Praise Worthy Prize

International Review on Computers and Software (IRECOS)

Vol 11, No 9 (2016)

Open Access Subscription or Fee Access

Table of Contents

Articles

2014-01-31

Seamless and Secure Design for Subsequent Handover in Mobile WiMAX Networks
H. F. Zmezm et al.
1376 views since: 2014-08-31

Texture Pattern Based Lung Nodule Detection (TPLND) Technique in CT Images
T. Kumar et al.
1130 views since: 2014-03-31

An Efficient Approach for Cancer Prediction Using Genomic Signal Processing
T. Inbamalar et al.
1062 views since: 2014-03-31

Survey and Analysis of Visual Secret Sharing Techniques
L. Anbarasi et al.
906 views since: 2014-09-30

Data Warehouse Development Based On Cloud Computing Using IBM Informix and IBM Cognos for Multifinance Industry
Dion Darmawan, Chrissandy Fernando, Andree Gunawan, Julian Ivandi

Application of Chaotic Encryption to Secure the Fingerprint Data
Ahmed Sabri, Mohamed Ouslim

Omitting Code Clones Based On Ranking the Called Clones
Shahenda Sarhan, Zyad Dhafer

Solving Complex Nurse Schedule Problem Using Particle Swarm Optimization
Mohamad Raziff Ramli

Building A Smart Identification System for Implementation of Motor Vehicles Subsidized Fuel Restriction
Indrastanti Ratna Widiasari, T. Arie Setiawan Prasida, Dian W. Chandra

---

Please send any questions about this web site to info@praiseworthyprize.com

Editorial Board

Editor-in-Chief

Professor Marios C Angelides, Brunel University London Department of Electronic and Computer Engineering, United Kingdom

Co-Editor

Dr. Harry Agius, Brunel University London Department of Electronic and Computer Engineering, United Kingdom

Editorial Board members

Francoise BALMAS, Universite Paris 8 Dept. Informatique, France

Vijay BHATKAR, International Institute of Information Technology, India

Arndt BODE, Technischen Universit, Germany

Wojciech CELLARY, The Poznan University of Economics - Department of Information Technology, Poland

Bernard COURTOIS, ENSIMAG - École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble - TIMA Laboratory, France

Damon DAYLAMANI-ZAD, Computing and Information Systems Department, Faculty of Architecture, Computing and Humanities, University of Greenwich, Old Royal Naval College, United Kingdom

Andre DE CARVALHO, University of Sao Paulo, Brazil

David Dagan FENG, University of Sydney - School of Information Technologies and Hong Kong Polytechnic University - Department of Electronic and Information Engineering, Australia

Ingo FROMMHOLZ, Department of Computer Science and Technology, University of Bedfordshire, University Square, United Kingdom

Borko FURHT, Florida Atlantic University - Department of Computer & Electrical Engineering and Computer Science, United States

David C. GIBBON, AT&T Labs - Research, United States

Peng GONG, University of California Berkeley - Division of Ecosystem Sciences Co-Director of CAMFER, United States

Defa HU, Hunan University of Commerce - School of Information, China

Michael N. HUHNS, University of South Carolina - Department of Computer Science, United States

Tom JACKSON, Centre for Information Management, School of Business and Economics, Loughborough University, United Kingdom

Ismail KHALIL, Johannes Kepler University Linz - Institute of Telecooperation, Austria

Panos KUDUMAKIS, Queen Mary, University of London, United Kingdom

Catalina M. LLADÓ, Universitat de les Illes Balears - Departament de Matemàtiques i Informàtica, Spain

Martín LÓPEZ-NORES, University of Vigo, Spain

Pascal LORENZ, Universite de Haute Alsace IUT de Colmar, France

Arthur G. MONEY, Department of Computer Science, College of Engineering, Design and Physical Sciences, Brunel University London, United Kingdom

Ali MOVAGHAR, Sharif University of Technology- Computer Engineering Department, Iran, Islamic Republic of

Fionn MURTAGH, University of Derby, College of Engineering and Technology, United Kingdom

Dimitris NIKOLOS, University of Patras - Computer Engineering & Informatics Departement, Greece

Ioannis PARASKEVOPOULOS, Department of Computing and Information Systems, University of Greenwich, United Kingdom

Witold PEDRYCZ, University of Alberta - Department of Electrical and Computer Engineering, Canada

Dana PETCU, Western University of Timisoara - Computer Science Department and Director of Institute e-Austria, Romania

Erich SCHIKUTA, Universität Wien - Department of Knowledge and Business Engineering, Austria

Christos SCHIZAS, University of Cyprus- Department of Computer Science, Cyprus

Arun K. SOMANI, Iowa State University - Department of Electrical and Computer Engineering, United States

Miroslav ŠVÉDA, Brno University of Technology, Czech Republic

Daniel THALMANN, EPFL - The Virtual Reality Lab (VRlab), Switzerland

Brijesh VERMA, Central Queensland University, Australia

Lipo WANG, Nanyang Technological University - School of Electrical and Electronic Engineering, Singapore

International Review on Computers and Software (I.RE.CO.S.), Vol. 11, N. 9
ISSN 1828-6003 September 2016

Virtual Camera Movement with Particle Swarm Optimization and Local Regression

D. A. Prima[1, 2], M. Hariadi[1], I. K. E. Purnama[1], T. Usagawa[3]

Abstract – *Manually placing and animating virtual camera in a dynamic virtual environment to generate computer animation is a complex and time-consuming process. For this reason, this paper proposes a method to take the sequential shot by gameplay, to determine the best camera placement candidates as an unorganized point set, to reconstruct a curve by the point set using local regression technique and to generate smooth camera movement based on those curves. Finally, the paper presents the application of this algorithm and it analyzes a number of problems in different dynamic virtual environments.*

Keywords*: Dynamic Environments, Gameplay, Virtual Camera, Virtual Cinematography*

Nomenclature

α	Desired horizontal angle between camera and target's front vector
β	Desired vertical angle between camera and target's front vector
γ	Object visibility
γ_c	Current visibility of target object
γ_d	Desired visibility of target object
ε_i	Error term representing random variable of y_i variation around x_i
θ	Vantage angles
σ	Object projection size
σ_c	Desired projection size
σ_d	Current projection size
$C_{x,y,z}$	Normalized camera position transformed in targeted relative space
$d(x)$	Distance along the abscissa from x to the most distant value within the data span
e_i	The list containing the position of the four extreme vertices in the field of view
m	Regression function that describes the data
N	The number of vertices
$S_{x,y}$	Screen width and height
$T_{x,y}$	Projected width and height of target object's proxy geometry
v_i	The position if the i^{th} vertex of the object mesh
w_i	Regression weight
x	Predictor value
x_i	Nearest neighbor of x defined by the data span
y_i	Independent variable of the i^{th} of the data

I. Introduction

The technology of computer animation and 3D game has reached a point where the level of realism is extremely high and the history of animation film has shown the importance of cinematography for a better engagement with the audience. Cinematography plays an important role for the audience because it determines how the audiences perceive it and it gives a way to the directors to express their ideas to the audiences through their films. Audiences have become accustomed to certain conventions in the movie, and they expect these conventions in any form of visual entertainment.

While making an animation film, placing and animating the camera are done by the animator or exclusively by the director of photography. However, manually placing the virtual camera within the virtual environment requires lots of modeling and is a time-consuming effort that must be repeated for each scene [1]. Cinematography means lights and camera setting, and it can be described as the art of storytelling.

In an interactive virtual world, the story is developed almost immediately. For this reason, in accordance to the scene elements, common phrases in cinematography, including framing, composition, motion, etc., can become too complex to be handled by the automatic camera control in a virtual environment. A film sequence is composed by sequential shots. To compose the visual properties of a shot, the cinematographer may change the size of a subject in the frame, the relative angle or height between the camera and subject, and the lights composition. To make an artful effect for the film, some editing decisions such as shot duration and the cut frequency, can be used by the filmmaker [2].

To make an animation film in the virtual world, we Virtual cameras can be used. They are very similar to cameras used in the films, with the only difference that virtual cameras are used in virtual world [3]. In general, basic virtual camera control problems are how to determine where camera should look (camera planning), how the camera moves (camera controlling), and which camera should take the shot (camera selecting) [4].

 DOI: 10.15866/irecos.v11i9.9801

A similar research has been attempted previously, although for different domains and goals. Virtual camera control can be applied to several applications like, airplane emergency simulation, to create cinematic replay of computer games [5] and [6], to create fly-through animation for 3D architectural models [7], to create stereoscopic photography in virtual environment [8], combined with UAV to for automatic cinematography and to capture cinema scene [9], [10], and [11]. Drucker *et al.* [12] presented constraint-based for camera control in the virtual environment. He *et al.* [13] encoded an expertise knowledge by the filmmaker into hierarchical finite state machine for virtual camera to automatically generate camera control for individual shots. Bares *et al.* [14] proposed a narrative generation camera planning for filming in the interactive world. Bares *et al.* [15] used recursive heuristic constraint solver to determine camera parameter.

Elson and Riedl [16] proposed a camBot to automatically select the camera angle for shooting a priori unknown script. Burelli *et al.* [17] used PSO approach for capturing a static scene as in comparison from [15] works. Lima *et al.* [18] used SVM trained with cinematography knowledge to select the best shot for creating dramatization effect of the scene. Burelli *et al.* [19] employed artificial potential fields to control both camera location and aim direction. Markowitz *et al.* [20] presented a successful attempt to give the game designer a more robust way to customize the portrayal of the game and to give the user the ability to create complex camera transitions to provide a more engaging game experience. Galvane *et al.* [21] used steering behavior approach to control collection of camera agents. Prima *et al.* [22] used a behavior trees approach for placing the secondary camera in machinima based animation. Ranon *et al.* [23] used an improved version of PSO by tuning the parameter as in comparison with [17] works. Burelli *et al.* [5] proposed a multi-objective approach for generating replay in the game. Burelli *et al.* [24] proposed benchmark methods for virtual camera control.

Galvane *et al.* [25] proposed virtual camera rail based on initial framing and last frame to determine camera movement. Christianson *et al.* [26] described some of the established cinematographic techniques. [27] implemented a method for various camera movements (pans, tilts, rolls, and dollies). [28] built video sequence for documentaries from images using only wheeled type camera movements. Friedman *et al.* [29] used automated reasoning applying a cinematic knowledge to determine camera behavior.

As mentioned in [25], the computation of one optimal camera position per frame leads to the creation of jerky and overly reactive camera. Proposed methods in [17] and [23] proved to be efficient and used only for the static scene. In this paper, the proposed system is an attempt to address the main shortcomings of the above approaches so to avoid jerky and overly reactive camera motion while using PSO to solve camera placement and configuration in the dynamic virtual scene.

The contribution of this paper consists in presenting a method to compute the best optimal camera position and orientation for the overall motion of target character and to compute smooth camera movement constrained on given smoothed position and orientation, taking into account the changes in elevation of the character as it moves in a scene. This method combines the computation of one optimal camera position per frame and smoothing method to account both aesthetic constraints and constraint on the camera.

The paper is organized as follows. Section 2 discusses previous related works. Section 3 describes an overview of the proposed approach. Section 4 presents the experimental results. Finally, Section 5 concludes the paper giving some perspectives for future work.

II. Related Works

Several researches and approaches have been done previously to deal with the automation of cinematography process in computer games and to generate 3D computer animation. The scope of these researches is to reduce time, the cost required, to improve the quality, and to simplify the virtual camera control.

An early work by Drucker and Zeltzer [12] with the optimization approach to satisfy a list of constraints, presented camera control framework to explore the 3D virtual environment (museum). Constraints on the camera parameter were based on analysis of the task required in the museum. Generic camera module by [12] consists by a controller to translate user input into camera state or into constraints, in an initializer that runs upon the activation of a module, in a constraints list to be satisfied while the module is active, and in a local state containing camera position, camera view, camera up vector and camera field of view. This camera module also has camera path planning that implements A* algorithm to calculate the path from start to the destination point.

