# Model-Guided 3D Sketching

**Figure 1:** *Spring Training 2009, Peoria, AZ.*


## Abstract

Abstract.



**Keywords:** 3d sketching, conceptual design, interface, canvas planes

**Concepts:** •**Computing methodologies** → **Image manipulation;** *Computational photography;*


## 1 Introduction

Drawing a 2D sketch is often considered more intuitive and much faster than directly creating a 3D model. To facilitate 3D browsing of sketched objects, various 3D sketching systems [Dorsey et al. 2007; Bae et al. 2008; Zheng et al. 2016], which aim to lift 2D input sketches to 3D, have been proposed for early concept design. Most of the existing 3D sketching systems focus on 3D interpretation of 2D sketches alone, leading to a rather ill-posed problem. Additional assumptions or user inputs are thus often needed to resolve interpretation ambiguities.

We observe that designers often have to further develop ideas based on existing 3D models, which for example are intermediate models from an iterative design process. Designers may also start with an initial 3D model either retrieved from a shape repository (e.g., via a sketching interface [Eitz et al. 2012]) or acquired by 3D-scanning a physical object for renovation [Chen et al. 2015]. While the existing 3D sketching systems typically support the rendering of 3D sketches overlaid with a given 3D model, they either completely ignore or do not fully exploit the information of the 3D model for sketch interpretation. To the best of our knowledge, none of them is directly applicable to even simple examples shown in **Figure ??**.

In this work we will explore how a given 3D model can facilitate 3D interpretation of 2D sketches, and present a model-guided 3D sketching interface. We focus on sketching renovations of man-made objects or scenes, which often contain rich regular features, e.g., parallel lines, planar surfaces etc. A straightforward model-guided 3D sketching approach is to let users explicitly specify a planar surface of the model as a 3D canvas plane, on which 2D sketched strokes are projected to create 3D sketches. This approach often requires frequent changes of camera view due to the visibility problem involved for plane selection. Worse, a 3D model typically has a small number of planar surfaces, significantly limiting the design space.

To address these problems we propose to perform a co-analysis of multiple 2D strokes, which are expected to lie in a single 3D plane. With our incremental interface a user draws strokes *plane by plane*, but without explicit 3D plane specification. Once a group of 2D strokes constrained to a single plane is drawn, our inference engine automatically constructs a 3D plane by examining the attachment of those strokes to the 3D model and their correlation with the salient linear features of the model. With the constructed 3D plane, the 3D position of this group of 2D strokes can be easily determined. ***[Hongbo: TODO: additional assumption on the input strokes]***

**TODO 1** *suggestive interface; fix; rough strokes; for novice users; annotation, renovation; 3D models, scenes; It is easy and simple to use, so that it will be a suitable tool for novice user to express the idea of design.*

## 2 Related Work

We are interested in inferring the depth of 2D strokes, which are typically sketched via graphic tablets. So the existing 3D sketching systems relying on special 3D input hardware (e.g., [Sachs et al. 1991; Tsang et al. 2002]) is beyond the scope of our review here. 3D interpretation of 2D strokes is essentially an ill-posed problem and requires additional information to fix them in 3D. For example, a stroke might be required to drawn from multiple viewpoints [Karpenko et al. 2004; Bae et al. 2008; Rivers et al. 2010]. In a single view, a stroke together with its shadow on a horizontal plane [Cohen et al. 1999] or its symmetric counterpart [Bae et al. 2008] might be needed for 3D interpretation of this stroke. While these multi-curve approaches provide general ways to sketch 3D curves, they require accurate drawing, which is quite challenging (e.g., due to perceptual foreshortening biases).

A more common approach is to first specify a 3D sketch surface or canvas, on which 2D strokes can then be anchored one by one. Various types of 3D sketch surface (e.g., planes, extruded surfaces [Tsang et al. 2004; Bae et al. 2008], freeform surfaces [Kara and Shimada 2006; Nealen et al. 2007], inflation surfaces [Grimm and Joshi 2012]), have been explored for interactive 3D sketching. Among them 2D planes embedded in 3D (e.g., camera plane [Bourguignon et al. 2001], parallel planes [Dorsey et al. 2007], co-axial planes [Dorsey et al. 2007], orthographic planes [Grossman et al. 2002; Tsang et al. 2004; Bae et al. 2008; Zheng et al. 2016]) are

the most popular. Such planes are often pre-configured and need to be explicitly activated (only one each time) before their use. The planes might be translated and rotated in 3D for example using the traditional 3D object transformation tools. Our work follows this line of research. However, we aim to automatically construct a 3D sketch plane by analyzing a group of 2D strokes lying on it and examining their relations to a 3D guidance model. We believe that our technique can reduce the efforts for mode switching, plane selection and manipulation, thus providing a smoother sketching experience.

