Arjan Egges
Roland Geraerts
Mark Overmars (Eds.)

LNCS 5884

Motion in Games

Second International Workshop, MIG 2009
Zeist, The Netherlands, November 2009
Proceedings

Lecture Notes in Computer Science 5884

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Arjan Egges Roland Geraerts
Mark Overmars (Eds.)

Motion in Games

Second International Workshop, MIG 2009
Zeist, The Netherlands, November 21-24, 2009
Proceedings

Springer

Volume Editors

Arjan Egges
Mark Overmars
Roland Geraerts
Universiteit Utrecht
Games and Virtual Worlds Group
PO Box 80.089, 3508TB Utrecht, The Netherlands
E-mail: {egges, roland, markov@cs.uu.nl}

Library of Congress Control Number: 2009938716

CR Subject Classification (1998): K.8, I.2.1, I.3, I.3.7, H.5, J.5

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

ISSN 1867-8211
ISBN-10 3-642-10346-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-10346-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12798742 06/3180 5 4 3 2 1 0

Preface

Following the very successful Motion in Games event in June 2008, we organized the Second International Workshop on Motion in Games (MIG) during November 21–24, 2009 in Zeist, The Netherlands.

Games have become a very important medium for both education and entertainment. Motion plays a crucial role in computer games. Characters move around, objects are manipulated or move due to physical constraints, entities are animated, and the camera moves through the scene. Even the motion of the player nowadays is used as input to games. Motion is currently studied in many different areas of research, including graphics and animation, game technology, robotics, simulation, computer vision, and also physics, psychology, and urban studies. Cross-fertilization between these communities can considerably advance the state of the art in this area. The goal of the workshop Motion in Games is to bring together researchers from this variety of fields to present the most recent results and to initiate collaboration. The workshop is organized by the Dutch research project GATE. In total, the workshop this year consisted of 27 high-quality presentations by a selection of internationally renowned speakers in the field of games and simulations. We were extremely pleased with the quality of the contributions to the MIG workshop and we look forward to organizing a follow-up MIG event.

November 2009

Arjan Egges
Mark Overmars
Roland Geraerts

Organization

Program Chairs

Mark Overmars	Games and Virtual Worlds group, Utrecht University, The Netherlands
Arjan Egges	Games and Virtual Worlds group, Utrecht University, The Netherlands

Local Chair

Roland Geraerts	Games and Virtual Worlds group, Utrecht University, The Netherlands

Program Committee

Boulic, Ronan	VRLab, EPFL, Lausanne, Switzerland
Chrysanthou, Yiorgos	University of Cyprus, Cyprus
Donikian, Stéphane	IRISA, Rennes, France
Di Fiore, Fabian	University of Hasselt, Hasselt, Belgium
Egges, Arjan	Utrecht University, The Netherlands
Faloutsos, Petros	University of California, USA
Geraerts, Roland	Utrecht University, The Netherlands
King, Scott	Texas A&M, Corpus Christi, USA
Komura, Taku	Edinburgh University, UK
Laumond, Jean-Paul	LAAS, Toulouse, France
Lin, Ming	University of North Carolina, USA
Magnenat-Thalmann, Nadia	MIRALab, Geneva, Switzerland
Manocha, Dinesh	University of North Carolina, USA
Multon, Franck	CNRS-INRIA, France
Müller, Heinrich	Dortmund University of Technology, Germany
Nijholt, Anton	Universiteit Twente, The Netherlands
O'Sullivan, Carol	Trinity College Dublin, Ireland
Overmars, Mark	Utrecht University, The Netherlands
Pelachaud, Catherine	CNRS, France
Pettré, Julien	IRISA, Rennes, France
Thalmann, Daniel	VRLab, EPFL, Lausanne, Switzerland
Zhang, Jian	Computer Animation Research Centre, Bournemouth University, UK

Sponsored by

Motion in Games 2009 was sponsored by the GATE project[1,2] and Microsoft Nederland[3].

[1] http://gate.gameresearch.nl

[2] The GATE project is funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

[3] http://www.microsoft.com/netherlands

Table of Contents

Avoidance Behaviour

Behaviour and Affect

Crowd Simulation

Motion Analysis and Synthesis

Navigation and Steering

Physics

Rendering and Video

Collision Avoidance between Avatars of Real and Virtual Individuals

René van den Berg[1], Juan Manuel Rejen[2], and Rafael Bidarra[1]

[1] Delft University of Technology
rene3.berg@planet.nl, r.bidarra@tudelft.nl
[2] iOpener Media GmbH
manuel@iopenermedia.com

Abstract. One of the most difficult challenges of associating virtual environments and real-world events deals with integrating real-world, tracked individuals into a virtual environment: while virtual individuals may interact with the avatars of real-world individuals, the latter obviously do not respond to the behaviour of the virtual individuals. To prevent collisions or unrealistic behaviour of either, one needs to 'artificially' modify the trajectory of the tracked individual. Moreover, after such modifications, one should end up returning that avatar back to the accurate representation of the real world.

In this paper we propose a strategy for solving these problems based on generic control functions, which are flexible enough to be fine-tuned on a domain-dependent basis. Control functions determine an offset from the tracked individual's actual trajectory and, when carefully parameterized, guarantee a smooth and automatic recovery after each trajectory modification. We discuss this approach, illustrate it with a variety of Gaussian control functions, analyze the believability of the resulting trajectory modifications, and conclude that control functions provide a sound and powerful basis for improving strategies of collision-avoiding trajectory modification and recovery. Most examples of domain-specific strategies discussed here are taken from the upcoming application area of real-time racing games, which provides both easily recognizable and very attractive situations, and has been the original motivation for this research.

Keywords: Trajectory modification, collision evasion, real-world integration.

1 Introduction

According to [4], a virtual environment (VE) is "a multi-sensory real-time simulation that immerses the participant in a multi-dimensional (usually 3D) graphical space, allows freedom of movement within the space, and supports interactions including the modification of most features of the space itself." For the purposes of this paper, the most important part of this definition is the fact that a VE supports interactions: the user can project himself into an environment and interact with it as if it were real.

Taking this concept one step further, we might imagine a VE that *is* real: a virtual copy of an actual environment. If we can copy each relevant static feature of a real-world environment, and also track its relevant dynamic features, the VE may present the user with a virtual clone of an actual environment.

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 1–12, 2009.

In such an environment, players may then interact with other individuals through avatars. For example, suppose we create a racing game in which the user is allowed to test his skills against the professional drivers, who are racing at the track at that very same time. The actions of each opponent are tracked and transmitted over the Internet to players at home, so that they may not only see, but even participate in the race [5].

Alas, such players can only participate in any real-world environment using *asymmetric interactivity*. Asymmetric interactivity means that although the player and the VE are influenced by the state of the real world, the real-world environment is not altered by any virtual interaction, nor does any real-world individual directly act upon any purely virtual object. Thus, the interaction must be simulated locally. For example, a real-world vehicle's trajectory may pass through a virtual object. To avoid collisions or even overlap in such cases, the trajectory of either the real-world or the virtual individual (the player's avatar) needs to be modified — but the player will not tolerate being moved around by external forces. Also, it is up to the active party to prevent collisions — if any individual is performing an overtaking maneuver, the responsibility of evading the other individual lies with him. Therefore, the system must intelligently modify the trajectory of the virtual individual.

Any modifications to a real-world individual's trajectory must be performed in a way that is guaranteed to recover, stable, and realistic or, at least, believable. A modification that is guaranteed to recover means that the influence of another individual on the behavior of the virtual individual decreases as the distance between them increases, until it is 0 at a certain point. Stability means that care should be taken to decrease the probability that the modification of one individual's trajectory brings it so close to another that the *other* individual will need to start maneuvering. Realism, or the less stringent requirement of believability, dictates that an observer or the user is not able to distinguish whether the trajectory perceived was modified or not, or at least the alterations are not disturbing in any way.

Our main contribution is the notion of a *control function*, a generic strategy to perform trajectory modifications on the avatars of real-world individuals in a way that is guaranteed to recover with time, and allows for the customization of the specific trajectory modification strategy based on current circumstances and the type of environment.

The rest of this paper is structured as follows: in the next section, we will present some previous research that is related to ours. Section 3 will highlight the context in which our approach can be used, followed by an explanation of the basic approach in Section 4. Section 5 provides the theory for possible extensions to the control function approach, and Section 6 will present some results, followed by conclusions and future work in Section 7.

2 Related Work

The specific domain of interaction between dynamic real-world individuals and virtual individuals is still very new. However, there are some areas with strong similarities to our current research area.

The field of robotic navigation is one example. Some interesting research has been performed in the area of motion planning in the presence of moving obstacles, proving

that the problem of finding an optimal solution to the problem is intractable [9]: the complexity of the problem prevents a solution from being found fast enough to be used in all but the most trivial cases. However, nearly optimal planning and collision avoidance are still thoroughly being researched. Recent results from this research include the idea of deforming precomputed roadmaps discussed in [3]. This approach features roadmaps made up of nodes, chained together by elastic links, which deform in the presence of obstacles.

Another field with strong relations to the current research is the simulation of human crowds or other groups. This field makes strong use of short-term predictions, collision-avoiding trajectories and goalseeking. A lot of emergent behavior can be seen in these simulations, provided they are adequately programmed: lane formations, for example, naturally occur in some synthesized environments, as well as in the real world.

Given that a model exists that can adequately simulate this behavior, it can also be used for predicting it. Discomfort fields [10], the use of local potential fields and the separation between high-level static path planning and local collision avoidance [11] are some of the important ideas that can be applied to the research at hand. Also, future work may incorporate some simple psychology for path prediction or modification, as proposed in, for example, [7].

Finally, an interesting angle on path prediction and recovery from incorrect movements is presented in [1].

3 Domain and Problem Characteristics

In this section, some important aspects of trajectory modification strategies are presented. This knowledge is required to understand the problems our approach deals with, and the assumptions made in developing our solution. A more in-depth treatment of related issues can be found at [12].

First, the real-world individuals are tracked within the real-world environment. This data is then transmitted using any suitable means to the local VE. It is in this local VE that we apply trajectory modification strategies.

To determine whether the trajectory of a real-world individual can be represented as-is or requires modification, it is necessary to obtain accurate trajectory predictions for both the real-world and the virtual individuals. The real-world individual's trajectory is often easier to predict, given accurate input variables. A simple physical extrapolation will suffice for most scenarios, since most "real" individuals are not inclined to sudden behavior alterations. Examples of such extrapolations are the various Dead Reckoning models, which are commonly used in such diverse applications as computer games and navigation systems to reduce network load [4], [1]. However, different environments require different Dead Reckoning models, as has been extensively researched in [8].

The behavior of the virtual individual is more difficult to predict: the player is more likely to "misbehave", or do something unexpected. In a car race, for example, it is reasonable to assume that no real-world individual will drive against traffic on purpose. A player is a lot more likely to misbehave in such circumstances, for a myriad of reasons, among which are the lack of social pressure, the willingness to test limits, and the fact that there is, after all, no actual danger involved. But predicting the virtual individual's trajectory *is* the most important: if no virtual individuals are near any real-world

individuals, no modification to the real-world individuals' trajectory is ever needed. If the user is not purposely misbehaving, a mix between a short-term physical model and a longer-term strategic model is ideal for prediction. Examples of this approach are the Hybrid Model [2], which alternates between a Dead Reckoning prediction and a strategical prediction, and the Hybrid Method [6], which enhances Dead Reckoning prediction with longer-term Interest Schemes.

Assuming that some adequate prediction scheme can be found, the predictions for both the virtual individual and each real-world individual are calculated and compared. If their predicted positions are, at some future point, too close together, overlap or collisions may occur. To prevent this, an evasive maneuver performed by the real-world individual is simulated — but only if that individual is the "active" party in the impending collision. In a car race, for example, the frontmost individual would not make space for the overtaking individual. Which individual, if any, performs the evasive maneuver should be determined by the system based on the current circumstances.

Another characteristic is that the prediction algorithm used for the real-world individuals depends on the nature of the interaction. If the tracking data of the real-world individuals is prerecorded, or sufficiently delayed in their presentation, the near future of their trajectory is already known. This future may be used as an "extremely accurate prediction". However, this necessitates an estimation of the predictability of that future; if it includes "surprising" elements, it should be rejected as a prediction and replaced by some other, more "naive" prediction scheme.

4 Basic Approach

We propose to control the evasive maneuver using a *control function*, which describes the modifications to the trajectory of the real-world individual. Trajectory modifications are only ever applied to real-world individuals, since they have no full knowledge of their surroundings in the virtual environment, as opposed to the virtual individuals. Furthermore, we focus here on theory aspects that are valid for all applications, leaving application-specific analyses, such as tuning physical constraints, for future work.

4.1 Basics of Control Functions

An n-dimensional control function is a combination of one or more mathematical functions $f(d) : \mathbb{R}^m \rightarrow \mathbb{R}^n$, with $m < n$, which determines the offset from the actual trajectory for an evading individual. The input variable d is the distance between the centers of the two individuals in their *unmodified* positions, and the output variables are offsets in certain directions. This notion of "certain directions" is kept deliberately vague, since control functions can be applied to a variety of coordinate systems, such as Cartesian, polar or cylindrical coordinates. In this paper, we will assume that the primary axis is always perpendicular to the tangent of the trajectory.

Control functions might not reflect physical restrictions, so it is up to the person using them to make sure that the chosen function is suitable for the application at hand in terms of maneuverability, velocity adjustments, et cetera.

Furthermore, instead of the normal distance between the individuals, we use the signed distance to distinguish between being ahead or behind the other individual. A

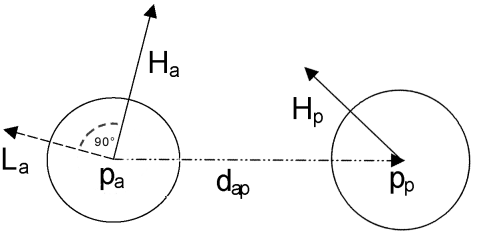


Fig. 1. The important variables of the trajectory modification. $\mathbf{p_a}$ and $\mathbf{p_p}$ are the positions of the active and passive individuals, respectively. $\mathbf{H_a}$ and $\mathbf{H_p}$ are their respective heading vectors, or the tangent vectors of their trajectories. $\mathbf{d_{ap}}$ is the vector from the active to the passive individual and $\mathbf{L_a}$ is the active individual's normalized *left-hand vector*, the primary axis for modification.

positive distance then indicates that the active individual is ahead of the passive one, that is, $\mathbf{d_{ap}} \cdot \mathbf{H_a} > 0$, and vice versa (see Fig. 1).

However, this signed distance might never be 0. This causes the output variable to display discontinuous behavior if the control function is not symmetric around $d = 0$, since in that case the distance, at some point, suddenly switches signs without passing through 0. To prevent this, the input variable used is the signed magnitude of the difference vector $\mathbf{d_{ap}}$ between the individuals projected onto the heading vector H_a, which is calculated by $\mathbf{d_{ap}} \cdot \mathbf{H_a}/|\mathbf{H_a}|$. This projected signed distance is then guaranteed to be continuous. The heading vector should not exhibit sudden changes, because otherwise the projected signed distance will behave erratically. For most environments, this is a reasonable requirement. If it is not, the passive individual could be projected onto the active individual's trajectory instead.

A very welcome consequence of using the distance as the input variable, is that the graph of the control function horizontally scales with velocity difference: no matter what the velocity difference between the individuals is, the resulting trajectory modification will always look smooth. This is illustrated in Fig. 2.

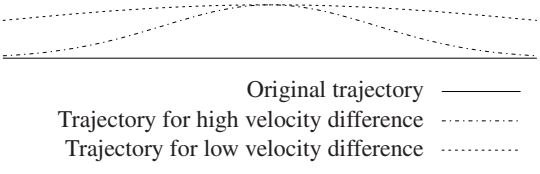


Fig. 2. The automatic "stretching" of the deformation function due to low velocity difference

4.2 Requirements and General Observations

To be suitable as a control function, a mathematical function $f(x) : \mathbb{R} \rightarrow \mathbb{R}^n$ should fulfill some requirements.

First of all, it should accurately reflect the capabilities of the individual for whom the modification strategy is meant. This includes, amongst others, the requirement of

continuity, so as not to yield any sudden changes in position or direction of movement.

Second, it should have *finite support*, which means that $f(x) > 0$ for only a finite set of input values. This region of support should contain $x = 0$ and be nearly symmetric around the y axis. This allows for a smooth transition between the original and the modified trajectory at the start and end of the maneuver.

Another general rule is that most functions should often be centered around the y axis. Also, the function should generally increase over exactly one interval and decrease over exactly one other interval: "swaggering" mostly decreases the believability of the resulting trajectory.

4.3 1-Dimensional Control Functions

A 1-dimensional (1-D) control function has an output value which is used as an offset in a direction perpendicular to the $\mathbf{H_a}$ vector. A default direction must be determined beforehand, so that the sign of the output becomes meaningful; in this paper, we assume that the default direction is L_a (see Fig. 1), 90 degrees clockwise from H_a as viewed from above, or "left". The trajectory will mostly be modified to the same side the active individual is on with respect to the passive individual: to the left if it is on the left, and vice versa.

To determine whether to modify the trajectory to the left or the right side, the following formula, which makes use of the *signum* function, is used:

$$\operatorname{sgn}(\mathbf{L_a} \cdot \mathbf{d_{ap}}) * \operatorname{sgn}(\mathbf{H_a} \cdot \mathbf{H_p}). \tag{1}$$

The result of this equation is multiplied by the outcome of the control function to yield a signed offset magnitude, which should be multiplied by $\mathbf{L_a}$ and added to p_a to obtain the active individual's new position. Special care should be taken in case Equation 1 yields 0. This may occur in two cases: if the heading vectors are perpendicular to each other, the evasion should occur as determined by first part of the equation, but if p_p lies on the line determined by H_a, the sign should be calculated as $-\operatorname{sgn}(\mathbf{L_a} \cdot \mathbf{H_a})$.

An excellent example of a 1-D control function is the Gaussian function. While it does not strictly comply with the "finite support"-rule, some of its other properties make it useful as a control function in every other respect — most importantly its nicely bell-shaped graph. The generic formula for a Gaussian function g is:

$$g(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{2}$$

for any set of real values for $a \neq 0$, b and $c \neq 0$. These parameters are, interestingly enough, nicely reflected in the shape of the graph: while a is reflected in the amplitude of the Gaussian, and thus the magnitude of the trajectory modification, b relates to its location along the x-axis and c is a measure for the width of the bell-shaped curve. The influence of the parameters is visualized in Fig. 3.

A slight detractor for the Gaussian function is that it can never satisfy $f(x) = 0$. This means that if the Gaussian function is used as a control function, the trajectory modification will never stop being active. However, the Gaussian function falls off quickly

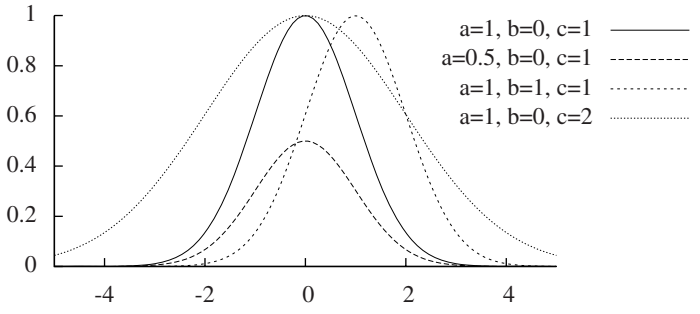


Fig. 3. The influence of the Gaussian parameters

enough as its input value increases or decreases to use a simple cutoff, fulfilling the finite support-requirement as well.

5 Advanced Control Function Extensions

In this section, we discuss two extensions to control functions: a 2-dimensional control function approach, and automatic configuration of control function variables.

5.1 2-Dimensional Control Functions

A 2-dimensional control function adds a longitudinal offset to the perpendicular offset. This can be used to fine-tune the trajectory generated by a 1-D control function. The net effect of an extra modification dimension is an apparent speed modification. Extra care should be taken to prevent unrealistic or impossible behavior, since the direction of movement is modified; for example, to seemingly slow down a bit, the individual moves less in the forward direction, while the movement in the perpendicular direction stays the same. The result of this modification may become unrealistic if the perpendicular movement overshadows the longitudinal movement.

The effect can be visualised as taking the graph of the primary, 1-D control function, and displacing each point on it in the longitudinal direction, where the displacement is controlled by a second 1-D control function. In the context of 2-D control functions, this new, 1-D part of the function is referred to as the *longitudinal control function*, while the previously discussed primary control function is referred to as the *perpendicular control function*.

In general, there are some simple rules to keep in mind when selecting a mathematical function to be used as the longitudinal control function:

- If the input and the output value have the same signs, the distance between the individuals is increased by the control function.
- A negative output value puts the individual "behind" its actual location. Conversely, a positive value means the individual is ahead of its true location.
- Anywhere the longitudinal deformation function has a positive derivative, the individual accelerates. By the same token, a negative derivative slows it down.

– Both the function and its first and second derivative should never have extremely high or low values, because this may have an adverse effect on the plausibility of the resulting movement. If the individual is projected too far ahead, for example, this may result in a velocity that is higher than the physical maximum of that individual.

Any function which follows these rules is fit to be used as a longitudinal deformation function. Keep in mind that a 2-D control function may be either a combination of multiple 1-D control functions, or a mathematical function $f(d) : \mathbb{R} \rightarrow \mathbb{R}^2$.

Our prime example of combining two 1-D control functions to yield a 2-D control function is using the Gaussian function as the perpendicular control function and its derivative as the longitudinal control function. This approach is especially useful since both functions can be manipulated by the same parameters b and c, as discussed in the previous section. Also, the Gaussian derivative has one distinct region where the result is positive and one region where the result is negative, and these regions are rotationally symmetric around the point $(b, 0)$. All this means that it can be used to accelerate first and then decelerate when the other individual is passed, or the other way round.

Moreover, if the same c value is used for the longitudinal and the perpendicular control function, the *inflection point* of the perpendicular modification coincides with the minimum or maximum longitudinal offset. This inflection point is where the perpendicular acceleration switches signs, which means that the perpendicular velocity starts decreasing where it was previously increasing, or vice versa.

5.2 Automatic Variable Configuration

Most control functions will have some parameters, allowing the system to control their shape, like the a, b and c parameters for the Gaussian function. However, any set of precomputed values will fall short in some situations. Thus, it is better to calculate appropriate values at runtime. Automatic parameter optimization is an important area of AI research, which has yielded various well-known approaches. These can also be applied to the problem at hand. However, some knowledge of how the parameters affect the maneuver may be useful.

The value of a should be calculated as a function of, amongst others, the minimum evasion distance necessary and the velocity difference between the active individual and the passive individual. The value of b should depend mostly on strategic considerations — for example, in a racing environment, a negative b would be used to represent a blocking maneuver after overtaking: the active individual "swoops in" sharply and cuts off the passive individual. However, a positive b would represent "sticking" to the actual course a little longer to be able to use the slipstream generated by the passive individual.

The value for c is the most difficult one: if this parameter is set too low, activating the algorithm may result in an unrealistic trajectory with high perpendicular velocities. However, setting it too high will result in unnecessary maneuvering and increasing recovery time. Also, if the value of c is not chosen carefully, this may result in "snapping" behavior: the displacement calculated as the trajectory modification behavior is activated is not as close to 0 as it should be. This is because the c dictates that an overtaking maneuver is started as soon as the individuals are m meters apart, while the impending collision is only detected when they are n meters apart. Now if $n < m$, the

offset $f(d)$ at the start of the maneuver is not 0, thus causing an extremely unrealistic instantaneous perpendicular movement. The value of c, then, is mostly a function of the difference in velocities: if this difference is higher, the c value should also be higher to allow for a longer "preparation distance" and eliminate sharp turns and high perpendicular velocities. However, if the velocity difference is lower, the c should be kept low, since the maneuver will otherwise "wake up" too late and snap into activity. A possible function for calculating c would be $c = k * \Delta v$, where Δv is the difference in velocity.

6 Results and Evaluation

We tested our theories by using the aforementioned control functions in both ideal and normal conditions. Two sets of tests were performed. One set deals with a synthetic environment with only one straight road, so that perpendicular displacement is only caused by the control function. In the other set of tests, data from an individual tracked at the Zandvoort circuit was used to test the interaction between real and virtual individuals in a realistic environment.

Table 1. Algorithm settings for testing environments

Test	Test1	Test2	Test3
a_1	5.0	5.0	5.0
a_2	-5.0	10.0	-25.0
b	0.0	0.0	0.0
c	0.5	2.5	1.0
v_a	10.0	50.0	70.0
v_p	5.0	25.0	60.0
v_{diff}	5.0	25.0	10.0

Table 2. Velocity changes resulting from the algorithms

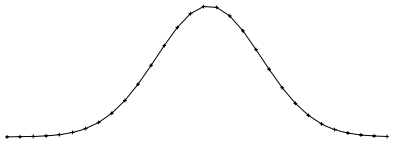
Parameter	Test1	Test2	Test3
a_1	5.0	5.0	5.0
a_2	-5.0	10.0	-25.0
b	0.0	0.0	0.0
c	0.5	2.5	1.0
v_a	10.0	50.0	70.0
v_p	5.0	25.0	60.0
v_{diff}	5.0	25.0	10.0
1D Contr. Func. $max(v_{err})$	0.45	0.18	0.066
2D Contr. Func. $max(v_{err})$	9.99	-3.99	24.97
2D Contr. Func. $avg(v_{err})$	3.19	1.46	8.70

6.1 Specially Prepared Environments

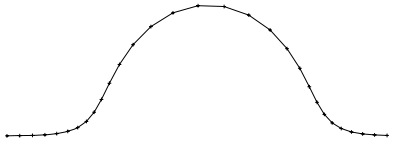
Two vehicles were created, one behind the other. They were both instructed to follow the road's middle line. Both vehicles were given a constant velocity, and that of the rearmost was the highest, necessitating overtaking maneuvers. In Table 1, the configuration variables for three different tests are shown. The a_1 and a_2 mean that these values are set for the perpendicular and longitudinal direction, respectively, while v_a and v_p refer to the velocities of the active and passive individuals. In each test, the b and c values were kept the same for the longitudinal and perpendicular control functions.

In Test 1, a small velocity difference is tested in a low velocity environment. Test 2 represents a large velocity difference in a medium-velocity environment, while Test 3 shows a small velocity difference in a high-velocity environment.

Obviously, it is hard to render the resulting 2-D time varying dataset on paper. Even though it is easy to visualize the trajectory itself, it is harder to visualize the velocities



(a) Test 1 using 1-D Gaussian control function



(b) Test 1 using 2-D Gaussian control function

Fig. 4. Subsampled trajectories for Test 1, synthetic environment. Vertically upscaled 5 times for visual inspection.

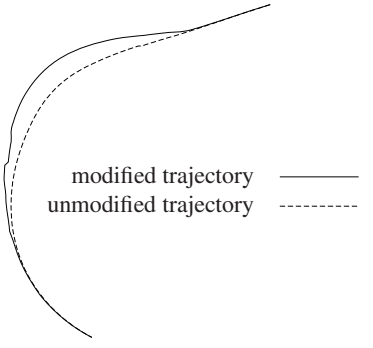


Fig. 5. Resulting Trajectories for one test run (1-D control function, realistic environment)

that go with the datapoints. Therefore, graphs were created using regular sampling from the output data: by rendering only one out of every n datapoints in the graph, velocities can be inferred from the distances between consecutive points. To allow for easier visual inspection, the perpendicular (vertical) direction is upscaled 5 times in the graphs. The shape difference between the 1-D and 2-D control function graphs in Fig. 4 is purely due to the addition of a longitudinal control function. Obviously, the maneuver resulting from the 2-D control function has more chance of properly avoiding a collision, but this comes at a price: its longitudinal velocity will be somewhat higher at some points, thus increasing the risk of violating physical restrictions, as seen in Table 2.

6.2 Realistic Environments

The performance of the algorithms was also evaluated using data tracked at the Zandvoort circuit. The dataset contains some noisy measurements, which shows in the output. By using subsampling and interpolation techniques, new datasets were created from the only available dataset for individuals with the same trajectory, but a different velocity. This was done by multiplying the basic velocity by different factors. Also, a number of datapoints was trimmed from the start of the dataset, so that each vehicle would have a different starting point. These types of datasets can be combined in a sheer endless number of ways, so only some representative results are presented.

When plotting the results, it turns out they are somewhat more erratic, largely a result of the noisy nature of the dataset. For example, in Fig. 5, the modified trajectory is not completely smooth. This is caused by incorrect measurements of location of the passive individual. This brings up an interesting and important point: control functions are highly sensitive to noise in the measured data. If control functions are to be successfully applied, prefiltering of location and heading data is absolutely required.

The fact that most of the errors in the resulting trajectory are a result of the noise instead of the algorithm can be proven by the “back-calculation” of the Gaussian. In

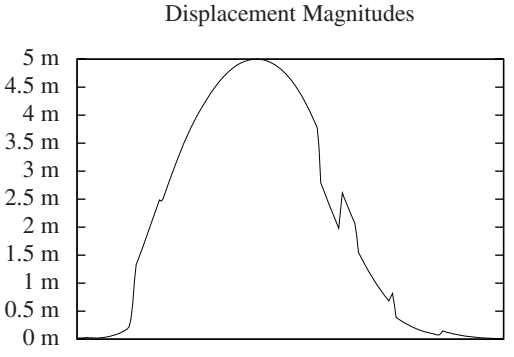


Fig. 6. The magnitudes of the actual offsets for one test

the areas where the input parameter to the Gaussian function increases or decreases monotonously, the shape of the graph should be exactly that of a (stretched) Gaussian. Indeed, it turns out that calculating $m(x_{err}, y_{err})$ for each resulting displacement, where $m(x_{err}, y_{err}) = \sqrt{x_{err}^2 + y_{err}^2}$ and x_{err} and y_{err} are the displacements in x and y direction, respectively, yields a graph that is globally shaped like a Gaussian, but contains some spikes, as shown in Fig. 6. The smallest of these spikes can be explained by the fact that the input parameter is not completely monotonous, but the larger spikes result from noisy data. Non-monotonicity of the input parameters is not problematic, since the Gaussian approach was designed to deal with it, but non-continuity *is* a serious problem.

7 Conclusions and Future Work

Some conclusions can be drawn from our research:

- The 1-D Control Function provides a useful and adequate modification strategy for some environments, *if* its parameters are correctly configured. In the Gaussian control function, especially c, which determines the length of the curve, should be carefully adjusted, preferably at real-time, based on the current circumstances.
- The 2-D control function algorithm is somewhat more dangerous and may increase or decrease velocities by large amounts if the parameters are not very carefully configured, as seen in Table 2. The results may be nice, since the shape of the perpendicular control function can be modified to better suit the current circumstances, but the configuration is somewhat harder, and failure will often be dramatic.

The most important opportunities for future work is the research of different control functions. The Gaussian function is an adequate control function for some environments, but there is no doubt that there are functions better suited to any number of different circumstances. Some of these functions may be created by first determining the desired shape and then performing Fourier analysis on the resulting graph. While this would yield a function that violates the finite-support requirement, it is trivial to reduce the function to only one period of the signal. It is important to realize that control

functions are more useful if their shape can be configured using parameters determined at real-time to adapt to the current circumstances. The automatic configuration of these variables is another interesting area for future work.

References

1. DeCarpentier, G.J.P., Bidarra, R.: Behavioral assumption-based prediction for high-latency hiding in mobile games. In: Proceedings CGAMES 2007 (2005)
2. Delaney, D., Ward, T., Mcloone, S.: On reducing entity state update packets in distributed interactive simulations using a hybrid model. In: Proceeding of the 21st IASTED International Multi-conference on Applied Informatics, pp. 10–13 (February 2003)
3. Gayle, R., Sud, A., Lin, M., Manocha, D.: Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, pp. 3777–3783 (2007)
4. Gossweiler, R., Laferriere, R., Keller, M., Pausch, R.: An introductory tutorial for developing multiuser virtual environments. Presence: Teleoperators and Virtual Environments 3(4), 255–264 (1994)
5. System for Simulating Events in a Real time Environment, U.S. Patent Application No. 12/106,263; for more information contact iOpener Media, http://www.iopenermedia.com
6. Li, S., Chen, C., Li, L.: A new method for path prediction in network games. Comput. Entertain. 5(4), 1–12 (2007)
7. Musse, S., Thalmann, D.: A hierarchical model for real time simulation of virtual human crowds. IEEE Transactions on Visualization and Computer Graphics 7(2), 152–164 (2001)
8. Pantel, L., Wolf, L.: On the suitability of dead reckoning schemes for games. In: NetGames 2002: Proceedings of the 1st workshop on Network and system support for games, pp. 79–84. ACM, New York (2002)
9. Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. J. ACM 41(4), 764–790 (1994)
10. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Papers, pp. 1160–1168. ACM, New York (2006)
11. Van Den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of multiple agents in crowded environments. In: SI3D 2008: Proceedings of the 2008 symposium on Interactive 3D graphics and games, pp. 139–147. ACM, New York (2008)
12. Project website, http://graphics.tudelft.nl/Game_Technology/vandenBerg

CA-LOD: Collision Avoidance Level of Detail for Scalable, Controllable Crowds

Sébastien Paris, Anton Gerdelan, and Carol O'Sullivan

GV2, Trinity College Dublin, Ireland
{Sebastien.Paris,gerdelaa,Carol.OSullivan}@cs.tcd.ie
http://gv2.cs.tcd.ie/metropolis

Abstract. The new wave of computer-driven entertainment technology throws audiences and game players into massive virtual worlds where entire cities are rendered in real time. Computer animated characters run through inner-city streets teeming with pedestrians, all fully rendered with 3D graphics, animations, particle effects and linked to 3D sound effects to produce more realistic and immersive computer-hosted entertainment experiences than ever before. Computing all of this detail at once is enormously computationally expensive, and game designers as a rule, have sacrificed the behavioural realism in favour of better graphics. In this paper we propose a new Collision Avoidance Level of Detail (*CA-LOD*) algorithm that allows games to support huge crowds in real time with the appearance of more intelligent behaviour. We propose two collision avoidance models used for two different *CA-LOD*s: a fuzzy steering focusing on the performances, and a geometric steering to obtain the best realism. Mixing these approaches allows to obtain thousands of autonomous characters in real time, resulting in a scalable but still controllable crowd.

Introduction

Our work contributes to the *Metropolis* project: an interdisciplinary effort involving computer graphics, engineering and neuroscience researchers aiming to develop a multisensory Virtual Dublin populated by scalable crowds with a high level of realism. Building on earlier work [1], the project is aiming to simulate large, scalable crowds while optimising the level of variety in animation, appearance and sound. Most relevant to our work, are the objectives to:

- Provide easily integrated populace and traffic for Virtual Environments that will be scalable: at the architectural level (i.e., from PC to large cluster exploiting multi-core

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 13–28, 2009.

architectures); the user level (i.e., from single user to massively multi-player); and the crowd level (i.e., thousands of people simulated in real-time).
- Increase the realism of these large crowds of people by adding variety in animation, appearance and sound, driven by perceptual models and metrics.
- Add real meaning to the simulations by endowing individual crowd members with appropriate, sentient behaviours that are based on cognitive and sociological models.
- Optimise the computational and memory resources necessary to achieve these real-time multisensory displays and ameliorate any consequential anomalies by applying principles of human perception and exploiting human perceptual limitations.

We have been investigating several ways in which to simulate plausible behaviours that take context into account [2]. In this paper we describe a candidate methods for simulating the navigation behaviours in Metropolis. Our aim is to push away the crowd behaviours' computational limitations without sacrificing the animation's realism and interactivity.

We first propose in *Section 2* a modular autonomous agent architecture focusing on the human's ability to move, and introduce the notion of Collision Avoidance Level Of Detail (*CA-LOD*). In *Section 3* we detail the fuzzy steering model that is used as a fast but unrealistic *CA-LOD*. In *Section 4* we propose another *CA-LOD* which improves the realism at the expense of the computation performance. Then in *Section 5* we evaluate the performance of each *CA-LOD*, and propose an extrapolation of the possible mixes to simulate huge crowds in real-time. We finally conclude in *Section 6* and present the promising future work for the presented framework.

1 Related Work

Animating large crowds of people in real time requires to address many problems: designing the simulation (either the virtual environment or the population), rendering the environment and the crowd, simulating the crowd behaviours, and animating the population to reflect its actions. Solutions to the rendering bottleneck are now well known, either by reorganising the scene-graph for efficient culling to improve static environments rendering [3], or by introducing graphical *LOD*s to improve the rendering of specific objects [4]. Specific solutions to crowds rendering also exist to greatly reduce the number of polygons to render each agent by applying pre-rendered textures to simple shapes using impostors or similar techniques [5]. All of these solutions focus on the computation done by the GPU while letting apart the behavioural algorithms which are generally executed by the CPU.

Indeed, animating hundreds to thousands people in real time requires at least simulating their movement in the environment. Animation oriented solutions exist, such as crowd patches [6] that define areas with pre-generated animations which can then be connected to create a large environment. Nevertheless, this kind of solution does not fit to game applications for two reasons. First, this method is too restrictive concerning the manageable environments which must fit to the input patches. Second, the crowd cannot be controlled in real-time to handle users' interactions, for example to adapt the autonomous agents' trajectories in order to avoid the player's agent. Another method, combining simulation and graphical *LOD*s [7], allows for more crowd reactivity but still

constrain the behaviours variety by pre-computing possible paths between a limited set of starting / destination points.

To handle the need for highly reactive and varied crowds, the behaviours can be managed using a microscopic simulation model where each agent is endowed with its own decision abilities [8]. The decision can be divided in two main categories. First, the high level behaviours manage the agent's mid term goals and select its immediate action by using a script, a set of rules, or even decision networks [9]. An optimisation of this decision layer, called *LOD-AI* (for LOD Artificial Intelligence), consists in limiting the decision process for the agents which are far from the camera [10]. Second, the low level behaviours manage the agent's ability to move thanks to two complementary mechanisms: the path planning which globally evaluates the movement through the static environment to reach a defined target; and the collision avoidance which locally adjusts the movement in order to avoid dynamic obstacles while following the previously computed path. When considering large crowds of people, both of these mechanisms are computationally critical. The path planning, which highly depends on the environment's scale, can be improved by using a hierarchical evaluation [11]. The collision avoidance is generally evaluated in games by fast rule-based algorithms [12,13] that allow simulating very large crowds but with poorly convincing behavioural results: wall crossing or late adaptation, mainly because of the difficulty to simultaneously handle multiple potential collisions with static and dynamic entities. A geometric approach [14] enhances the produced trajectory realism at the expense of the computation time, by applying cognitive principles, such as anticipation, while considering all the obstacles at the same time. The particle based approach [15] is more used for high density simulations and is less adapted to animation because of the oscillations in the produced movement and the lack of anticipation in the decision.

We propose in this paper a novel approach focusing on the improvement of the collision avoidance computation performance and realism. The proposed model takes place as a specialised *LOD-AI*, called Collision Avoidance LOD (*CA-LOD*), and is integrated to a modular autonomous agent architecture. We then define three *CA-LOD* represented by a special level without collision avoidance, and two levels of increasing complexity and realism.

2 Overall Approach

2.1 The Simulation Model

We propose to simulate crowds of people using a microscopic simulation model. Thereby, each virtual human composing the crowd is represented by an autonomous agent endowed with individual decision abilities. As shown in *Figure 1*, the first decision mechanism represents *high level behaviours* which select the agent's destination and desired speed. This decision layer will be left apart as it is out of the scope of this paper.

The next mechanism is the *path planning*, that analyses the environment's topology in order to produce and update the path to the selected destination. The environment description we use is based on a Delaunay's triangulation [16], thus the path planning produces a set of edges to traverse in order to reach the destination (see *Figure 2(a)*).

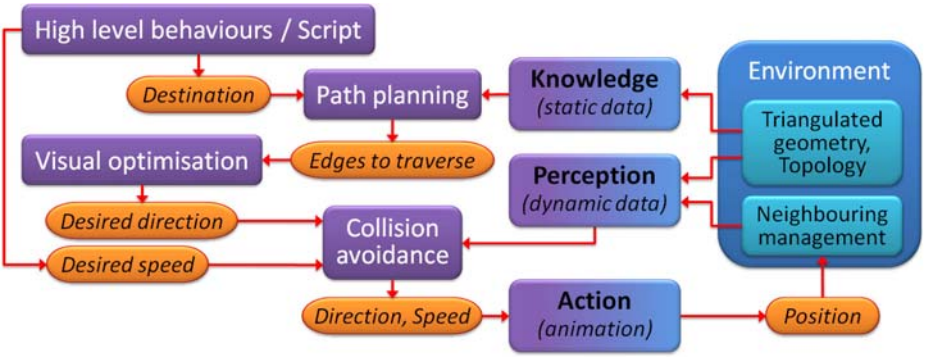


Fig. 1. Virtual human's simulation architecture. In purple, the agent's successive decision mechanisms; in orange, the transferred data.

Then, a complementary process, called *visual optimisation*, analyses the produced path to extract the optimal direction of movement, resulting in an overall path smoothing as shown in *Figure 2(b)*.The triangulated path given to each character as its primary navigation instruction handles all of the static obstacle avoidance, and in general the animated characters follow this visually optimised path directly. Because the path is directly derived from the *geometry* it does not take dynamic obstacles (other moving characters) into account. We therefore need to augment this system with an obstacle-avoidance module.

Our agent architecture is analogous to the classic robot *stack architecture* [17]. As can be seen in *Figure 1*, the collision-avoidance module needs to operate as a closed system that, in only those special conditions where collision-avoidance is necessary, overrides movement instructions from the *visual optimisation* layer. The entities to avoid, i.e. the walls and other agents, are retrieved using a perception algorithm that dynamically extracts the visible elements depending on the agent's direction and field of view

(a) Edges to traverse. (b) Resulting visual optimisation.

Fig. 2. Path planning in Trinity College Dublin. The underlying subdivision is represented with traversable gray edges and obstacle red edges; the resulting path is shown in blue.

as well as the surrounding topology. Finally, the collision avoidance mechanism provides the final direction and speed of movement that are used to animate the agent and to compute its new position.

To conclude, we propose an autonomous agent model managing the human's ability to move. This model allows individual decision, thereby taking into account individual factors such as the destination, the desired speed, or even different preferences for path planning [11]. Moreover, the modularity of this model, with well identified communication between several independent mechanisms, allows one component algorithm to be changed while keeping the entire architecture functional.

2.2 Obtaining a Scalable Model

In order to simulate very large crowds it is essential to achieve the best computation performances. Because the collision avoidance algorithm is highly dependent on external conditions, it needs to be refreshed with a relatively high frequency compared to the other behaviours. This fact, combined with the cost of the algorithm itself, results in a computation bottleneck for large crowds simulation.

