Intuitive Path-based Camera Control for Dynamic Scenes

Aritoki Kawai[*†] Tsuneya Kurihara[*‡] Tomoyuki Nishita[*]

[*]The University of Tokyo [†]Sony Computer Entertainment Inc. [‡]Central Research Laboratory, Hitachi, Ltd.

Abstract

This paper presents an intuitive user interface for a virtual camera controlled by a path and a set of constraints. Here, the constraints mean where an object is to be projected on the screen. In many cases, the camera position plays a more important role than other settings such as the orientation and the field of view angle of the camera. Therefore, in this paper, the user sets the camera path and the screen constraints, and then, both the orientation and the field of view angle are computed automatically. Thanks to the low computational cost in the presented method, real time control is possible. We generate several sample animations to confirm the effectiveness of this method.

1 Introduction

Positioning and controlling a virtual camera is important in 3D computer graphics. We cannot create a desired image without an appropriate camera setting. Generally, it is hard to understand how to control a camera for non-expert users and it may be difficult to get the desired images and movies in mind. This is caused by the camera control scheme that requires the user to directly manipulate the camera parameters relating with the camera position or orientation in order to control the camera. In the camera control scheme in which the camera parameters are directly manipulated, it is required to recognize the position and orientation of the camera, and the position of the objects that focused on in 3D space. It is also difficult to predict the resulting image accurately before generating it. We propose a camera control interface by which even novice users can control the camera readily and intuitively by setting camera path and constraints on the screen.

This paper is composed as follows. In Section 2, the research about camera control in computer graphics is described. In Section 3, the overview of the camera system proposed in this paper is described. In Section 4, first the method to compute the camera parameters by a set of constraints on the screen is explained. Next the algorithm to deal with camera position independently and the actual interface using the approach is described. In Section 5, the effectiveness of the proposed approach is validated through the resulting animation. Last, conclusion of this research is described and future work is referred to in Section 6.

2 Related Work

The purpose of a camera control is to assist the users to set the camera parameters. Early works in computer graphics use a key frame method, where the users set camera parameters at a set of points that compose the camera trajectory; *i.e.* key frames and between key frames each camera parameter is computed by interpolation [5]. However, computed paths are generally not realistic and far from the users' desired images. This is caused by the feature of the key frame method, in which the object focused on at that time cannot be kept being projected to the desired position on the screen during the interpolation. The control to keep a camera turning to a direction of a certain object is possible, however, the object is only projected to the center of the screen. Furthermore, we cannot control how to view more than two objects at all.

Various techniques have been proposed in order to solve this problem. Blinn proposed an approach by which the projected position of two objects on the screen can be controlled [4]. Because this method is based on vector algebra, it can be calculated efficiently. However, only two points can be controlled.

In order to solve the problem mentioned above, Gleicher and Witkin [1] proposed a technique called *through-the-lens camera control* (*TTL*) to control a camera intuitively by directly controlling two-dimensional points on the screen on which three-dimensional points are projected instead of setting parameters of the camera directly. This approach offers a user-friendly interface to control a camera by manipulating points on the screen. This approach needs low computational cost, thus this is suitable for the case where the calculation must be done in real time. However, it is difficult to predict behavior of the camera, especially the camera position in three-dimensional space, due to controlling the camera only by constraints on the screen. Furthermore, this method cannot deal with an occlusion

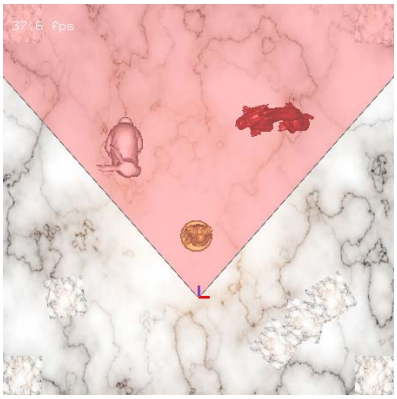


Figure 1: The window for viewing the 3D scene

Figure 2: The window for the view from the camera

problem, because of regarding an object as one point.

