This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2860016, IEEE Transactions on Visualization and Computer Graphics

Model-Guided 3D Sketching

Pengfei Xu Hongbo Fu Youyi Zheng Karan Singh Hui Huang Chiew-Lan Tai

Abstract—We present a novel 3D model-guided interface for in-situ sketching on 3D planes. Our work is motivated by evolutionary design, where existing 3D objects form the basis for conceptual re-design or further design exploration. We contribute a novel workflow that exploits the geometry of an underlying 3D model to infer 3D planes on which 2D strokes drawn that are on and around the 3D model should be meaningfully projected. This provides users with the nearly modeless fluidity of a sketching interface, and is particularly useful for 3D sketching over planes that are not easily accessible or do not preexist. We also provide an additional set of tools, including sketching with explicit plane selection and model-aware canvas manipulation. Our system is evaluated with a user study, showing that our technique is easy to learn and effective for rapid sketching of product design variations around existing 3D models.

Index Terms—3D sketching, conceptual design, interface, canvas

1 Introduction

Sketching is a quintessential mode of visual communication that designers use extensively throughout product design life-cycles, particularly in early stages of conceptual design. While 2D drawing of 3D curves from one or more view-points is a mature traditional art-form, the computational problem of lifting these sketched 2D strokes into 3D curves continues to be a grand challenge due to the infinitum of 3D curves that could project to any given 2D stroke. The large body of existing sketch-based modeling systems reduces this search space to a small set of plausible 3D curves by an explicit drawing workflow [1], [2], [3], [4], establishing drawing systems, scaffolds and geometric priors [5], [6], [7], enforcing inter-stroke regularities [8], or narrowing the sketching domain [9]. Prior art has largely used a 3D model, if at all, strictly as a visual reference [4], a projection surface [10] or as an inner layer for shell-like modeling [9]. We observe that man-made objects have a plurality of geometric features that can provide a strong suggestive prior to automatically infer 3D sketch planes onto which sketched 2D strokes can be projected. It is also shown in the recent literature that, for early concept design, curves sketched in planes are not overly restrictive, but rather desirable for man-made objects [1], [6].

We thus develop a novel model-guided 3D sketching system (Figure 1). The initial 3D model for a design iteration can come from scanning an existing 3D object [11], or from template examples retrieved from a shape repository (e.g., via a sketching interface [12]). As described in an exploratory study [4], explicit 3D plane selection and control disrupts the sketching process and often requires frequent changes of viewpoint, since good views from which to manipulate the plane and good views from which to sketch on the plane, tend to be near orthogonal to each other. Our contribution is a novel sketching workflow: users can draw strokes *without explicit 3D plane specification*. To achieve this we first process the 3D model to determine salient geometric features such as planar regions and straight lines, which provide strong cues for candidate 3D sketch planes. Plane prediction is further made tractable by the observation that most man-made conceptual design objects have a significantly reduced design space of potential planar surfaces for sketching. We then perform a co-analysis of multiple 2D strokes, which we expect to lie on a single 3D plane. Once a group of 2D strokes constrained on a single plane is drawn, our system automatically infers a 3D plane through selecting subsets of these strokes as construction lines and examining their correlations with the salient geometric features of the 3D model. The 2D strokes are then projected onto the inferred 3D plane.

Our novel sketching workflow provides users with the nearly modeless fluidity of a sketching interface. It enables 3D sketching over planes that are not directly selectable from a reference 3D model or do not preexist (e.g., for designing shelf dividers in Figure 1(a)). Since our technique employs the geometric constraints from a 3D model, it is able to handle imprecisely drawn strokes reasonably well. We also provide the traditional workflow of sketching with explicit selection of 3D planes when they are easily accessible, and 3D plane manipulation tools. By using these two sketching workflows and creating 3D sketches plane by plane, users are able to produce 3D concept (re-)design around 3D models with ease. Our tool has been tested by artists drawing on various man-made models (see Figures 1 and 14). Our user study shows that even novice users are able to quickly learn how to use our system, reproduce target 3D sketches, and create interesting new 3D sketches.

- *P. Xu is with the College of Computer Science & Software Engineering, Shenzhen University. E-mail: xupengfei.cg@gmail.com*
- *H. Fu is with the School of Creative Media, the City University of Hong Kong. E-mail: hongbofu@cityu.edu.hk*
- *Y. Zheng is with the College of Computer Science, Zhejiang University. E-mail: zyy@cad.zju.edu.cn*
- *K. Singh is with the Department of Computer Science, the University of Toronto. E-mail: karan@dgp.toronto.edu*
- *H. Huang is with the College of Computer Science & Software Engineering, Shenzhen University. E-mail: hhzhiyan@gmail.com*
- *C. Tai is with the Department of Computer Science & Engineering, the Hong Kong University of Science & Technology. E-mail: taicl@cs.ust.hk*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2860016, IEEE Transactions on Visualization and Computer Graphics

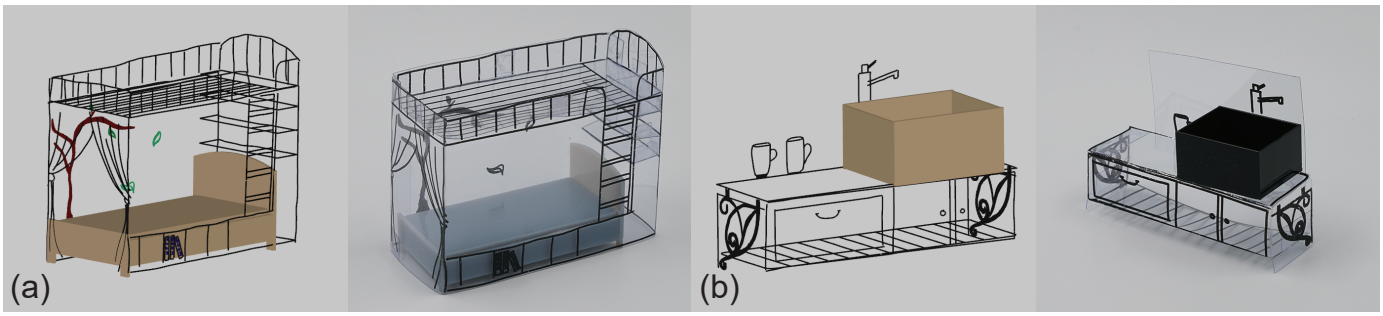


Fig. 1. Our model-guided 3D sketching system allows users to create interesting sketch-based redesign of existing 3D models with ease. The right in each example is the physical realization of 3D sketches around 3D-printed models.

2 Related Work

3D sketching via 2D input devices has been widely studied. This problem is essentially ill-posed and requires additional information to fix input 2D strokes in 3D. A 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 [2], [13], freeform surfaces [14], [15], [16], inflation surfaces [17]) have been explored for interactive 3D sketching. Among them 2D planes embedded in 3D (e.g., camera plane [4], [16], [18], parallel planes [1], co-axial planes [1], orthographic planes [2], [6], [13], [19]) are the most popular. However, such systems like *Mental Canvas* [1] often rely on pre-configured planes, and require explicit mode switching, frequent plane selection and manipulation, thus disrupting the sketching process [4]. We show that explicit plane selection is unnecessary, providing a smoother experience when the desired 3D planes can be derived from a given 3D model. Our work is complementary to these existing works, just like the two complementary sketching workflows in our system. The sketching workflow with explicit plane selection can be enhanced by integrating existing tools like tick-based plane selection from *EverybodyLovesSketch* [3].

Most of the existing 3D sketching systems focus on the creation of 3D sketches from scratch; only very few works explicitly discuss their use in the context of existing 3D models. *SketchingWithHands* [7] focuses on 3D sketching around a 3D virtual hand model to facilitate designing handheld products. Bourguignon et al. [18] determine the depth of a canvas plane parallel to the camera plane by examining the attachment of a stroke to a general 3D model. The *OverCoat* technique [20] enables 3D painting by introducing isosurfaces of a proxy model as canvas. *SecondSkin* [9] uses a 3D model as an inner layer for shell-like modeling. In contrast, our technique explores a different design space and does not require the resulting curves to be in close proximity to the surface of existing models (Figure 14). Schmidt et al. [5] proposed to first incrementally construct a linear 3D scaffold and then infer 3D curves with a higher degree of freedom from the scaffold. While they did not attempt to exploit the information of a 3D model for sketch interpretation, it is possible to extend their system for model-driven sketching by extracting and using sharp edges of the model as existing scaffold curves. However, it is unclear how to incorporate on-canvas sketching (e.g., for hatching effects in Figure 5) or attachment of 3D sketches to existing model faces (similar to the example in Figure 6(b)) other than model edges. In addition, their system requires every curve to be accurate and cannot handle rough sketches. Recently, Krs. et al. [21] introduce a sketching interface for producing 3D curves that wrap around given 3D object(s). However, the design space of their 3D curves are largely constrained to the offset surfaces of the existing geometry. The recent work of [22] allows the design of detailed 3D objects in situ by combining 2D and 3D sketching, which however requires modern AR hardware.