A* algorithm was also used by [30] to navigate the virtual camera to follow the character movement through the virtual environment. The virtual environment was built using the Unreal engine. The pathfinding algorithm was implemented after the creation of navigation mesh.

He *et al*. [13], demonstrated the virtual camera control in the context of a "virtual party". Their system consists by three parts: the first part is the real-time application, the second part is the virtual cinematographer (VC) modeled into a hierarchical finite state machine, and the third part is the renderer. The VC sent queries to the real-time application, and then VC received events and geometric information from the real-time application in return. Renderer received the camera and acting hints from VC, and it also received animation parameters, static geometry, actor models, lights, etc. The VC had the camera module to implement different camera placements and idioms used to combine shots into sequences.

Bares *et al*. [14] proposed a constraint -based camera planning in the interactive virtual world. The viewer can manipulate the narrative by changing the character eyesight range and its speed which will affect the cinematic goal selector. The characters and or object needed to be filmed are informed to cinematography planner by cinematic goal planner. After that cinematography planner will formulate cinematic constraint problem that will be solved by the constraint solver.

Another work by Bares *et al*. [15] employed a heuristic constraint solver to compute camera shot that closely matched the desired shot given by the user using the storyboard frame. Their camera system supported fifteen types of constraints. The constraint could be applied to one or two specified objects or to the camera itself.

Elson and Riedl [16] worked on Cambot, a not real-time camera planning application that functioned as virtual director for offline machinima production. Cambot used script, character blocking, possible shot composition, and reel as an input parameter. Cambot had four dimensions of constraints: location, blocking, view, and scene constraint.

Burelli *et al*. [17] proposed an approach for the virtual camera problem using *Particle Swarm Optimization* (hereinafter, PSO) for a static scene. It can optimize all the camera parameters inside a feasible volume. Such parameters consist of three position coordinates, three orientation angles, and one field of view angle. Occlusion check is done by raycasting from the camera to the bounding box of the object and testing the ray intersection with other objects in the scene.

Barry and Ross [31] used the Multiobjective PSO algorithm for virtual photography. They used PSO for satisfying the supplied set of aesthetic rules, consisting of four constraints: the rule of the third, the horizon line, object of interest, and color palette.

Support vector machine (SVM) was used by Lima *et al*. [18] for real-time camera control in storytelling environments. SVM was trained with cinematography knowledge to solve problems of shot selection. The input of their SVM was the environment, scene, and actor important features. To train SVM, based on cinematographic rules and principles, they simulated a real film situation. Their system consists of the following modules: the scriptwriter, the scenographer, the cameraman and the director that communicate with each other.

Burelli and Jhala [19] extended to address the camera path planning within virtual environments by modeling both camera movement and orientation parameter with the multiple Artificial Potential Fields (APF). Three constraints are supported by their system, visibility, projection size, and view angle. The implementation of APF suffers of a local minimum problem for the complex environment. An improvement of APF by prioritizing the frame constraint and calculating the initial position based on the first vantage angle or relative position, enforcing the frame coherence and consisting in interpolating the actor trajectory, and dynamically tuning the weights of frame constraint in the objective function was proposed by Burelli [32].

Markowitz *et al*. [20] used the behavior tree to describe how a single camera behaves like cinematographer assigned by smart events. Behavioral response and parameter information inside the event are stored in "smart events". The cinematic shot idioms are chosen among the smart events. The behavior tree was also used by [22] for placing the secondary camera in the virtual environment for a 3D fighting game type.

Another work by Galvane *et al*. [21] used the steering behavior for controlling multiple cameras as agents.

Their camera agents have two steering behaviors, the scouting behavior to search the scene for events and tracking behavior to follow an event. The computation of events is based on the orientation, position, speed, and type of the characters. They designed five camera steering forces (framing, obstacle avoidance, camera separation, wandering, and containing force) to control the camera position and three camera steering torques (aiming, avoidance, and wandering torques) to control the camera rotation.

The qualitative evaluation is measured by using *activity* metric that shows overall active character compared to the inactive characters.

An improvement of PSO was proposed by Ranon *et al*. [23]. The test scenario was a static virtual environment consisting by a city, a house, and a room. The algorithm they proposed had the aim to overcome some limitations in virtual camera control like very specific and limited kind of objective or to find suitable viewpoints in the complex scenario in real world application. To minimize the computation time, the number of particle initialization was divided into two sets of the particle (random and non-random) so at least one property satisfaction could be obtained.

For the dynamic scene, for example a cinematographic replay for a 3D shooter game, a multi-objective approach was proposed by Burelli and Preuss [5]. They prefered the S-Metric Selection Evolutionary Multiobjective Optimization Algorithm (SMS-EMOA) than NonDominated Sorting Algorithm-II because it is known to deal with a higher number of objectives. The performance was measured using average best fitness value over the objectives and time steps. The performance of their approach could be improved by tuning or selecting more suitable algorithms.

A work by Fourati *et al*. [33] proposed virtual camera control to visualize stimuli human body behaviors. In their work, the target for camera orientation approximation was based on the pelvis position of the avatar. The path of the virtual camera was also based on Hip and the pelvis of the avatar.

Sanokho *et al*. [34] maintained the real camera path involving to extract real data from the movies or motion capture and then retargeted the path into a new 3d environment.

Harris corner detector was used to reconstruct the camera trajectory from the sequence of images.

Lai *et al.* [35], designed a pattern-based representation of discourse to encode communicative goals and camera techniques. The patterns were composed by the shot database annotating the shot from real movies, the camera patterns comprised of specific shot types or transitions, and the virtual director selecting the best shot for the scenario based on the current action, the involved character and the communicative goal.

A works by Dib *et al*. [36], explored the effect of perspective view in educational animations of building construction management tasks by comparing the egocentric perspective view (first person view) and the exocentric perspective view (third person view).

To create a camera behavior predictor, parameters about the players' gaze, camera movement, players' actions and game context were recorded and then the resulting data identifying different virtual camera behavior pattern were analyzed by using machine learning [37].

An approach using Hidden Markov Model to reproduce elements of style extracted from real movies was proposed by [38]. Their work consisted in extracting cinematographic elements, shot transitions, and the relation between shot description and action.

A benchmark for virtual camera control was proposed by Burelli and Yannakakis [24] by measuring the accuracy (how close the best camera configuration is found by the algorithm), reliability (how often the algorithm succeeds to provide an acceptable solution), and initial convergence time (how much time it takes the algorithm to converge to an optimal camera configuration). Three test environment (a forest, a house, and a rocky clearing) were selected to evaluate levels of complexity. The environments were composed only by a static structure, but the test problem included either one, two or three moving subjects. Galvane *et al*. [25] proposed the computation of camera rail by taking the initial and the final framing specified by the user. The framing consisted by on screen desired positions, on screen sizes and vantage angles. The test scene involved one moving character, two static characters, and two moving characters. The problem was solved using an optimization process which took a shot duration and length of the rail as the inputs. Hu and Zhang [39] proposed a semi-automatically virtual camera control by encapsulating camera movement path which is defined by a series of nodes, camera look target move path, and camera properties change path (camera field of view and clipping plane) into a game object in Unity game engine.

Camera movement path and camera look at target move path are manually placed by the user in the game scene. Chen and Li [40], designed the scripting language and they implemented a camera planning system to generate camera configuration according to scenario descriptions. They used a hierarchical decision tree to determine how to map the story context into the camera configuration.

Their system had two modules, event generator module preparing the action events and cinematographer module to determine how to take the shot by the given story context.

Hsu *et al*. [41], presented a camera motion design for volume data animations. Their work involved generating road map by the volume data representation, generating an initial path and dynamic path refinement. The roadmap was constructed using medial axis. The initial path consisted by a linear segment connecting source and destination point derived from the road map.

Preuss *et al*. [42], presented a virtual camera composition based on niching and restarting the evolutionary algorithm. The camera composition properties consisted in object visibility, object projection size, object view angle, and object frame position. The aim of their work was to assess that niching and restarting the based algorithm can provide a good solution to the problem, investigating the tradeoff between diversity and quality, and collecting experimentally based knowledge about the landscape structure to fasten the algorithm implementation.

One of the issue in virtual camera control is collision avoidance. The work by Christie [43] attempted to address this issue, by using the principle of shadow maps to evaluate the visibility of a single or multiple targets within specified search space. The process is divided into four steps: the first step is to compute the visibility volume; the second step is to render from both viewpoints using depth map and symmetric frustum; the third step is to combine the visibility information, by expressing the frustums intersection as 3D trilinear coordinate system; the last step is to move and choose the next camera position by providing parameters that could capture the camera's behavior in a common situation.

A physic based model for the virtual camera was proposed by Lixandru and Zordan [44] for tracking a moving character in the interactive game scenario. The system consisted by a motorized pan-tilt head and an active suspension system. They employed a combination of a proportional-derivative servo and a feedforward component to keep the target in sight when it is moving.

To avoid jerking camera motion caused by a discontinuous value of torques, they implemented feedforward with a linear ramp.

Erdem and Halıcı [45] proposed to evaluate an aesthetic aspect of rendered game scene for directing the virtual camera in real-time while the game is being played by using regression technique. The evaluation of visual aesthetics is based on color, lighting and exposure, composition, figure-ground separation, and texture and sharpness.

III. Research Method

This paper focuses on the automated computation of smooth camera movement during the motion of target character and the specified set of camera constraints.

The target character is specified on one, two and three characters moving in a scene.

The proposed camera system method is divided into three consecutive steps: (i) to determine the best camera position during the motion of target character(s), (ii) to compute the smoothed position of the camera from steps (i) and to compute the appropriate camera orientation along the smoothed position, (iii) to move the camera along the smoothed position.

III.1. Determine Best Camera Position

To determine the best camera position, the input of this process is the motion of the character along the scene as well as the user specified framings constraints, the number of the frame to be computed and a total number of character(s). These framings consist of vantage angles, object projection size, and object visibility.

As in [24], vantage angles (1), object projection size (2) and object visibility (3) as the satisfaction function can be defined respectively as follows:

$$f_\theta = f_\alpha \cdot f_\beta$$
$$f_\alpha = 1 - \frac{-arctan(\frac{C_x}{C_z}) - \alpha}{180}$$
$$f_\beta = 1 - \frac{|arcsin(C_y) - \beta|}{180} \quad (1)$$

where α is the desired horizontal angle between camera and target's front vector, β is the desired vertical angle between camera and target's front vector, and C is the normalized camera position transformed in targeted relative space:

$$f_\sigma = 1 - |\sigma_c - \sigma_d|$$
$$\sigma_c \begin{cases} {}^{T_y}/_{S_y} \ if \ {}^{T_y}/_{S_y} > {}^{T_x}/_{S_x} \\ {}^{T_x}/_{S_X} \quad Otherwise \end{cases} \quad (2)$$

where σ_c is the desired projection size, S_y and S_X are the screen height and screen width respectively and T_y and T_x are the projected height and width of the target object's proxy geometry respectively:

$$f_\gamma = 1 - |\gamma_d - \gamma_c|$$
$$\gamma_c = \frac{\sum_{i=1}^{N} infov(v_i)}{N} \frac{\sum_{i=1}^{5}(1 - occ(e_i))}{5}$$
$$infov(x) = \begin{cases} 1 & if\ x\ is\ in\ the\ frustum, \\ 0 & otherwise \end{cases} \quad (3)$$
$$occ(x) = \begin{cases} 1 & if\ x\ is\ occluded, \\ 0 & otherwise \end{cases}$$

where γ_c is the current visibility of target object, γ_d is the desired visibility, v_i is the position if the i^{th} vertex of the object mesh, N is the number of vertices, $infov(x)$ calculates whether a point is included in the field of view, and $occ(x)$ is to calculate whether point x is occluded by another object or not. The present research relies on the optimization method used by Burelli *et al.* [17], *Particle Swarm Optimization (PSO)* that provides an efficient implementation to compute actual camera configuration by these framings constraints specifications, but different optimization techniques can be used to solve this problem. In the presented case, *PSO* search space for virtual camera composition consists by seven parameters: three camera coordinate for the position, three camera coordinate for the orientation, and the camera field of view.

