Mesh Trailer: optimizing camera paths for virtual environment navigation

Andrea Baldacci* CNR-ISTI
Daniele Bernabei† CNR-ISTI
Fabio Ganovelli‡ CNR-ISTI
Massimiliano Corsini§ CNR-ISTI
Roberto Scopigno ¶ CNR-ISTI

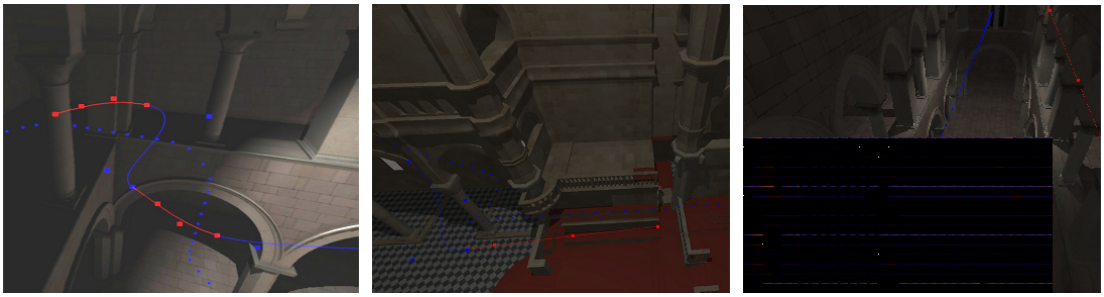

Figure 1: *Some examples of automatic path optimization*

Abstract

In recent years, the rapid growth in the use of 3d graphics brought to the creation of a very large number of 3d models, sometimes made available in large databases. This evolution has pushed research towards building *concise* presentations visualizing *salient* views of the 3d mesh to the user. This work contributes to this research field with a novel framework to build a cinematographic video presentation of a 3d scene. In our system, the user sketches the desired movements of the camera using common interfaces such as mouse and keyboard. The resulting shots are converted to a camera path using Bézier curves.Then, shots are edited to comply to cinematographic constraints and then the editing is performed, i.e. these shots are edited to assemble the final sequence. While in real filmmaking, the shots can only be cut, we allow more complex editing, i.e. changing the trajectory and the camera orientation. We mathematically define the the editing of camera movement as a constrained optimization problem where the unknowns are the modifications of the input shots, and the constraints are the total running time and the stylistic choices, and where the objective function is the difference of the *optical flow* from the original input movements.

1 Introduction

Our idea is to create an entire pipeline needed to create a video sequence for presenting a 3d scene. Starting from the input phase, the user uses the sketching functionality to interactively generate the desired camera path moving inside the 3d scene visualized with our previewer application. The result is a path expressed as a series of curves describing position, orientation and lens parameters of the camera. In the next step, an automatic editing of the path is performed to clean up imprecise camera movements derived from the sketching phase and to meet certain stylistic constraints that the user may impose. Multiple path exploring the same mesh can be generated. The film editing module aims to join the different path segmentes in a continuum, using typical transitions techniques used in the movie industry. This task requires two different tools. The first is the temporal optimization of the path, that is: modifing the temporal duration of the path preserving the sense of the camera motions. This simplification is typical in film editing as a director may want to cut part of the video segments to better mix the different segments together, to apply a retiming to certain movements or simply to eliminate unnecessary parts. The second tool necessary to compose the different video segments is the path cutting functionality that choose how to transition from one video segment to the next. At the end, the whole video sequence can be rendered using one of the many offline rendering engines available for free. In this work, we have developed only the subsequent modules: path sketching, path normalization, path editing and path simplification. We left the other modules for future works.