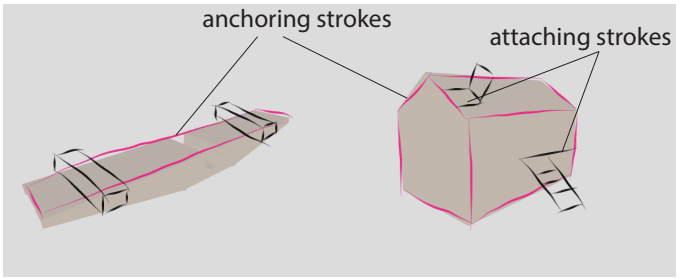


Fig. 2. *SmartCanvas* in action when using the rendered models as input reference images. Our system needs much less user intervention (without drawing the strokes in pink) to achieve the same results.

3D sketching has also been studied in the context of single or multiple images. Existing approaches mainly use images as references for 2D sketching, and employ existing sketch interpretation techniques or their variations to fix 2D strokes in 3D (e.g., sketch planes adopted in [13], [23], [24], modified Lipson optimization in [25]). Very recently, Zheng et al. [6] proposed *SmartCanvas*, which takes an image as reference and formulates the sketch interpretation as a selection problem from a set of context-induced canvas planes. Since their incremental sketching process requires strokes to be attached to previously drawn strokes, their system often leverages additional strokes (pink ones in Figure 2, which are not needed in our system) to anchor or host a newly sketched stroke. While their system is rather powerful, it requires excessive user intervention for camera calibration, relation annotation (e.g., to explicitly annotate the inclining face of the roof in Figure 2 (right)), stroke grouping, etc.

Face planarity is one of the fundamental geometric constraints used in automated interpretation of *a complete sketch* (e.g., [26], [27], [28], [29]). Many existing works aim at reconstruction of 3D polyhedra (with planar faces). Recently, Xu et al. [8] present a mathematical framework to infer

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2860016, IEEE Transactions on Visualization and Computer Graphics

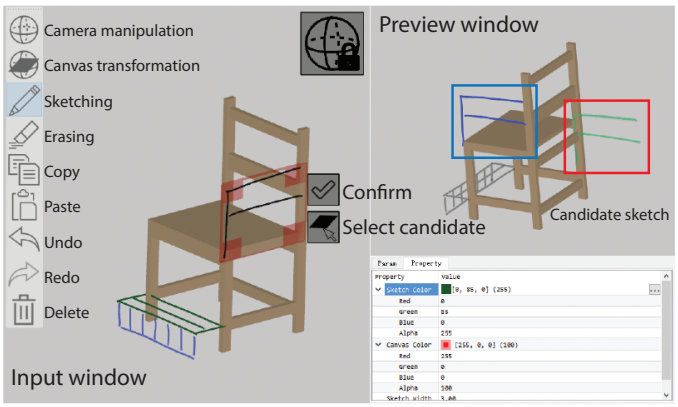


Fig. 3. Main interface of our system.

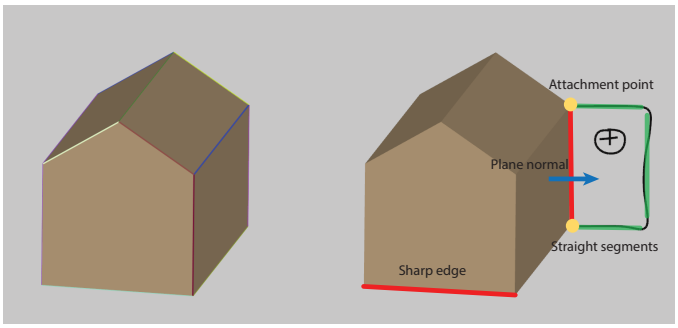


Fig. 4. Left: the detected linear features. Right: User-drawn strokes are broken into straight segments (in green). They are parallel with the detected linear features (in red) or the normals (in blue) of dominant planes. These features are transparent to the user.

piecewise-smooth curve networks from 2D drawings. Such curve networks can be finally surfaced to create 3D surface models [30]. Since 2D sketched strokes need to be ultimately mapped to 3D models to be reconstructed, these works all require relatively clean input drawings. In contrast, our technique accepts more casually sketched strokes as input and aims for rapid concept redesign of a given model.

Due to the simplicity and intuitiveness, many sketch-based interfaces have been proposed for 3D modeling and editing (see an insightful survey in [31]). A large category of existing techniques for sketch-based modeling from scratch are *reconstruction-based* (e.g., [15], [32], [33], [34], [35]). They require accurate drawing, since the input strokes are somehow mapped directly to the output model. Recently Li et al. [36] present an in-context sketch-based modeling tool for quickly modeling swept surfaces on top of an RGBD image, which provides structural information to infer the 3D position of a pair of user strokes (for defining a swept surface). Sketching has also been demonstrated powerful for editing existing shapes (e.g., [37], [38], [39], [40], [41]). These approaches often use the existing models as sketch surfaces and aim at generating shape variations. Our goal is in a sense more similar to the retrieval-based approaches (e.g., [42], [43], [44], [45]), 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. Although the sketches of these retrieval approaches are often more expressive, inferring 3D positions of the strokes is not their objective.

3 User Interface

Our interactive system allows a user to create 3D sketches on and around an existing model. We adopt planar canvases to anchor 2D input strokes in 3D. To reduce potential ambiguities, we use an incremental strategy and require the user to input strokes *plane by plane*. Our system supports the following two sketching workflows: 1) first 3D plane selection (if such a plane exists and is easily accessible) and then further in-plane sketching (Figure 6(a)); 2) first sketching (without plane selection) and then 3D plane confirmation, followed by an optional step of in-plane sketching (see Figure 5 and the accompanying video). Our discussion below mainly focuses on the latter scenario, where our main contribution lies.

3.1 Basic Operations

As shown in Figure 3, our main user interface is composed of two windows: an input window for sketching and a preview window. The user starts with loading a 3D model. We focus on man-made models, which typically have a rich set of regular structures. The model is displayed in the input window with orthographic projection so that 3D parallelism is retained in the screen space. The user may rotate the camera view of the input window via an Arcball-like interface [46]. To facilitate sketching at different scales, we also provide the pan and zoom tools in the input window. A set of auxiliary features such as copy-and-paste, erasing, undo/redo, deletion are also included.

3.2 Sketching without Explicit Plane Selection

Sketching without explicit plane selection allows the user to easily create 3D sketches even when the underlying 3D planes cannot be easily derived from the 3D model without additional operations for camera manipulation or plane manipulation (Figure 5(a)).

In the sketching mode, if the user starts to draw without explicitly selecting any plane, the system automatically determines that the user wants a plane (we call it a *planar canvas* hereafter) to be inferred from the drawn strokes, which later needs to be confirmed by the user. Each time the user draws a set of strokes that will lie on some planar canvas. Our system makes use of a subset of strokes as image-space *construction lines* to construct a 3D planar canvas. Specifically, any stroke is a potential construction line if it roughly shares a collinearity or parallelism relation with a linear feature (Figure 4) of the 3D model, or is attached to the 3D model in the image space. Figure 7 shows various combinations of model-stroke relations which can lead to the construction of a 3D plane. The user may also rely on the confirmed 3D sketches to construct a subsequent planar canvas, similar to using the 3D model as guidance (Figure 6(e)) and the idea of an evolving scaffold [5].

Note that the user neither needs to draw the strokes with a required manner (Figure 6(c)), nor labels which input strokes should be construction lines, since the system will automatically identify them by checking their geometric relations with the model features. This allows a smoother sketching process, especially when such construction lines are intended to be part of the final sketches. When the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2860016, IEEE Transactions on Visualization and Computer Graphics

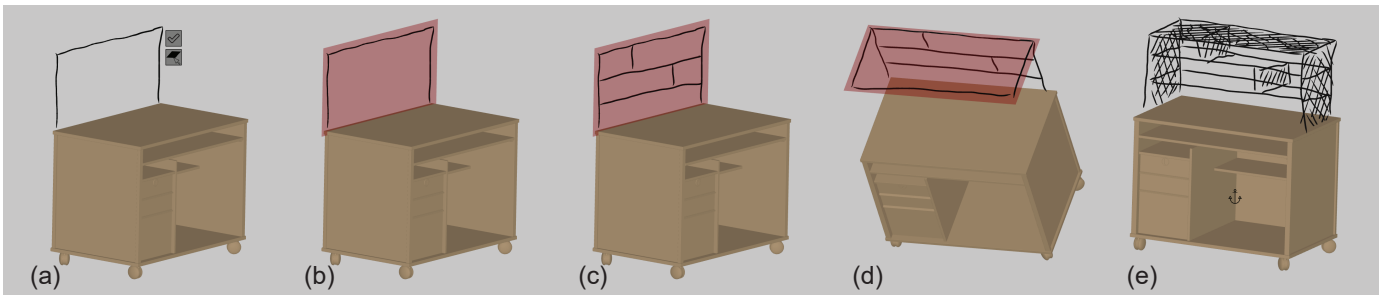


Fig. 5. Inference-enabled sketching workflow: first sketching (a) and then 3D plane confirmation (b), followed by in-plane sketching (c). These steps are repeated to add sketches plane by plane (d), leading to the final result (e).

