ADAPTIVE CAMERA MOTION GENERATION FOR PROCEDURAL GUIDANCE AND UNDERSTANDING IN VIRTUAL ENVIRONMENTS

Shan Li, Yuzhong Shen
Department of Computational Modeling & Simulation Engineering, Old Dominion University
Norfolk, VA
sliu004@odu.edu, yshen@odu.edu

ABSTRACT

Virtual environments provide an effective means of training for many applications. Performing procedures that consist of several steps is common in these applications. This paper presents an innovative adaptive camera motion generation approach through combining control point selection, path generation, and camera control to explore the details of adjacent steps in a procedure and provide procedural guidance and understanding in virtual environments. In this approach, two methods are proposed to automatically create control points of the camera motion path and target path. The first method uses the components of the object of interest involved in adjacent steps in a procedure. The element at each step is set as the target. Each control point's coordinate value is calculated according to the target's position and size corresponding to this control point and the position change trend of adjacent objects after this target. The second method generates the control points based on the object's bounding box, where the control point location is set to a default value and further adjustable by the user. Then, the 3D natural cubic spline interpolation algorithm generates camera motion paths and corresponding target paths. The camera aims at the target path when it moves along its route to eliminate the camera instability problem and adapts its projection frustum to fit models of different sizes. Experimental results illustrate that the proposed approach can automatically generate camera motion to provide the users with a smooth and informed view of the system and effortlessly guide them to the next step in the procedure.

ABOUT THE AUTHORS

Shan Liu is a Ph.D. student of the Computational Modeling & Simulation Engineering (CMSE) at Old Dominion University. Her research interests lie in Virtual Reality and Augmented Reality, especially in the digital instruction manual.

YUZHONG Shen received his B.S. degree in Electrical Engineering from Fudan University, Shanghai, China, M.S. degree in Computer Engineering from Mississippi State University, Starkville, Mississippi, and Ph.D. degree in Electrical Engineering from the University of Delaware, Newark, Delaware. His research interests include virtual reality, augmented reality, visualization and computer graphics, transportation modeling and simulation, general modeling and simulation and signal and image processing.

Dr. Shen is Professor and Chair of the Department of Computational Modeling and Simulation Engineering (CMSE) at Old Dominion University. He has a joint appointment with the Department of Electrical and Computer Engineering of Old Dominion University. Prior to joining Old Dominion University, Dr. Shen worked as an Engineer and a Senior Engineer with Weifang Hua-Guang Technologies, China, as a Research Assistant with National Science Foundation Engineering Research Center for Computational Field Simulation at Mississippi State University, as a Research Assistant with the Department of Electrical and Computer Engineering at the University of Delaware, and as a Senior Research Scientist with Virginia Modeling, Analysis, and Simulation Center (VMASC) at Old Dominion University. Dr. Shen is a Senior Member of IEEE.

ADAPTIVE CAMERA MOTION GENERATION FOR PROCEDURAL GUIDANCE AND UNDERSTANDING IN VIRTUAL ENVIRONMENTS

Shan Li, Yuzhong Shen
Department of Computational Modeling & Simulation Engineering, Old Dominion University
Norfolk, VA
sliu004@odu.edu, yshen@odu.edu

INTRODUCTION

Over the last decades, through theoretical and experimental studies, many researchers have found that virtual environments (including Virtual Reality and Augmented Reality) promote an extensive range of fields, from education to manufacturing, by providing users an interactive, immersive, and explorable experience (Bracq et al. 2019; Loch et al. 2019). Many professions in various fields require critical skills that must be learned and enhanced via frequent training, which can be hazardous and costly, such as fire rescue and aviation. Computer simulations using virtual environments have been developed to offer controlled and safe training scenarios as an alternative and complement to the actual training with improved learning outcomes. Trainees can practice, make mistakes, and enhance situational awareness in simulated environments without endangering themselves. Many professions have widely adopted simulation-based training, such as healthcare, defense, education, aviation, and other applications. The virtual environments in simulation-based training play a critical role in the effectiveness of simulation and training. A probably and carefully designed and created virtual environment can significantly improve learning outcomes.

Virtual environments can be explored in different ways. One standard view mode provided by 3D modeling software tools is called the Free view mode, in which the user can pan, zoom, or orbit around the object or part of interest. The Free view mode provides users the essential capability to understand the environment or object but can be burdensome to attain ideal views for certain parts, especially those in complex environments. Most 3D modeling software tools provide the Focus view mode that is usually activated by pressing the F key to overcome this issue. In the Focus view mode, the part of interest selected either by name or a mouse click is automatically zoomed and moved to the center of the screen to provide an improved but not optimized view. Users usually need to perform more user interactions, e.g., orbit, to obtain an optimal view of the part of interest. Both Free view mode and Focus view only provide static views of the virtual environment and no animations.

One critical task that frequently occurs in these fields is understanding or learning procedures, such as assembling a machine or troubleshooting a fault or failure. As standard in product manuals, the procedures are usually decomposed into several steps or tasks, sometimes accompanied by graphical illustrations. Traditional printed product manuals are difficult to follow, and very often, users are frustrated by the lack of clarity in the manual, e.g., the part of interest cannot be found, or the location of the next step is not precise. Many issues in traditional product manuals are caused by the fact that graphical illustrations are disconnected, static rendering of parts of the system and thus fails to provide an overall view effectively. For example, after finishing a step of a procedure, the user is often at a loss as to what to do next because the location for the next step cannot be found or the part for the next step cannot be found. The user has to resort to the overall system illustration if there is one. Obscure or misleading product manuals can cause erroneous operations, resulting in various damages and losses.