In addition, many approaches have been proposed regarding a camera control as a constraint satisfaction problem [7, 8, 9]. This framework offers a declarative tool for the camera planning to users. For example, a user sets a condition that an object A is projected to the left side of the screen and an object B is projected to the right side of the screen during a certain duration time, then a system solves these inputs as a constraint satisfaction problem. A common problem of these techniques is high computational cost. Classical constraint programming techniques such as propagation cannot directly handle these problems within a reasonable computational cost. Bares *et al.* propose a heuristic-based complete search algorithm [6]. Jardillier and Languénou propose a method to set a movement of a camera by an intellectual method [9]. However, only simple camera work can be generated by these methods. Christie *et al.* propose an approach that limits the camera movement to basic one which is established and used in cinematography, and reduce the computational cost compared with the previous works [2]. However, a computational cost remains high, and it needs much time for preprocessing. In addition, computation time increases as camera work becomes complicated.

Although the camera parameters can be computed to satisfy constraints on the screen by using the technique of Gleicher and Witkin [1], it is difficult to predict the behavior of the camera in world coordinate system. In many cases, however, the camera position is wanted to move along the path the user specifies. For example, in the case of marathon broadcast, the field of view angle and the orientation of the camera have no constraints. On the other hand, the position of the camera is always the position of the broadcast van because the camera is fixed on it. To realize the camera work in such cases, it is desirable that the field of view angle and the orientation of the camera is automatically computed to satisfy constraints on the screen, however, the position of the camera is independently controlled in order to move along the path that the user specifies.

3 Overview

In the camera system used in this paper, a camera has seven degrees of freedom as parameters; *i.e.* one is field of view angle, three is position in 3D space and residual three is rotations. Because all of the position and the orientation of the camera in the 3D space and the scene viewed through the camera had better be able to be viewed concurrently for usability, thus two windows are used in the system. One is the window through which whole 3D scene, where all objects exist, can be viewed. In this window, we set the camera path (see Figure 1). The camera frustum volume can also be viewed in this window. The other is the window through which we view the 2D projected scene (Figure 2). In Figure 2, some control points (small squares) used as constraints can also be viewed. We manipulate these control points on the screen to control the camera, not directly set camera parameters.

Steps of a camera control procedure are as follows (Figure 3).

1. Camera path setting

Set a camera path using spline curve or sketch interface [3]. In Figure 3, the camera path is drawn with a bold line.

2. Segmentation of the camera path

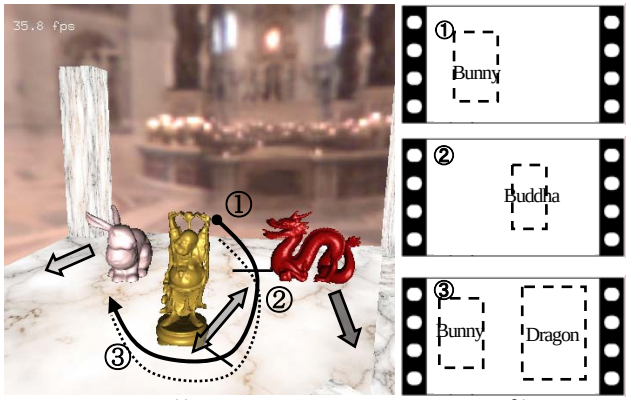
Subdivide the camera path to some segments. In each segment, different screen-space constraints are used. In Figure 3, the camera path is subdivided into three

segments.

3. Setting screen-space constraints in each segment

Specify the projected positions of objects in each segment, through manipulation of control points on the screen. For example, the control points are set in such a way that the Stanford Bunny is projected to the left side of the screen at the segment (1) (Figure 3 right top).

For making an animation, both the field of view angle and the orientation of the camera are calculated automatically in order to satisfy the set of constraints, while the camera moves along the path in each segment.



(a) (b)

Figure 3: An input example:
(a) An example of camera path setting
(b) An example of setting constraints on the screen

4 Algorithm

To realize the proposed system, we apply the TTL scheme [1] and extend it in order to control the camera position independently from constraints on the screen.

