$\arg\min_{\theta,t_i,u_i} \sum_i |c(\theta,t_i) - v(s_i,u_i)|^2 \tag{6}$$

where θ is the set of variables defining the curve, $c(\theta, t_i)$ is a point on the curve at parameter t_i, and v is the 3D point defined by the parameter u_i along the ray through stroke sample s_i. We solve this problem via gradient descent, using the standard *foot-point* approach [Wang et al. 2006]. First θ is held constant and new ray and curve parameters $\{t_i, u_i\}$ are found using the nearest point on $c(\theta)$ to each ray, and then $\{t_i, u_i\}$ are fixed and θ updated. To speed convergence, we fix $\{t_i\}$ to a regular sampling, optimize θ and $\{u_i\}$, then tune the result using the full optimization. The curve variables θ are the 3D Bezier segment endpoints and tangent vectors. Hard constraints are enforced by rewriting components in terms of fewer variables. For example, a direction constraint $\mathbf{d}$ on a tangent vector replaces three variables (x, y, z) in the definition of c with the single-variable point $(t\mathbf{d}_x, t\mathbf{d}_y, t\mathbf{d}_z)$.

Our line inference strategy needs only minor alterations to support inference of c from a stroke s. The same model as in Equation 1 is used, simply replacing l with c. For the term $\mathbf{S}(s, c)$, instead of using only the endpoints, we sample c with a fixed number of points N and project the samples onto the image plane, creating a set of 2D points c_j, and then evaluate a mixture-of-Gaussians:

$$\mathbf{S}(s, c) \;=\; \frac{1}{N} \sum_j \mathcal{G}(d_s(c_j), k_r \delta_j) \tag{7}$$

where d_s determines the distance to the stroke polyline, and δ_j is again the velocity-based uncertainty radius, measured at the nearest point on s and scaled by a constant factor to account for the increased imprecision of freehand curve drawing (we use $k_r = 2$).

As in the segment case, we collect a set of potential position and tangent direction constraints from the scaffold based on 2D proximity (Figure 9), and enumerate all possible combinations. Planarity is explicitly enforced if all constraints are co-planar. Also, if the constraints are mirror-symmetric about the plane perpendicular to the vector between the two endpoints, we generate combinations with and without a symmetry constraint. Hence, the constraint set for a curve is $\mathbf{c}^c = \{\mathcal{C}^{pln}, \mathcal{C}^{sym}, \mathcal{C}^{pt}_1 \ldots \mathcal{C}^{pt}_n\}$, and the term $\mathbf{C}$ is again a product of constant terms. For planarity and symmetry f is 1.5 and 1.1, respectively, if the constraints are satisfied, and 1 otherwise. For each constrained curve point $f = 1.5$ if the point has a tangent constraint, and 1.1 otherwise. To normalize for the number of positional constraints we scale $\mathbf{C}$ by $(1 + \lambda e^{-\lambda(n-2)})$, where $\lambda = 0.25$ controls the falloff of the exponential distribution.

This normalization is approximate, so including an extra constraint point will increase $\mathbf{C}$, even if it causes the curve to vary wildly in depth. Also, the $\mathbf{S}$ term must be tuned to allow for very "sketchy" strokes. Hence, compared to line inference, the quality of curve inference depends much more on the geometry term $\mathbf{G}$. We have experimented with many factors, including arc-length, depth variation, tangent vectors, and so on, but these showed no improvement over the following, based on total absolute curvature:

$$\mathbf{G}(c) = \mathcal{G}\left(\frac{\int_t |\kappa(c,t)| dt}{\kappa_{opt}} - 1, r_\kappa\right) \tag{8}$$

Here $\kappa(c, t)$ is the curvature of c at t and $r_\kappa = 2$. The ideal or optimal curvature κ_{opt} is defined as $\pi t_{\max}/\text{arclength}(c)$, which results in $\mathbf{G}(c) = 1$ for circular arcs.

Lacking a reasonable model of how drawing errors affect sketched curves, we cannot sample the distribution of possible curve strokes as we did for line segments, so we select the curve with the highest value $\mathbf{F}(c)$. Again, the list of other likely curves could be useful in a suggestive interface. Once a curve has been selected we check to see if it is close to an elliptic arc, and if so, snap it to the arc. This allows perfect circles and ellipses to be drawn, which are frequent elements of design drawings [Bae et al. 2003].