The modular nature of our agent architecture gives us some flexibility in terms of the algorithm used for collision-avoidance behaviour. We propose three different collision avoidance models, assigned to three successive levels of detail of increasing computation performance and decreasing realism (*Figure 3(a)*):

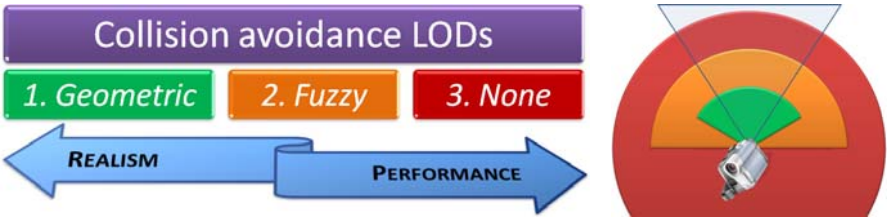
LOD 1 (high). Our geometric model takes into account the entire entity's perception of its immediate surrounding by computing an intermediate geometric representation. The resulting movement is time consistent as other entities' moves are anticipated, providing a high level of realism.

LOD 2 (medium). Our fuzzy logic model only takes into account one potential obstacle at a time. It uses a rule table to quickly make decisions and smoothly interpolate movement instructions. The resulting movement is collision free for low densities of people, but can result in interpenetration for higher densities, providing a restricted level of realism.

LOD 3 (low). The lowest level of detail removes the collision avoidance. This method simply applies the input speed and direction, resulting on an exact path following with agents inter-penetrations, providing a poor level of realism.

In the same way as for geometric LOD algorithms, the Collision Avoidance Level Of Detail (*CA-LOD*) can be chosen depending on the distance to the camera (*Figure 3(b)*): the larger the distance to the camera, the lower the *CA-LOD*; the same principle may be applied for angular distance to the camera direction.

With *CA-LOD*s we propose a novel crowd simulation model, allowing very large crowds to be simulated while keeping precise control over the simulated agents. In the following sections, we detail the collision avoidance models needed for the two higher *CA-LOD*s outlined above, and we present a preliminary comparative benchmarking of these systems.



(a) The three LODs composing the collision avoidance model.

(b) Possible LOD assignment around the camera.

Fig. 3. Scalable collision avoidance principle. In (b) the triangle shows the camera field of view.

3 Fuzzy Steering

We have made use of fuzzy controllers extensively in our previous work [18,19]. Fuzzy logic has been employed by other agent simulations for its flexible nature - easily expanding to accommodate more complex input variables [20]. Our fuzzy controller in the Metropolis traffic simulation is a two-component system comprising a route-following controller and a dynamic obstacle avoidance controller; these systems are counterbalanced and roughly follow a route of way-points whilst simultaneously avoiding static and dynamic obstacles. To adapt this for our crowd-navigation system, we are using the obstacle avoidance component to only handle special cases where dynamic obstacle avoidance is required. Fuzzy logic allows us to represent a *partial truth*, or imprecise values between *completely true* and *completely false*. This gives us a mechanism for discriminating imprecise or changing data into a small group of overlapping *fuzzy sets*. From this foundation we can create very simple judgement-based reasoning, or *fuzzy inference*, to deal with complex real-world data; mimicking human decision-making. Fuzzy Logic systems require very little computational overhead, and can also produce smooth transitional outputs. Fuzzy Logic is therefore an ideal candidate for modelling human behaviour in large-scale, real time simulations.

In *Figure 4* we consider an example scenario where an agent controls a character that is facing up the page. There are two obstacles nearby; *Obs.0* and *Obs.1*. We have classified the environment into three overlapping sets for distance (*NEA, MED*, and *FAR*), and angle from current heading (*NAR, MID*, and *WID*). Obs.0 is outside our maximum range of consideration, and is thus ignored, Obs.1 is our *nearest obstacle* and straddles both NEA and MED distances, and is in-between both MID and WID angles. In this case we evaluate all four combinations of rules, and blend the results together - weighted by the degree of membership in each of the distance and angle sets. This blending or *aggregation* allows us to smoothly transition between each rule and provides us with very smooth curves of motion that then require no motion post-processing before animations can be applied.

The obstacle-avoidance system considers the *change in heading angle* and *distance* to the nearest obstacle. Both of these systems classify the real inputs into overlapping *rough sets*, where they can be classified in human terms for a quick look-up-table type decision. The mapping of these inputs is designed so that a series of rules will progressively move the subject character away from its route, avoid the obstacle, then finally

return to its route. Figure 4 illustrates this scenario, and shows the range of distances and angles used as input. Table 1 provides our design for fuzzy input set mappings. We produce fuzzy sets for classifying (or *fuzzifying*) real inputs into our rough sets, which are illustrated in Figure 5.

Because we want to represent a range of different characters with the same fuzzy system, we have expressed our outputs in terms of a characters's *maximum velocity* (v_{max}). The steering adjustment outputs are also expressed relative to the characters's current speed; such that resultant movement vector of the character is to some extent scalable, and we can expect similar rates of turn at different speeds. Fuzzy Output set mapping functions for speed and steering are illustrated in Figures 6 *(a)* and *(b)*, respectively.

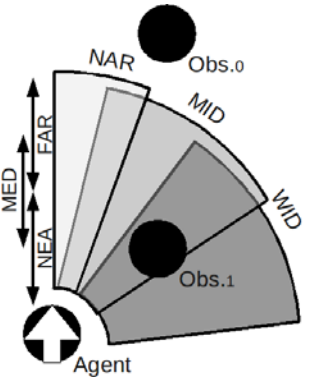


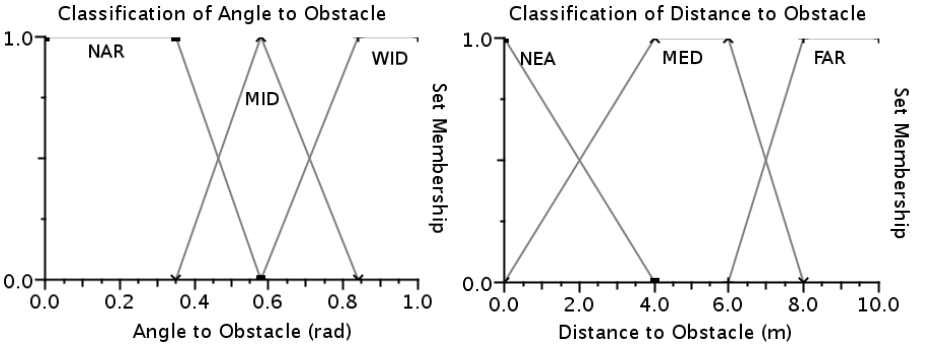
Fig. 4. A scenario illustrating the design of input distances and angles to the fuzzy obstacle avoidance system

The labour-intensive task is then to design fuzzy rules. These map all of the different combinations of fuzzy input values to fuzzy outputs. Whilst we can manually tune these rules to produce a satisfying output for one particular character, the fuzzy systems need to be recalibrated for every character with different performance characteristics or physical dimensions. As we wish to simulate a large variety of characters, this task becomes a serious constraint-based problem which we are tackling in our ongoing works by way of an automatic-calibration and self-training system [21,22]. Our initial rules are provided in Tables 2(a)-2(b). We note that not all of our output sets have been used in these rules, and that there is certainly scope for designing or evolving more balanced rule-sets.

Once all of the rules have been evaluated we use an aggregation *centre-of-gravity* function to merge the outputs for each system. Equation 1 gives us the aggregation function for velocity. To obtain the real output *defuzzified* velocity (v_{defuzz}) we take the membership values (mem_i) for each fuzzy output set (from *ZER* to *TOP*) that we evaluated with our FAMMs, multiply each by its set centre value (v_i) (from Figure 6),

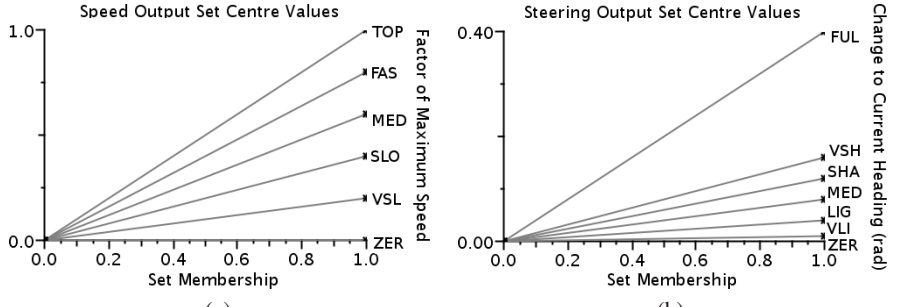
Table 1. Fuzzy Input Term Definitions for obstacle avoidance. An optimisation here is that we are only considering angles from the current heading of a character, on *one side*. We simply flip the same fuzzy sets over to evaluate obstacles on the other side of the character - this then requires half as many fuzzy sets and simplifies calculation.

Real Term	Rough Set Value Range	Fuzzy Term
Wide Arc	$> 0.58rad$	WID
Mid-range Arc	$0.35 - 0.84rad$	MID
Narrow Arc	$0 - 0.58rad$	NAR
Far Distance	$> 6m$	FAR
Medium Distance	$0 - 8m$	MED
Near Distance	$0 - 4m$	NEA



(a) (b)

Fig. 5. Fuzzy Input Set Membership Functions for classifying the *angle (a)* and *distance (b)* to the nearest obstacle in fuzzy terms. Angles here are absolute radians to the left or right of the current heading of a character, so that obstacles on the left hand side of a character are treated the same as obstacles to the right.



(a) (b)

Fig. 6. Fuzzy Output Value centres for obstacle avoidance desired *speed* as a factor of maximum speed and *steering adjustment*; a modifier. The defuzzified output speed factor will be multiplied with the character's top allowable speed to produce a desired speed as a crisp number. The steering modifier is subtracted from the route-following steering factor.

Table 2. 3x3 Fuzzy Associative Memory Matrices (FAMMs) for Desired Speed (a) and Steering (b). These tables are the core of the fuzzy system, and express the rules that match each possible combination of fuzzy input angles and distances to a fuzzy output. We can see that if an obstacle is a *near* distance at a *mid-range* angle then we will travel at a *very slow* speed and make a *medium* turn away from the obstacle. This system provides a very fast rule look-up table for real-time operation.

(a) Desired Speed.

	NEA	MED	FAR
NAR	ZER	VSL	SLO
MID	VSL	SLO	MED
WID	SLO	MED	FAS

(b) Desired Steering.

	NEA	MED	FAR
NAR	SHA	MED	VLI
MID	MED	VLI	ZER
WID	VLI	ZER	ZER

sum the result together, and finally divide this by the sum of all of the centre values.

$$v_{defuzz} = \sum_{i=top}^{zer} (mem_i.v_i) \; / \; \sum_{i=top}^{zer} v_i \tag{1}$$

4 Geometric Steering

The geometric steering model avoids collisions based on the following assumptions:

- Resolving collision avoidance consists of selecting the instantaneous speed and direction of movement;
- The selected movement must take into account the static and dynamic obstacles, as well as the desired movement of the entity;
- The selected movement must be as consistent as possible through time, avoiding decision oscillations.

The proposed model is similar to the potential fields because it merges multiple collisions sources in a single representation. Nevertheless, our model adds two major advantages by allowing individual reasoning and subdividing the environment in a more accurate way than with grids (which are commonly used for potential fields). Three successive steps are necessary to take the optimal movement decision:

1. Construct a local geometric representation of the agent's surroundings, merging information about the static and dynamic obstacles while computing area potentials;
2. Extract from the previous representation the area with the best potential, considering the environmental constraints and the agent's desired movement;
3. Constrain the agent's movement by the movement allowed in the previously selected area.

Even if this approach is based on our previous work [14] we have improved this technique, notably considering the way to construct the temporary representation. Let us now detail each computation step.

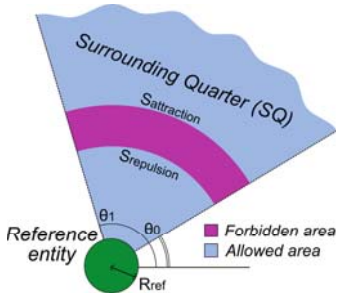
4.1 The Local Geometric Representation

The local geometric representation subdivides the agent's surrounding in quarters storing the movement's constraints for a direction interval (*Figure 7*(a)). The following data are stored for each surrounding quarter SQ:

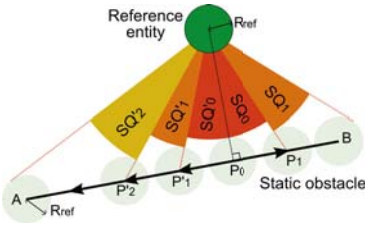
The angle interval. $[\theta_0;\theta_1]$ defining the range of directions covered by the quarter.

The time interval. t defining the time range $[R_t;R_{t+1}]$ with $R_t = \begin{cases} 0 & \text{if } t = 0 \\ k^t & \text{otherwise} \end{cases}$ corresponding to the quarter data; the time range follows an exponential scale to obtain a more precise subdivision in the near future while allowing long time prediction (we use $k = 2$ in our simulations).

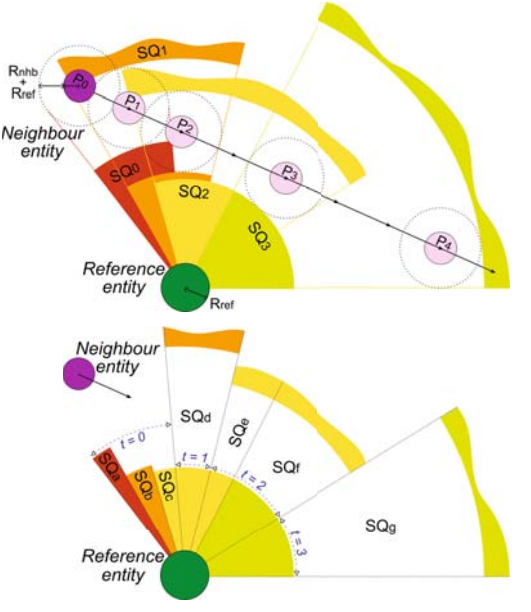
The repulsion speed. $Sr \geq 0$ defining the maximal allowed speed to avoid collisions.



(a) Surrounding quarter principle.



(b) Static obstacle subdivision.



(c) Dynamic obstacle subdivision: on top, per time interval quarter extraction; on bottom, resulting intersected quarters.

Fig. 7. Local representation used for the geometric collision avoidance. For simplification purpose, the repulsive and attractive distances are represented instead of the corresponding speeds.

The attraction speed. $Sa \geq 0 \in$ defining the minimal necessary speed to avoid collisions.

The potential factor. F which scores the quarter's attractiveness for the agent's next move (the lower the best); we will see later that this factor reflects the agent's *effort* to avoid a collision in this quarter.

Initially, only one quarter exists with the following values: $[\theta_0;\theta_1] = [0;2\pi]$ which covers the entire agent's surrounding; $Sr = Sa = undefined$ which is a special value meaning that the quarter does not constrain the agent's speed; $F = 0$ which is the neutral potential value; and $t = undefined$. When a new quarter SQ_n is inserted, the existing overlapped quarters SQ_o are intersected to produce the appropriate subdivision, and their data are updated based on these rules:

- **if** $\begin{pmatrix} \vee & t_o > t_n & \wedge\ Sr_o = undefined \\ \vee & (\ t_o = t_n \vee t_o = undefined \vee t_n = undefined\ ) & \wedge\ Sr_n \neq undefined \\ & & \wedge\ Sr_n < Sr_o \end{pmatrix}$ **then** $Sr_o \leftarrow Sr_n$
- **if** $\begin{pmatrix} \vee & t_o > t_n & \wedge\ Sa_o = undefined \\ \vee & (\ t_o = t_n \vee t_o = undefined \vee t_n = undefined\ ) & \wedge\ Sa_n \neq undefined \\ & & \wedge\ Sa_n > Sa_o \end{pmatrix}$ **then** $Sa_o \leftarrow Sa_n$
- **if** $\left(t_n \neq undefined \wedge t_n < t_o\right)$ **then** $t_o \leftarrow t_n$
- $F_o \leftarrow F_o + F_n$

Static obstacle subdivision. We propose two subdivision methods, one for the static obstacles (typically walls) and the other for the dynamic obstacles (other agents). The static obstacles are subdivided in quarters around the projected point P_0 of the agent's position (*Figure 7*(b)). The subdivision follows an exponential step on the same way as the exponential time intervals. Defining $[AB]$ the obstacle segment and S_{ref} the agent's desired speed, each subdivision point P_i is obtained by the following formula:

$$\begin{array}{l} P_i = P_0 + \vec{\Delta_i} \\ P'_i = P_0 - \vec{\Delta_i} \end{array} \text{ where } \vec{\Delta_i} = k^{i-1}.\frac{S_{ref}}{\| \vec{AB} \|}.\vec{AB} \text{ for } i > 0$$

The successive pairs of points $(P_i; P_{i+1})$ are used to compute each quarter SQ_i; if both points are not on the segment $[AB]$ they are ignored; if one point is not on $[AB]$, the corresponding quarter's angle is replaced by the external tangent to a circle with the same radius as the agent and centred on the corresponding segment's extremity. Finally, each quarter's internal data are computed on this way:

- $t = undefined$ because neither quarter corresponds to predicted data, but only to a geometric subdivision.
- $Sr_i = \| \vec{P_{ref}P_i} \| - R_{ref}$ with P_{ref} the position of the reference agent.
- $Sa = \infty$ because the obstacle cannot be crossed.
- $F_i = \begin{cases} 0 & \text{if } S_{ref} \leq Sr_i \\ 1 - Sr_i/S_{ref} & \text{otherwise} \end{cases}$

Dynamic obstacle subdivision. The dynamic obstacles' quarters are obtained by anticipating the movement of a neighbour considering his current velocity $\vec{V_{nhb}}$ (top of *Figure 7*(c)). A predicted position P_t is computed for each time value $t > 0$ by $P_t = P_0 + k^{t-1}.\vec{V_{nhb}}$. Then, each pair of successive position $(P_t; P_{t+1})$ is used to configure a quarter SQ_t: the angles are the outside tangents from the reference agent's position P_{ref} to the circles defined by P_t and P_{t+1} with a radius $R_{sum} = R_{nhb} + R_{ref}$. Each quarter's internal data are computed on this way, assuming $Dmin_t = \min\{\| \vec{P_{ref}P_t} \|; \| \vec{P_{ref}P_{t+1}} \|\}$ and $Dmax_t = \max\{\| \vec{P_{ref}P_t} \|; \| \vec{P_{ref}P_{t+1}} \|\}$:

- t is the value used to compute the quarter.
- $Sr_t = (Dmin_t - R_{sum})/k^t$
- $Sa_t = \begin{cases} \infty & \text{if } t = 0 \text{ (neighbour's current position is uncrossable)} \\ (Dmax_t + R_{sum})/k^{t-1} & \text{otherwise} \end{cases}$
- $F_t = \begin{cases} 0 & \text{if } S_{ref} \leq Sr_t \vee S_{ref} \geq Sa_t \\ \min\{S_{ref} - Sr_t; Sa_t - S_{ref}\}/(S_{ref}.k^t) & \text{otherwise} \end{cases}$

4.2 Selecting the Best Movement Area

Once all the surrounding obstacle have been processed and merged into the geometrical representation, the best movement area can be obtained by selecting the quarter with the lowest potential factor. In order to take into account the agent's desired movement

direction θ_{ref}, an additional direction change factor F_d is computed. The relative importance $\lambda \geq 0$ of the direction change toward the speed change is then applied in order to obtain the finally compared factor $F_f = F_q + \lambda.F_d$

The direction-change factor reflects the agent's effort to deviate from his optimal trajectory (*Figure* 8). It is sensible to a maximal allowed direction change $0 < \theta_{max} \leq \pi$, which can be configured depending on the agent's physical constraints for example. Defining θ_{SQ} the angular distance between a quarter and D_{ref} (which may be null if the quarter encloses D_{ref}), the direction-change factor is computed this way:

$$F_d = \begin{cases} \infty & \text{if } \theta_{SQ} > \theta_{max} \\ \theta_{SQ}/\theta_{max} & \text{otherwise} \end{cases}$$

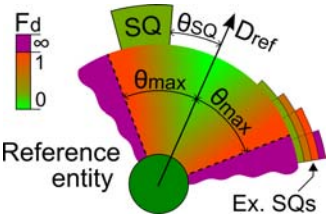


Fig. 8. Direction-change factor computation

4.3 Final Movement Decision

When the best surrounding quarter is selected, obtaining the final movement is trivial. The collision free direction is the nearest direction to D_{ref} that is covered by the quarter. The final speed S_{fin} is computed based on the selected quarter's data, thanks to this algorithm:

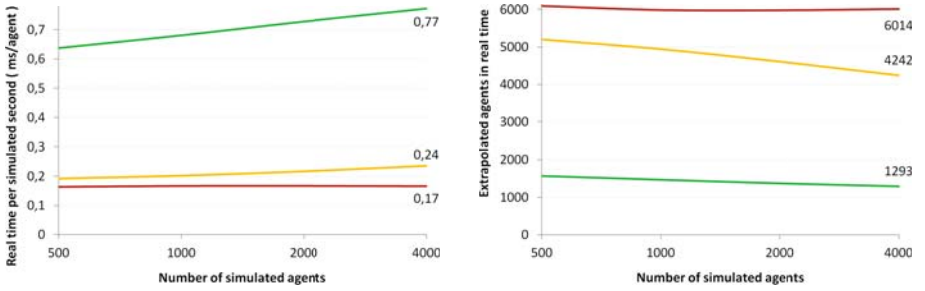
```
1  if S_ref ≤ Sr ∨ S_ref ≥ Sa then
2      S_fin ← S_ref
   else
3      Dec ← S_ref − Sr
4      Acc ← Sa − S_ref
5      if Dec ≤ Acc then
6          S_fin ← Sr
       else
7          S_fin ← Sa
       end
   end
```

Lines $1-2$ manage the case where the agent's movement is not constrained, allowing the desired speed. In the other case, the necessary deceleration (l.3) or acceleration (l.4) are computed. The less constraining option is then chosen (l.5), resulting in a slower (l.6) or faster (l.7) speed.

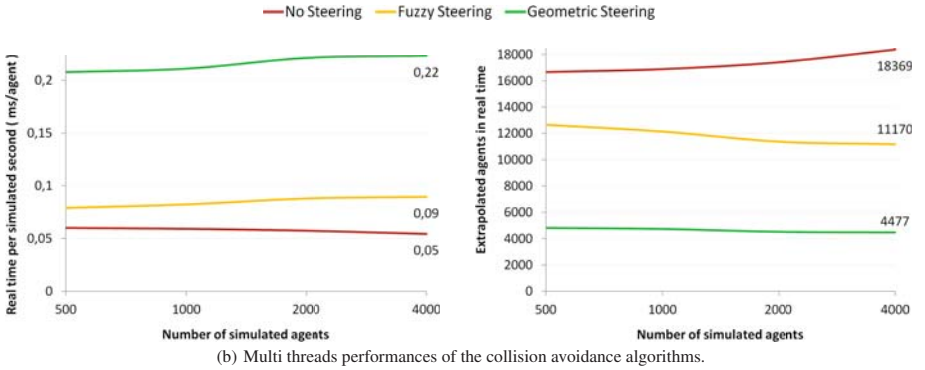
5 Results

The proposed virtual human model has been implemented into a crowd animation tool called *Metropolis*. This tool is used to animate huge crowds of people, with a final objective to simulate Dublin city in real time with highly capable virtual humans. Each collision avoidance model has been tested in Metropolis, but the LOD selection algorithm is still a work in progress.

We have run series of tests with each collision avoidance model to obtain their respective performances, allowing us to have an indication of the benefit to mix these models in the final *CA-LOD* framework. These experiments only take into account the behavioural computation, the rendering being deactivated to not influence the results.



(a) Single thread performances of the collision avoidance algorithms.



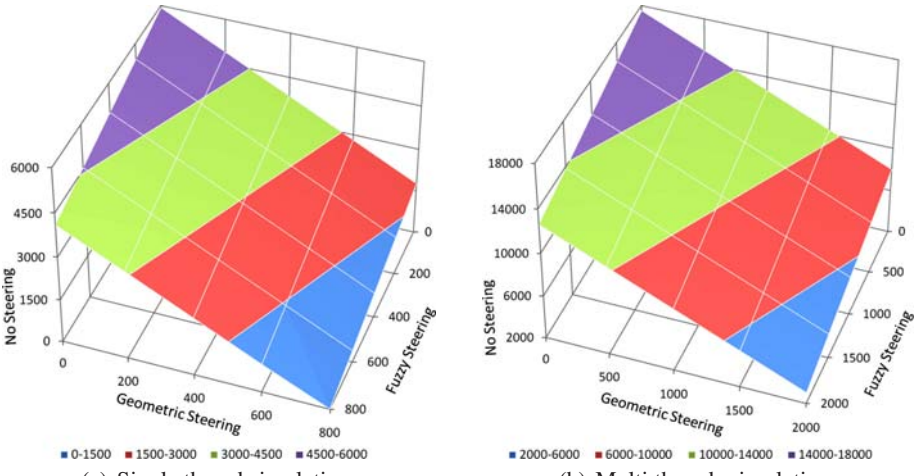
(b) Multi threads performances of the collision avoidance algorithms.

Fig. 9. Performances of each collision avoidance algorithm

Each agent is randomly generated in a navigable area of the Trinity College virtual environment (approximately $2km^2$), and pick a random target in the same way (which is renewed when reached).

Figure 9 shows the performances obtained on an Intel Core2 Quad Q8200 2.33GHz. These results are an average of four simulation runs during fifteen simulated minutes. For convenience, each test is represented on the left with the required real time to compute an agent during one simulated second, and on the right with the corresponding expected simulated agents in real time. These graphs show that the no-steering model has a relatively constant computation performance whatever the number of simulated agents, because it does not take into account the agent's neighbourhood. This model displays the maximal computation performance we can expect from the *CA-LOD* model. The fuzzy-steering model is approximately 35% slower, with a small influence of the total number of simulated agents, mainly because this model only takes into account one near neighbour at a time. The geometric-steering model is 75% slower than without steering, with also a small influence of the overall number of simulated agents because part of the input is the topology which is unchanging.

Finally, an extrapolation of the possible repartition between the models to obtain real time performances is shown in *Figure 10*. The expected performances are very good, allowing thousands of agents to be simulated even in single thread with hundreds of agents using the best *CA-LOD*.



(a) Single thread simulation. (b) Multi threads simulation.

Fig. 10. Possible models repartition, in number of agents, allowing real-time performances

6 Conclusion and Perspectives

We have presented in this paper a novel framework for autonomous agents involving Collision Avoidance Levels Of Detail. Our objective is to propose a highly scalable model while maintaining the simulated crowd controllable to allow interactive applications. Three *CA-LOD*s of increasing complexity are proposed and described: the deactivation of the steering behaviour for the worst case; a fuzzy steering only taking into account one obstacle for the medium case; and a geometric steering with anticipation over all surrounding obstacles for the best case. An evaluation of the computation cost for each collision avoidance model is finally done, from which we extrapolate the possible repartitions between the models in order to obtain real-time performances on a reference computer. The obtained results allow to foresee real-time applications with huge interactive crowds.

Future work will first focus on the evaluation of different strategies allowing to choose each agent's *CA-LOD*. Indeed, we want to propose a model capable of choosing the optimal repartition between the *CA-LOD*s in order to maintain real-time performances while achieving the best visual realism. To do so, perceptual studies will be done in order to evaluate the areas where each collision avoidance model can be used to obtain the minimal realism loss. Then, an algorithm will be proposed to automatically choose the best repartition based on the dynamic analysis of the simulation performances. We envisage to use a genetic algorithm in order to automatically configure the best repartition based on measurable criteria, such as the number of non-avoided visible collisions with respect to the camera distance. Another perspective concerns the extension of the number of managed *CA-LOD*s, for example by limiting the inputs of the geometric steering to a maximal number of neighbouring obstacles. Finally, we plan to extend this approach to handle on the same decision basis all the possible level of details for an autonomous agent, such as other *LOD-AI*s, animation *LOD*s, or geometric *LOD*s.

Acknowledgement

We wish to thank *Science Foundation Ireland* (SFI) for funding the *Metropolis* project. We also want to thank Simon Dobbyn for his great work on the *Metropolis* system, as well as the entire *GV2* team for their participation to all parts of this project.

References

1. Hamill, J., O'Sullivan, C.: Virtual dublin - a framework for real-time urban simulation. In: Proc. of the Winter Conference on Computer Graphics, vol. 11, pp. 1–3 (2003)
2. Peters, C., Ennis, C.: Modeling groups of plausible virtual pedestrians. IEEE Computer Graphics and Applications 29(4), 54–63 (2009)
3. Wimmer, M., Bittner, J.: Hardware occlusion queries made useful. In: Pharr, M., Fernando, R. (eds.) GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation. Addison-Wesley, Reading (2005)
4. Luebke, D., Watson, B., Cohen, J.D., Reddy, M., Varshney, A.: Level of Detail for 3D Graphics. Elsevier Science Inc., New York (2002)
5. Dobbyn, S., Hamill, J., O'Conor, K., O'Sullivan, C.: Geopostors: a real-time geometry/impostor crowd rendering system. ACM Trans. Graph. 24(3), 933 (2005)
6. Yersin, B., Maim, J., Pettré, J., Thalmann, D.: Crowd Patches: Populating Large-Scale Virtual Environments for Real-Time Applications. In: I3D 2009 (2009)
7. Pettré, J., de Ciechomski, P.H., Maïm, J., Yersin, B., Laumond, J.P., Thalmann, D.: Real-time navigating crowds: scalable simulation and rendering. Computer Animation and Virtual Worlds 17(3-4), 445–455 (2006)
8. Paris, S., Donikian, S.: Activity-driven populace: a cognitive approach for crowd simulation. Computer Graphics and Applications (CGA) special issue Virtual Populace 29(4), 24–33 (2009)
9. Yu, Q., Terzopoulos, D.: A decision network framework for the behavioral animation of virtual humans. In: Metaxas, D., Popovic, J. (eds.) Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, pp. 119–128 (2007)
10. O'Sullivan, C., Cassell, J., Vilhjálmsson, H., Dingliana, J., Dobbyn, S., McNamee, B., Peters, C., Giang, T.: Levels of detail for crowds and groups. Computer Graphics Forum 21(4), 733–741 (2003)
11. Paris, S., Donikian, S., Bonvalet, N.: Environmental abstraction and path planning techniques for realistic crowd simulation. Computer Animation and Virtual Worlds 17, 325–335 (2006)
12. Reynolds, C.W.: Steering behaviors for autonomous characters. In: Game Developers Conference 1999 (1999)
13. Lamarche, F., Donikian, S.: Crowds of virtual humans: a new approach for real time navigation in complex and structured environments. Computer Graphics Forum 23, 509–518 (2004)
14. Paris, S., Pettré, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. In: Computer Graphics Forum, Eurographics 2007, vol. 26(3), pp. 665–674 (2007)
15. Helbing, D., Buzna, L., Johansson, A., Werner, T.: Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transportation Science 39(1), 1–24 (2005)
16. Paris, S., Mekni, M., Moulin, B.: Informed virtual geographic environments: an accurate topological approach. In: The International Conference on Advanced Geographic Information Systems & Web Services (GEOWS). IEEE Computer Society Press, Los Alamitos (2009)

17. Choset, H.M., Hutchinson, S., Lynch, K.M., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press, Cambridge (2005)
18. Gerdelan, A.P.: A solution for streamlining intelligent agent-based traffic into 3d simulations and games. Technical Report CSTN-072, IIMS, Massey University, North Shore 102-904, Auckland, New Zealand (January 2009)
19. Gerdelan, A.P.: Driving intelligence: A new architecture and novel hybrid algorithm for next-generation urban traffic simulation. Technical Report CSTN-079, Institute of Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand (February 2009)
20. Dougherty, M., Fox, K., Cullip, M., Boero, M.: Technological advances that impact on microsimulation modelling. Transport Reviews 20(2), 145–171 (2000)
21. Gerdelan, A.P., Reyes, N.H.: Towards a generalised hybrid path-planning and motion control system with auto-calibration for animated characters in 3d environments. In: Advances in Neuro-Information Processing. LNCS, vol. 5507, pp. 25–28. Springer, Heidelberg (2008)
22. Gerdelan, A.P.: Architecture design for self-training intelligent vehicle-driving agents: paradigms and tools. Technical Report CSTN-088, Institute of Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand (April 2009)

Exploiting Motion Capture to Enhance Avoidance Behaviour in Games

Ben J.H. van Basten[1], Sander E.M. Jansen[1,2], and Ioannis Karamouzas[1]

[1] Center for Advanced Gaming and Simulation, Utrecht University
[2] TNO Human Factors, Soesterberg,
The Netherlands
{basten,sanderj,ioannis}@cs.uu.nl

Abstract. Realistic simulation of interacting virtual characters is essential in computer games, training and simulation applications. The problem is very challenging since people are accustomed to real-world situations and thus, they can easily detect inconsistencies and artifacts in the simulations. Over the past twenty years several models have been proposed for simulating individuals, groups and crowds of characters. However, little effort has been made to actually understand how humans solve interactions and avoid inter-collisions in real-life. In this paper, we exploit motion capture data to gain more insights into human-human interactions. We propose four measures to describe the collision-avoidance behavior. Based on these measures, we extract simple rules that can be applied on top of existing agent and force based approaches, increasing the realism of the resulting simulations.

Keywords: motion analysis, motion capture, collision avoidance, human-human interaction.

1 Introduction

Virtual characters are commonly used in modern games and simulations. Typically, these characters have to navigate toward their desired locations in a human-like manner while avoiding collisions with other characters. As a result, visually compelling and natural looking avoidance behaviour has become a necessity for interactive virtual worlds and games.

Several models have been proposed over the past twenty years to simulate individuals, groups and crowds of characters. However, little effort has been made to actually understand how humans solve interactions and avoid colliding with each other in real-life. In this paper, we conduct an empirical study using motion capture to investigate interaction during human-human avoidance. We propose four measures to quantify this behaviour, namely *collaboration*, *clearance*, *anticipation* and *synchronization*. Furthermore, we analyse the effect of typical human characteristics (such as height and gender) on the avoidance behaviour. We choose height and gender since these can be easily distinguished in simulations.

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 29–40, 2009.

The rest of this paper is organised as follows. Section 2 gives an overview of related work. In Section 3, we discuss the experimental setup and present our devised measures. In Section 4, we present the results of our analysis. Overall conclusions are provided in Section 5. The discussion and suggestions for further research are presented in Section 6.

2 Related Work

Human interactions have been widely studied in the field of sociology and psychology. Goffman's [1] theory on pedestrian interactions has been influential in this field. According to his observations, two processes govern the avoidance behaviour of pedestrians, the *externalization* and the *scanning*. In the externalization, the pedestrian uses his body gestures to inform the others about his intentions. At the same time, he continuously scans the environment to gather the critical signals sent by the other pedestrians. Eventually, a coordination of actions is achieved between two interacting pedestrians and potential collisions are resolved. Wolff [2] argues that the collaboration is an essential part of the interaction and pedestrians expect to be cooperative upon interacting with each other.

Another important concept in interpersonal interactions is the notion of personal space. Sommer [3] describes the personal space as the portable territory around an individual that others should not violate. It regulates the psychophysical distance that the individual needs to maintain in order to feel comfortable. According to Hall [4], this distance can be used to determine the nature of the relationship between the individuals and decreases as the the level of intimacy increases. In an attempt to classify the distances that people prefer to keep, Hall identifies four main distances (intimate, personal, social and public). However, he argues that the proposed interpersonal distances can vary significantly depending on the gender and the age of the interacting individuals, as well their ethical and cultural background. In addition, the reported distances were based on static observational conditions (e.g. people waiting on a train platform), whereas the actual distances that people prefer to maintain while walking may be different.

Since Hall's work, a considerable amount of research focuses on the interpersonal spatial behaviour (see for example [5,6,7,8,9]). Dabbs and Stokes [7] reported that standstill pedestrians grant more space to approaching male pedestrians than to female pedestrians. They also indicated that the social context can influence the distance between individuals (for example attractive women are given more space than unattractive women). In contrast, Sobel and Lillith [6] reported that females were given more personal space than males. The contradiction could be due to the fact that Dabbs and Stokes studied the violation of stationary personal spaces, whereas Sobel and Lillith focused on nonstationary personal spaces (that is both interacting pedestrians were moving). Caplan and Goldman [8] suggested that besides gender, the physical dominance can also affect the size of the personal space. In their observations, pedestrians invaded the space of short people more frequently compared to the space of tall people.

Experimental research on pedestrian interactions has also become increasingly popular among the civil and traffic engineering community. One of the main

objectives in this community is to evaluate the quality and the design of walking infrastructures and pedestrian facilities. Therefore, several empirical studies have been conducted over the past years to gain more insights into both the microscopic and macroscopic characteristics of pedestrian flows (e.g. [10,11,12]). Based on these studies a number of pedestrian simulation models have been proposed capable of reproducing the empirically observed pedestrian behaviours, such as the formation of lanes when people cross in opposite directions. The most popular in this field is Helbing's social force model [13,14]. Helbing uses physical forces to describe the social interactions between the pedestrians. Although his model has been successfully used in many simulation applications, the local behaviour of the pedestrians is far from natural. Due to lack of anticipation and prediction, the virtual pedestrians interact when they get sufficiently close, which results in unrealistic motions.

Pedestrian and crowd simulation has also received a lot of attention in the animation, graphics and virtual environment community and numerous models have been proposed for simulating individuals, groups and crowds of interacting characters. Treuille *et al* [15] have recently presented a new approach for realistic simulation of large groups of characters moving toward a common goal. Their approach unifies global navigation and local collision avoidance into a single framework and reproduces specific crowd phenomena. However it is not suited for individual characters with distinct characteristics and goals. Another common approach is to decouple global planning from the local avoidance (see for example [16,17,18,19,20]). Many interesting collision avoidance approaches have been proposed in the past based on variants of agent-based methods [21], including rule-based techniques [22,23] and reactive behavioural models [17], to name just a few. More recently, van den Berg *et al* [24] have introduced the concept of *Reciprocal Velocity Obstacle* for local navigation. The idea here is that each character adapts its velocity in order to avoid collisions with other characters as well as with the environment. Although the method generates collision-free motions, the resulting simulations haven't been validated with actual (empirical) pedestrian flow data. Close to the aforementioned work, Paris *et al* [25] devised an elegant collision avoidance method that predicts potential collisions within time and resolves them by adapting the speed and/or the orientation of the virtual characters. The parameters of their model were calibrated using experimental motion capture data. Their approach was empirically compared with real-world data. However, in their simulations, the resulting character flow does not look realistic, since upon interacting with each other the characters abruptly stop and change their orientations. More recently, Pettré *et al* [26], proposed an egocentric model for solving interactions between virtual pedestrians based on experimental studies.

Alternatively, data-driven techniques have also been explored for simulating interacting virtual characters. These approaches use example behaviors from video or motion capture data to drive the simulation of virtual characters. In [27], a database of human trajectories is learnt from video recordings of real pedestrian crowds. During the simulation, each virtual character searches the

database and selects the trajectory that is closely related to its situation. A similar approach has been proposed by Lee *et al* [28] aiming at realistic group behaviors, whereas more recently Kyriakou and Chrysanthou have presented a novel approach based on texture synthesis [29]. The main advantage of all these example-based methods is that they can realistically simulate crowds of virtual humans. However, their applicability is limited by the size of the example databases. In addition, these approaches are too computationally expensive for real-time interactive applications like computer games.

In this paper we exploit motion capture recordings in order to gain a better understanding into how humans interact with each other in real life. Our work is related to several of the aforementioned empirical studies from the area of sociology, psychology, civil and traffic engineering. The main objective of our research, though, is to simulate virtual humans that behave in a natural way, that is solve interactions and avoid inter-collisions like real humans do. Our approach bears also some resemblance with example-based approaches, as well as with the work of Pettré *et al*. However, since we strive for a general solution, we have devised four measures that allow us to describe the interaction behaviour of the participants in a more abstract level. These measures are then used to derive behavioural rules that can be easily incorporated on top of a wide range of simulation models.

3 Experimental Setup

This section elaborates on the experimental setup and our proposed measures that can be used to describe interaction behaviour.

3.1 Participants

9 female and 13 male participants (age between 19 and 32, $M = 23.4, SD = 3.1$) gave informed consent to participate in this experiment. All were free of any known neurological or orthopaedic disorders, or any impediments to normal locomotion, as verified by self-report. From the 22 participants, we selected 18 pairs based on gender (male-male, female-male, female-female). We also annotated for each of the participants whether he/she is short or tall. Cut off length lay at 170 cm for males and 160 cm for females. Thus, each pair consisted of a short-short, short-tall or tall-tall combination.

3.2 Procedures

For each pair, a trial consists of both people walking between two points marked on the floor (s_1 and s_2). They start at the same time and walk in opposite directions, thus having to avoid each other somewhere along the path. The distance between s_1 and s_2 was 415 cm. In order to shift attention from the task, participants were provided with a cognitive workload task. During every trial, the participants needed to memorize a number, printed on a note at the start point.

Fig. 1. The two participants walk in opposite direction, starting from s_1 and s_2

This number has to be written down at the end point. A schematic representation of the setup is depicted in Figure 1. Each pair performed 3 trials, resulting in a total of 54 recordings.

3.3 Tracking

All trials were recorded using a Vicon motion capture system [30] consisting of 8 Vicon MX40 near-infrared cameras (100 fps). Each participant wore a suit with 34 reflective markers. Figure 2 shows two screenshots of a recorded trial in the motion capture software.

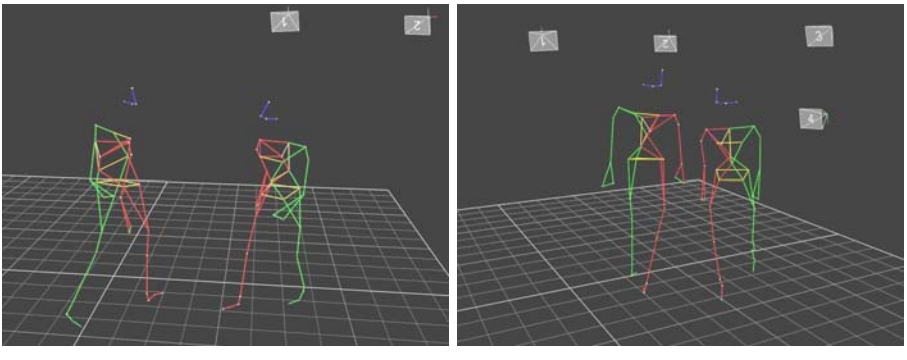


Fig. 2. Two screenshots of a male-male trial

3.4 Variables

When looking at games or simulations, characters can be easily distinguished by gender and height. Therefore, we study the influence that these characteristics have on the collision avoidance behaviour. This is investigated by comparing 4 different measures: *collaboration*, *clearance*, *anticipation* and *synchronisation*.

