# Digital Styling for Designers: 3D Plane-Symmetric Freeform Curve Creation Using Sketch Interface

Seok-Hyung Bae[1,2], Ryugo Kijima[2], and Won-Sup Kim[3]

[1] Virtual Reality Research Center, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
bae@vr.kaist.ac.kr
http://vr.kaist.ac.kr/∼bae
[2] Virtual System Laboratory, Gifu University, 1-1 Yanagido, Gifu, 501-1193, Japan
{bae,kijima}@vsl.gifu-u.ac.jp
[3] Department of Industrial Design, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
iron@mail.kaist.ac.kr

**Abstract.** The 3D evaluation of design shapes is an essential step in product styling. Thus, physical models of final-stage design alternatives have been made in tradition, and recently the effort to substitute them with CAD models has been tried. Whereas, designers in the early phase of the design-development stage where almost design concepts are determined, still use raster-type 2D graphics S/Ws. It causes not only the difficulty of evaluating 3D shapes but also the serious severance of a digital dis-connectivity with downstream processes. This paper presents a method of directly constructing 3D plane-symmetric freeform curves with a sketch interface, as the first step of developing a sketch-based 3D-freeform shape creation S/W for designers. A curve drawn by the designer within the see-through box in an arbitrary perspective view, is simultaneously converted to a real space curve without the 3D ambiguity problem except only special cases to be specified.

## 1 Introduction

As the customer's respect to products is being transformed from functionality to design, the importance of *product styling* is highly increasing in the whole product-development process. The more aesthetic the style of products is, the more important to evaluate 3D shapes of them before actual production become. For the reason, making physical models made of clay, wood, resin, form urethane, etc., is common in design studios [16]. However, physical models can not always be used because they require a lot of money and time. Thus, recently, the effort to substitute physical models with computer models using CAD S/Ws, has been explored.

So far, this kind of endeavor has focused on only final (or semi-final) stage design alternatives, and designers in the early phase of the design-development

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 701–710, 2003.

stage where most of design concept is fixed still uses *raster-type* 2D graphics S/Ws. Although this kind of S/Ws enables designers 1) to *rapidly* generate various ideas, 2) to *flexibly* hybridize existing styles, and 3) to *sensuously* express virtual images in their minds, they have serious problems as follows: 1) the difficulty of *3D evaluation*, 2) *no digital connectivity* with downstream processes.

In this paper, as our first attempt to develop a *direct* 3D freeform-shape creation S/W allows the both of 3D evaluation and digital continuity, a method of constructing *3D plane-symmetric freeform curves* with a *sketch interface* is given. The designer-drawn freeform curves symmetric to the center plane–many products are plane symmetric including automobiles except little *avant-garde* styles–are simultaneously converted to 3D space curves without causing the 3D ambiguity in virtue of its *strong* constraint–the *plane symmetry* (a little special viewing-situations in which our method does not work will be specified in the context).

The method to be proposed has a *great* assumption that *well-trained* designers can *exactly* express 3D shapes on a 2D plane based on *design perspective*. Actually, professional industrial-designers practice and practice so that they can sketch the *accurate* plane images of virtual 3D objects because sketch is not only a means of presentation but also that of idea development [6]. We valued the designer's works based on design perspective as resultants from a *pretty* accurate *human* graphics-rendering pipe line.

The organization of this paper is as follows: related work to the direct creation of computer models including space curves will be given in Section 2. In Section 3, so-called a *perspective 3D sketch* scheme extended from the 2D sketch interface proposed in our companion paper [1] is suggested. In Section 4, the overview of a method converting 2D curves drawn in a perspective view to 3D curves is given, and then the actual calculation procedure of finding 3D points from 2D points is presented with special case treatments in Section 5. The software implementation is given in Section 6 followed by discussions and conclusions in Section 7.