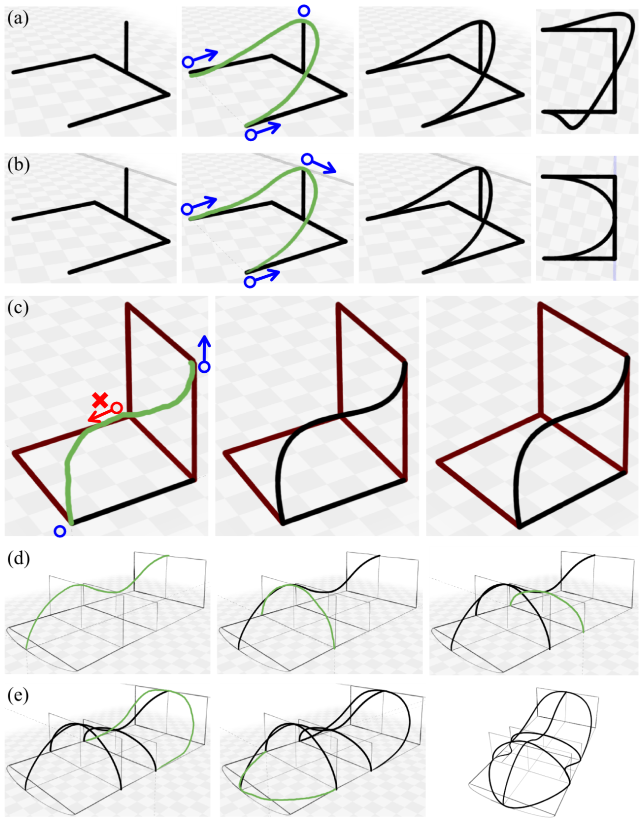

Figure 9: *In row (a), the curve lacks a tangent direction at the middle point, and so the unconstrained 3D shape is optimized to fit the projection. A guideline is added in row (b), fixing the tangent direction, which in turn allows a symmetry constraint to be satisfied. In row (c) an accidental constraint is detected, but ignored by our inference strategy. In (d,e) a series of strokes (green) are drawn over a simple scaffold. Although there are many possibilities, our inference technique picks out the intended constraint sets.*

A limitation of this approach is that the number of curves to be tested grows exponentially as constraint points are added. We observe that $\mathbf{G}$ has a maximum value of 1, $\mathbf{C}$ can be computed without fitting the curve, and $\mathbf{S}$ will always be maximized by unconstrained tangents, thus defining an upper bound on the total fitness of any curves with matching point constraints and further-constrained tangents. After fitting a curve we can compute these upper bounds and discard any matching curves with a maximum possible fitness lower than the current best fitness, often resulting in order-of-magnitude reductions in the number of curves that need be tested. For example, over $390,000$ constraint combinations were generated for the 9-point helix in Figure 10a, but only $1,377$ curves were actually fit. The total fitting time was still over a minute, but would be significantly reduced by proper optimization and parallelization. Drawing the same helix with two 5-point segments (Figure 10b) results in roughly a 2 second delay for each curve.

4 Evaluation

A major advantage of our approach is that since most artists are familiar with analytic drawing techniques, the key concepts and interactions in our interface are immediately understood. We have had several artists successfully draw basic 3D shapes with no training whatsoever. In this sense the tool succeeds at being a transparent in-

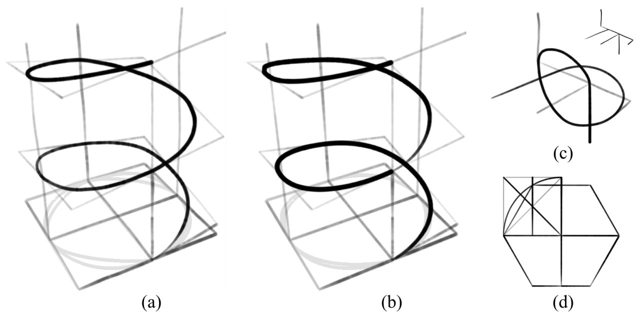

Figure 10: *Complex self-intersecting curves can be drawn in a single-view with the appropriate scaffold. The helix in (a) was created with a single stroke, although incremental sketching is more accurate (b). We can also draw knots (c), and use traditional geometric techniques to construct precise polygons (d).*