Collaboration. This measure indicates to what extent both participants contribute to the lateral distance at the moment of passing. This moment is defined as the time where the line l_2 between the *trunks* [31] of the two participants is orthogonal to the straight line l_1 between s_1 and s_2. We approximate the trunk by interpolating markers on the chest and the back of the participant. At the moment of passing at least one of the participants keeps a lateral distance to the ideal line l_1. In the example depicted in Figure 3, participant 1 has a distance of d_1 and participant 2 has a distance of d_2 at the moment of passing.

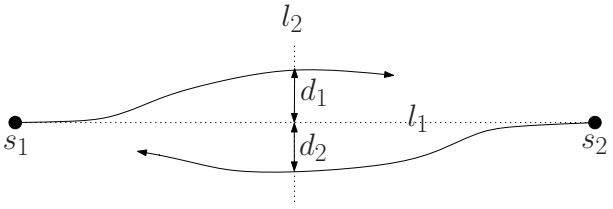


Fig. 3. During avoidance, participants need to deviate from l_1

We consider collaboration to be high when these lateral distances are equal and low when they differ a lot. In order to normalize this measure we use the following formula:

$$Collaboration = 1 - \frac{|d_1 - d_2|}{\max(d_1, d_2)}$$

Clearance. The *clearance* is defined as the minimal distance between the participants during the entire trial. This is approximated by the minimum distance between both set of markers. We also annotate each trial with the markers that determine this minimal distance. Let $\mathbf{M_1}$ and $\mathbf{M_2}$ be the marker sets of the two participants, then the clearance is defined as:

$$Clearance = \min\{dist(m_1, m_2) : m_1 \in \mathbf{M_1}, m_2 \in \mathbf{M_2}\}$$

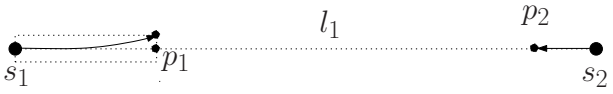
Note that clearance is based on Euclidean 3D distance and is not normalized.

Anticipation. At some point during the recording, at least one of the participants has to deviate from the line l_1 in order to avoid a collision. *Anticipation* deals with the position at the moment of deviation. Note that collaboration is determined by lateral clearance, whereas anticipation deals with frontal clearance.

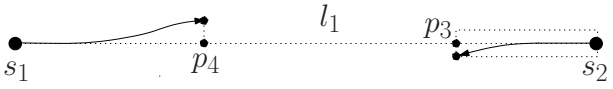
For simplicity, let us assume that participant 1 will deviate first. This moment is determined when the participant deviates more than 10 cm from the line l_1 (See Figure 4(a)). The position of the participant is represented by interpolating the hip markers and projecting it down on the ground plane. The position of participant 1 at t_1 is projected onto l_1 and is denoted as p_1. We denote p_2 as the position of participant 2 (that has not yet deviated) at t_1, again projected on l_1.

Participant 2 might also deviate from l_1. This is depicted in Figure 4(b). We denote the time and position of deviation of participant 2 (projected on l_1) as t_2 and p_3 respectively. The position of participant 1 projected on l_1 at time t_2 is denoted as p_4.

We define the *anticipation* as the normalized sum of the distances between the two participants at the two moments of deviation. In our setup, we determine the anticipation as follows:



(a) At t_1, participant 1 will deviate in order to avoid collision.



(b) At t_2, participant 2 might also deviate from l_1.

Fig. 4. Anticipation deals with the position of both people at the respective moments of deviation

$$Anticipation = \frac{dist(p_1, p_2) + dist(p_3, p_4)}{2 \cdot dist(s_1, s_2)}$$

A high anticipation of 1.0 means that both participants deviate immediately at s_1 and s_2. A low anticipation of 0.0 means that both participants deviate at the very last moment. In case participant 2 does not deviate, but follows l_1, then $dist(p_3, p_4)$ is 0.0.

Synchronisation. When looking at the respective moments of deviation, we consider the avoidance behaviour of the participants to be synchronized when they both start deviating at the same time. We determine the moments of deviation t_1 and t_2 as described above. *Synchronisation* is then defined as follows:

$$Synchronisation = 1 - \frac{|t_1 - t_2|}{\max(t_1, t_2)}$$

High synchronisation indicates that both people start deviating nearly simultaneously, while low synchronisation means that one moves much sooner than the other. Note that synchronisation can still be high even though deviation starts late.

3.5 Statistical Analysis

Of the 54 recordings made, 4 were excluded from analysis due to incomplete data sets. Because of unequal sample sizes, Kruskall-Wallis ANOVA's [32] were performed on each of the four proposed measures. This test is a non-parametric alternative to the classic analysis of variance. None of the ANOVA's showed significant effects but since this was assumed to be caused by the small size of the data set it was decided to perform pairwise comparisons. These were done using t-tests for independent samples. We use Levene's test to check the assumption of equal variance. Whenever this assumption was violated, Welch t-test was performed instead [33].

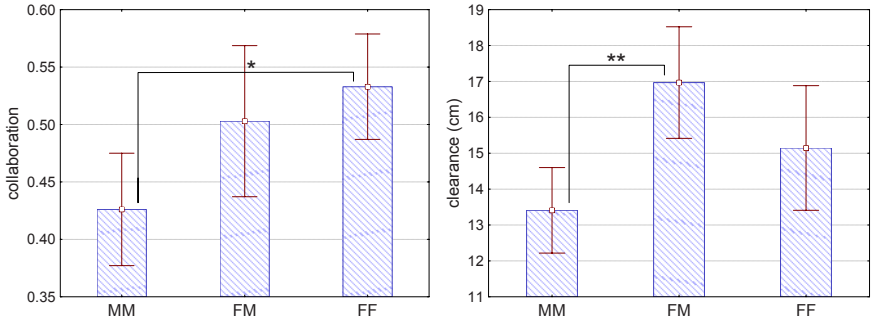


Fig. 5. Left: Collaboration (on a scale of 0 to 1) as a function of gender.1.0 indicates perfect collaboration, while 0.0 means that one of the two did all the avoiding. Right: Clearance as a function of gender interaction. Vertical bars denote standard error of the mean. Significant differences are illustrated by * ($p < 0.1$) and ** ($p < 0.05$).

4 Results

The results of the statistical analyses are divided in separate sections based on gender and height. For anticipation and synchronization no significant results were found. Therefore we do not report these corresponding statistics.

4.1 Gender

Pairwise comparison shows that two males collaborate significantly less than two females when avoiding each other, $p < 0.06$. Furthermore, the minimum clearance between two males avoiding each other is smaller than that between a male and female, $p < 0.04$. See Figure 5 for complete results.

4.2 Height

Two tall people collaborate significantly less than two short people ($p < 0.03$) or a short and tall person ($p < 0.03$). Furthermore, there is a difference in clearance between two short people and a mixed pair. The former pair has a larger clearance ($p < 0.08$). See Figure 6 for complete results.

4.3 Additional Observations

The minimum distance for each pair was always between points on the arms of the participants. Only once did we observe an exception to this rule, which occurred when a collision was avoided at the very last moment by sidestepping of both people. In this case the upper legs were the closest points. Furthermore, the recordings show no preference for passing on a specific side. In 54% of the trials, passage was on the right side. During the remaining trials, passage was on the left side.

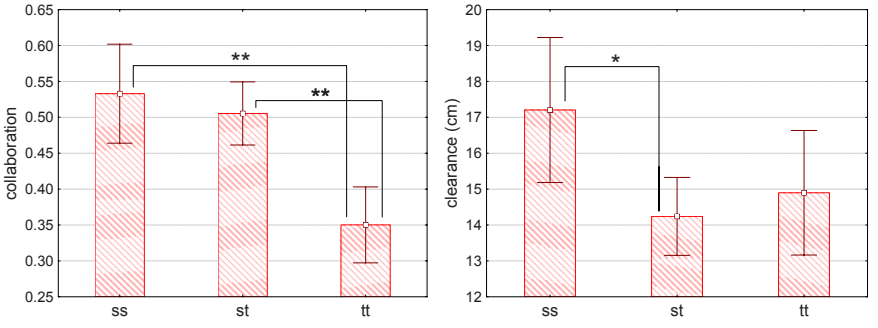


Fig. 6. Left: Collaboration (on a scale of 0 to 1) as a function of height. A value of 1.0 indicates perfect collaboration, while a value of 0.0 means that one of the two did all the avoiding. Right: Clearance as a function of height. Vertical bars denote standard error of the mean. Significant differences are illustrated by * ($p < 0.1$) and ** ($p < 0.05$).

5 Conclusion

The results of this study indicate that certain pairs of people have different ways of avoiding each other depending on gender and height. We devised several measures that can describe this behaviour. The measure of *collaboration* indicates to what extent lateral distance is shared between two people. We found that two males collaborate less than two females and that a pair of tall people (gender independent) also collaborate less than either two short people or a mixed pair.

The second measure is *clearance*. This indicates how close two people allow each other to pass. From our results, we conclude that the minimum distance between two males is smaller than that between a male and female. Furthermore, there is a larger clearance between two short people than between a short and a tall person.

The measure of *anticipation* takes into account the spatial relation between the deviation actions. The measure of *synchronization* indicates the temporal relationship between the two deviation moments. We found no effects of gender and height for both the anticipation and synchronization measures. However, we do think that these are useful measures to describe avoidance behaviour.

In conclusion, we have shown that people can differ in the way they avoid each other depending on their gender and height. In our analysis, we have described this behavior in terms of several interaction measures. We are confident, though, that such knowledge can also be incorporated into existing models for local collision avoidance (e.g. [24,25]), leading to more realistic character interactions. In particular, we believe that characteristics such as gender and height are clearly distinguishable in a simulation. Therefore, our goal is to derive from the results of our statistical analysis simple rules that can be easily integrated on top of existing reactive navigation methods. Regarding the collaboration results, for example, the following (relaxed) rule can be devised:

```
for each pair of agents
    if both agents are male
        only one of the two performs an avoidance maneuver
    else
        both agents perform an avoidance maneuver
```

A similar rule can also be obtained for the clearance of the virtual characters. As an example we incorporated these rules into an existing simulation framework (see Figure 7). Although our work is still in progress, preliminary results are quite promising.

Fig. 7. Character interactions in a virtual environment. The character with the dark blue shirt follows a straight path forcing the other character to maneuver in order to avoid a collision. Note also the small clearance at their moment of passing. In contrast, the two female characters take an equal share in the effort to avoid colliding and maintain a relatively large clearance.

6 Discussion and Future Work

The main focus of this paper is on deriving interaction measures. Because of the preliminary nature of this study, the data sets were relatively small (N=22). For future work we plan to do additional recordings so we can perform a full-factorial analysis of the data.

Furthermore, in this paper, we focus on one-on-one interactions. However, there are situations in which more than two people are involved in avoidance behaviour. Especially in crowds, multiple people might avoid each other simultaneously. Besides individuals, we would like to take into account couples or small groups. We are also investigating how our approach can scale to large and complex environments.

In addition, it would be very useful to evaluate the naturalness of the generated motions by conducting a user study [34,35]. We can then determine whether our approach improves the perceived realism of existing local avoidance models.

Acknowledgments

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

References

1. Goffman, E.: Relations in public: microstudies of the public order. Basic books, New York (1971)
2. Wolff, M.: Notes on the behaviour of pedestrians. People in places: The sociology of the familiar, 35–48 (1973)
3. Sommer, R.: Personal Space. The Behavioral Basis of Design. Prentice-Hall, Englewood Cliffs (1969)
4. Hall, E.T.: The Hidden Dimension, 1st edn. Doubleday, Garden City (1966)
5. Hartnett, J., Bailey, K., Hartley, C.: Body height, position, and sex as determinants of personal space. Journal of psychology 87, 129–136 (1974)
6. Sobel, R., Lillith, N.: Determinant of nonstationary personal space invasion. Journal of social psychology 97, 39–45 (1975)
7. Dabbs Jr., J., Stokes III, N.: Beauty is power: the use of space on the sidewalk. Sociometry 38, 551–557 (1975)
8. Caplan, M., Goldman, M.: Personal space violations as a function of height. Journal of Social Psychology 114, 167–171 (1981)
9. Gérin-Lajoie, M., Richards, C., McFadyen, B.: The circumvention of obstacles during walking in different environmental contexts: A comparison between older and younger adults. Gait and Posture 24(3), 364–369 (2006)
10. Helbing, D., Molnár, P., Farkas, I., Bolay, K.: Self-organizing pedestrian movement. Environment and Planning B 28, 361–384 (2001)
11. Hoogendoorn, S., Daamen, W.: Self-organization in pedestrian flow. In: Hoogendoorn, S.P., Luding, S., Bovy, P.H.L., Schreckenberg, M., Wolf, D.E. (eds.) Traffic and Granular Flow 2003, pp. 373–382 (2005)
12. Teknomo, K.: Application of microscopic pedestrian simulation model. Transportation Research PartF: Traffic Psychology and Behaviour 9(1), 15–27 (2006)
13. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Physical Review E 51, 4282–4286 (1995)
14. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature 407, 487–490 (2000)
15. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Transactions on Graphics 25(3), 1160–1168 (2006)
16. Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. Computer Graphics Forum 23, 509–518 (2004)
17. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: SCA 2005: ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 19–28 (2005)
18. Geraerts, R., Overmars, M.: The corridor map method: A general framework for real-time high-quality path planning. Computer Animation and Virtual Worlds 18, 107–119 (2007)
19. Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., Manocha, D.: Real-time navigation of independent agents using adaptive roadmaps. In: ACM symposium on Virtual reality software and technology, pp. 99–106 (2007)

20. Karamouzas, I., Geraerts, R., Overmars, M.: Indicative routes for path planning and crowd simulation. In: FDG 2009: Proceedings of the 4th International Conference on Foundations of Digital Games, pp. 113–120 (2009)
21. Reynolds, C.W.: Steering behaviors for autonomous characters. In: The proceedings of Game Developers Conference, pp. 763–782 (1999)
22. Loscos, C., Marchal, D., Meyer, A.: Intuitive crowd behaviour in dense urban environments using local laws. Theory and Practice of Computer Graphics (2003)
23. Musse, S.R., Thalmann, D.: A model of human crowd behavior: Group interrelationship and collision detection analysis. In: Workshop of Computer Animation and Simulation of Eurographics, pp. 39–51 (1997)
24. van den Berg, J.P., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: ICRA, pp. 1928–1935. IEEE, Los Alamitos (2008)
25. Paris, S., Pettré, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. Computer Graphics Forum 26(3), 665–674 (2007)
26. Pettré, J., Ondrej, J., Olivier, A.H., Crétual, A., Donikian, S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In: ACM SIGGRAPH/Eurographics symposium on Computer animation (2009)
27. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphics Forum 26, 655–664 (2007)
28. Lee, K., Choi, M., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: SCA 2007: ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 109–118 (2007)
29. Kyriakou, M., Chrysanthou, Y.: Texture synthesis based simulation of secondary agents. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, pp. 1–10. Springer, Heidelberg (2008)
30. Vicon: Motion capture systems from vicon, http://www.vicon.com
31. Arechavaleta, G., Laumond, J.P., Hicheur, H., Berthoz, A.: The nonholonomic nature of human locomotion: a modeling study. In: International Conference on Biomedical Robotics and Biomechatronics, pp. 158–163 (2006)
32. Kruskal, W., Wallis, W.: Use of ranks in one-criterion variance analysis. Journal of the American Statistical Association, 583–621 (1952)
33. Field, A.: Discovering statistics using SPSS. Sage Publications Ltd. (2009)
34. van Basten, B.J.H., Egges, A.: Evaluating distance metrics for animation blending. In: FDG 2009: Proceedings of the 4th International Conference on Foundations of Digital Games, pp. 199–206 (2009)
35. Jansen, S., van Welbergen, H.: Methodologies for the user evaluation of the motion of virtual humans. In: Proceedings of the 9th International Conference on Intelligent Virtual Agents, vol. 5773, pp. 125–131. Springer, Heidelberg (2009)

A Predictive Collision Avoidance Model for Pedestrian Simulation

Ioannis Karamouzas, Peter Heil, Pascal van Beek, and Mark H. Overmars

Center for Advanced Gaming and Simulation, Utrecht University,
The Netherlands
{ioannis,pccheil,phbeek,markov}@cs.uu.nl

Abstract. We present a new local method for collision avoidance that is based on collision prediction. In our model, each pedestrian predicts possible future collisions with other pedestrians and then makes an efficient move to avoid them. Experiments show that the new approach leads to considerably shorter and less curved paths, ensuring smooth avoidance behaviour and visually compelling simulations. The method reproduces emergent behaviour like lane formation that have been observed in real crowds. The technique is easy to implement and is fast, allowing the simulation in real time of crowds of thousands of pedestrians.

Keywords: collision avoidance, interaction, pedestrian simulation.

1 Introduction

Virtual city environments are nowadays commonly used in computer games, simulations, training applications, and online communities. Although in many such applications the cities look very realistic, to become more lively and appealing, they need to be populated with a large number of simulated pedestrians. On one hand, these virtual pedestrians have to be visually attractive to the viewer. On the other hand, they should be able to move through the virtual world in a natural looking way.

Over the past years numerous models have been proposed to simulate individuals, groups and crowds of characters. Treuille *et al.* [1] have recently presented a dynamic potential-field approach that unifies global navigation and local collision avoidance into a single framework. However, their approach is limited to homogeneous groups of characters moving toward a common goal and cannot simulate individuals with distinct characteristics and goals. An alternative approach is to decouple global planning from local collision avoidance [2,3,4,5]. Typically, a graph-based method is used to direct the global motion of the character, whereas its local behaviour is governed by some sort of agent-based [6] or force-field approach [7].

At the local level, the way that virtual characters interact and avoid collisions with each other is essential for the realism of the generated motions. Helbing simulated the behaviour of pedestrians using (social) forces related to physics [7]. Since his original work, a number of models have been proposed to simulate

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 41–52, 2009.

crowds of pedestrians under normal and emergency situations (e.g. [8,9]). Nevertheless, in all these approaches, due to lack of anticipation and prediction, the characters interact when they get sufficiently close. Consequently, the resulting motions tend to look unnatural and contain undesirable oscillations. The problem becomes more obvious in large and cluttered environments, with pedestrians constantly changing their orientations, pushing each other and moving back and forth.

An alternative way for solving interactions between virtual humans is based on collision prediction. In these approaches, the agents' trajectories are linearly extrapolated and used to determine collisions in the near future. Based on this idea, Reynolds has introduced the *unaligned collision avoidance* behaviour [6], whereas Feurtey devised an elegant collision detection algorithm that predicts potential collisions within time and resolves them by adapting the speed and/or the trajectories of the agents. Inspired by Feurtey's work, Paris *et al.* [10] presented an anticipative model to steer virtual pedestrians without colliding with each other. Similarly, Shao and Terzopoulos [5] proposed a number of reactive behavioral routines to determine the avoidance maneuvers of the agents. More recently, van den Berg *et al.* [11] introduced the concept of *Reciprocal Velocity Obstacle*. Finally, Pettré *et al.* [12] proposed an egocentric model for local collision avoidance that is based on experimental studies and can be automatically calibrated from the experimental data.

Recently, example-based techniques have also been explored for simulating interacting virtual characters [13,14]. However, these approaches are too computationally expensive and cannot be used for real-time interactive applications.

Contributions. In this paper, we present a new local method for realistic character avoidance. It is based on the hypothesis that an individual adapts its route as early as possible, trying to minimise the amount of interactions with others and the energy required to solve these interactions. Building upon this hypothesis, in our model, a pedestrian computes with which other pedestrians is in collision course within a certain anticipation time. It calculates how such collisions will take place and then make an efficient move to avoid them. Consequently, the characters do not repel each other, but rather anticipate future situations avoiding all collisions long in advance and with minimal effort.

We show that the new approach leads to energy-efficient motions and considerably less curved paths for the pedestrians, resulting in a smooth, natural flow. The generated motions are oscillation-free. The method reproduces emergent behaviours, like lane formation, that have been observed in real crowds. Since it is a predictive rather than a reactive approach, it is somewhat similar in nature to the approaches of [5,10,11] mentioned above. However, it is based on a force-field approach and, hence, it is much easier in its formulation and implementation and considerably faster, allowing real-time simulations of crowds of thousands of pedestrians. Our approach bears also some resemblance with the avoidance behaviour proposed in [6]. In our model, though, the direction and the magnitude of the collision avoidance force is computed in a different way, ensuring smooth avoidance behaviour and visually pleasing simulations.

2 Pedestrian Interactions

In this section we present some key concepts regarding how pedestrians interact with each other in real life. In the next sections, we use these concepts to formulate a model of realistic collision avoidance.

Scanning and Externalization. Human interactions have been widely studied in the field of sociology. Of particular importance are the studies of Goffman [15] related to how people behave at a microscopic level. According to his observations, two processes govern the avoidance behaviour of pedestrians, the *externalization* and the *scanning*. In externalization, a pedestrian uses its body language to inform the others about its intentions, that is, to indicate its planned course. At the same time, it continuously scans the environment to gather the signals given out by other pedestrians. Eventually, a voluntary coordination takes place and a collision is resolved.

Personal Space. Another important concept in social interactions is the *personal space* that surrounds an individual. More formally, the personal space can be defined as the portable territory around an individual which others should not invade. It regulates the safe distance that an individual needs to maintain from others in order to feel comfortable. The size and the shape of the personal shape are constantly changing depending on the crowd density, as well as the travel speed of the pedestrian. According to Goffman [15], the personal space can be represented as an oval, narrow to the sides of the individual and long in front of him/her.

Principle of Least Effort. The *principle of least effort* originates from the field of psychology and states that given different possibilities of actions, people select the one that requires the least effort [16]. Consequently, we can assume that, upon interacting with each other, pedestrians try to avoid unnecessary detours and follow energy-efficient trajectories, that is paths that reduce the amount of movement and turning effort.

3 Pedestrian Simulation Model

In our problem setting, we are given a virtual environment in which n pedestrians $P_1, ...P_n$ have to navigate toward their specified goal positions $\mathbf{g}_i$ without colliding with the environment and with each other. For simplicity we assume that each pedestrian moves on a plane or a terrain and is modeled as a disc with radius r_i. At a fixed time t, the pedestrian P_i is at position $\mathbf{x}_i(t)$, defined by the center of the disc, has an orientation $\theta_i(t)$ and moves with velocity $\mathbf{v}_i(t)$. This velocity is limited by a maximum speed $u_i^{\max}$, that is $\|\mathbf{v}_i(t)\| \leq u_i^{\max}$. Hereafter, for notational convenience, we will not explicitly indicate the time dependence.

We propose a (social) force model to obtain realistic collision avoidance. The behaviour of each pedestrian P_i derives from the following assumptions:

1. At each step of the simulation the pedestrian P_i is trying to reach its desired goal position $\mathbf{g}_i$. Thus, a goal force $\mathbf{F}_g$ is applied that attracts the pedestrian to its goal.
2. The pedestrian P_i prefers to move toward its destination with a certain speed u_i^{pref}. It tends to reach this speed gradually within a certain time τ. Along with the first assumption, the goal force $\mathbf{F}_g$ can be defined as:

$$\mathbf{F}_g = \frac{1}{\tau}(u_i^{\text{pref}}\mathbf{n}_{gi} - \mathbf{v}), \tag{1}$$

   where $\mathbf{n}_{gi} = \frac{\mathbf{g}_i - \mathbf{x}_i}{\|\mathbf{g}_i - \mathbf{x}_i\|}$. Note that $\mathbf{F}_g$ is similar to the force described in Equation (2) of Helbing's social force model [7].
3. As P_i advances toward its goal, it also has to avoid collisions with the environment (e.g. building walls). Let $\mathcal{W}$ denote the set of walls that are present in the virtual environment. Then, a repulsive force $\mathbf{F}_w$ is exerted from each wall $w \in \mathcal{W}$ toward the pedestrian. This force can be defined as:

$$\mathbf{F}_w = \begin{cases} \mathbf{n}_w \dfrac{d_s + r_i - d_{iw}}{(d_{iw} - r)^{\kappa}}, & \text{if } d_w - r_i < d_s \\ \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

   where the constant κ indicates the steepness of the repulsive potential, $\mathbf{n}_w$ is the normal vector of the wall, d_{iw} defines the shortest distance between P_i and the wall and d_s denotes the safe distance that the pedestrian prefers to keep from the buildings.
4. The pedestrian P_i keeps a certain psychophysical distance ρ_i from other pedestrians in order to feel comfortable. This distance defines the so-called personal space of the pedestrian. For efficiency reasons, we model this space as a disc $B(\mathbf{x}_i, \rho_i)$ centered at the current position of the pedestrian and having radius $\rho_i > r_i$. In all of our simulations, this led to realistic behaviour and hence, the use of an elliptical personal space was not necessary.
5. The pedestrian P_i perceives the environment to detect imminent and future collisions with other pedestrians. A collision at some time $t_c \geq 0$ occurs when another pedestrian P_j invades into the personal space of P_i, that is:

$$\exists\, t_c \geq 0 \mid d_{ij} \leq \rho_i + r_j, \tag{3}$$

   where $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ denotes the distance between the pedestrians' centers.
6. Given a certain anticipation time t_α, the pedestrian P_i resolves potential collisions within this time in advance by adjusting its trajectory. As a result, it safely navigates toward its goal, evading other pedestrians with minimal effort. To simulate this behaviour, an evasive force $\mathbf{F}_e$ is applied on P_i. Note that the anticipation time t_α can vary between pedestrians.

4 Avoiding Collisions

The major novelty of our approach is the evasive force $\mathbf{F}_e$ that a pedestrian selects in order to avoid collisions and near collisions with other pedestrians. Given a pedestrian P_i, our collision avoidance algorithm consists of four steps.

4.1 Collision Prediction

In the first step of our algorithm we compute the set $CP_i^{t_\alpha}$ of pedestrians that are on collision course with the pedestrian P_i, given a certain anticipation time t_α. To achieve this, we first infer the desired velocity $\mathbf{v}_i^{\text{des}}$ of P_i. We compute $\mathbf{v}_i^{\text{des}}$ as the sum of its actual velocity and the velocity derived by virtually applying only the goal force and the wall repulsive forces:

$$\mathbf{v}_i^{\text{des}} = \mathbf{v}_i + \left(\sum \mathbf{F}_w + \mathbf{F}_g\right) \Delta t, \tag{4}$$

where Δt is the size of the time step of the simulation.

Then, we estimate the future position of P_i based on its current position $\mathbf{x}_i$ and desired velocity $\mathbf{v}_i^{\text{des}}$ as follows:

$$\mathbf{x}_i' = \mathbf{x}_i + t\mathbf{v}_i^{\text{des}} \tag{5}$$

Similarly, we predict the future motions of all the other pedestrians that P_i can see by linearly extrapolating their current trajectories. The viewing area of P_i is defined by its desired direction of motion and its field of view. Note that the pedestrian P_i can only estimate the actual velocities of the other pedestrians and not their desired ones, since it does not know their corresponding goal positions. Thus, from the perspective of P_i, the future position of a pedestrian P_j is given by:

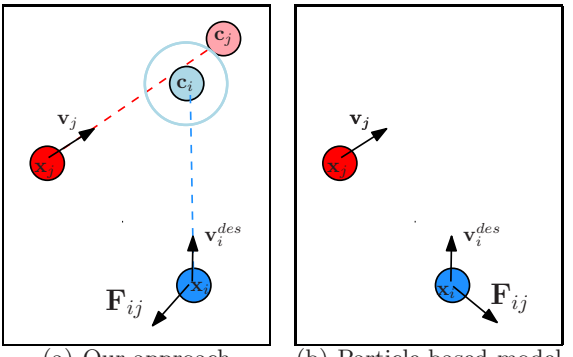
$$\mathbf{x}_j' = \mathbf{x}_j + t\mathbf{v}_j \tag{6}$$

We can now determine whether the pedestrian P_i collides with another pedestrian P_j. According to (3) a collision occurs when the pedestrian P_j lies inside or intersects the personal space $B(\mathbf{x}_i, \rho_i)$ of P_i. By performing a Minkowski sum between the disc $B(\mathbf{x}_i, \rho_i)$ and the disc $B(\mathbf{x}_j, r_j)$ of P_j, the problem is reduced into a ray-disc intersection test that results in the following equation:

$$\|\mathbf{x}_j - (\mathbf{x}_i + \mathbf{v}t)\| = \rho_i + r_j, \tag{7}$$

where $\mathbf{v} = \mathbf{v}_i^{\text{des}} - \mathbf{v}_j$. Solving the above equation for t, we can estimate the possible collision times tc_{ij} between the personal space of the pedestrian P_i and the pedestrian P_j. If the equation has no solution or a single solution, then no collision takes place. If there are two solutions ($t1$ and $t2$), three cases can be distinguished:

- $t1, t2 \leq 0$: this is a past collision and can be ignored.
- $t1 < 0 < t2 \vee t2 < 0 < t1$: this is an imminent collision, i.e. $tc_{ij} = 0$, and hence, the pedestrian P_j is inserted into the set $CP_i^{t_\alpha}$.
- $t1, t2 \geq 0$: a collision will occur at time $tc_{ij} = min(t1, t2)$. If $tc_{ij} \leq t_\alpha$, the pedestrian P_j is inserted into the set $CP_i^{t_\alpha}$.



(a) Our approach (b) Particle-based model

Fig. 1. Example of the force used to avoid a future collision. Note that the direction of the force depends on the relative positions at the moment of the impact, rather than on the current positions of the pedestrians. The light blue disc represents the personal space of the pedestrian P_i.

4.2 Selecting Pedestrians

Having computed the set $CP_i^{t_\alpha}$, we sort it in order of increasing collision time and keep the first N pedestrians. Preliminary experiments (Section 6) have indicated that this number can be kept small (between 2 to 5 pedestrians), ensuring smooth avoidance behaviour. This not only reduces the running time of our algorithm, but also reflects natural human behaviour. In real-life, an individual tries to avoid a limited number of other pedestrians, usually those that are on collision course with him in the coming short time. Similarly, the virtual pedestrian P_i tries to evade the N pedestrians with which it will collide first.

4.3 Avoidance Maneuvers

We now show how the pedestrian P_i can avoid a potential collision with another pedestrian P_j that belongs to the set $CP_i^{t_\alpha}$. Let tc_{ij} be the time of collision between the two pedestrians. Let also $\mathbf{c}_i = \mathbf{x}_i + tc_{ij}\mathbf{v}_i^{\text{des}}$ and $\mathbf{c}_j = \mathbf{x}_j + tc_{ij}\mathbf{v}_j$ denote the locations of pedestrians P_i and P_j at time tc_{ij}; $\mathbf{c}_i$ and $\mathbf{c}_j$ derive from (5) and (6) respectively.

Based on these future locations, we select an evasive force $\mathbf{F}_{ij}$ for the pedestrian P_i, so that it can smoothly avoid the pedestrian P_j. The direction of the force is given by the unit vector $\mathbf{n}_{c_ic_j} = \frac{\mathbf{c}_i - \mathbf{c}_j}{\|\mathbf{c}_i - \mathbf{c}_j\|}$ pointing from $\mathbf{c}_j$ to $\mathbf{c}_i$. As an example, consider Fig. 1(a). The blue pedestrian will hit the red pedestrian at the back. Therefore, it makes a slight move toward the latter and eventually will pass behind it. In contrast, in a typical particle-based system the red pedestrian would have forced the blue one to move in the opposite direction, leading to a new collision in the near future (Fig. 1(b)).

The magnitude of the evasive force $\mathbf{F}_{ij}$ is approximated by a piecewise function f(D), where $D = \|\mathbf{c}_i - \mathbf{x}_i\| + (\|\mathbf{c}_i - \mathbf{c}_j\| - r_i - r_j)$ is the distance between the current

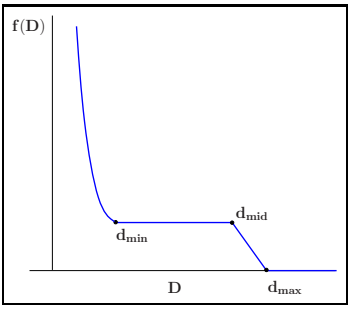


Fig. 2. The magnitude of the avoidance force as a function of the distance D

position $\mathbf{x}_i$ of pedestrian P_i and its future position $\mathbf{c}_i$ plus the distance between the two pedestrians at their time of collision. As can be inferred from Fig. 2, the function is defined for four intervals. The threshold $d_{\max}$ determines the start of the avoidance maneuver, whereas $d_{\min}$ defines the beginning of an impenetrable barrier between the pedestrians. The threshold d_{mid} regulates the start of the constant part of the function. This part is used to eliminate jerky behaviour. The three thresholds $d_{\min}$, d_{mid} and $d_{\max}$ can vary among the pedestrians to simulate different avoidance behaviours.

4.4 Computing the Evasive Force

In the last step of our algorithm, we compute the total evasive force $\mathbf{F}_e$ that is applied on the pedestrian P_i given its set $CP_i^{t_\alpha}$. Two approaches can be distinguished. The first is to take into account each one of the N pedestrians independently and compute a corresponding force $\mathbf{F}_{ij}$ as described above. Then, the evasive force can be defined as the weighted sum of those N forces:

$$\mathbf{F}_e = \sum_{j}^{N} w_{ij}\mathbf{F}_{ij}, \tag{8}$$

where the weighting factor w_{ij} gives a higher priority to the most imminent collisions.

The second approach, which differs from existing solutions, is to apply the forces sequentially (iterative approach). The idea here is as follows. Let P_j be a pedestrian in the set $CP_i^{t_\alpha}$ and let $\mathbf{F}_{ij}$ be the evasive force that is exerted on P_i. Let us also, for notational convenience, name this force $\mathbf{F}_{\mathrm{v}}$. Then, we determine whether the pedestrian P_i still collides with each one of the remaining $N-2$ pedestrians in the set, after (virtually) applying the force $\mathbf{F}_{\mathrm{v}}$. In other words, we first estimate the desired velocity of P_i by including $\mathbf{F}_{\mathrm{v}}$ into the (4). Next, we iterate over the $N-2$ pedestrians in the set and compute the time of collision tc between P_i and each one of the selected $N-2$ pedestrians. If a collision still exists (i.e. $0 \leq tc \leq t_\alpha$), we compute the avoidance force as before and then add it to $\mathbf{F}_{\mathrm{v}}$.

We repeat this process for each one of the N pedestrians in the set $CP_i^{t_\alpha}$. Then, the total evasive force $\mathbf{F}_e$ of pedestrian P_i is determined as the average of all the forces $\mathbf{F}_{\mathrm{v}}$. Experiments have confirmed that by applying the forces sequentially a smoother and more realistic avoidance behaviour is achieved, as forces are only added if they are still required to avoid further collisions.

5 Implementation

This section provides implementation-specific details regarding our proposed pedestrian simulation model.

Efficient Collision Prediction. During each simulation step, we have to determine for each pedestrian P_i which other pedestrians are on collision course with him/her given a certain anticipation time. A naive implementation scheme would be to iterate over all other pedestrians checking whether a collision will take place. However, a more efficient implementation is to prune the search based on some cutoff distance, e.g. the maximum distance that the pedestrian P_i can travel given its anticipation time and maximum speed. Then, P_i only has to consider a limited number of pedestrians for potential collisions. Proximity computations to these pedestrians can be accelerated using a spatial hash data structure for answering nearest neighbor queries (see for example [5]).

Adding Variation. To increase the realism of the simulation some variation and irregularity is needed among the virtual pedestrians. Thus, a noise force $\mathbf{F}_n$ is also introduced in our model. This force, on one hand, allows us to take into account random variations in the individual behaviours, as well as incorporate mistakes that individuals make while avoiding each other. On the other hand, such a force is also needed to avoid artifacts that arise from symmetrical patterns, as well as to resolve extreme cases where two pedestrians have exactly opposite directions. More realism can also be achieved by varying several of the parameters of our model, such as the anticipation time of each pedestrian or the preferred speed.

Time Integration. The result from our proposed method is a system of positions, velocities and forces. To simulate this system, we first discretise our model in time by choosing a time step Δt for our simulation. Then, in each cycle of the simulation, we compute for each pedestrian P_i the sum of all the forces that are acted upon P_i and update its position and velocity using numerical integration. We also update the pedestrian's orientation θ_i by inferring it from the weighted average of its desired velocity and the current velocity.

6 Results

We have implemented our proposed method to test its applicability in real-time applications and validate the quality of the generated motions. All experiments

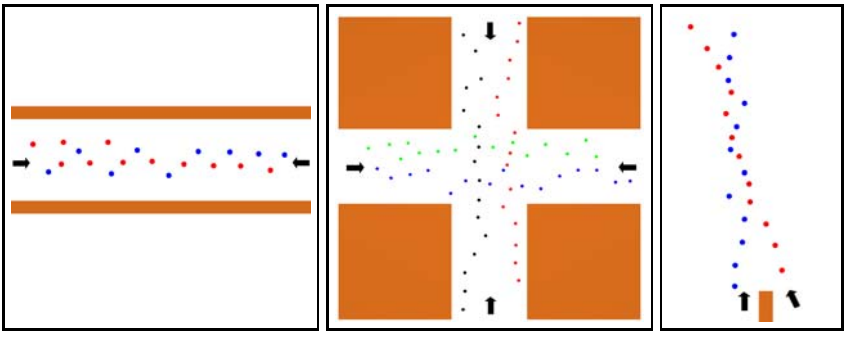


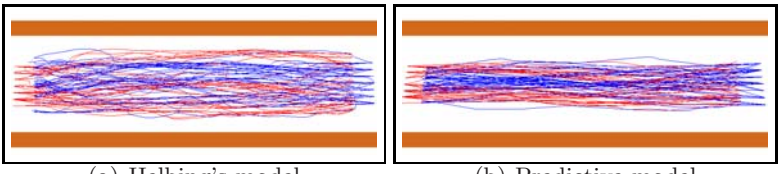
Fig. 3. Different simulation scenarios. Left: Hallway scenario. Middle: Four groups interacting at a crossing. Right: Crossing pedestrian flows.

were performed on a 2.5 GHz Intel Xeon CPU. Note that although this machine has four CPU cores, only one core was used for path planning.

We first calibrated the parameters of our model by performing a number of simple test-case scenarios as proposed in [17]. During these scenarios we recorded for each time step of the simulation, the position and the direction of each pedestrian. Then, a benchmark tool was implemented to analyse the quality of the generated motions and detect unrealistic behaviours. For that reason, a number of quantitative quality metrics have been devised. In particular, we used the integral of the square of the curvature to measure the smoothness of a pedestrian's path. We also computed the time it takes for a pedestrian to reach its goal, the length of the path it follows, as well as the average speed at which the pedestrian moves. Since we strive for energy-efficient motions, the total acceleration of the pedestrian and the total degrees it turned were also reported to indicate the amount of movement and turning effort spent by the pedestrian [17]. From the derived statistics, we determined optimal settings for our model. We experimentally confirmed that a smoother avoidance behaviour is achieved by applying the evasive forces iteratively, as explained in Section 4.4. The experiments showed that, using this approach, only a limited number of potential collisions N needs to be taken into account (2-5).

Having performed the initial calibration of our model, we ran a diverse set of simulation scenarios (Fig. 3). We refer the reader to `http://people.cs.uu.nl/ioannis/pam` for the results we obtained in simulating the aforementioned scenarios as well as for other simulations. In all of our experiments, we observed the phenomenon of dynamic lane formation that has been widely studied in pedestrian and crowd literature. In our model, though, the characters anticipate future collisions and adapt their motions in advance, evading others in a human-like manner. Even in crowded scenarios, our approach leads to a smooth and natural flow. Pedestrians scan the environment to find a free space to move into. Other individuals follow and a queue (lane) is formed that efficiently resolves the congestion.

We have also compared our approach to the Helbing's social force model [7] and the Reciprocal Velocity Obstacle (RVO) approach [11]. For a fair comparison,



(a) Helbing's model (b) Predictive model

Fig. 4. Example paths for the hallway scenario

we first optimised the two models following an approach similar to the one described above. In all test-cases, we noted that pedestrians using Helbing's model tend to adapt their motions rather late in order to avoid a potential collision. Thus, the resulting paths have a high curvature and the amount of unnecessary movements performed by the pedestrians is large. With the predictive approach, though, pedestrians plan early for collisions, avoiding unnecessary detours (see Fig. 4 for an example). In the RVO, each pedestrian adapts its velocity such that no collision will take place with other pedestrians and the environment. In low-density crowds, the method works very well. However, as soon as the environment becomes crowded, the RVO produces unrealistic motions. The pedestrians have to frequently speed up, slow down or change their orientations to safely reach their goals. On the contrary, our method ensures a smooth flow and more energy efficient motions.

We also quantitatively compared the models using our proposed evaluation metrics. Table 1 summarizes the corresponding statistics for the hallway scenario (note that similar results were obtained for all the other scenarios). A t-test was performed afterwards to determine how significant the results were. The analysis has shown that, indeed, our approach leads to shorter paths compared to the paths generated by Helbing's model, $p < 0.01$. The predictive approach ensures time-efficient motions allowing the pedestrians to move with significant higher speeds and follow much more smoother paths than pedestrians in Helbing's model (the differences in time, speed and smoothness are statistically significant, i.e. $p < 0.01$). In addition, using our approach, the number of interactions between the characters is reduced, which leads to more energy-efficient movements. Pedestrians have to spend less effort to avoid potential collisions and thus, the total acceleration is significantly lower compared to Helbing's model, $p < 0.01$. The same also applies to the turning effort of the pedestrians.

Statistical analysis has also confirmed our empirical observations regarding the RVO. Although no significant differences were found for the average travelling time and the speed of the pedestrians, our approach led to considerably less-curved paths than the RVO, $p < 0.01$. The analysis also indicated that the amount of the movement effort is lower using the predictive model compared to the amount spent by the pedestrians in the RVO, $p < 0.01$. Similarly, there is a significant difference in the amount of turning effort between pedestrians that employ our predictive avoidance method and pedestrians that use the RVO, $p < 0.01$.

Besides the quality of the generated motions, we are also interested in the performance of our proposed approach. To demonstrate its usability in real-time

Table 1. Statistics for the hallway scenario derived from the simulation of 100 agents

	Time		Path Length		Avg Speed		Smoothness		Total Accel		Degrees Turned	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Helbing	30.33	3.71	37.12	2.34	1.23	0.09	1.08	1.08	55.73	28.37	323.22	225.79
Predictive	25.05	2.10	35.14	1.79	1.40	0.05	0.06	0.08	9.4	6.01	76.62	52.29
RVO	24.51	1.79	34.56	0.35	1.41	0.01	0.10	0.07	18.75	10.17	106.2	56.14

interactive applications, we simulated 1,000 fully animated characters in a virtual park environment. Each character enters the park through a random gate and has to advance toward a randomly selected exit. To increase the realism of the simulation, we varied the preferred and maximum speeds among the characters, as well as the anticipation time of each character and the size of its personal space. The simulation was updated at 10 fps, whereas a parallel thread was used to render the characters and the scene at 30 fps. Our model led to smooth motions and thus, we did not have to decouple the simulation from the rendering. Since, on average, the CPU was busy for only 25% of each time step, we conclude that our model can be used for real-time simulation of thousands of characters. Finally, we also simulated the interactions of pedestrians at the crosswalks of an outdoor virtual city environment. Our approach resulted in a smooth avoidance behaviour and a visually pleasing simulation.