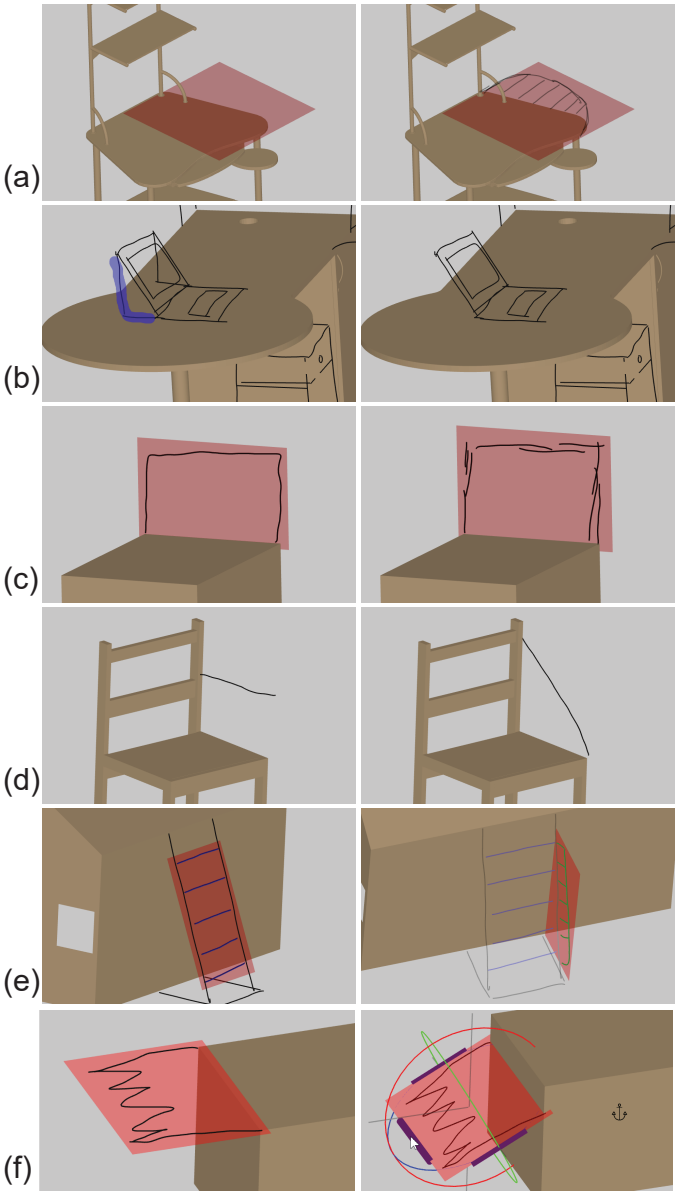


Fig. 6. (a) A sketching workflow with explicit plane selection. (b) Erasing. (c) Supporting various drawing habits (left: single stroke; right: over-sketched multiple strokes). (d) Line-by-line sketching. (e) Sketching over previously confirmed sketches. (f) Model-aware canvas transformation.

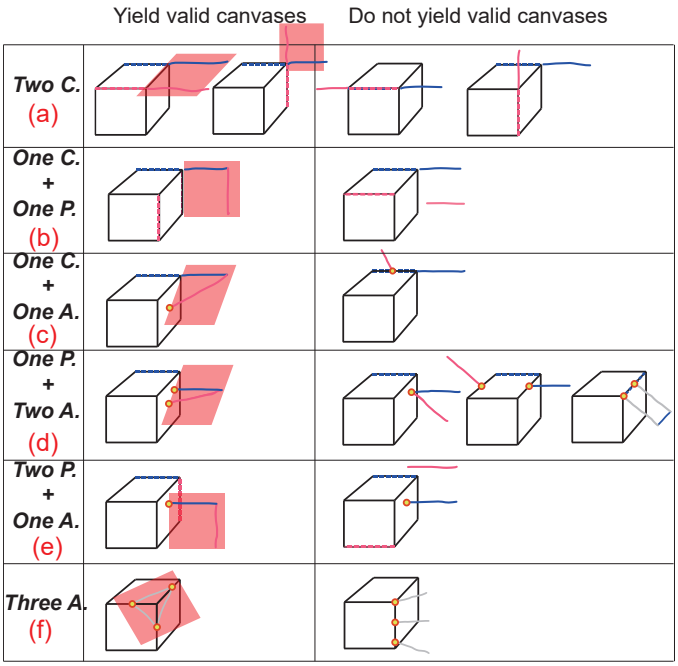


Fig. 7. Illustration of possible combinations of model-stroke relations yielding or not yielding valid canvases. Recall that collinearity involves a corresponding sharp model edge while parallelism involves a corresponding linear model feature (sharp edge or plane normal). Lines of the same colors exhibit a collinearity or parallel relation. For simplicity, only parallelism with sharp model edges is illustrated here; parallelism with plane normals is similar.

construction lines are input solely for the construction of a 3D planar canvas and are not intended to appear in the final sketches, the user can easily erase them using our erasing tool (Figure 6(b)). While it is not required, the user is recommended to input the construction lines prior to other lines so that a desired planar canvas is inferred soonest possible for confirmation. This effectively helps reduce the degree of ambiguities which generally increases with the number of strokes.

In addition, when the user draws one straight stroke, our system also tries to infer its 3D position (Figure 6(d)). Since a 3D line cannot determine a 3D plane, no planar canvas is associated with the inferred 3D line. If the user continues to draw new strokes, all the strokes are used for canvas inference as described before.

3.3 Feedback

When the user draws strokes in the input window, the system automatically finds a default planar canvas and lifts the

drawn strokes into 3D. We display the resulting 3D sketches of the current set of strokes from a different view angle in the preview window (see Figure 3). To better examine the current interpretation results, the user may interactively change the view angle in the preview window using the same Arcball interface (see the accompanying video).

In addition, due to the inherent ambiguities of user's strokes, the desired planar canvas may not be inferred correctly occasionally. Thus, our algorithm proposes a set of candidate planar canvases and displays the corresponding sketch interpretation results in the preview window (Figure 3). This interface is inspired by previous suggestive interfaces (e.g., [6], [13], [47]). Strokes are color-coded to show different interpretations. Since the current set of strokes might have been built upon previous sets of planar sketches, instead of displaying candidate sketches only, for completeness we display all the sketches in the preview window and render the previous sketches semi-transparently in gray to reduce visual clutter (Figure 3).

3.4 Plane Confirmation

If the user is satisfied with the default 3D interpretation (shown in blue in the preview window) and the corresponding 3D planar canvas, s/he may confirm it by simply clicking one of the four corners of the default 3D plane (Figure 3). The system will then display the confirmed 3D plane in a semi-transparent color in the input window. The user may continue adding new strokes onto this planar canvas, which will not be inferred again. It is suggested that the user makes plane confirmation as early as possible, since more strokes might lead to more interpretation candidates.

When a desired interpretation result is a candidate appearing in the preview window but is not the default one, the user may either continue drawing more strokes to refine the canvas inference or enter a *candidate selection* mode by clicking a floating button (Figure 3). In the candidate selection mode, by default, the view angle in the input window is set to be the same as that in the preview window, and all the candidate sketches are displayed in the input window. The user may select a candidate plane by single-clicking on the corresponding candidate sketch. Without mode switching, the camera view can be changed freely to facilitate easy selection. After selecting the desired planar canvas, the user may confirm by clicking the tick-symbol button. This ends the candidate selection mode and switches back to the sketching mode, with the last sketching view restored and the confirmed plane displayed. The user may continue sketching on the confirmed canvas.

The user can reuse or transform the existing canvases (Figure 6(f)). Since the final 3D sketches are created with respect to the 3D reference model, we design model-aware canvas transformation tools. First, the translation axes are always defined as the principal axes of the 3D model (by Principal Component Analysis) and are not influenced by the rotation tools. Second, snapping with respect to the 3D model is enabled during canvas translation or rotation. By using two different widgets, the user may rotate a canvas with respect to the canvas center or one of its bounding box edges. The latter is convenient for achieving hinge-like configurations between the 3D canvas and model (Figure 6(f)).