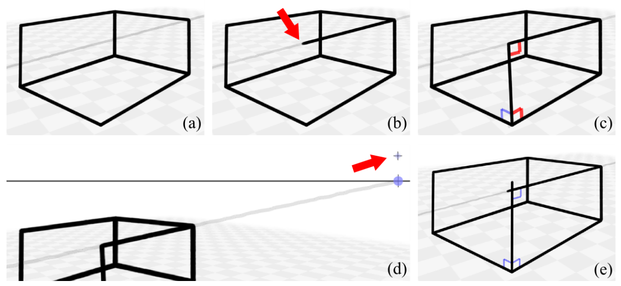

Figure 11: *Tracing along a guideline (a) without a vertical intersection results in an unconstrained endpoint (b). A connecting vertical segment is then skewed from perpendicularity, indicated with red joint markers (c). These misalignments result in additional vanishing points (d) and can propagate through the drawing. An extended vertical stroke snaps to the desired right-angle (e).*

terface to a traditional drawing style. As most artists do not strictly follow the rules of analytic drawing, however, we must inevitably explain implementation details. We have also observed several fundamental conflicts between constraints and freehand drawing, which we illustrate with a few examples.

4.1 Inference Limitations

Line inference works very well in practice. In unambiguous cases the proximity-based $\mathbf{S}(l, s)$ term reproduces the *nearest*-snapping behavior commonly used in CAD tools like Sketchup [Google Inc. 2009]. In complex scaffolds such as those in Figure 15, ambiguity is frequent and nearest-snapping performs poorly. Here our constraint and geometry priors usually allow the intended line to be drawn without needing to hunt for a completely unobstructed viewpoint.

Failures can occur if the artist attempts to take freehand shortcuts. A common example is shown in Figure 11, where snapping to an unconstrained endpoint results in a non-perpendicular edge. These misalignments can lead to problems later in the drawing, such as segments which appear to be co-planar but do not actually intersect in 3D. This particular case can be avoided by adding an explicit vertical guideline or drawing an extended vertical stroke, but this may not be obvious to the artist. Increasing the weight of perpendicularity constraints (Figure 7) leads to the desired result, but also restricts intentional creation of non-perpendicular segments. To communicate this ambiguity, we display temporary visual feedback in the form of traditional "right-angle" markers, which are colored blue for 90° angles and red for near-perpendicular connections.

In general, our experience has been that curve inference usually selects the intended constraint set, but re-drawing may be necessary to find the 2D curve that will produce the intended 3D shape. We found that in most cases this was not due to inference failures, but rather to an inability on the part of the artist to correctly draw the projection of the intended curve. The camera manipulation needed to evaluate the inferred 3D curve can be reduced with ground-plane shadows and temporary indicators of the included constraint points and tangent directions. Once the right set of constraints is found, experimental re-sketching could be avoided with multi-view corrective oversketching [Kara and Shimada 2007] or direct manipulation of the constrained tangents.

Because we must allow for significant fitting error to handle imprecise strokes, our inference strategy tends to prefer curves with a larger number of constrained vertices. Our curvature score (Equation 8) rules out most spurious constraints, although it can also discard multi-segment curves with tight bends, which then must be drawn as individual segments. More problematic is that in some cases spurious constraint points are included that only have a small effect on the fitting score (Figure 12). Since our constraint collection thresholds are in image-space, these cases can often be avoided by zooming in. Alternately, to avoid camera manipulation we have added a simple "scratch-out" gesture which temporarily suppresses vertices from the constraint collection phase.

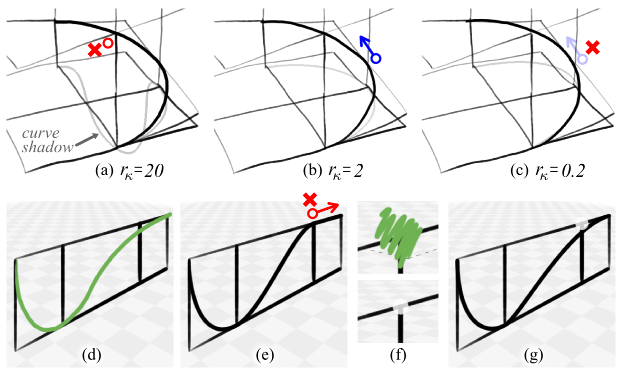

Figure 12: *If the curvature factor r_κ is too permissive, unintended constraints will not be discarded (a), but if it is too restrictive then intended constraints (b) may also be skipped (c). We show ground-plane shadows to better support visual evaluation from a single view. The stroke in (d) results in an over-constrained curve (e) which cannot be resolved by the curvature score. A scratch-out gesture over the offending vertex temporarily exempts it from constraint collection (f), leading to the desired result (g).*