The other considered parameters such as aspect ratio and planes (both near and far planes) have a fixed value. *PSO* [46] is a simple computationally efficient optimization technique based on the social-psychological model of social influence and social learning. Because a standard animation required 24 or 30 frames per second, the number of the frame to be computed is needed. Some works require 60 frames per second to produce animation or fast pace game typed. An higher frame rate used will make more smooth animations, but it would make the render time becoming longer, hence this paper used 30 frames per second to composite an animation. So if the animation to be made has a duration of 5 seconds, then the number of frames needed to be calculated will be 150 frames. During the character motion in a scene, for every one second, the proposed system will perform 30 times *PSO* computation to compute raw camera position and configuration. The system will continuously compute the raw camera position and camera configuration as long as there is a change of position, orientation, and direction, if the there is no change in position, orientation and direction of a character, the system will use the last best camera configuration provided by *PSO*.

Fig. 1 illustrates the computation of the best camera position as raw data during the character motion in a scene computed by using *PSO*.

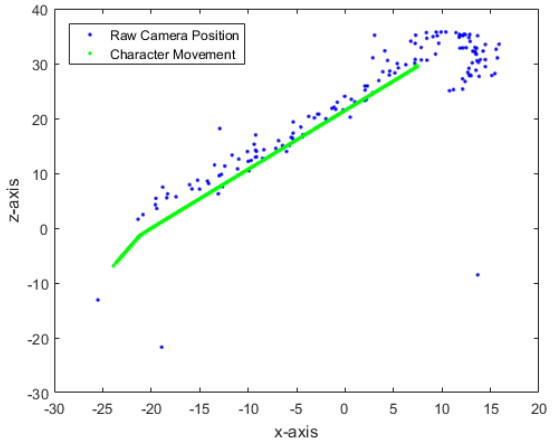


Fig. 1. An example of best camera position (blue dots) computed in the case of single moving character (green line) seen from the top view

The blue dots represent the best camera position during character motion, which later will be processed to compute smooth camera movement, and the green line represents the character movement in a scene.

III.2. Computing Smooth Camera Position

To compute the smooth camera position by the given best camera position, the current study proposes to use the local regression method, because of its simplicity and because it does not require the smoothness and regularity required by other methods.

The model of local regression can be defined as:

$$y_i = m(x_i) + \varepsilon_i \tag{4}$$

where $m(x_i)$ is the regression function that describes the data and ε_i is an error term representing random variable of y_i variation around x_i. Then it is needed to compute the regression weight for each data point. To compute local regression, data span and regression weight should be defined.

In particular, regression weight can be defined as:

$$w_i = \left(1 - \left|\frac{x - x_i}{d(x)}\right|^3\right)^3 \tag{5}$$

where x is the predictor value, x_i are the nearest neighbor of x defined by the data span, and $d(x)$ is the distance along the abscissa from x to the most distant value within the data span. In order to smooth the data, first it is necessary to do a regression on y coordinate as a function of x coordinate ignoring z coordinate.

The next step will be to do a regression on z coordinate as a function of x coordinate ignoring y coordinate, and the last step is to combine them as single line in 3D coordinate, putting the x coordinate in a privileged position respect to other two coordinates (y) and (z). Figs. 2 show the local regression on z coordinate as a function of x coordinate ignoring y coordinate applied to raw camera position given by *PSO* with some different data span. When different values are given to the data span, the regression results will also be different.

Figs. 3 show the local regression on y coordinate as a function of x coordinate ignoring z coordinate applied to raw camera position given by *PSO*. After the smooth camera position is computed, then the results will be used as fixed parameter values for three camera position constraint mentioned before, for the next optimization process.

From the conducted experiments, it is possible to suggest to use a data span of 0.5 to provide the best result for smoothing the raw camera position. Fig. 4 shows the combined coordinate as a single line in 3D coordinate, where the x coordinate is in a privileged position respect to the other two coordinates (y) and (z).

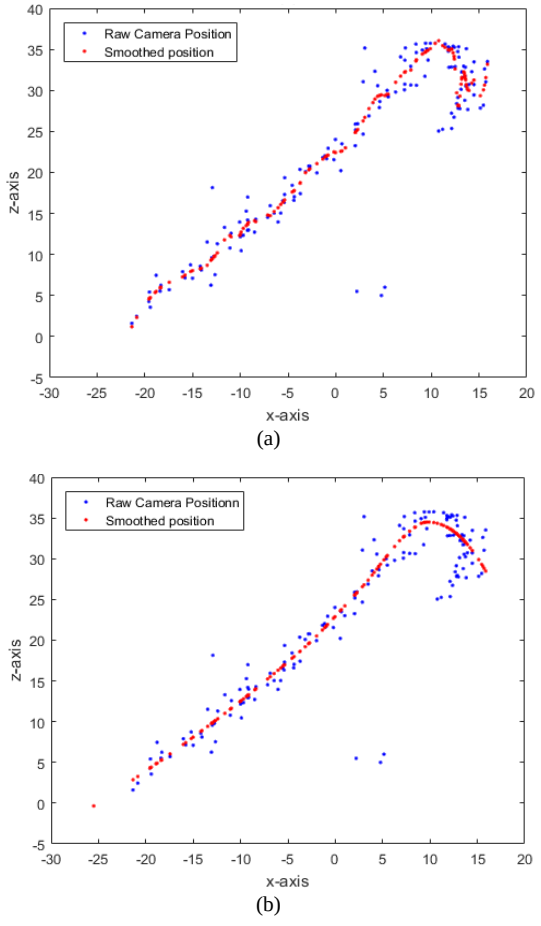
The next step is the curve fitting to create a smooth camera path along the smoothed camera position.

The common curve fitting is B-spline model and quadratic Bezier curve. Figs. 5 show the spline model curve fitting applied to smoothed camera position.

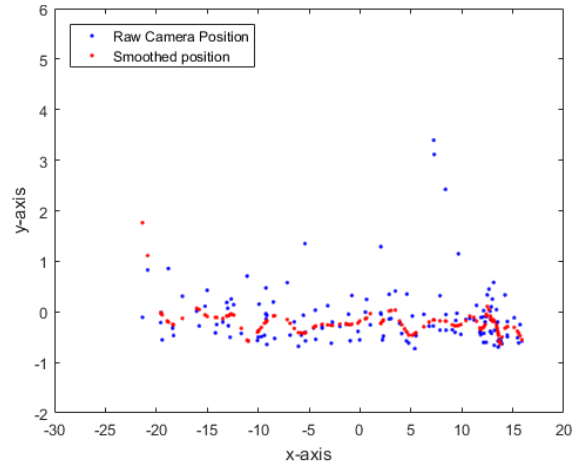
III.3. Camera Movement

To move the camera along the smoothed position while maintaining an optimal framing satisfaction over target character, it was necessary to try each time step (in the presented case each time step is 1 seconds over 30 frame), to position the camera on the smoothed camera position and then the camera dynamically tracks the character motion along this smoothed camera position.

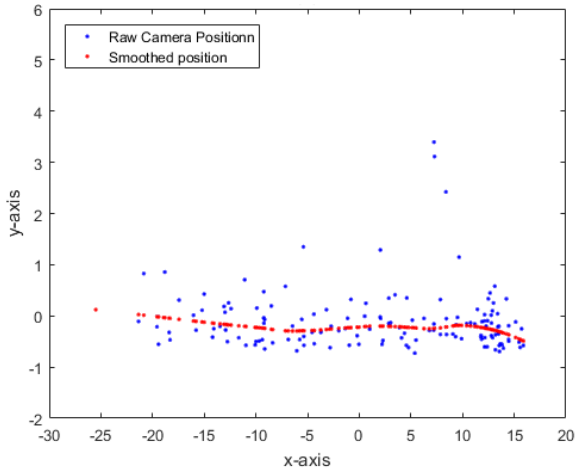
Camera speed and camera velocity will be relative to character speed and velocity. The proposed system consists of finding the camera position on smoothed curve each and move the camera along the curve for each time step. The presented optimization process then subjects to find the best camera orientation (*i.e.* pan, tilt, and roll) along the smoothed camera position, so the camera position will always belong to the smoothed position. Because the camera speed and camera velocity will be relative to the character speed and velocity it can be assumed that when the character is not moving, the camera speed and velocity will be zero.



Figs. 2. Examples of smooth camera position (red dots) computed to approximate raw camera position from *PSO* (blue dots) in the case of single moving character, (a) using data span 0.1 and (b) using data span 0.5



(a)



(b)

Figs. 3. Examples of smooth camera position (red dots) computed to approximate raw camera position from *PSO* (blue dots) in the case of single moving character, (a) using data span 0.1 and (b) using data span 0.5

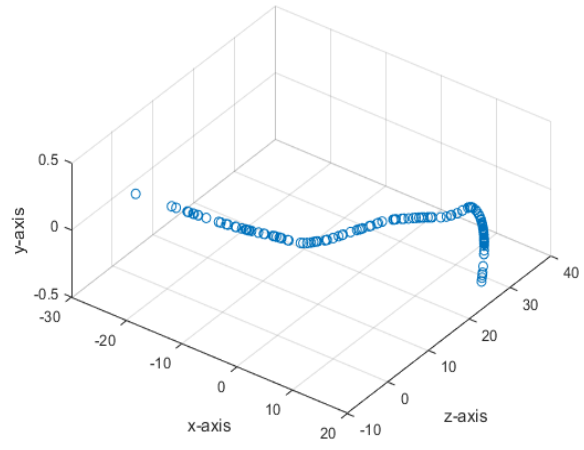
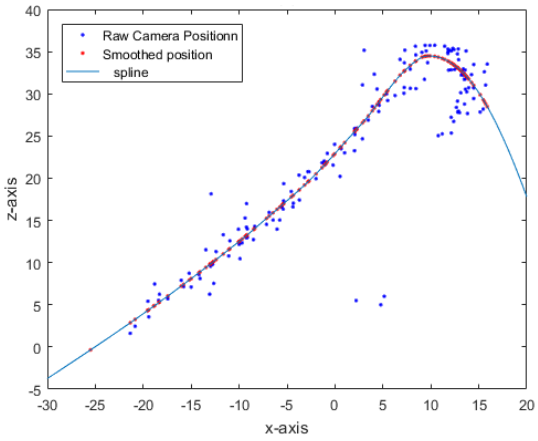


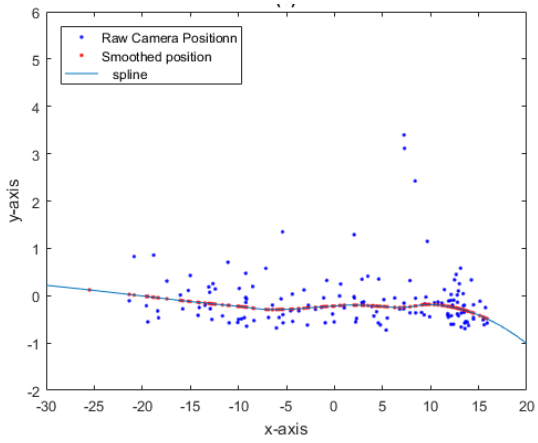
Fig. 4. The smoothed camera position in 3D coordinate

If the character starts to move the camera will accelerate until the character speed becomes constant.

When the character is approaching the destination point, the character speed will be decreasing, and camera speed will also be decreasing until the character stops.



(a)



(b)

Figs. 5. Spline model curve fitting applied to smoothed camera position (a) local regression on z coordinate and (b) local regression on y coordinate

IV. Implementation and Experimental Results

This section presents an analysis of the performance of the proposed system on a number of scenes and scenarios. To maximize the accuracy at the expenses of performance, the original scene mesh was used, instead of using simple or low polygon mesh.

For comparison results, all the experiments were run on a Core i7-3630QM @2.4Ghz, 16 GB of RAM, and NVidia GeForce GT 650m with 2GB video memory and a Core i7-4790 @3.2Ghz, 16 GB of RAM, and NVidia GeForce GTX 980 with 4GB video memory. The *PSO* was configured to use 30 particles in each iteration, 0.5 as c_1 and c_2 values, and 1.2 to 0.2 linearly decrease as inertia weight value.