4 Model-Guided Planar Canvas Inference

The main challenge of our system is to infer a 3D planar canvas from a set of 2D strokes lying on it, with the help of a 3D guidance model being sketched over. Our solution is based on a key observation: when people sketch over an existing model to depict a new content, some of the input strokes will typically act as the structural guidelines complying with the structure of the underlying model and they usually share certain geometric relations with the model. We refer to such strokes as potential *construction lines*, and exploit the structural relations between the construction lines and the model for 3D planar canvas inference.

4.1 Model-Stroke Coupling

In order to bridge between the 3D model and the 2D drawn strokes, we first exploit some dominant features of the model, which will later be coupled with strokes according to structural relations to assist in their 3D positioning. Literature has shown that our human visual system is extremely sensitive to sharp features and corners [48]. We thus extract these discriminative features from the model. Since we focus on the inference of 3D planes from the model, we are particularly interested in the basic 3D elements that can contribute to the construction of a 3D plane, which could also be associated with user drawn strokes. To this end, we extract the following two types of shape features: *sharp edges* and *dominant plane normals*. The two basic features naturally enable us to exploit the linearity association between the model and the user strokes, thus facilitating plane inference. We treat these two types of features as basic blocks of constructing elements for 3D plane estimation. We term them as *linear features*.

The sharp model edges are extracted using a similar method as in [49]: we first extract the closed-region shape feature curves and break them into straight line segments as sharp edges via polygonal approximation of curves [50]. The *dominant normals* are the normals of dominant planes, which are obtained by clustering the face normals of the model with Mean-Shift [51] (using the bandwidth of 0.3, which corresponds to the angle of $\frac{\pi}{10}$) to detect coplanar faces. Too short model edges (shorter than 50 pixels in our implementation) and too small dominant planes (smaller than 2,500 square pixels) are filtered out, since they do not clearly indicate any structural information. Note that these thresholds are not affected by the camera settings, since they are used in the screen space. Their values depend on the specification of a monitor. In our experiments, we adopted a 13.3-inch Wacom Cintiq Companion 2 tablet with the display resolution of 2560×1440. Other threshold values with a unit of pixel in this paper also depend on this device. Figure 4 shows the two types of linear features. As our system assumes that groups of strokes lie on planar canvases, it is hence a natural idea to extract such linear features for subsequent analysis. Only linear features that are visible in the current view are used for further processing as users typically do not use invisible features for sketching guidance.

We next exploit the linearity of the user-drawn strokes. Specifically, we detect straight segments in a given stroke. A straight segment is a contiguous part of a stroke which

is nearly straight (Figure 4). Such straight segments can be found by approximating a stroke as a polyline [50] and filtering out short segments (with length < 50 pixels in our implementation). For example, the circular stroke and the plus-sign strokes therein in Figure 4 are finally filtered out.

We detect the *collinearity* and *parallelism* relations between the obtained straight segments of all strokes and the linear features of the model (Figure 7). Note that collinearity is a special case of parallelism, however we consider them separately, since collinearity provides more informative cues than parallelism. The collinearity relation is detected between a stroke straight segment and a sharp model edge while the parallelism relation is detected between a stroke straight segment and a linear model feature (i.e., a sharp model edge or a plane normal). The relations are detected in the screen space. A straight segment and a linear feature are considered parallel if $|d \cdot d'| > 0.95$, where d and d' are the unit directions of these two segments. Two parallel segments are considered to be collinear if the distances between their endpoints and the fitted straight line are smaller than 50 pixels. An endpoint of a stroke is considered to have the *Attachment* relation with the model if it is on or near the model in the screen space. For more content-awareness, attachment has the following three levels, with decreasing priority: corners, sharp model edges, and the surface of the model. In each level, the attachment distance threshold is set as 50 pixels.

We detect the above relations in screen space by matching between the projected 3D features onto screen space and the drawn strokes. This is due to the following observation: when the user draws 2D strokes, although s/he perceives the model and the drawn strokes in 3D, s/he mainly uses their 2D information as reference. To a certain extent, this alleviates problems due to weak drawing skill or inherent perceptual biases in the estimates of foreshortened shapes and dimensions [52].

4.2 3D Planar Canvas Inference

Matching between the extracted model features and the (stroke segments) determines how to infer a 3D planar canvas for back-projecting 2D strokes to 3D. However, due to ambiguities in the input, some detected relations might actually be unwanted. Therefore, it might not be feasible to find a canvas that meets all the detected relations. Nevertheless, we observe that, despite input ambiguities, if the correct canvas is found, in most situations the user's intended relations are satisfied. Motivated by this observation, we propose a two-step approach to estimate the canvas.

First, we construct a set of candidate canvases that can be obtained from some combination of the relations. We list the combinations that are able to yield valid canvases, which are illustrated in Figure 7 along with the examples of combinations that do not yield valid canvases. These valid combinations (Figure 7 (a)-(f) middle) provide the normal and position information for estimating a canvas, while the invalid ones fail to provide sufficient information. We examine all the possible combinations and check whether they are able to form candidate canvases according to the valid conditions listed below. Please see the appendix in the supplemental material for the details of canvas computation.

Two collinearity. (Figure 7(a))
The corresponding two sharp model edges span a 3D plane, no matter if they are parallel or not.

One collinearity + one parallelism. (Figure 7(b))
The corresponding sharp edge and linear feature are not parallel.

One collinearity + one attachment. (Figure 7(c))
The attached 3D point is not on the corresponding sharp edge (or its extension).

One parallelism + two attachments. (Figure 7(d))
The two attached 3D points are not at the same position, and the line connecting them is not parallel with the corresponding shape linear feature.

Two parallelism + one attachment. (Figure 7(e))
The corresponding two shape linear features are not parallel.

Three attachments. (Figure 7(f))
The three attached 3D points are not collinear.

Since each stroke straight segment might have collinearity (resp., parallelism) relations with multiple model sharp edges (resp., linear features), there are often multiple candidate canvases, whose number typically increases with the number of strokes. Next we rank the candidate canvases according to how well they preserve the detected relations. Given a canvas, the 2D strokes can be converted into 3D by a simple back-projection to the canvas. We then check whether the resulting 3D straight segments still preserve the previously detected relations in 2D. We check the parallelism and collinearity relations using the same approach for detection, but in 3D. The distance threshold for collinearity is computed by projecting a segment of 50-pixel length to a canvas orthogonal to the eye direction. Further, the attachment relation can also be validated in 3D. Its threshold is computed in the same way. For each canvas and the corresponding resulting 3D sketch, we compute the following scores to measure how well the 3D sketch complies with the detected relations:

$$
\begin{aligned}
&E = w_c E_c + w_p E_p + w_a E_a, \text{ with} \\
&E_c = \frac{\sum_{s_i \in \mathcal{S}'_c} L(s_i)}{\sum_{s_i \in \mathcal{S}_c} L(s_i)}, E_p = \frac{\sum_{s_i \in \mathcal{S}'_p} L(s_i)}{\sum_{s_i \in \mathcal{S}_p} L(s_i)}, \\
&E_a = \frac{\alpha_v |\mathcal{A}'_v| + \alpha_e |\mathcal{A}'_e| + \alpha_f |\mathcal{A}'_f| + \alpha_s |\mathcal{A}'_s|}{\alpha_v |\mathcal{A}_v| + \alpha_e |\mathcal{A}_e| + \alpha_f |\mathcal{A}_f| + \alpha_s |\mathcal{A}_s|},
\end{aligned}
\tag{1}
$$

where E_c, E_p, and E_a are the relation preservation scores of collinearity, parallelism, and attachment, respectively. These three terms measure the percentage of strokes which share relations with the model. A higher value indicates a higher possibility that the 3D sketch is the desired one. w_* are the weights used to control the relative contributions of the different types of relations. In our system we use a fixed set of weights: $w_c = 2.0$, $w_p = 1.0$, and $w_a = 2.0$, emphasizing collinearity and attachment relations, since they contain the positional information, which is more reliable. When computing E_c and E_p, $\mathcal{S}_c$ and $\mathcal{S}_p$ are the sets of 2D straight segments, each of which, denoted as s_i, shares a collinearity/parallelism relation, respectively, with a 3D linear feature. $\mathcal{S}'_c$ and $\mathcal{S}'_p$ are the sets in which each segment s_i retains a collinearity/parallelism relation

(with the corresponding model linear feature) after back projection to 3D. $L(\cdot)$ denotes the length of a stroke straight segment, which serves as a weighting scheme to emphasize long strokes. If more strokes retain collinearity (parallelism) relations after back projection, E_c (E_p) will be closer to 1. $\mathcal{A}_*$ are the sets of attachment points with the subscripts $\{v, e, f, s\}$ indicating whether the attachment is to a corner, edge, model face, or established 3D sketch. Similarly, $\mathcal{A}'_*$ denotes the attachment points sets containing points that remain attached to the corresponding 3D position after back projection. The weights α_* are used to control the relative contribution of different attachment types, and are set as $\alpha_v = 1.0$, $\alpha_e = 0.6$, $\alpha_f = 0.3$, and $\alpha_s = 0.6$, according to their reliability. The reliability is based on the difficulty to form the attachment relation. For example, corner attachment has a stricter criteria, and thus it is more likely to be the user's intention if detected.

