Efficient Camera Path Planning Algorithm for Human Motion Overview

I-Cheng Yeh, Chao-Hung Lin, Hung-Jen Chien, and Tong-Yee Lee
National Cheng-Kung University, Taiwan

Abstract

Camera path planning for character motions is a fundamental and important research topic, benefiting many animation applications. Existing optimal-based approaches are generally computationally expensive and infeasible for interactive applications. In this paper, we propose an efficient approach that can take many constraints of finding the camera path into account and can potentially enable interactive camera control. Instead of solving a highly complicated camera optimization problem in a spatiotemporal four-dimensional space, we heuristically determine the camera path based on an efficient greedy-based tree traversal approach. The experimental results show that the proposed approach can efficiently generate a smooth, informative, and aesthetic camera path that can reveal the significant features of character motions. Moreover, the conducted user study also shows that the generated camera paths are comparable to those of a state-of-the-art approach and those made by professional animators.

Keywords: camera path planning, viewpoint selection, visualization

1 Introduction

Generally, the manipulation of mocap data requires a viewer to determine efficiently a proper camera path for viewing the time-varying motions of human characters. In this study, the main theme of camera path planning is finding a sequence of suitable camera configurations that can clearly display or illustrate relevant characters and properly emphasize significant motion features while obeying cinematographic rules [1] and visibility constraints [2]. Most previous studies have focused on exploring optimal paths based on visual quality measurements [3] that integrate all related visual factors. However, this optimization problem is complex and time consuming because of a significant number of visual factors and the huge search space. Thus, approaches based on an optimization solver are generally infeasible in real-time computing, and can only deal with short clips. In contrast, we aim at developing an algorithm that efficiently selects a near-optimal camera path and accounts for various potential visual factors.

A naïve approach to solving the camera path optimization problem is simply considering a camera path that collects the best viewpoints of all frames. However, this simple method generally generates a highly unsmooth path because slight changes in character orientation may cause significant changes in estimating the best viewpoints [4]. A possible approach to solving this problem is addressing the trade-off between the defined visual metrics for the best viewpoints and frame coherence [4, 5]. Inspired by this trade-off strategy, we introduce an efficient approach that is capable of accounting for many possible visual factors and obtaining a near-optimal camera path to emphasize significant motion features. Based on the previous work [4] which integrates various visual metrics to generate a viewpoint space for each frame, we further consider a more elaborate visibility descriptor to measure viewpoint quality. Instead of solving a highly complicated camera optimization problem in a spatiotemporal four-dimensional (4D) space, we heuristically determine the camera path based on an efficient greedy-based tree traversal approach. Thus, searching an optimal path in a huge search space is simplified as a tree traversal problem. The computational cost is very low, and a near-optimal path can be obtained. In addition, some cinematographic techniques such as camera cuts, slow motion and multi-view replay can be easily added to the proposed scheme.

2 Related Work

In this section, we survey previous studies that are most closely related to our work. For a more comprehensive study on camera control, we refer the reader to [6].

There have been many studies reporting automatic camera control strategies. Many of them have formularized the camera path problem into an optimization problem that maximizes local properties (e.g., subject's visibility in each frame) while considering global properties, such as camera speed [4, 5, 7, 8, 9]. In [7, 8, 9], the optimal position of the camera is first determined for each individually shot subject according to some defined constraints, and then all camera parameters are automatically refined based on a general-purpose continuous optimization scheme. To

satisfy the constraints, the constraint solver attempts to find each camera parameter. This requires an exhaustive search over a huge space. Thus, some studies have aimed at reducing the size of the search space by utilizing standard hierarchical data structures [10, 11], using stochastic search approaches over the search space [9], restricting the solution to only a small set of camera configurations [5, 10, 12], or adopting quantum annealing [13] that is usually used for searching in a huge space with many local minima. In this paper, a greedy-based search heuristics is used to solve the camera path optimization problem approximately but efficiently. The proposed approach is very efficient because it reduces the optimization problem, which has complex configurations over a huge high-dimensional space, into a simple tree traversal problem defined over a tree structure. The computational cost is very low, and a near-optimal path can be obtained.

Occlusion between objects is one of the main problems in camera path planning. This problem can be reduced or even solved by integrating the related metric into the framework presented in [4, 8, 9, 14]. In addition to handling such object occlusions, Assa et al. [4] further examined the self-occlusions of various human limbs. In this paper, an additional visibility factor in object occlusions, named foreground factor, is considered and integrated with the metric of static scene occlusion. To the best of our knowledge, this extension of occlusion constraint has not yet been addressed in previous studies on camera control.

Recently, some efficient camera control techniques have been proposed [5, 15, 16]. The approach proposed by Kwon and Lee [15] is based on the integrated area spanned by the bone motions of characters. They focused on viewpoint selection, and used selected viewpoints to interpolate camera paths. Assa et al. [16] focused on dynamically and efficiently selecting candidate viewpoints, instead of optimizing a complex quality measurement. The selection is based on the correlation between each view stream and the motion in a given scene. Halper et al. [5] addressed the trade-off between selecting the best viewpoints, which are measured by a set of viewpoint quality metrics [8, 17], and enforcing frame coherence. Inspired by this balancing concept [5] and the dynamic viewpoint selection concept [16], we introduce an efficient approach based on both tree traversal and greedy search strategy, which does not only account for all potential visual metrics to select the best viewpoints but also obtains a near-optimal camera path to emphasize significant motion features.