On the other hand, virtual environments can effectively address the issues in traditional procedural manuals or instructions by introducing smooth transitions (or animations) between adjacent steps or tasks in the procedure. The animations between adjacent steps in a procedure provide an overall view of the system and seamlessly guide the user to the next step. In general, computer animation means the change of images on a computer screen via various means, e.g., object movement, camera motion, lighting, or color changes. The component or the part of interest can be highlighted using multiple approaches to facilitate understanding or learning a process or procedure. For example, the color of the component of interest can be tentatively changed to red, or the size of the component can be tentatively enlarged for easy identification of the component.

Meanwhile, camera motion is an excellent means to implement the transition between adjacent steps in a procedure (Luo, Li, and Mo 2017). A virtual camera is used in a virtual environment to generate a 2D image of the virtual environment to be displayed on the computer screen, just like an actual physical camera that captures the natural world and produces a 2D image. The Free view mode and Focus view mode discussed above are also generated using a virtual camera. When the virtual camera moves in the virtual environment, it animates the virtual environment, just like the actual cameras used to make films. Camera motion, camera angles, and zoom play a critical role in cinematography and directly affect the film's overall success.

There are two ways to generate camera motions for illustrating adjacent steps in a procedure. The first approach manually generates the camera motion by determining the camera path, target path, orientation, and zoom. While this approach can work as is done in filmmaking and game development, it is laborious and requires manual work for each new procedure and object. A better approach would be the automatic generation of the camera motion based on objects of interest in adjacent steps. In this approach, camera positions, target positions, and camera orientations are automatically computed as a function of time to provide a smooth and informed view of the system and guide the user to the next step effortlessly. This paper proposes two control point selection methods for adaptive camera motion generation for procedural guidance and understanding in virtual environments. The first approach utilizes specific or designated components in the virtual environment to achieve the control points, while the second approach generates the control points based on the object's bounding box. The natural cubic spline is then utilized to generate the camera path from the control points and compared with the Bezier curve. A corresponding target path that consists of points at which the camera points is then generated accordingly to produce smooth views of the system when the camera is in motion.

RELATED WORK

There has been much interest in applying virtual environments to learning and training (Dwivedi et al. 2018; Loch et al. 2019; Bracq et al. 2019). Dwivedi et al. (2018) focused on creating well-presented scenes for users to understand tasks better and optimize their operations. Loch et al. (2019) proposed an adaptive virtual training system that provides initial before real-time adaptations during training and includes several procedure instructions in the form of static illustrative pictures and texts. Bracq et al. (2019) designed a virtual simulator for surgical procedures teaching. This virtual simulator uses several instruction videos to explain basic concepts and training contents. Most of these existing virtual systems use static images and text or pre-recorded videos to help learners or trainers. However, this is not conducive to understanding the entire procedure effectively. Automatic roaming based on specific procedures would be the right choice for procedural guidance in virtual environments.

The general method for automatic camera roaming includes three steps: obtaining the path control points using an interactive method or algorithms, determining roaming paths by connecting the control points using interpolation algorithms, and controlling the camera move along the paths to explore the virtual model (Luo, Li, and Mo 2017). Many approaches have been proposed in the literature to find the best viewpoints for cameras. Rapidly-exploring random trees (RRTs) have successfully generated collision-free waypoints (Yang and Sukkarieh 2008). The standard RRT algorithm produces a time-parameterized set of control inputs to move from the initial to the goal state. Ji and Shen (2006) presented a method that combines static view selection with dynamic programming to select time-varying viewpoints and produce a smooth animation (Ji and Shen 2006). However, few viewpoint selection methods took into account the subsequent curve generation. Sokolov, Plemenos, and Tamine (2006) presented a two-step method that first determines a minimum set of good viewpoints and then calculates a path around the model based on these viewpoints (Sokolov, Plemenos, and Tamine 2006).

Since the camera cannot follow a path with some sharp turns or angles, it is necessary to smooth the path. Several methods have been proposed to generate a smooth camera path (Hsu, Zhang, and Ma 2013; Luo, Li, and Mo 2017; Li et al. 2018). Some interpolation curves, including the natural cubic splines (Bartels, Beatty, and Barsky 1987), Bezier curves, Hermite splines (Hajji et al. 2018), and Cardinal splines (Luo, Li, and Mo 2017), are good choices for generating paths with continuous curvature. The Hermite splines need parameter values and derivatives associated with each control point and give a first-order parameter continuity (C1). It is challenging to choose the derivatives at endpoints so that patches line up nicely in a visually pleasing way. The Cardinal splines can’t solve this problem because they still need to worry about the first derivative. Moreover, the Cardinal splines have no control of derivatives at endpoints. These problems have, quite literally, hampered their range of application. The Bezier curve with C1

continuity has been maturely developed and used in computer graphics and related fields. Bezier curves are smooth, but it is difficult to set control points for the Bezier curve in some cases. It is because although the first and last control points among a set of control points defining a Bezier curve are always the endpoints of the curve, the intermediate control points generally do not lie on the curve. However, it is easy to set control points for the natural cubic splines with second-order parameter continuity (C2) because the curves are constructed of piecewise third-order polynomials which pass through control points. Nevertheless, the natural cubic splines are visually not smooth enough compared with the Bezier curve, especially when the control points form sharp curves. Therefore, how to generate a smooth curve is still an open problem.