4.1 Camera Control on the Screen

Let $\mathbf{x}$ be the world-space point that projects to image-space point $\mathbf{p}$, and $\mathbf{V}(\mathbf{q})$ be the function of the camera parameter $\mathbf{q}$, $\mathbf{h}$ be a function that converts homogeneous coordinates into 2D image coordinates, defined by

$$\mathbf{h}(\mathbf{x}) = \left[\frac{x_1}{x_4}, \frac{x_2}{x_4}\right], \tag{1}$$

where x_i is the component of homogeneous point $\mathbf{x}$. Thus, we obtain the following equation:

$$\mathbf{p} = \mathbf{h}(\mathbf{V}(\mathbf{q})\mathbf{x}). \tag{2}$$

Assuming that the world-space point $\mathbf{x}$ is fixed, the image point $\mathbf{p}$ is a function of the camera parameters $\mathbf{q}$. Because $\mathbf{h}$ and generally $\mathbf{V}(\mathbf{q})$ are nonlinear, Equation (2) is nonlinear. We consider indirectly solving this nonlinear equation. By applying the chain rule, we obtain the expression for the image velocity $\dot{\mathbf{p}}$:

$$\dot{\mathbf{p}} = \mathbf{h}'(\mathbf{V}\mathbf{x})\left(\frac{\partial(\mathbf{V}\mathbf{x})}{\partial \mathbf{q}}\right)\dot{\mathbf{q}}, \tag{3}$$

where $\mathbf{h}'(\mathbf{x})$ is the matrix representing the derivative of $\mathbf{h}(\mathbf{x})$, given by

$$\mathbf{h}'(\mathbf{x}) = \begin{bmatrix} \frac{1}{x_4} & 0 \\ 0 & \frac{1}{x_4} \\ 0 & 0 \\ -\frac{x_1}{x_4^2} & -\frac{x_2}{x_4^2} \end{bmatrix}. \tag{4}$$

Here, $\dot{\mathbf{q}}$ is the time derivative of $\mathbf{q}$, and $\partial(\mathbf{V}\mathbf{x})/\partial\mathbf{q}$ is the matrix representing the derivative of the transformed point $\mathbf{V}\mathbf{x}$ with respect to $\mathbf{q}$.

Here, we define the matrix

$$\mathbf{J} = \mathbf{h}'(\mathbf{V}\mathbf{x})\left(\frac{\partial(\mathbf{V}\mathbf{x})}{\partial \mathbf{q}}\right), \tag{5}$$

so that

$$\dot{\mathbf{p}} = \mathbf{J}\dot{\mathbf{q}}. \tag{6}$$

We obtain Equation (6) giving $\dot{\mathbf{p}}$ as a linear function of $\dot{\mathbf{q}}$, even though $\mathbf{p}$ is a nonlinear function of $\mathbf{q}$.

Even if $\dot{\mathbf{p}}$ is given, however, Equation (6) cannot be solved simply. This means that the camera parameters cannot be determined by using only an image-space point velocity. Here we set a criterion $\dot{\mathbf{q}}_0$, and then $\dot{\mathbf{q}}$ which minimizes the difference of $\dot{\mathbf{q}}_0$ is selected. For example, $\dot{\mathbf{q}}_0 = 0$ means that camera parameters which change minimally are selected. This is represented as the following equation,

$$\text{minimize} \quad E = \frac{(\dot{\mathbf{q}} - \dot{\mathbf{q}}_0)(\dot{\mathbf{q}} - \dot{\mathbf{q}}_0)}{2}. \tag{7}$$

We have described a method to control one point on the screen. Next, we refer to controlling multiple points. Because $\mathbf{J}$ is determined by $\mathbf{x}$, Equation (6) is prepared for each point to control. By solving the simultaneous equation, we can control multiple points.

When world-space point $\mathbf{x}$ itself moves by the time, image-space point $\mathbf{p}$ also changes. Therefore we can control the moving point by solving Equation (8) where an additional term is added to Equation (3).

$$\dot{\mathbf{p}} = \mathbf{h}'(\mathbf{V}\mathbf{x})\left(\frac{\partial(\mathbf{V}\mathbf{x})}{\partial \mathbf{q}}\right)\dot{\mathbf{q}} + \mathbf{h}'(\mathbf{V}\mathbf{x})\mathbf{V}\dot{\mathbf{x}}. \tag{8}$$

4.2 Independent Camera Position Control

Now we will extend the original TTL scheme [1] to be able to deal with the camera position independently. Equation (6) means that the velocity of the image-space point is determined by the linear sum of a set of differential of camera parameters. The camera has seven degrees of freedom; position, orientation and field of view angle. The camera position of the next time is known because a user sets the camera path, which means that differential of the camera position is also known.

The constraints on the screen are automatically adjusted with the precomputed effect of differential of the camera position. The residual four degrees of the freedom of the camera parameters are computed by the method in Section 4.1 based on the new constraints that incorporates the effect of the differential of the camera position, which enables to manipulate the camera position independently without interrupting satisfying constraints on the orientation and field of view angle of the camera on the screen.