3 Viewpoint Quality Measurement

In this paper, we only address on finding the parameter of camera position, whereas the viewing direction is set as the vector looking at the root node, the view-up vector is perpendicular to look-at vector and close to z-axis, and the field-of-view angle is set to 60 degrees in our experiment. Thus, in this section we describe the quality measurement for viewpoint only. The metrics for viewpoint quality measurement is defined according to visibility requirements and aesthetic elements. In addition to the metrics, character facing, viewing distance, widest aspect, limb visibility, and static scene occlusion, discussed in the work [4], we consider user-defined constraints in the viewpoint quality measurement and improve the estimation of static scene occlusion. In the work of Assa et al. [4], the metric of widest aspect is formulated as the angle between the viewpoint angle and the third eigenvector, which is determined by principal components analysis on the positions of pose joints. The results of this estimation are sufficiently similar to the real size of the object silhouette, which is suggested as a metric of viewpoint quality by Polonsky et al. [18]. The metric of limb visibility (or called character visibility) is calculated by examining the silhouette of each of the six main body parts (head, torso, two legs, and two arms) from various locations. The calculation of this metric can be accelerated by simplifying the main body parts into a set of fitted ellipsoids. Each ellipsoid is specified by a specific color; thus, the number of rendered pixels for each color indicates the visibility of a body part. Following the viewpoint metrics described in [5], the metric of character facing is defined as the distance between the candidate viewpoint and the three-quarter view, which is a view preferred by most people according to the psychophysical experiments [19]. The metric of viewing distance is simply estimated by the distance between the position of the candidate camera and the user-defined optimal viewing distance. In the estimation of static scene occlusion [4, 5], the animated character must remain visible to the viewer. However, this simple visibility estimation is not sufficient in illustrating animated characters. For example, in 1, some unwanted static objects appear in front (left figure) and at the back (right figure) of the motion object. Obviously, in terms of the visibility of the animated character, people prefer the right figure to the left figure. It is because that people generally focus on objects in the foreground in a motion scene. Therefore, in addition to the occlusion factors suggested by Halper et al. [5] (we refer the reader to this work for more details about these factors), we add the foreground factor to this metric. This can be efficiently achieved by checking if the animated character lies in front of the obstructions or not. We simply give a significant penalty in cases where the animated character lies at the back of obstructions in this metric [5].

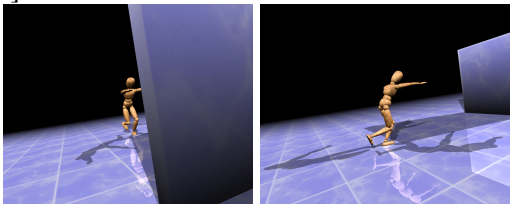


Figure 1: Example of static scene occlusion. The animated character lies at the back (left) and in front (right) of the obstruction.

Although all possible metrics and pose saliency suggested by [4] are taken into account in the mea-

surement of the viewpoint and camera path quality, it is still difficult to satisfy various user requirements. Therefore, the proposed system allows users to set constraints on camera parameters in an interactive manner. Users can set the camera on a specific pose (frame) and a camera position. Our system can efficiently generate a camera path responding to this user requirement. To add and solve these constraints, a possible solution is adding them as constraints in the camera path finding problem and solving them using an optimization approach. However, adding constraints in tree traversal significantly increases computational complexity. Thus, we set these constraints as a metric of viewpoint quality, and integrate them with other quality metrics for efficient computation. The metric for user-defined constraint is simply set as the summation of position and viewing-direction distances to the user-selected position and its corresponding viewing direction. Then, the viewpoint quality(VQ) is formulated as the linear combination of these metrics:

$$\begin{aligned} VQ(f,p) = {} & w_a \cdot Facing(f,p) + w_d \cdot Distance(f,p) + \\ & w_w \cdot Widest(f,p) + w_v \cdot LimVisibility(f,p) + \\ & w_o \cdot Occlusion(f,p) + w_{fu} \cdot UserDefined(f,p) \end{aligned} \quad (1)$$

where f is the frame index and p is an arbitrary position. The parameters w_a, w_d, w_w, w_v, and w_o are the weighting factors of character facing, viewing distance, widest aspect, limb visibility, and static scene occlusion, respectively. The parameterw_{fu} is the weighting factor of the user-defined constraint in frame f. With the aid of these parameters and the efficient algorithm on camera path finding, users can tune weightings to meet their requirements in an interactive manner. Note that to evaluate our approach, as well as for purposes of comparison, we set the parameter w_{fu} to 0.5 for frames with user-defined constraints; the other parameters are set to 0.1 in all experiments. To force the camera path passing through the user-defined viewpoints, we assign a larger weight to user-defined constraints. Besides, the weight for user-defined constraints is applied not only to the specified frame but also to neighboring frames. The weights w_{fu} are exponentially decayed over neighboring frames. In this manner, the proposed method moves the camera gradually to the specified position.