Camera control is a fundamental problem in computer graphics and related fields. Many methods have been proposed to meet specific needs in different areas, such as robotics, character animations, and computer games (Hsu, Zhang, and Ma 2013). Blinn (1988) presented his pioneer work in this area (Blinn 1988), which developed a framework for configuring a camera so that two objects that traveled along different curved paths appeared on screen at given coordinates, and derived the camera position and orientation in the space from the desired image. Drucker and Zeltzer (Drucker and Zeltzer 1994) proposed a camera motion algorithm based on the cinematography's shot concept. Several researchers applied various constraints, such as geometric collisions and visual occlusions, to produce optimal camera motion. Stoev and Straßer (2002) presented a criterion to improve the scene's depth impression and maximize the projected area. Afterward, they introduced pseudo-events to take events from a future point into account to smooth the camera motion (Stoev and Straßer 2002). Hsu, Zhang, and Ma (2013) proposed a multi-criteria algorithm coupled with a force-directed routing algorithm to enable rapid camera paths (Hsu, Zhang, and Ma 2013). However, these methods are designed for a particular data set with normal distribution or need interactive tuning to select interest points. This paper proposes a simple and effective method by combining control point selection, path generation, and camera orientation to provide procedural guidance and understanding in virtual environments.

THE PROPOSED PROCEDURE GUIDANCE METHOD

Computer graphics and visual simulations use a virtual camera to generate 2D views (or images) of 3D virtual environments. Therefore, the virtual camera is the most fundamental and critical element in visual simulations. The virtual camera is used to mimic an actual camera in the real world, and as such, it possesses many properties of a real camera. As illustrated in Figure 1, a virtual camera is determined by its position, target position, camera orientation (Up direction), and viewing frustum. The first three parameters correspond to positioning and orienting a real camera, while the viewing frustum corresponds to the zoom of a real camera. Camera motion designs center around determining the optimal values of camera parameters for various purposes, such as long shots, full shots, and closeups. This section describes the proposed adaptive camera motion generation approach for procedural guidance and understanding in virtual environments.

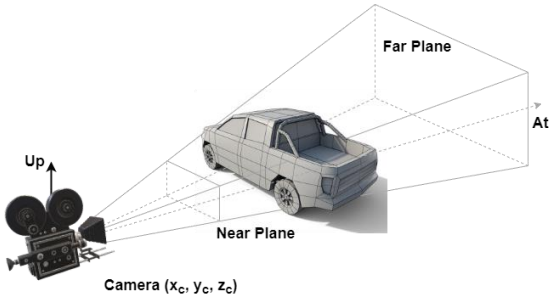

Figure 1. A camera located at (x_c, y_c, z_c) in world space looks at the vehicle model and captures part of the vehicle model. Moreover, the perspective viewing volume is specified.

Selection of Control Points for Camera Path

The purpose of camera motion for procedural guidance is to transition the camera from one step of a procedure to the next step. This paper assumes that each procedure step only involves one system component portrayed by the procedure. Therefore, for two adjacent steps of the procedure, the camera must transition from the current step (the starting component) to the next step (the ending component). The most straightforward camera path would be a linear motion connecting the starting and ending elements along a line segment. However, the simple approach has several problems. If the starting and ending components are located at opposite sides of the system, the linear camera path will pass through the system's internal parts invisible to the users. It would not help the user understand the relative positions of the two components. In addition, even the two components are on the same side of the system. Still, they

may have different viewing directions, e.g., perpendicular viewing directions, and a simple linear camera path would either produce abysmal results or not work at all. Therefore, a better approach is to use parametric polynomial curves to generate a camera path that helps users understand component positions and the overall structure.

The control point is one of the essential parameters of the polynomial curves, which will determine the smoothness and other characteristics of the curve. It makes control point selection a crucial issue for adaptive camera motion. This paper proposes two methods to generate control points under different known conditions. The first method computes the control points based on the locations of components in adjacent steps in a procedure. The second method determines the control points based on the bounding box of the entire system of interest. The first step of each method is to compute the bounding box of the virtual model, which is also essential for adaptively displaying most models with different sizes.

Component-based Selection

The corresponding camera target path also needs to be created except for the camera motion path. The camera target path needs to pass through these model components. The first and third target points are the centers of the adjacent step's model components, respectively. And then, this method calculates the second target point based on these two target points. For simplicity, the second target point is first set as the middle point between the first target point and the third target point and then adjusted on an individual axis among the X-axis, Y-axis, and Z-axis based on the desired path. For example, if the adjacent steps' components are the rear of a vehicle model and its front, then the first target is the rear of the vehicle model, and the third target is its front part. The initial setting of the second target point is in the middle of the vehicle model. If the desired path passes through the roof, adjust the ordinate value of the second target point. After three target points are set, the method automatically produces control points based on the corresponding target points, the model components' locations, sizes, and variation trends in the specific procedures.