4.2 User Case Studies

We had six users perform pilot experiments with the software - graphics researchers G1 and G2, designers D1 and D2 and architects A1 and A2. All subjects were highly proficient with both drawing and 3D modeling, but only G1 had used similar '3D drawing' interfaces. A2 was a paid participant, and D1 and D2 were contractors at Autodesk Inc. Each session lasted from 2 to 4 hours, some results are shown in Figure 13.

Subjects G1, D1, and A1 were provided with 15-minute introductory training, after which we answered questions but otherwise did not intervene. The artists quickly grasped the basics of line drawing, but had difficulty with curve drawing. During follow-up discussions we found that subjects could not adequately explain to us

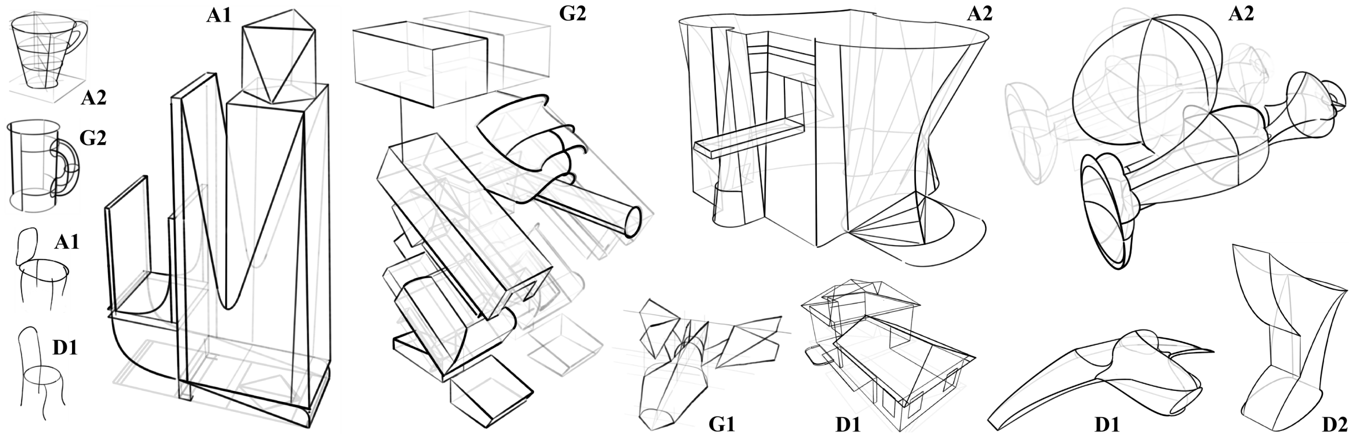

Figure 13: *After simple warm-up exercises (left), subjects sketched 3D drawings of varying complexity, labeled here with the subject number (see text). We have removed most remaining scaffold lines, and manually lightened hidden lines, to make the final form clearer.*

how the system was interpreting their curves. Additional training provided to D1 resulted in a significant improvement in the ability to construct curved forms. Hence, we conducted extended training sessions with subjects G2, D2, and A2. After a short tutorial we had them draw a cube (this required our guideline ticks), a coffee mug, and then one or two more complex drawings. Throughout the session we intervened when the subjects appeared confused although this was rarely necessary after the first hour. With this guidance, subjects learned how to draw curves more effectively, and were able to construct more complex drawings.

Overall, subjects exhibited similar levels of proficiency with most aspects of the the tool within a few hours, and were satisfied with their ability to express ideas. We found that the architects were the most comfortable with the perspective-drawing interface, and clearly enjoyed using it. In a post-mortem questionnaire, subject A2 answered the question "What did you like about the drawing interface?" with "very natural way of drawing" and "I would use this tool for my first design incursions where I would normally be sketching on paper".