IV.1. Scene Setup

In this experimentation, three test environment in the 3D model were selected. They include two outdoor scenes and one indoor scene with different geometrical characteristics: a forest, a warehousing complex, and a restaurant.

All the scenes are challenged with a different level of complexity. Table I presents the complexity measures about the scene. In each scene, the character is positioned and moves around the environment. Each scene will be tested using one, two and three characters. The camera will track 5 seconds of character motion. The characters are shaped like human beings, have human-like proportion and height as illustrated in Fig. 6.

Detailed character triangles and vertex measure are shown in Table II. Figs. 7 show the *forest* scene used in this experiment. The *forest* environment is a virtual outdoor environment in 3D composed of a cluster of trees, small plant, and some hut. The characters are placed and move around these trees that act as occluders and obstacles. Figs. 8 show the *warehouse* scene used in this experiment. The *warehouse* environment is a virtual outdoor environment in 3D composed by block of buildings, solid wall, and stairs that can be used by the character. Figs. 9 show the *restaurant* scene used in this experiment. The *restaurant* environment is a virtual indoor environment in 3D composed by a closed space separated by wall, solid beams, open doors, transparent window, tables, chairs, lamp and many detailed small items acting as occluders. The outside of the restaurant is composed by some trees, a street lamp and a hydrant.

The environment or this scene is composed only by static objects. The test problems include one, two and three moving characters. For the forest scene, the character will move from the center part of the scene (center hut) towards the forest hut at the southeast area of the scene. For the warehouse scene, the character will move from the center part of the scene towards the southwest building; this movement includes the character to use the stair, and in the last scene, the restaurant scene, the character will move from inside the restaurant heading out to the southwest area near the street.

TABLE I
COMPLEXITY MEASURES OF THE SCENE

Scene	Objects	Triangles	Vertices
Forest	562	1.1 M	1.4 M
Warehouse	470	537.3 K	787.9 K
Restaurant	765	1.1 M	1.7 M

TABLE II
CHARACTERS TRIS AND VERTICES

Characters	Triangles	Vertices
Young Male	12.729	13.942
Young Female	12.434	12.842
Old Male	11.995	8102

Fig. 6. Characters used in the experiment

(a)

(b)

Figs. 7. Forest Scene, (a) top view, (b) perspective view.

(a)

(b)

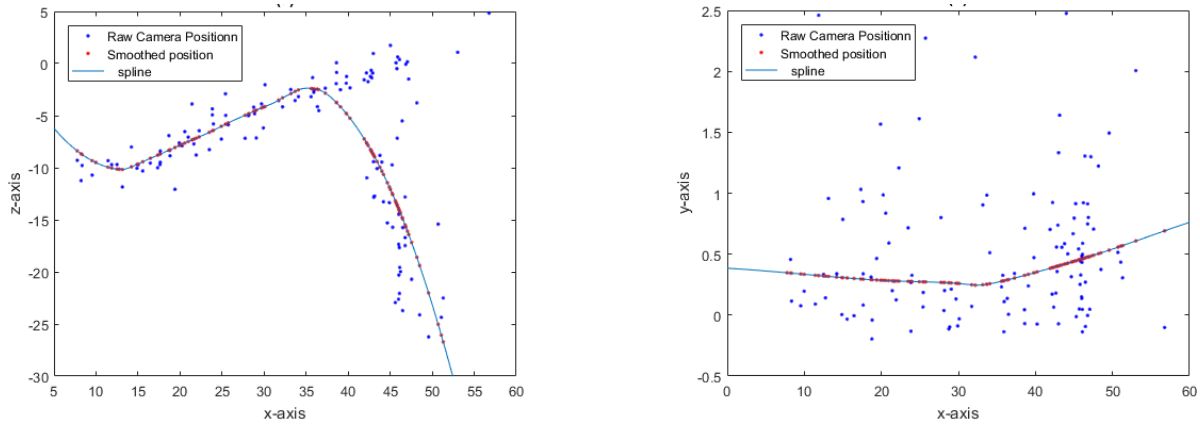
Figs. 8. Warehouse Scene, (a) top view, (b) perspective view

IV.2. Scene 1 – Forest

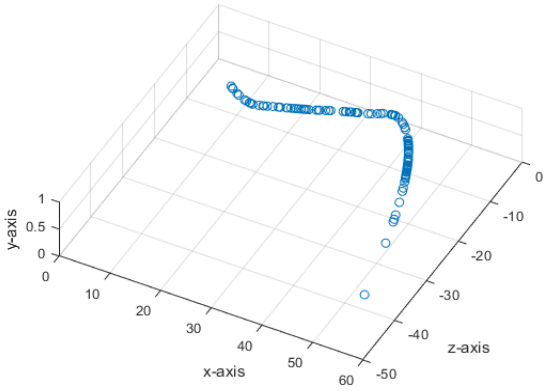
In the first experiment, the camera tracks the single character as he moves in the environment for 5 seconds from the center of the scene to the southeast area of the scene. In the second and third experiments, the camera tracks two and three characters with the same direction as before. Figs. 10 show the best camera position computed using *PSO*, the smoothed position and the curve fitting for the smoothed position for one character.

(a) (b)

Figs. 9. Restaurant Scene, (a) top view, (b) perspective view



(a) (b)



Figs. 10. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate

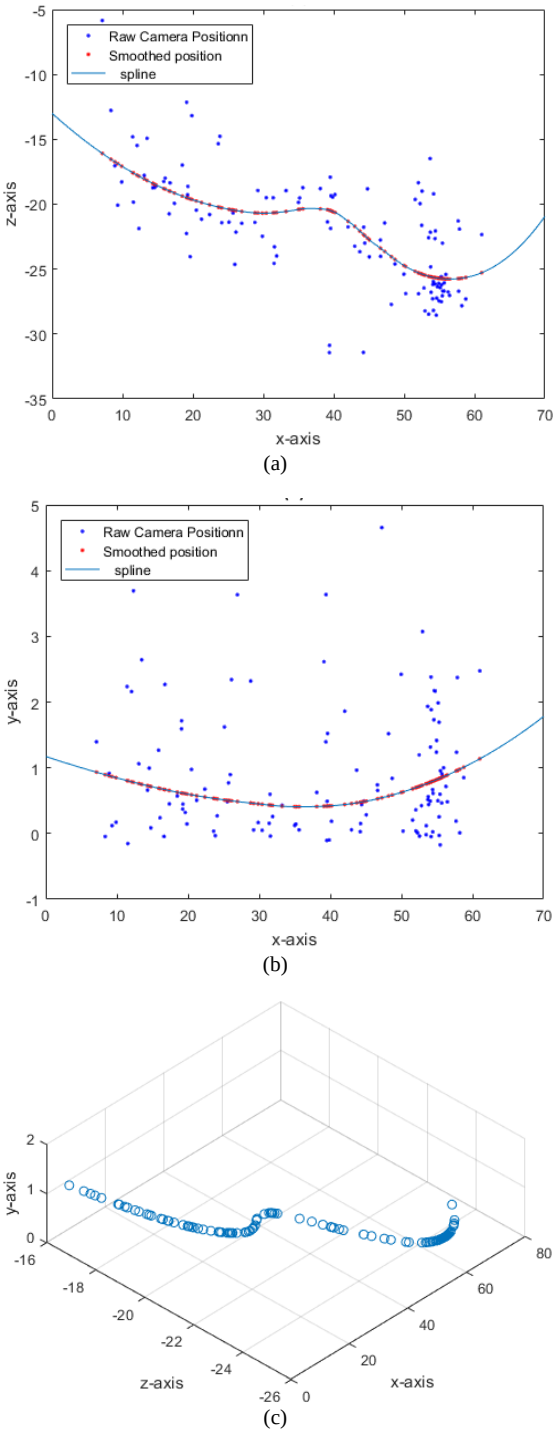
1s 2s 3s 4s

5s

Figs. 11. Camera position and frame result during one-character motion

Figs. 11 show the camera position in the forest scene during one-character motion. Figs. 12 show the best camera position computed using *PSO*, a smoothed position and a curve fitting for the smoothed position for two characters. Figs. 13 show the camera position in the forest scene during two characters' motions.

Figs. 14 show the best camera position computed using *PSO*, a smoothed position and a curve fitting for the smoothed position for three characters.



Figs. 12. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for two characters

IV.3. Scene 2 – Warehouse

In the first experiment, the camera tracks single character as he moves in the environment for 5 seconds.

The character will move from the center part of the scene towards the southwest building; this movement includes the character to use the stair.

For the second and third experiment, the camera tracks two and three characters with the same direction as before. Figs. 15 show the camera position in the forest scene during three characters' motions.

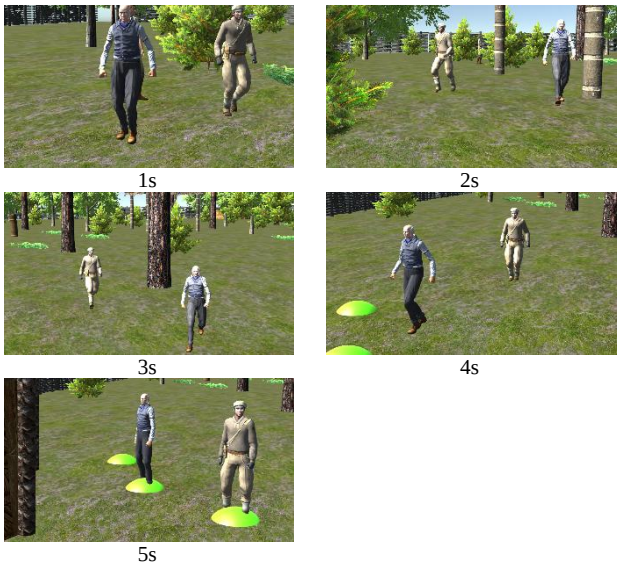
Figs. 16 show the best camera position computed using *PSO*, smoothed position and curve fitting for the smoothed position for one character; the high values changes in y coordinate as seen in Fig. 13(c) indicate that the characters' motion involves the changes of elevation in a scene. Figs. 17 show the camera position in the warehouse scene during one-character motion.

Figs. 18 show the best camera position computed using *PSO*, a smoothed position and a curve fitting for the smoothed position for two characters.