Figure 2 shows the algorithm of calculating its value for each axis in one control point coordinate, where p_i represents the coordinate of the i^{th} control point along an axis, $i = \{1, 2, 3\}$, $p = \{x, y, z\}$, p_{oi} is the corresponding axis coordinate of the i^{th} target point that is the center of the i^{th} model component, s_{pi} is the size of the i^{th} component along the corresponding axis, τ_{pi} is an adjustable parameter of the i^{th} control point. Each control point's coordinate value is calculated according to the target's position and size corresponding to this control point. The position-changing trends of other targets correspond to this target. Let us take the x-axis value of the first control point as an example. Suppose the x-axis coordinates of the second and third target points are getting larger and larger relative to that of the first target point. In that case, the x-axis coordinate of the first control point should be smaller than the x-axis value of the first target point to smooth the camera motion curve. The magnitude of the reduction is based on the parameter τ_{x1} and the target's size. T_{x1} is calculated based on the model components' corresponding cross-sections on the two planes involved in the axis to avoid the generated path penetrating the model, written as

$$\tau_{x1} = \max(dia_{XZ} - OC_{1XZ}, dia_{XY} - OC_{1XY}) \quad (1)$$

where dia_{XZ} is the diagonal length of the model's bounding box on the X-Z plane, dia_{XY} is the diagonal length of the model's bounding box on the X-Y plane, OC_{1XZ} is the projection length of the vector formed by the first target point and the center of the model on the X-Z plane, where 1 is the sequence number of the first target, and OC_{1XY} is the vector's projection length formed by the first target point and the model's center on the X-Y plane.

Bounding-based Selection

This method determines the control points based on the bounding box's size and position of the model component, where the control point location is further adjustable by the user. The first and third target points are set based on the desired roaming path, and the second target point is generated based on these two target points. The second target point is first set as the middle point between the first target point and the third target point and then adjusted on an individual axis among the X-axis, Y-axis, and Z-axis according to the desired path and the spatial relationship between this point and the model. Moreover, in this method, these target points are on one side or line of the model's bounding box. Each target point has one corresponding control point. The control points are also located outside the model, and the distance to the target points is adjustable.

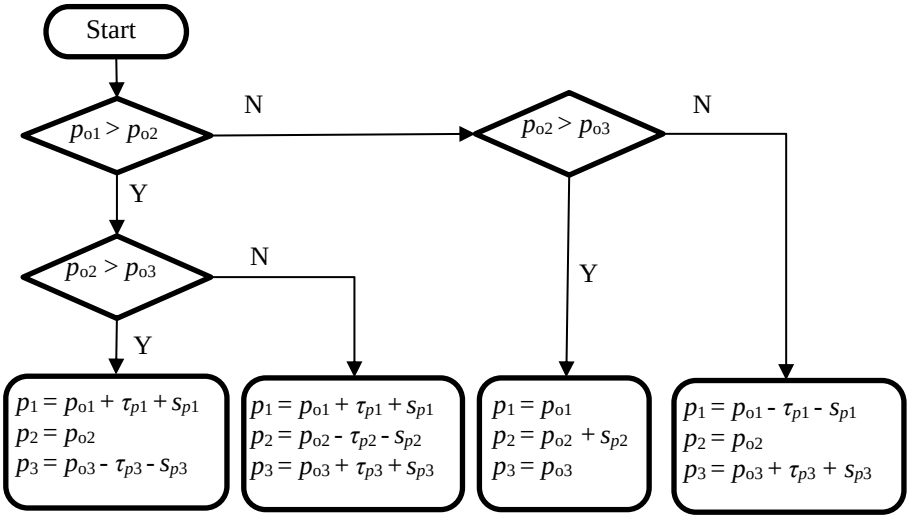

Figure 2. The algorithm for calculating each coordinate value of a control point, where each coordinate value is $\boldsymbol{p_i}$, $\boldsymbol{p} = \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$, $\boldsymbol{i} = \{\boldsymbol{1}, \boldsymbol{2}, \boldsymbol{3}\}$.

The coordinates of the control points are generated based on the corresponding target point and the center of the model; the equation is as

$$Q_i = C + (\alpha_i + 1)(O_i - C) \qquad (2)$$

where Q_i is the i^{th} control point, C is the center of the model, O_i is the i^{th} target point, α_i is a nonnegative adjustable parameter for the i^{th} control point, $i = \{1, 2, 3\}$.

Path Generation

A smooth motion path is a fundamental requirement of automatic roaming. Many curve fitting methods have been used to generate smooth paths, such as natural cubic splines and Bezier curves. Bezier curves are widely used to model smooth curves. However, there are some problems with Bezier curves (Luo, Li, and Mo 2017; Li et al. 2018). Because a Bezier curve does not pass through the intermediate control points, it is challenging to select them properly, especially for systems that automatically suit all models and generate control points. In addition, the Bezier curve only has C^1 continuity, and the natural cubic spline has C^2 continuity. This paper uses the 3D natural cubic spline interpolation algorithm to generate a 3D path based on the control points.

The natural cubic spline interpolation includes cubic polynomials whose gradients match the measured data points. Moreover, these cubic polynomials are continuous up to their second derivative. A natural cubic spline curve is defined below in the matrix notation of a general cubic spline curve.