The main problem of using a specific type of sketch surface is that the space of possible 3D curves is limited to those lying on such sketch surfaces. To address this problem, Schmidt et al. [2009] proposed to first incrementally construct a linear 3D scaffold and then infer more freeform 3D curves from the scaffold. Our input 3D model for guidance is similar to their 3D scaffold in a sense that both of them provide geometric constraints (e.g., snapping [Bier 1990]) to filter the space of admissible 3D curves. Since the approach of Schmidt et al. incrementally fixes strokes in 3D *one by one*, without using or forming any sketch canvas, it does not support casual sketching of 3D curves (**Figure ??**), which cannot induce geometric constraints from the scaffold.

Face planarity is one of the fundamental geometric constraints used in automated interpretation of *a complete sketch* (e.g., [Lipson and Shpitalni 1996; Chen et al. 2008; Zou et al. 2015]). Many existing works aim for the reconstruction of 3D polyhedra (with planar faces). Recently, Xu et al. [2014] present a mathematical framework to infer piecewise-smooth curve networks from 2D drawings. Such curve networks can be finally surfaced to create 3D surface models [Pan et al. 2015]. Since 2D sketched strokes need to be ultimately mapped to 3D models to be reconstructed, these works all require relatively clean input drawings. Instead, our technique accepts more rough strokes as input and aim for rapid concept redesign of a given model.

Most of the above 3D sketching systems focus on the creation of 3D sketches from scratch, and only very few works explicitly discuss their use in the context of existing 3D models. Schmidt et al. [2009] render 3D sketches overlaid with an existing 3D model, but do not exploit any model information for sketch interpretation. Bourguignon et al. [Bourguignon et al. 2001] determine the depth of a canvas plane parallel to camera plane by examining the attachment of a stroke to a given 3D model. The *OverCoat* technique [Schmid et al. 2011] enables 3D painting by introducing isosurfaces of a proxy model as canvas. Similarly, *SecondSkin* require strokes sketched on and around 3D geometry to build new layered structures. In contrast, we aim to automatically derive canvas planes from a given model to anchor 2D strokes, which are not necessarily on, around or attached to the model.

3D sketching has also been studied in the context of single or multiple images. The existing approaches mainly use images as reference for 2D sketching, and employ the existing sketch interpretation techniques or their variations to fix 2D strokes in 3D (e.g., sketch planes adopted in [Tsang et al. 2004; Paczkowski et al. 2011], modified Lipson optimization in [Lau et al. 2010]). Very recently, Zheng et al. [2016] proposed to first derive a 3D cuboid from a reference image (to align with respective vanishing directions) and then sweep out candidate canvas planes from the three orthogonal faces of the cuboid. Similar to ours, their technique then formulates sketch interpretation as a selection problem from a set of context-inducted canvas planes. While their system is rather powerful, it might take several hours for first-time users to get familiar with their system. This is largely due to excessive user intervention needed for camera calibration, relation annotation, stroke grouping, etc. We aim for an easy-to-use model-guided 3D sketching system by heavily exploiting the stroke-model relations to minimize user intervention.

Due to the simplicity and intuitiveness, many sketch-based interfaces have been proposed for 3D modeling and editing (see an insightful survey in [Olsen et al. 2009]). A large category of existing techniques for sketch-based modeling from scratch are *reconstruction-based* (e.g., [Zeleznik et al. 1996; Igarashi et al. 1999; Karpenko and Hughes 2006; Nealen et al. 2007]). They require accurate drawing, since the input strokes are somehow mapped directly to the output model. Sketching has been demonstrated powerful for editing existing shapes (e.g., [Singh and Fiume 1998; Nealen et al. 2005; Olsen et al. 2005; Kara et al. 2006]). These approaches often use the existing models as sketch surfaces and aim for the generation of shape variations. Our goal is more similar to the retrieval-based approaches (e.g., [Shin and Igarashi 2007; Lee and Funkhouser 2008; Xu et al. 2013; Fan et al. 2013]), which intend to add new parts (models) into an existing model (scene) by retrieving the most similar parts (models) from a shape repository with respect to an input sketch as query. These approaches, however, are less interested in inferring 3D positions of the strokes, and after retrieval discard them, though sketches are often more expressive.