The quality of each viewpoint in the potential space is calculated by Eq. 1. We can form a viewpoint quality map in which the value of each point represents the quality or cost of the camera path planning. Note that the viewpoint quality map for each individual metric can be calculated and combined, except for the metric of user-defined constraints. For these constraints, we recalculate the quality maps for frames with constraints defined interactively by users. Generally, there are few user-defined constraints; hence, quality maps are recalculated only in a few frames. Figure 2 shows some viewpoint quality maps for the defined metrics.

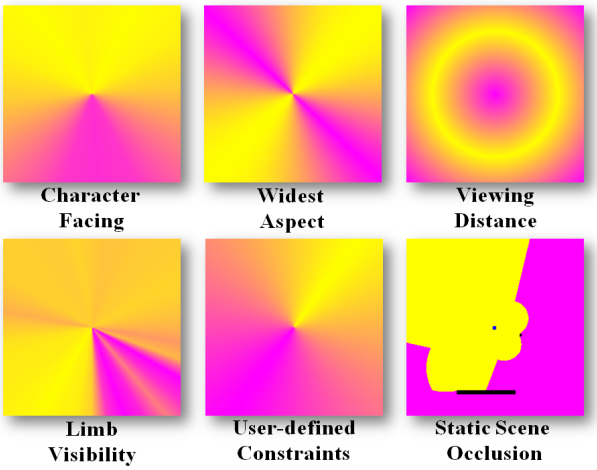


Figure 2: Viewpoint quality maps for defined metrics. Quality is visualized by colors ranging from yellow (highest quality) to pink (lowest quality).

4 Camera Path Planning

Once the viewpoint quality map for each frame is generated, a camera path is subsequently determined using these maps. In fact, these maps define a very large spatiotemporal 4D space. To find a camera path in this space, we first select the best viewpoint (i.e., viewpoint with a minimal value in the quality map) for each frame. If some user-defined constraints are set in some frames, we recalculate the quality maps and reselect the best viewpoints for these frames.

In this paper, we propose an efficient two-layer camera path planning approach. Our approach is designed based on a binary tree structure and a greedy traversal strategy. We schematically illustrate the work flow of the proposed approach in Figure 4. The approach consists of two layers: tree traversal in the temporal domain (the first layer) and greedy path finding in the spatial domain (the second layer). In the first layer, a binary tree for a given motion clip is dynamically constructed and simultaneously traversed in the temporal domain. In this binary tree, a node represents a candidate viewpoint for a frame of motion clip and each node is determined in the second layer as well as in the spatial domain. In this tree structure, an edge represents a possible path from a node to its descending node, which is also determined in the second layer. Instead of solving a complex optimization problem defined over a high-dimensional space, we efficiently search a near-optimal path in the spatiotemporal 4D viewpoint quality map. To speed up the camera path search, a binary tree on the 4D quality map space is dynamically constructed and simultaneously traversed.

We assume that the camera path from the first frame to the $(i-1)^{th}$ frame is determined. We want to select an appropriate viewpoint not only for frame i but also for the determined camera path. Thus, the naïve approach of selecting candidate viewpoints from viewpoints with local minimal values on the quality map is inappropriate. To take both the smoothness of the

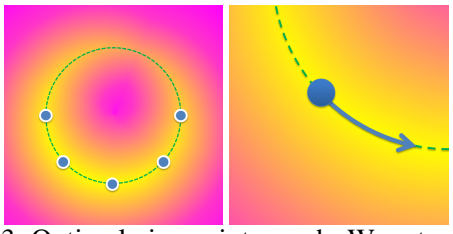


Figure 3: Optimal viewpoint search. We set some initial positions on the sphere with the radius of optimal viewing distance (left) for the steepest descent algorithm (right) to find the optimal viewpoint.

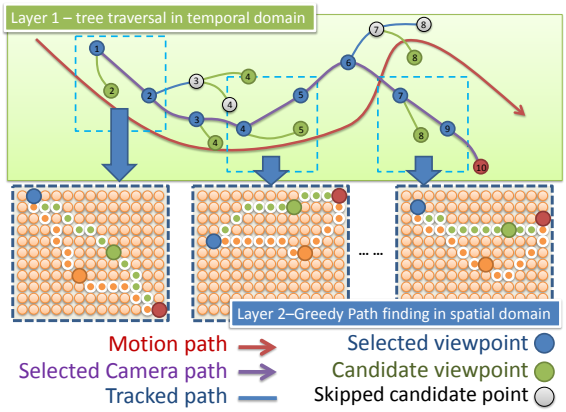


Figure 4: Workflow of the proposed two-layer camera path planning.