The computer scientists and architects both noted that it was difficult to represent intended surfaces with only a few 3D curves. The designers seemed more adept at this, so it may be an issue of experience. The most frequently reported problem was that verifying the 3D shape of inferred geometry and guidelines required view rotation. As most subjects understood our right-angle markers without explanation, we believe additional transient feedback may help to reduce projective ambiguity and minimize the need for rotation. The misalignment issues caused by nearly-parallel vanishing directions (Figure 11) were frequent, and noted by most subjects as a significant issue. Automatic filtering may help here, but perhaps the most promising approach was suggested by subject A2, who wanted explicit control over vanishing directions in the same way as we provided for guidelines.

Analysis of several drawing sessions logs are shown in Figure 14. We note that the proportions of time spent drawing and exploring the 3D form, and of strokes which are guidelines, is relatively similar between subjects and drawings. This suggests to us that during the in-depth training sessions, subjects reached some comparable level of proficiency. However, we found that most subjects did not use analytic drawing techniques in the way we had expected. They were reluctant to leave scaffold segments in the drawing, erasing them immediately after use and preferring to rely on our transient guidelines if possible. This in turn limited their ability to specify 3D curve tangents, as guidelines can only be constructed along existing vanishing directions. Several subjects did point out this problem, and after instruction were able to the sketch tangent scaffolds necessary to draw their intended curves, but they still felt that a more explicit tangent manipulation interface was desirable.

In an effort to determine the capabilities of our tool in the hands of an expert user, the first author trained himself in analytic drawing techniques using [Ching 1997] and [Robertson 2003]. Figure 15 shows some drawings created during two 3-day periods of intensive use. During this extended experimentation we learned how to construct arbitrary tangent directions using small triangles (Figure 15b, inset) and create arbitrary polyhedral scaffold blocks by cutting them out of boxes (Figure 15c). These emergent tools afforded us much more control over curve shape and 3D orientation, suggesting that artists who invest the time to learn analytic drawing will find our approach even more effective.

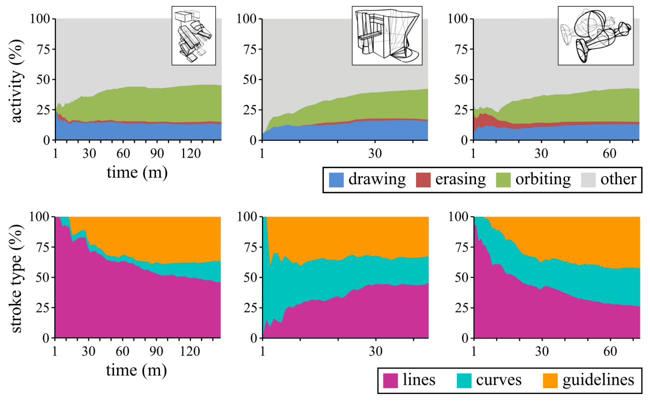

Figure 14: *Proportions of overall activity (top) and stroke type (bottom) accumulated over time, for three drawing sessions (insets). Activity time is determined by pen contact with display.*

5 Conclusions and Future Work

Motivated by pencil-and-paper analytic drawing, we have described an approach to constrained inference of 3D lines and curves from single-view sketches. Our pure-inference interface supports creation of 3D curve networks comparable to those demonstrated in recent works such as ILoveSketch [Bae et al. 2008], but without the