$$[x(t)\ y(t)\ z(t)] = [1\ t\ t^2\ t^3]\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = [1\ t\ t^2\ t^3]\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_y \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \qquad (3)$$

where $m_{lk}, l, k = \{1, \ldots, 4\}$ are constants determined according to different cubic splines, and these parameters constitute the basis matrix in the equation; $P_j = (x_j, y_j, z_j)$, $j = \{1, \ldots, 4\} - 1$ is a 3D vector that the users select for different cubic splines; t is a parameter $t \in [0,1]$; a_v, b_v, c_v, d_v, $v = \{x, y, z\}$ are parameters to be solved.
Use $x(t)$ as an example to describe the details of the natural cubic spline curve. Consider $x(t)$ for a set of $n + 1$ points (x_0, x_1, …, x_n), the i^{th} piece of $x(t)$ is represented by

$$x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \qquad (4)$$

where $i = 0, \dots, n-1$,

$$\begin{aligned} a_i &= x_i(0), \\ b_i &= x_i'(0), \\ c_i &= 3\big(x_i(1) - x_i(0)\big) - 2x_i'(0) - x_i'(1), \\ d_i &= 2\big(x_i(0) - x_i(1)\big) + x_i'(0) + x_i'(1), \end{aligned} \qquad (5)$$

and the derivatives are also required to match at the points, so

$$\begin{aligned} x_i(1) &= x_{i+1}(0) \\ x_i'(1) &= x_{i+1}'(0) \\ x_i''(1) &= x_{i+1}''(0) \\ x_0(0) &= x_0 \\ x_{n-1}(0) &= x_{n-1} \\ x_0''(0) &= 0 \\ x_{n-1}''(0) &= 0 \end{aligned} \qquad (6)$$

The above equations are generated for each natural cubic spline segment and rearranged to form a matrix function of the first derivatives $x_i'(0)$, $i = 0, \dots, n-1$. The control parameters for each curve segment are then solved to produce the natural cubic spline curve (Bartels, Beatty, and Barsky 1987).

Camera Control

In this approach, camera control includes two aspects. The first one is to let the system display different models of different sizes. No matter what model is imported, this system can generate a view that fits the screen size, which is friendly for switching models with considerable size differences.

For adaptively displaying different sizes models, the camera needs to adjust its initial position to fit the model. The distance from the camera to the side closest to the camera is

$$d = {}^{y}/_{2a \tan 0.5\alpha} \qquad (7)$$

where *y* is the model's actual height, *a* is the ratio of the model's height on the screen to the screen height and adjustable, α is the camera's view angle and equals 60.

Another issue of camera control is to adjust the camera's orientation when the camera moves along its 3D path. The camera moves along the 3D path generated in Section Path Generation to implement component exploration for a selected virtual model. As illustrated in Figure 3, the target curve is generated, another 3D natural cubic spline curve based on the virtual model's target points' coordinates. The camera looks at the target curve when it moves along its tracking path and rotates to make its Look At vector point at the target curve's current position to eliminate the camera instability problem that frequently occurs to automatic roaming. Then, the animation of the camera is realized across keyframes.

EXPERIMENT RESULTS

The proposed approach has been implemented in the Unity 3D engine to demonstrate its unique ability to allow adaptive camera movement to guide procedural guidance. Figure 4 shows the automatically generated 3D camera motion path for two adjacent steps in a troubleshooting procedure for the MK-18 rifle, Failure of Magazine to Lock in Weapon, using the 3D natural cubic spline curve. This procedure consists of 4 steps: 1) adjust magazine catch, 2) disassemble and clean, 3) replace magazine catch spring, and 4) finally replace magazine catch. For automating the maintenance procedure, the camera looks first at the magazine catch, then at the spring, and then at the magazine

catch. Two adjacent steps are used as an example to show the generated camera path. The first target is the magazine catch in these two adjacent steps, and the last target is in the spring. The desired path is from the magazine catch through the weapon's top to the spring. So, the middle target is set on the top of the weapon model based on the rules in the component-based selection method. Next, the control points are selected based on the component-based selection method.

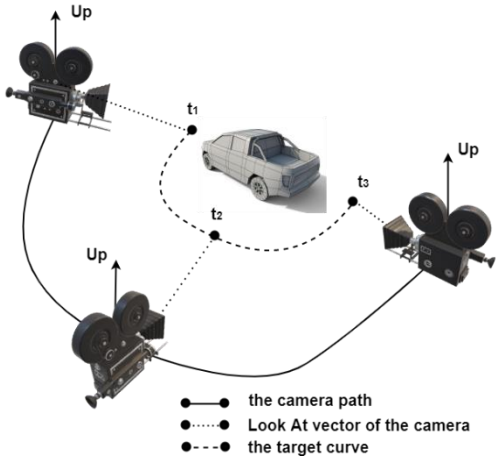

Figure 3. The specification of a camera path in a 3D environment. The camera path defines the location of the camera. The camera's orientation is calculated based on the camera path and the target curve. The time parameters, t_1, t_2, t_3, are different for different curves. If the camera path is generated using the natural cubic spline curve, the path consists of two pieces, $t_1 = 0$ and $t_2 = 1$ are for the first piece, and $t_2 = 0$ and $t_3 = 1$ are for the second piece. If the path is generated using the Bezier curve, $t_1 = 0$, $t_2 = 0.5$, and $t_3 = 1$.