## 2 Related Work

We categorized research on *direct* 3D computer-model creations as following three based on the user interface: 1) *sketch-based* methods, 2) *suggestive* methods, and 3) *VR-interface* methods. Sketch-based methods have been mostly applied to creating CSG-like models composed of simple primitives [7][15][17] where the most interest was focused on the *primitive recognition* from a rough scribble, and the *topology reconstruction*. There were little studies about directly creating freeform-shapes: one is Teddy [11] for *rounded* freeform models, which, however, is not appropriate for product styling. A suggestive interface is to forecast possible subsequent operations to be executed by users, and to gives action alternatives [4][12]. It shows great possibility for creating simple polygon models, but is limited for freeform-shapes. Recently, much research with 3D interfaces (VR interfaces) has been done. However, VR techniques for the direct creation

of freeform shapes looks still immature in a practical point of view (heavy equipment, precision problem, etc.).

There are two representative studies about 3D space-curve creation: Cohen, et al. [3] suggested a method by sketching a curve and its shadow curve on the floor in a perspective view. Grossman, et al. [10] proposed the reverse way where a depth plane (surface) is constructed first, and the curve drawn on orthographic plane is projected on it.

## 3 Perspective 3D Sketch

The sketch interface proposed in this paper, *perspective 3D sketch*, is straightforwardly extended from the intuitive 2D sketch interface proposed in our companion paper [1], which enables the designer to freely create accurate curves intended by allowing repetitive scribbling (scribbles already drawn are spread and grayed out as the number of them increases) and inducing the designer's adaptation. Shown in Fig. 1 is the flowchart of the perspective 3D sketch scheme.

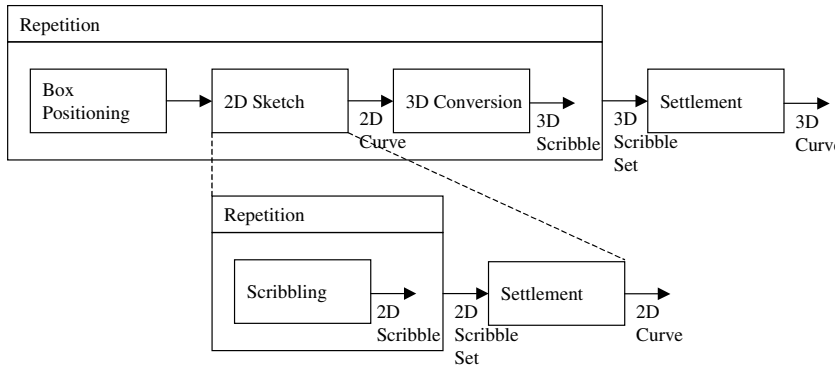


**Fig. 1.** Flowchart of perspective 3D sketch

The designer can choose an arbitrary perspective view by rotating the *see-through* box provided as a *reference unit* for estimating the dimension of 3D curves to be projected with it [6]. Then he/she draws a curve, $\tilde{\mathbf{c}}(t)$, on 2D image plane (Fig. 2(a)). By repetition of the previous procedure, a set of 3D curves, $\{\mathbf{c}_i(t)\}$, based on the designer's adaptation are generated (Fig. 2(b)), and at last, a final curve, $\bar{\mathbf{c}}(t)$, is settled (Fig. 2(c)).

## 4 Conversion of 2D Curve to 3D

The *pinhole-camera model* widely used in computer graphics and machine vision is composed of the *optical center*, $\mathbf{e}$, and the *retinal* (or image) *plane*, $\Pi_R$ [9].

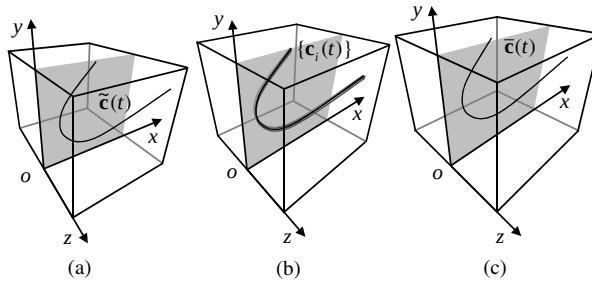


**Fig. 2.** Illustrative example of perspective 3D sketch

Between an arbitrary 3D point, $\mathbf{p}$, and its projection point, $\tilde{\mathbf{p}}$, there is a relation–a kind of *morphism*–as written as:

$$\tilde{\mathbf{p}}^w = \mathbf{H}\mathbf{p}^w \tag{1}$$

where $\mathbf{H}$ is the *perspective projection matrix* ($rank(\mathbf{H}) = 3$), $\mathbf{p}^w$ and $\tilde{\mathbf{p}}^w$are the *homogeneous coordinates* of $\mathbf{p}$ and $\tilde{\mathbf{p}}$, respectively.

In general, the inverse problem of perspective projection–finding $\mathbf{p}$ from $\tilde{\mathbf{p}}$–is an *under determined* problem, which has an infinite numbers of solutions. In this study, the above *3D ambiguity* is resolved by imposing the *plane-symmetry* condition, and a unique solution can be calculated. Fig. 3 shows our system configuration to be considered where $\mathbf{q}$ is the intersection point of the see-through box and the optical ray passing through $\tilde{\mathbf{p}}$ and $\mathbf{p}$.

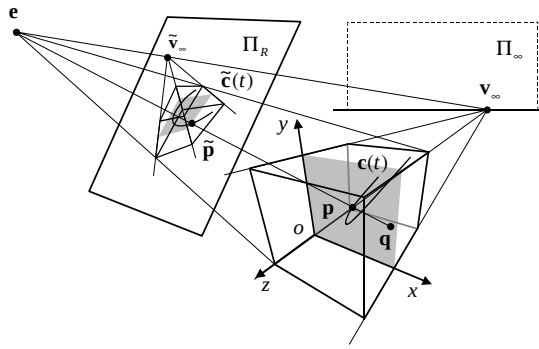


**Fig. 3.** System configuration for inverse perspective projection

The overall procedure of converting a 2D curve drawn by the designer to a 3D plane-symmetric freeform curve is as follows: 1) polygonizing a 2D parametric-curve, 2) matching a 2D-point pair, 3) finding a 3D-point pair on see-through

box, 4) finding a 3D-point pair looked for, and 5) creating a resulting 3D parametric-curve.

### 4.1 Parametric-Curve Polygonization

Given a parametric 2D curve, $\tilde{\mathbf{c}}(t)$, is sampled, and 2D point set, $\{\tilde{\mathbf{p}}_i(\tilde{x}_i, \tilde{y}_i)\}$, is obtained by the parametric-curve polygonization algorithm [5][13].

### 4.2 Matching 2D-point Pair

Using the 3-point perspective [6], generally used by industrial designers, a 2D-point pair expected to symmetric to the center plane in 3D space, can be acquired. First, the vanishing point, $\tilde{\mathbf{v}}_\infty$, is simply found by extending the edges of the see-through box parallel to z-axis, and given point, $\tilde{\mathbf{p}}_1 \equiv \tilde{\mathbf{p}}_i$, is mapped to its corresponding point, $\tilde{\mathbf{p}}_2$, by calculating the intersection point between a line passing through $\tilde{\mathbf{p}}_1$ and $\tilde{\mathbf{v}}_\infty$, and $\tilde{\mathbf{c}}(t)$ (or its linear interpolant obtained by polygonization) (see Fig. 4).

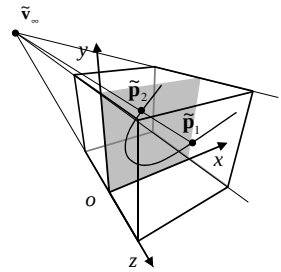


**Fig. 4.** 2D-point pair matching using vanishing point

### 4.3 Finding 3D-point Pair on See-Through Box

Instead of directly finding 3D-space points, $(\mathbf{p}_1, \mathbf{p}_2)$–as mentioned earlier, it is impossible to calculate them without the 3D ambiguity, their projection points on see-through box, $(\mathbf{q}_1, \mathbf{q}_2)$ are calculated from 2D points, $(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2)$, as follows (see Fig. 5(a) and refer our viewing system configuration given in Fig. 3) : For each 2D point, 1) finding *two* faces its optical ray pass through, 2) choosing the face between them, on which its 2D image point is closer to the center for more accurate calculation, 3) calculating two parameters of an *affine combination* in the 2D space, $(\mu, \eta)$, using two corresponding vanishing points, and 4) obtaining the 3D projection point on the see-through box using affine combination in the 3D space with the parameters previously calculated. For example, as shown