# 3 User Interface

Our prototype is composed of two windows: an input window for sketching and a preview window for displaying candidate 3D sketches, which are rendered in a specified view angle such that the user can easily perceive the results. Our user interface is then mainly devised into two interlaced stages: sketching and feedback. In the sketching stage, the user performs sketching on top of the screen. In a default mode, our system immediately returns 3D strokes for preview. Then, in the feedback stage, the user selectively confirms the current displayed results in 3D or override the results if she finds the default results not correct. To keep the sketching process smooth, we provide an intuitive feedback interface so as not to intervene the sketching process, which is necessary only when the current 3D content is falsely anchored. In the following, we describe the interface for sketching and feedback in detail.

## 3.1 Sketching

To start with our system, the user first loads a 3D model which will be sketched over. This model is displayed in the input window with orthogonal projection, thus the 3D parallel relation is retained in the screen space. The view angle can be manipulated in **the positioning mode** with rotation, translation, and zooming. When a desired view angle is obtained, the user can enter **the sketching mode** to draw sketch over the model. The drawing is performed using either mouse, digital pen, or a touch device, in a way similar to conventional drawing on paper. To convert the input strokes into 3D, we adopt an incremental strategy to reduce the potential ambiguities which arises with stroke number and let the user to resolve ambiguities only when necessary. In particular, during the drawing, each time the user draws a set of strokes, before the feedback stage, we assume all the set of strokes lie on some planar canvas. Then, in a key stage, our system finds the planar canvas automatically by inferring its position and orientation from the relations between the drawn strokes and the model. Each time when the user draws a new stroke, this planar canvas will be updated. When the user finishes a set of strokes, she provides her feedback using the feedback interface (detailed in the Section 3.2) to confirm the current set of 3D strokes and the drawn strokes will be converted into 3D by projecting them on the planar canvas. Then she can continue to draw another set of strokes or manipulate the view angle. The user can also enter **the erasing mode** to modify the previous drawn 3D

strokes (see in the accompanying video).

### 3.2 Feedback

When the user draws strokes in the input window, our system automatically finds a default planar canvas and convert the drawn strokes into 3D. However, the 3D information of the sketch can not be perceived in the input window without changing the view angle. Therefore, we display the resulting 3D sketches in the preview window in a different view angle. In addition, since the user's input may have ambiguities, the potential planar canvas may not be unique. In this case, our algorithm also finds other possible candidate planar canvases and display the candidate planar canvases with the associated 3D sketches to the user in the preview window. The user, can select one from the candidate sketches, if the default one is not desired. To select the desired 3D sketch and associated canvas, the user simply use the stylus or finger to stroke across the desired candidate. The user can also change the view angle to facilitate the selection process, if necessary. The selection mode and viewing mode can be switched by touching a floating button, which are displayed at the bottom of the preview window.

During the incremental sketching process, besides drawing strokes, an important task for the user to perform is to help confirm the current 3D contents and correct falsely estimated 3D strokes if exists. We find such interaction intuitive and typical as in traditional drawing system where a user often needs to confirm the current content using some emphatic means (e.g., by drawing heavy strokes over the contours of a character). To accomplish such task, a simple solution could be to include buttons on the task bar, such that the user could click these buttons to issue the desired commands. However, such interfaces often come with additional travel distance cost as the user needs to move the mouse or finger from one place to another, hesitating the drawing process. Instead, we design a user-friendly interface for this task. We adopt floating buttons to minimize the stylus or finger movement when the user wants to issue a command. During the drawing of a stroke, the buttons are invisible. When the stroke is finished, the floating buttons are shown next to the drawn strokes transparently. If the user continues to draw, the buttons fades away and will not introduce occlusion problem.

## 4 Methodology

Our algorithm is based on an important observation: when people sketch over an existing model to depict a new content, some of the user strokes will typically behave as the structural guidelines comply with the structure of the underlying model and usually share relations with the model. Therefore we can use the structural relations between the strokes and the model to estimate the 3D information of the drawn 2D strokes.