7 Conclusion and Future Work

In this paper, we presented a novel approach for simulating the collision avoidance behaviour of interacting virtual pedestrians. The intuition behind our approach is based on observed behaviour of real people. Each virtual character scans the environment to detect future collisions with other characters. It predicts how these collisions will take place and tries to resolve them in the current simulation step by making the most efficient move. In all of our experiments, the characters exhibit smooth behaviour and evade each other in a natural way. In addition, our proposed method is relatively easy to implement and yields to real-time performance.

For future work, we plan to automatically calibrate our model and validate our approach by exploiting existing video and motion capture data, as proposed in [12]. We would also like to combine our system with existing global path planning approaches. This will allow us to scale our method to very large and complicated environments. Furthermore, using these approaches, our framework can be easily extended to capture a wide variety of crowd characteristics. For example, we can include in our simulations coherent groups of characters, or characters that wander through a virtual environment without any specific goal.

Acknowledgments

This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

References

1. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Transactions on Graphics 25(3), 1160–1168 (2006)
2. Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. Computer Graphics Forum 23, 509–518 (2004)
3. Geraerts, R., Overmars, M.: The corridor map method: A general framework for real-time high-quality path planning. Computer Animation and Virtual Worlds 18, 107–119 (2007)
4. Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., Manocha, D.: Real-time navigation of independent agents using adaptive roadmaps. In: ACM symposium on Virtual reality software and technology, pp. 99–106 (2007)
5. Shao, W., Terzopoulos, D.: Autonomous pedestrians. Graphical Models 69(5-6), 246–274 (2007)
6. Reynolds, C.W.: Steering behaviors for autonomous characters. In: The proceedings of Game Developers Conference, pp. 763–782 (1999)
7. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Physical Review E 51, 4282–4286 (1995)
8. Helbing, D., Buzna, L., Johansson, A., Werner, T.: Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transportation Science 39(1) (2005)
9. Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: SCA 2007: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 99–108 (2007)
10. Paris, S., Pettré, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. Computer Graphics Forum 26(3), 665–674 (2007)
11. van den Berg, J.P., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: ICRA, pp. 1928–1935. IEEE, Los Alamitos (2008)
12. Pettré, J., Ondrej, J., Olivier, A.H., Crétual, A., Donikian, S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In: SCA 2009: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 189–198 (2009)
13. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphics Forum 26, 655–664 (2007)
14. Lee, K., Choi, M., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: SCA 2007: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 109–118 (2007)
15. Goffman, E.: Relations in public: microstudies of the public order. Basic books, New York (1971)
16. Zipf, G.K.: Human Behavior and the Principle of Least Effort. Addison-Wesley, Reading (1949)
17. Singh, S., Kapadia, M., Faloutsos, P., Reinman, G.: Steerbench: a benchmark suite for evaluating steering behaviors. Computer Animation and Virtual Worlds (2009)

Applying Affect Recognition in Serious Games: The PlayMancer Project

Maher Ben Moussa and Nadia Magnenat-Thalmann

7 Route de Drize, 1227, Carouge/Geneva Switzerland
Ph.: +41 22 379 12 13
{benmoussa,thalmann}@miralab.unige.ch

Abstract. This paper presents an overview and the state-of-art in the applications of 'affect' recognition in serious games for the support of patients in behavioral and mental disorder treatments and chronic pain rehabilitation, within the framework of the European project PlayMancer. Three key technologies are discussed relating to facial affect recognition, fusion of different affect recognition methods, and the application of affect recognition in serious games.

Keywords: Affective Computing, Emotion Recognition, Affect Recognition, Emotion Fusion, Affect Fusion, Serious Games.

1 Introduction

Affect recognition is a multi-disciplinary research involving, for example, psychology, machine learning, computer vision, signal processing, speech analysis, and so on. Affect recognition research can generally be composed into three types: vision based; audio based; and bio-sensors based types. By combining these affect recognition modalities, researchers believe in achieving higher accuracy as such multimodal communicative signals work in a complementary way [48].

Multimodal affect recognition can be applied in different scenarios including video conferencing, intelligent homes, medical rehabilitation, driver monitoring, etc. In the PlayMancer project, multimodal affect recognition, in combination with serious games, is applied in two medical rehabilitation scenarios: treatment of behavioral and mental disorders; and chronic pain rehabilitation [38][39][40][[41][42]. This paper focuses on the application of affect recognition in serious games for rehabilitation, in particular, the facial affect recognition system that is being developed for the PlayMancer project with the utilization of multiple modalities.

2 Background

The following sections describe the state of the art of serious games and affect recognition.

2.1 Serious Games

The term Serious Games was introduced by Abt [47] in 1970, before even the popularity of video games. Although the term was initially referred for board and card

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 53–62, 2009.

games, his description of 'serious' also applies to modern computer games. He suggests that serious games are games that have explicit and carefully thought-out educational purposes and are not intended to be played primarily for amusement.

Serious games are nowadays also used for medical therapy. Beale and Kato et al [43][44] developed serious games for helping adolescents and young adults with cancer, where as Coyle et al [45] and Brezinka et al [46] developed serious games for patients suffering from anxiety, depression and behavioral disorder.

2.2 Face-Based Affect Recognition

For us as human beings, face plays an important role in communicating and understanding the affective state and intensions of each other. Human emotions are universal and culturally independent as Charles Darwin noted, as early as one and half century ago [1]. Paul Ekman [2] supported the universal nature of emotions and has distinguished six universal emotions derived from facial expressions: fear, disgust, anger, sadness, surprise and joy. While it is very easy for humans to detect faces analyze the facial expressions, and interpret its affects, developing a system that automatically does the same interpretation is extremely difficult. Due to its usefulness and challenges, this field of research has been very active in the last three decades and much breakthrough has been discovered [49][50]. When we speak about facial expression analysis, we must distinguish between two main streams: facial affect analysis and facial muscle motion analysis, where the former is the aim of the PlayMancer project.

Facial affect recognition system usually involves several steps as shown in Fig. 1 - processing of the static images or video inputs and then extracting the affective state of the detected faces. Firstly, the faces are detected and extracted from the input, followed by the extraction of the facial features from these faces. The resulting facial feature information is analyzed to recognize the affective state of the person.

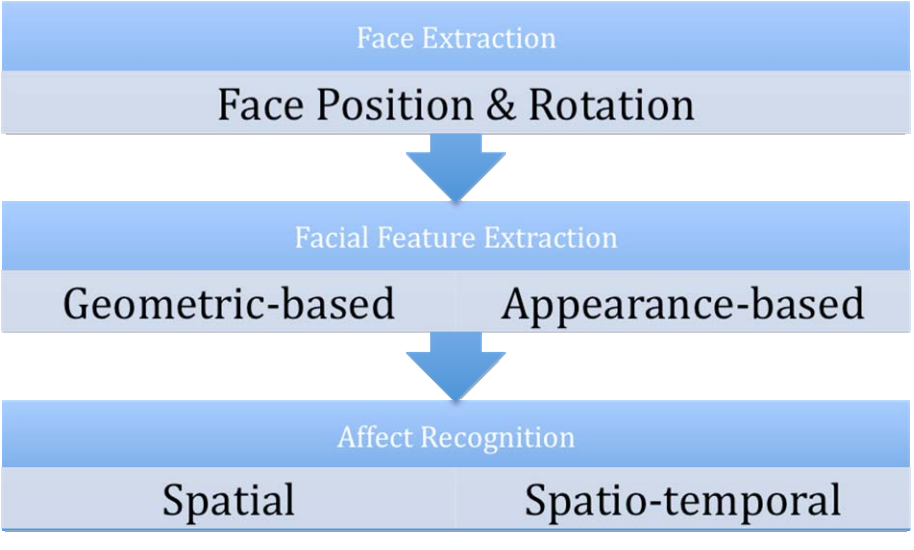


Fig. 1. Facial Affect Recognition Process

2.2.1 Face Extraction

Face extraction is the step where all the regions in the scene containing a human face are detected. There are several notable approaches to develop a robust facial detection system based on e.g., statistical learning techniques applied on appearance features [17][18][19][20]. Most commonly employed facial detection system is Viola and Jones [15], where they developed a robust real-time face detector based on a set of rectangle features and trained by the machine learning technique AdaBoost. A detailed survey can be found in [16].

2.2.2 Facial Feature Extraction

Facial feature extraction is the process of detecting and tracking of the facial features from the extracted faces. We can distinguish between two types of facial feature: 'geometric' features and 'appearance' features. Geometric features represent the shape of the facial components (such as mouth, nose, eyes, etc) and the locations of these feature points (corners of mouth, nose, eyes, etc), as exemplified in [21][22][23][3]. Appearance features represent the appearance changes of the face skin (wrinkles, furrows, etc). In these approaches, Gabor filter [28] is often applied to extract facial changes as multi-scale, multi-orientation Gabor wavelet coefficients [24][25]. Other research suggests that hybrid approach that integrates geometric and appearance features (combining geometric features and appearance features) can achieve better results for some facial expressions [26][27].

A large number of implementations of facial expression analysis do not only detect geometric features, but also track them over time. Most of these implementations utilize standard optical flow algorithms (Lucas-Kanade[6] or Kanade-Lucas-Tomasi[4]) to track facial feature points [7], while others use color-based observation model such as [29]. To address the errors caused by noise, change in lightning, clutter, etc., in sequential data, many feature tracking implementations apply 'sequence state' estimation techniques to improve and correct the data. These approaches rely on statistics to estimate the correct value of parameters in present of noise, e.g. Kalman filtering and particle filtering. Zhang & Ji [30] and Gu & Zi [31] utilized kalman filtering in facial feature tracking and Patras & Pantic [29] utilized particle filtering. Patras & Pantic [5] and Su et al [32] combine particle filtering with spatial belief propagation to enforce the correlation between different facial features (mouth, nose, eyes, etc).

2.2.3 Affect Recognition

The information acquired from facial feature extraction is used to recognize the facial affective state of the person in the scene. Two main techniques for affection recognition are 'spatial' and 'spatio-temporal' based approaches. In the spatial approach, the spatial information of the detected facial features in one frame is processed to recognize the affect. Examples of spatial approach are [23] where Cohen et al applied *Bayesian network* to recognize the six basic facial emotions, and [33] where *Ford support vector machines* was used to recognize the facial emotions. In spatio-temporal approaches, the temporal information of the frame sequences is used, such as in the work by Cohen et al [23] and Bartlett et al [25] where *HMM* was applied in the temporal domain to recognize the affect and in the work of Cohn et al [34] where a *rule-based* classifier was used.

2.3 Fusion of Affect Recognition Modalities

Humans communicate and perceive multimodal communicative signals in a complementary way. Therefore, in a similar manner, when developing a system that handles different affect recognition modalities, we cannot handle the inputs retrieved from each modality independently. In addition, such a system should handle imperfect data also fuse the input retrieved from different modalities according to a training model. Most approaches use probabilistic models for training using previously observed data in order to fuse the multimodal input. Some approaches rely on simple *Bayesian interference* [35] and while some on probabilistic graphical models such as *Hidden Markov Models*, *Bayesian networks* and *Dynamic Bayesian networks* [36][37].

A lot of research has been conducted in the fusion of different affect recognition modalities. Some researchers have concentrated on audio-visual fusion where affect recognition from voice and face are fused to create a more reliable system [8][9][10][11][12]. Other researchers have mainly concentrated on fusing affect recognition from bio-signals and voices [13][14]. To the best of our knowledge, there has never been an affect recognition system using the combination of face, voice, and bio-signal modalities. These modality fusion researches can be further classified into two main approaches for fusion of affective states: 'feature level' fusion and 'decision level' fusion. In feature level, the data is fused directly after the feature extraction. Feature vectors from each modality are fused and affective states are classified by one global classifier [10][11][12]. In contrast, in the decision level, affective states are classified for each modality by its local classifier and the results from these local classifiers are later fused in the decision layer [8][9][13].

3 PlayMancer Project

In the PlayMancer project, a serious game for rehabilitation is developed by *Serious Games Interactive*[1]. The scenarios of this serious game are described in section 3.1 below. This game is influenced by the emotions of the patients with the focus on the affects, as described in section 3.2. For affect recognition, there will be three different affect recognition modalities: face, audio, and bio-sensors. This paper focuses on face-based modality as described in section 2.2.2. Further, the fusion of different modalities was described in section 2.2.3.

3.1 Scenarios

In the PlayMancer project, there are two rehabilitation scenarios: Treatment of behavioral and mental disorders and chronic pain rehabilitation. For the treatment of behavioral and mental disorders, the PlayMancer consortium is developing serious games together with the psychiatric department of Bellvitge hospital in Barcelona for treating patients with Eating Disorders (ED) and Pathological Gambling (PG) [42]. The game scenario is an adventure game called "Islands" where the patient is confronted with various challenging situations. The affective state of the patient has a strong influence on the game. The aim is to change underlying attitudinal,

[1] http://www.seriousgames.dk/

cognitive-behavioral and emotional processes of patients (in other words, to teach the patient how to control his own emotions) as imposed by the clinical goals.

For chronic pain rehabilitation, the same game scenario of "Islands" is used with design changes to address the musculoskeletal pain disorders. Here, the aim was to improve physical functions, and to some extent, of pain-related maladaptive cognitions (i.e. fear of movement). This scenario is developed together with the rehabilitation centre Roessingh in Enschede (the Netherlands)

3.2 Targeted Affective States

For the game scenarios and clinical domains in the PlayMancer project, the following affective states and mood have been identified as useful:

- Joy
- Anger
- Sadness
- Surprise
- Neutral (calm)
- Boredom

Affect recognition modules are being trained by the collected data to recognize these affective states.

3.3 Facial Affect Recognition Module

The Facial Affect Recognition module provides feedback to the PlayMancer platform through the Multimodal Emotion Fusion component with respect to the affective state of the game player. This component functions by analyzing the facial expression of the game player who is recorded by camera, and classifies affects according to a training model.

Fig. 2. Facial Feature Tracking

For the affect recognition, the component relies on geometric feature extraction. The facial feature points are detected in the initial stage and are tracked over time as shown in Figure 2. The detection of the feature points is based on [1] where Gentle-Boost algorithm is applied on using Gabor filter of the face image. The tracking of the detected points relies on motion-based tracking techniques where the feature points tracking is implemented using the Kanade-Lucas-Tomasi algorithm [4][6]. For the error recovery, our approach is similar to [32] where it is handled by particle filtering with spatial belief propagation.

For the affect recognition, our approach is to develop spatial affect recognition using SVM and spatio-temporal based on HMM. It will result in a comparative study regarding these classification models and their combination with fusion models.

3.4 Affect Fusion Module

The Affect Fusion module (Figure 3) fuses emotion output retrieved from various emotion recognition components and provides feedback to the PlayMancer platform with respect to the affective state of the game player. This component fuses the affective information retrieved from different components according to a training model.

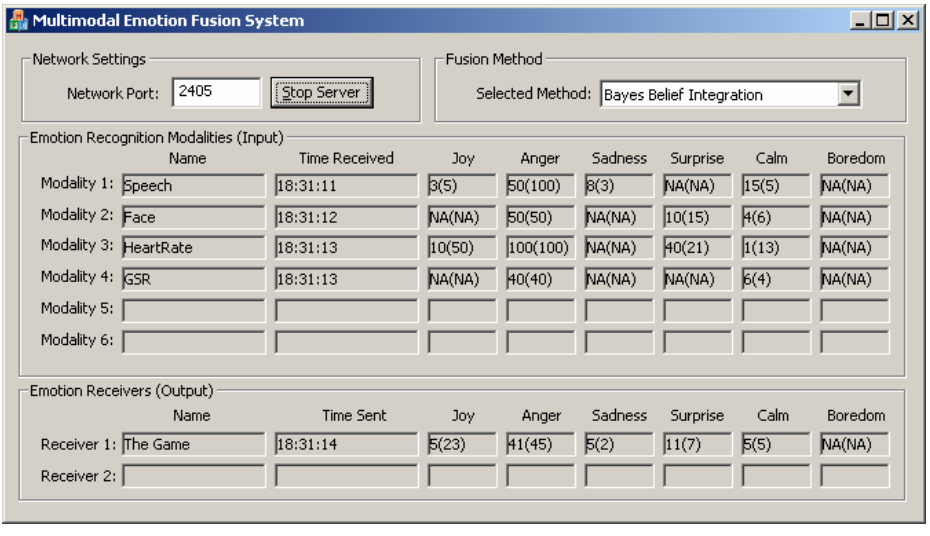


Fig. 3. Affect Fusion Module

In the PlayMancer project, the Multimodal Emotion Fusion component fuses face, speech and bio-signals. The classification happens on the decision level, where emotions from each modality are classified by the modality's local classifier (as can be seen at Figure 4).

For the affect recognition, our approach is to develop and evaluate affect fusion using simple *Bayesian interference* and using probabilistic graphical model *Hidden Markov Models*. This will also result in a comparative study regarding these fusion models.

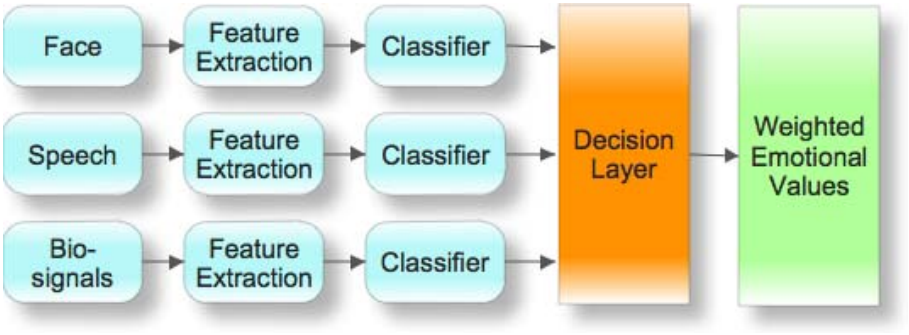


Fig. 4. Architecture of the Multimodal Emotion Fusion component

4 Conclusion

In this paper, we have highlighted major approaches for serious games in medical domain and also in the field of facial affect recognition and multimodal affect fusion.

We discussed the development of affect recognition systems for the PlayMancer project and how it is applied to serious games intended for rehabilitation of patients with treatment of behavioral and mental disorders and chronic pain rehabilitation.

The goal of this study is to evaluate different affect recognition approaches in combination with serious games, as described in Sections 3.3 and 3.4. Our ongoing aim is to find the optimal combination for facial affect classifiers and multimodal affect fusion classifiers.

Acknowledgement

The project is funded by the European research projects PlayMancer (FP7-ICT-215839-2007). The authors wish to thank all the members of the project consortium for their support.

References

1. Darwin, C.: The expression of the emotions in man and animals John Murray, London (1872)
2. Ekman, P.: Emotion in the Human Face. Cambridge University Press, New York (1982)
3. Vukadinovic, D., Pantic, M.: Fully automatic facial feature point detection using Gabor feature based boosted classifiers. In: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 2 (2005)
4. Tomasi, C., Kanade, T.: Detection and tracking of point features Technical Report CMU-CS-91-132, Carnegie Mellon University (1991)
5. Patras, I., Pantic, M.: Particle filtering with factorized likelihoods for tracking facial features. In: Proceedings of Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 97–102 (2004)

6. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: International joint conference on artificial intelligence, vol. 3, pp. 674–679 (1981)
7. Cohn, J., Reed, L., Ambadar, Z., Xiao, J., Moriyama, T.: Automatic analysis and recognition of brow actions and head motion in spontaneous facial behavior. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 1 (2004)
8. Zeng, Z., Hu, Y., Roisman, G.I., Wen, Z., Fu, Y., Huang, T.S.: Audio-visual spontaneous emotion recognition. In: Huang, T.S., Nijholt, A., Pantic, M., Pentland, A. (eds.) ICMI/IJCAI Workshops 2007. LNCS, vol. 4451, pp. 72–90. Springer, Heidelberg (2007)
9. Pal, P., Iyer, A., Yantorno, R.: Emotion detection from infant facial expressions and cries. In: 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2006 Proceedings, vol. 2 (2006)
10. Sebe, N., Cohen, I., Gevers, T., Huang, T.: Emotion recognition based on joint visual and audio cues. In: 18th International Conference on Pattern Recognition, ICPR 2006, vol. 1 (2006)
11. Song, M., Bu, J., Chen, C., Li, N.: Audio-visual based emotion recognition-a new approach. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol. 2 (2004)
12. Zeng, Z., Tu, J., Pianfetti, B., Liu, M., Zhang, T., Zhang, Z., Huang, T., Levinson, S.: Audio-visual affect recognition through multi-stream fused HMM for HCI. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 2 (2005)
13. Kim, J., Bee, N., Wagner, J., Andre, E.: Emote to win: Affective interactions with a computer game agent. In: Lecture Notes in Informatics (LNI)-Proceedings, vol. 50 (2004)
14. Kim, J., André, E., Rehm, M., Vogt, T., Wagner, J.: Integrating information from speech and physiological signals to achieve emotional sensitivity. In: Ninth European Conference on Speech Communication and Technology (2005)
15. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
16. Yang, M., Kriegman, D., Ahuja, N.: Detecting faces in images: A survey. IEEE Transactions on Pattern analysis and Machine intelligence, 34–58 (2002)
17. Rowley, H.: Neural Network-Based Face Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23–38 (1998)
18. Sung, K., Poggio, T.: Example-based learning for view-based human face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(1), 39–51 (1998)
19. Pentland, A., Moghaddam, B., Starner, T.: View-based and modular eigenspaces for face recognition. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings CVPR 1994, pp. 84–91 (1994)
20. Schneiderman, H., Kanade, T.: A statistical model for 3d object detection applied to faces and cars. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Los Alamitos (2000)
21. Gokturk, S.B., Bouguet, J.Y., Tomasi, C., Girod, B.: Model-based face tracking for view independent facial expression recognition. In: Proc. IEEE Int'l Conf. Face and Gesture Recognition, pp. 272–278 (2002)
22. Pantic, M., Patras, I.: Dynamics of facial expression: Recognition of facial actions and their temporal segments from face profile image sequences. IEEE Transactions on Systems, Man, and Cybernetics, Part B 36(2), 433–449 (2006)

23. Cohen, I., Sebe, N., Cozman, F., Cirelo, M., Huang, T.: Coding, analysis, interpretation, and recognition of facial expressions. Computer Vision and Image Understanding, Special Issue on Face Recognition (2003)
24. Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with gabor wavelets. In: Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pp. 200–205 (1998)
25. Bartlett, M., Braathen, B., Littlewort-Ford, G., Hershey, J., Fasel, I., Marks, T., Smith, E., Sejnowski, T., Movellan, J.: Automatic analysis of spontaneous facial behavior: A final project report Institute for Neural Computation MPLab TR 2001, 8 (2001)
26. Tian, Y., Kanade, T., Cohn, J.: Facial expression analysis Handbook of face recognition, pp. 247–276. Springer, Heidelberg (2005)
27. Zhang, Z., Lyons, M., Schuster, M., Akamatsu, S.: Comparison between geometry-based and Gabor-wavelets-based facial expression recognition using multi-layer perceptron. In: Proceedings of the 3rd. International Conference on Face & Gesture Recognition, p. 454 (1998)
28. Daugman, J.: Complete discrete 2-d Gabor transforms by neural networks for imageanalysis and compression. IEEE Transactions on Acoustics, Speech and signal processing 36(7), 1169–1179 (1988)
29. Patras, I., Pantic, M.: Tracking deformable motion. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 2 (2005)
30. Zhang, Y., Ji, Q.: Active and dynamic information fusion for facial expression understanding from image sequences. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 699–714 (2005)
31. Gu, H., Ji, Q.: Information extraction from image sequences of real-world facial expressions. Machine Vision and Applications 16(2), 105–115 (2005)
32. Su, C., Zhuang, Y., Huang, L., Wu, F.: A two-step approach to multiple facial feature tracking: Temporal particle filter and spatial belief propagation. In: Proceedings Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 433–438.
33. Ford, G.: Fully automatic coding of basic expressions from video. Technical Report INC-MPLab-TR-2002.03, Machine Perception Lab, Institute for Neural Computation, University of California, San Diego (2002)
34. Cohn, J., Kanade, T., Moriyama, T., Ambadar, Z., Xiao, J., Gao, J., Imamura, H.: A comparative study of alternative FACS coding algorithms. Robotics Institute, Carnegie Mellon University (2001)
35. Chanel, G.: Emotion assessment for affective-computing based on brain and peripheral signals, University of Geneva (2009)
36. Garg, A., Naphade, M., Huang, T.: Modeling video using input/output Markov models with application to multi-modal event detection. Handbook of Video Databases: Design and Applications (2003)
37. Garg, A., Pavlovic, V., Rehg, J.: Boosted learning in dynamic Bayesian networks for multimodal speaker detection. Proceedings of the IEEE 91(9), 1355–1369 (2003)
38. Kalapanidas, E., Watanabe, H., Davarakis, C., Kaufmann, H., Aranda, F., Lam, T., Ganchev, T., Konstantas, D.: PlayMancer: A European Serious Gaming 3D Environment. In: Conjustion with ICSOFT 2008 the 2nd International Workshop on e-health Services and Terchnologies, pp. 51–59 (2008)
39. Conconi, A., Ganchev, T., Kocsis, O., Papadopoulos, G., Fernández-Aranda, F., Jiménez-Murcia, S.: PlayMancer: A Serious Gaming 3D Environment. In: Proceedings of the 2008 International Conference on Automated solutions for Cross Media Content and Multichannel Distribution, pp. 111–117 (2008)

40. Kalapanidas, E., Fernandez-Aranda, F., Jimenez-Murcia, S., Kocsis, O., Ganchev, T., Kaufmann, H., Davarakis, C.: Games for a cause, games for health: Where to go from here Issue on New challenges for the video game industry. Comunications and Strategies 73, 105–120 (2009)
41. Kalapanidas, E., Davarakis, C., Fernández-Aranda, F., Jiménez-Murcia, S., Kocsis, O., Ganchev, T.: PlayMancer: Games for Health with Accessibility in Mind (2009)
42. Jiménez-Murcia, S., Fernández-Aranda, F., Kalapanidas, E., Konstantas, D., Ganchev, T., Kocsis, O., Lam, T., Santamaría, J.J., Raguin, T., Breiteneder, C., Kaufmann, H., Davarakis, C.: Playmancer Project: A Serious Videogame as Additional Therapy Tool for Eating and Impulse Control Disorders (2009)
43. Beale, I., Kato, P., Marin-Bowling, V., Guthrie, N., Cole, S.: Improvement in cancer-related knowledge following use of a psychoeducational video game for adolescents and young adults with cancer. Journal of Adolescent Health 41(3), 263–270 (2007)
44. Kato, P., Cole, S., Bradlyn, A., Pollock, B.: A video game improves behavioral outcomes in adolescents and young adults with cancer: a randomized trial. Pediatrics, Am Acad Pediatrics 122(2), e305 (2008)
45. Coyle, D., Matthews, M., Sharry, J., Nisbet, A., Doherty, G.: Personal investigator: a therapeutic 3D game for adolecscent psychotherapy. Interactive technology and smart education 2(2), 73–88 (2005)
46. Brezinka, V., Hovestadt, L.: Serious games can support psychotherapy of children and adolescents. In: Holzinger, A. (ed.) USAB 2007. LNCS, vol. 4799, pp. 357–364. Springer, Heidelberg (2007)
47. Abt, C.: Serious Games. The Viking Press, New York (1970)
48. Jaimes, A., Sebe, N.: Multimodal Human-computer Interaction: A Survey. Computer Vision and Image Understanding 108(1-2), 116–134 (2007)
49. Pantic, M., Bartlett, M.: Machine analysis of facial expressions Face recognition, pp. 377–416 (2007)
50. Tian, Y., Kanade, T., Cohn, J.: Recognizing action units for facial expression analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(2), 97–115 (2001)

A Comparative Review of Reactive Behaviour Models as Proposed in Computer Graphics and Cognitive Sciences

Stéphane Donikian

INRIA Rennes - Bretagne Atlantique,
Campus de Beaulieu, 35042, Rennes Cedex, France
stephane.donikian@inria.fr
http://www.irisa.fr/prive/donikian

Abstract. This paper is presenting different models introduced in computer graphics to model reactive human behaviours. Then it continues with the presentation of different theories in cognitive sciences about reactive behaviours. We are then comparing models introduced in computer graphics with theories introduced in cognitive sciences.

1 Introduction

Behavioural models have been introduced to design autonomous entities like organisms and living beings. In this paper, we will focus on anthropomorphic characters, and we will compare different formalisms introduced in computer sciences during the last twenty years with theories proposed by cognitive scientists. In computer sciences, the goal in modelling human behaviours is not to reproduce the human brain and body complexity but to propose a software architecture able to reproduce believable human behaviours in dedicated activity contexts such as car driving or pedestrian navigation. To model a human behaviour, it is necessary to address different issues such as perception, memory, face and body motion control including emotion expressivity, and action selection. We will focus here in what is called decision or action selection even if it could not exist apart from perception and action.

2 Behavioural Models Proposed in Computer Sciences

2.1 First Generation of Reactive Behavioural Models

One can place historically the first works on behavioural animation at the end of the 80s with in particular the article of C. Reynolds on the animation of clouds of birds [24]. A first set of approaches was studied in parallel in the literature, to the middle of the 90s, concerning the definition of the decision-making part of the behavioural model:

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 63–74, 2009.

Sensor-effector: This approach defines the behaviour of objects from a set of sensors and effectors interconnected by a network of intermediate nodes transforming the passing-by information [34,33,22]. This type of approach includes also the neural network models [31]. The way an object behaves depends on the perception which it has of its environment and the way this perception is passed on through the network in the effectors which produce the movement of objects. This approach has the advantage to be able to generate a very important quantity of different movements (choice of the set of parameters) but stays on the other hand at a level of very weak abstraction. Furthermore this kind of model functions like a black box in which it is impossible to modify the slightest parameter without having to resume the complete process of configuration. Furthermore, the percentage of good controllers is decreasing in an exponential way with the growth of the number of parameters to be taken into account; it is then misleading to try defining a complex behaviour with this approach.

Behaviour rules: as the previous approach, the approach by rules of behaviour takes as input data information conveying a certain perception of the environment and produced as output a control over the motion of objects. Here, the behaviour of objects is defined by a set of rules [24]. The possible behaviour of an object can be represented by a decision tree, every branch representing a different behaviour. An action satisfying the conditions of the current environment will be chosen by the application of a tree traversal algorithm. Behaviours allowed by this approach are of a higher level of abstraction than the previous one. The heart of the problem in this approach lies in the weighing of the various behaviours. The simplest solution consists in making an implicit choice on the order of rules, but this solution does not allow the specification of complex behaviours. Weighing the different branches of the tree allows not to privilege always the same rule, whereas the consideration of experts' hierarchy allows to confront several concurrent behaviour, the final choice staying at the upper level of the tree [9,32].

Finite automaton: the automaton defines the various possible sequences of behaviour [29]. This approach finds very quickly its limits when it is a question of modelling a little complicated behaviour. While modelling the behaviour of a car driver, Booth et al. [5] have shown the necessity of using several state-machines whose execution should be coordinated.

The report that we can make on these various works is that they are specific models conceived to be applied to particular cases, in which objects and their environment are relatively simple and the perception and action capabilities are limited. Besides, there is a big disparity in the behaviour allowed by each of the approaches. Either the level of abstraction is very weak and the only behaviours that may be specified are reflex ones (sensor-effector approach), or the level of abstraction is more important but then the environment is reduced, completely defined, and the perception and the action of the entity are in contrary of a very weak level. These models remain in every case relatively simple, with limited

fields of perception and action, and besides they do not take into account the temporal aspect, while it is essential.

2.2 Second Generation Handling More Complex Reactive Behaviours

To report the decision-making complexity, it is necessary to handle collectively the continuous and discreet aspects, to coordinate the concurrent behaviours and to manage their organizational structure. That is why the first two approaches were rather fast abandoned for the benefit of an approach based on state-machines in their parallel and hierarchical versions: stacks of state-machines (EPFL, Switzerland) [23], set of communicating state-machines (PaT-Nets) (University of Pennsylvania, the United States) [2], hierarchy of parallel state-machines (HCSM) (University of Iowa, the United States) [1] and parallel transition systems organized into a hierarchy (HPTS) (IRISA, France) [13]. This approach is now also common in the animation and game industries. More recently, models have been proposed to handle uncertainty either by using Bayesian programming [15] or decision network [35] which is a generalization of Bayesian network. Even if those models are hierarchical in their structure, they do not allow managing the coordination between concurrent behaviours as only pure parallelism without any relation between parallel decision networks can be managed.

Let us detail now HPTS which stands for Hierarchical Parallel Transition System. HPTS is a reactive behaviour description model combining the state-machine approach and the multi-agent approach. Every state of every automaton is an agent possessing an internal state, being able to receive stimuli and to provoke them in reaction. Any active state of the system is receiving a stream of input data, delivering a stream of output data, and possessing a continuous and discreet control. A state of the system is either a terminal state, or a composite state. G. Moreau [20] proposed an implementation of the HPTS model, including a programming language and a compiler allowing to generate an equivalent C++ code. Each state has its own activity status with three possible values: *active, suspended, inactive* and dedicated functions are executed when the status is changing: *begin, end, wait, resume*. The control parameters allow influencing either by external or internal decision the behaviour to be adopted.

A filtering function manages the coherence of the actions proposed by the active states of the parallel sub-state-machines. When concurrent behaviours are proposed by different sub-behaviours, this function is in charge of making a choice and to deliver a unified behaviour to the upper layer. An automaton is always executed after its child allowing a behaviour to select and to mix the propositions supplied by its sub-components. So, concurrent behaviours can work in parallel and be arbitrated. This model was successfully used to model the behaviour of car drivers and pedestrians [12,30]. This language was thus extended [11] to manage random choice between different possible behaviours with weighting rates and to manage also the reaction time between the decision and the action.

2.3 Competitive and Cooperative Approaches for Action Selection

According to Clancey [8], all the goal directed behaviours are not always obtained by inference or compilation, but some actions may reproduce simply cultural motives while others are coordinated without deliberation just based on attention and adaptation. A person does not perform several tasks strictly in parallel, but several tasks are going to take place by merging attentively several parallel interests. It is useful to offer a mechanism allowing to execute different behaviours in parallel without programming by hand the synchronization of their execution. Moreover, while describing a behaviour it should not be necessary to take into account all the possible behaviours that may interfere with it as this may evolve depending on the context.

Two kinds of action selection algorithms exist: the cooperative approach and the competitive approach. The cooperative approach allows combining several potentialities of action, whereas the other one keeps only an action among all the potentially practicable ones. The first approach, because of the expressiveness of the combination of the actions is mainly based on the usage of arithmetical functions on quantitative data, whereas the second approach, thanks to the use of cost functions coupled with information of a more qualitative nature, is going to allow determining the best action to realize according to the context.

HPTS proposes both approaches through the notion of integration function at every level of the hierarchy. It is in fact a programmable function, and the programmer so has the choice at each level to make a competitive or a cooperative choice, according to the nature of the behaviour to be handled.

Concerning the competitive approach, action selection algorithms were proposed in the field of multi-agent systems, most being extensions of the ASM algorithm (Action Selection Mechanism) introduced by P. Maes [19]. We can quote the algorithm of V. Decugis and J. Ferber [10] who is addressing an interesting problem: how to combine in the same model reaction and planning abilities in real-time applications (in the particular case of robotics). They so proposed a hierarchical ASM in which the lowest levels concern basic reactive behaviours and the highest levels integrate more complex behaviours (in the same way than the hierarchical parallel state-machine approach). At every level, a mechanism of arbitration must be used to choose among the actions proposed by the parallel ASMs.

B. Rhodes proposed another hierarchical extension of the ASMs called PHISH-Nets [25]. This model authorizes the use of parameterized actions and allows defining relations between actions which are either of conflicting type or of antecedent type. These models allow performing reactive planning but the inconvenience concerns the request of an exhaustive specification of all the possible interactions between actions. Furthermore, it exists in these models cases for which no decision can be taken.

F Lamarche [17] suggested automating the mixture of behaviours in respect of their relative importance, by using the cooperative approach. To illustrate the problem of this work, let us take a concrete example: a person is in front of a table, she reads a book while drinking a coffee and smoking a cigarette. This

sentence describes a relatively simple behaviour for a human being, however in term of behavioural animation, it raises diverse problems. On one hand, it is constituted by three *independents* behaviours which take place simultaneously: read a document, drink a coffee, and smoke a cigarette. In the previous models, describing this type of behaviour is relatively complex, as it is necessary to be able to order them so as to respect certain number of constraints such as not drinking the coffee while having a cigarette in the mouth, nor manipulating pages of the book with the hand holding the cup of coffee. On the other hand, when one want to coordinate three behaviours described separately, it is necessary to avoid the re-coding of a specific behaviour dedicated to their simultaneous realization.

The objective is to be able to launch them and let them be executed together automatically according to circumstances, envy and physical constraints of the virtual human. The biggest synchronization problems come from the use of the internal resources of the agent. By internal resources, we mean the gaze, the hands, the feet, or more generally any dependence, physical or not, limiting the parallel execution of different behaviours. To reach this objective, three new notions were introduced within the model HPTS: resources, priorities and degrees of preference, to give birth to the HPTS++ model[17]. The mutual exclusion on the use of resources allows defining at any time all the compatible behaviours. The priority is a coefficient which indicates the importance of a behaviour in a given context. This coefficient can illustrate either the adequacy (activation) or the inadequacy (inhibition) between the behaviour and the environment. The priority can be dynamically defined and evolve upon time.

The degree of preference is a numerical value associated with every transition of a state-machine. This value allows describing the inclination of the behaviour to use this transition when the associated condition is true. Thus, depending on its value, this coefficient has different meanings. If the value is positive, the transition favours the realization of the behaviour. If the value is negative, this transition does not favour the realization of the behaviour, but allows describing a way to adapt the behaviour while releasing some resources needed by another behaviour or to coherently terminate the behaviour. A null value will not influence the behaviour.

This system allows describing behaviours in a fine way, with all their adaptation possibilities. The description of a new behaviour, thanks to the mechanism of interlocking avoidance, does not require the knowledge of the already described behaviours. The coordination of all the active behaviours becomes so generic and automatic. A complete description of the automatic algorithm of coordination is described in [17] and fully illustrated by the example of the *reader / drinker / smoker*.

3 Theories Proposed in Cognitive Sciences

The hierarchical nature of the levels of behaviour is admitted today collectively. For A. Berthoz [3], the complex problem of the motor control was resolved by

a hierarchy of levels of control which, each, is applied on internal models of the system which precedes it. The theory of control in behavioural psychology, such as described by R. Lord and P. Levy [18], resumes the principle of force-feedback loops, while spreading it to all the behavioural processes from the simple task of visual servoing to the regulation of the social behaviour. For Lord and Levy, the genericity of the retroaction loops for the description of the behaviour results from the hierarchical nature of the systems of control, even if the nature of the activities of control can be very different according to the levels. The common point between all these levels lies in the comparison between a perceived state and an expected state and in the preservation of the error in acceptable limits. Every level only knows the lower level and receives only errors already elaborated by internal models which compare a state wished with a real state.

The brain contains several schemas (mechanisms of simulation of action) of the body that are independent from the real body itself [3]. The superior organs which make the decisions do not work inevitably by arranging directly sensory information. These centres know only the state of the lower levels of execution which contain models of the levels which they control and especially which estimate the errors between what they imposed and what is executed.

Lord and Lvy [18] emit the hypothesis that control of the human processes is produced by a mutual interaction between two mechanisms *top-down* and *bottom-up*. The top-down control is abstract and is strongly connected to the current intentions and to the established plans whereas the bottom-up control is more guided by the data, so relieving the information supplied by the perceptive system to recognize the conflicts. The ascending regulation is a necessary complement to the downward control mechanism, so assuring that the cognitive system will note and will answer correctly the physiological and psychological demands. Nevertheless, the detection of conflicts at the hierarchical level corresponding to the management of the tasks should not have to create too much cognitive overload to that corresponding to the symbolic reasoning. On the other hand, the conflicts are capable of interrupting the thought to redirect the attention and are generative of new models capable of surmounting them.

This hierarchical nature of the behaviour is taken into account in the models of type hierarchical parallel state machines with a management of the bidirectional exchanges (downward control then ascending information feedback) between levels. The models of type hierarchical ASM and PHISH-Nets also integrate a hierarchical structure but with only an ascending mechanism of communication allowing the superior level to choose among the actions proposed at the lower level.

3.1 Attentional Mechanisms for Action Selection

Shallice [27] has proposed two kinds of attentional mechanisms for the selection of actions: the *Supervisory Attentional System* and the *Contention Scheduling*. The contention scheduling system is used to select in an automatic manner actions to be done when the situation is routine. The typical example is the car driving: when passed the learning phase, the driver can change its speed and

orientation, while talking with passengers or listening radio. To be able to do that, the action selection should be done automatically as an important part of the cognitive load is used for the speech information processing.

The driver can make a gear, which implies a simultaneous manipulation of the clutch pedal and the control lever, while conversing with his passengers, listening to the radio or still thinking of scheduling its activities of the week-end. When he is confronted with a new situation or when parameters of a situation creature of habit change, the *Contention Scheduling* cannot intervene any more, because it cannot select himself the adequate actions. In that case, it is the *Supervisory Attentional System* which forces choices made by the *Contention Scheduling*.

In the whole of the models presented previously, only HPTS ++ can manage in an automatic way this coordination of several threads of activity thanks to its management of the physical resources.

3.2 Mechanisms of Activation and Inhibition

Brain is a simulator of action, a generator of hypothesis. Anticipate and predict consequences of actions based on the memory of past ones are some of its fundamental properties. There is neither any mechanism of perception apart from action, nor any mechanism of attention apart from action selection. To decide is not only to choose between several solutions, it consists also in blocking undesired behaviours, which means inhibiting [3]. For J. Kühl [16], the treatment of tasks is protected from interruptions, particularly for complex tasks. Inhibiting mechanisms are very important as they prevent unnecessary elements to enter the working memory[1]. A mechanism of willingness is used to avoid any interruption of an ongoing behaviour by using a direct inhibition of all competitive behaviours [18]. R. Lord and P. Levy are suggesting the use of several complementary rules:

Proposition 1: the instantiation of a goal is going to privilege the close information in the mean of category:
- (a) by increasing the speed in which we can reach this information;
- (b) by increasing the envy to access such information.

Proposition 2: the activation of a goal is going to avoid the instantiation of competitor goals:
- (a) by increasing the latency of their activation;
- (b) by reducing the envy to access to such information;
- (c) by producing negative primary effects.

Proposition 3: the normal realization of a purpose deactivates the referring structures, by releasing the cognitive system of the positive and negative effects.