in Fig. 5(b), between the two projection faces of $\tilde{\mathbf{p}}_1$–$(F_{xy}^{front}, F_{yz}^{rear})$, $F_{xy}^{front}$ is selected, and $(\mu, \eta)$ are calculated using two vanishing points, $(\tilde{\mathbf{v}}_\infty^{'}, \tilde{\mathbf{v}}_\infty^{''})$, so that

$$\tilde{\mathbf{p}}_1 = (1-\mu)(1-\eta)\tilde{\mathbf{v}}_1 + \mu(1-\eta)\tilde{\mathbf{v}}_2 + (1-\mu)\eta\tilde{\mathbf{v}}_3 + \mu\eta\tilde{\mathbf{v}}_4 \tag{2}$$

where $\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \tilde{\mathbf{v}}_4\}$ are the 2D image vertices of $F_{xy}^{front}$. Then, the 3D projection point on $F_{xy}^{front}$, $\mathbf{q}_1$, is acquired as:

$$\mathbf{q}_1 = (1-\mu)(1-\eta)\mathbf{v}_1 + \mu(1-\eta)\mathbf{v}_2 + (1-\mu)\eta\mathbf{v}_3 + \mu\eta\mathbf{v}_4 \tag{3}$$

where $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ are the 3D vertices of $F_{xy}^{front}$.

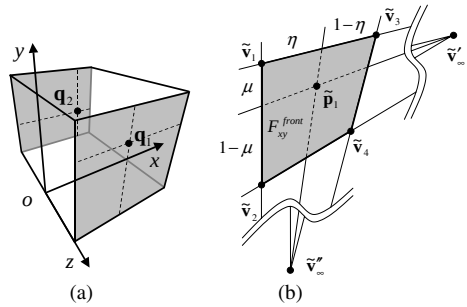


**Fig. 5.** Finding 3D-point pair on see-through box using vanishing points

### 4.4 Finding 3D-point Pair in Space

Now, we can define two optical rays, $(\mathbf{l}_1(t), \mathbf{l}_2(s))$, from the optical center, $\mathbf{e}$, using $(\mathbf{q}_1, \mathbf{q}_2)$ as follows:

$$\mathbf{l}_1(t) = \mathbf{e} + t(\mathbf{q}_1 - \mathbf{e}), \tag{4}$$

$$\mathbf{l}_2(s) = \mathbf{e} + s(\mathbf{q}_2 - \mathbf{e}). \tag{5}$$

The 3D points, $\mathbf{p}_1 = \mathbf{l}_1(t^*) = (p_{1x}, p_{1y}, p_{1z})$ and $\mathbf{p}_2 = \mathbf{l}_2(s^*) = (p_{2x}, p_{2y}, p_{2z})$, which we want to find, must satisfy the center-plane symmetry condition. That is, $p_{1x} = p_{2x}$ & $p_{1y} = p_{2y}$ & $p_{1z} = -p_{2z}$. Thus, a solution-parameter pair, $(t^*, s^*)$, can be obtained by solving a system of three linear equations as follows (for more details, see Section 5):

$$t(q_{1x} - e_x) = s(q_{2x} - e_x), \tag{6}$$

$$t(q_{1y} - e_y) = s(q_{2y} - e_y), \tag{7}$$

$$e_z + t(q_{1z} - e_z) = -e_z - s(q_{2z} - e_z). \tag{8}$$

### 4.5 3D Parametric-Curve Creation

As the final step of converting a 2D curve to a 3D space curve, the approximation of $\{\mathbf{p}_i(x_i, y_i, z_i)\}$ to a parametric curve such as B-spline (or Bezier) curve, $\mathbf{c}(t)$, is performed by applying standard curve fitting methods [2][8][14] keeping a plane symmetry.