### 4.1 Relations between Shape and Strokes

Before detecting the relations between the strokes and shape, we first extract straight segment from the drawn 2D strokes, and linear features from the shape. A straight segment is a consecutive part of stroke which is nearly straight. We use a simple growing algorithm to find the straight segments. Given a new stroke, we first uniformly resample it. The distance between sample points is **10px** in our prototype. Then from the beginning of this resampled stroke, we add the points to current straight segment one by one, if the new added points will not introduce a sharp turning. Otherwise, current straight segment is finished and a new straight segment will be constructed. Then we filter out the short ones (less than **50px**).

The linear features of the shape include sharp edge and main face normal. The sharp edges are extracted using a similar method in **[I-wire]**. After extracting the sharp edges, we detect the straight ones similar to detecting the straight segment. The main face normal is obtained by first clustering the faces of the model with similar normal. The clustered faces forms several shape plane canvases. The normals of these canvases are considered as the linear feature of the shapes.

After obtaining the straight segment of the stroke and linear feature of the shape, we detect the following relations: **Colinearity.** This relation is detected between a straight segment and a straight sharp edge. The normals of shape plane canvases will not be used for detection since they miss position information. **Parallel.** This relation is detected between a straight segment and a shape linear feature, including both straight sharp edge and normal of shape plane canvas. These two relations are both detected in the screen space.

Another important relation between the stroke and shape is **Attachment**. If a endpoint of the stroke is over or close to the shape, we consider that this endpoint attaches to the shape. To make the attachment more content-aware, the detection is performed in three levels. First, the attachment is detected between the endpoint and the corner of the shape. If no attachment relation is found, the detection continues between the endpoint and the sharp edges. If no attachment relation is found in this level, we then detect the attachment between the endpoint and the shape's face.

### 4.2 Canvas Estimation

The canvas is obtained from the relations between shape and strokes. Due to the ambiguity of the input, some of the detected relations are actually undesired. Therefore it is not wise to find a canvas which tries to fulfill all the detected relations. One observation is that, although the user's input contains ambiguity, most of the relations should be able to comply with each other, if the correct canvas is found. Motivated by this observation, we propose a two step approach to estimate the canvas.

First, we construct candidate canvases. Candidate canvases are all possible canvases which could be obtained by the combination of the relations. We found that the following combinations are able to yield valid canvases.

**Two colinearity.** If the corresponding two straight sharp edges could spans a plane, this combination can yield a valid canvas.

**One colinearity + one parallel.** If the corresponding two straight sharp edges are not parallel, this combination can yield a valid canvas.

**One colinearity + one attachment.** If the attached 3D point is not colinear with the corresponding straight sharp edge, this combination can yield a valid canvas.

**One parallel + two attachments.** if the two attached 3D points are not at the same position, and direction line between them is not parallel with the corresponding shape linear feature, this combination can yield a valid canvas.

**Two parallel + one attachment.** If the corresponding two shape linear features are not parallel, this combination can yield a valid canvas.

**Three attachments.** If the three attached 3D points are not colinear, this combination can yield a valid canvas.

After obtaining the candidate canvas, we rank them according to how well these canvases preserve the detected relations. Given a canvas, we can convert the 2D strokes into 3D by a simple projection. After this conversion, the straight segments are all in 3D

forms. We then check whether these straight segments share relation with the shape linear features. In addition, the attachment relation can also be validated. For each canvas and associated 3D sketch, we obtain the following scores to measure the degree of relation preservation.

$$
\begin{aligned}
E_c &= \sum_{s_i \in \mathcal{S}_c} L(s_i) \\
E_p &= \sum_{s_i \in \mathcal{S}_p} L(s_i) \\
E_a &= \sum_{a_i \in \mathcal{A}_a} W(a_i)
\end{aligned}
\tag{1}
$$

where $E_c$, $E_p$, and $E_a$ are relation preservation score of colinearity, parallel, and attachment respectively. $s_i$ is a straight segment. $\mathcal{S}_c$ and $\mathcal{S}_p$ are the sets containing the straight segments which share colinearity/parallel relation with a shape linear feature. $L(\cdot)$ means the length of a straight segment. $a_i$ is an attachment point. $\mathcal{A}_a$ is the set containing the attachments which are preserved after conversion to 3D. $W(\cdot)$ is the weight of the attachment, whose value is determined by the type of attachment: **To be determined.**