Proposition 4: the repeated failure of a goal can deactivate the referring structures, by releasing the cognitive system of the positive and negative effects.

[1] Active memory dealing with both the treatment and preservation of requested short term information.

Proposition 5: the automatic follow-up and detection of conflicts is an important bottom-up control mechanism which integrates biologic needs as well as treatments at the symbolic level.

Deliberation means that there are at least two solutions. The brain has to make a clear choice between solutions in competition. A. Berthoz postulates that a double mechanism of modularity and hierarchy is used:

- Modularity, because the basal ganglia of the thalamocortical base are specialized in the control of the movements of the glance or the gesture or the memory.
- Hierarchy, because it is possible that these parallel modules present a hierarchical crosswise connection

A. Berthoz [3] presents three kinds of architectures that have been proposed in the literature for action selection:

1. Subsumption: actions endowed with a hierarchical index allowing selecting them automatically with respect to a fixed order.
2. Organization of actions in a network: reverse inhibitive connections between all actions in a distributed action network that is connected to sensors and effectors.
3. Supervisor or central selector: it selectively activates the circuits. This approach allows to drastically reduce the number of connections and offers at the same time more flexibility. It combines the advantages of modularity and centralism.

Link can be made between the first approach and behavioural rules and finite automaton approaches presented in 2.1. The second approach is represented by the sensor-actuator approach presented in 2.1 and by the competitive action selection mechanisms presented in 2.3. As far as we know, the only representative of the third approach is HPTS++ presented in 2.3.

3.3 Theory of Activity

For W.J. Clancey [8], the theory of activity is a previous form of the situated cognition that will be presented in the next paragraph. The three levels of the theory of activity are:

- the activity (motivations): forces operating on the decision taking process;
- the action (goals): they define what should be done;
- the operations (conditions): they define how it should be done.

M. Sierhuis [28] is defining the activity as follows :

An activity is a collection of actions performed by one individual, socially constructed, situated in the physical world, taking time, effort, and application of knowledge. An activity has a well-defined beginning and end, but can be interrupted.

To understand the activity, it is necessary to take into account the fundamentally social aspect of the human activity. Our activity as human being is always forged, forced and is made significant by our continual interactions with the worlds of the work, the family and the other communities to which we belong. An activity is so not only a thing that we make, but a way of interacting. Any human activity is deliberated, but a purpose is not necessarily a problem which must be resolved and every action must not be necessarily motivated by a task to be carried out. W.J. Clancey [7] takes the example of a person who listens to some music while driving her car on her way back home. This activity is a part of the driving practice for many persons but is in no way a necessary sub-purpose to reach their destination.

According to Clancey, the motives underlying the human behaviour were imperfectly characterized through the problem solving theory introduced by Allan Newell into his unified theory of cognition [21]. All the goal oriented behaviours are not obtained by inference or compilation. Certain actions simply reproduce cultural motives, whereas some others are coordinated without deliberation by using mechanisms of attention and adaptation. Clancey discusses the notion of parallelism of tasks. A person does not perform multitasking in parallel, but several tasks are going to take place by merging attentively several parallel interests. He says that the conceptualization of this notion is still primitive and that its nature is not developed in the neuropsychological theories.

In [4] A. Berthoz and J.L. Petit evoke the existence of at least five closed loops of neuronal circuits (thalamus-cortex-ganglions of the base) which work in a autonomous way and in parallel so allowing to control the movements of eyes, limbs, memory and feelings, these closed loop systems coordinating together in a dynamic way. Contrary to the problem solving approach, Clancey defends that an activity is not necessarily interrupted when needed, as, for example, if another activity becomes pressing or if an external condition comes to interrupt what the person was making. A mechanism of competitive activation is implied. The starting and ending of an activity are more subtle than a purely goal oriented decision. Clancey still says that there is no opposition between sequential and parallel notions but a coupling of them. The coupled sub-systems should be organized together in real-time. The parallelism allows organizing several activities at the same time, whereas the serialism forces the treatment of ordered forms of sequences of action. The parallelism is fundamental to couple behaviours and to temporally order them, in particular when it implies several sensori-motor modalities.

In the whole set of behavioural models, only the hierarchical parallel state-machines allow this combination of the sequential and parallel aspects, and within these models only HPTS ++ can take into account the various sensori-motor modalities and the activation/inhibition principles in a generic way.

3.4 Embodied and Situated Cognition

It is common to consider that only the symbolic approach can be used to model the cognitive behaviours, because these last ones were often introduced as being

problems to be solved that are formulated under the condition / action form. However, this traditional approach of the cognitive sciences and the artificial intelligence was confronted with problems such as that of the Chinese room [26]: the fact of having symbols and rules allowing to manipulate them does not define any kind of intelligence, because at no time there is a usage of their meaning. The intelligence can be described as the sum of the previous physical experiences of the agent acquired through its interaction with its environment.

In other words, the intelligence of an agent is based on its previous interactions with the physical world. Brooks [6] so introduced the hypothesis of the physical grounding which postulates that the intelligence of an agent must be based on the interaction between the physical agent and its environment. According to Harnad [14], the symbols have to acquire their common sense, it is what he calls the symbol grounding problem: *an artificial system based completely on the manipulation of symbols never glimpses semantics which is associated with them.* To find a solution to this problem, he proposes that symbols are based on a process of invariant extraction from sensori-motor signals, defined in three stages:

Iconisation: the processing of the signals into iconic representations or icons;
Discrimination: the capacity to judge if two inputs are identical or different and if they are different, in what way they differ;
Identification: the capacity to assign a unique answer to a class of inputs, treating them quite as equivalents of a certain way.

Whereas there is a consensus in the situated and embodied cognition community to say that the traditional approach of the cognition as computational process is incomplete even erroneous. There is, on the other hand, no consensus on the definition of the foundations of this new approach.

4 Conclusion

The main assessment which can be taken on the links existing between action selection models and the works coming from cognitive sciences on the action selection mechanisms is that there are very few relations. Only a few discussions can be found in the literature in behavioural animation or in artificial intelligence about the cognitivist foundations of the proposed models, set apart the recent field of the embodied and situated cognition. This last research field is really multidisciplinary and is gathering researchers in cognitive sciences, in robotics and in artificial intelligence. Finally, within the reactive models proposed in computer sciences, HPTS++ is the sole model that integrates most of the presented notions.

However, decision is not self-sufficient and should be integrated in a multi-layered behavioural architecture. Usually, those architectures are based on rule-based systems and are referring to classical unembodied and unsituated cognitive sciences. New cognitive architectures should be developed taking into account embodiment and situation. Such an architecture should be hybrid to manage together bottom-up and top-down control.

Acknowledgment

This work has been funded in part by the European Commission under grant agreement IRIS (FP7-ICT-231824).

References

1. Ahmad, O., Cremer, J., Hansen, S., Kearney, J., Willemsen, P.: Hierarchical, concurrent state machines for behavior modeling and scenario control. In: Conference on AI, Planning, and Simulation in High Autonomy Systems, Gainesville, Florida, USA (1994)
2. Badler, N.I., Reich, B.D., Webber, B.L.: Towards personalities for animated agents with reactive and planning behaviors. In: Petta, P., Trappl, R. (eds.) Creating Personalities for Synthetic Actors. LNCS (LNAI), vol. 1195, pp. 43–57. Springer, Heidelberg (1997)
3. Berthoz, A.: Emotion and Reason. The cognitive neuroscience of decision making. Oxford University Press, Oxford (2006)
4. Berthoz, A., Petit, J.L.: Physiologie de l'action et Phénoménologie. Odile Jacob (2006)
5. Booth, M., Cremer, J., Kearney, J.: Scenario control for real-time driving simulation. In: Fourth Eurographics Workshop on Animation and Simulation, Politechnical University of Catalonia, pp. 103–119 (September 1993)
6. Brooks, R.A.: Elephants don't play chess. Robotics and Autonomous Systems 6, 3–15 (1990)
7. Clancey, W.J.: Situated Cognition: On Human Knowledge and Computer Representations. Cambridge University Press, Cambridge (1997)
8. Clancey, W.J.: Simulating activities: Relating motives, deliberation, and attentive coordination. Cognitive Systems Research 3(3), 471–499 (2002)
9. Coderre, B.: Modeling behavior in petworld. In: Artificial life, pp. 407–420. Addison-Wesley, Reading (1988)
10. Decugis, V., Ferber, J.: Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. In: Autonomous Agents 1998, pp. 354–361. ACM, New York (1998)
11. Donikian, S.: HPTS: a behaviour modelling language for autonomous agents. In: Autonomous Agents 2001, Montreal, Canada, pp. 401–408. ACM Press, New York (2001)
12. Donikian, S., Moreau, G., Thomas, G.: Multimodal driving simulation in realistic urban environments. In: Progress in System and Robot Analysis and Control Design. LNCIS, vol. 243, pp. 321–332. Springer, Heidelberg (1999)
13. Donikian, S., Rutten, E.: Reactivity, concurrency, data-flow and hierarchical preemption for behavioural animation. In: Eurographics Workshop on Programming Paradigms in Graphics, pp. 137–153 (1995)
14. Harnad, S.: The symbol grounding problem. Physica D 42, 335–346 (1990)
15. Le Hy, R., Arrigoni, A., Bessière, P., Lebeltel, O.: Teaching bayesian behaviours to video game characters. Robotics and Autonomous Systems 47(2-3), 177–185 (2004)
16. Kuhl, J.: Volitional aspects of achievement motivation and learned helplessness: towards a comprehensive theory of action control. In: Maher, B.A. (ed.) Progress in experimental personality research, vol. 13, pp. 99–171. Academic Press, New York (1984)

17. Lamarche, F., Donikian, S.: Automatic orchestration of behaviours through the management of resources and priority levels. In: Autonomous Agents and Multi Agent Systems, Bologna, Italy. ACM, New York (2002)
18. Lord, R.G., Levy, P.E.: Moving from cognition to action: A control theory perspective. Applied Psychology: an international review 43 (3), 335–398 (1994)
19. Maes, P.: Situated agents can have goals. In: Maes, P. (ed.) Designing Autonomous Agents, pp. 49–70. MIT Press, Cambridge (1990)
20. Moreau, G., Donikian, S.: From psychological an real-time interaction requirements to behavioural simulation. In: Computer Animation and Simulation 1998. SpringerComputerScience, pp. 29–44. Springer, Heidelberg (1998)
21. Newell, A.: Unified theories of cognition. Harward University Press (1990)
22. Ngo, J.T., Marks, J.: Spacetime constraints revisited. In: Kajiya, J.T. (ed.) Computer Graphics (SIGGRAPH 1993 Proceedings), Anaheim, California, vol. 27, pp. 343–350 (1993)
23. Noser, H., Thalmann, D.: Sensor based synthetic actors in a tennis game simulation. In: Computer Graphics International 1997, Hasselt, Belgium, pp. 189–198. IEEE Computer Society Press, Los Alamitos (1997)
24. Reynolds, C.: Flocks, herbs and schools: A distributed behavioral model. In: SIGGRAPH 1987 (1987)
25. Rhodes, B.J.: Phish-nets: Planning heuristically in situated hybrid networks. Master's thesis, Massachusets Institute of Technology (1996)
26. Searle, J.R.: Minds, brains, and programs. Behavioral and Brain Sciences 3, 417–424 (1980)
27. Shallice, T.: Specific impairments of planning. Philosophical Transactions of the Royal Society 298, 199–209 (1982)
28. Sierhuis, M.: Modeling and Simulating Work Practice. BRAHMS: a multiagent modeling and simulation language for work system analysis and design. PhD thesis, University of Amsterdam (September 2001)
29. Magnenat Thalmann, N., Thalmann, D., Renault, O.: A vision-based approach to behavioral animation. Journal of Visualization and Computer Animation 1(1), 18–21 (1990)
30. Thomas, G., Donikian, S.: Virtual humans animation in informed urban environments. In: Proceedings of the Conference on Computer Animation (CA 2000), pp. 112–121. IEEE, Los Alamitos (2000)
31. Travers, M.: Animal construction kits. In: Artificial life, pp. 421–442. Addison-Wesley, Reading (1988)
32. Tu, X., Terzopoulos, D.: Artificial fishes: Physics, locomotion, perception, behavior. In: Computer Graphics (SIGGRAPH 1994 Proceedings), Orlando, Florida, pp. 43–50 (1994)
33. van de Panne, M., Fiume, E.: Sensor-actuator networks. In: Kajiya, J.T. (ed.) Computer Graphics (SIGGRAPH 1993 Proceedings), vol. 27, pp. 335–342 (1993)
34. Wilhelms, J., Skinner, R.: An interactive approach to behavioral control, pp. 1–8 (June 1989)
35. Yu, Q., Terzopoulos, D.: A decision network framework for the behavioral animation of virtual humans. In: Metaxas, D., Popovic, J. (eds.) Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, pp. 119–128 (2007)

Data Driven Evaluation of Crowds

Alon Lerner[1], Yiorgos Chrysanthou[2], Ariel Shamir[3], and Daniel Cohen-Or[1]

[1] Tel Aviv University, Israel
[2] University of Cyprus, Cyprus
[3] The Interdisciplinary Center, Israel

Abstract. There are various techniques for simulating crowds, however, in most cases the quality of the simulation is measured by examining its "look-and-feel". Even if the aggregate movement of the crowd looks natural from afar, the behaviors of individuals might look odd when examined more closely. In this paper we present a data-driven approach for evaluating the behaviors of individuals within a simulated crowd. Each decision of an individual agent is expressed as a state-action pair, which stores a representation of the characteristics being evaluated and the factors that influence it. Based on video footage of a real crowd, a database of state-action examples is generated. Using a similarity measure, the queries are matched with the database of examples. The degree of similarity can be interpreted as the level of "naturalness" of the behavior. Essentially, this sort of evaluation offers an objective answer to the question of how similar are the simulated behaviors compared to real ones. By changing the input video we can change the context of evaluation.

1 Introduction

The simulation of computer generated crowds has progressed in leaps and bounds from its conception more than two decades ago. In films, computer games and other virtual world applications we have seen simulations of every type of crowd imaginable, from flocks of dinosaurs to plain everyday pedestrians. Despite the fact that these crowds differ in size, behavior and method of simulation, the common goal remains the same; to simulate the most naturally looking crowd possible.

Different types of crowds may present different types of natural behaviors. Pedestrians do not behave nor look like people playing sports. How would one know if the simulation looks natural? The only means for evaluating the results of a simulation so far were subjective, e.g. looking at it and deciding whether it "looks natural". In general, this is a valid approach for the global "look-and-feel" of the simulation.

A crowd is defined by a large number of simulated individuals. In order to assess a simulated crowd's quality in full, it must be examined in detail, since a single individual whose behavior deviates from some "regular" or "normal" pattern may hinder the quality of the entire simulation. Even a human paying close attention to a simulation will find it difficult to measure the quality of all individual behaviors at all times. And so the need for an effective automatic measure becomes a necessity. Evaluating individual behaviors also depends on one's interpretation of the basic notion of how people behave. Moreover, multiple interpretations are valid, and different people may choose to focus on different aspects of the same behavior. How can we define what is "normal" or "natural"?

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 75–83, 2009.

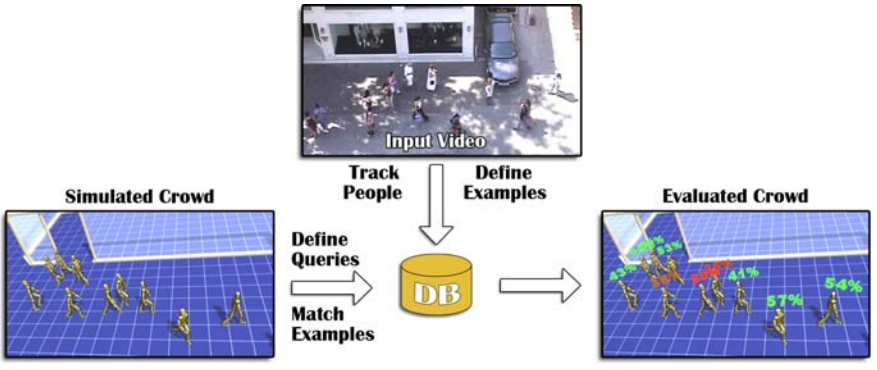


Fig. 1. Overview of context dependant evaluation: a specific context is defined by an input video. The video is analyzed and a database of examples is created. For a given simulation, each individual trajectory is analyzed and compared to the "normal" behaviors defined by the examples in the database. The results provide an evaluation score from 0 to 100 for each individual at any time (the numbers above each individual).

In this paper we utilize a data-driven approach to assess individual behaviors within crowds. The example set, which represents individual behaviors that are considered "normal", defines the context for the evaluation. The example set is created by examining videos of real crowds and analyzing the individual trajectories and the attributes that describe the conditions under which they occurred. During an evaluation, a similar analysis is applied to the simulated trajectories, which results in a set of queries that are compared to the example set. The similarity between a query and the examples provides the grading for the simulated individual's behavior in a specific point in space and time. Such an evaluation validates "natural" behaviors and points out potentially "curious" ones within a context (see Figure 1). For instance, different types of crowds can be evaluated using different videos of crowds of a similar nature. By evaluating the behavior of *individuals* within the crowd, specific moments in time and space can be automatically identified as problematic and possibly corrected. Finally, considering the comparisons as a whole, yields a global evaluation, which reflects on the overall quality of the simulation.

Our main contribution is a data driven approach for evaluating simulated crowds based on videos of real crowds. We present results on evaluating individual behaviors within crowds for short-term intervals, although long term ones could also be evaluated using a different measure. We use different videos and different types of simulations and show examples of how "curious" behaviors could be found automatically.

2 Previous Work

Simulating crowds is a challenge that has been examined in such diverse fields as computer graphics, civil engineering, sociology and robotics. The literature is rich with a variety of different approaches, all trying to simulate a natural looking crowd, [11, 18].

Considering the diversity of the methods, particularly the ones for simulating pedestrian crowds, emphasizes the complexity of the question "what makes a crowd seem natural?" For example, some works try to minimize the amount of effort required to reach a goal, [7, 17], which originates from the notion that people are individuals that have a goal and will try to follow an "optimal" path to reach it while maintaining their speed. Other works, such as [2, 5, 19], try to simulate the flow of the crowd and pay less attention to the correctness of individual trajectories. Different works focus on different aspects of crowds. All claim to simulate natural looking crowds, and to some degree they are all correct. It is unclear which attributes are more important when behavior is concerned and since the common evaluation method is subjective, then all opinions are valid.

Although some earlier works did try to assess the quality of individual trajectories, such as [15], only recently have people begun to pay attention to the evaluation of complex multi-character environments. Ennis et al. [3] evaluate the plausibility of pedestrian formations. They manually reconstruct crowd formations from annotated photographs, modify the positions and orientations of the people and perform a perceptual evaluation of the modified static images through user experiments. It would be difficult to extend their results to animated crowds. Singh et al. [16] gain insight into the quality of a simulator using predefined test scenarios and a measure that assigns a numerical value to the average number of collisions and the time and effort required for the agents to reach their goal. Pelechano et al. [12] explore the egocentric features that a crowd simulator should have in order to achieve high levels of presence and thus be used as a framework for validating simulated crowd behaviors. Paris et al. [10] use motion capture data of interactions between pedestrians for two ends. First, they analyze the data and learn parameters for more accurate collision avoiding maneuvers. Second, they validate the results by visually comparing simulated results to the captured ones.

Data-driven techniques have been used to simulate crowds. In the work of Lee et al. [8], *state-action* examples are obtained from crowd videos. The state of an agent includes the position and movement of nearby agents, its own motion, and certain environmental features. The action is a two-dimensional vector corresponding to the agent's resulting speed and direction. In the work of Lerner et al. [9] the state stores a similar set of attributes, only at greater detail. The action is a trajectory segment that can be concatenated to a simulated agent's trajectory.

In the vision community there are many works for clustering and classifying trajectories, some also detect abnormal ones [1, 4, 6, 13, 14]. There are several significant conceptual differences between these works and ours. First, they usually do not transfer the information learned about the trajectories from one scenario to another. Second, these works, for the most part, focus on the global characteristics of the trajectories, rather than the details. Therefore, they usually classify them as a whole, rather then evaluate the quality of trajectory segments. Last, the trajectories are usually considered on their own without accounting for the stimuli that may have influenced them.

3 Overview

The underlying concept of most crowd simulators can be described by the *state-action* paradigm. For each simulated individual (*subject person*) in a specific point in time and space, a *state S* is defined as a set of potentially influential attributes, such as the person's

position, speed and the direction and position of nearby individuals. Some function is then used to compare this state to predefined states, either explicitly or implicitly (e.g. a set of rules), and an *action A* is chosen and assigned to the subject person. The action can be, for example, a velocity vector or a trajectory segment.

An evaluator can be described in similar terms, aside from two key differences. First, in an evaluation process all of the state attributes, such as the full trajectories of the individuals, are known. This allows for a more accurate definition of the "natural" action that should be performed. Second, an evaluation requires a comparison between the action that should have been performed and the one that actually was performed, thus determining its quality.

An evaluation measure requires the definition of the state attributes S, the action representation A, and a similarity function between state-action pairs. The state S should include all of the attributes that potentially affect a person's decision to choose an action. For example, in the short-term measure we propose, we defined the state as the densities of the people in different regions surrounding the subject person. The action A is defined as the subject person's positions along a two second long trajectory interval.

In a preprocessing stage, the individuals observed in one or more input videos are tracked, their trajectories are analyzed and examples of state-action pairs are created. During an evaluation, the simulated trajectories are analyzed in a similar manner and *query* state-action pairs are defined. For each query, we search for the most similar examples in the example set using a similarity function. This is a two stage process. First, a set of examples whose states best match the query state are found. Then, their actions are compared. Our distance measure uses a normalized combination of both the state and action distances, in order to account for the cases where close matches for the state cannot be found. This two-stage process assures a context dependent evaluation of individual decisions, where the similarity to the closest example defines the quality of the decision embodied by the query (Figure 1).

4 Example Set and Simulations

To define a specific context for an evaluation we require a video of a crowd of a similar nature. The video should be shot from an elevated position and have a clear view of the crowd below. We implemented a simple manual tracker which requires only a few clicks of the mouse in order to track a person's trajectory. Different types of crowds are represented by different input videos (see Figure 2).

An example is defined from each time step along every trajectory of the input. We exclude only the trajectory parts in which the state and action cannot be fully defined. When adding the examples to a database, we use a greedy algorithm to cluster the examples and remove redundant ones.

We shot several videos of different crowds, Figure 2, according to which we built two databases. The first was created from two videos of a sparse crowd, whose combined length is 12 minutes and contains 343 trajectories. The second was created from a 3.5 minute long video of a dense crowd, which contains 434 trajectories. In our unoptimized implementation the time required to preprocess a sparse database was under an hour, while for a dense database over an hour. An average of about 1KB of data was stored per example.

Fig. 2. Sample frames from two input videos used to define example databases. One for dense crowds and one for sparse ones. The trajectories of people (some are shown in red) were tracked manually before analysis.

To illustrate the versatility of our evaluation, we implemented two types of simulations. A simple *rule-based* simulation, where the agents maintain their speed and direction and perform only gradual turns trying to minimize collisions and an *example-based* simulation [9], which consists of agents walking on their own or in small groups. In this simulation the agents usually walk along relatively smooth trajectories, however, abrupt changes in speed or direction do appear.

5 Evaluation of Short Term Decisions

A person constantly makes short-term decisions. These are the decisions that cause him to stop, turn, slow down or speed up. These decisions are not influenced by some global objective, but rather by the local conditions surrounding the person. The impact of these decisions is immediate and short lived. Their effects can be found in short segments of a person's trajectory.

We define a measure, which we term the *density measure*, whose state attributes consist of samples of the local densities of the people surrounding the subject person. The action is defined as a two second long trajectory segment. One second before and one second after the current position. The segment is aligned such that the position and orientation of the subject person in the middle of the segment is aligned with the origin of a global coordinate system.

To define the state we divide the area surrounding the subject person into regions, Figure 3, and for each region store samples of the number of people that appear in it over a short time. This yields a compact representation of the local changes in densities in the vicinity of the subject person. The motivation for using this state definition stems from the common belief that people's reactions are influenced by the density of the people in their immediate vicinity. We define a similarity function between a query state-action pair and an example pair. The function measures the similarity between the actions (differences in positions along the trajectories) and the distance between the states (differences in densities for the surrounding regions).

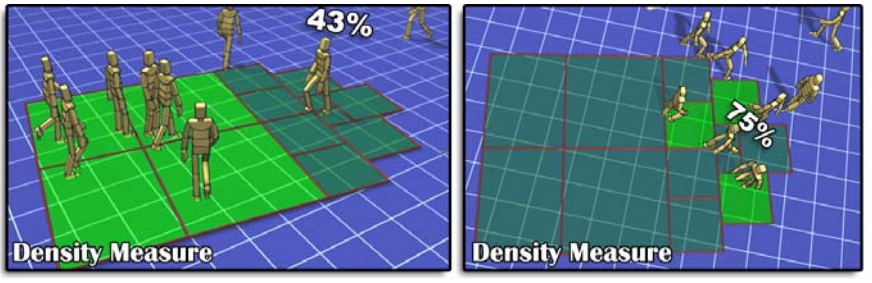


Fig. 3. In the density measure the state is composed of samples of the number of people in each one of the regions surrounding the subject person. Five samples in different times are used.

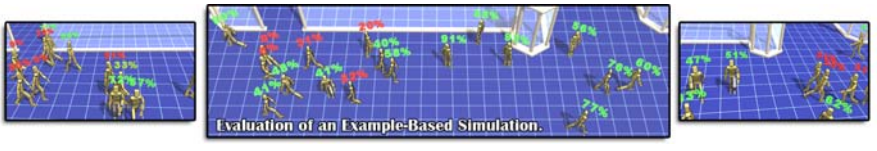


Fig. 4. A few examples from the evaluation of a crowd simulation using a sparse crowd video input and the density measure. The short-term decisions that did not find an appropriate match in the database are marked in red.

6 Results

In one of our experiments we used the database created by videos of a sparse crowd as input and evaluated the different crowds using the density measure. Results of evaluating an example-based simulation appear in Figure 4. The percentages represent the quality of the matches that were found. Zero percent means that no match was found and a hundred percent means that a perfect match was found. Low quality matches are highlighted in red. A close inspection of the evaluation results shows that, for the most part, low quality matches correspond to "curious" behaviors, traffic congestion, collisions or near misses. In Figure 4 center, the agent that received a 20% similarity value stopped walking abruptly and the four others marked in red, either walked towards an imminent collision or performed "conspicuous" evasive manoeuvres. The same can be said for most of the highlighted agents in Figure 4 left and right.

Figure 5 shows a quantitative comparison between the evaluation of the example-based simulation, the rule-based simulation and a real sparse crowd different from the input data. The columns represent ranges of similarity values (or evaluation scores), and the height of each column the relative number of queries which received a value in the range.

As can be seen, the real data received significantly better scores than the simulation. However, the example-based simulation which is collision free, contains interactions between the individuals and seems similar in nature to the input video, found a greater number of high quality matches compared to the rule-based simulation.

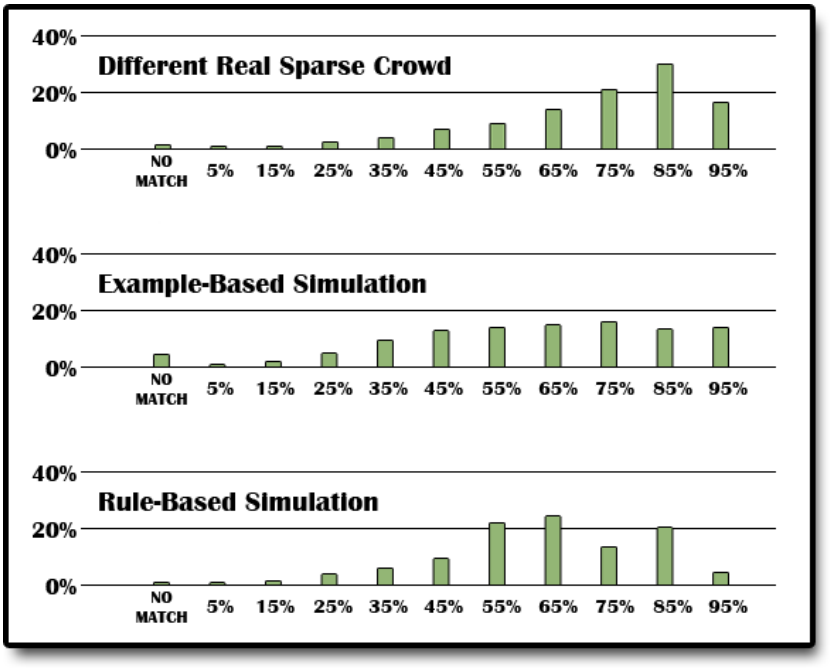


Fig. 5. A quantitative comparison between the distribution of evaluation scores when evaluating the example-based simulation, rule-based simulation and real data against the sparse crowd database. The horizontal axis represents the match values and the vertical axis the relative number of matched queries.

The most noticeable "curious" short-term decisions either involve collisions or evasive maneuvers whose goal is to avoid collisions. The method finds these and other "curious" behaviors, since similar examples were not found. On the other hand, in some cases the agents involved in collisions or near misses were deemed "natural". The reason is that a similar example where one person walked very close to another person was found in the database. Finding or not finding matches does not unequivocally mean that the behaviors are "natural" or "curious", but rather it raises the probability of them being so.

The time required to evaluate a crowd varies depending on the size of the database and the size of the simulated crowd. The simulations presented here were several minutes in length and contained a few dozen people in them. In our un-optimized implementation evaluating their short-term decisions required between 5 and 20 minutes. Using dedicated data structures to optimize the search for the best matching examples can significantly reduce the processing time.

7 Conclusions

We have presented a data-driven approach for evaluating individual behaviors in a crowd simulation. Such an approach gives the evaluation a context and a goal, since

it changes the fundamental question from "how natural is this crowd?" to the relatively simpler one of "how does it compare to this real crowd?". Using our method as a post processing stage for a simulation, it can focus the viewer's attention on the potentially "curious" behaviors in time and space, hence reducing the effort needed for evaluation. Another significant advantage of the data-driven model is the simplicity by which the model can be changed. Changing the type of crowd which appears in the input video effectively changes the behavior model without the need to modify the evaluation application. Finally, this technique is not limited to evaluating crowds generated by a specific simulator, but rather evaluates the trajectories regardless of their source, therefore it can also be used to locate potentially "curious" behaviors in real crowds.

One disadvantage of our model is the need to obtain the right video and to track individual trajectories in it. To the best of our knowledge, to date there are no adequate automatic trackers that can accurately track people in a crowd. We used manual tracking, which is a tedious job. However, each video requires processing only once and can be used any number of times. Another limitation of the data-driven approach is the fact that if a matching example was not found in the example set it does not necessarily indicate that the behavior is "curious". It could indeed be a natural behavior which was simply not observed in the input video.

References

[1] Brand, M., Kettnaker, V.: Discovery and segmentation of activities in video. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 844–851 (2000)

[2] Chenney, S.: Flow tiles. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 233–242. Eurographics Association Aire-la-Ville, Switzerland (2004)

[3] Ennis, C., Peters, C., O'Sullivan, C.: Perceptual evaluation of position and orientation context rules for pedestrian formations. In: Proceedings of the 5th symposium on Applied perception in graphics and visualization, USA, pp. 75–82. ACM New York, NY (2008)

[4] Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., Maybank, S.: A system for learning statistical motion patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(9), 1450–1464 (2006)

[5] Hughes, R.L.: The Flow of Human Crowds. Annual Review of Fluid Mechanics 35, 169–182 (2003)

[6] Johnson, N., Hogg, D.: Learning the distribution of object trajectories for event recognition. In: BMVC 1995: Proceedings of the 6th British conference on Machine vision, Surrey, UK, vol. 2, pp. 583–592. BMVA Press (1995)

[7] Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. Comput. Graph. Forum 23(3), 509–518 (2004)

[8] Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 109–118 (2007)

[9] Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by Example. Computer Graphics Forum 26(3), 655–664 (2007)

[10] Paris, S., Pettre, J., Donikian, S.: Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach. Computer Graphics Forum 26(3), 665–674 (2007)

[11] Pelechano, N., Allbeck, J., Badler, N.: Virtual Crowds: Methods, Simulation, and Control. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, San Francisco (2008)
[12] Pelechano, N., Stocker, C., Allbeck, J., Badler, N.: Being a part of the crowd: towards validating VR crowds using presence. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, vol. 1, pp. 136–142 (2008)
[13] Porikli, F.: Trajectory distance metric using hidden markov model based representation. In: IEEE European Conference on Computer Vision, PETS Workshop (2004)
[14] Porikli, F., Haga, T.: Event detection by eigenvector decomposition using object and frame features. In: CVPRW 2004: Proceedings of the, Conference on Computer Vision and Pattern Recognition Workshop (CVPRW 2004), vol. 7, p. 114. IEEE Computer Society Press, Los Alamitos (2004)
[15] Reitsma, P.S.A., Pollard, N.S.: Evaluating motion graphs for character navigation. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 89–98. Eurographics Association Aire-la-Ville, Switzerland (2004)
[16] Singh, S., Naik, M., Kapadia, M., Faloutsos, P., Reinman, G.: Watch Out! A Framework for Evaluating Steering Behaviors. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, p. 200. Springer, Heidelberg (2008)
[17] Sud, A., Andersen, E., Curtis, S., Lin, M., Manocha, D.: Realtime path planning for virtual agents in dynamic environments. In: Proc. of IEEE VR (2007)
[18] Thalmann, D., Musse, S.R.: Crowd Simulation. Springer, Heidelberg (2007)
[19] Treuille, A., Cooper, S., Popovic, Z.: Continuum crowds. ACM Trans. Graph. 25(3), 1160–1168 (2006)

Variety Is the Spice of (Virtual) Life

Carol O'Sullivan

GV2: Graphics, Vision and Visualisation Group
Trinity College Dublin

Abstract. Before an environment can be populated with characters, a set of models must first be acquired and prepared. Sometimes it may be possible for artists to create each virtual character individually - for example, if only a small number of individuals are needed, or there are many artists available to create a larger population of characters. However, for most applications that need large and heterogeneous groups or crowds, more automatic methods of generating large numbers of humans, animals or other characters are needed. Fortunately, depending on the context, it is not the case that all types of variety are equally important. Sometimes quite simple methods for creating variations, which do not over-burden the computing resources available, can be as effective as, and perceptually equivalent to, far more resource-intensive approaches. In this paper, we present some recent research and development efforts that aim to create and evaluate variety for characters, in their bodies, faces, movements, behaviours and sounds.

1 Introduction

Virtual characters can now be rendered and animated with increasing levels of realism for movies and games. However, no matter how compelling these characters appear individually, once two or more are simulated to form groups or crowds, new challenges emerge. These challenges may be in the way they physically interact with or react socially towards each other, but a particular challenge is to ensure that each individual character looks unique, i.e., that it does not appear to be obviously replicated or *cloned* many times in a scene. There is little more disturbing for the viewer than to see multiple instances of the same character performing the same actions and repeating the same noises or phrases over and over again. In the movie Madagascar, for example, artists created five different kinds of lemurs with 12 variations of hair type, giving 60 unique combinations [13] to create the illusion of a large group of individual animals. However, if the goal is to create realistic humans, creating a crowd of unique and realistic characters is extremely expensive in terms of people hours, memory resources and run-time computation costs.

Therefore, short of manually creating and/or capturing new models, motions and sounds for every single character, some level of cloning is required in most practical applications. Fortunately, tricks can be used to disguise cloned characters, perhaps exploiting graphics hardware to support the rapid generation of

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 84–93, 2009.

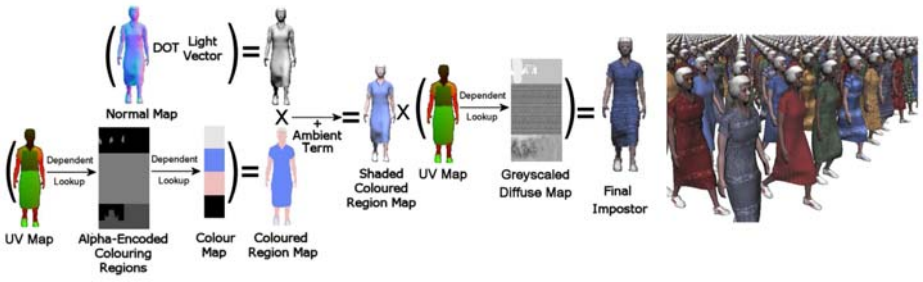


Fig. 1. (left) A typical rendering pipeline that supports variation, and (right) an example of adding variation to one character using 8 different diffuse textures [9] [10]

certain types of character variations [14]. For example, the types of variations that are important at the *macroscopic* level i.e., when viewing a crowd of people at a distance, are likely to be much different from *microscopic* factors, i.e., when viewing characters up close. As in most graphical applications, perceptual considerations are very important, and in our recent work we investigated what the effect of different variety schemes are, and how best to balance the need for variety with the practical considerations of the application [22] [23].

1.1 Colours and Textures

The easiest and least resource intensive measure for hiding cloned characters in a crowd is simply to vary their colours and/or textures. Using perceptual studies where people were asked to find clones in a scene as quickly as possible, we have found that both types of variation are effective at differentiating human character models who are otherwise identical. In other words, people found such 'colour clones' harder to find in a scene [22], while varying the colours and textures of only the top garments of characters was found to be as effective as full variation [23] (see Figure 3). Hardware accelerated per body part colour modulation [8] [9] [30] and texture variation [7] [10] are two of the most popular methods for variation in real-time crowd applications. Figure 1 shows a typical pipeline, while Figure 2 shows a typical crowd (from our Metropolis project) rendered in realtime using similar techniques.

1.2 Bodies

Varying body shapes is another means of generating lots of novel details in crowds. For example, a number of template human body scans could be used to build a space of human body shapes, where parameterisation could allow novel models to be created by morphing between these templates [1]. Several more such approaches have been proposed (e.g., [2] [3] [28]). However, such acquired body data has not yet been applied successfully in crowd simulation, perhaps due to the difficulties with rendering, clothing and animating the models effectively.

Fig. 2. A crowd scene from our *Metropolis* crowd system, where colour and texture variation have been used to generate a heterogeneous crowd

Nevertheless, with recent advances (e.g., [16]), the application to crowd simulation is rapidly becoming feasible. It still remains to be seen, however, whether the resulting body shapes and their retargetted motions are actually realistic enough for applications such as games and movies. Another useful source of variation is in the sex [21] (see Figure 4), age or emotional posture of a character (e.g., by making them more stooped or straight).

1.3 Faces

In our recent eye-tracking study of saliency in crowd scenes, we found that people look almost exclusively at the faces (and upper torsos) of virtual humans, when asked to determine whether clones are present in a scene or not [23]. In a set of validation experiments, we found that varying the textures of faces, e.g, adding beards and make-up (see Figures 3, 5 and 6), was as effective as more computationally expensive methods that varied the geometry of the face. These results are consistent with those reported by Sinha et al. [29], who present an overview of important features for face recognition. However, they explain that the shape of the head is also an important cue for recognition. There is clearly scope for further investigation of these effects and their potential exploitation for procedural facial variation.

Also important here is the effect of macroscopic and microscopic factors. At what distance is variation of external facial features (e.g., hair, head shape,

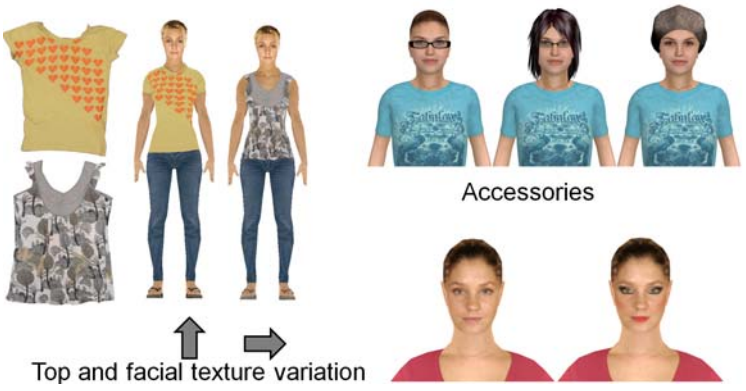


Fig. 3. Disguising clones using texture variations of top garments and faces is cheap and effective. Accessories, though more expensive, also add significantly to the perceived variety of a crowd [23].

Fig. 4. Creating variety by varying the 'male-ness' and 'female-ness' of virtual characters [21]

beard, and jaw-line) needed to make the identities of crowd members look different, and when are similarities between more subtle internal facial features more noticeable (e.g., noses, eyebrows, eyes, mouth)? In previous research designed to assess how the relative contributions of internal and external features change as a function of image resolution, it was found that the two feature sets reverse in importance as resolution decreases [19]. At what distance does facial variation cease to be of any importance and other cues become more informative?

1.4 Animations

So far we have only discussed varying the physical appearance of virtual characters - however, how they *move* is also important, as a population of people all walking in the same way will not be very compelling. Nevertheless, in our studies we have found that appearance clones are much easier to detect than motion clones [22], and even something as simple as making the characters walk out of step can make the task of finding clones significantly harder. This suggests

Fig. 5. Possible facial variations using hair, facial texture and accessories

that more resources should be expended on changing physical appearance than motions and it may be the case that, at the macroscopic level, using only a small number of clips (e.g., one female and one male) could be sufficient. However, as the number of motion clones and the time spent viewing a scene increases, so too do the detection rates. Therefore, methods for varying the motions of individuals are also desirable.

The most natural movements for virtual characters are derived from motion capture systems, which produce a large amount of data. Therefore, optimising the balance between realism, storage and character reactivity is an important issue. Finding a way to create a variety of realistic motions from a smaller subset of real motions is particularly challenging, and it is not always the case that the resulting motions are realistic. Furthermore, certain anomalies can be particularly noticeable, e.g., when a small and thin person's walk is applied to a model of somebody taller and stockier, or when a man's walk is applied to a female model [21].

There has been research in the animation and psychology communities into creating morphed motions for human characters. Giese and Poggio represent complex motion by linear combinations of a small number of prototypical image

Fig. 6. A crowd with variations in colour, textures (top garments and faces) and accessories (hair, hats and glasses)

sequences (e.g., walking, running or limping motion combinations) [15]. Brand and Hertzmann's style machines consist of a learned model that can be used to synthesise novel motion data that interpolate or extrapolate styles, such as ballet or modern dance [5]. Hsu et al also used learning to translate the style of one motion to another, while preserving the overall properties of the original motion - e.g., they could transform a normal walk into one with a sneaky crouch [18]. However, all of these approaches involve interpolation between quite different types or styles of motion, and do not provide a full solution to the more difficult problem of creating novel motions that have the subtle characteristics that make each person individual.

Furthermore, the perceptual responses of the viewer should be taken into account, for example: to what extent can variety be added to a motion without changing its meaning? What are the factors that determine whether a motion appears realistic or consistent for a character? What is the relationship between physical correctness of motion and our perception of that motion? When composing motions from sub-parts (from potentially different characters) what are the rules for deciding whether the composition is realistic or plausible [17]? Such questions pose interesting challenges for the future.