4.3 Camera Path Setting

We can set the camera path by manipulating control points of a curved line or using sketch interface.

Using control points setting method, we adopt the Catmull-Rom Spline curve [10] because it interpolates the control points therefore the path shape is predictable intuitively.

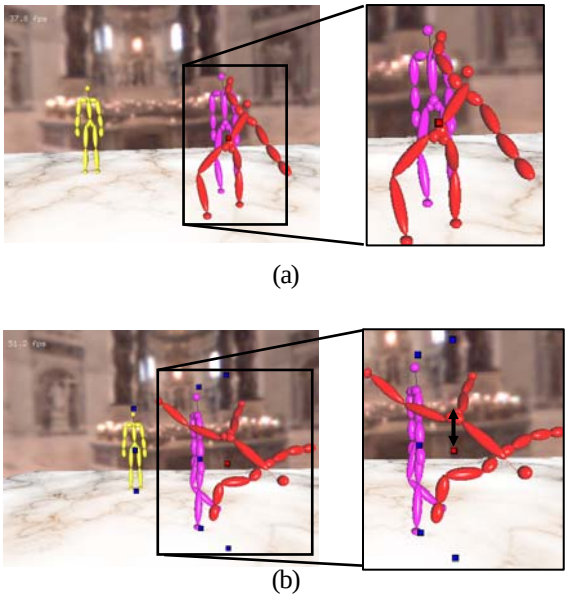
As the sketch interface, we adopt the approach [3] in which we control not only the curve itself but the shadow projected on the ground perpendicularly in order to specify the curve shape clearly.

The reason why we set the camera path interactively is that setting a camera path roughly is easy and the occlusion problem that many camera systems suffer from can be avoided thanks to the merit that we set the path while looking at the scene.

4.4 Control Points Setting

Control points given as constraints are set on the local coordinate of the objects. By picking a displayed vertex of each object, we set it as a control point. Moreover, utilizing the information of the bounding box, an arbitrary position of each object on the local coordinate can be set as a control point. Control points shown in Figure 2 are set by the latter method.

When we set a control point on multiple-joint structure such as a human body, additional care must be taken. For example, there is a problem that a camera might shake in correspondence with the vertical motion of the waist up and down if we fix a control point on the route (the waist) of a character (Figure 4). For solving this problem, an option is prepared in which the height of control points can be fixed.



(a)

(b)

Figure 4: An example of the dynamically transformed objects
(a)The root position of the front character is represented as square.
(b)While the character turns catherine wheels, even the root which changes the least in height changes the position to the top of the arrows end. Thus if fix the height of the root position on the screen, the camera cannot avoid shaking

4.5 User Interface

We explain the user interface of the camera system with Figure 3. At first we set a camera path (Figure 3 (a)). Next we manipulate a control point on the screen so that Bunny is projected in segment (1) to the left side of the screen (Figure 3 (b) top). The other control points are set likewise. These control points are fixed at the positions on the screen in each segment. Even if each object (Buddha, Bunny and Dragon) moves along the arrow, the constraints have been satisfied. During this operation manipulating constraints on the screen, the system warns the user if the field of view angle of the camera becomes wider than that used by actual cinematographers.

At the boundary of two segments, we have two options to deal with the camera parameter. Each camera parameter can be interpolated in arbitrary time. In addition, it is possible

that the camera parameters change immediately at the boundary.

5 Results

We have generated animations to show the effectiveness of the proposed camera control interface. Figures 5 and 6 are the captured images of the animation. Each object and a camera move on the paths that are specified by the user (top to bottom, from 1st to 7th steps). In Figure 5, we specify the camera parameter so that the whole body of Buddha is projected to the right side of the screen until the camera overtakes the Buddha (1st and 2nd steps), and after the overtaking, the upper part of the body of Buddha is projected to the center of the screen (3rd to 7th steps). Figure 6 shows the following example: The camera moves on a specified path, focusing on the screen so that the whole body of the Bunny is projected to the left side of the screen in the first segment (1st to 4th steps), and in the last segment, Bunny is set to be projected to the left side of the screen and Dragon is set to be projected to the right side of the screen (5th to 7th steps). In these examples, three control points are specified on each object by using the bounding-box information. In these examples, three control points are specified on each object; high, middle and low height. These examples show that with few operations we can set a path, orientation and field of view angle of a camera easily.