After getting these scores for each canvas and associated 3D sketch, we use the following strategy to rank them. First, we rank the candidates according to their $E_p$, and keep only the first **ratio1** candidates. **To be continued.**

### 4.3 Adding Details on Existing Canvas

The surface in each step the user draws strokes on is extracted from the model, based on shape understanding. We extract useful information including sharp edges, dominant faces, dominant normals, and skeletons, to determine the surface. Sharp edge is defined as the salient linear feature of the model. Dominant face is defined as the salient 2D feature. Dominant normal is the normal of dominant face, which can be considered as another type of linear feature. Skeleton of the shape is useful when the shape has a skinny structure.

The matching between the extracted feature from the model and the input strokes determines how to interpret the 2D sketch in 3D. The matching is performed in 2D, i.e., we project the extracted 3D feature to screen plane, and get the matching between the projected feature and the drawn strokes. This is based on the following observation: when user draws 2D stroke, although she perceives the model and the drawn strokes in 3D, she use the their 2D information as reference. Therefore the matching in screen space is more reliable. We matching the drawn strokes with the extracted features, including sharp edges, dominant faces, dominant normals, and skeletons, to determine the surface on which the user draws.

Projecting the drawn strokes to the determined surface will generate a group of 3D strokes. However, since the 2D strokes drawn by the user are rough, the generated 3D strokes may deviate from the desired position. Therefore we need to postprocess the 3D strokes to obtain a more satisfactory 3D sketch. This beautification process is based on the following criteria. First, the 2D version of the beautified 3D strokes should similar to the original 2D strokes. This criterion ensures that the user's input is respected. Second, the beautified 3D strokes should match the features of the models better. This criterion ensures that the results follows the expectation of the user. This beautification process can be realized by an optimization approach.

## References


BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. 2008. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM, 151–160.

BIER, E. A. 1990. Snap-dragging in three dimensions. In *ACM SIGGRAPH Computer Graphics*, vol. 24, ACM, 193–204.

BOURGUIGNON, D., CANI, M.-P., AND DRETTAKIS, G. 2001. Drawing for illustration and annotation in 3d. In *Computer Graphics Forum*, vol. 20, Wiley Online Library, 114–123.

CHEN, X., KANG, S., XU, Y., DORSEY, J., AND SHUM, H. 2008. Sketching reality: Realistic interpretation of architectural designs. *ACM Transactions on Graphics (TOG) 27*, 2, 11.

CHEN, X. A., COROS, S., MANKOFF, J., AND HUDSON, S. E. 2015. Encore: 3d printed augmentation of everyday objects with printed-over, affixed and interlocked attachments. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM, New York, NY, USA, UIST '15, 73–82.

COHEN, J. M., MARKOSIAN, L., ZELEZNIK, R. C., HUGHES, J. F., AND BARZEL, R. 1999. An interface for sketching 3d curves. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM, 17–21.

DORSEY, J., XU, S., SMEDRESMAN, G., RUSHMEIER, H., AND MCMILLAN, L. 2007. The mental canvas: A tool for conceptual architectural design and analysis. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, IEEE, 201–210.

EITZ, M., RICHTER, R., BOUBEKEUR, T., HILDEBRAND, K., AND ALEXA, M. 2012. Sketch-based shape retrieval. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4, 31:1–31:10.

FAN, L., WANG, R., XU, L., DENG, J., AND LIU, L. 2013. Modeling by drawing with shadow guidance. In *Computer Graphics Forum*, vol. 32, Wiley Online Library, 157–166.

GRIMM, C., AND JOSHI, P. 2012. Just drawit: a 3d sketching system. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SBIM '12, 121–130.

GROSSMAN, T., BALAKRISHNAN, R., KURTENBACH, G., FITZMAURICE, G., KHAN, A., AND BUXTON, B. 2002. Creating principal 3d curves with digital tape drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 121–128.

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

KARA, L. B., AND SHIMADA, K. 2006. Construction and modification of 3d geometry using a sketch-based interface. In *Eurographics workshop on sketch-based interfaces and modeling*, 59–66.