Besides detecting the relations between 2D strokes and the model, we also detect relations between the 2D strokes and previously established 3D sketches (e.g., the attachment relations considered in Equation 1). After a new set of strokes are converted into 3D, we only store the normal of their canvas plane for further detection of parallelism relation and do not consider the new 3D edges introduced by these strokes as linear features for detecting relations. This is because the directions of those reliable new 3D edges are usually the same as existing model sharp edges. When the input shape contains many geometry details, the number of detected linear features could be very large (see Figure 8), possibly making our algorithm sensitive to the number of input strokes. To alleviate this problem, we cluster the detected lines according to their orientations using Mean-Shift with the bandwidth of 0.3 and only keep the dominant (top 10 in our experiments) groups.

An important assumption of our canvas inference algorithm is that the input strokes depict a planar shape, which can be determined by a combination of detected collinearity, parallelism and attachment relations. This assumption, however, does not always hold in practice. For example, the user may draw a stroke that depicts a linear shape (e.g., a horizontal support between two legs of a chair) or decorative strokes (on an existing canvas) from which no reliable relation can be detected. We extend our algorithm to handle such cases. For linear shapes, we use the detected relation to determine the 3D line on which the stroke is drawn. For decorative strokes, we simply project them onto the shape's dominant planes and the previously established sketch canvases that are underneath or nearby. The canvas with the smallest depth is chosen for back projection.

5 Results and Discussions

To show the effectiveness of our tool for 3D concept redesign, we invited three art students to extensively use our system. All of them had good 2D drawing skills but zero experience in 3D sketching and little knowledge in 3D modeling. They were briefed on how to use our system in a one-hour training session and then went home to do the 3D sketching without any further support from the authors. Each of them was given 9 models and returned us about 8 3D sketches around the models within 1 or 2 days. Figures 1 and 14 show the models given to them and the selected artworks of theirs. We were told they spent on average about one hour creating each sketch, including coming up with idea, testing, and real sketching. When they were asked to reproduce some of the sketches (e.g., Figure 1(b)), it took less than 10 minutes.

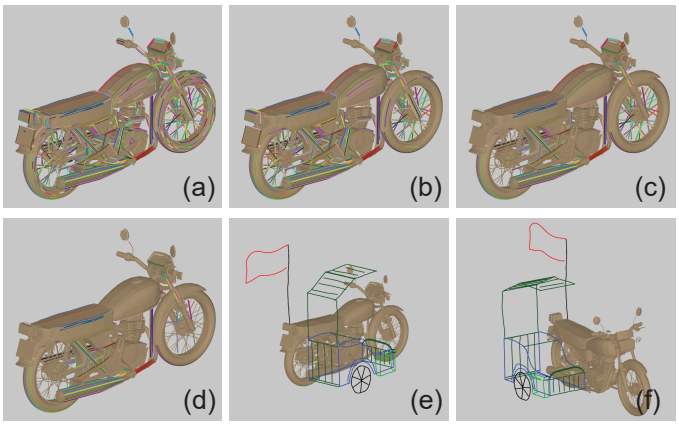


Fig. 8. Our algorithm filters out noisy linear features by keeping the dominant oriented groups of linear features when the model complexity increases. (a) to (d): 10%, 30%, 50%, 70% features were discarded, respectively. (e) and (f) shows a possible 3D sketching result shown in different views) around this model.

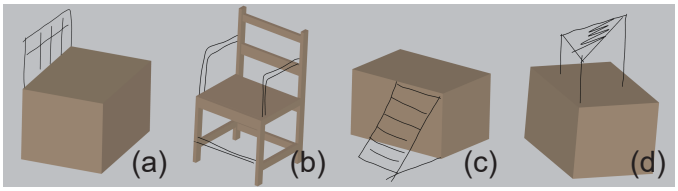


Fig. 9. Pretest examples used in our pilot study (Task I).

The artists commented that although they had no prior 3D sketching experience, all of them were interested in using our tool for quick 3D concepts and idea presentation in the future. One artist said *"although it needs time to understand the logic of the software, it usually gives me my expectation after I am familiar with it"*. This was echoed by another artist, saying that *"the interaction with this system is fluent, and the results are overall reasonable. Although it takes more time to create new planes, I got more experienced with it after a few practices"*. They also suggested new features, for example, making the copy-and-paste feature available for a group of canvases, instead of only individual ones in our current implementation.

5.1 Pilot Study

We conducted a pilot study. We would like to see if *novice* users are able to reproduce given 3D sketches and create their own designs using our tool. We did not evaluate the quality of the generated sketches since they are more dependent on the drawing skills of the users.

Participants. We recruited eight participants to help with our study. Among them, two participants were familiar with 3D modeling software like Maya, while others had no relevant experience. Three participants had reasonable good 2D drawing skill. Since the performance of 3D browsing is not of our interest, at the beginning of the study we asked the participants to practice until they got familiar with camera control.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2860016, IEEE Transactions on Visualization and Computer Graphics

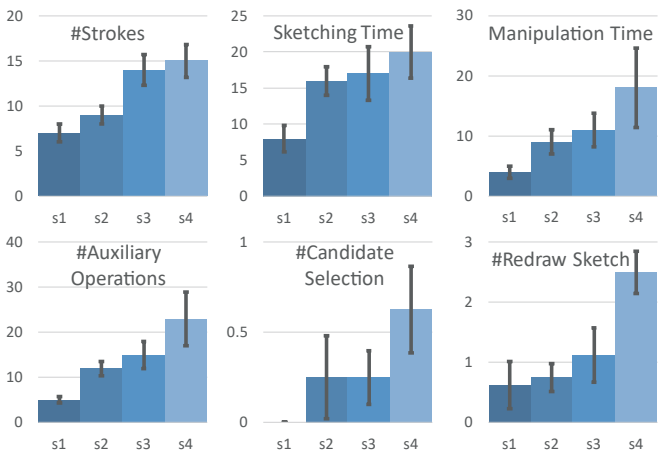


Fig. 10. Statistics of Task I. s1-4 refers to the target sketches (a-d) in Figure 9. The auxiliary operations include undo, erasing, deletion, single-click, and plane confirmation. The error bars are the standard errors of the mean.

Apparatus. The study was conducted on a 13.3-inch Wacom Cintiq companion 2 tablet with an i5 1.7Ghz processor. This device supports both finger and pen input. We recommended pen input, since it led to a more natural pencil-and-paper sketching experience.

Task I. The first task was to evaluate how fast the participants can learn the basic workflow of our system. We asked the participants, after a short introduction, to reproduce the simple 3D sketches shown in Figure 9 using first sketching and then plane confirmation, without explicit plane selection or transformation.

Each participant was first briefed on how a 3D plane can be mathematically determined and then shown a live demo illustrating different cases in Figure 7. The plane-by-plane sketching workflow without pre-selection of canvas as well as various auxiliary operations (e.g., undo, erasing, deletion, line-by-line sketching) were also demonstrated. This tutorial session lasted approximately 15 mins. Then each participant was given a 15-min practice session, in which they first tried our tool freely and then were asked to reproduce a target 3D sketch (different from those used in the pretest and posttest), with our help if needed.

Next each participant was asked to perform a pretest of reproducing four simple 3D sketches on his or her own, which were carefully designed to involve the core basic features of our tool. The target sketches (Figure 9) were given on a laptop such that the user could preview the sketches in 3D. In the first sketch, the participants were asked to sketch on a plane that could be directly derived from the model but invisible in the current view. The second sketch trained the basic skills of plane-by-plane sketching without explicit plane selection. The third sketch involved the creation of strokes on a hinge plane that did not exist in the model but required an indirect anchoring (i.e., by first creating the side plane). In the last sketch, the participants were expected to create a floating plane which required three single strokes to create three attachments for anchoring the plane.

Statistics. We recorded the following information during the tests: the total reproduction time of each sketch, the time spent on sketching and other operations, the number of strokes, and the number of various kinds of operations. Since our tool is not aimed for precise 3D drawing, it is difficult for the users to produce target sketches exactly. We manually checked whether the target sketches were reproduced reasonably well. In case the participants could not achieve certain intended effects after several trials, they could approach us for hints.