## 5 Solving System of Linear Equations for Finding 3D-point Pair

As given in (6)˜(8), our inverse projection of a 2D-point pair to 3D points under the plane-symmetry condition is an *over-determined* problem having *two* unknowns and *three* equations. Because of the pinhole-camera model assumption $(t, s > 0)$, the solution space is $(t^*, s^*) \in (0, \infty) \times (0, \infty)$. For most cases, $(t^*, s^*)$ can be simply determined by minimizing the sum of least-squares errors, $f = (\mathbf{NF} - \mathbf{G})^T(\mathbf{NF} - \mathbf{G})$, where $\mathbf{NF} = \mathbf{G}$ is the matrix form of (6)˜(8). However, there exist special viewing conditions in which the proposed method does not work. Thus, the analysis of these special cases will be followed with the consideration of the number of equations.

First of all, let us consider the cases that one of three equations vanishes as shown in Fig. 6. The case that only (6) vanishes, that is, $q_{1x} - e_x \approx 0$ & $q_{2x} - e_x \approx 0$, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are all on the same plane parallel to the yz-plane (see Fig. 6(a)). In the case, it is possible to calculate $(t^*, s^*)$ with remaining two equations. Similarly, if (7) vanishes, the solution exists (Fig. 6(b)). However, when (8) vanishes, that is, $q_{1z} - e_z \approx 0$ & $q_{2z} - e_z \approx 0$ & $e_z \approx 0$, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are all on the xy-plane because $p_{1x} \approx p_{2x}$ & $p_{1y} \approx p_{2y}$ & $p_{1z} \approx -p_{2z}$ or $\mathbf{p}_1 \approx \mathbf{p}_2$ & $\mathbf{q}_1 \approx \mathbf{q}_2$. Thus, there are an infinite numbers of solutions (Fig. 6(c)) (it is so called the *impossible-to-solve* case).

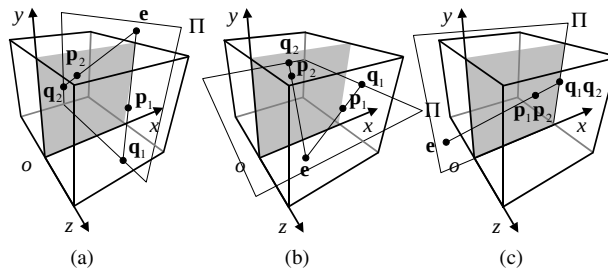


**Fig. 6.** Special cases only one equation vanishes

Now, let us consider the only one equation remains. They are all the impossible-to-solve cases (see Fig. 7). If (6) and (7) vanish, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$

are all on the line parallel to the z-axis as shown in Fig. 7(a). If only (6) or (7) remains, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are all on the same line parallel to the x-axis, or y-axis (on the xy-plane or center plane), respectively.

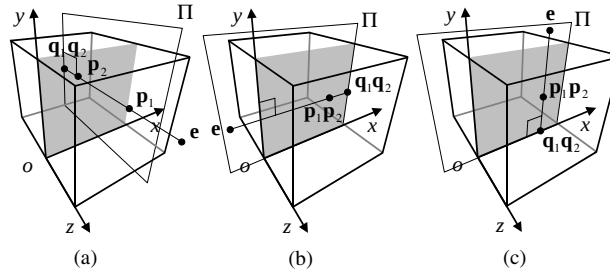


**Fig. 7.** Special cases only one equation remains

If all three of (6)˜(8) vanish, $\{\mathbf{e}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_1, \mathbf{q}_2\}$ are the coincident point (the *trivial-solution* case).