6 Conclusion and Future Work

In this paper, we have proposed an intuitive camera control interface to automatically compute the orientation and field of view angle of the camera satisfying the constraints. The constraints are set by the position of each object by manipulating control points on the screen. Then, the camera moves along a user-determined path and the orientation and the field of view angle are computed automatically. We showed a method to deal with the camera position separately from the constraints about the orientation and the field of view angle given by manipulating control points on the screen. We showed a method to deal with the camera position separately from the constraints about the orientation and the field of view angle given by manipulating control points on the screen. Because the proposed method uses the differential calculation in order to solve the constraints, the process needs low computational cost thus can be carried out in real time. Therefore, the camera setting process in the proposed system can be done intuitively and interactively.

In the presented approach, however, there remain many interactions the user should do, such as camera path setting or manipulation of control points. As future works, we consider development of a method to calculate a detailed camera path from roughly sketched one automatically. In addition, we consider development of a method to generate appropriate camera work by giving a size and projected position of each object so that a user does not need to control a control point directly as a set method of constraints on the screen.

References

[1] Gleicher M. and Witkin A., Through-the-Lens Camera Control, In *Proc. of SIGGRAPH'92*, pp. 331-340, 1992.

[2] Christie M., Languénou E. and Granvilliers L., Modeling Camera Control with Constrained Hypertubes, In *Proc. of CP 2002*, pp. 618-632, 2002.

[3] Cohen J., Markosian L., Zeleznik R., Hughes J. and Barzel R., An interface for sketching 3D curves, *ACM Symposium on Interactive 3D Graphics*, pp. 17-21, 1999.

[4] Blinn J, Where am I ? What am I looking at ?, IEEE Computer Graphics & Applications, pp. 76-81, 1988.

[5] Foley J., Dam A., Feiner S., and Hughes J., Computer Graphics: Principles and Practice, Addison-Wesley Publishing Co., 1990.

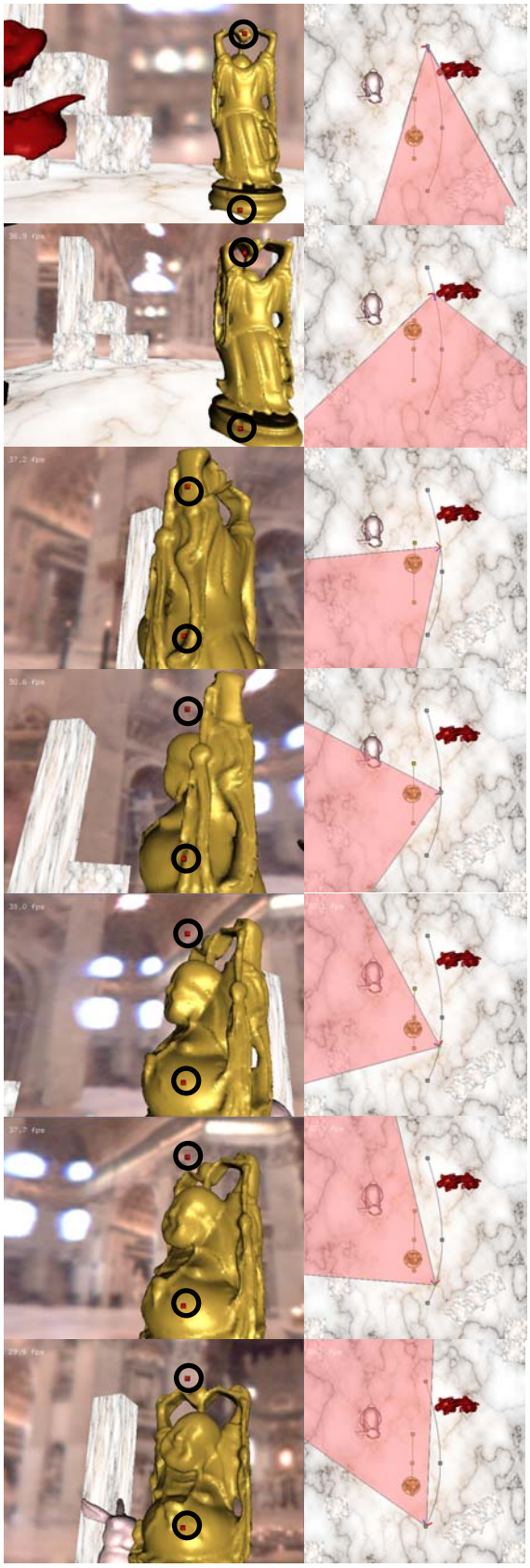
[6] Bares W., Gregoire J., Lester J., Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds, In *Proc. of AAAI-98/IAAI-98*, pp. 1101-1106, 1998.