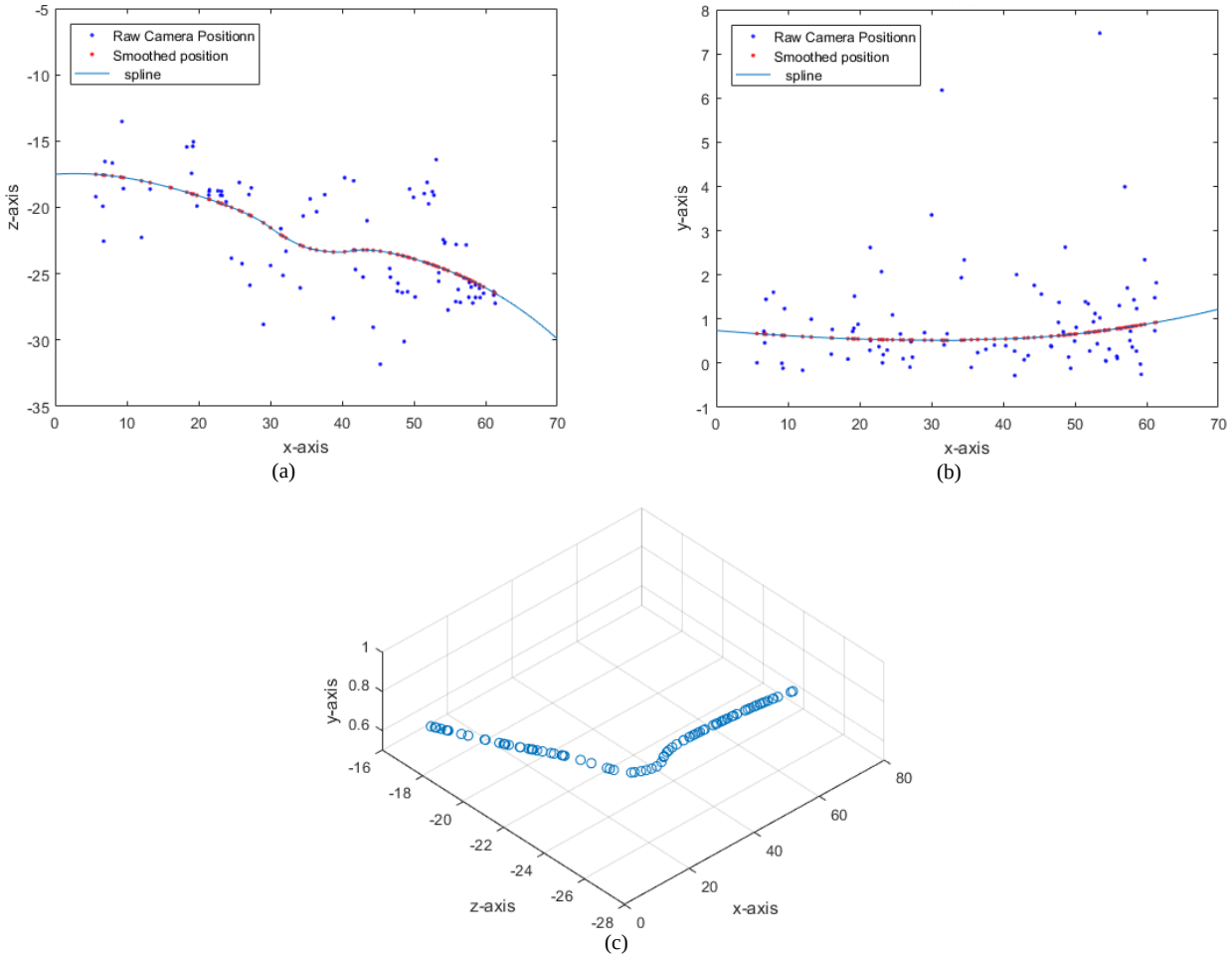
Figs. 19 show the camera position in the warehouse scene during two characters' motions. Figs. 20 show the best camera position computed using *PSO*, a smoothed position and a curve fitting for the smoothed position for three characters. Figs. 21 show the camera position in the warehouse scene during three characters' motions.

IV.4. Scene 3 – Restaurant

In the first experiment, the camera tracks the single character as he moves in the environment for 5 seconds from the center of the scene to the southeast area of the scene. For the second and third experiment, the camera tracks two and three characters with the same direction as before. Figs. 22 show the best camera position computed using *PSO*, a smoothed position and a curve fitting for the smoothed position for one character.

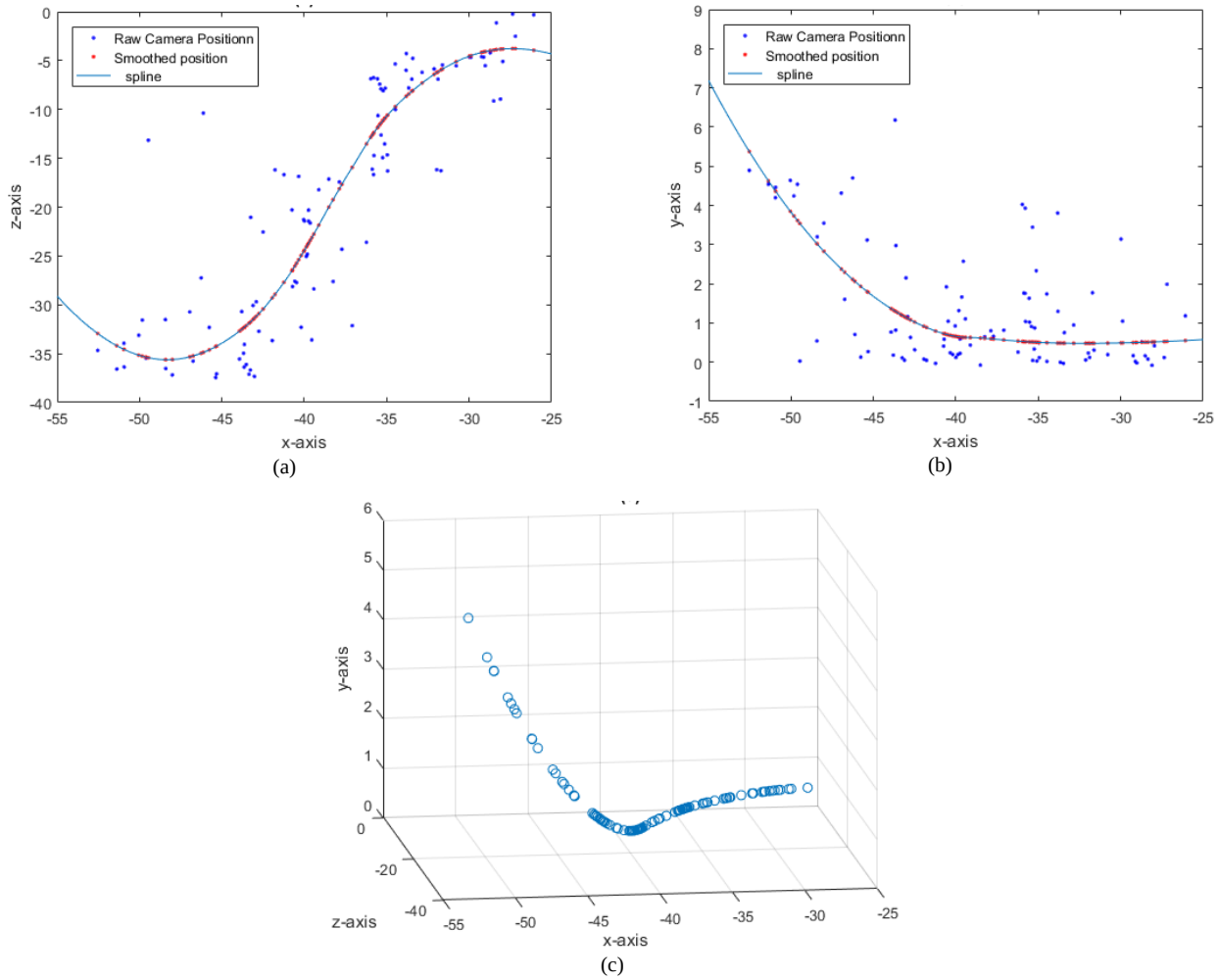


Figs. 13. Camera position and frame result during two characters' motions



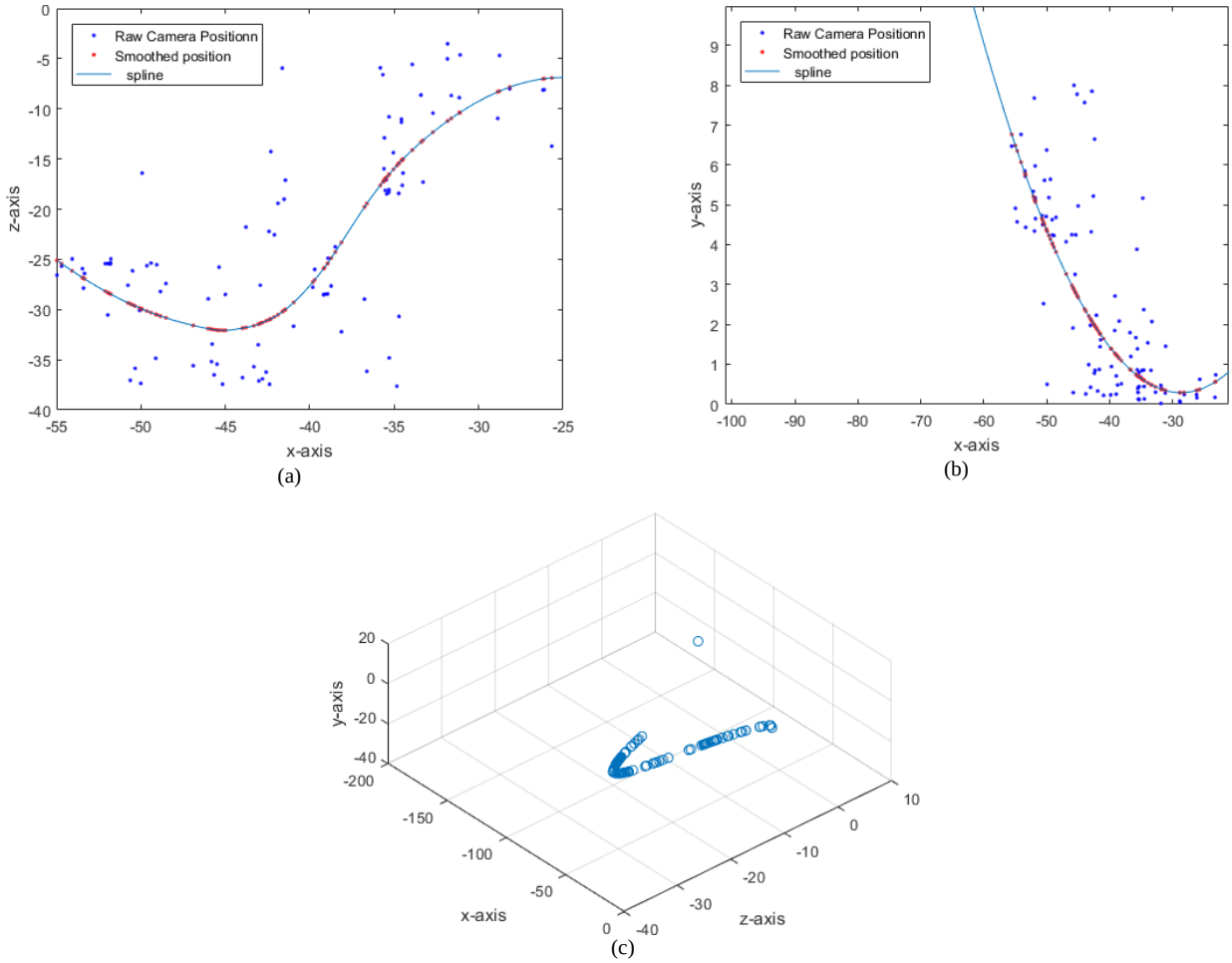
Figs. 14. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for three characters.