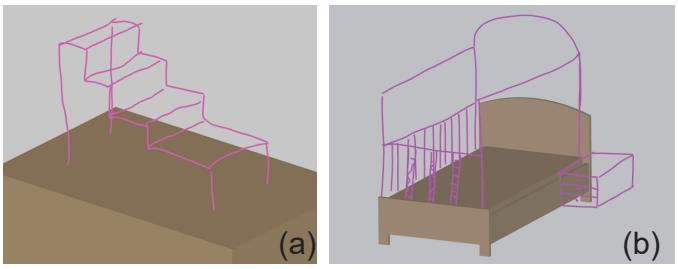


Fig. 11. Posttest examples used in our user study (Task II).

For each of the above tested cases, most of the participants finished the first three tasks in an average time of less than 2 mins per task. We observe that our tool needs some basic 3D modeling knowledge to get familiar with. For instance, in the last case, some of the participants took a longer time ($3 \sim 7$ mins) as they misunderstood the reference image and tried to draw the detail sketches on top of a non-existing plane (they created two side planes instead of constructing the top floating plane using three attachments). Not surprisingly, the two participants with previous 3D modeling experience were able to generally learn and use our tool much faster. Nevertheless, all participants learned our tool fairly quickly in this task. Figure 10 shows the average numbers of strokes and auxiliary operations, and the averaged timings (in seconds) for sketching and camera control. All the participants performed consistently well.

Task II. In this task, we required each participant to reproduce two target 3D sketches (Figure 11) with increasing complexity than those in the pretest.

Since our core technique is the automatic plane inference, in this task, we compared our system to a basic version of manual plane selection as used in the *Mental Canvas* system [1]. *Mental Canvas* is a sketch-based system for conceptual design. It has several interfaces to facilitate the sketch creation process, e.g., the automatic best view selection. Their interface that is most relevant to our work is its manual mode. In the manual mode of the *Mental Canvas* system, three sets of equally-spaced axis-aligned (i.e., the x, y, and z) 3D planes, 6 planes per direction, were exposed to the user. The user could switch between the three sets of planes by clicking a button. Once a plane was selected, the user could rotate or translate it and its associated 3D sketches, but with no snapping or inference. The original 3D model was also exposed to the user for visual reference. The user can choose to hide the 3D model if necessary.

The participants were asked to draw the sketches in Figure 11 using each of the two modes, the inference mode and the manual mode, whose presentation order was counterbalanced. We collected the same statistics as in Task I. Figure 12 shows the results. From Figure 12, it is not unexpected to see that the sketching timings in both modes were similar, and the manual mode even got slightly less

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2018.2860016, IEEE Transactions on Visualization and Computer Graphics

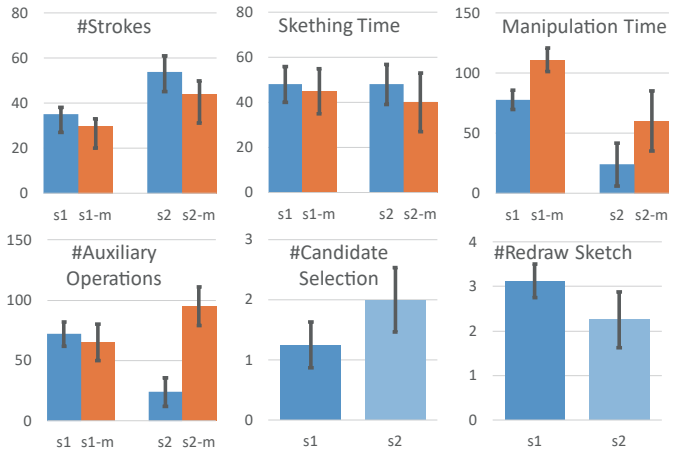


Fig. 12. Statistics of Task II. s1 and s2 refer to the target sketches (a-b) in Figure 11, and "-m" means results obtained using the manual plane selection and manipulation mode. The auxiliary operations include undo, erasing, deletion, single-click, and plane confirmation. The error bars are the standard errors of the mean.

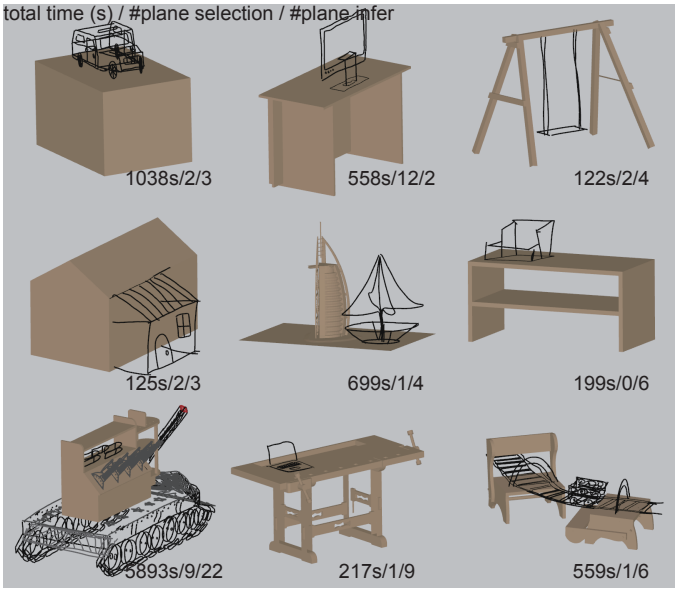


Fig. 13. Sketches created by individual users for Task III. Not all the examples are shown.

sketching time as the sketching was direct once the canvas was fixed. However, when using the manual mode, the participants spent a lot of more time in translating and rotating the canvases to put them at desired locations. The t-test showed a statistically significant difference in canvas manipulation time (Figure 12) for both target sketches in Task II ($s1 : p < 0.009; s2 : p < 0.012$). For auxiliary operations, there was a significant difference when creating sketch 2 ($p < 0.009$). These statistics indicate that our interface can greatly reduce the user interaction for canvas manipulation compared with the traditional interface. While these statistics show the superiority of our system, the manual plane manipulation could be a complementary tool for our system to help users express themselves more freely. We also noticed that the exposure of the 3D model was largely helpful in determining the position and scales of the drawn sketches in the manual mode.

An important finding is the robustness of our canvas inference algorithm. Our system infers and dynamically updates the canvas during the drawing procedure. The following two statistics can reveal the correctness of the inference algorithm, i.e., the number of candidate selection and the number of sketch redrawing (Figure 10 and Figure 12). It can be easily seen from these figures that very seldom did a participant need to enter the candidate selection mode to select a correct canvas, or redraw a sketch to get a better inference. Here sketch redrawing is recognized by a sequence of undo operations, or sketch deletion operation in the sketching mode. This criteria actually is quite loose and therefore the actual number of sketch redrawing might be even lower.

The most time consuming part of our system, as observed from Task II, is the same as in Task I, i.e., most of the participants did not have a good sense of perspective drawing, and thus often drew strokes without paying sufficient attention to the model features. Another fact is that when the participants wanted to draw floating planes or planes that form certain angles with existing planes, they sometimes ignored the mathematical aspects about unique determination of a 3D plane and directly drew a floating plane such as the example shown in case 3 of Figure 7 (d) right. Again, the statistics showed that our tool is suitable for novice users. Augmented with a short period of training, all participants learned to draw well. The quality of the created sketches vary with different users, but there seems no significant difference between the proposed and compared tools.

Task III. As the last part of our pilot study, the participants were asked to freely design renovations of 3D models. For completeness, they were also introduced to the other sketching workflow (plane selection followed by in-plane sketching) and canvas transformation, and allowed to use a mixture of these workflows to redesign. In total we include 9 3D models, including cube, shelf, child, swing, house, workbench, architecture, and table (Figure 13). Each participant was given three of the 9 models so that each model was sketched over multiple times. To provide inspiration, each model came with one possible sketch, which the participants did not need to follow. Figure 13 shows a gallery of sketches created by the participants. The participants typically took several minutes to a dozen minutes to create such redesigns of 3D models. The numbers of inferred planes and explicitly selected planes indicate that inference-enabled sketching was always preferred by our participants. We observed that the sketches created by our participants were interesting but not very complex. We confirmed with them that they were mainly limited by their creativity instead of the expressiveness of our tool.

We conducted a usability test to further evaluate our system. At the end of the study, each participant was asked to complete a questionnaire, which included a standard system usability scale (SUS) with 10 questions [53]. 6 of the participants found our system easy to use and felt confident using the system. Most of the participants found that the technical tutorial was necessary before using the system but the tutorial was easy to follow and intuitive. Also, 7 of the participants found that various functions in our system were well integrated. On the other hand, a majority

Fig. 14. Gallery of 3D sketches generated by our system.

of the users pointed out that although having 3D models can largely help the creation and anchoring of new contents, the planar assumption somehow restricted the expressive power as a general 3D sketching tool. Moreover, in some circumstances, novice user will not and even do not want to understand and learn the fundamental concepts behind how the canvases are created. In such cases, a deterministic 3D sketching tool such as mental canvas might be easier to learn.