KARA, L. B., D'ERAMO, C. M., AND SHIMADA, K. 2006. Pen-based styling design of 3d geometry using concept sketches and template models. In *Proceedings of the 2006 ACM symposium on Solid and physical modeling*, ACM, 149–160.

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

KARPENKO, O., HUGHES, J. F., AND RASKAR, R. 2004. Epipolar methods for multi-view sketching. In *Proceedings of the First Eurographics conference on Sketch-Based Interfaces and Modeling*, Citeseer, 167–173.

LAU, M., SAUL, G., MITANI, J., AND IGARASHI, T. 2010. Modeling-in-context: User design of complementary objects with a single photo. In *SBIM 2010*.

LEE, J., AND FUNKHOUSER, T. 2008. Sketch-based search and composition of 3D models. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

LIPSON, H., AND SHPITALNI, M. 1996. Optimization-based reconstruction of a 3d object from a single freehand line drawing. *Computer-Aided Design 28*, 8, 651–663.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3, 1142–1147.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: designing freeform surfaces with 3d curves. In *ACM Transactions on Graphics (TOG)*, vol. 26, ACM, 41.

OLSEN, L., SAMAVATI, F. F., SOUSA, M. C., AND JORGE, J. A. 2005. Sketch-based mesh augmentation. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*.

OLSEN, L., SAMAVATI, F. F., SOUSA, M. C., AND JORGE, J. A. 2009. Sketch-based modeling: A survey. *Computers & Graphics 33*, 1, 85–103.

PACZKOWSKI, P., KIM, M. H., MORVAN, Y., DORSEY, J., RUSHMEIER, H., AND O'SULLIVAN, C. 2011. Insitu: Sketching architectural designs in context. *ACM Trans. Graph. 30*, 6 (Dec.), 182:1–182:10.

PAN, H., LIU, Y., SHEFFER, A., VINING, N., LI, C.-J., AND WANG, W. 2015. Flow aligned surfacing of curve networks. *ACM Trans. Graph. 34*, 4 (July), 127:1–127:10.

RIVERS, A., DURAND, F., AND IGARASHI, T. 2010. 3d modeling with silhouettes. *ACM Trans. Graph. 29*, 4, 1–8.

SACHS, E., ROBERTS, A., AND STOOPS, D. 1991. 3-draw: A tool for designing 3d shapes. *IEEE Computer Graphics and Applications 11*, 6, 18–26.

SCHMID, J., SENN, M. S., GROSS, M., AND SUMNER, R. W. 2011. Overcoat: an implicit canvas for 3d painting. *ACM Trans. Graph. 30* (August), 28:1–28:10.

SCHMIDT, R., KHAN, A., SINGH, K., AND KURTENBACH, G. 2009. Analytic drawing of 3d scaffolds. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 149.

SHIN, H., AND IGARASHI, T. 2007. Magic canvas: interactive design of a 3-d scene prototype from freehand sketches. In *Proceedings of Graphics Interface 2007*, ACM, 63–70.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 405–414.

TSANG, M., FITZMAURICE, G. W., KURTENBACH, G., KHAN, A., AND BUXTON, B. 2002. Boom chameleon: simultaneous capture of 3d viewpoint, voice and gesture annotations on a spatially-aware display. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, ACM, 111–120.

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

XU, K., CHEN, K., FU, H., SUN, W.-L., AND HU, S.-M. 2013. Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models. *ACM Transactions on Graphics 32*, 4, Article No. 123.

XU, B., CHANG, W., SHEFFER, A., BOUSSEAU, A., MCCRAE, J., AND SINGH, K. 2014. True2form: 3d curve networks from 2d sketches via selective regularization. *ACM Transactions on Graphics 33*, 4.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. SKETCH: An interface for sketching 3D scenes. In *Proceedings of ACM SIGGRAPH*, 163–170.

ZHENG, Y., LIU, H., DORSEY, J., AND MITRA, N. J. 2016. Snmartcanvas: Context-inferred interpretation of sketches for preparatory design studies. *Computer Graphics Forum (Proceedings of Eurographics 2016)*.

ZOU, C., CHEN, S., FU, H., AND LIU, J. 2015. Progressive 3d reconstruction of planar-faced manifold objects with drf-based line drawing decomposition. *IEEE Transactions on Visualization and Computer Graphics 21*, 2, 252–263.