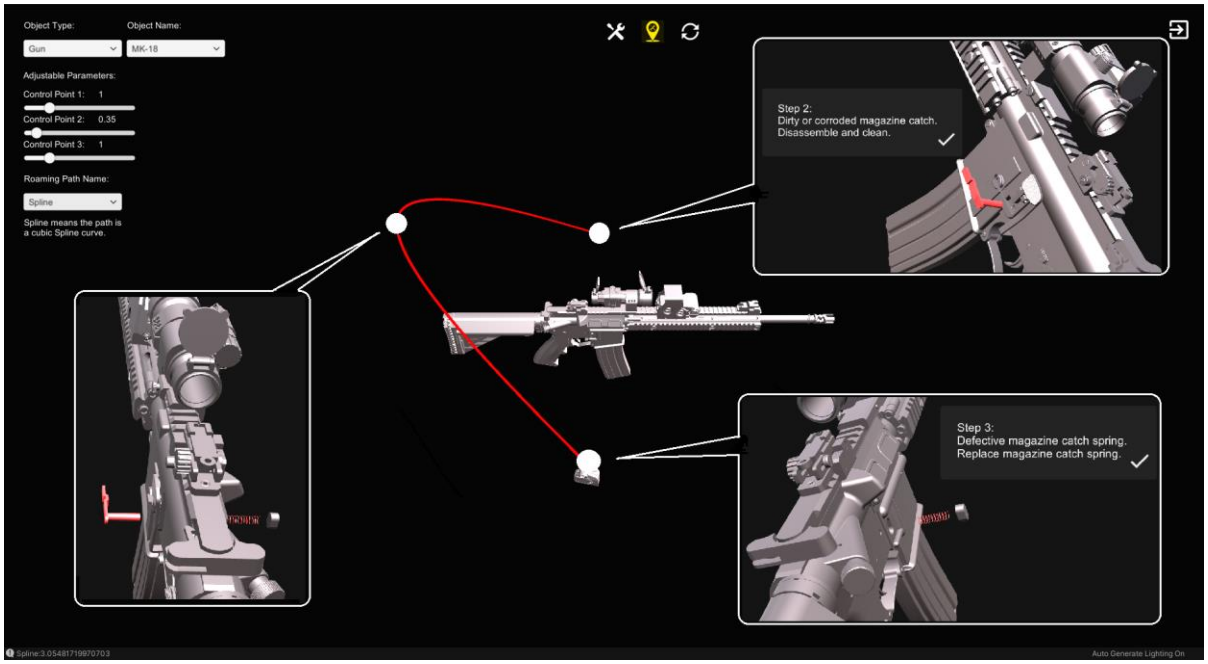

Figure 4. A screenshot of automatic camera motion for a troubleshooting procedure of the MK-18 rifle model. The Adjustable parameters are used to adjust control points in the control-point selection method. The red curve is the generated camera path that moves around the weapon model. The camera path uses the 3D natural cubic spline curve to transition from the second step to the third step of a troubleshooting procedure for the model.

The generated 3D natural cubic spline camera paths based on the bounding-box method are shown in Figure 5 and Figure 6. The initial value of α_i, $i = \{1, 2, 3\}$, in equation (1) for the weapon model in Figure 4 and Figure 5 is $(1, 0.35, 1)$, and $\alpha_i \in [0, 5]$ set by the heuristic method and suitable for most models. These three control point parameters can be used to adjust the relative positions between the three control points and the model. Users can change these three parameters to generate a custom path. Figures 5 and 6 show the effect of these three parameters, wherein the initial parameters are 1, 0.35, and 1, the adjusted parameters are 2, 0.25, and 2. The red curves in Figure 5 and Figure 6 are the generated 3D natural cubic spline curves.

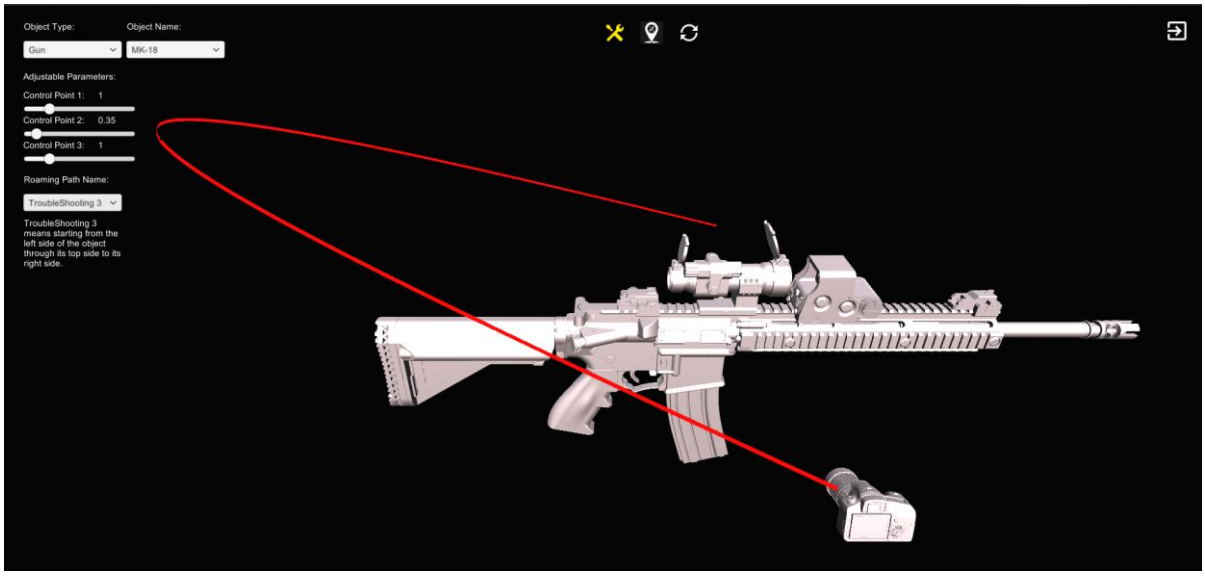

Figure 5. A 3D natural cubic spline path with initial control point parameters from the weapon model's left side through its rear to its right side. The Adjustable parameters are set as 1, 0.35, and 1.