In summary, there are *seven* special viewing-cases: *one* trivial-solution case, *four* impossible-to-solve cases, and *two* possible-to-solve cases. In fact, four impossible-to-solve cases can be re-categorized as following two: 1) the optical center is positioned on the center plane, and 2) the optical rays are parallel to the z-axis.

For above possible-to-solve cases, $(t^*, s^*)$ can be explicitly calculated. For the case only (7) vanishes ($q_{1x} - e_x \neq 0$ & $q_{2x} - e_x \neq 0$ & $q_{1y} - e_y \approx 0$ & $q_{2y} - e_y \approx 0$), by substituting $t = (q_{2x} - e_x)/(q_{1x} - e_x)$ into (8), we can obtain the following equation, $s\{(q_{1z} - e_z)(q_{2x} - e_x) + (q_{1x} - e_x)(q_{2z} - e_z)\} = -2(q_{1x} - e_x)e_z$, and then $(t^*, s^*)$ except the case that the optical center is located at the infinite point where $(q_{1z} - e_z)(q_{2x} - e_x) + (q_{1x} - e_x)(q_{2z} - e_z) \approx 0$ (see Appendix A).

## 6 Implementation

A simple program for the proposed method is implemented as a *Java Applet* using Java$^{\text{TM}}$ 2 Platform Standard Edition (J2SE$^{\text{TM}}$) and Java 3D$^{\text{TM}}$ API as shown in Fig. 8. For a pen-based user interface, WACOM Intuos$^{\text{TM}}$2 tablet ($9'' \times 12''$) is used. Our Java Applet is available at http://vr.kaist.ac.kr/˜bae.

## 7 Discussions and Conclusions

In this paper, we proposed a method of constructing 3D plane-symmetric curves using an intuitive sketch interface, which was inspired from a fact that a lot of industrial products have plane-symmetric forms. The designer can arbitrary choose

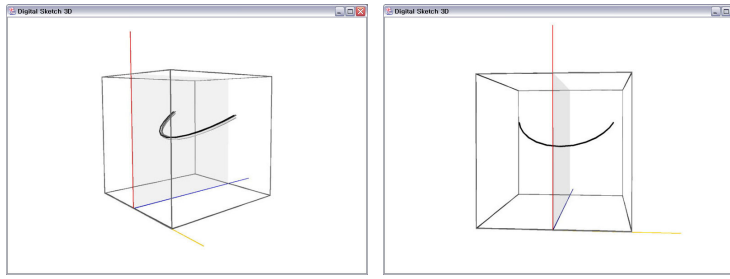

**Fig. 8.** Software implementation: repetitive 3D scribbling and settled freeform curve

a 3D perspective view, and draw a projected form of the curve intended. By repetition of the perspective 3D sketch (visual feedback and behavioral adaptation process), the designer can obtain the final form of a 3D space curve. Because of the assumption of a plane-symmetry, the proposed method does not suffer the 3D ambiguity problem unlike general inverse-projection of converting 2D image objects to 3D space objects (the exceptional viewing cases–only two–were addressed in Section 5). A prototype for the proposed method was made as simple Java Applet program, and tested by a group of product designers.

One important point is that the proposed method is relies on the designer's accurate space sense. The assumption can be thought as reasonable remembering that projective geometry, which computer graphics is based on, was started from Renaissance painters' effort to correctly reproduce the perspective effects in images of the world that they were observing. Actually, many designers took a great interest in our program, and agreed it has much possibility to be developed as a powerful tool for design specialists.