Our algorithm involves adjustable parameters (e.g., for detecting collinearity, parallelism, attachment relations), however, all the results were achieved using the default fixed set of parameter values. While the existing 3D sketching systems (e.g., [1], [5]) might be able to generate visually similar results, they mostly require a significantly larger amount of user intervention (c.f. Task II), which is greatly re-

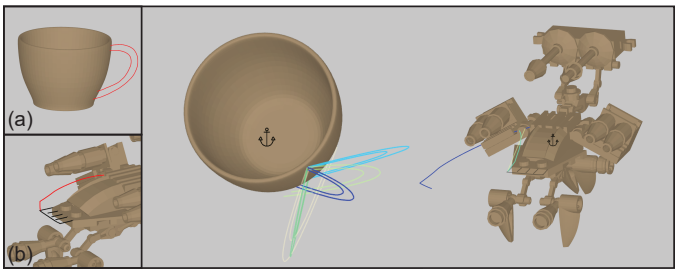


Fig. 15. Our system does not handle well (a) freeform shapes containing rich nonlinear/non-planar features or (b) complex models with excessive features. The images on the left show the input sketches, highlighted in red. The middle and right images show the candidate 3D sketches, differentiated by the colors. The default ones are with navy blue.

duced with the help of the rich geometric information from the 3D guidance models. As sketching is inherently noisy, errors due to either the input devices (e.g., difficult to use a mouse for precise drawing) or perceptual foreshortening biases are unavoidable. In such circumstances, our method resorts the user to the candidate selection phase to anchor the correct planes. Nevertheless, throughout experiments, we found our tool is able to tolerate such errors well. We record the number of candidate selection operations in all the three tasks, as can be seen from Figures 10 and 12. The average number in Task III was also kept at a low rate (< 3 per example, see also in the accompanying video).

Limitations. Our tool allows users to express themselves quickly, but at the cost of imprecise drawings. The resulting sketches might not be well connected anywhere and their regularity can be improved. Such effects might be achieved by using snapping-like geometric constraints [54] for incremental or global sketch beautification. Our current sketches are always embedded in 3D planes, since our canvases are planes. For creating non-planar curves, we might adopt an approach similar to [5], [9], [21] and use our current sketches as a scaffold. We have tested our tool on relatively simple models only. When a 3D model is complex (Figure 15 (b)), it might provide excessive features, which might confuse our algorithm. A possible solution is to provide an interactive tool so that users can first select model parts or features of interests for inference guidance. In addition, since our system mainly uses linear and planar features of 3D models, it is not suitable for freeform or organic shapes with rich nonlinear or non-planar features (e.g., mug, as shown in Figure 15 (a)). Lastly, since our current implementation requires clean models as input, it is not robust against the noise in input models and thus cannot be directly applicable to models represented as noisy 3D point clouds.

6 Conclusion and Future Work

We have presented a model-guided 3D sketching system. Our tool is particularly useful for quick concept redesign of existing 3D models by sketching renovations. We adopt an incremental plane-by-plane sketching interface. A key feature of our system is that users are able to sketch the renovations of a 3D model without explicit canvas selection, leaving our system to automatically infer a desired planar canvas by correlating a group of sketched strokes with the model. A more traditional sketching workflow involving explicit plane selection is naturally integrated into a single system. A pilot study confirmed that our tool is simple and easy to use, and allows quick sketch-based redesign of various man-made objects. It would be promising to integrate our tool into the 3D modeling software, e.g., SketchUp, for early stage design, or 3D animation software, e.g., Autodesk Maya, for roughly illustrating the key frames.

Fig. 16. Physical realization of 3D sketches around 3D-printed models.

We have focused on the implementation of the core sketching operations for the current prototype. Our tool will be more efficient if more auxiliary features are included such as more powerful canvas organization tools like those in *Mental Canvas* [1], automatic best-view selection for previewing canvas candidates, and those suggested by the artists who tested our system. Our rendering of 3D sketches may be improved by considering stroke visibility, possibly with the help of user-specified visibility masks. Our plane inference algorithm might be easily extended to exploit the similarity between user drawn strokes and feature curves on a 3D model [8], [9]. The plane evaluation could also be improved by considering existing sketches and using an optimization approach to find the best plane which fits the existing content. As discussed earlier, we are interested in achieving non-planar 3D curves and more precise drawings, without sacrificing too much the ease of use. How to turn our 3D sketches into 3D surface models is also an interesting but challenging problem to explore in the future, since our sketches are relatively rough and need to be refined before plugging them into existing techniques for surfacing curve networks [30]. Finally, we are very interested in achieving physical realization of the created 3D sketches so that they can be integrated with real-world objects. Figures 1 and 16 show our proof of concept, where 3D models are 3D-printed and 3D sketches are 2D-printed onto transparency sheets plane by plane.

ACKNOWLEDGEMENTS

We would like to thank Jiyuan Zhang for the initial exploration of this project, the reviewers for their constructive comments, and the user study participants for their time. This work was partially supported by grants from NSFC (61502306, 61602310, 61522213, 61761146002, 61861130365), Guangdong Science and Technology Program (2015A030312015), Shenzhen Innovation Program (JCYJ20170302154106666, KQJSCX20170727101233642, JCYJ20151015151249564), the Research Grants Council of HKSAR (CityU11300615, CityU11204014, HKUST16201315, HKUST16210718), NSERC, ACIM-SCM, and the Open Project Program of State Key Lab. of Virtual Reality Technology and Systems, Beihang University (VRLAB2018C11).

REFERENCES

[1] J. Dorsey, S. Xu, G. Smedresman, H. Rushmeier, and L. McMillan, "The mental canvas: A tool for conceptual architectural design and analysis," in *PG '07*, 2007, pp. 201–210.

[2] S.-H. Bae, R. Balakrishnan, and K. Singh, "Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models," in *UIST '08*, 2008, pp. 151–160.

[3] ——, "Everybodylovessketch: 3d sketching for a broader audience," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, 2009, pp. 59–68.

[4] J. McCrae, N. Umetani, and K. Singh, "Flatfitfab: Interactive modeling with planar sections," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14. New York, NY, USA: ACM, 2014, pp. 13–22.

[5] R. Schmidt, A. Khan, K. Singh, and G. Kurtenbach, "Analytic drawing of 3d scaffolds," in *ACM Transactions on Graphics*, vol. 28, no. 5, 2009, p. 149.

[6] Y. Zheng, H. Liu, J. Dorsey, and N. J. Mitra, "Smartcanvas: Context-inferred interpretation of sketches for preparatory design studies," *Computer Graphics Forum (Proceedings of Eurographics 2016)*, 2016.

[7] Y. Kim and S.-H. Bae, "Sketchingwithhands: 3d sketching handheld products with first-person hand posture," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 2016, pp. 797–808.

[8] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. Mccrae, and K. Singh, "True2form: 3d curve networks from 2d sketches via selective regularization," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

[9] C. De Paoli and K. Singh, "Secondskin: Sketch-based construction of layered 3d models," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 126:1–126:10, Jul. 2015.

[10] L. B. Kara and K. Shimada, "Sketch-based 3d-shape creation for industrial styling design," *IEEE Comput. Graph. Appl.*, vol. 27, no. 1, pp. 60–71, 2007.

[11] X. A. Chen, S. Coros, J. Mankoff, and S. E. Hudson, "Encore: 3d printed augmentation of everyday objects with printed-over, affixed and interlocked attachments," in *UIST '15*, 2015, pp. 73–82.

[12] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa, "Sketch-based shape retrieval," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 31, no. 4, pp. 31:1–31:10, 2012.

[13] S. Tsang, R. Balakrishnan, K. Singh, and A. Ranjan, "A suggestive interface for image guided 3d sketching," in *CHI '04*, 2004, pp. 591–598.

[14] L. B. Kara and K. Shimada, "Construction and modification of 3d geometry using a sketch-based interface," in *SBIM '06*, 2006, pp. 59–66.

[15] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Fibermesh: designing freeform surfaces with 3d curves," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 41.

[16] J. Leung and D. M. Lara, "Grease pencil: Integrating animated freehand drawings into 3d production environments," in *SIGGRAPH Asia 2015 Technical Briefs*, 2015, pp. 16:1–16:4.

[17] C. Grimm and P. Joshi, "Just drawit: a 3d sketching system," in *SBIM '12*, 2012, pp. 121–130.

[18] D. Bourguignon, M.-P. Cani, and G. Drettakis, "Drawing for illustration and annotation in 3d," in *Computer Graphics Forum*, vol. 20, no. 3, 2001, pp. 114–123.