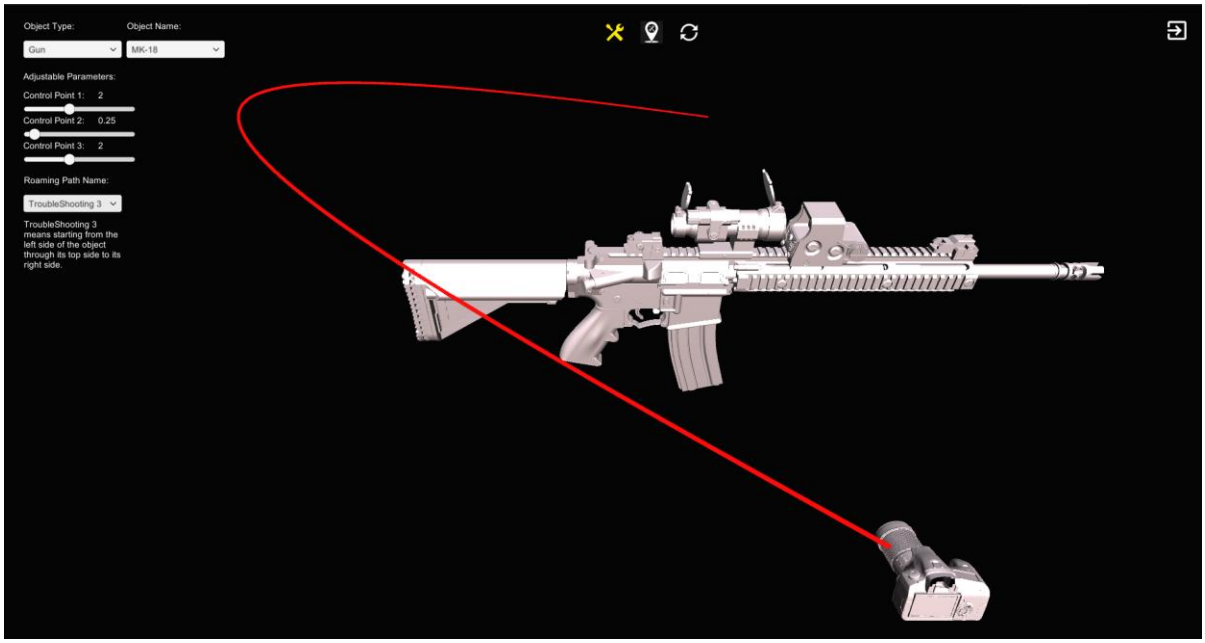

Figure 6. A 3D Natural cubic spline path with adjusted control point parameters from the weapon model's left side through its rear to its right side. The Adjustable parameters are set as 2, 0.25, and 2.

The Bezier curves have been widely used to generate different paths and have different polynomial functions depending on the number of control points. The Bezier curve with four control points is a commonly used trade-off method (Li et al. 2018). Since the Bezier curve only passes through the first and last control points, and other control

points are not on the curve, its control point positions are usually set manually or interactively. Therefore, the Bezier curve is not a good option for automatic camera motion generation. In this paper, the Bezier curve illustrates other performances of interpolation algorithms, such as smoothness and cost time. The two intermediate control points of the Bezier curve are manually selected in this paper. Figure 6 shows the automatically produced path using the 3D Bezier curve for the procedure in Figure 3. Finding the appropriate control points for the 3D Bezier path is challenging. Furthermore, the 3D Bezier curve is more time-consuming than the 3D natural cubic spline. The simulation experience based on the natural cubic spline curve needs 3.05s; instead, the simulation experience using the 3D Bezier curve requires 10.86s, shown at the left bottom corner of Figure 4 and Figure 7, respectively. Therefore, based on the results of these experiments, it can be concluded that the 3D natural cubic spline curve is suitable for procedural guidance in virtual environments.