camera path and viewpoint quality into consideration, we select the candidate viewpoint of frame i from possible camera paths between the selected viewpoint of frame $(i-1)$ (the start of the paths) and the best viewpoint of frame i (the destination of these possible paths). In addition, a speed constraint is set on the candidate viewpoint selection to synchronize the speeds of the camera and the animated character. To avoid a dead-end path and for computational simplicity, we only select two candidate camera paths. The first path can be easily determined by the best-first search A^* algorithm that is based on the quality map of frame i. To ensure that the determined paths and the selected candidate viewpoints are not too close, we mark the regions near the candidate viewpoint selected in the first path (the gray viewpoints in Figure 5). To determine the second path, we adopt the A^* least-cost path algorithm again under the constraint that the path cannot pass through the marked viewpoints shown in gray color. Under the speed constraint (i.e., camera movement cannot pass through the red curve shown in Figure 5), we heuristically select such a viewpoint along the found least-cost path with the maximum camera movement starting from frame $(i-1)$, thereby approaching the best viewpoint as closely as possible. For example, in Figure 5, the colors in the map ranging from pink to yellow represent viewpoint quality from low to high. The blue viewpoint is the current viewpoint and the brown viewpoint is the best viewpoint of the next frame. Two candidate paths can be determined according to the quality of the map and the A^* algorithm. With the help of the speed constraint, the purple and green viewpoints are selected for these two candidate paths. In our implementation, the speed limitation is simply set according to character motion speed, as shown in Figure 6. We first smooth the motion speed by a Gaussian filter to obtain smooth camera movement (or avoid abrupt changes in camera speed) (Figure 6, middle). Subsequently, we further linearly re-scale the smoothed speed to a narrow range with faster speed to take the smoothness of the camera path into more consideration (Figure 6, right) and obtain a constraint on maximal speed. The range is set to $[MaxSpeed*0.8, MaxSpeed*0.5]$; the constant $MaxSpeed$ is the maximum motion speed. This range means that a faster camera speed is set as a speed constraint. The rationale behind this re-scaling is very simple. In Figure 6(left), there is a significant variation in the moving speed of the human character. Thus, if our camera follows the human character movement completely, we cannot have a stable/smooth camera movement. Therefore, we smooth this speed and then re-scale it to yield a stable camera movement.

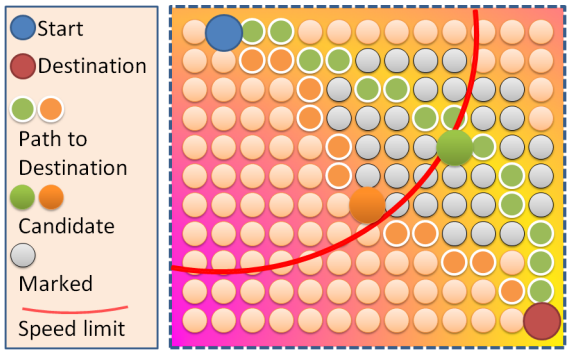


Figure 5: Illustration of candidate viewpoint selection in the spatial domain.

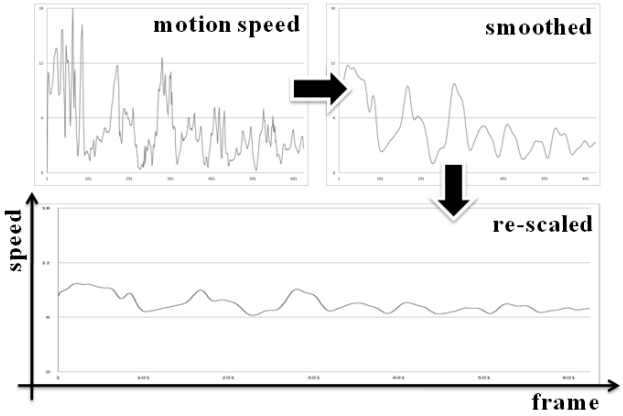


Figure 6: Camera speed constraint. The motion speed (left) is first smoothed (middle) and then linearly re-scaled to a desired range (right).

The proposed tree traversal algorithm (i.e., camera path determination) is described below with a pseudo-code. The candidate viewpoints obtained from the second layer are the children nodes of the current node, denoted as $node_c$ in the pseudo-code. By a

greedy strategy, the binary tree is dynamically grown (or constructed) and traversed in two steps: (1) we extract the candidate viewpoint, denoted as $node_m$, from the set of candidate viewpoints, denoted as V; and (2) we determine two new candidate viewpoints for the node selected in the first step using the function $NewCandidate()$, and add these two candidate viewpoints to set V. These two steps are iteratively performed until set V becomes empty or the last frame of the clip is reached. To avoid passing through low-quality viewpoints, we set a quality constraint on the selected viewpoint $node_m$. This quality constraint is performed by the function $TVQ(node_m)$ in the pseudo-code. If the summation of the viewpoint quality of the nodes within the window located at $node_m$ (the window size is 30 nodes in our experiment) is lower than a defined threshold (i.e., $TVQ(node_m) > q_{thr}$), we back-trace to its parent and even to its ancestor nodes until the quality criterion is satisfied. However, to take efficiency into account and avoid too many back-tracings, we limit the process of back-tracing by checking if the path distance between the current node and the selected candidate is less than or equal to a defined threshold d_{thr} using the function $Distance()$. In Figure 7, for example, the red node in the 4^{th} step is an unqualified viewpoint. Therefore, we back-trace to its ancestor nodes to find a qualified node. In this case, the qualified node appears in the grandparent node. Note that in our implementation, the threshold q_{thr} is automatically decreased and the threshold d_{thr} is increased when a dead-end path is met. In addition, To reduce the distortion artifacts known as aliasing when representing a quality map by a lower resolution grid, we perform a smoothing operation with a small Gaussian kernel on the determined viewpoints and viewing directions during the tree traversal.