1.5 Behaviours

Ulicny and Thalmann introduced the idea of *levels of variety* [32], where they define the lowest level (LV0) as providing only a single solution for each action; the next level (LV1) allows a choice from a finite number of solutions; while the highest variety level (LV2) allows an infinite number of choices. They note the challenge of applying this concept to traditional AI approaches in crowd

Fig. 7. Varying conversational behaviours to enrich populated scenes

simulation, such as path planning, where the goal is to find an optimal path for an agent to get from one point to another. Without introducing variety, a crowd of virtual humans could end up queuing to get to the end-point, rather than behaving in a more natural way. They overcome this problem by specifying way-points for agents that are randomly distributed within a region close to the path nodes, thus introducing some variety in the paths followed. Another possible approach is to take individual preferences and knowledge of the environment into account when choosing a path for an agent [26]. Recently, Paris and Donikian presented a cognitive architecture to simulate behaviours for fully configurable autonomous agents, thus allowing variety in the decision-making process for high level goals as well as reactive behaviours [25]. Variety can also be added to crowd simulations by assigning individual characteristics and personality traits to agents (e.g., altruistic, dependent, rude, tolerant) [6, 11].

As before, investigating how humans perceive crowds can help us to determine how plausible or believable different behavioural simulations (both at a macroscopic and microscopic level) are. We have investigated some of the factors that affect the perception of pedestrian formations in different contexts [27, 12]. We have also examined how to add variety in conversational groups and the effects on the perception of realism when individuals' motion clips from several conversations are 'mixed and matched' to generate a variety of different conversational groups [20] (see Figure 7).

1.6 Sounds

Finally, for a crowd to be realistic they should be making appropriate crowd sounds. Recent approaches to audio rendering of complex scenes use perceptual metrics to decide how to cluster multiple sound sources and how to selectively allocate resources (e.g., to visible sources) [24, 31]. As with models and

motions, capturing, storing and rendering sounds presents a significant burden on any system. Representing, clustering and synthesizing new speaking sounds is acknowledged to be an especially difficult case. Statistical parametric speech synthesis, especially approaches based on Hidden Markov Models (HMM) have recently been used to create a variety of different voices [4]. It remains to be seen if such methods could be used to create a large variety of voices in a crowd, or if indeed this level of variety is important for speech, depending on the task. At what point does an individual voice become important? Are people more or less sensitive to variations in voices than in appearance or motions, and how do these factors interact? (So far, we have found in our studies of appearance and motion that the former dominates - will sound affect this?)

Finally, synchronising congruent sounds with character animations is particularly challenging. For example, at a microscopic level footsteps should be synchronised with the footplants of an animated character, and the sound should be congruent with their appearance - e.g., a heavy person with boots should sound different from a woman with high heels, or an appropriate conversational sound should emanate from a person talking as they pass the camera. Should speech also be fully synchronised with the facial animation and gestures of each up-close character? When are such details no longer necessary, as at a macroscopic level e.g., for large crowds viewed from a distance, they almost certainly would not be discernable.

1.7 Conclusions

In this paper, an overview has been presented of current research into creating variety for virtual humans, with a particular emphasis on crowd simulation applications. It is certainly not intended to be an exhaustive survey of the field, but rather to highlight some of the challenges that face researchers and practitioners. While advances have been made in each individual domain of modelling, rendering, animation, behaviour simulation and audio synthesis, further interesting research directions will involve combining and synchronising the variations created to deliver a coherent and rich multisensory percept.

Acknowledgements

The Higher Education Authority of Ireland and Science Foundation Ireland (project Metropolis) have funded the research in Trinity College Dublin on crowd variety. Thanks to all the researchers who have contributed to this work, in particular to Rachel McDonnell and Simon Dobbyn for their help in preparing this manuscript.

References

1. Allen, B., Curless, B., Popović, Z.: The space of human body shapes: reconstruction and parameterization from range scans. In: ACM SIGGRAPH 2003, pp. 587–594 (2003)

2. Allen, B., Curless, B., Popović, Z., Hertzmann, A.: Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 147–156 (2006)
3. Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: Scape: shape completion and animation of people. ACM Transactions on Graphics 24(3), 408–416 (2005)
4. Black, A.W., Zen, H., Tokuda, K.: Statistical parametric speech synthesis. In: Acoustics, Speech and Signal Processing, 2007, vol. 4, pp. IV–1229–IV–1232 (2007)
5. Brand, M., Hertzmann, A.: Style machines. In: ACM SIGGRAPH 2000, pp. 183–192 (2000)
6. Braun, A., Musse, S.R., de Oliveira, L.P.L., Bodmann, B.E.J.: Modeling individual behaviors in crowd simulation. In: Computer Animation and Social Agents (CASA 2003), p. 143. IEEE Computer Society, Los Alamitos (2003)
7. De Heras, C.P., Schertenleib, S., Mäim, J., Thalmann, D.: Real-time shader rendering for crowds in virtual heritage. In: Proceedings of the 6th international Symposium on Virtual Reality, Archeology and Cultural Heritage (VAST 2005), pp. 1–8 (2005)
8. De Heras, C.P., Schertenleib, S., Mäim, J., Thalmann, D.: Reviving the roman odeon of aphrodisias: Dynamic animation and variety control of crowds in virtual heritage. In: VSMM: Virtual Systems and Multimedia, pp. 601–610 (2005)
9. Dobbyn, S., Hamill, J., O'Conor, K., O'Sullivan, C.: Geopostors: a real-time geometry / impostor crowd rendering system. ACM Transactions on Graphics 24(3), 933 (2005)
10. Dobbyn, S., McDonnell, R., Kavan, L., Collins, S., O'Sullivan, C.: Clothing the masses: Real-time clothed crowds with variation. In: Eurographics Short Papers, pp. 103–106 (2006)
11. Durupinar, F., Allbeck, J., Pelechano, N., Badler, N.: Creating crowd variation with the ocean personality model. In: AAMAS 2008: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pp. 1217–1220 (2008)
12. Ennis, C., Peters, C., O'Sullivan, C.: Perceptual evaluation of position and orientation context rules for pedestrian formations. In: APGV 2008: Proceedings of the 5th symposium on Applied perception in graphics and visualization, pp. 75–82 (2008)
13. Galli, P.: Madagascar tech turns imagination into reality (2005), eWeek.com
14. Galvao, R., Laycock, R., Day, A.M.: Gpu techniques for creating visually diverse crowds in real-time. In: VRST 2008: Proceedings of the 2008 ACM symposium on Virtual reality software and technology, pp. 79–86 (2008)
15. Giese, M., Poggio, T.: Morphable models for the analysis and synthesis of complex motion patterns. International Journal of Computer Vision 38(1), 59–73 (2000)
16. Hasler, N., Stoll, C., Sunkel, M., Rosenhahn, B., Seidel, H.-P.: A statistical model of human pose and body shape. Computer Graphics Forum (Eurographics 2009) 28(2), 337–346 (2009)
17. Heck, R., Kovar, L., Gleicher, M.: Splicing upper-body actions with locomotion. Computer Graphics Forum (Eurographics 2006) 25(3), 459–466 (2006)
18. Hsu, E., Pulli, K., Popović, J.: Style translation for human motion. ACM Transactions on Graphics (SIGGRAPH 2005) 24(3), 1082–1089 (2005)
19. Jarudi, I.N., Sinha, P.: Relative contributions of internal and external features to face recognition. Technical Report CBCL Paper #225/AI Memo #2003-004, Massachusetts Institute of Technology (2003)

20. McDonnell, R., Ennis, C., Dobbyn, S., O'Sullivan, C.: Talking bodies: Sensitivity to de-synchronisation of conversations, vol. 6(4) (to appear, 2009)
21. McDonnell, R., Jörg, S., Hodgins, J.K., Newell, F., O'Sullivan, C.: Evaluating the effect of motion and body shape on the perceived sex of virtual characters. TAP 5(4), 1–14 (2009)
22. McDonnell, R., Larkin, M., Dobbyn, S., Collins, S., O'Sullivan, C.: Clone attack! perception of crowd variety. ACM Transactions on Graphics (SIGGRAPH 2008) 27(3) (2008)
23. McDonnell, R., Larkin, M., Hernández, B., Rudomín, I., O'Sullivan, C.: Eye-catching crowds: saliency based selective variation. ACM Transactions on Graphics (SIGGRAPH 2009) 28(3) (2009)
24. Moeck, T., Bonneel, N., Tsingos, N., Drettakis, G., Viaud-Delmon, I., Alloza, D.: Progressive perceptual audio rendering of complex scenes. In: I3D 2007: Proceedings of the 2007 symposium on Interactive 3D graphics and games, pp. 189–196 (2007)
25. Paris, S., Donikian, S.: Activity-driven populace: a cognitive approach for crowd simulation. IEEE Computer Graphics and Applications (Virtual Populace special issue) 29(4), 34–43 (2009)
26. Paris, S., Donikian, S., Bonvalet, N.: Environmental abstraction and path planning techniques for realistic crowd simulation. Computer Animation and Virtual Worlds 17, 325–335 (2006)
27. Peters, C., Ennis, C.: Modelling groups of plausible virtual pedestrians. IEEE Computer Graphics and Applications (Virtual Populace special issue) 29(4), 54–63 (2009)
28. Seo, H., Cordier, F., Magnenat-Thalmann, N.: Synthesizing animatable body models with parameterized shape modifications. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 120–125 (2003)
29. Sinha, P., Balas, B., Ostrovsky, Y., Russell, R.: Face recognition by humans: 19 results all computer vision researchers should know about. Proceedings of the IEEE 94(11), 1948–1962 (2006)
30. Tecchia, F., Loscos, C., Chrysanthou, Y.: Visualizing crowds in real-time. Computer Graphics Forum (Eurographics 2002) 21(4), 753–765 (2002)
31. Tsingos, N., Gallo, E., Drettakis, G.: Perceptual audio rendering of complex virtual environments. In: ACM SIGGRAPH 2004, pp. 249–258 (2004)
32. Ulicny, B., Thalmann, D.: Towards interactive real-time crowd behavior simulation. Computer Graphics Forum 21(4), 767–775 (2002)

Interactive Modeling, Simulation and Control of Large-Scale Crowds and Traffic

Ming C. Lin[1], Stephen Guy[1], Rahul Narain[1], Jason Sewall[1], Sachin Patil[1], Jatin Chhugani[2], Abhinav Golas[1], Jur van den Berg[1], Sean Curtis[1], David Wilkie[1], Paul Merrell[1], Changkyu Kim[2], Nadathur Satish[2], Pradeep Dubey[2], and Dinesh Manocha[1]

[1] Department of Computer Science
University of North Carolina
Chapel Hill, NC, U.S.A.
[2] Throughput Computing Lab
Intel Corporation
Santa Clara, CA, U.S.A.
{lin,sjguy,narain,sewall,sachin,golas,
berg,seanc,wilkie,pmerrell,dm}@cs.unc.edu
{jatin.chhugani,changkyu.kim,nadathur.rajagopalan.satish,
pradeep.dubey}@intel.com

Abstract. We survey some of our recent work on interactive modeling, simulation, and control of large-scale crowds and traffic for urban scenes. The driving applications of our work include real-time simulation for computer games, virtual environments, and avatar-based online 3D social networks. We also present some preliminary results and proof-of-concept demonstrations.

Keywords: Velocity Obstacle, Multi-Agent Simulation.

1 Introduction

Aggregates of numerous entities, such as a group of people and fleet of vehicles, form complex systems that exhibit interesting biological, social, cultural, and spatial patterns observed in nature and in society. Modeling of the collective behaviors remains an open research challenge in computer graphics, robotics, architecture, physics, psychology, social sciences, and civil and traffic engineering, as complex systems often exhibit distinct characteristics, such as emergent behaviors, self-organization, and pattern formation, due to multi-scale interactions among individuals and groups of individuals. Despite of decades of observation and studies, collective behaviors are particularly not well understood for groups with *non-uniform spatial distribution* and *heterogeneous behavior characteristics*, such as pedestrian and vehicle traffic in urban scenes, evacuation flows in complex structures, and coupled human-natural systems.

In this paper, we survey our recent works on addressing the problem of modeling, simulating, and directing the large-scale inidvidual agents in complex dynamic environments. Our works focus on (1) understanding the general principles

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 94–103, 2009.

that characterize the behavior of macroscopic dynamics for aggregate systems, and (2) identifying simplified, approximate models that capture their movement under varying conditions. This set of algorithms and techniques can potentially provide computational tools for motion planning, designing game plays, and developing simulation engines that are simple and easy to use for authoring games with numerous agents.

We present several complementary approaches for local collision avoidance and global navigation of multiple virtual entities in an interactive environment. These methods can be applied to interactive crowd simulation, motion synthesis, and coordination of multiple autonomous agents in computer games. These include (a) a new local collision avoidence algorithm between multiple agents [GCK+09]; (b) a novel formulation to model aggregate dynamics of dense crowds [NGCL09]; (c) an efficient simulation method of continuum traffic [SWML09]; and (d) an effective approach to direct and control virtual crowds using navigational fields [PvdBC+09]. Each of these methods are targetted at modeling, simulating, and directing the interaction between human crowds or vehicle traffic at various scales and densities. They are complementary techniques that can be integrated to capture aggregate dynamics of a complex urban scene.

2 ClearPath

We present a highly parallel and robust collision avoidance approach, *ClearPath*, for multi-agent simulation [GCK+09]. Our formulation extends and generalizes the concept of *velocity obstacles* (VO) [FS98] for local collision avoidance among dynamic obstacles. We use an efficient velocity-obstacle based formulation that can be combined with any underlying multi-agent simulation. We show that local collision avoidance computations can be reduced to solving a quadratic optimization problem that minimizes the change in underlying velocity of each agent subject to non-collision constraints.

We introduce a polynomial-time algorithm for agents to compute collision-free, 2D motion in a distributed manner. We pose the local collision avoidance problem for N agents as a combinatorial optimization problem. We extend the VO formulation by imposing additional constraints that can guarantee collision avoidance for each agent during the discrete interval. The Fast Velocity Obstacle (FVO) is defined using a total of four constraints. The *two* boundary cone constraints of the FVO are same as that of RVO [vdBLM08, vdBPS+08]:

$$\mathrm{FVO}_{L_B^A}(\mathbf{v}) = \phi(\mathbf{v}, (\mathbf{v}_A + \mathbf{v}_B)/2, \mathbf{p}^{\perp}_{ABleft}) \geq 0$$
$$\mathrm{FVO}_{R_B^A}(\mathbf{v}) = \phi(\mathbf{v}, (\mathbf{v}_A + \mathbf{v}_B)/2, \mathbf{p}^{\perp}_{ABright}) \geq 0$$

Additionally, we impose *two* more types of constraints:

Type-I Constraint - Finite time interval: We only guarantee collision avoidance for the duration ΔT. We compute a finite subset of the RVO cone that corresponds to the forbidden velocities that could lead to collisions in ΔT. The truncated cone. Due to efficiency reasons, we replace $\gamma_{AB}(\mathbf{v})$ with

aconservative linear approximation $\Gamma_{AB}(\mathbf{v})$. This additional constraint is represented as $\text{FVO}_{TB}^{A}(\mathbf{v}) = \Gamma_{AB}(\mathbf{v}) = \lambda\left(M - \widehat{\mathbf{p}_{AB}^{\perp}} \times \eta,\ \mathbf{p}_{AB}^{\perp}\right)$, where

$$\eta = \tan\left(\sin^{-1}\frac{\mathbf{r}_A + \mathbf{r}_B}{|\mathbf{p}_{AB}|}\right) \times (|\mathbf{p}_{AB}| - (\mathbf{r}_A + \mathbf{r}_B)), \text{and}$$
$$M = (|\mathbf{p}_{AB}| - (\mathbf{r}_A + \mathbf{r}_B)) \times \widehat{\mathbf{p}_{AB}} + \frac{\mathbf{v}_A + \mathbf{v}_B}{2}$$

Type-II Constraint - Consistent velocity orientation: Without loss of generality, we force each agent to choose its *right* side and impose this constraint as $\text{FVO}_{CB}^{A}(\mathbf{v}) = (\phi(\mathbf{v}, \mathbf{p}_A, \mathbf{p}_{AB}^{\perp}) \leq 0)$. This is also a conservative formulation to guarantee collision-free motion.

Any feasible solution to all of constraints, which are separately formulated for each agent, will guarantee collision avoidance. We solve this problem as a quadratic optimization function with non-convex linear constraints for each agent. It can be shown to be NP-Hard [GCK+09] for non-constant dimensions via reduction to quadratic integer programming. It has a polynomial time solution when the dimensionality of the constrains is constant – two in our case. In practice, ClearPath is more than one order of magnitude faster than prior velocity-obstacle based methods.

P-ClearPath: Parallel Collision Avoidance: We further show that ClearPath is amenable to data-parallelism and thread-level parallelism on commodity processors and suggest a parallel extension. The resulting parallel extension, P-ClearPath, exploits the structure of our optimization algorithm and architectural capabilities such as gather/scatter and pack/unpack to provide improved data-parallel scalability. ClearPath operates on a per-agent basis in a distributed manner, finding each agent's nearest neighbors and computes a collison-free velocity w.r.t. those neighbors. There are *two* fundamental ways of exploring Data-Level parallelism (henceforth referred to as DLP).

Intra-Agent: Consider Fig. 1(a). For each agent, we explore DLP within the ClearPath computation. Since the agents operate in 2D, they can perform their X and Y coordinate updates in a SIMD fashion. This approach does not scale to wider SIMD widths.

Inter-Agent: Operate on multiple agents at a time, with each agent occupying a slot in the SIMD computation. This approach is scalable to larger SIMD widths, but needs to handle the following two issues:

1. Non-contiguous data access: In order to operate on multiple agents, ClearPath requires *gathering* their obstacle data structure into a contiguous location in memory. After computing the collision-free velocity, the results need to be *scattered* back to their respective non-contiguous locations. Such data accesses become a performance bottleneck without efficient *gather/scatter* operations.

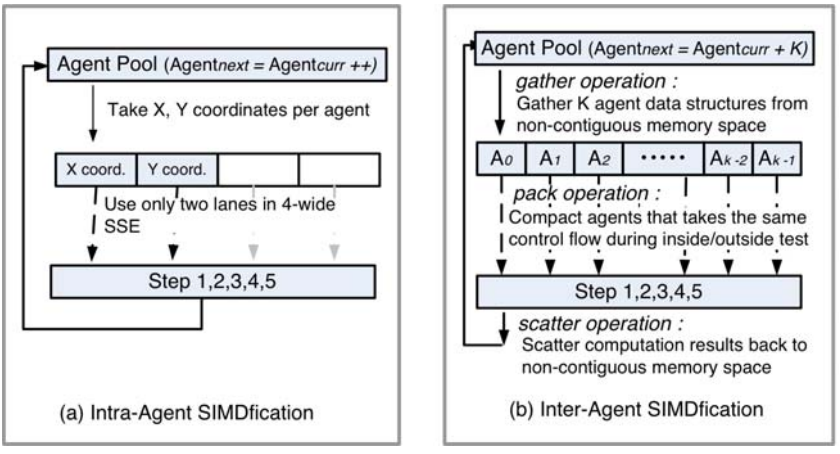


Fig. 1. Data-Parallel Computations: (a) Intra-Agent SIMDfication for SSE; (b) Inter-Agent SIMDfication for wide SIMD

2. Incoherent branching: Multiple agents within a SIMD register may take divergent paths. This degrades SIMD performance, and is a big performance limiter during intersection computations and inside/outside tests. One or more of the agents may terminate early, while the remaining ones may still be performing comparisons.

Current SSE architectures on commodity CPUs do not have efficient instructions to resolve the above two problems. Hence, we used the intra-object SIMDfication approach and obtain only moderate speedups. P-ClearPath adopts the Inter-agent approach, and performs computation on $\mathcal{K}$ agents together. Fig. 1(b) shows a detailed mapping of the various steps in ClearPath algorithm. For collision-free velocity computation, each agent A_i is given as input its neighboring velocity obstacles (truncated cones) and the desired velocity. For more detail, see [GCK+09].

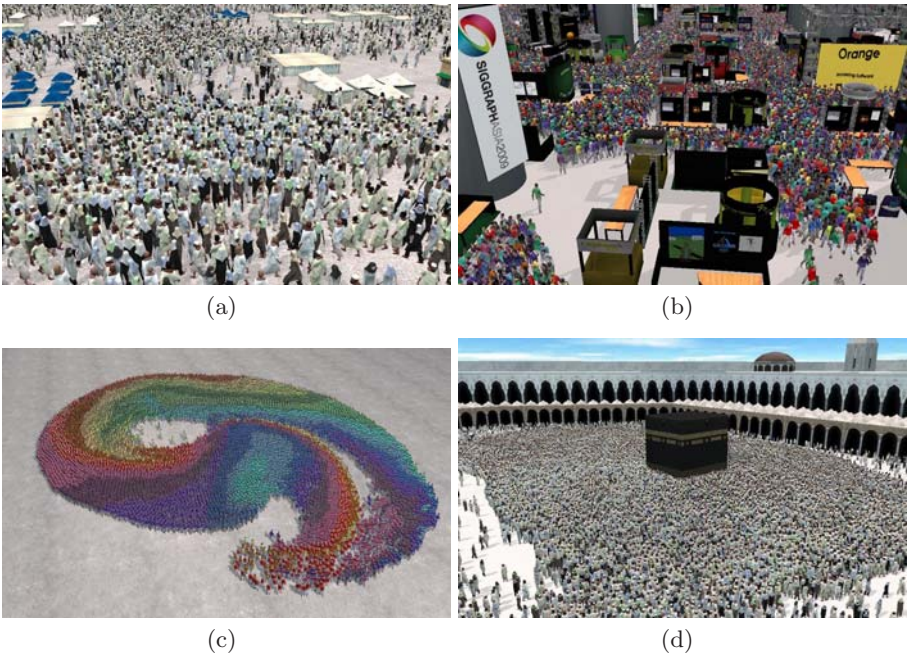
Results: We evaluate its performance in various scenarios on different platforms like current multi-core CPUs and the upcoming many-core processor code-named Larrabee. In practice, P-ClearPath demonstrates 8-15X speedup on a conventional quad-core processor over prior VO-based algorithms on similar platforms. When executed on a Larrabee simulator with 32-64 cores, P-ClearPath achieves additional speedup of up to 15X, resulting in up to 100-200X speedup over prior VO-based approaches.

Overall, for simple game-like scenarios with a few hundred agents, P-ClearPath takes about 2.5 milliseconds on a single Larrabee core, while a complex simulation with few hundreds of thousands of heterogeneous agents takes only 35 milliseconds on the simulated 64-core Larrabee processor. To the best of our knowledge, P-ClearPath is the first scalable collision avoidance algorithm for multi-agent simulations with a few hundred thousand agents at interactive rates.

3 Hybrid Crowds

Dense crowds exhibit a low interpersonal distance and a corresponding loss of individual freedom of motion. This observation suggests that the behavior of such crowds may be modeled efficiently on a coarser level, treating its motion as the flow of a single aggregate system. Based on such an abstraction, we develop a novel inter-agent avoidance model which decouples the computational cost of local planning from the number of agents, allowing very large-scale crowds consisting of hundreds of thousands of agents to be simulated scalably at interactive rates.

Our method combines a Lagrangian representation of individuals with a coarser Eulerian crowd model, thus capturing both the discrete motion of each agent and the macroscopic flow of the crowd. In dense crowds, the finite spatial extent occupied by humans becomes a significant factor. This effect introduces new challenges, as the flow varies from freely compressible when the density is low to incompressible when the agents are close together. This characteristic is shared by many other dynamical systems consisting of numerous objects of finite size, including granular materials, hair, and dense traffic. We propose a new mathematical formulation to model the dynamics of such aggregate systems in a principled way, which is detailed in [NGCL09].



(a) (b)

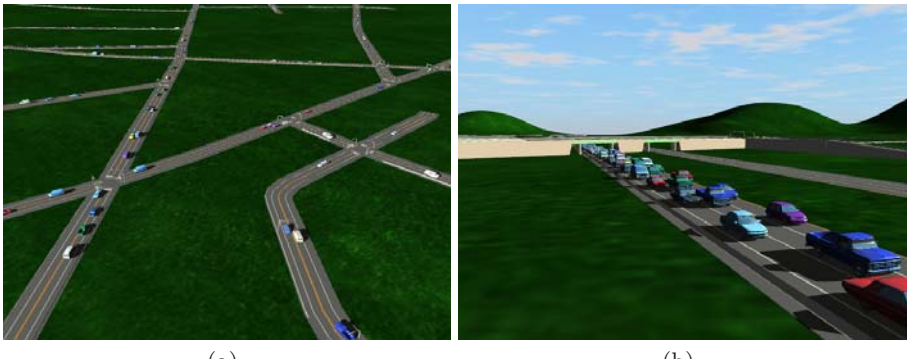
(c) (d)

Fig. 2. Some examples of large, dense crowds simulated with our technique. (a) 100,000 pilgrims moving through a campsite. (b) 80,000 people on a trade show floor. (c) 10,000 agents attempt to cross over to the opposite points of a circle and meet in the middle, before forming a vortex to resolve the deadlock. (d) 25,000 pilgrims with heterogeneous goals in a mosque.

Results: Fig. 2 shows some of our results for dense crowds of up to 100,000 agents closely packed in complex scenes simulated at interactive rates with our techniques. We measured the performance of our algorithm on an Intel Core i7-965 machine at 3.2 GHz with 6 GB of RAM. Even with very large numbers of agents, we can achieve close to interactive performance. Our method supports general scenarios with independent, heterogeneous agents; the number of unique goals has no effect on the performance.

4 Continuum Traffic

There exists a vast amount of literature on modeling and simulation of traffic flows, with existing traffic simulation techniques generally focusing on either microscopic [CSS00] or macroscopic behaviors ([AR00, Zha02]). However, little attention has been paid to the possibility of extending macroscopic models to produce detailed 3D animations and visualization of traffic flows.



(a) (b)

Fig. 3. Images from our traffic simulator: (a) A bird's-eye view of a busy intersection; (b) a traffic jam occurring on a highway

In this section, we present an overview of a method for efficient simulations of large-scale, real-world networks of traffic using macroscopic (continuum-level) dynamics that uses microscopic, individual vehicle information to display each vehicle [SWML09]. We model the movement of many vehicles with a single computational cell — while individual vehicle information facilitates visual representation and allows for per-vehicle information be incorporated into the large-scale simulation.

Our technique produces detailed, interactive simulation of traffic flows on a wide variety of road types, including urban streets, multi-lane highways, and winding rural roads. Our approach extends a continuum, per-lane flow model by introducing a novel model of lane changes that uses our discrete vehicle representation. We introduce new techniques for describing the behavior of vehicles at intersections. This simulation technique is able to effectively utilize the processing power of many-core shared memory architectures for scalable simulation.

We also validate the results of our traffic simulations with real-world observed traffic flows.

Results: We have tested our technique on a number of synthetic road networks, a digital representation of a modest-sized city, and on a high-volume traffic on a short segment of freeway. Our city data set features sparse rural roads, short dense arrangements punctuated with stoplights, and a highway running through it. This data set was automatically generated from publicly-available GIS data from the US Census Bureau's TIGER database, which is widely available and has excellent coverage of the roads in the United States. Fig. 3 shows vehicles from our simulation of the six-lane stretch of I-80 freeway in Emeryville, California and the results of our simulation.

Our approach leverages an inexpensive continuum solve with a lightweight, particle-like visual representation of each vehicle to produce realistic motion synthesis of vehicle motion on large networks. Rather than be input-sensitive to the number of vehicles in the network, the compute expense of our technique is directly related to the number of computational cells that must be update each step. The small city data data set shown in Fig. 3 covers an area over 23 square kilometers and contains over 180 km with of lanes. The scenarios in the accompanying video have as many as 2500 vehicles in the network at a given time, and these simulations can be peformed on a single core of an Intel Core2 process at frame rates in excess of 120 video frames/second.

5 Directing Crwods Using Navigation Fields

Most existing agent-based systems assume that each agent is an independent decision making entity. Some of the methods also focus on group-level behaviors and complex rules for decision making. The problem with these approaches is that interactions of an agent with other agents or with the environment are often performed at a local level and can sometimes result in undesirable macroscopic behaviors. Due to the complex inter-agent interactions and multi-agent collision avoidance, it is often difficult to generatedesired crowd movements or motion patterns that follow the local rules.

In this work, we address the problem of directing the flow of agents in a simulation and interactively control a simulation at runtime [PvdBC+09]. Our approach is mainly designed for goal-directed multi-agent systems, where each agent has knowledge of theenvironment and a desired goal position at each step of the simulation. The goal position for each agent can be computed from a higher-level objective and can also dynamically change during the simulation.

Our approach uses discretized *guidance fields* to direct the agents. Based on these inputs, we compute a unified, goal-directed, smooth *navigation field* that avoids collisions with the obstacles in the environment. The guidance fields can be edited by a user to interactively control the trajectories of the agents in an ongoing simulation, while guaranteeing that their individual objectives are attained. The microscopic behaviors, such as local collision avoidance, personal

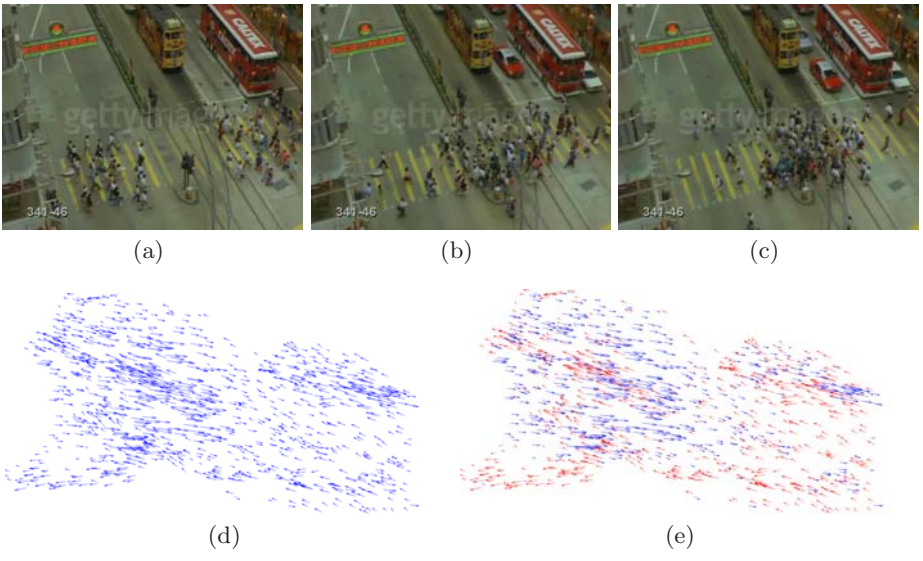


Fig. 4. Motion patterns detected in a video of a crosswalk in Hong Kong. (a,b,c) Frames from the original video; (d) Motion flow field; (e) Detected motion patterns.

space and communication between individual agents, are governed by the underlying agent-based simulation algorithm.

Results: This approach is general and applicable to a variety of existing agent-based methods. We illustrate the usefulness of our approach in the context of several simulation scenarios shown in Fig. 5. The user edits the simulation by specifying guidance fields, that are either drawn by the user or extracted from a video sequence (Fig. 4). The overall approach can be useful from both artistic and data-driven perspectives, as it allows the user to interactively model some macroscopic phenomena and group dynamics.

6 Discussion

In this paper, we have presented a brief survey of algorithms for real-time collision avoidance, modeling, simulation, and control of multiple virtual agents, including both humans and vehicles, in dynamic scenes. We also demonstrated their applications on interactive crowd simulation for games and virtual environments.

Among these methods, ClearPath is designed to take advantages of upcoming many-core architectures to achieve real-time collision avoidance of hundreds of thousands of discrete agents at interactive rates. In contrast, our hybrid representation for large-scale crowds are better suited for highly dense scenarios, capable of simulating upto millions of indvidiual agents in nearly real time by exploiting their constrained movement. But, its implementation on many-core architecture is yet to be explored.

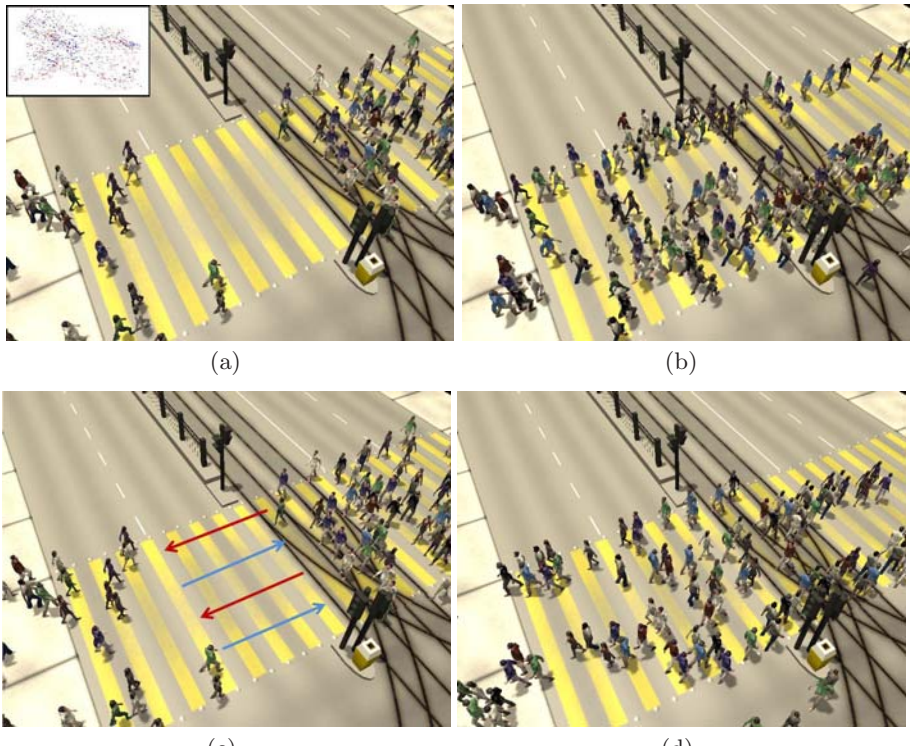


Fig. 5. Crosswalk Simulation: (a) Video-based motion patterns from Figure 4(e) used as guidance fields; (b) Agent motion generated by the navigation fields. (c) Sketch-based guidance fields to specify lane formation in the simulation; (d) Lane formation generated by goal-directed navigation fields.

Our simulation technique for modeling continuum traffic is a high-level, macroscopic method for simulating a wide variety of traffic condition. We would like to further develop the idea of a coupled continuum-discrete traffic simulation to take advantage of the unique advantages each has to offer; continuum models are fast and can handle large areas inexpensively, while discrete models are capable of describing more individualistic behavior. Another natural extension would be an integration of traffic and crowd simulations to model a urban scene in all aspects.

Our current approach to direct and control crowds using navigation fields is general and versatile. It can use any low-level collision avoidance algorithm. The system allows the user to specify the navigation fields by either sketching paths directly in the scene via an intuitive authoring interface or by importing motion flow fields extracted automatically from crowd video footage. This technique is complementary to other methods and most suitable for interactive editing and testing during the conceptual game design.

Acknowledgments. The research presented here is supported in part by Army Research Office, Intel, National Science Foundation, and DARPA/RDECOM.

References

[AR00] Aw, A., Rascle, M.: Resurrection of second order models of traffic flow. SIAM Journal of Applied Mathmatics (60), 916–938 (2000)

[CSS00] Chowdhury, D., Santen, L., Schadschneider, A.: Statistical Physics of Vehicular Traffic and Some Related Systems. Physics Reports 329, 199 (2000)

[FS98] Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. International Journal on Robotics Research 17(7), 760–772 (1998)

[GCK+09] Guy, S., Chhugani, J., Kim, C., Satish, N., Lin, M.C., Manocha, D., Dubey, P.: Clearpath: Highly parallel collision avoidance for multi-agent simulation. In: Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2009)

[NGCL09] Narain, R., Golas, A., Curtis, S., Lin, M.C.: Aggregate dynamics for dense crowd simulation. ACM Transactions on Graphics (Proc. of ACM SIGGRAPH Asia) (2009)

[PvdBC+09] Patil, S., van den Berg, J., Curtis, S., Lin, M.C., Manocha, D.: Directing crowd simulations using navigation fields. Technical report, Department of Computer Science, University of North Carolina (May 2009)

[SWML09] Sewall, J., Welkie, D., Merrell, P., Lin, M.C.: Continuum traffic simulation. Technical report, Department of Computer Science, University of North Carolina (May 2009)

[vdBLM08] van den Berg, J., Lin, M.C., Manocha, D.: Reciprocal velocity obstacles for realtime multi-agent navigation. In: Proc. of IEEE Conference on Robotics and Automation, pp. 1928–1935 (2008)

[vdBPS+08] van den Berg, J., Patil, S., Seawall, J., Manocha, D., Lin, M.C.: Interactive navigation of individual agents in crowded environments. Proc. of ACM Symposium on Interactive 3D Graphics and Games, 139–147 (2008)

[Zha02] Zhang, H.M.: A non-equilibrium traffic model deviod of gas-like behavior. Transportation Research B (36), 275–290 (2002)

A Velocity-Curvature Space Approach for Walking Motions Analysis

Anne-Hélène Olivier[1], Richard Kulpa[1,2], Julien Pettré[2], and Armel Crétual[1]

[1] M2S, University of Rennes 2, France
[2] Bunraku, IRISA / INRIA-Rennes, France
{anne-helene.olivier,richard.kulpa,armel.cretual}@uhb.fr,
julien.pettre@irisa.fr

Abstract. This paper presents an automatic turns detection and annotation technique which works from unlabeled captured locomotion. Motion annotation is required by several motion capture editing techniques. Detection of turns is made difficult because of the oscillatory nature of the human locomotion. Our contribution is to address this problem by analyzing the trajectory of the center of mass of the human body into a velocity-curvature space representation. Our approach is based on experimental observations of carefully captured human motions. We demonstrate the efficiency and the accuracy of our approach.

Keywords: Motion analysis, turn detection, velocity-curvature space, motion segmentation.

1 Introduction

Locomotion is probably the most studied behavior in the field of virtual humans animation. A majority of interactive locomotion synthesis techniques are based on motion capture databases. Motion-capture based techniques are efficient, produce believable motions and are relatively easy to use in practice. Several approaches exist. Motion concatenation techniques assemble portions of captured motions in a sequential manner in order to generate new motions [1,2,3,4,5]. Motion blending techniques simultaneously consider several motion sources in order to generate animations, whose features are not existing in the initial motion database [6,7,8,9,10]. Learning techniques can also be used to build locomotion controllers on top of motion database [11]. Preprocessing and annotating motion data is a required stage for all of these approaches.

In this paper, we focus on the problem of detecting and annotating turning steps and straight steps into a captured walking motion. This problem is difficult because the human walk is oscillatory by nature. During straight steps, none of the human body limb is moving linearly. Instantaneous walking direction is constantly changing. On the contrary, during turns, the trajectory of some limbs such as the pelvis may appear more linear than during straight steps. Most of previous techniques used to detect and annotate turns first compute the walking direction. Walking direction is then filtered to remove oscillations. Direction

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 104–115, 2009.

changes that overpass manually tuned thresholds enable the detection of turns. This type of methods fails to correctly detect slight turns or provides inaccurate localization of turns in time.

Our main contribution is a velocity-curvature space based approach to analyze walking motions. First, from a set of carefully recorded walking motions, we demonstrate that the center of mass (CoM) trajectory respects an invariant law linking mean curvature and mean velocity at each step during straight walking. On the contrary, we demonstrate that turning steps violate this law. Second, we apply this law to automatically detect and annotate turns and straight steps into any motion-captured walking motions. Our technique is also capable of distinguishing three different strategies that humans use to perform turns: spin turns, step turns and complex turns. We demonstrate our method is accurate and able to successfully detect slight turns. This method is then interesting for video games to program realistic locomotion controller.

2 Related Work

Our goal is to annotate each step of a captured walking motion. Our problem is thus a motion analysis one which falls in the topic of both Biomechanics and Computer Animation. In this section, we first provide a short overview of the Biomechanics state of the art about turning motions before describing how turns are detected in motion captures according to Computer Animation practices.

In the Biomechanics field, Glaister et al. [12] estimate that the overall proportion of turning steps during locomotion is about 35% to 45%. Based on video sequences, they annotate and categorize motions by hand. The method is suitable to study large space configurations, nonetheless, it is neither automatic nor accurate. Another possibility to evaluate curved walking is to use external equipments such as a force plate: Taylor et al. analyze 90° turns using such an equipment in [13]. Although a force plate is interesting in order to study gait parameters, it is not designed to delimit a turn: only part of the motion can be considered as it matches a specific location. A geometric method to detect turns in captured locomotion is described in [14]. A 90° turn was bounded by computing a quadrant thanks to the two axes of the straight lines surrounding the turn. This method does not allow to detect whether there is a turn or not. It only allows to determine which steps of the path are actually turning ones using the *a priori* knowledge that there is a turn. More generally, Biomechanics studies does not highlight one single definition of a turn. Indeed, nearly each study dealing with walking in curved path uses its own method. In conclusion, our work first aims at proposing a simple and objective definition of a turn during natural walking (i.e. without additional external constraints). In practice, we use this definition to distinguish turns from straight steps into unlabeled walking sequences.

In the Computer Animation field, segmenting a locomotion into steps can be achieved by analyzing foot support patterns [15]. These information can come from a motion annotation stage that provides a description of each step: turning or straight. The motion annotation based on examples [3,16] needs a user

intervention with a manual cut-out on a high number of data in order to create a set of reference motions describing the performed actions. The quality of the resulting annotation is dependent on the quality of this initial manual annotation stage. Another approach, and probably the most used in practice, analyses the direction changes of the walking trajectory. The walking trajectory is often estimated from a combination of several markers' trajectory. The CoM of the human body is a typical example. Kwon and Shin [10] propose an automatic step analysis technique based on the CoM trajectory (later re-used in [17]). They first distinguish walking steps from running steps as well as transitions between these two types of motion by analyzing the changes of altitude of the CoM (in parallel with foot support phases). Then, authors estimate the average acceleration, velocity and turning angle of the CoM trajectory over each step. The turning angle is used to evaluate the presence of turns. Unfortunately, as explained in Section 3.2, turning angle has a non-null value during straight walking due to the oscillations of the CoM trajectory around the followed path. Thus, the turning angle is a relatively bad criterion to detect turns as discussed in Section 4. On one hand, a low-pass filter can eliminate the presence of oscillations around the followed path, but, on the other hand, filtering makes hardly detectable the presence of slight turns and especially their delimitation in time. Shiratori et al. automatically annotate dancing motions in [18]. Their method is also based on the CoM trajectory analysis. This technique is not applied to walking motions, nevertheless, they use a set of thresholds to distinguish different phases of the considered motions. Such thresholds are also used to distinguish the different phases of a locomotion in other works: when the turning angle overcomes a manually tuned threshold, a turn is detected. Unfortunately, the amplitude of natural turning angle oscillations during straight steps is dependent on the walking velocity. The use of a single threshold to enable turn detection cannot provide satisfying results. In comparison, our turn detection is independent of the walking velocity because it observes the co-variation of the walking velocity with the curvature of the CoM trajectory. We are then able to automatically provide a trajectory segmentation for any walking motion. Park et al. fit each locomotion cycle (two steps) with arc of circles in order to estimate walking parameters in [7,8]. In practice, it is noticeable that a relatively large radius is found when trying to fit an arc of circle to a straight walking cycle. Indeed, asymmetries may exist between the two steps composing the cycle because one of the step is a turning one or because the tangential velocity is changing along the cycle. Subtle variations in one of the two considered steps result in large variations for the fitted arc of circle. In comparison, our method to detect turns is robust to tangential variation changes during walking.