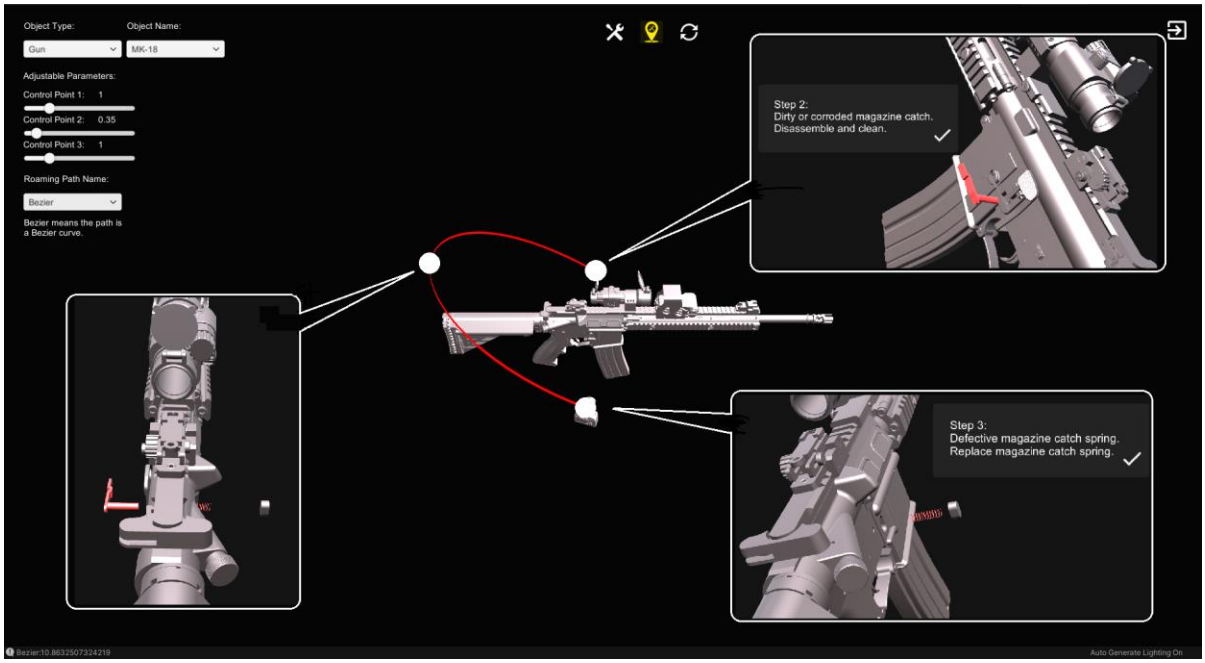

Figure 7. The camera path is generated using the 3D Bezier curve to transition from the second step to the third step of a troubleshooting procedure for the MK-18 rifle. The red curve is the generated camera path along which the camera moves around the weapon model.

CONCLUSIONS

This paper proposed an adaptive camera motion generation method for procedural guidance in virtual environments. It designs two ways to select control points for the camera path automatically. Then the paper uses the 3D natural cubic spline algorithm to generate a camera motion path and compares it with the 3D Bezier algorithm. Then a controlled camera moves along the 3D path to provide users with optimized views for the virtual environment. The experimental results illustrate that the 3D natural cubic spline algorithm is more suitable for automatic path generation, requiring a shorter simulation time. Moreover, when the user learns a procedure while assembling a machine or doing maintenance, the automatically generated camera motion could effortlessly guide the user to the next step. The user can also modify control point parameters in real-time to achieve a more customized visual experience. This approach would result in users exploring the virtual model's components for specific procedures in a machine's assembly or maintenance process.

REFERENCES

Bartels, Richard H., John C. Beatty, and Brian A. Barsky. 1987. "Hermite and Cubic Spline Interpolation." In *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, 9-17. San Francisco, CA: Morgan Kaufmann.

Blinn, Caltech Jim. 1988. "Where Am I? What Am I Looking at?" *IEEE Computer Graphics & Application* 8 (4):76-81.

Bracq, Marie-Stéphanie, Estelle Michinov, Bruno Arnaldi, Benoît Caillaud, Bernard Gibaud, Valérie Gouranton, and Pierre Jannin. 2019. "Learning Procedural Skills with a Virtual Reality Simulator: An Acceptability Study." *Nurse Education Today* 79:153-160.

Drucker, Steven M., and David Zeltzer. 1994. "Intelligent Camera Control in a Virtual Environment." Proceedings of Graphics Interface.

Dwivedi, Prateek, David Cline, Cecil Jose, and Ronak Etemadpour. 2018. "Manual Assembly Training in Virtual Environments." 2018 IEEE 18th International Conference on Advanced Learning Technologies, Mumbai, India, Jul. 9-13, 2018.

Hajji, Said, Abdellah Lamnii, Boujemaa Danouj, and Mounir Lotf. 2018. "Hermite Interpolation By Piecewise Cubic Trigonometric Spline with shape parameters." Proceedings of the 1st International Conference of Computer Science and Renewable Energies (ICCSRE 2018), Ouarzazate, Maroc, Nov. 22-24, 2018.

Hsu, Wei-Hsien, Yubo Zhang, and Kwan-Liu Ma. 2013. "A Multi-Criteria Approach to Camera Motion Design for Volume Data Animation." *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS* 19 (12):2792-2801.

Ji, Guangfeng, and Han-wei Shen. 2006. "Dynamic View Selection for Time-Varying Volumes." *IEEE Transactions on Visualization and Computer Graphics* 12 (5):1109-1116.

Li, Jiayao, Ruizhi Sun, Chunming Cheng, and Sicong Li. 2018. "Roaming Path Generation Algorithm and Optimization based on Bezier Curve." *IFAC-PapersOnLine* 51 (17):339-345.

Loch, Frieder, Mina Fahimipirehgalin, Julia N. Czerniak, Alexander Mertens, Valeria Villani, Lorenzo Sabattini, Cesare Fantuzzi, and Birgit Vogel-Heuser. 2019. "An Adaptive Virtual Training System Based on Universal Design." *IFAC-PapersOnLine* 51 (34):335-340.

Luo, Lihong, Jiazhen Li, and Jianqing Mo. 2017. "A Cardinal-based Algorithm for Automatically Roaming in Virtual Scene." 2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, Sept. 20-22, 2017.

Sokolov, Dmitry, Dimitri Plemenos, and Karim Tamine. 2006. "Methods and Data Structure for Virtual World Exploration." *The Visual Computer* 22:506-516.

Stoev, Stanislav L., and Wolfgang Straßer. 2002. "A Case Study on Automatic Camera Placement and Motion for Visualizing Historical Data." IEEE Conference on Visualization 2002, Boston, MA, USA, Oct. 27 -Nov. 1, 2002.

Yang, Kwangjin, and Salah Sukkarieh. 2008. "3D Smooth Path Planning for a UVA in Cluttered Natural Environments." 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, Sept. 22-26, 2008.