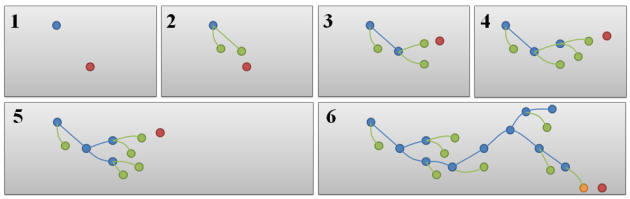


Figure 7: Illustration of our greedy binary tree traversal. The red node is the best viewpoint and also the target in that step. The blue nodes are the traversed nodes.

5 Cinematographic Techniques

Some cinematographic techniques, such as multi-camera, slow motion and multi-view replay techniques, are easily realized through our proposed scheme. The integration of these techniques into our scheme is described in this section.

5.1 Multi-camera technique

The multi-camera technique is necessary when character motions are fast and changeful. Because of the speed constraint, it is difficult to sufficiently capture the characteristic features of motions from a camera path. Therefore, we implement the effect of the multi-camera technique to split the camera path into multi-paths in our system. This is achieved by simply selecting the best viewpoint when the viewing direction of the selected viewpoint is much different from the viewing direction of the best viewpoint. In the implementation, we check the angle between the viewing directions of the best viewpoint and the selected viewpoint. If the angle is larger than a defined threshold, the selected viewpoint is simply replaced by the best viewpoint, and the camera position changes/jumps from the selected viewpoint to the best viewpoint to imitate the effect of a camera cut.

5.2 Slow motion and multi-view replay techniques

The cinematographic effects of slow motion and multi-view replay are frequently used in professional games, such as a shot in soccer and a slam dunk in basketball. In our system, users can add these effects on a selected short clip. Our system generates multi-paths for this clip to enhance its characteristic features. Following the rules in cinematography, two or three action rebroadcasts are appropriate for important actions. Therefore, our system generates three different paths for the selected action. In addition to the generated camera path, we select two other camera paths: the three-quarter views calculated by the metric of character facing, and the widest aspect views calculated by the metric of widest aspect. The path of the three-quarter view can clearly show the front of the character and its action. The path of the widest aspect view can show the characteristic features of this action.

6 Experimental Results and Discussion

The experimental results were evaluated on an Intel Core i7 860 PC (4G memory, using single core). The thresholds used in the camera planning algorithm are identical in all experiments. The results of the camera path planning are shown in Figures 11. From the viewpoints of key-frames (see the images at the side) and the generated camera path, our approach can generate smooth and informative camera paths. For clearly evaluating the generated camera path, we refer the reader to the supplementary video. The time performance is shown in Table 1. For a clip with about 1,000 frames, our system usually takes less than 5 seconds (including the calculation of best viewpoints) to generate a camera path.

We evaluated the camera planning results using two measurements. The first measurement, the standard deviation of the camera speed (E_{dev}), can estimate the smoothness of the camera moving speed.

$$E_{dev} = \sqrt{\frac{1}{N-1}\sum_{f=2}^{N}(s_f - \bar{s})^2} \qquad (2)$$

where $s_f = path[f] - path[f-1]$ and $\bar{s}$ is the average camera speed. The second measurement, the standard

Table 1: Time performance. 1^{st} column: datasets; 2^{nd} column: the number of frames in the data set (#Frame); 3^{rd} column: execution time for best viewpoint determination (VP); 4^{th} column: execution time for camera path planning (CPP);

Dataset	frame (♯)	VP (sec.)	CPP (sec.)	Total (sec.)
Example 1	625	0.29	1.15	1.44
+ User defined	660	0.28	4.16	4.44
+ SceneOcc	645	0.48	2.97	3.45
Example 2	646	0.35	0.73	1.08
Example 3	1005	0.64	1.11	1.75
Example 4	1069	0.50	5.18	5.68

deviation of the camera viewing direction ($E_{direction}$), can estimate the smoothness of the viewing direction.

$$E_{direction} = \frac{1}{N}\sum_{f=1}^{N} |v(f) - v(f-1)| \qquad (3)$$

where $v(f)$ is the viewing direction of the viewpoint of frame f. We compare the generated camera paths not only with the most related approaches [4, 5] but also with the paths made by professional animators. The statistical results are shown in Table 2. The smoothness of camera position and viewing direction (i.e., E_{dev} and $E_{direction}$) in our approach are superior compared to those of related approaches, except for $E_{direction}$ in the case of the paths made by professional animators. This is because animators pay much more attention to the smoothness of viewing direction than to camera position. As for the camera path quality, we refer the readers to the accompaniment video for visual evaluation.