2 Path Sketching

The user can move around inside the scene using two common camera controllers: a trackball controller and a "wasd" controller our interactive scene exploration application The user can start recording his scene exploration just pressing the recording button. While the recording is active, the camera position, orientation and lens parameters are sampled at equally spaced time instants. The result is a sequence of samples describing the camera movements. Our aim is to calculate a picewise cubic Bézier curve that fits the sampled data and that descibes how the camera parameters vary with respect to time. We start considering that the camera parameters involved are:

$$\boldsymbol{f}_{camera}(u) = [P(u)\ \ Q(u)\ \ F_l(u)\ \ F_d(u)\ \ T(u)] \qquad (1)$$

where $[P_x\ P_y\ P_z]$ is a vector representing the position of the camera, $[Q_x\ Q_y\ Q_z\ Q_w]$ is a quaternion representing the orientation of the camera, F_l and F_d are respectively the focal length and the focal plane distance and, finally, T is the time instant in which the sample has been taken. Thus, a camera data sample is an eleven-dimensional vector. Instead of deriving an indipendent curve for each component, we obtain a picewise eleven-dimensional cubic Bézier curve. Our fitting method is borrowed directly from [Schneider 1990], which adaptively fits a piecewise cubic Bézier curve to the sampled data. As unit quaternions belong to the 4-sphere, a non Euclidean space, we pretrasform them to a four dimensional euclidean vector $\mathbb{R}^4$ using the rational mapping presented in [DeLoura 2001].

*e-mai:andrea.baldacci@isti.cnr.iti
†e-mai:daniele.bernabei@isti.cnr.iti
‡e-mai:fabio.ganovelli@isti.cnr.iti
§e-mai:massimiliano.corsini@isti.cnr.iti
¶e-mai:roberto.scopigno@isti.cnr.iti

3 Path Editing

The actual editing of the curves is performed with a series of *modifiers* that can act on a single curve or on group of curves. The modifiers purpose is to change the trajectory of the camera, its orientation and to reduce or to increase the path duration.

3.1 Trajectory modifier

Bézier curves can be deformed maintaing continuity and derivability by using a constrained optimization method. In particular, we can deform the camera path through a field of vectors which [Hilario et al. 2011] connect the Start points (points on the original curve), S_i $i = 1, ..., l$, with the Target points (points on the modified curve), T_i . The modified Bézier curve $S_\varepsilon(\alpha(t))$ is thus designed to pass through the target points. The trajectory deformation is calculated with a constrained optimization method where the objective is to minimize the changes of the shape minimizing the distance between the intitial Bézier curve and the modified one. The objective function is subject to a number of constraints which are exspressed using the Lagrange Multipliers theorem. Essentially, the system solution are the perturbations ε_i to be applied to each control point to achieve the desired result. Basically, the cost function to be optimized is:

$$\min_{\varepsilon} \int_{t_0}^{t_f} \|S_\varepsilon(\alpha(t)) - \alpha(t)\|^2 = \min_{\varepsilon} \int_{t_0}^{t_f} \left\| \sum_{i=1}^{n} \varepsilon_i \cdot B_{i,n}(t) \right\|^2 \quad (2)$$

The cost function in (2) must be extended to handle a set of k concatenated Bézier curves and continuity constraints must be imposed on the joint points of the concatenated curves. We remand to [Hilario et al. 2011] for an in depth treatment of this part.

3.2 Orientation modifier

The principle to change the orientation of the camera is the same as deforming in the tridimensional space: we pick n points along the quaternion interpolation curve (which lies on the hypershpere) and apply an offset to each, thus changing the orientation of the camera in each point. More pratically, we pick n time instants and for each we choose a target point representing the desired orientation at that time exspressed as a quaternion.

3.3 Speed modifier

The speed modifier allows to speed up or speed down the camera, or, more precisely, to scale the duration of a curve segment of our path. The first and last control points of each curve segments contains the time at which the camera enters and the time at which the camera leaves the same curve respectively. To scale the duration of a curve segment we have simply to scale the time values of the four control points by the same scale factor w_{speed} and then to adjust the time values of the neighbour curves to ensure speed continuity.

3.4 Trim modifier

An other way of reducing the path duration is to use the trim modifier which allows to cut off any given part of a path. Virtually, two time values t_{start} and t_{end} have to be chosen to define the interval that we want to preserve. Then the curve will be cut using the deCasteljau algorithm.