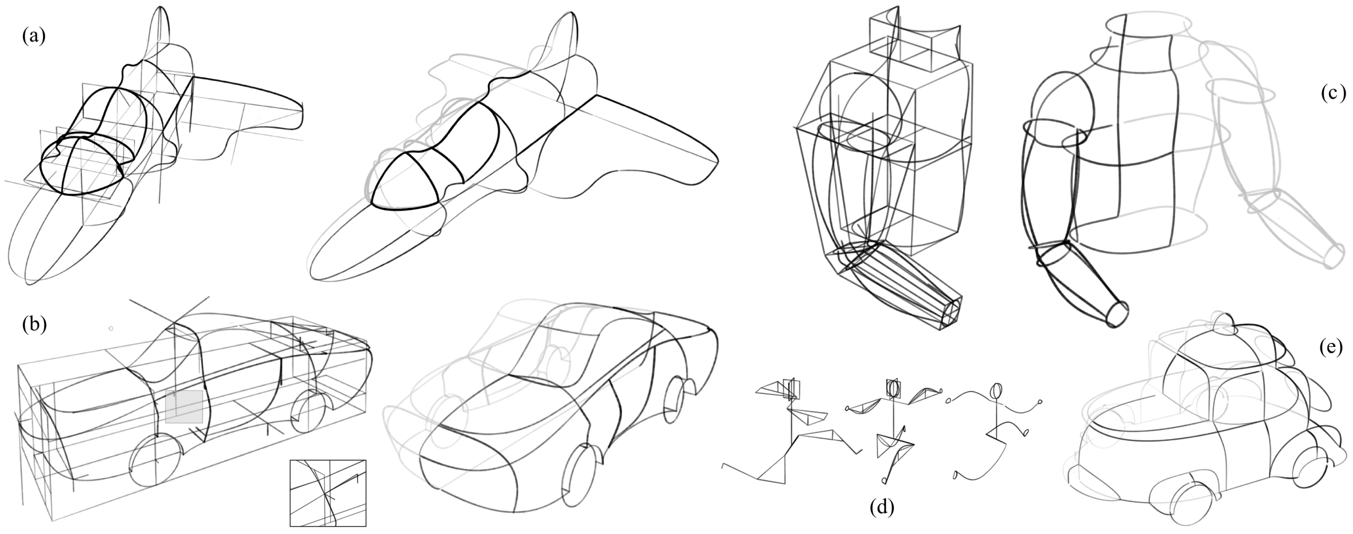

Figure 15: *Examples created by the first author while learning to draw analytically. Early models like the fighter (a, 1 hour) were limited to basic curves contained in well-defined boxes. More complex curves, such as those running the length of the car body (b, 3 hours), were possible once we discovered how to draw arbitrary tangent directions (inset). To create the torso (c, 45 minutes) we followed [Ching 97], creating a full scaffold by 'cutting' arbitrary polyhedra out of boxes and then drawing the feature curves. The stick figure in (d) was sketched without any editing in under 5 minutes, and the taxi (e) in a 30 minute session.*

need to specify a drawing mode or manipulate the camera for each curve. By inferring constraints from a well-defined 3D scaffold, it is more likely that the curve is in the right spatial location, partially mitigating the effects of drawing skill and perceptual errors.

Although artists readily understand that scaffolds must be assembled to draw curves, we have found that learning how to construct a suitable scaffold often requires some geometric ingenuity. The helix scaffold in Figure 10 provides a good example - once known, it can be constructed quickly, but it was only discovered after several failed attempts. The process laid out in design drawing books [Ching 1997] for blocking out or *massing* a shape before adding curves is highly relevant when using our interface. However, freehand shortcuts are often taken on paper, so even design-drawing experts will have an adjustment period as they learn how to construct proper 3D scaffolds. Similar sorts of geometric "best-practices" are common both in design drawing and in 3D modeling, suggesting that users will find this learning process to be tractable.

Even with the constraints that analytic drawing places on the sketching process, comments from architects and designers have been highly positive, and we have found a strong stated preference for drawing from a single fixed view. While we suspect benefits of occasional camera manipulation would quickly be discovered, single-view drawing appears to be highly valued by artists, and may provide a more gradual transition to advanced 3D drawing techniques. Our interface is particularly suited to architectural drawing (Figure 16), and could also be used to reconstruct objects from photographs [Sinha et al. 2008], even those with curved surfaces.

Our scaffold drawing techniques may be beneficial in less constrained tools [Bae et al. 2003], while their similarity constraints could also be integrated into our inference engine. A hybrid approach would allow the designer to draw freehand, but also construct accurate scaffolds when desirable. In light of perceptual drawing limitations [Schmidt et al. 2009], this is likely to be necessary. We have frequently encountered disbelief from those who try our software and find that their perspective intuition can be radically incorrect. This discovery is virtually always followed by a suggestion to provide our tool as a teaching aid for perspective drawing.

Since one of our goals was to explore the boundaries of a "pure-inference" drawing tool, we took a minimalist approach to our analytic drawing interface. Sketchup-style interactions for tasks like specifying angles and dimensions, drag-and-drop scaffold editing, and simple 3D tasks like extrusion, would improve design efficiency. Recent work in variational editing of curve networks [Gal et al. 2009] could be adapted to deform scaffolds while taking our inferred constraints into account. However, we do find it quite satisfying to simply draw, without the need for more traditional interactions. In that context, interesting future directions could include analytic-drawing-aware versions of 2D image-editing tools like cut-and-paste and the clone brush.