Table 2: Statistical comparison between our approach and the related approaches.

	E_{dev}	$E_{direction}$
Our approach	0.961	3.437
Animators	1.599	3.363
[5]	10.039	9.892
[4]	2.569	3.436

In Figure 8, we demonstrate the advantage of the proposed approach on scene occlusion. If a static object is added to the scene, the visual quality of the motion character is greatly affected in some frames when we ignore the occlusion of the static object or when we use the approaches suggested by [4, 5] (middle figure). In contrast, considering the foreground in the metric of static scene occlusion, our approach can greatly improve the visual quality (right figure).

We experimented on the metric of user-defined constraints and test if our approach can satisfy user requirements. The results are shown in Figure 9. The left top figure is the user-desired viewpoint, a side view of the character. If the user-defined constraints are not included in the measurement of viewpoint quality, the generated viewpoint in this frame is a frontal view, as shown in Figure 9(a), not satisfying

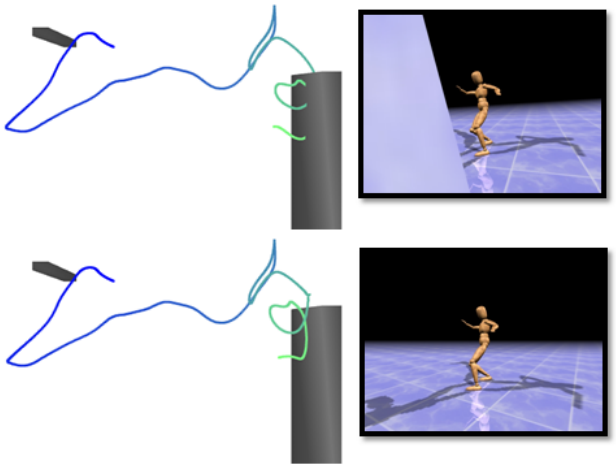


Figure 8: Scene occlusion. A camera path generated without the metric of scene occlusion (up), and a camera path generated with the metric of scene occlusion (down).

the user requirement. In contrast, with the aid of this metric, the proposed approach can satisfy the user requirement, as shown in Figure 9(b). To strengthen

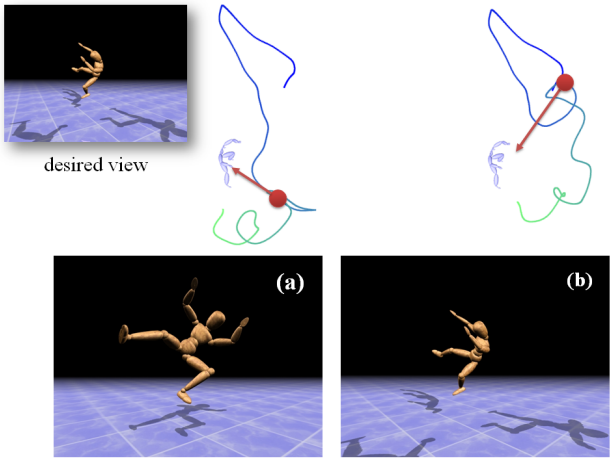
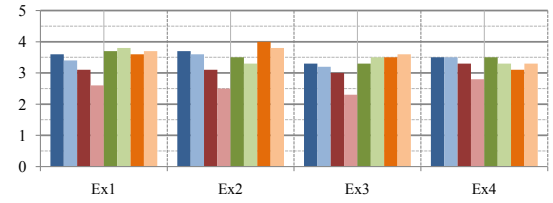


Figure 9: Demonstration of user-defined constraints. The camera path without (a) and with (b) the user-defined constraint, that is, the user-desired viewpoint as shown in the top left. The viewpoint and viewing direction of the viewpoint shown at the bottom are the red dot and the red arrow shown at the top.

further the comparison of our approach and related approaches, we conducted a user study involving 30 computer graphics students and researchers aged 24 to 40 years. Participants were required to grade some video clips showing the same animation by different camera control approaches. Participants were blind to the study objectives and the tested video clips were given at random. Participants were asked to grade two problems: (a) how well each clip describes the character action, and (b) how professional the camera control looks. A score may range from 0 to 4, where 4 represents the best quality. The results of the user study,

as shown in Figure 10, indicate that our approach is better than [5]. Moreover, our results are comparable to those of the optimal approach [4] and the work of professional animators.



	Ex1	Ex2	Ex3	Ex4
Our approach	**3.6**/3.4	**3.7**/3.6	**3.3**/3.2	**3.5**/3.5
[HHS01]	**3.1**/2.6	**3.1**/2.5	**3.0**/2.3	**3.3**/2.8
[ACYL08]	**3.7**/3.8	**3.5**/3.3	**3.3**/3.5	**3.5**/3.3
Animator	**3.6**/3.7	**4.0**/3.8	**3.5**/3.6	**3.1**/3.3

Figure 10: User study results. The dark colors and bold numbers are the scores for problem (a), and the light colors and non-bold numbers are the scores for problem (b).