CAMERA ORIENTATION	CAMERA POSITION
CONSTANT	CONSTANT
TANGENT TO THE PATH	MOVING STRAIGHT
TARGETING A POINT	MOVING SMOOTHLY
SMOOTH	

Table 1: *Camera motions classification*

4 Cinematographic constraints

From the cinematografic point of view, we focused our attention mainly on three type of shots: the "steady-cam shot", the "tracking shot" and the "crane shot". First, our system analyzes the path sketched by the user and each curve composing the path is classified according to the type of movement performed by the user. Our classification is reported in table 1. When the user wants to simulate a "steady-cam" shot, usually he tries to move the camera while looking in the direction of movement. For simplicity, we call this movement: "tangent movement". After having recognized what the user intented to do, the motion is turned into a perfect "tangent movement" by adding a constraint to the camera orientation (using the orientation modifier explained in 3.2) . The other type of constraint, the "target constraint", is useful when the user simulates a shot similar to a "tracking shot" or to a "crane shot," where the camera moves while focusing on the same character or object. For this type of movement, the area approximately tracked by the user is automatically discoverd and the camera is retargeted precisely to that area.

5 Path Optimization

As introduced in section 1, the director can reduce or, eventually, even increase the duration of a camera path. The optimization reduces (or increases) the path duration by using the path modifiers presented in section 3. The algorithm converges to the duration value chosen by the user while preserving the "sense" of the camera motion. An effective characterization of the sense of camera motion is the optical flow of the 3D scene. The optical flow is measured at equidistant time intervals for the original path and compared to the same measurement made for the modified path, giving us the degree of similarity beetween the two paths.

To compare two flow fields, for each time sample chosen, two texture containing the optical flow vectors of the paths are compared. The direction and the magnitude of vectors in corresponding texel position are compared. The output is a texture having the same dimension as the input textures where each texel contains a normalized factor measuring the similarity of the vector pair corresponding to that texel. Given a pair of vectors V_{f1} V_{f2}, their normalized similarity factor S is computed as the weighted sum of the angular and magnitude differences:

$$S = W_d \cdot \left\langle \frac{V_{f1}}{\|V_{f1}\|}, \frac{V_{f2}}{\|V_{f2}\|} \right\rangle + (1 - W_d) \frac{\|V_{f1}\|}{\|V_{f2}\|} \ , \|V_{f2}\| \geq \|V_{f1}\| \quad (3)$$

Where W_d is a customizable weight factor. The function we want to minimize is the weigthed sum of the duration error with respect to the desired duration chosen by the user and the dissimilarity of the modified path with respect to the original one. More formally:

$$E = DurationError \cdot \alpha + FlowError \cdot (1 - \alpha) \quad (4)$$

The objective function (4) is dependent on a number of variables that control the path editing. Each variable automatically controls one of the tool presented in section (3). Tools can work on a sin-

gle curve of the picewiese curve describing our path, on groups of curves or on the whole path.

6 Conclusion and future works

Scene exploration and salient view selection are novel areas of research. This work contributes to this research field with a novel framework which introduces the cinematographic language and tools to generate the video presentation of a tridimensional static scene. In particular, the core of this work is the path optimization tool which is a novel instrument to edit the camerawork. This tool introduces a new idea of path optimization which takes into account the perceptive aspect of the camera movements using the optical flow. The framework we introduced opens up to a variety of extensions and studies. the most relevant extension would be to implement an automatic montage system which can assemble multiple video sequences to obtain the final movie.

References

DELOURA, M. 2001. *Game Programming Gems 2*. Charles River Media, Inc., Rockland, MA, USA.

HILARIO, L., MONTÉS, N., MORA, M. C., AND FALCÓ, A. 2011. Real-time bézier trajectory deformation for potential fields planning methods. In *IROS*, 1567–1572.

SCHNEIDER, P. J. 1990. Graphics gems. Academic Press Professional, Inc., San Diego, CA, USA, ch. An algorithm for automatically fitting digitized curves, 612–626.