Figs. 15. Camera position and frame result during three characters' motions



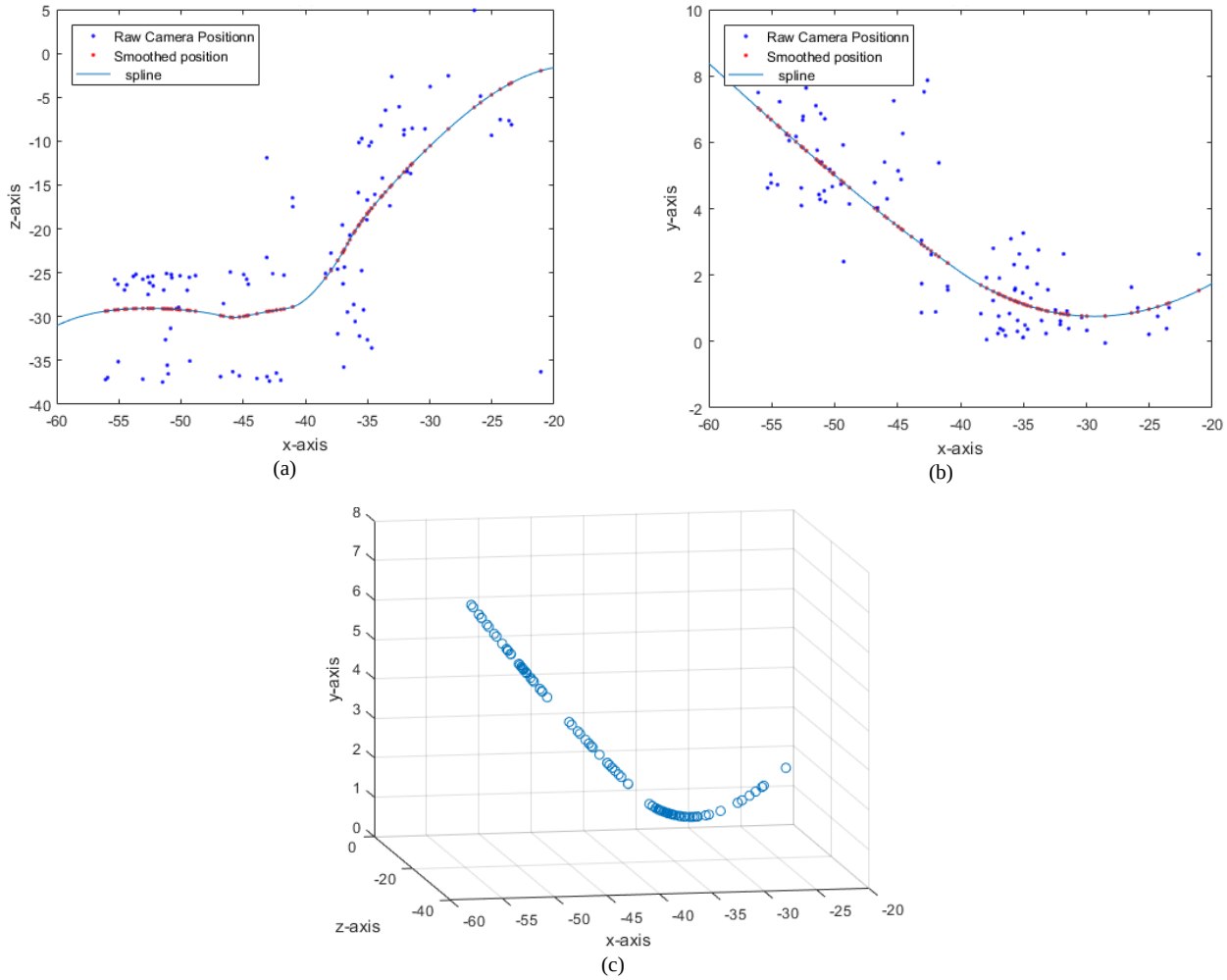
Figs. 16. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for one character

Figs. 17. Camera position and frame result during one-character motion

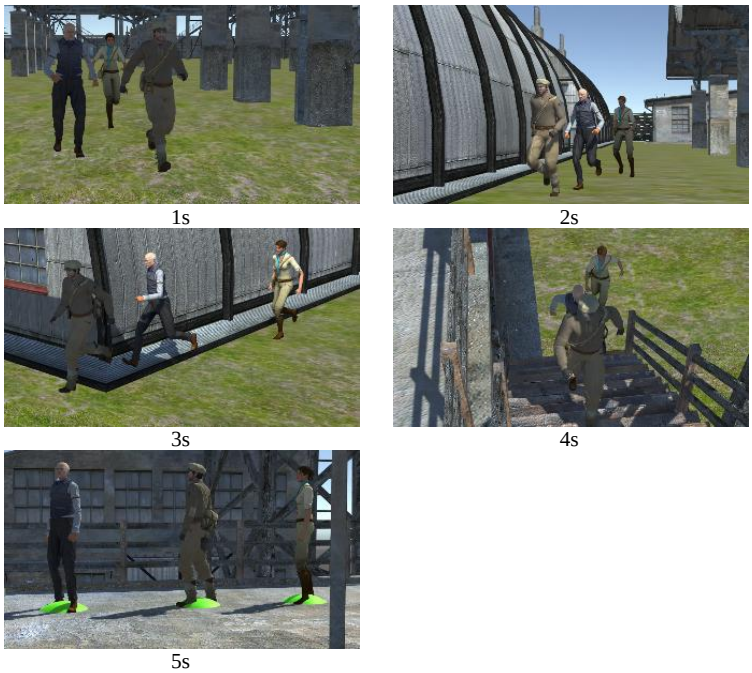


Figs. 18. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for two characters

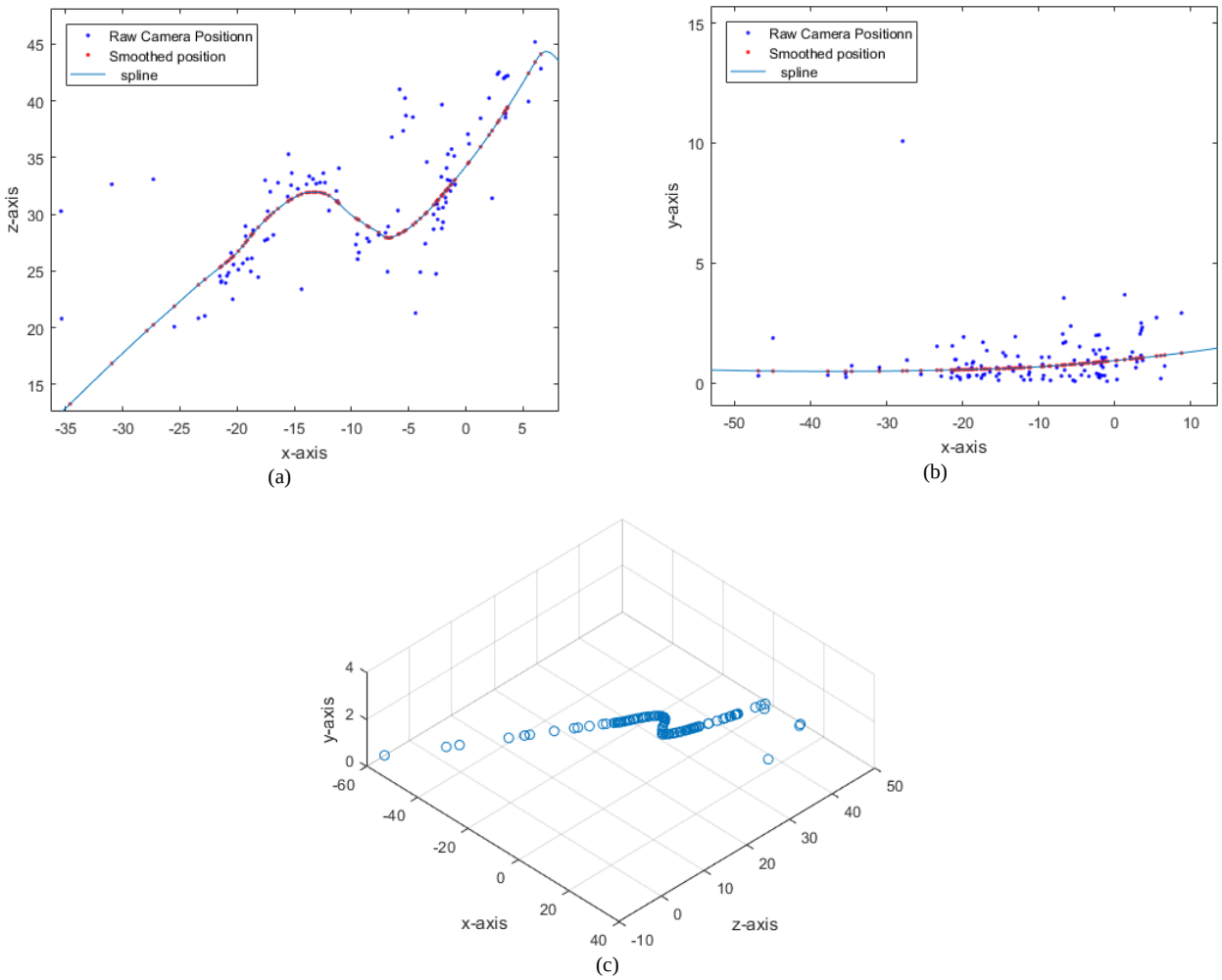
Figs. 19. Camera position and frame result during two characters' motions



Figs. 20. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for three characters