7 Conclusions and Future Work

An automatic and efficient camera path planning approach was introduced. The optimal searching approach defined over a high-dimensional space is reduced to a simple tree-traversal problem defined over a double-layer binary tree structure. In addition, a more elaborate visibility descriptor on scene occlusion is proposed, and user-defined constraints are integrated in the measurement of viewpoint quality. Our approach is capable of integrating many potential visual metrics to select high-quality viewpoints while efficiently obtaining not only near-optimal but also smooth camera paths. The balancing of the best viewpoints and frame coherence can be efficiently achieved in our scheme. Experimental results show that our approach can generate smooth and informative camera paths to reveal successfully the significant motion features of characters. The results of our user study also show that the generated camera paths are comparable to optimal results generated by state-of-the-art approaches and by professional animators. Moreover, the computational cost or the interactive time of our approach is much lower than that of optimal-based approaches and professional animators. In the near future, we plan to improve the efficiency of our scheme on the computation of static scene occlusion, which is the bottleneck of viewpoint measurement, by GPU computing. Moreover, we plan to extend viewpoint quality measurement from evaluating single-character motions to multi-character motions.

References

[1] Li-wei He, Michael F. Cohen, and David H. Salesin. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. SIGGRAPH '96, pages 217–224, 1996.

[2] Michela Mortara and Michela Spagnuolo. Technical section: Semantics-driven best view of 3d shapes. *Comput. Graph.*, 33:280–290, 2009.

[3] Marc Christie, Rumesh Machap, Jean marie Norm, Patrick Olivier, and Jonathan Pickering. Virtual camera planning: A survey. In *In Proceedings Smart Graphics*, pages 40–52, 2005.

[4] Jackie Assa, Daniel Cohen-Or, I-Cheng Yeh, and Tong-Yee Lee. Motion overview of human actions. *ACM Trans. Graph.*, 27:115:1–115:10, 2008.

[5] Nicolas Halper, Ralf Helbing, and Thomas Strothotte. A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. In *Eurographics 2001*, volume 20(3), pages 174–183, 2001.

[6] Marc Christie and Patrick Olivier. Camera control in computer graphics. In *Eurographics 2006 State of the Art Reports*, pages 89–113, 2006.

[7] Steven M. Drucker and David Zeltzer. Camdroid: A system for implementing intelligent camera control. In *In 1995 Symposium on Interactive 3D Graphics*, pages 139–144, 1995.

[8] Thainimit S. Bares, W. H. and S. McDermott. A model for constraint-based camera planning. In *AAAI 2000 Spring Symposium on Smart Graphics*, pages 84–91, 2000.

[9] Nick Halper and Patrick Olivier. Camplan: A camera planning agent. In *AAAI 2000 Spring Symposium on Smart Graphics*, pages 92–100, 2000.

[10] Frank Jardillier and Eric Languénou. Screen-space constraints for camera movements: the virtual cameraman. *Computer Graphics Forum*, 17(3):175–186, 1998.

[11] Frédéric Benhamou, Frédéric Goualard, Éric Languénou, and Marc Christie. Interval constraint solving for camera control and motion planning. *ACM Trans. Comput. Logic*, 5:732–767, 2004.

[12] Ting-Chieh Lin, Zen-Chung Shih, and Yu-Ting Tsai. Cinematic camera control in 3d computer games. In *WSCG*, pages 289–296, 2004.

[13] B. Apolloni, C. Carvalho, and D. de Falco. Quantum stochastic optimization. *Stochastic Processes and their Applications*, 33(2):233–244, 1989.

[14] Jonathan H Pickering. *Intelligent Camera Planning for Computer Graphics*. PhD thesis, University of York, 2002.

[15] J.-Y Kwon and I.-K Lee. Detemination of camera parameters for character motions using motion area. *The Visual Computer*, 24:475–483, 2008.

[16] Jackie Assa, Lior Wolf, and Daniel Cohen-Or. The virtual director: a correlation-based online viewing of human motion. *Comput. Graph. Forum*, 29(2):595–604, 2010.

[17] Patrick Olivier, Jon Pickering, Nicolas Halper, and Pamela Luna. Visual composition as optimi-

sation. In *AISB Workshop on AI and Creativity in Entertainment and Visual Art*, pages 22–30, 1999.

[18] Oleg Polonsky, Giuseppe Patanè, Silvia Biasotti, Craig Gotsman, and Michela Spagnuolo. What's in an image: Towards the computation of the "best" view of an object. *The Visual Computer*, 21(8-10):840–847, 2005.

[19] Volker Blanz, Michael J. Tarr, and Heinrich H. Bülthoff. What object attributes determine canonical views. *Perception*, 28(5):575–600, 1999.