Two omissions from our tool are inference of surfaces and support for silhouette curves. These are related problems - given silhouettes it may be possible to infer surfaces. Once surfaces are available, we can explore analytic drawing on surfaces, which is in some cases the only way to accurately specify the intended 3D shape of a curve. Hand-painted rendering of 3D objects (Figure 1d) is also widely practiced in visual design [Robertson 2003], and it would be interesting to explore how this data could be utilized to more fully specify surface shapes.

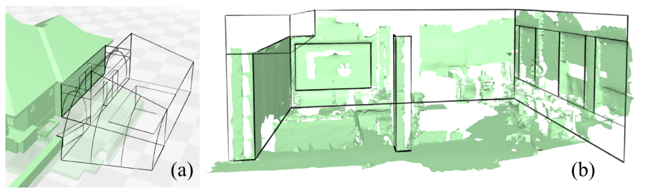

Figure 16: *Analytic drawing is widely used in architecture. In addition to novel buildings, we can quickly sketch renovations in the context of existing structures (b), or reconstruct the interior of a room from noisy and incomplete scan data (c).*

Acknowledgements

The authors are indebted to Seok-Hyung Bae, Ramtin Attar, Venk Prabhu, Igor Mordatch, Simon Breslav, Hyunyoung Song, Keenan Crane, Aaron Hertzmann, Chris Cheung, the other members of the DGP and Autodesk Research, and our reviewers, for their encouragement and constructive criticism. This work was funded in part by NSERC and MITACS.

References

AUTODESK INC., 2009. Autodesk AliasStudio. autodesk.com/aliasstudio.

BAE, S.-H., AND KIJIMA, R. 2003. Digital styling for designers: in prospective automotive design. In *Proc. Virtual Systems and Multimedia*.

BAE, S.-H., KIM, W.-S., AND KWON, E.-S. 2003. Digital styling for designers: Sketch emulation in computer environment. In *Proc. ICCSA*, 690–700.

BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In *Proc. UIST '08*, 151–160.

BIER, E. 1990. Snap-dragging in three dimensions. In *Proc. I3D '90*, 193–204.

BOURGUIGNON, D., CANI, M.-P., AND DRETTAKIS, G. 2001. Drawing for illustration and annotation in 3D. *Comp. Grap. Forum 20*, 3, 114–122.

BUXTON, B. 2007. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann.

CHEN, X., KANG, S. B., XU, Y.-Q., DORSEY, J., AND SHUM, H.-Y. 2008. Sketching reality: Realistic interpretation of architectural designs. *ACM Trans. Graph. 27*, 2, 1–15.

CHING, F. D. K. 1997. *Design Drawing*. Wiley.

COHEN, J., MARKOSIAN, L., ZELEZNIK, R., HUGHES, J., AND BARZEL, R. 1999. An interface for sketching 3D curves. In *Proc. I3D '99*, 17–21.

DAS, K., DIAZ-GUTIERREZ, P., AND GOPI, M. 2005. Sketching freeform surfaces using network of curves. In *Proc. SBIM '05*.

DO, E. Y. 2002. Drawing marks, acts, and reacts: Toward a computational sketching interface for architectural design. *Artif. Intell. Eng. Des. Anal. Manuf. 16*, 3, 149–171.

DORSEY, J., XU, S., SMEDRESMAN, G., RUSHMEIER, H., AND MCMILLAN, L. 2007. The Mental Canvas: A tool for conceptual architectural design and analysis. In *Proc. Pacific Graphics*.

EGGLI, L., HSU, C.-Y., BRUDERLIN, B., AND ELBER, G. 1997. Inferring 3D models from freehand sketches and constraints. *Computer-Aided Design 29*, 2, 101–112.

GAL, R., SORKINE, O., MITRA, N., AND COHEN-OR, D. 2009. iWIRES: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph 28*, 3, Article 33.

GLEICHER, M., AND WITKIN, A. 1994. Drawing with constraints. *Vis. Comput. 11*, 1, 39–51.

GOOGLE INC., 2009. SketchUp 7. http://sketchup.google.com.

GROSSMAN, T., BALAKRISHNAN, R., KURTENBACH, G., FITZMAURICE, G., KHAN, A., AND BUXTON, B. 2002. Creating principal 3D curves with digital tape drawing. In *Proc. CHI '02*, 121–128.

IGARASHI, T., AND HUGHES, J. 2001. A suggestive interface for 3D drawing. In *Proc. UIST '01*, 173–181.