## References

1. Bae, S.-H., Kim, W.-S., Kwon, E.-S.: Digital Styling for Designers: Sketch Emulation in Computer Environment. The Proceedings of CGGM (2003)
2. Choi, B.K.: Surface Modeling for CAD/CAM. Elsevier, NY (1991)
3. Cohen, J., Markosian, L., Zeleznik, R., Hughes, J., Barzel, R.: An Interface for Sketching 3D Curves. The Proceedings of SI3DG (1999)
4. Cypher, A.: Eager: Programming Repetitive Tasks by Example. The Proceeding of CHI (1991)
5. de Figueiredo, L.H.: Adaptive Sampling of Parametric Curves. In: Paeth, A.W. (eds): Graphics Gems V. Academic Press, Boston (1995)
6. Doblin, J.: Perspective: A New System for Designers. Whitney Publications, NY (1956)
7. Eggli, L., Hsu, C.-Y., Bruderlin, B.D., Elber, G.: Inferring 3D Models from Freehand Sketches and Constraints. CAD 29(2) (1997) 101-122
8. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide 5th Edition. Academic Press, NY (2002)

9. Faugeras, O., Luong, Q.-T.: The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications. The MIT Press, Cambrigde (2001)
10. Grossman, T., Balakrishnam, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., Buxton, B.: Creating Principal 3D Curves with Digital Tape Drawing. In Proceedings of CHI (2002)
11. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: Sketch Interface for a 3D Freeform Design. The Proceedings of SIGGRAPH (1999)
12. Igarashi, T., Hugh, J.F.: A Suggestive Interface for 3D Drawing. The Proceedings of UIST (2001)
13. Lane, J. M., Riesenfeld, R.F.: A Theoretical Development for Computer Generation and Display of Piecewise Polynomial Surfaces. IEEE Transactions on PAMI 2(1) (1980) 35-46
14. Piegl, L., Tiller, W.: The NURBS Book. Springer-Verlag, NY (1995)
15. Schweikardt, E., Gross, M. D.: Digital Clay: Deriving Digital Models from Freehand Sketches. The Proceedings of CHI (2002)
16. Yamada, Y.: Clay Modeling: Techniques for Giving Three-dimensional Form to Idea. Car Styling Extra Issues 93(1/2) (1993)
17. Zeleznik, R. C., Herdon, K. P., Hughes, J. F.: SKETCH: An Interface for Sketching 3D Scenes. The Proceedings of SIGGRAPH (1996)

## Appendix A: Optical Center at Infinite Point

Let us consider the equality, $(q_{1z}-e_z)(q_{2x}-e_x)+(q_{1x}-e_x)(q_{2z}-e_z)\approx 0$, when only one equation, (7), vanishes as shown in Fig. 9(a). From the pinhole-camera assumption $(0<t<\infty)$, $(q_{1x}-e_x)(q_{2x}-e_x)>0$. Thus, $(q_{1z}-e_z)(q_{2z}-e_z)<0$, and there are two cases satisfying it: $q_{2z}<e_z<q_{1z}$ *or* $q_{1z}<e_z<q_{2z}$. Without the loss of generality, we can choose the case of $q_{2z}<e_z<q_{1z}$ (see Fig. 9(b)). The modified form of the given equality can be written as $(q_{1z}-e_z)/(q_{1x}-e_z)\approx(e_z-q_{2z})/(q_{2x}-e_x)$, and it means two triangles, $(\triangle\mathbf{e}^*\mathbf{q}_1^*\mathbf{r}_1^*, \triangle\mathbf{e}^*\mathbf{q}_2^*\mathbf{r}_2^*)$, must be have almost *foldaway* forms. Because the case now treated is one of the possible-to-solve cases, $\mathbf{e}$ can not be positioned on the center plane or $e_z\neq 0$. Therefore, $e_z\rightarrow\pm\infty$ or $(t^*,s^*)=(\infty,\infty)$.

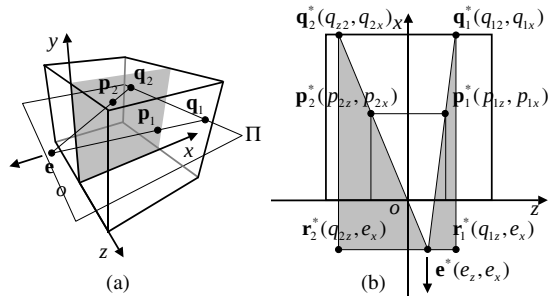


**Fig. 9.** Optical center at infinite point