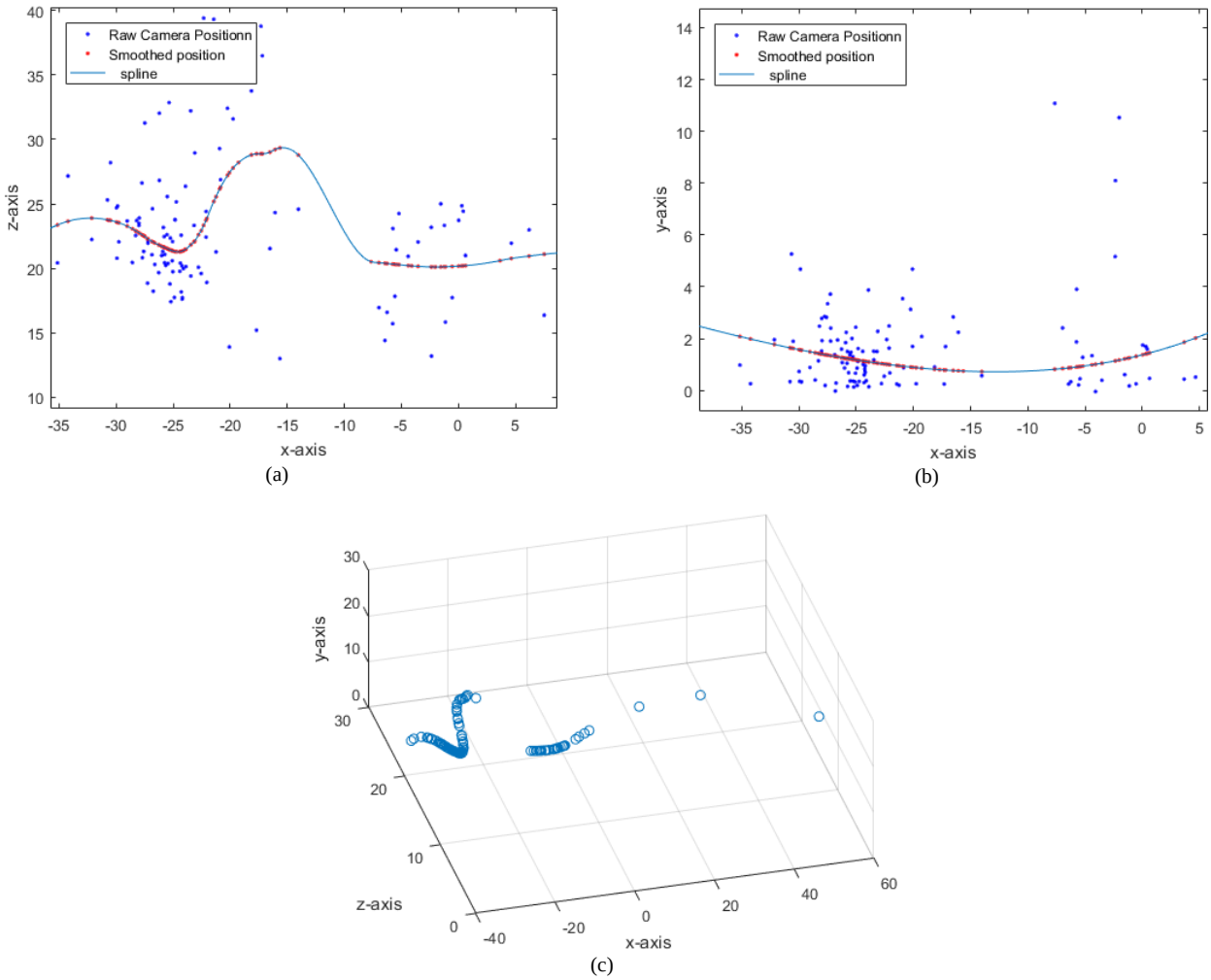


Figs. 21. Camera position and frame result during three characters' motions



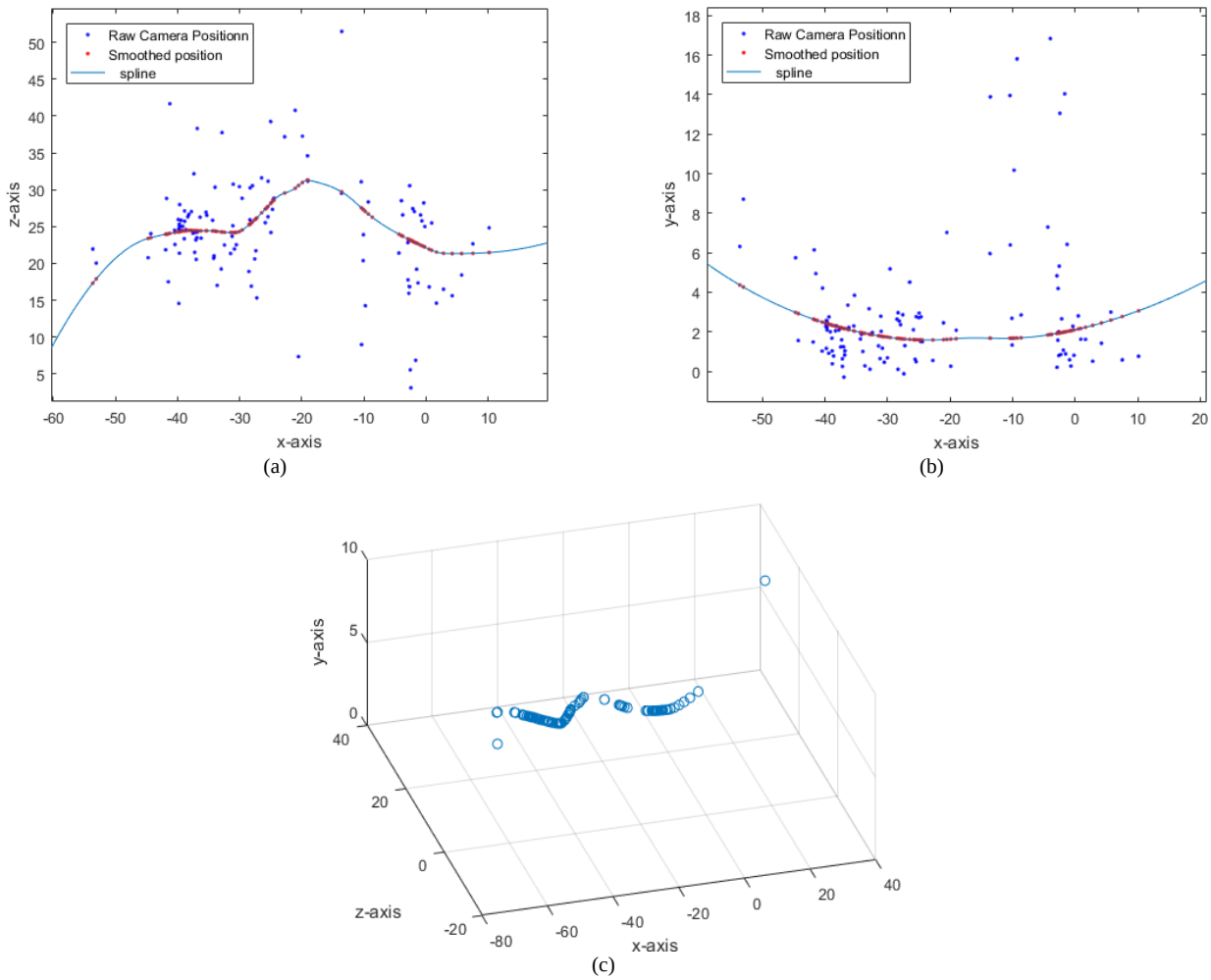
Figs. 22. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for one character

Figs. 23. Camera position and frame result during one-character motion



Figs. 24. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for two characters

Figs. 25. Camera position and frame result during two characters' motions



Figs. 26. (a) Smoothed camera position for z coordinate as a function for x, (b) for y coordinate as a function for x, and (c) combined 3D coordinate for two characters

Figs. 27. Camera position and frame result during three characters' motions

Figs. 23 show the camera position in the restaurant scene during one-character motion.

Figs. 24 show the best camera position computed using *PSO*, a smoothed position and a curve fitting for the smoothed position for two characters.

Figs. 25 show the camera position in the restaurant scene during two characters' motions.

Figs. 26 show the best camera position computed using *PSO*, a smoothed position and a curve fitting for the smoothed position for three characters.

Figs. 27 show the camera position in the restaurant scene during three characters' motions.

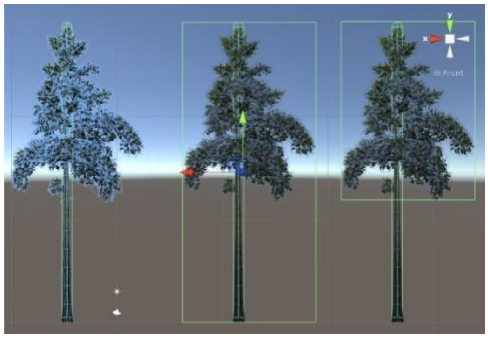
In the restaurant scene, many small occluders, like the lamp, table, chairs, small space between the wall and transparent window, make the computation higher than another scene.

Collider designs that are not made properly by the designer can lead to inaccurate optimization results.

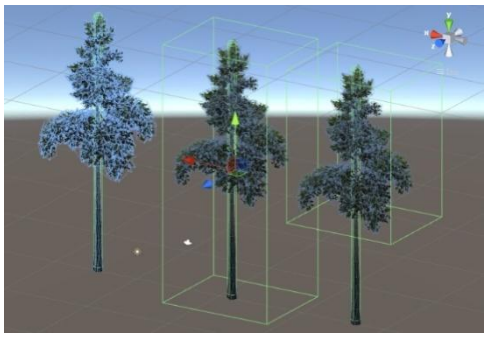
IV.5. Visibility

As mentioned in [17], using bounding box is not optimal because the collision detection uses raycasting from the camera to the bounding box of the object, so it is suggested to use combined box collider than using single bounding box collider for objects like trees and using mesh collider for a solid object like a building.

Figs. 28 show the different types of collider applied to the tree object. The result of different colliders applied into the forest scene are shown in Figs. 29.



(a)



(b)

Figs. 28. Different colliders applied to trees object, from left to right respectively using mesh collider, single bounding box, and combined bounding box. (a) front view, (b) perspective view

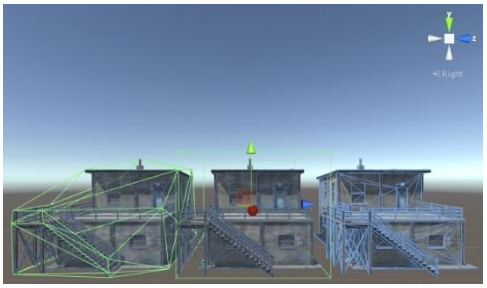
(a)

(b)

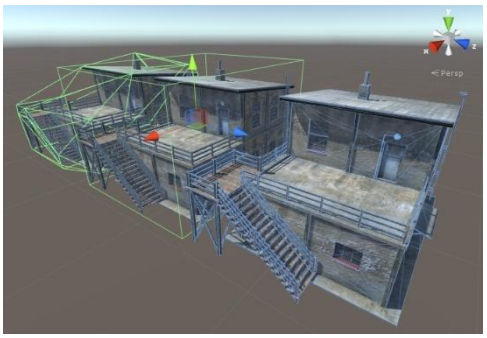
Figs. 29. (a) Using combined bounding box, (b) using mesh collider

It can be seen in Fig. 29(b) that mesh collider is not optimal for the forest scene because tree leafs and branch can act like partial occluders to the scene, so it is possible to see only the character feet. Fig. 29(a) shows the character proportional to the screen.

Figs. 30 show the different types of collider applied to building object. The results of different colliders applied into the warehouse scene are shown in Figs. 31.



(a)



(b)

Figs. 30. Different colliders applied to building object, from left to right respectively using convex mesh collider, single bounding box, and mesh collider. (a) front view, (b) perspective view

As illustrated in Fig. 31(c), mesh collider is optimal for the warehouse scene because it is possible to see all the characters climbing the stair, respect to the use of box

collider (Fig. 31(b)) that allows to see only the head of the characters and to the use of convex mesh collider (Fig. 31(a)) where the viewpoint is on the opposite side of the stair. Figs. 32 show the different types of collider applied to building objects.

The result of different colliders applied into the restaurant scene are shown in Figs. 33. It can be seen in Fig. 33(a) that mesh collider gives a better viewpoint than by using box collider as shown in Fig. 33(b).

From the experiment, it can be concluded that different uses of collider type for the same object will have different final frame results, depending on the scene itself. A not properly designed collider will affect the quality of result frame.

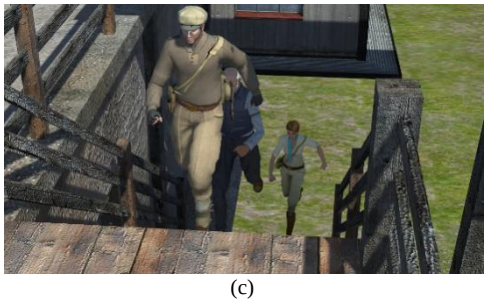
IV.6. Performance Evaluation

To provide an overview of the performance of the presented method for different scenes and character motions, each scene problem and character motion were solved 10 time, so to provide a reliable estimate of the measured statistics.

The proposed approach was tested in two different machine setups, a desktop workstation, and a mobile workstation, to see if there is a relation between machine computing capabilities and computation time.

(a)

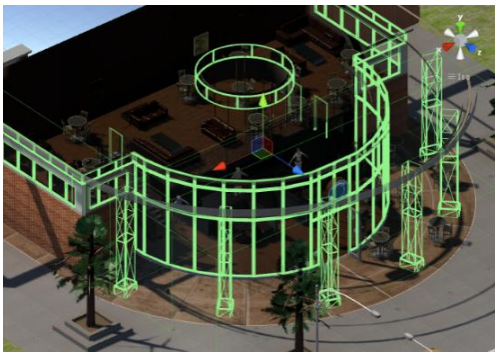
(b)



(c)

Figs. 31. Results from warehouse scene using different colliders, (a) convex mesh collider, (b) box collider, and (c) mesh collider

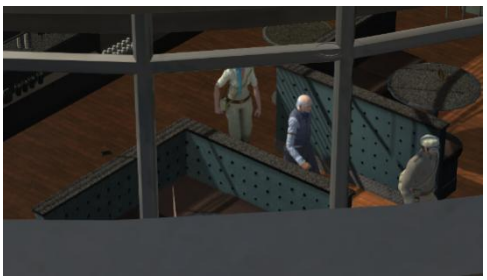
(a)



(b)

Figs. 32. Different colliders applied to the glass window and column object, (a)using bounding box, and (b) using mesh collider

(a)



(b)

Figs. 33. The results from restaurant scene using different colliders, (a) mesh collider and (b) box collider

Table III and Table IV show the computation spent in each tested scene for 5 seconds duration (150 frames) for the image resolution of 1152 × 648 pixels. As shown in Table III and Table IV, the computation time increases when the number of the characters in a scene increases, but there is no significant performance increase when using a desktop workstation or using a mobile workstation. The highest time difference is only around 6 seconds for all the tested scenes.

TABLE III
COMPUTATION TIMES-DESKTOP WORKSTATION

Scene	Min (s)	Max (s)	Average (s)	Total Computation (s)
Scene 1-Forest				
1 Character	0.051	0.200	0.066	9.942
2 Characters	0.047	0.282	0.058	8.630
3 Characters	0.056	0.206	0.069	10.37
Scene 2-Warehouse				
1 Character	0.044	0.218	0.062	6.239
2 Characters	0.066	0.206	0.079	7.919
3 Characters	0.070	0.220	0.091	9.056
Scene 3-Restaurant				
1 Character	0.041	0.198	0.051	6.132
2 Characters	0.047	0.176	0.058	6.973
3 Characters	0.054	0.241	0.067	8.019

TABLE IV
COMPUTATION TIMES-MOBILE WORKSTATION

Scene	Min (s)	Max (s)	Average (s)	Total Computation (s)
Scene 1-Forest				
1 Character	0.065	0.270	0.092	13.817
2 Characters	0.078	0.196	0.102	15.356
3 Characters	0.086	0.245	0.111	16.705
Scene 2-Warehouse				
1 Character	0.067	0.257	0.101	10.068
2 Characters	0.072	0.556	0.113	11.342
3 Characters	0.088	0.164	0.114	11.388
Scene 3-Restaurant				
1 Character	0.066	0.188	0.090	10.382
2 Characters	0.073	1.066	0.113	13.511
3 Characters	0.087	0.281	0.113	13.601

IV.7. Comparison with other Methods

What makes such approach different from [17] and [23] is that both previous works are applied to the static scene, where the character in a scene is not moving, while the proposed method involves a motion of a one, two, and three characters.