3 Modeling of Walking Trajectories

3.1 Experimental Motion Dataset

Our study is based on a set of experimental data. 15 subjects performed straight walking trajectories and 10°, 20°, 30°, 45°, 60°, 90°, 120°, 135° and 150° turns

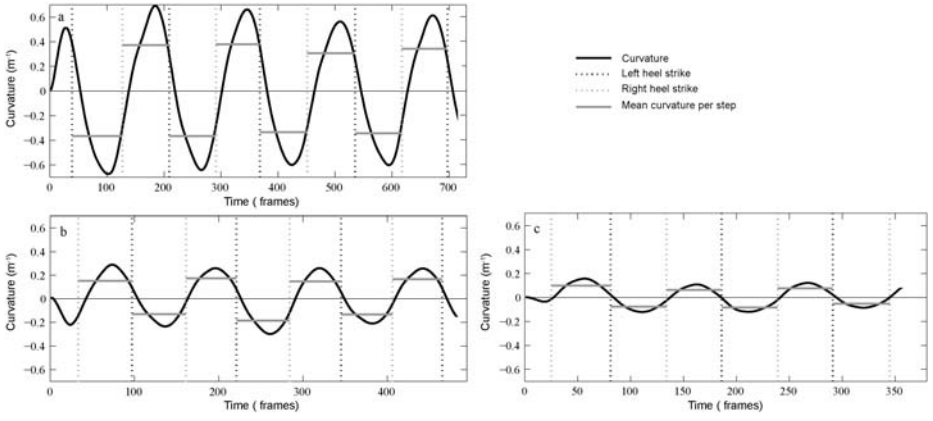


Fig. 1. Instantaneous curvature and mean curvature per step of the CoM trajectory during walking at slow (a), comfort (b) and fast (c) velocity

respectively at natural, slow and fast walking speed. Each trajectory is performed four times, as a result, we recorded 120 motions per subject. 3D kinematic data were recorded thanks to 12 high resolution Vicon MX cameras (Oxford Metrics, Oxford, UK) at a sampling rate of 120Hz. Thirty four reflective markers were attached to the subject skin on standardized landmarks in order to compute the CoM trajectory. Reconstruction and labeling were performed using Vicon IQ software (Oxford Metrics, Oxford, UK) and following computations using Matlab (Mathworks, Natick, MA). After a raw data filtering based on a cubic smoothing spline (csaps matlab function, p=0.99 with 1 representing no filtering), we extracted the CoM position at each frame. The detection of heel strikes was performed using the methodology described in [19] in order to segment the trajectory step-by-step. Then, we computed mean curvature (C) and mean velocity (V) for each step.

3.2 Segmented CoM Trajectory: A Step-by-Step Analysis

Walking trajectory is a succession of straight and curved paths that occur more or less often depending on the performed activities [12]. Straight walking is not actually a straight path but rather a succession of slight curves that cancel each other step-by-step. Therefore, CoM trajectory rolls from side to side around the mean direction of progression and is synchronized with the stepping activity: after a left stance (resp. right), CoM goes to the right (resp. left). Figure 1 plots both the instantaneous and the mean curvature of each step of the CoM trajectory during straight walking at different speeds (slow, comfort and fast walking). Oscillations of the CoM result from the external forces applied to each foot during support phases: therefore considering this stepping activity is fundamental when analyzing the walking motion.

From this point of view, two stepping strategies have been experimentally described for human locomotion according to the stance foot used for turning:

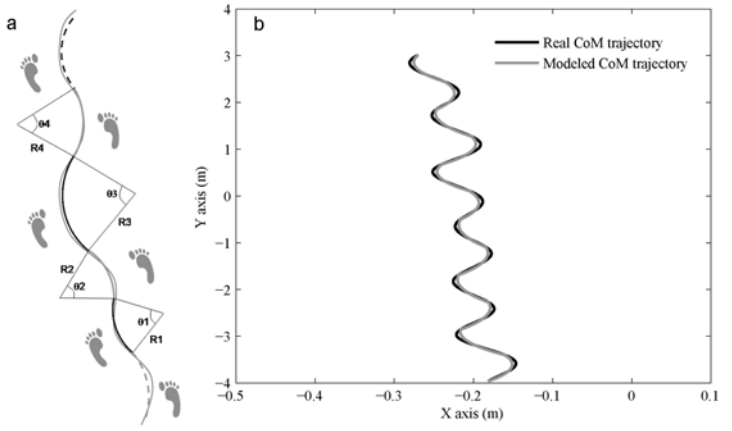


Fig. 2. a) Step-by-step trajectory model of the CoM trajectory based on a continuous composition of arcs of circles. Each arc is geometrically defined by a curvature C ($C = \frac{1}{R}$) and one angle θ. b) Superimposition of a recorded CoM trajectory with the resulting geometrically modeled trajectory. Our model accurately follows real data.

the spin turn and the step turn strategies [20,13]. During a spin turn, the turn is performed on the inner stance foot whereas during a step turn, the turn is performed on the outside one. In other words, compared with the standard behavior of the CoM in straight walking, a spin turn is a lack of curvature or an opposite curvature on the inner stance foot whereas a step turn is an increase of curvature on the outside one. This definition assumes that the turn is achieved in one step or at least that there is one preponderant step. However, studies based on video analysis showed that turns can be performed using a more complex strategy using a multi-step mechanism [12].

In conclusion, because the stepping activity determines curvature changes of the walking trajectory, our approach considers walking in a discrete way step-by-step in order to detect the turns.

3.3 Geometric Modeling of the CoM Path

To model the CoM trajectory step-by-step, several approaches can be used with several levels of complexity (arc of a circle, spline, complex polynomial). The more complex the model is, the more difficult the computation is in order to fit the original trajectory. We proposed to model walking trajectory with a simple method using a succession of arcs of a circle per step. Each arc is characterized by a couple of geometric parameters as shown in Figure 2a: a curvature C and one angle θ. Curvature is the inverse of the radius of curvature (cf. equation 1). We have chosen curvature instead of radius of curvature because the latter takes infinite values at each inflection points. Such a case occurs at each step in straight walking when the CoM rolls from side to side according to the stepping activity. θ is computed as the angle formed by the two radials delimiting the arc of circle

modeling one single step. C is computed as the mean curvature of the trajectory for the corresponding step. Instantaneous curvature C_i (m^{-1}) is computed as follow:

$$C_i = \frac{1}{R_i} = \frac{det(V_i, A_i)}{V_i^3} \tag{1}$$

with R_i the instantaneous radius of curvature (m), V_i the instantaneous horizontal velocity ($\mathrm{m.s}^{-1}$) and A_i the instantaneous horizontal acceleration ($\mathrm{m.s}^{-2}$). An example of CoM trajectory superimposed with our step-by-step trajectory model is shown in Figure 2b. The scale of the X-axis was voluntary modified in order to highlight the accuracy of our model: only slight differences appear between the reference and the model trajectories. Our approach is relevant insofar as the root mean square error (RMS), computed over all our experimental dataset, between actual CoM trajectory and the step-by-step trajectory model can be neglected (cf. Table 1). Then, this step-by-step model with arcs of a circle gives us a simple geometric description of the path followed by the CoM. In order to fully model the CoM trajectory, we now have to account for walking velocity information.

Table 1. RMS error between CoM trajectory and the step-by-step trajectory model with arc of circles computed on all straight walking trials for all subjects

	Slow	**Comfort**	**Fast**
RMS (mm)	7 ±1.5	4.1 ±0.9	3.1 ±1.1

3.4 Invariant Law Linking Mean Curvature and Velocity during Straight Walking

In the previous section, we introduced a geometric model of the CoM path during walking. In order to take temporal parameters into account, mean velocity V ($m.s^{-1}$) of the subject at each step is computed. Figure 1 illustrates that velocity and curvature are intimately linked: the higher the speed, the lower the curvature of the CoM trajectory. For all available straight walks among our experimental dataset, Table 2 shows the measured mean curvature C and mean velocity V of each step computed on all straight walking trials for all subjects. Table 2 also

Table 2. Mean curvature (C), mean velocity (V) and intra-individual coefficients of variation Cv_C of C and Cv_V of V computed on all straight walking trials for all subjects according to walking velocity

	Slow	**Comfort**	**Fast**
C (m^{-1})	0.38 ±0.15	0.18 ±0.05	0.09 ±0.10
V ($\mathrm{m.s}^{-1}$)	0.85 ±0.15	1.27 ±0.18	1.81 ±0.18
Cv_C(%)	9.06±3.42	3.19±1.58	1.41±0.44
Cv_V(%)	3.76±0.72	2.78±0.65	3.06±0.74

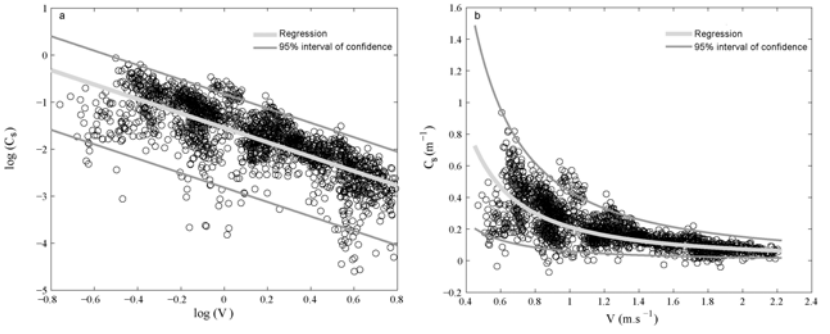


Fig. 3. a) Log-log space relation between all couples $[V, C_s]$ of the modeled CoM trajectory for each step in straight walking. b) Relation between all couples $[V, C_s]$ of the modeled CoM trajectory for each step in straight walking.

provides the coefficients of variation of these parameters between steps. Mean intra-individual coefficients of variation are computed as the mean percentage of the standard deviation related to the mean of each parameter value across each subject. Variations remain very small as expected for straight walking. We conclude that the CoM pattern resulting from our model over each step is highly repetitive for any given subject.

In straight walking, curvature is on the right after a left stance and on the left after a right stance. To describe this behavior, a new variable C_s, the signed curvature, is introduced. Its value is the absolute value of C. Its sign is positive when it verifies this behavior and negative otherwise. All our approach is based on the set of all the couples $[V, C_s]$ for all the trials and for all the subjects. The relation computed between all of them describes then the behavior of straight walking. It is characterized by a linear relation in the log-log space (R^2=0.56, p<0.001; with $log(C_s) = -1.53log(V) - log(1.55)$), illustrated in Figure 3a. It can be expressed in the velocity-curvature space, allowing us to determine a 95% interval specific of the relation between V and C_s in straight walking (cf. Figure 3b).

3.5 Turn Detection in the Velocity-Curvature Space

Results show that a relation exists between C_s and V for each step of straight walking trajectories. This relation allows to identify a set ϵ of couples $[V, C_s]$, illustrated by the gray shape in Figure 4. Then, for a given trial consisting of turning and straight steps, a turning step can be detected if one or several couples $[V, C_s]$ associated with a step are outside the set ϵ.

With this method, turning strategies previously described in the literature can be expressed in the velocity-curvature space according to the set ϵ of couples $[V, C_s]$ corresponding to straight walking. Then, a step turn can be considered as an increase of curvature: it is detected if one step is above the set ϵ in gray (cf. Figure 4b). Conversely, a spin turn is defined as a lack of curvature or a curvature

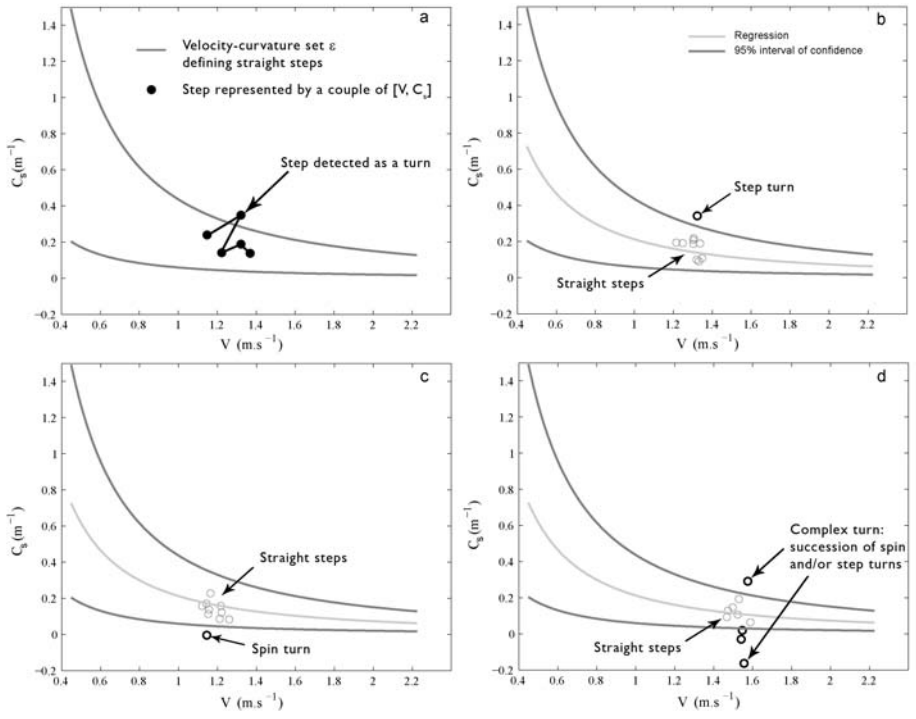


Fig. 4. a) Methodology to detect a turn within a given trajectory. The gray set ϵ is the specific behavior between mean velocity (V) and mean signed curvature (C_s) computed for each step during straight walking. Black dots are the couples $[V, C_s]$ for each step of a given trajectory. If one or several couples are outside the set ϵ, corresponding steps are identified as turns. All the points within the gray bounds represent straight steps. A point above is a step turn (b) and a point under is a spin turn (c). Complex turns are successions of spin and step turns (d).

with negative sign below the set ϵ (cf. Figure 4c). A complex strategy is finally detected when several steps are successively outside the set ϵ (cf. Figure 4d).

As we know the performed action in our experimental dataset, we are able to evaluate our method accuracy. The percentage of false negative trials (i.e. all couples $[V, C_s]$ are within the set ϵ but the trial is actually a turn) is small: Table 3 gives the proportion of correct detection with respect to the effective turning angle. Results show that 100% of turns are detected for a 30° turn and above. Nevertheless, some errors can be made in the detection for 10° (31.1%) and 20° (3,3%) turns. However, these turns are very far from those of daily living. Indeed, Sedgeman et al. notice in [21] that most turns are performed with angles between 76° and 120° during unconstrained walking. Therefore, this disadvantage does not limit excessively its use on large database. Our method is thus relevant to study natural walking motion. Finally note that the proposed method is also able to identify the stepping strategy while turning (cf. Figure 5). This additional

Table 3. Percentage of correct turn detection for the modeled CoM trajectory according to the angle of the turn

	10°	**20°**	**30°**	**45°-150°**
Valid detection (%)	68.9	96.7	100	100

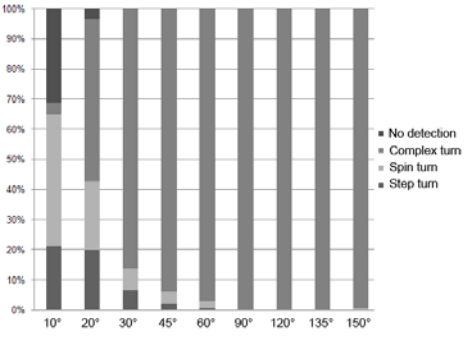


Fig. 5. Turning strategy quantification according the angle of the turn. Four cases are quantified: no detection (false-negative result), step turn, spin turn and complex turn. The latter represents a multi-step strategy.

information is relevant to distinguish motions having different intrinsic properties and to avoid matching them when blending motions for example. Results finally showed that the multi-step strategy is preferentially used by subjects in order to perform a turn. In order to create walking model for video games, it could be important to have information on the foot stance initiating the multi-step turn. In that case, there is no preferential stance foot to initiate the turn when considering all speed conditions. Nevertheless, a slight difference can be observed depending on the walking speed: subjects tend to initiate the turn preferentially with the outer stance foot for the slow condition and with the inner stance foot for the fast one.

4 Application and Discussion

We have proposed a method that is able to detect turns in any motion-captured walking. This method is based on a set of experimental trajectories made up of one single turn each. However, in everyday life, navigation is mainly a succession of straight and curved movements. Since our method is based on curvature step-by-step, it provides all the information about the nature of the motion (straight line or turn). It is then generic and can therefore be applied on all navigation movements including those with successions of turns. In such a case, the sign of mean curvature (C) indicates the turning direction. This step-by-step approach provides moreover smoother data since it is based on mean values of velocity and curvature.

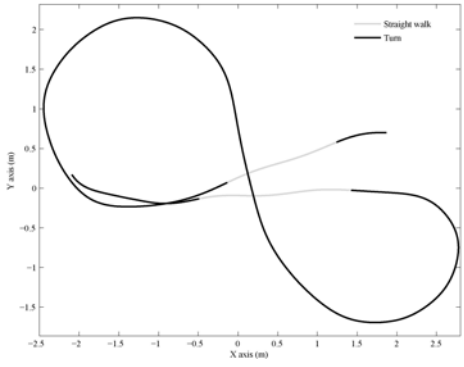


Fig. 6. Automatic segmentation of the CoM trajectory based on the velocity-curvature space. Black and gray parts are respectively turns and straight walks.

Figure 6 represents the detection of turns and straight lines on a complex walking motion. Black and gray parts are respectively turns and straight portions of the walk. This example illustrates the error that could be made with manual segmentation of the trajectory. Indeed, the mid-part of the trajectory seems to be made of straight steps. Velocity and curvature parameters positively indicate that these steps are in fact transitions between two opposite turns.

Intuitively, one can think the same detection could be done simply by using a threshold on the variation of the direction. The problem is that using such a threshold induces some difficulties. Indeed, CoM oscillations always exist due to the stepping activity so the threshold is then hard to determine. Depending on the motion there can be not any or several valid thresholds to detect turns. Thresholds can also be variable according to subjects and to walking velocities. Therefore, it is difficult to standardize the detection and to apply it to all the motions automatically due to this threshold variability. Our method overcomes this drawback because it does not need any manual threshold tuning. Our criterion is robust to inter-individual variability and takes walking velocity changes into account. Consequently this method is completely automatic and does not need manual tuning. This method based on a common law of motion can then be complementary to example-based ones by helping the first identification of walking motions thanks to an automatic process.

The function parameters can be applied directly to a large population of walkers. In such a large population, the wide variety of morphologies can imply a dispersion of kinematic parameters. As performed classically in biomechanics, a normalization of the data by the legs length may increase the quality of invariance of the law. Even if this method has been set up on various subjects and can then be used on a large database of walking motions, it can be refined for a group of persons or a given performer. In the latter case, such an enhancement can be obtained by initializing our model on individual straight walking trajectories. The subject has then to perform several straight walks at various velocities in order to compute a significant statistical relation. This refinement

is interesting to enhance accuracy for very small turns. In any case, the shape of the set defining straight steps is invariant, only its size changes.

Moreover, our method is not dependent on the turning strategy. Indeed, a turn can be performed using spin, step or complex strategies. Depending on the strategy used, misinterpretation may occur. For example, several authors use the mean variation of the CoM orientation during each step. Unfortunately, it may lead to error when during a turn there is a lack of oscillation of the CoM. Such a case occurs during a step turn: the variation of CoM orientation can be equal to zero even during the turn.

Since the turning strategies detection is synchronized with the steps as well as motion clips in a motion database, these information can also be associated to the walking motions. During animation, the motions can then be adapted in different ways depending on the kind of turn currently performed (spin or step turn). This feature is useful to improve locomotion models to enhance realistic displacement of virtual characters in video games. In path planning for example, when the character evolves in a constrained environment, it may be necessary to accurately control the position of the footprints [22]. These footprints can then be precisely generated and adapted along the path according to the turning strategy.

5 Conclusion

In this paper, we proposed a method to automatically detect turns based on a velocity-curvature space representation of the walk. This method is accurate. It detects 100% of turns in standardized conditions for angles of at least 30°. Moreover, this detection does not require any threshold or manual tuning and provides a trajectory segmentation synchronized on the steps. Captured motions can be easily segmented into turns and straight walks in order to be included in a motion database. As a perspective, this method could be applied on other kinds of locomotor motions such as running in order to enrich motion database.

Acknowledgments

This research was founded by the ANR-PsiRob Locanthrope Project.

References

1. Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K., Pollard, N.S.: Interactive control of avatars animated with human motion data. ACM Trans. Graph. 21(3), 491–500 (2002)
2. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. ACM Trans. Graph. 21(3), 473–482 (2002)
3. Arikan, O., Forsyth, D.A., O'Brien, J.F.: Motion synthesis from annotations. ACM Trans. Graph. 22(3), 402–408 (2003)

4. Shin, H.J., Oh, H.S.: Fat graphs: constructing an interactive character with continuous controls. In: SCA, pp. 291–298 (2006)
5. Safonova, A., Hodgins, J.K.: Construction and optimal search of interpolated motion graphs. ACM Trans. Graph. 26(3), 106 (2007)
6. Rose, C., Cohen, M.F., Bodenheimer, B.: Verbs and adverbs: Multidimensional motion interpolation. IEEE CGA 18(5), 32–40 (1998)
7. Park, S.I., Shin, H.J., Shin, S.Y.: On-line locomotion generation based on motion blending. In: SCA, pp. 105–111 (2002)
8. Park, S.I., Shin, H.J., Kim, T.H., Shin, S.Y.: On-line motion blending for real-time locomotion generation: Research articles. Computer Animation and Virtual Worlds 15(3-4), 125–138 (2004)
9. Glardon, P., Boulic, R., Thalmann, D.: Pca-based walking engine using motion capture data. In: Computer Graphics International, pp. 292–298 (2004)
10. Kwon, T., Shin, S.Y.: Motion modeling for on-line locomotion synthesis. In: SCA, pp. 29–38. ACM, USA (2005)
11. Treuille, A., Lee, Y., Popović, Z.: Near-optimal character animation with continuous control. ACM Trans. Graph. 26(3) (2007)
12. Glaister, B., Bernatz, G., Klute, G., Orendurff, M.: Video task analysis of turning during activities of daily living. Gait & Posture 25(2), 289–294 (2007)
13. Taylor, M., Dabnichki, P., Strike, S.: A three-dimensional biomechanical comparison between turning strategies during the stance phase of walking. Human Movement Science 24(4), 558–573 (2005)
14. Olivier, A.H., Crétual, A.: Velocity/curvature relations along a single turn in human locomotion. Neuroscience Letters 412(2), 148–153 (2007)
15. Boulic, R., Ulicny, B., Thalmann, D.: Versatile walk engine. Journal of Game Development 1, 29–52 (2004)
16. Muller, M., Baak, A., Seidel, H.: Efficient and robust annotation of motion capture data. In: SCA (2009)
17. Kwon, T., Shin, S.Y.: A steering model for on-line locomotion synthesis. Computer Animation and Virtual Worlds 18(4-5), 463–472 (2007)
18. Shiratori, T., Nakazawa, A., Ikeuchi, K.: Rhythmic motion analysis using motion capture and musical information, pp. 89–94 (2003)
19. Fusco, N., Crétual, A.: Instantaneous treadmill speed determination using subject's kinematic data. Gait & Posture 28(4), 663–667 (2008)
20. Hase, K., Stein, R.: Turning strategies during human walking. Journal of Neurophysiology 81(6), 2914–2922 (1999)
21. Sedgeman, R., Goldie, P., Iansek, R.: Development of a measure of turning during walking. In: Advancing rehabilitation: Proceedings of the inaugural conference of the faculty of health sciences. La Trobe University (1994)
22. Choi, M.G., Lee, J., Shin, S.Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Trans. Graph. 22(2), 182–203 (2003)

Motion Pattern Encapsulation for Data-Driven Constraint-Based Motion Editing

Schubert R. Carvalho, Ronan Boulic, and Daniel Thalmann

EPFL VRlab
Station 14, CH 1015 Lausanne, Switzerland
{schubert.carvalho,ronan.boulic,daniel.thalmann}@vrlab.ch
http://vrlab.epfl.ch

Abstract. The growth of motion capture systems have contributed to the proliferation of human motion database, mainly because human motion is important in many applications, ranging from games entertainment and films to sports and medicine. However, the captured motions normally attend specific needs. As an effort for adapting and reusing captured human motions in new tasks and environments and improving the animator's work, we present and discuss a new data-driven constraint-based animation system for interactive human motion editing. This method offers the compelling advantage that it provides faster deformations and more natural-looking motion results compared to goal-directed constraint-based methods found in the literature.

Keywords: Motion Models, Motion Editing, Inverse Kinematics, PCA.

1 Introduction

Technological advances in motion capture (mocap) has allowed to provide high quality motions for computer animation. However, the captured animations almost always meet specific needs. Therefore, modifying and reusing these motions in new situations became an increasing area of research known as *motion editing*. In the past few years, there has been a lot of work in motion editing in the graphics community. The proliferation of mocap database has enabled the research and development of data-driven or model-based techniques [1,2,3,4,5,6]. Basically, these methods focus in constructing a model from mocap data - the data are usually represented by a low-dimensional space known as the *latent space* - to generate new motions from existing ones. Essentially because this representation can be exploited to analyze and synthesize the human motion within a low-dimensional space of physically balanced motions. Furthermore, model-based approaches are capable to generate more natural-looking motions compared to goal-directed ones, but the solutions are limited to the database. On the contrary, goal-directed approaches, for example, the ones based on pseudoinverse techniques that performs optimizations in the joint space - which includes the great majority of constraint-based methods - can achieve a wide range of

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 116–127, 2009.

user-defined tasks. However, their disadvantage is the frequent lack of naturalness when it comes to reproduce human activities.

As an effort of taking the advantages of model-based and goal-directed methods, we introduce a data-driven constraint-based motion editing technique, which combines the particularities of model-based and the versatility of goal-oriented approaches. Our method is based on the connection between linear motion models such as Principal Component Analysis (PCA), which is used to estimate a set of Principal Components (PC) and Principal Coefficients (PCs) [7], and Prioritized Inverse Kinematics (PIK), which is used to provide interactive motion editing. These links allowed us to construct a framework capable to solve a constraint-based optimization problem within the latent space. As a result, system performance is improved compared to pure goal-direct methods. Furthermore, to make easier the animator's work, we build a system to allow animators to generate natural-looking motions from key-frame constraints (i.e., the constraint is specified at specific poses) and key-trajectory constraints (i.e., the constraints are specified over a set of frames representing the trajectory of end-effectors). In order to enforce the spatio-temporal motion flow, continuity is imposed through the PC of the underlying motion pattern. By making use of the PC space, it becomes very efficient to enforce the fluidity of the motion compared to joint space-based methods. Accordingly, the editing becomes more intuitive.

To validate our framework, we have designed a number of experiments to adapt walking jump, reaching and golf swing motions to different situations and environment. The results are also compared against a constraint-based technique that performs deformations in the joint space [8]. After reviewing related work, we start describing the new approach for Data-Driven Constraint-Based Motion Editing.

2 Related Work

Note that, because of the vastness of the subject, this review is not intended to be exhaustive. Constraint-based techniques [9,8,10,11,12] provide a powerful framework to adapt recorded motions to different characters or to new situations from a set of user-specified constraints, such as end-effectors positions and end-effectors trajectories. To enforce a motion deformation an IK solver is normally used to solve user-specified constraints. To preserve the continuity between frames some IK solvers refer to the previous frame in order to choose a solution for the current one. When this is not the case other techniques such as filtering, B-Splines curves, collision avoidance and displacement maps are used to enforce the naturalness between frame. The main drawbacks of these continuity strategies are the time wasted traversing the character's substructures and computing convolution operations that degrade system performance. Data-driven approaches rely on mocap data to restrict the solution space of natural-looking motions. Motion graphs and motion interpolation are used to produce new motions from a database [13,14,15]. These techniques are not able to handle

end-effector constraints that are not represented in the motion database. These techniques present limitations in handling constraints over multiple frames. A number of researches have developed techniques to synthesize human motion in a low-dimensional space [1,16,6,2,3], both linear and non-linear models are used. Despite the fact that natural-looking motions can be produced, these techniques present some limitations regarding the database size, handling user-specified constraints or computation performance. A data-driven constraint-based system was proposed in [17], but the system was not capable to handle key-frame trajectory constraints. Recently, Recently, Raunhardt and Boulic [5], have been combined a data-driven and goal-oriented methods to construct a hybrid postural control approach overcoming their limitations. The main focus of the work is to treat the issue of controlling a full-body goal-directed motion (i.e. reach) where locomotion is not necessary. The model is learned from *pose* variations rather than *motion* data. The main difference between this work and ours is that, the optimization is preformed in the joint space rather than the latent space of the underlying motion pattern.

3 Motion Pattern Encapsulation

We refer to the process of learning the underlying motion pattern as *motion encapsulation*. We define a character pose, as a state vector describing the 3D global position, $\mathbf{P}_{root}$ and the 3D global orientation $\mathbf{Q}_{root}$ of the root node, and a set of joint orientations θ, represented by three parameters of the corresponding exponential map [18]:

$$\mathbf{\Theta} = \left\{ \mathbf{P}_{root}, \mathbf{Q}_{root}, \theta_n \right\}. \tag{1}$$

As we use motion capture (mocap) data from real people, and as each person tends to perform the same activity with some variability in speed, the database contains motions that, in general, have different durations. So, a duration normalization is necessary in our approach because the PCA's parameters are estimated from complete motions. The normalization is carried out by using quaternion spherical linear interpolation (Slerp) [19]. A motion is then represented as a line vector of the form:

$$\mathbf{\Psi}_i = \{\mathbf{\Theta}_1 \dots \mathbf{\Theta}_k \dots \mathbf{\Theta}_N\}. \tag{2}$$

$\mathbf{\Theta}_k$ is therefore a pose corresponding to a frame index k. Since a given motion consists of N poses, the motion vector has dimension: $D = (n \cdot N)$.

Once the mocap data are arranged in a matrix form, we use PCA to find a low-dimensional data representation, which efficiently acquires the important nuances of a specific motion pattern. By applying PCA, any normalized motion $\mathbf{\Psi}_i$ can be approximated as:

$$\mathbf{\Psi}_i \approxeq \alpha_i \mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_\circ \tag{3}$$

where $\mathbf{\Psi}_\circ$ is the mean motion, $\mathbf{E}_{\mathbf{\Psi}}$ are the Principal Components (PC) also referred as the *eigen-motions* and α_i are the so called Principal Coefficients

(PCs) that characterize the motion (i.e., the *latent space*). And a pose, here also referred as an *eigen-pose*, can be computed as a function of the scalar coefficients, α_i $(i = 1, \ldots, m)$ and the frame index k:

$$\psi\left(k, \alpha_1, \ldots, \alpha_m\right) \cong \alpha_i \mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}. \tag{4}$$

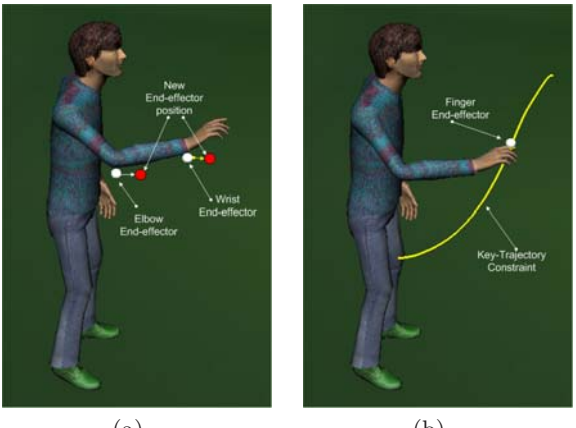
m represents the number of Principal Components that are required to reconstruct a desired percentage of the database [17]. Note that, the PC and the PCs are estimated off-line because the PC remain constant. We learn PCA motion models from three motion patters: walking jumps, golf swings and reaching motions. In the next section, we show how to explore PCA by building a framework to perform optimizations directly in the latent space.

4 Data-Driven Constraint-Based Optimization

4.1 Constraints

Providing interactive ways to manage constraints allow users to easily and rapidly modify preexisting human animations. In particular, geometric constraints, such as the position of an end-effector in the three-dimensional space [20] or the trajectory of an end-effector [8], are more intuitive for interactive manipulation because the user can specify a goal just by dragging an end-effector to a new position. We equip animators with two types of constraints: key-frame and key-trajectory constraints as illustrated in Figure 1, both are firstly created in the joint space and then mapped into the latent space.

Essentially, key-trajectory constraints are modeled as a *Kochanek-Bartels* spline [21]. This type of constraints allow the animator to edit a motion by specifying end-effectors across the whole motion as continuous trajectories. The



(a) (b)

Fig. 1. (a) key-frame constraints. (b) Key-trajectory constraints.

aim of this formulation is to reduce the work of the animator of manually specifying complete end-effector trajectories. On the other hand, with key-frame constraints end-effectors are attached directly on the character's body and dragged to knew positions. Key-frame constraints are pure positional constraints, that is, the effector does not follow any specific trajectory. A key-frame constraint is simply represented by a three-dimensional point x expressed in the so-called *task space*. The animator can use this type of constraints to edit the whole motion by constraining one key-frame on the motion.

4.2 Low-Dimensional Inverse Kinematics (LIK)

In this section, we demonstrate how standard IK techniques can be adapted to our needs. If we consider the pose of a virtual character as a n-dimensional vector (joint space, Eq. 1) and the position of the end-effector, x, as a p-dimensional vector (task space), the IK function can be defined as:

$$\mathbf{\Theta} = f^{-1}(x). \tag{5}$$

To solve this equation by using the well-known *resolved motion rate control* RMRC [22] technique, we need to compute the Jacobian matrix of the forward kinematics function, $x = f(\mathbf{\Theta})$. For a redundant manipulator the damped pseudo-inverse technique can be used [23]:

$$\Delta\mathbf{\Theta} = J(\mathbf{\Theta})^{\dagger^{\xi}} \Delta\mathbf{x} \tag{6}$$

where, $J(\mathbf{\Theta})^{\dagger^{\xi}}$ is the damped pseudo-inverse of the $p \times n$-dimensional joint space Jacobian matrix $J(\mathbf{\Theta}) = \partial x / \partial \mathbf{\Theta}$. The damped factor, ξ, is added to prevent Jacobian's singularities and to stabilize IK solutions. For a redundant manipulator, i.e., $n > p$, the problem is ill-posed (i.e., there is no unique solution). Hence, IK algorithms should determine one solution to Eq. 6 given many possibilities.

As an effort to restrict solutions within the space of feasible motions (i.e., the latent space), we integrate the human behavior directly in the optimization process to obtain more realistic natural-looking solutions. In practice, to compute the pose increment in the latent space we need to solve following equation:

$$\Delta\alpha = J(\alpha)^{\dagger^{\xi}} \Delta x \tag{7}$$

where, $J(\alpha)^{\dagger^{\xi}}$ is the pseudo-inverse of the PCs Jacobian $J(\alpha) = \partial x / \partial \alpha$, which relates a variation Δx in the Cartesian space with a variation $\Delta\alpha$ in the latent space. The computation of $J(\alpha)$ can be easily carried out with the chain rule [24]. So, once we have the new increment $\Delta\alpha$ the new pose can be computed in two steps. First, $\Delta\alpha$ is added with respect to α to obtain the final principal coefficients,α^{υ}:

$$\alpha^{\upsilon} = \alpha + \Delta\alpha. \tag{8}$$

Finally, by using Eq. 4, we compute the new state vector as follows:

$$\psi\,(k, \alpha^{\upsilon}) \approx \alpha^{\upsilon}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}. \tag{9}$$

In the next section, we show how to extend the current approach to deal with multiple tasks and to resolve their possible conflicts in the latent space.

4.3 Low-Dimensional Prioritized Inverse Kinematics (LPIK)

Let us define a set of p tasks, each one satisfying its goal $\mathbf{g}_j$: $x(\alpha)_j = \mathbf{g}_j$, $j \in [1, \ldots, p]$, and having its corresponding increment, $\Delta x = (\Delta x_1, \ldots, \Delta x_p)$.

In the case of multiple tasks occurring at the same key time k, the optimal solution α^* should satisfy all of them, i.e., $x(\alpha^*)_j = \mathbf{g}_j$, $\forall_j$. However, this may be difficult because some of the tasks may be in conflict. Tasks are said to be in conflict when they are not achievable simultaneously.

To solve a conflict task, we use a technique based on a priority strategy. The major strength of this technique is that prioritized constraints are sorted into priority-layers. As a result, constraints belonging to the highest priority layer are enforced first (e.g., feet on the ground). Then, those of the next priority-layer are satisfied as much as possible without disturbing the previous constraints, and so on. Based on a previous work done by [22,20], we reformulate the prioritized strategy to work in the latent space as follows [17]:

$$\Delta\alpha = J(\alpha)^{\dagger^{\xi}} \Delta x + \mathrm{P}_{N(J(\alpha))}\mathrm{z} \tag{10}$$

$$\mathrm{P}_{N(J(\alpha))} = \mathrm{I}_m - J^{\dagger}(\alpha)J(\alpha).$$

where $J(\alpha)^{\dagger}$ is the $m \times p$ pseudo-inverse of J_α, $\mathrm{P}_{N(J(\alpha))}$ is the $m \times m$ projector operator onto the null-space of J_α, I_m is the $m \times m$ identity matrix and z is the m-dimensional PCs variation vector. The algorithm that solves Eq. 10 is presented in [17]. During the editing the animator has to setup the optimizer's parameters, that is, the number of iterations and the damping factor.

5 Imposing Continuity

We introduce two techniques for imposing motion continuity based on PCA. First, the strategy elaborated for imposing continuity from key-trajectory constraints uses the eigen-motion space. As in standard per-frame constraint-based motion editing techniques [9,8], the user needs to relax the motion by using ease-in/ease-out time intervals to prevent discontinuities. Note that, the $\{\alpha\}$ vector describes a complete motion belonging to the latent space. Therefore, solving the problem in the latent space for many poses for determining a unique $\{\alpha^{\upsilon}\}$ satisfying all the user-specified constraints across the motion, can easily over-constraints the problem due to the small size of the latent space. In practice, we ease the constraints by using a frame by frame approach, such that, from to beginning to the ending of the motion sequence the LPIK optimizes the initial $\{\alpha\}$ for each constrained frame obtaining a new set of principal coefficient vectors, such as: $\{\alpha_1^{\upsilon}, ..., \alpha_k^{\upsilon}, ..., \alpha_N^{\upsilon}\}$. Therefore, each pose is computed using Eq. 4. Figure 2 illustrates the process. Second, in our framework, impose continuity

from key-frame constraints is a straightforward process using the Eq. 3. As the principal coefficients are learned from complete motions, each vector $\{\alpha\}$ characterizes a complete animation. Therefore, if the parameter vector is optimized for a specific key, k, resulting from a constraint imposed on a pose $\mathbf{\Theta}_k$, the updated set $\{\alpha^{\upsilon}\}$ defines one full motion belonging to the latent space. Once the solution coefficients are computed, they are also used to define the other poses on the motion by using Eq. 3. Figure 2(b) illustrates the process.

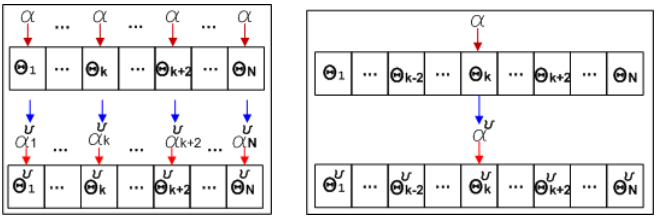


Fig. 2. (a) Once the LPIK converges to a $\{\alpha_k^{\upsilon}\}$ satisfying all the constraints of a pose k, the system uses Eq. 4 to recover the deformed pose: $\mathbf{\Theta}_k^{\upsilon} = \alpha_k^{\upsilon}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}$. The process is repeated frame by frame. (b) Motion editing from specific key-time constraints. The complete animation $\{\mathbf{\Psi}^{\upsilon}\}$ is computed as: $\mathbf{\Psi}^{\upsilon} = \alpha^{\upsilon}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}$.

6 Overview of the System

6.1 Implementation

The motion editing system that we use as a basis for all experiments has been implemented in three languages: C++, C and Matlab. The system is subdivided in off-line and on-line computations. We used Matlab to construct the PCA motion model. We chose Matlab because it allows easy matrix manipulation, it is stable and it is well adopted in the scientific community. The PCA parameters are stored as text files, which are loaded in the system's data structure when it is started. The computationally expensive calculations, such as the optimization and the motion generation strategy, and the PCA synthesis are all done in C and C++. In all the experiments reported in this chapter are run on a 3.2GHz Pentium Xeon(TM) with 1GB memory.

6.2 System Interface and Functionalities

The data-driven constraint-based motion editing system proposed in this work was integrated into the Autodesk/Maya software as plug-in and MEL scripts. The character model can be loaded from description files, and interactively edited in the application. The deformed animation can be saved for future use, either as a full-body motion or as a set of latent variables. The window devoted to the management of end-effectors and other objects in the scene is shown in left side

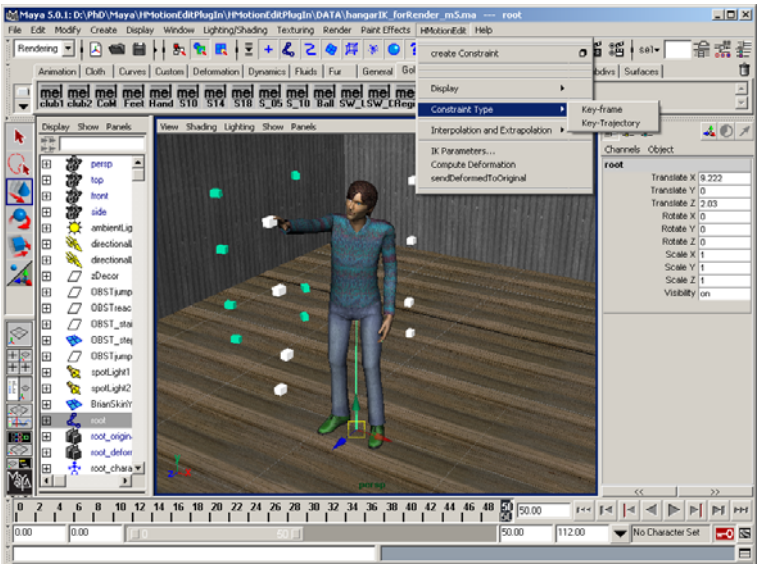


Fig. 3. Data-driven constraint-based motion editing system interface. The green and white cubes represent the reaching positing in the database.

of Figure 3. End-effector can be created by selecting among a list of predefined joints available in the window (left side of Figure 3). Another alternative and more direct solution is to pick a point on the surface of a character body with the mouse pointer, in the application window. This allows to control any visible part of the character body just by clicking on it.