[7] Halper N., Helbing R., Strothotte T., A Camera Engine for Computer Games, Managing theTrade-Off Between Constraint Satisfaction and Frame Coherence, In *Proc. of the Eurographics'2001 Conference*, pp. 174-183, 2001.

[8] Bares W., Mcdermott S., Boudreaux C., Thainimit S., Virtual 3D Camera Composition from Frame Constraints, In *Proc. of the eighth ACM international conference on Multimedia*, pp. 177-186, 2000.

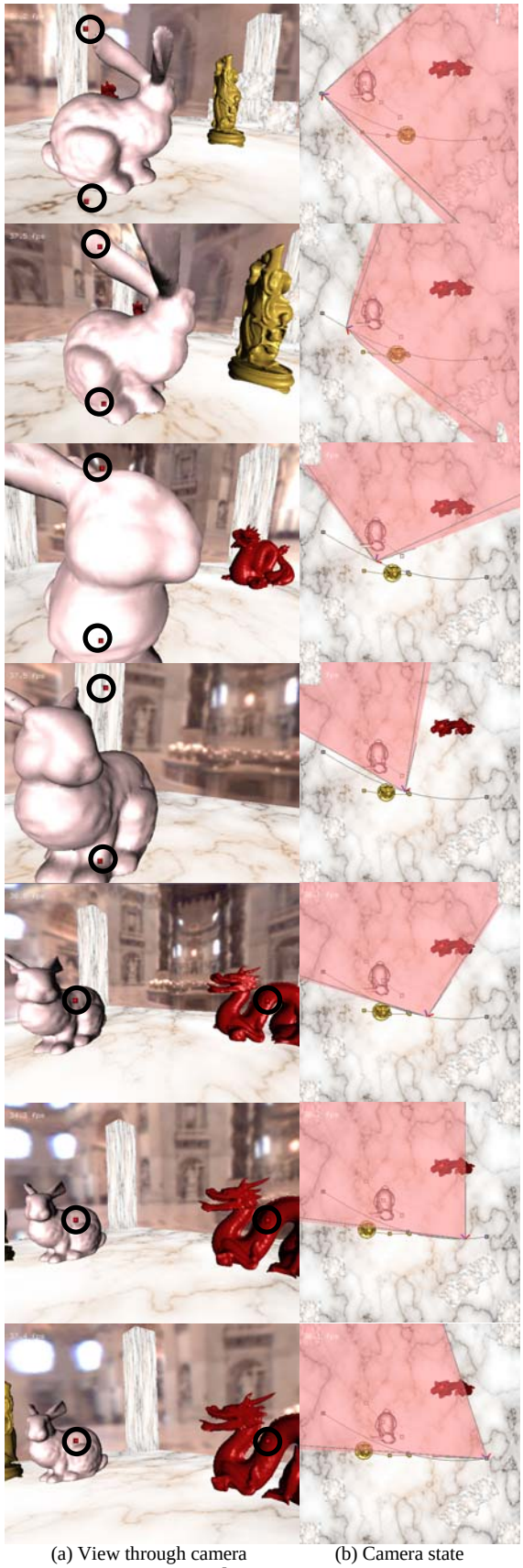
[9] Jardillier F., Languénou E., Screen-Space Constraints for Camera Movements: the Virtual Cameraman, In *Proc. of EURO-GRAPHICS-98*, pp. 175-186, 1998.

[10] Catmull E., Rom R., A Class of Local Interpolating Splines, in R. E. Barnhill, R. F. Riesenfeld (ed.), Computer Aided Geometric Design, Academic Press, NewYork, pp.317-326, 1974.



(a) View through camera (b) Camera state

Figure 5: Resulting images 1



(a) View through camera (b) Camera state

Figure 6: Resulting images 2

Each point in circle means constraint. Each point is on the top, middle and bottom of the central axis of each object.
In Figure 5, constraint setting is changed at third step. In Figure 6, that is change at 5th step.
By no changes of the circle positions, resulting images show that constraints are satisfied.