[19] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. Fitzmaurice, A. Khan, and B. Buxton, "Creating principal 3d curves with digital tape drawing," in *CHI '02*, 2002, pp. 121–128.

[20] J. Schmid, M. S. Senn, M. Gross, and R. W. Sumner, "Overcoat: an implicit canvas for 3d painting," *ACM Trans. Graph.*, vol. 30, pp. 28:1–28:10, 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964923

[21] V. Krs, E. Yumer, N. Carr, B. Benes, and R. Mech, "Skippy: Single view 3d curve interactive modeling," in *ACM Transactions on Graphics (SIGGRAPH 2017)*, 2017, p. to appear.

[22] R. Arora, R. H. Kazi, T. Grossman, G. Fitzmaurice, and K. Singh, "Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ," in *CHI 2018*, 2018.

[23] M. Xin, E. Sharlin, and M. C. Sousa, "Napkin sketch: handheld mixed reality 3d sketching," in *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*. ACM, 2008, pp. 223–226.

[24] P. Paczkowski, M. H. Kim, Y. Morvan, J. Dorsey, H. Rushmeier, and C. O'Sullivan, "Insitu: Sketching architectural designs in context," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 182:1–182:10, 2011.

[25] M. Lau, G. Saul, J. Mitani, and T. Igarashi, "Modeling-in-context: User design of complementary objects with a single photo," in *SBIM '10*, 2010.

[26] H. Lipson and M. Shpitalni, "Optimization-based reconstruction of a 3d object from a single freehand line drawing," *Computer-Aided Design*, vol. 28, no. 8, pp. 651–663, 1996.

[27] X. Chen, S. Kang, Y. Xu, J. Dorsey, and H. Shum, "Sketching reality: Realistic interpretation of architectural designs," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 2, p. 11, 2008.

[28] C. Zou, S. Chen, H. Fu, and J. Liu, "Progressive 3d reconstruction of planar-faced manifold objects with drf-based line drawing decomposition," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 2, pp. 252–263, 2015.

[29] L. Li, Z. Huang, C. Zou, C.-L. Tai, R. W. Lau, H. Zhang, P. Tan, and H. Fu, "Model-driven sketch reconstruction with structure-oriented retrieval," in *SIGGRAPH Asia 2016 Technical Briefs*, 2016.

[30] H. Pan, Y. Liu, A. Sheffer, N. Vining, C.-J. Li, and W. Wang, "Flow aligned surfacing of curve networks," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 127:1–127:10, 2015.

[31] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, no. 1, pp. 85–103, 2009.

[32] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes, "SKETCH: An interface for sketching 3D scenes," in *Proceedings of ACM SIGGRAPH*, 1996, pp. 163–170.

[33] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3D freeform design," in *SIGGRAPH '99*, 1999, pp. 409–416.

[34] O. A. Karpenko and J. F. Hughes, "Smoothsketch: 3d free-form shapes from complex sketches," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 589–598, 2006.

[35] Y. Gingold, T. Igarashi, and D. Zorin, "Structured annotations for 2D-to-3D modeling," *ACM Transactions on Graphics*, vol. 28, no. 5, p. 148, 2009.

[36] Y. Li, X. Luo, Y. Zheng, P. Xu, and H. Fu, "Sweepcanvas: Sketch-based 3d prototyping on an rgb-d image," in *UIST '17*, 2017.

[37] K. Singh and E. Fiume, "Wires: a geometric deformation technique," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 405–414.

[38] J. Jorge, N. Silva, and T. Cardoso, "Gides++," *Proceedings of 12th Encontro Português de Computação Gráfica*, vol. 2, no. 9, 2003.

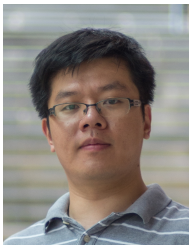
[39] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1142–1147, 2005.

[40] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based mesh augmentation," in *SBIM '05*, 2005.

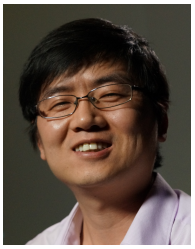
[41] L. B. Kara, C. M. D'Eramo, and K. Shimada, "Pen-based styling design of 3d geometry using concept sketches and template models," in *SPM '06*, 2006, pp. 149–160.

[42] H. Shin and T. Igarashi, "Magic canvas: interactive design of a 3-d scene prototype from freehand sketches," in *Proceedings of Graphics Interface 2007*, 2007, pp. 63–70.

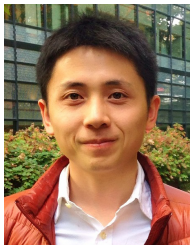
[43] J. Lee and T. Funkhouser, "Sketch-based search and composition of 3D models," in *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, Jun. 2008.
[44] K. Xu, K. Chen, H. Fu, W.-L. Sun, and S.-M. Hu, "Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models," *ACM Transactions on Graphics*, vol. 32, no. 4, p. Article No. 123, 2013.
[45] L. Fan, R. Wang, L. Xu, J. Deng, and L. Liu, "Modeling by drawing with shadow guidance," in *Computer Graphics Forum*, vol. 32, no. 7, 2013, pp. 157–166.
[46] K. Shoemake, "Arcball: a user interface for specifying three-dimensional orientation using a mouse," in *Graphics Interface*, vol. 92, 1992, pp. 151–156.
[47] T. Igarashi and J. F. Hughes, "A suggestive interface for 3D drawing," in *UIST '01*, 2001, pp. 173–181.
[48] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends. Comput. Graph. Vis.*, vol. 3, no. 3, pp. 177–280, 2008.
[49] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or, "iwires: an analyze-and-edit approach to shape manipulation," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 33.
[50] P. Rosin, "Techniques for assessing polygonal approximations of curves," *IEEE TPAMI*, vol. 19, no. 6, pp. 659–666, 2002.
[51] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.
[52] R. Schmidt, A. Khan, G. Kurtenbach, and K. Singh, "On expert performance in 3d curve-drawing tasks," in *SBIM '09*, 2009, pp. 133–140.
[53] J. Brooke, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.
[54] E. A. Bier, "Snap-dragging in three dimensions," in *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 2. ACM, 1990, pp. 193–204.



Pengfei Xu is an Assistant Professor of College of Computer Science and Software Engineering at Shenzhen University. He received his BS degree in Math from Zhejiang University, China, in 2009 and his Ph.D. degree in Computer Science from Hong Kong University of Science and Technology in 2015. His primary research lies in Human Computer Interaction and Computer Graphics.



Hongbo Fu received a BS degree in information sciences from Peking University, China, in 2002 and a PhD degree in computer science from the Hong Kong University of Science and Technology in 2007. He is an Associate Professor at the School of Creative Media, City University of Hong Kong. His primary research interests fall in the fields of computer graphics and human computer interaction. He has served as an Associate Editor of The Visual Computer, Computers & Graphics, and Computer Graphics Forum.



Youyi Zheng is a Researcher (PI) at the State Key Lab of CAD&CG, College of Computer Science, Zhejiang University. He obtained his PhD from the Department of Computer Science and Engineering at Hong Kong University of Science & Technology, and his M.Sc. and B.Sc. degrees from the Department of Mathematics, Zhejiang University. His research interests include geometric modeling, imaging, and human-computer interaction.

Karan Singh is a Professor of Computer Science at the University of Toronto. He co-directs the graphics and HCI lab, DGP, has over 100 peer-reviewed publications, and has supervised over 40 MS/PhD theses. His research interests lie in interactive graphics, spanning art and visual perception, geometric design and fabrication, character animation and anatomy, and interaction techniques for mobile, Augmented and Virtual Reality (AR/VR). He has been a technical lead for the Oscar award winning software Maya and was the R&D Director for the 2004 Oscar winning animated short Ryan. He has co-founded multiple companies in the interactive graphics space including Arcestra (architectural design), Conceptualiz (surgical planning), and JanusVR (AR/VR).

Hui Huang is a Distinguished Professor of Shenzhen University, where she directs the Visual Computing Research Center in College of Computer Science and Software Engineering. She received her PhD in Applied Math from The University of British Columbia in 2008 and another PhD in Computational Math from Wuhan University in 2006. She is the recipient of NSFC Excellent Young Researcher program and Guangdong Technology Innovation Leading Talent award in 2015. Her research interests are in computer graphics and scientific computing, focusing on point-based modeling, geometric analysis, 3D acquisition and creation.

Chiew-Lan Tai is a Professor of Computer Science at the Hong Kong University of Science and Technology. She received the BSc degree in mathematics from the University of Malaya, MSc in computer & information sciences from the National University of Singapore, and DSc degree in information science from the University of Tokyo. Her research interests include geometry processing, computer graphics, and interaction techniques.