IGARASHI, T., KAWACHIYA, S., TANAKA, H., AND MATSUOKA, S. 1998. Pegasus: a drawing system for rapid geometric design. In *Proc. CHI '98*, 24–25.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3D freeform design. In *Proc. SIGGRAPH '99*, 409–416.

KALLIO, K. 2005. 3D6B Editor: Projective 3D sketching with line-based rendering. In *Proc. SBIM '05*.

KARA, L. B., AND SHIMADA, K. 2007. Sketch-based 3D-shape creation for industrial styling design. *IEEE Comput. Graph. Appl. 27*, 1, 60–71.

KARPENKO, O., AND HUGHES, J. 2006. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Trans. Graph. 25*, 3, 589–598.

KARPENKO, O., HUGHES, J., AND RASKAR, R. 2004. Epipolar methods for multi-view sketching. In *Proc. SBIM '04*.

LEE, S., FENG, D., AND GOOCH, B. 2008. Automatic construction of 3D models from architectural line drawings. In *Proc. I3D '08*, 123–130.

LIPSON, H., AND SHPITALNI, M. 1996. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design 28*, 651–663.

MASRY, M., KANG, D., AND LIPSON, H. 2005. A freehand sketching interface for progressive construction of 3D objects. *Comp. & Graph. 29*, 563–575.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. FiberMesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph. 26*, 3, Article 41.

NICHOLLS, A., AND KENNEDY, J. 1995. Foreshortening in cube drawings by children and adults. *Perception 24*, 1443–1456.

PUGH, D. 1992. Designing Solid Objects Using Interactive Sketch Interpretation. In *Proc. I3D '92*, 117–126.

REITH, E., AND LIU, C. H. 1995. What hinders accurate depiction of projective shape? *Perception 24*, 995–1010.

ROBERTSON, S. 2003. *How to Draw Cars the Hot Wheels Way*. MotorBooks.

ROBERTSON, S., 2004. The techniques of scott robertson volume 1: Basic perspective form drawing. Gnomon Workshop. Instructional DVD.

SCHMIDT, R., ISENBERG, T., JEPP, P., SINGH, K., AND WYVILL, B. 2007. Sketching, scaffolding, and inking: a visual history for interactive 3D modeling. In *Proc. NPAR 07*, 23–32.

SCHMIDT, R., SINGH, K., AND BALAKRISHNAN, R. 2008. Sketching and composing widgets for 3d manipulation. *Comp. Graph. Forum 27*, 2, 301–310.

SCHMIDT, R., KHAN, A., KURTENBACH, G., AND SINGH, K. 2009. On expert performance in 3D curve-drawing tasks. In *Proc. SBIM '09*.

SINHA, S., STEEDLY, D., SZELISKI, R., AGRAWALA, M., AND POLLEFEYS, M. 2008. Interactive 3D architectural modeling from unordered photo collections. *ACM Trans. Graph. 27*, 5, Article 159.

SUTHERLAND, I. E. 1963. Sketchpad: A man-machine graphical communication system. In *Proc. Spring Joint Comput. Conf.*, 329–346.

TAYLOR, L., AND MITCHELL, P. 1997. Judgements of apparent shape contaminated by knowledge of reality: Viewing circles obliquely. *British J. Psych. 88*, 653–670.

TOLBA, O., DORSEY, J., AND MCMILLAN, L. 2001. A projective drawing system. In *Proc. I3D '01*, 25–34.

TSANG, S., BALAKRISHNAN, R., SINGH, K., AND RANJAN, A. 2004. A suggestive interface for image guided 3D sketching. In *Proc. CHI '04*, 591–598.

VARLEY, P., TAKAHASHI, Y., MITANI, J., AND SUZUKI, H. 2004. A two-stage approach for interpreting line drawings of curved objects. In *Proc. SBIM '04*, 117–126.

WANG, W., POTTMANN, H., AND LIU, Y. 2006. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Trans. Graph. 25*, 2, 214–238.

WELCH, W., AND WITKIN, A. 1994. Free-form shape design using triangulated surfaces. In *Proc. SIGGRAPH '94*, 247–256.

ZELEZNIK, R., HERNDON, K., AND HUGHES, J. 1996. SKETCH: an interface for sketching 3D scenes. In *Proc. SIGGRAPH '96*, 163–170.