Algorithm 1 (Layer 1): Camera Path Determination

$V \leftarrow \emptyset$ {The stack of candidate viewpoints}
$V \leftarrow V \bigcup node_s$ {nodes is the root node of binary tree(i.e., the best viewpoint of the first frame)}
$node_c \leftarrow node_s$ {set $node_s$ to be the current node which is denoted as $node_c$}
repeat
  $node_m \leftarrow POP(V)$ {step1: extract and remove the top node from V}
  if $TVQ(node_m) > q\_thr$ and $Distance(node_m, node_c) \leq d\_thr$ **then**
    {check if the summation of viewpoint quality of nodes within the window located at $node_m$ is great than the defined threshold q_thr, and the path distance between $node_m$ and $node_c$ is less than and equal to d_thr}
    $node_c \leftarrow node_m$ {set $node_m$ as the current node}
    $node_a, node_b \leftarrow NewCandidate(node_c)$ {where $VQ(node_a) > VQ(node_b)$}
    {step 2: select the new candidate viewpoints in Layer 2 and add them to V}
    $PUSH(V, node_b)$ {push $node_b$ to V}
    $PUSH(V, node_a)$ {push $node_a$ to V}
  end if
until V is empty or reach the last frame in the clip

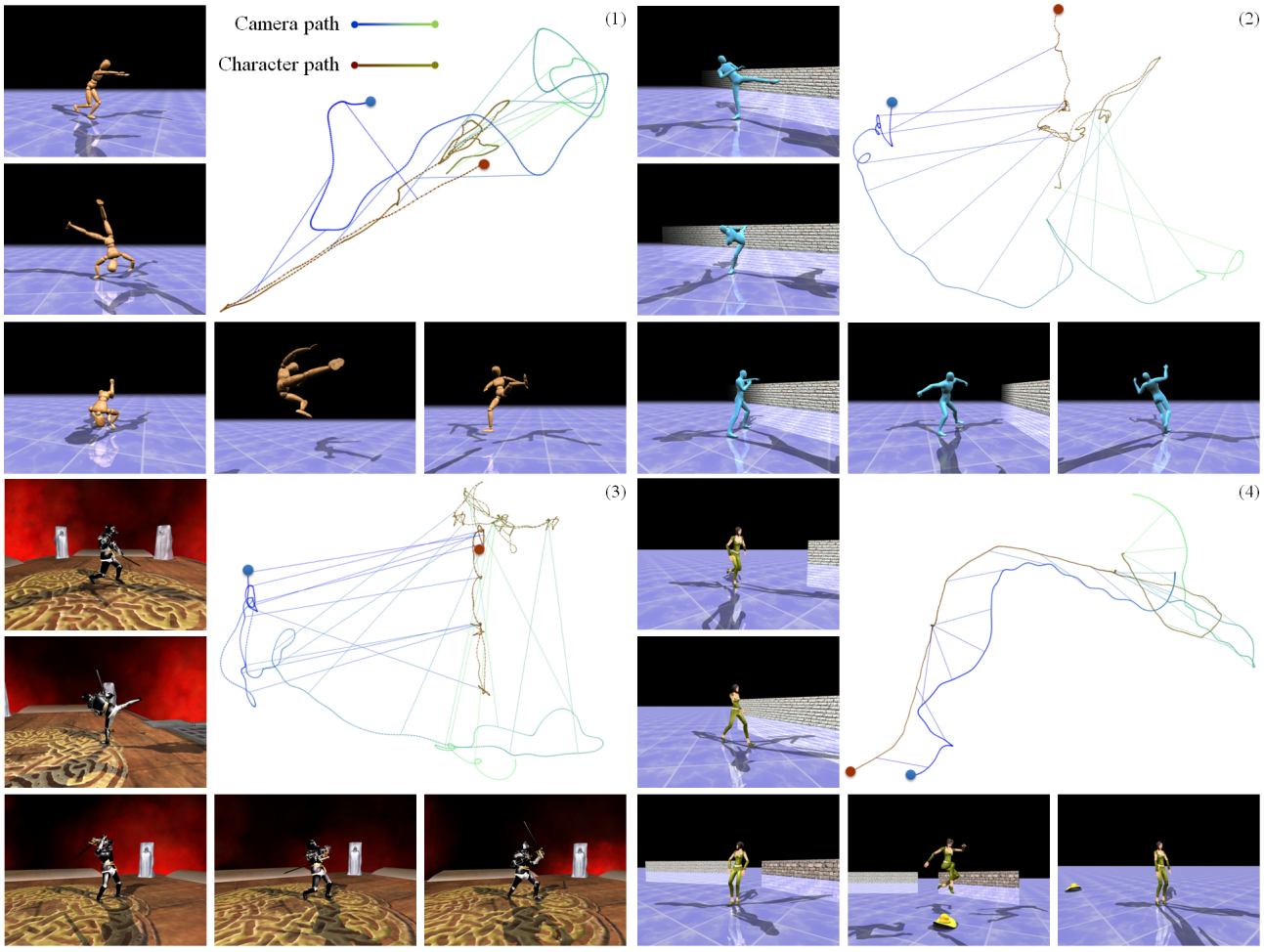


Figure 11: Results of the camera path planning (Example 1-4). Some viewpoints of the key-frames are shown at the side.