The other differences are how to compute the raw trajectory for the camera movement. In [25] authors used the motion of the character as well as the user to specifiy the initial and the last framing as the input, then the camera trajectory is computed by linking the initial and the final camera configuration; while the proposed method uses the character motion computed using pso to create the raw trajectory, and then smoothing the result by applying the local regression method.

V. Conclusion

This paper introduced an approach to create smooth camera movement for tracking the character motion in a dynamic scene by combining optimization process and local regression technique to avoid over reactively and jerky camera movement while computing for one optimal camera position per frame.

Nevertheless, additional experiments on other complex scenes involving a moving obstacle, or unpredictable events will be authors future research directions to find the most appropriate solution for virtual camera control algorithm.

References

[1] R. Ranon, L. Chittaro and F. Buttusi, "Automatic Camera Control Meets Emergency Simulation : An Application to Aviation Safety," *Computer & Graphics*, 2015.

[2] D. Arijon, Grammar of the Film Language, Hasting House Publishers, 1976.

[3] J. Bennett and C. P. Carter, "Adopting virtual production for animated filmaking," in *7th Annual International Conference on Computer Games, Multimedia and Allied Technology (CGAT 2014)*, Singapore, 2014.

[4] J. Chen and P. Carr, "Autonomous Camera Systems: A Survey," in *Twenty-Eighth AAAI Conference on Artificial Intelligence Intelligent Cinematography and Editing*, 2014.

[5] P. Burelli and M. Preuss,, "Automatic Camera Control: A Dynamic Multi-Objective Perspective," in *Applications of Evolutionary Computation*, 2014.

[6] Q. Galvane, R. Ronfard, M. Christie and N. Szilas, "Narrative-Driven Camera Control for Cinematic Replay of Computer Games," in *Motion In Games*, Los Angeles, 2014.

[7] P. Knöbelreiter, R. Berndt , T. Ullrich and D. W. Fellner, "Automatic fly-through camera animations for 3D architectural repositories," in *Computer Graphics Theory and Applications (GRAPP)*, Lisbon, 2014.

[8] L. Wang , Z. Zhen , X. Zhang and M. Sato, "Adaptive camera control method for efficient stereoscopic photography," in *International Conference on Industrial Technology (ICIT)*, Taipei, 2016.

[9] Q. Galvane, J. Fleureau, F.-L. Tariolle and P. Guillotel, "Automated Cinematography with Unmanned Aerial Vehicles," in *WICED: Eurographics Workshop on Intelligent Cinematography and Editing*, 2016.

[10] J. Fleureau, Q. Galvane, F.-L. Tariolle and P. Guillotel, "Generic Drone Control Platform for Autonomous Capture of Cinema Scenes," in *2nd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, 2016.

[11] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz and P. Hanrahan, "An interactive tool for designing quadrotor camera shots," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, 2015.

[12] S. M. Drucker and D. Zeltzer, "Intelligent Camera Control in a Virtual Environment," in *Proceedings of Graphics Interface*, 1994.

[13] L.-w. He, D. H. Salesin and M. F. Cohen, "The Virtual Cinematographer:A Paradigm for Automatic Real-Time Camera Control and Directing," in *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[14] W. H. Bares, J. P. Grégoire and J. C. Lester, "Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds," in *Innovative Applications of Artificial Intelligence*, 1998.

[15] W. H. Bares, S. McDermott, . C. Boudreaux and . S. Thainimit, "Virtual 3D camera composition from frame constraints," in *ACM International Conference on Multimedia*, 2000.

[16] D. K. Elson and M. O. Riedl, "A Lightweight Intelligent Virtual Cinematography System for Machinima Production," in *Artificial Intelligence and Interactive Digital Entertainment*, 2007.

[17] P. Burelli, L. D. Gaspero, A. Ermetici and R. Ranon, "Virtual Camera Composition with Particle Swarm Optimization," in *Smart Graphics*, 2008.

[18] E. E. Soares de Lima, A. E. Ciarlini and B. Feijó, "Support Vector Machines for Cinematography Real-Time Camera Control in Storytelling Environments," in *Brazilian Symposium on Digital Games and Entertainm*, 2009.

[19] P. Burelli and A. Jhala, "Dynamic Artificial Potential Fields for Autonomous Camera Control," in *Artificial Intelligence for Interactive Digital Entertainment*, 2009.

[20] D. Markowitz, A. Shoulson, N. I. Badler and J. T. Kider.Jr, "Intelligent Camera Control Using Behavior Trees," in *4th International Conference, MIG 2011*, Edinburgh, 2011.

[21] Q. Galvane, M. Christie, R. Ronfard, C.-K. Lim and M.-P. Cani, "Steering Behaviors for Autonomous Cameras," in *Motion in Games*, 2013.

[22] D. A. Prima, B. B. Ferial Java, E. Suryapto and M. Hariadi, "Secondary Camera Placement using Behavior Trees," in *2013 International Conference on QiR (Quality in Research)*, Yogyakarta, 2013.
[23] R. Ranon and . T. Urli, "Improving the Efficiency of Viewpoint Composition," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 5, pp. 795 - 807, 2014.
[24] P. Burelli and G. N. Yannakakis, "A Benchmark for Virtual Camera Control," in *Lecture Notes in Computer Science*, 2015.
[25] Q. Galvane, M. Christie, C. Lino and R. Ronfard, "Camera-on-rails: Automated Computation of Constrained Camera Paths," in *Motion in Games*, 2015.
[26] D. B. Christianson, S. E. Anderson, L. W. He, D. H. Salesin, D. S. Weld and M. S. Cohen, "Declarative camera control for automatic cinematography," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.
[27] S. M. Drucker, T. A. Galyean and D. Zeltzer, "Cinema : a system for procedural camera Movements," in *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, New York, 1992.
[28] C. Callaway, E. Not, A. Novello, C. Rocchi, O. Stock and M. Zancanaro, "Automatic cinematography and multilingual NLG for generating video documentaries," *Artificial Intelligence*, p. 57–89, 2005.
[29] D. Friedman and Y. A. Feldman, "Automated cinematic reasoning about camera behavior," *Expert Systems with Applications*, p. 694–704, 2006.
[30] A. Z. Fanani , D. A. Prima, B. B. F. Java, E. Suryapto, M. Hariadi and I. K. E. Purnama, "Secondary camera movement in machinema using path finding," in *International Conference on Technology, Informatics, Management, Engineering, and Environment (TIME-E)*, 2013.
[31] W. Barry and B. J. Ross, "Virtual photography using multi-objective particle swarm optimization," in *The 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014.
[32] P. Burelli, "Implementing Game Cinematography: Technical Challenges and Solutions for Automatic Camera Control in Games," in *Eurographics Workshop on Intelligent Cinematography and Editing*, 2015.
[33] N. Fourati, J. Huang and C. Pelachaud, "Dynamic stimuli visualization for experimental studies of body language," in *10th Workshop on Multimodal Corpora, In conjunction with LREC 2014 (MMC 2014)*, Reykjavik, Iceland, 2014.
[34] C. Sanokho, C. Desoche, B. Merabti, T. Y. Li and M. Christie, "Camera Motion Graphs," in *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, 2014.
[35] P. C. Lai, H. Y. Wu, C. Sanokho, M. Christie and T. Y. Li, "A Pattern-based Tool for Creating Virtual Cinematography in Interactive Storytelling," in *12th International Symposium on Smart Graphics*, Taipei, Taiwan, 2014.
[36] H. N. Dib, N. A. Villani and J. Yu, "Computer Animation for Learning Building Construction Management: A Comparative Study of First Person versus Third Person View," in *E-Learning, E-Education, and Online Training Volume 138 of the series Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 2014.
[37] P. Burelli, "Adaptive Virtual Camera Control Trough Player Modelling," *User Modeling and User-Adapted Interaction*, vol. 25, no. 2, 2015.
[38] B. Merabti, M. Christie and K. Bouatouch, "A Virtual Director Using Hidden Markov Models," *Computer Graphics Forum*, 2015.
[39] W. Hu and X. Zhang, "A Semiautomatic Control Technique for Machinima Virtual Camera," in *International Conference on Computer Science and Electronics Engineering (ICCSEE)*, Hangzhou, 2012.
[40] C.-H. Chen and T.-Y. Li, "Context-aware Camera Planning for Interactive Storytelling," in *Ninth International Conference on Computer Graphics*, Hsinchu, 2012.
[41] W.-H. Hsu, Y. Zhang and K.-L. Ma, "A Multi-Criteria Approach to Camera Motion Design for Volume Data Animation," *IEEE Transactions on Visualization and Computer Graphics* , vol. 19, no. 12, 2013.
[42] M. Preuss, P. Burelli and G. N. Yannakakis, "Diversified Virtual Camera Composition," in *Applications of Evolutionary Computation*, Springer Berlin Heidelberg, 2012, pp. 265-274.
[43] M. Christie, J.-M. Normand and P. Olivier, "Occlusion-free Camera Control for Multiple Targets," in *Eurographics Symposium on Computer Animation*, Lausanne, 2012.
[44] E. T. Lixandru and V. Zordan, "Physical Rig for First-person, Look-at Cameras in Video Games," in *Motion in Games*, 2014.
[45] A. N. Erdem and U. Halıcı, "Game Camera Direction Based on Computational Aesthethics Using Machine Learning," *IEEE Computer Graphics and Applications*, no. 99, 2016.
[46] R. C. Eberhart and J. Kennedy., "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, 1995.

Authors' information

[1]Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.

[2]Universitas Surabaya, Surabaya, Indonesia.

[3]Kumamoto University, Kumamoto, Japan.
E-mails: delta@staff.ubaya.ac.id
mochar@te.its.ac.id
ketut@te.its.ac.id
tuie@cs.kumamoto-u.ac.jp

Delta Ardy Prima received the bachelor degree in information technology from Electrical Engineering Polytechnic Institute of Surabaya (EEPIS), Surabaya in 2007. He received his master degree from Electrical Engineering Department of Institut Teknologi Sepuluh Nopember (ITS), Surabaya in 2009. Currently he is a staff member of informatics engineering, University of Surabaya (UBAYA). Now he studying at doctoral program in Electrical Engineering Department ITS Surabaya. His research interest focuses in Video and Image Processing, Game Technology and Intelligent System

Mochamad Hariadi received the B.E. degree in Electrical Engineering Department of Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, in 1995. He received both M.Sc. and Ph.D. degrees in Graduate School of Information Science Tohoku University Japan, in 2003 and 2006 respectively. He is a staff member of Electrical Engineering Department of Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. He is the project leader in joint research with PREDICT JICA project Japan and WINDS project Japan. His research interests are in Video and Image Processing, Data Mining and Intelligent System. He is a member of IEEE, IEICE, and IAENG.

I. Ketut Eddy Purnama received the bachelor degree in Electrical Engineering from Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia in 1994. He received his Master of Technology from Institut Teknologi Bandung, Bandung, Indonesia in 1999. He received Ph.D degree from University of Groningen, the Netherlands in 2007. Currently, he is a staff member of Electrical Engineering Department of Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. His research interest are in Data Mining, Medical Image Processing and Intelligent System.

Tsuyoshi Usagawa joined Kumamoto University in 1983 right after he received the M.E. degree from Tohoku University. In 1988, he received Dr. Eng. from Tohoku University. Since 2004 he has been a professor and he is a member of IEEE, ASA, ASJ, INCE/J etc. He is interested in e-Learning contents and system, and acoustic signal processing.