7 Experiments

In this section, we have designed three case studies to evaluate the usability and the performance of the proposed data-driven constraint-based motion editing system. To validate our framework, we perform experiments to compare our approach against a prioritized constraint-based technique that performs deformations in the joint space [8], regarding performance, robustness, simplicity, continuity and realism, by synthesizing golf swing, walking jump and reaching motions. For a fair comparison between both techniques, in the joint constraint-based system, we set the parameters of the optimizer as suggested by the authors. We made the same with our system.

In the first case study, we have retarget a golf swing, with 132 frames, executed on a flat ground to a up slope ground with 7° anti-clockwise, by adjusting the feet position and by shifting the hit position of the golf club head, Figure 4. We used a motion model learned from 16 normalized golf swings played on up slopes, for angles ranging from 0.5° to 5.0° (anti-clockwise), by increments of 0.5°, each motion has 132 frames. We used a database percentage of 98%, which

(a) (b)

Fig. 4. The initial hit position is illustrated by a transparent ball. In both methods the golfer hits the ball (a,b), but with the PIK the club head hit the ground (b).

(a) PIK (b) LPIK

Fig. 5. Pink arrow shows the current foot position and the red shows the sliding gap

gave $m = 15$ dimensions, and we set the optimizer's parameters to: 100 number of iterations and $\xi = 0.8$. The adjusted posture, at frame 94, and the position of the golf club with respect to the hands were the same in both cases. A key-frame constraint were used for the data-driven approach and key-trajectory constraints were used instead for the the joint approach, as it cannot handle key-frame constraints. It is important to mention that, the 7° slope was not learned by the model. The motion editing system based on the joint optimization generated a less realistic swing motion compared to one based on the latent space. We observed that, the character's left hand lost the contact with the golf club and the club head hit the ground. On the contrary, with the proposed approach, only one key-frame constraint was necessary - on the hit pose - to achieve a globally continuous motion. The computing performance needed to generate the up slope motion for the joint and latent approaches were 102 and 3 seconds, respectively.

In the second case study, we considered a simple task: the character has to reach an object. We then attached one constraint on the character's hand, at frame 50, to drive the reach in the direction of the goal. The constraint was activated over all frames because the character had to follow the path from

Fig. 6. Fist row shows the input and the generated motion by the PIK-based system. Second row the same for the LPIK-based system. The pink arrow shows the current foot position, the red the sliding gap and the blue the anticipation.

start to end. This constraint configuration was used in both systems. We used a motion model learned from 16 normalized reaching motions, such that each motion has 50 frames. We considered a database percentage of 98%, which gave $m = 15$ dimensions, and we set the optimizer's parameters in both systems to: 100 number of iterations and $\xi = 10$. We observed in Figure 5 that the PIK-based system generated a motion in which the character slides to reach the green bar. On the other hand, the latent space generated an animation without introducing motion artifacts foot slides.

In the last case study, we took a walking jump sequence of $0.8m$ and we modify it to produce a higher jump. By attaching one constraint on the character root node, at frame 15, which is the time of the apex of the jump, and moving the constraint to a higher location. We used key-trajectory constraint for the joint approach and key-frame constraints for the latent one. We have used a motion model learned from 89 normalized motions of five men and one woman performing walking jumps of 3 lengths ranging from $0.4m$ to $1.2m$, by increments of $0.4m$, such that each motions has 26 poses. We used a database percentage of 95%, which gave $m = 30$ dimensions, and we set the optimizer's parameters in both systems to: 100 number of iterations and $\xi = 10$. The end-effector's final goal was the same in both systems. Figure 6 shows the generated motions from key-trajectory and key-frame constraints. The PIK-based system produced a higher jump motion, but by moving the constraint to a higher position made the character lose ground contact, which is not desired because its not natural.

On the other hand, LPIK-based system not only generated a higher jump, but also the resulted animation kept ground contact.

8 Conclusions

In this work, we have proposed a new approach for data-driven constraint-based motion editing. Our technique is based on the link between linear motion models such as Principal Component Analysis (PCA) and Prioritized Inverse Kinematics (PIK). The connection of both techniques allowed us to construct a Low-dimensional Prioritized Inverse Kinematics (LPIK) framework. The solver handles deformation problems within the latent space of some underlying motion pattern. By making use of the pattern space, we increased the possibilities of performing IK in the space of feasible poses.

We have demonstrated that our method achieves a degree of generality beyond the motion capture data. For example, we have retarget a golf swing motion to a 7° up slope using constraints that cannot be satisfied directly by any motion in the database, and have found that the quality of the generated motion was believable. We have seen that our approach is well-suited to deal with deformations and retargeting problems. We have demonstrated the robustness of our approach, deforming three types of movements and comparing the results against a goal-directed constraint-based technique that perform optimizations in the joint space. The system behaves well and robustly, as long as the model has sufficient information to handle the user-specified constraints.

Acknowledgments. The authors would like to thank to Autodesk/Maya for their donation of Maya software. Many thanks to Mireille Clavien for video production. This project was sponsored by the EPFL - Sport and Rehabilitation Engineering program.

References

1. Safonova, A., Hodgins, J.K., Pollard, N.S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. ACM Trans. Graph. 23(3), 514–521 (2004)
2. Glardon, P., Boulic, R., Thalmann, D.: Robust on-line adaptive footplant detection and enforcement for locomotion. Vis. Comput. 22(3), 194–209 (2006)
3. Chai, J., Hodgins, J.K.: Constraint-based motion optimization using a statistical dynamic model. ACM Trans. Graph. 26(3), 8 (2007)
4. Urtasun, R., Glardon, P., Boulic, R., Thalmann, D., Fua, P.: Style-based motion synthesis. Computer Graphics Forum (CGF) 23(4), 799–812 (2004)
5. Raunhardt, D., Boulic, R.: Motion constraint. Vis. Comput. 25(5-7), 509–518 (2009)
6. Shin, H.J., Lee, J.: Motion synthesis and editing in low-dimensional spaces. Comput. Animat. Virtual Worlds 17(3‐4), 219–227 (2006)
7. Jolliffe, I.T.: Principal Component Analysis. Springer, Heidelberg (1986)

8. Le Callennec, B., Boulic, R.: Interactive motion deformation with prioritized constraints. Graphical Models 68(2), 175–193 (2006); Special Issue on SCA 2004
9. Gleicher, M.: Comparing constraint-based motion editing methods. Graphical models 63(2), 107–134 (2001)
10. Kulpa, R., Multon, F., Arnaldi, B.: Morphology-independent representation of motions for interactive human-like animation. In: EUROGRAPHICS, August-September 2005, vol. 24, pp. 343–352 (2005)
11. Lee, J., Shin, S.Y.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of ACM SIGGRAPH (1999)
12. Liu, L., Zhao-qi, W., Deng-Ming, Z., Shi-Hong, X.: Motion edit with collision avoidance. In: Proceedings of the WSCG, January 2006, pp. 303–310 (2006)
13. Arikan, O., Forsyth, D.A.: Interactive motion generation from examples. In: SIGGRAPH 2002: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 483–490. ACM, New York (2002)
14. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. ACM Trans. Graph. 21(3), 473–482 (2002)
15. Mukai, T., Kuriyama, S.: Geostatistical motion interpolation. ACM Trans. Graph. 24(3), 1062–1070 (2005)
16. Grochow, K., Martin, S.L., Hertzmann, A., Popovi, Z.: Style-based inverse kinematics. ACM Trans. Graph. 23(3), 522–531 (2004)
17. Carvalho, S.R., Boulic, R., Thalmann, D.: Interactive low-dimensional human motion synthesis by combining motion models and pik. Computer Animation & Virtual Worlds 18 (2007); Special Issue of Computer Animation and Social Agents (CASA 2007)
18. Grassia, F.S.: Practical parameterization of rotations using the exponential map. Journal of Graphics Tools 3(3), 29–48 (1998)
19. Shoemake, K.: Animating rotation with quaternion curves. In: SIGGRAPH 1985, pp. 245–254. ACM Press, New York (1985)
20. Baerlocher, P.: Inverse Kinematics Techniques of The Interactive Posture Control of Articulated Figures. Phd thesis, cole Polytechnique Fdral de Lausanne (EPFL) - IC School of Computer and Communication Sciences (2001)
21. Kochanek, D.H.U., Bartels, R.H.: Interpolating splines with local tension, continuity, and bias control. SIGGRAPH Comput. Graph. 18(3), 33–41 (1984)
22. Whitney, D.E.: Resolved motion rate control of manipulators and human prostheses. IEEE Trans. Man-Mach. Syst. 10, 47–53 (1969)
23. Maciejewski, A.A.: Dealing with the ill-conditioned equations of motion forarticulated figures. In: IEEE Computer Graphics and Applications, May 1990, vol. 10, pp. 63–71 (1990)
24. Carvalho, S.R., Boulic, R., Thalmann, D.: Motion pattern preserving ik operating in the motion principal coefficients space. In: Proceedings of 15-th WSCG, January-February 2007, pp. 97–104 (2007)

Real-Time Character Control for Wrestling Games

Edmond S.L. Ho and Taku Komura

Institute of Perception, Action and Behaviour
School of Informatics, University of Edinburgh

Abstract. This paper proposes a method to simulate the real-time interactions of tangling motions by two virtual wrestlers in 3D computer games. The characters are controlled individually by two different players - one player controls the attacker and the other controls the defender. We make use of the topology coordinates which are effective to synthesize tangling movements. The attacker's movements are simulated by changing the topology coordinates at every frame, and the defender is controlled to escape from such an attack by inverse kinematics. The experimental results show the methodology can simulate realistic competitive interactions of wrestling in real-time, which is difficult by previous methods.

Keywords: character animation, motion capture.

1 Introduction

Wrestling is a major field in 3D computer games. The motions in wrestling involve close contacts between the characters such as squeezing and locking. Simulating such motions is not easy as they require a great amount of close contacts and collision avoidance. In fact, in most of the wrestling games, such motions are designed carefully in advance by the animators and the game players have little control over the characters once complex tangling motions are started. This is completely different from real wrestling - wrestlers have lots of degrees of freedom to escape from the attackers, and attackers need to carefully select their motion to lock the defender. Such complex interactions of the wrestlers are rarely simulated in the existing wrestling games.

In this paper, we make use of the topology coordinates [5] in order to simulate such complex interactions in real-time. The attacker is controlled so that its topology coordinates approach to those of the target configuration. The player can switch the attacks according to the configuration of the attacker and the defender. On the other hand, the defender can escape from such attacks by kinematically controlling the body.

Our interface provides large degrees of freedom to the game players while minimizing the complexity of control, which can increase the attractiveness of wrestling games.

2 Related Work

We first briefly review the recent wrestling games and their interfaces. Next, we review researches of two topics that are related to character control in wrestling games - real-time character control and close interactions of multiple characters.

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 128–137, 2009.

Wrestling games have been attracting millions of game players around the world and has become one of the major categories in computer games. In the old games, the attacks were limited to hits such as punches, kicks or chops. Therefore, the users could only repeatedly press the buttons to give large damage to the opponent character.

The recent advanced games allow the characters to conduct complex tangling attacks such as back drops, rear-choke hold and full nelson attacks. In order to launch such motions, the user is supposed to press the button at the correct pose and timing, or select the part to attack by the pointing device [1]. Motions that involve tangling are usually just replayed during run-time as real-time editing of such motions can easily result in collision and penetration of the body segments. The attractiveness of the games will be greatly enhanced if the game players have access to the details of the tangling motions during the game play.

Real-time character control: Here we review a number of techniques which are useful for real-time control of the virtual wrestlers. We first review techniques of inverse kinematics (IK) which is a basic technique to edit character motions, and then their extensions to handle dynamics and tangling motions.

Inverse kinematics (IK) [20,13,14,23,21] is a basic technique that is often used for real-time control of characters. Methods to control characters of arbitrary morphology [7,3] have also been proposed. IK methods can be divided into (1) CCD-based approaches [9], (2) analytical approaches [11], (3) particle-based approaches [7,3] and (4) quadratic programming based approaches [20,13,21].

Among these approaches, the quadratic programming based approach has an advantage to simulate wrestling motions as it can enclose constraints based on dynamics [22,8,16] and topological relationships [5] into the solver. Our approach is built on top of Ho and Komura [5], which propose to handle tangling motions by adding constraints into an optimization-based IK solver.

Multiple Character Interactions: The simulation of interactions between multiple characters has many applications such as computer games, virtual environments and films. Liu et al.[12] simulate the close dense interactions of two characters by repetitively updating the motion of each character by spacetime constraints. Lee and Lee [10] simulate the boxing match by using reinforcement learning. Treuille et al [19] also use reinforcement learning to simulate the pedestrians avoiding each other. Shum et al [17] use min-max search to find the optimal action in a competitive environment. They also propose a real-time approach based on an automatically produced finite state machine [18]. These researches do not handle very close interactions such as holding or wrestling. Ho and Komura [4] generate wrestling motions by finding the topological relationship of characters from the template postures and use PD control to simulate movements where the topological relationship is kept the same. If we want to simulate a scene where the topological relationship of characters changes in time, we cannot apply such a method. A method to dynamically update the postures and the topological relationships is required. [6] propose to evaluate the similarity of character postures based on the topological relationships. When equivalent postures are found, the postures are linearly interpolated at the level of generalized coordinates. However, no method has enabled game players to interactively change the topological relationship of virtual characters in real-time. We propose such an approach in this paper.

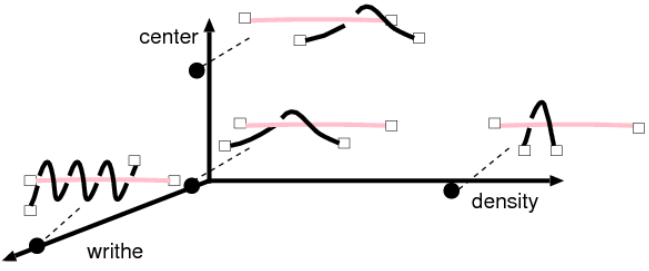


Fig. 1. The three axes in topology space : writhe, center and density. The center, which specifies the central location of the twist relative to each strand, is actually composed of two scalar parameters, although it is represented by a single axis in this figure. The density tells which strand is playing the major role to compose the twist.

3 Methodology

We first briefly review the topology coordinates [5], which is the basic technique used to control the tangling motions. Next, we explain how the topology coordinates can be applied to the control of virtual wrestlers in computer games.

3.1 Topology Coordinates

Topology coordinates enable characters to tangle their bodies with other characters without conducting global path planning methods. The topology coordinates are composed of three attributes, the writhe, center and density. The first attribute **writhe** counts how much the two curves are twisting around each other. Writhe can be calculated by using Gauss Linking Integral (GLI) [15] by integrating along the two curves γ_1 and γ_2 as:

$$GLI(\gamma_1, \gamma_2) = \frac{1}{4\pi} \int_{\gamma_1} \int_{\gamma_2} \frac{d\gamma_1 \times d\gamma_2 \cdot (\gamma_1 - \gamma_2)}{\|\gamma_1 - \gamma_2\|^3} \tag{1}$$

where $\times$ and $\cdot$ are cross product and dot product operators, respectively. The GLI computes the average number of crossings when viewing the tangle from all directions.

Curves can twist around each other in various ways. In order to further specify the status of the two chains, we introduce the other two attributes, **center** and **density**. Examples of changing these attributes for a pair of strands are shown in Figure 1. The center, which is composed of two scalar parameters, explains the center location of the twisted area, relative to each strand. The density, which is a single scalar parameter, explains how much the twisted area is concentrated at one location along the strands. When the density is zero, the twist is spread out all over the two strands. When the density value is either very large or very small, we can say one strand is playing a major role to compose the twist, as it is twisting around the other strand which is kept relatively straight (Figure 1). When the density turns from negative to positive, or vice versa, the strand playing the major role switches.

3.2 Controlling the Characters

We represent the bone structure of the virtual wrestlers by a set of line segments. Therefore, now we will mathematically define the topology coordinates of serial chains. Let us assume we have two chains S_1 and S_2, each composed of n_1 and n_2 line segments, connected by revolute, universal or gimbal joints (Figure 2). In this case, we can compute the total writhe by summing the writhes by each pair of segments:

$$w = GLI(S_1, S_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} T_{i,j} \tag{2}$$

where w represents the writhe, $T_{i,j}$ is the writhe between segment i on $S1$ and j on $S2$. Let us define a $n_1 \times n_2$ matrix $\mathbf{T}$ whose (i, j)-th element is $T_{i,j}$, and call this the **writhe matrix**. The writhe matrix explains how much each pair of segments from S_1 and S_2 contributes to the total writhe value. Various twists of two serial chains and the corresponding writhe matrices are shown in Figure 3.

The topology coordinates can be updated by changing the distribution of the elements in the writhe matrix using basic operations such as rotation, translation and scaling. Rotating the elements results in changing the density. Translating the elements results in changing the center. Scaling the whole matrix results in changing the writhe. Let us define these operations by $R(M,d)$, $Tr(M,c)$ and $S(M,w)$, respectively, where M is the input matrix and (d,c,w) are topology coordinates, each of which representing the density, center and writhe, respectively.

Rather than directly manipulating the writhe matrix of the characters, we first compute an ideal, desired writhe matrix and try to minimize the difference of the character's writhe matrix and the desired writhe matrix. The desired writhe matrix T_d that corresponds to topology coordinates (d,c,w) is computed by sequentially applying $R()$, $Tr()$ and $S()$ to a matrix $\mathbf{I}$, which is a $n_1 \times n_2$ matrix who has values evenly distributed at the $(n_2+1)/2$-th column if n_2 is odd, or at both the $n_2/2$ and $n_2/2+1$-th column if it is even:

$$\mathbf{T} = S(Tr(R(\mathbf{I}, d - \frac{\pi}{4}), \mathbf{c}), w). \tag{3}$$

where

$$\mathbf{I} = \begin{cases} \begin{pmatrix} 0 \cdots, \frac{1}{n_1}, \cdots, 0 \\ \vdots \\ 0 \cdots, \frac{1}{n_1}, \cdots, 0 \end{pmatrix} (n_2 \text{ is odd}) \\ \begin{pmatrix} 0 \cdots, \frac{1}{2n_1}, \frac{1}{2n_1}, \cdots, 0 \\ \vdots \\ 0 \cdots, \frac{1}{2n_1}, \frac{1}{2n_1}, \cdots, 0 \end{pmatrix} (n_2 \text{ is even}) \end{cases} \tag{4}$$

and $\frac{pi}{4}$ is an offset to adjust the density d due to its definition [5].

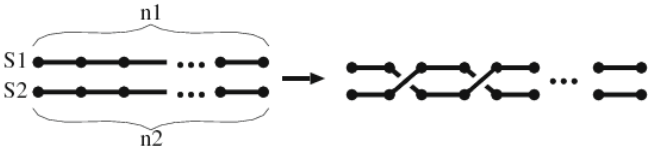


Fig. 2. Twisting a chain of line segments around each other

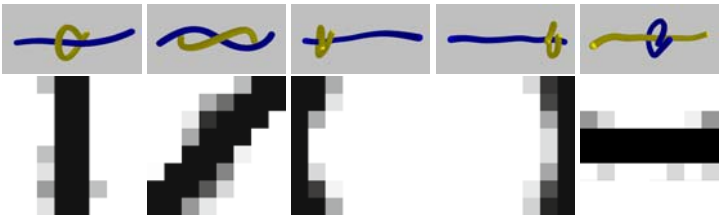


Fig. 3. (upper) Tangles with different density and center, and (lower) the distribution of elements with large absolute values in the corresponding writhe matrices. The darkness represents the amplitude of the absolute value.

Once the desired writhe matrix T_d is computed, the character is guided to the desired posture by updating the generalized coordinates so that the writhe matrix of the character T becomes similar to the desired writhe matrix T_d. This problem is solved by quadratic programming:

$$\min_{\Delta\mathbf{q_1},\Delta\mathbf{q_2},\delta} \|\Delta\mathbf{q_1}\|^2 + \|\Delta\mathbf{q_2}\|^2 + \|\delta\|^2 \text{ s.t.} \tag{5}$$

$$\Delta\mathbf{T} = \frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1} + \frac{\partial\mathbf{T}}{\partial\mathbf{q_2}}\Delta\mathbf{q_2} \tag{6}$$

$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma(1 \leq i \leq n_1, 1 \leq j \leq n_2) \tag{7}$$

$$\mathbf{T} + \Delta\mathbf{T} - \mathbf{T^d} + \delta = \mathbf{0} \tag{8}$$

$$\mathbf{r} = \mathbf{J_1}\Delta\mathbf{q_1} + \mathbf{J_2}\Delta\mathbf{q_2} \tag{9}$$

where $(\mathbf{q_1}, \mathbf{q_2})$ are the generalized coordinates of the two chains, $(\Delta\mathbf{q_1}, \Delta\mathbf{q_2})$ are their updates to be made at this iteration, $\Delta\mathbf{T_d}$ is the update of the writhe matrix, σ is a threshold, that is set to 0.2 in our experiments to avoid the segments to approach too close to each other, δ is a vector of slack parameters introduced to minimize the difference of the desired writhe matrix and that of the controlled characters, Equation (9) represents the other kinematical constraints which can be linearized with respect to $(\Delta\mathbf{q_1}, \Delta\mathbf{q_2})$ when the movement is small, such as the movements of any parts of the body in Cartesian coordinates or the center of mass. $\mathbf{J_1}, \mathbf{J_2}$ are the Jacobians of this constraint, and $\mathbf{r}$ is the linearized output of this constraint. The updated generalized coordinates $(\mathbf{q_1} + \Delta\mathbf{q_1}, \mathbf{q_2} + \Delta\mathbf{q_2})$ correspond to the target topology coordinate at the next time step, $(w + \Delta w, d + \Delta d, \mathbf{c} + \Delta\mathbf{c})$. By solving Equation (5) at every time step, we simulate the interactions of two virtual wrestlers.

3.3 Real-Time Control of Wrestlers

In this subsection, we explain how to simulate the interactions of two wrestling characters, each controlled by different game players. We assume one character is attacking the other character by moves that involve a lot of tangling.

Let us first give an overview of the computation of the characters' motions. The motion of each character is computed sequentially by two different quadratic programming problems. The attacker's movement is guided by the topology coordinates - once the attack is specified, the attacker tries to tangle its body with the defender's body according to the target configuration. The attack can be switched in the middle if the player thinks a different attack is more effective under the current configuration. The defender needs to escape from the attacks by controlling the body segments involved in the tangling process. The player uses a pointing device to move a body segment and the posture of the defender is computed by inverse kinematics based on quadratic programming.

Now the method to control the attacker is explained. The attacker's motion is determined based on the defender's current posture and the target topology coordinates in the next time step. Let us assume the configurations of the attacker and defender are represented by $\mathbf{q_1}$ and $\mathbf{q_2}$, respectively. Solving Equation (8) is not a good idea to compute the motion of the attacker, as the updates of the attacker's movements takes into account the movements of the defender in the next time step. This makes the defender difficult to escape from the attacker. Therefore, we solve the following problem for computing the motion of the attacker:

$$\min_{\Delta \mathbf{q_1}, \delta} \|\Delta \mathbf{q_1}\|^2 + \|\delta\|^2 \text{ s.t.} \tag{10}$$

$$\Delta \mathbf{T} = \frac{\partial \mathbf{T}}{\partial \mathbf{q_1}} \Delta \mathbf{q_1} \tag{11}$$

$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma (1 \leq i \leq n_1, 1 \leq j \leq n_2) \tag{12}$$

$$\mathbf{T} + \Delta \mathbf{T} - \mathbf{T^d} + \delta = \mathbf{0} \tag{13}$$

$$\mathbf{r_1} = \mathbf{J_1} \Delta \mathbf{q_1} \tag{14}$$

where $\mathbf{r_1}$ represents the kinematic parameters of the attacker. This technique adds an effect of physiological delay for launching a response motion with respect to the defender's movements, which increases the realism of the interaction. It also adds an essence of a game play to the interaction between the two virtual wrestlers as the attacker may not be able to achieve the target topology coordinates if the defender is controlled well.

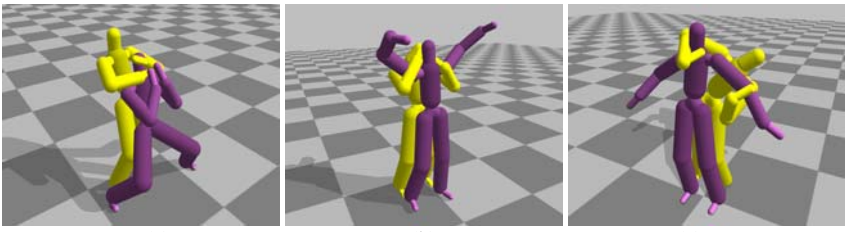
Next, the defender's movement is computed by solving the following inverse kinematics problem:

$$\min_{\Delta \mathbf{q_2}} \|\Delta \mathbf{q_2}\|^2 \text{ s.t.} \tag{15}$$

$$\Delta \mathbf{T} = \frac{\partial \mathbf{T}}{\partial \mathbf{q_2}} \Delta \mathbf{q_2} \tag{16}$$

$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma (1 \leq i \leq n_1, 1 \leq j \leq n_2) \tag{17}$$

$$\mathbf{r_2} = \mathbf{J_2} \Delta \mathbf{q_2}. \tag{18}$$



(a) (b) (c)

Fig. 4. Various wrestling interactions created by the proposed method

where $\mathbf{r_2}$ represents the kinematic parameters of the defender, based on the input from the pointing device and any other kinematic constraints. The GLI between every segment pairs are still considered to avoid penetrations.

The attacker's motion is computed first by solving *Equation* (10) and the attacker's posture is updated. Then, the defender's motion is computed by solving *Equation* (15). Once both character's motion is updated, the time counter is increased.

The key point of controlling the defender is to move the body such that it can efficiently escape from the attacks. Such movements are those which can reduce the writhe value by little motion. For example, when the attacker starts to shift to a configuration of a rear-choke hold (Figure 4 (a)) , which is an attack to squeeze the neck from behind, the most efficient way to escape is to knee down at the last moment, which requires little movement. On the other hand, if the attacking player can predict such escaping moves of the defender and switch to another move that can make use of such movements, the player can efficiently tangle the attacker's body to the defender.

4 Experimental Results

We have simulated a number of interactions between two virtual wrestlers both controlled by game players. Those include rear-choke hold (Figure 4 (a)), Full Nelson hold (Figure 4 (b)) and a number of different squeezing motions from the back. The virtual wrestlers start from separate postures and the attacker approaches to the defender to tangle its body with it.

In our demos, we used a human character model of 42 DOF. All DOFs are used when controlling the characters. For the human character model, when one character is controlled, we can obtain an interactive rate of 40 frames per second when using a Pentium IV 3.2 GHz PC. When two characters are simultaneously controlled, we can still obtain a frame rate of 30 frames per second. We use ILOG CPLEX 9.1 [2] as our quadratic programming solver.

The movement of the attacker is controlled by specifying the topology coordinates. In the first animation, the attacker performed an attack by tangling i) it's right arm with the neck of the defender and ii) it's left arm with the left arm of the defender. Without controlling the defender, this attack can be performed easily by changing the topology coordinates of the attacker as shown in Figure 5.

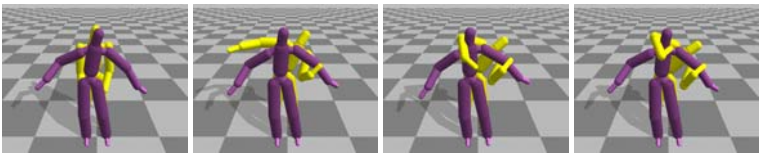


Fig. 5. Without controlling the defender (in purple), the attacker (in yellow) can tangle with the defender easily by changing the topology coordinates

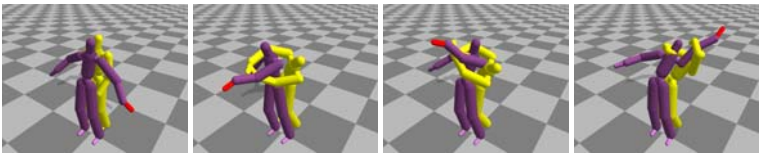


Fig. 6. The defender (in purple) cannot escape from the attack if the player does not control it quick enough

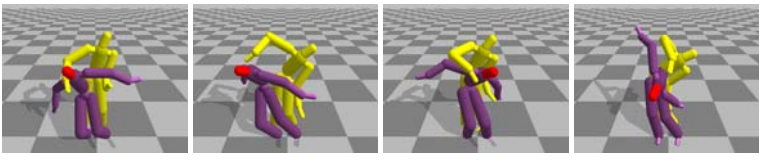


Fig. 7. The defender (in purple) escapes from the attack

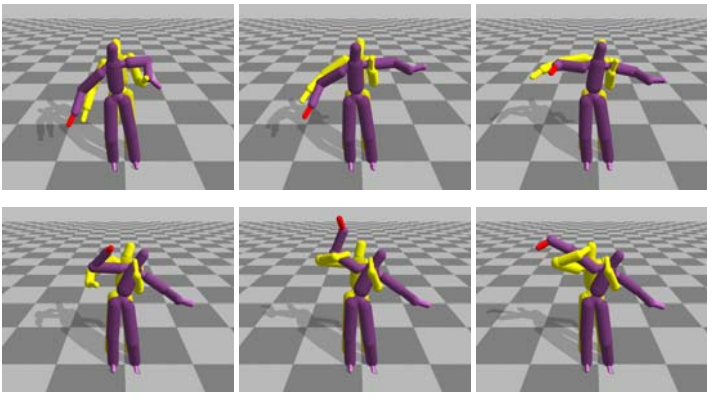


Fig. 8. The attacker (in yellow) switched the attack in the middle in order to lock the defender (in purple)

In the second example, the defender tries to escape from the attacks shown in Figure 5. In our implementation, the player can specify kinematic constraints on the defender by dragging the body segments using a pointing device. In Figure 6, the player dragged the left hand (colored in red) of the defender. However, the defender fails to escape from the attack as it was not controlled quick enough. In Figure 7, the defender escapes from the attack successfully by moving the torso quickly and vigorously.

The attacker can also switch to another wrestling move in the middle of the attack. In the third example, the defender escapes from the attack by blocking the right arm of the attacker in the early stage of the interaction. Then the attacker switches to another wrestling move by tangling its right arm with the right arm of the defender and its left arm with the torso and neck of the defender. Finally the attacker successfully locks with the defender. The screenshots of this interaction are shown in Figure 8.

5 Discussions and Future Work

In this paper, we have introduced a method to make use of topology coordinates for real-time interactions of two virtual wrestlers each of which controlled by game players. By computing the movements of the two characters by two independent quadratic programming problems, we can increase the reality of the interactions and the essence of the game play. Our experimental results show that the application of topology coordinates greatly increases the attractiveness of wrestling games from the following view points:

1. The players, especially the defender has great access to the kinematics of the virtual wrestlers which increases the variety of movements.
2. Although the kinematics are controlled interactively, the topology coordinates automatically guide the characters to avoid collisions and penetrations.

In the future, we would like to conduct a user study to examine our user interface and seek for better ways to control the characters.

References

1. WWE: Smackdown vs Raw, http://www.smackdownvsraw.com
2. ILOG Inc, CPLEX, http://www.ilog.com
3. Hecker, C., Raabe, B., Enslow, R.W., DeWeese, J., Maynard, J., van Prooijen, K.: Real-time motion retargeting to highly varied user-created morphologies. ACM Trans. Graph. 27(3), 1–11 (2008)
4. Ho, E.S.L., Komura, T.: Wrestle alone: Creating tangled motions of multiple avatars from individually captured motions. In: Proceedings of Pacific Graphics 2007, pp. 427–430 (2007)
5. Ho, E.S.L., Komura, T.: Character motion synthesis by topology coordinates. Computer Graphics Forum (Special issue of Eurographics 2009) 28(2) (2009)
6. Ho, E.S.L., Komura, T.: Indexing and retrieving motions of characters in close contact. IEEE Transactions on Visualization and Computer Graphics 15(3), 481–492 (2009)
7. Jakobsen, T.: Advanced character physics. In: Game Developers Conference Proceedings, pp. 383–401 (2001)

8. Kagami, S., Kanehiro, F., Tamiya, Y., Inaba, M., Inoue, H.: Autobalancer: An online dynamic balance compensation scheme for humanoid robots. In: DonaldK, B.R., Lynch, M., Rus, D. (eds.) Robotics: The Algorithmic Perspective Workshop on Algorithmic Foundations of Robotics, pp. 329–340 (2001)
9. Kulpa, R., Multon, F., Arnaldi, B.: Morphology-independent representation of motions for interactive human-like animation. Computer Graphics Forum 24(3), 343–351 (2005)
10. Lee, J., Lee, K.H.: Precomputing avatar behavior from human motion data. In: Proceedings of 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 79–87 (2004)
11. Lee, J., Shin, S.Y.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of SIGGRAPH 1999, pp. 39–48 (1999)
12. Liu, C.K., Hertzmann, A., Popović, Z.: Composition of complex optimal multi-character motions. In: ACM SIGGRAPH / Eurographics Symposium on Computer Animation, pp. 215–222 (2006)
13. Nakamura, Y., Hanafusa, H.: Inverse kinematics solutions with singularity robustness for robot manipulator control. Journal of Dynamic Systems, Measurement, and Control 108, 163–171 (1986)
14. Phillips, C.B., Zhao, J., Badler, N.I.: Interactive real-time articulated figure manipulation using multiple kinematic constraints. Computer Graphics 24(2), 245–250 (1990)
15. Pohl, W.F.: The self-linking number of a closed space curve. Journal of Mathematics and Mechanics 17, 975–985 (1968)
16. Shin, H., Kovar, L., Gleicher, M.: Physical touch-up of human motions (2003)
17. Shum, H.P.H., Komura, T., Yamazaki, S.: Simulating competitive interactions using singly captured motions. In: Proceedings of ACM Virtual Reality Software Technology 2007, pp. 65–72 (2007)
18. Shum, H.P.H., Komura, T., Yamazaki, S.: Simulating interactions of avatars in high dimensional state space. In: ACM SIGGRAPH Symposium on Interactive 3D Graphics (i3D) 2008, pp. 131–138 (2008)
19. Treuille, A., Lee, Y., Popovic', Z.: Near-optimal character animation with continuous control. ACM Transactions on Graphics 26(3), 7:1–7:7 (2007)
20. Whitney, D.E.: Resolved motion rate control of manipulators and human prostheses. IEEE Transactions on Man-Machine Systems 10, 47–53 (1969)
21. Yamane, K., Nakamura, Y.: Natural motion animation through constraining and deconstraining at will. IEEE Transactions on Visualization and Computer Graphics 9(3), 352–360 (2003)
22. Yamane, K., Nakamura, Y.: Dynamics filter - concept and implementation of on-line motion generator for human figures. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 688–694 (2000)
23. Zhao, J., Badler, N.I.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. ACM Transactions on Graphics 13(4), 313–336 (1994)

Motion Planning and Synthesis of Human-Like Characters in Constrained Environments

Liangjun Zhang, Jia Pan, and Dinesh Manocha

Dept. of Computer Science
University of North Carolina at Chapel Hill
{zlj,panj,dm}@cs.unc.edu
http://gamma.cs.unc.edu/DHM

Abstract. We give an overview of our recent work on generating naturally-looking human motion in constrained environments with multiple obstacles. This includes a whole-body motion planning algorithm for high DOF human-like characters. The planning problem is decomposed into a sequence of low dimensional sub-problems. We use a constrained coordination scheme to solve the sub-problems in an incremental manner and a local path refinement algorithm to compute collision-free paths in tight spaces and satisfy the statically stable constraint on CoM. We also present a hybrid algorithm to generate plausible motion by combing the motion computed by our planner with mocap data. We demonstrate the performance of our algorithm on a 40 DOF human-like character and generate efficient motion strategies for object placement, bending, walking, and lifting in complex environments.

1 Introduction

Human-like characters are widely used in games and other applications including computer animation and virtual environments. One of the challenging problems in this area is to automatically generate natural-looking human-like motion for these characters that also satisfies various geometric and dynamics constraints. Current methods to generate such motion are either limited to the characters moving in open spaces or are tedious and involve considerable input from the artists or animators. As a result, there is considerable interest in developing automated algorithms and tools for motion generation. Besides games and animation, the need for human motion simulation also arises in many other applications, including robotics, biomechanics, and virtual human modeling. In this paper, we give an overview of our recent work on generating naturally-looking human motion that satisfies geometric and kinematic constraints in complex environments [1,2].

One of the main challenges in planning the human motion lies in the high number of DOF (degrees-of-freedom) that are needed to represent the motion of the characters. Even a simple model of the articulated skeleton has more than 30 DOF. Some of the earlier works often used a bounding cylinder to approximate the lower body for planning [3,4]. Recently, sample-based methods have been

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 138–145, 2009.

applied to generate the motion for various talks such as manipulation, reaching, locomotion, and foot step planning [6,7,8,9]. Prior approaches of planning the human motion are mainly limited to relatively simple environments. When planning in constrained environments with a high number of obstacles or tight-spaces, sample-based planner suffer from narrow passage problems. In addition, due to randomness sampling, the motion generated by sampling-based planners is often jerky and unnatural.

On the other side, due to the availability of mocap data, data-driven approaches have been increasingly used in animation and game design for synthesizing character motion [20,21]. Mocap data can be applied to the characters by warping, local modification or optimization to synthesis motion that stratifies kinematic and dynamic constraints. However, these approaches may not be able to handle constrained environment with complex obstacles.

In this paper, we give an overview of our recent work on generating plausible human motion for constrained environments. We first present a whole-body motion planning algorithm for human-like characters. In order to reduce the overall dimensionality of the problem, the planning problem for a human-lie character is decomposed into a sequence of low-dimensional sub-problems. We use a constrained coordination scheme to solve the sub-problems in an incremental manner. In order to satisfy the stable constraints, we present a local path refinement algorithm.

In order to further improve the quality of the motion, we present a hybrid approach that combines the motion computed by our planner with mocap data to compute smooth and plausible paths. We use automated techniques to search a mocap database for plausible motion clips and generate a smooth, blended motion. We highlight the performance on generating motion for different tasks including object placement, object picking, and walking-and-bending.

2 Motion Planning for Human-Like Characters

Planning a collision-free path for a human-like character by taking into account all the DOF is often difficult due to the underlying high dimensionality of the search space. Our algorithm decomposes a human-like character into a set of body parts, i.e. $\{A^0, A^1, ..., A^n\}$. Fig. 1 shows such a decomposition. Based on the decomposition, we reduce the whole-body planning problem to multiple sub-problems of lower dimensions and compute the motion for the body parts in a sequential order.

2.1 Constrained Coordination

In our algorithm, the planning of the path for the k^{th} body part is coordinated with the executing of paths of the first $k-1$ body parts computed earlier. The constrained coordination scheme we propose is based on prior work of priority or incremental coordination for multi-robot planning [10,11]. One feature of our coordination scheme is that when planning for the k^{th} body part, the paths for the first $k-1$ body parts can possibly be updated or refined.

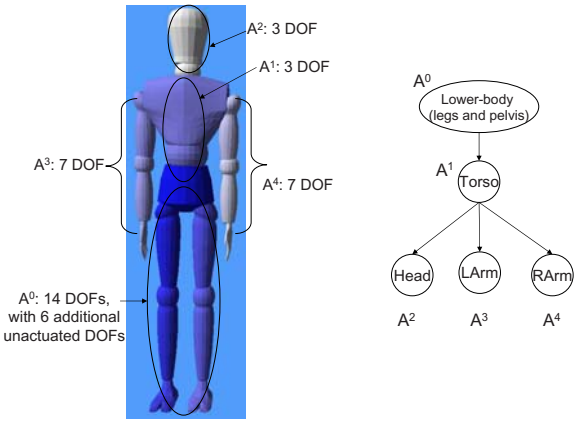


Fig. 1. A human-like character with 40 DOF and one hierarchical decomposition scheme used to plan the motion of this model. Our approach computes the motion for different body parts in a sequential manner by starting from the root of the hierarchy of the decomposition.

We describe the main idea behind constrained coordination by taking into account two objects, A and B. A collision-free path $M^A(t)$ for A is computed by ignoring B. Next, a collision-free path for the system $\{A, B\}$ is computed by coordinating A and B. During the coordination, a *path constraint* for A is imposed so that the configuration of A should lie on the path $M^A(t)$. If B has n DOF, the coordination of the system $\{A, B\}$ is the $n + 1$ dimensional search space, since A is constrained on a one dimensional path with the parameter t. Intuitively, this approach computes a path for B (i.e. $M^B(t)$), based on the original trajectory ($M^A(t)$) computed for A. Such a hard constraint, however, may result in either an inefficient planner or its failure in terms of computing a solution that satisfies all the constraints. This issue can arise when we are attempting to compute a collision-free path in a cluttered environment or in a narrow passage. In order to address this issue, we use a local refinement scheme that modifies the computed trajectory $M^A(t)$, as it computes a collision-free path for B [1].

Our algorithm uses a sample-based planner to compute a path during each stage and we design an implicit local refinement scheme that can be integrated with any sample-based approach. The two main steps of sample-based planning are generating samples in the free space and computing an interpolating motion between those samples using local planning. Instead of explicitly modifying the path computed in the previous stage, our algorithm performs *constrained sampling* and *constrained interpolation* so that the generated samples or interpolations are allowed to move away from the constraining path up to a threshold. In this way, we perform the path refinement step implicitly.

2.2 Planning Stable Motions

In addition to collision-free and joint limit constraints, the motion of human-like characters is also subject to statically or dynamically stable constraints. The computed postures should either be statically stable, i.e. the projection of the center of mass of the character (CoM) lies inside the foot support polygon, or dynamically stable, i.e. the zero moment point (ZMP) lies inside the support polygon [12]. However, due to the computational complexity of simultaneously planning motion that is collision-free and satisfies dynamic constraints, most previous approaches tend to first compute a statically stable and collision-free path and then transform the path into a dynamically stable trajectory [13,14,15]. These two steps are iterated until both types of constraints are satisfied. Our approach can be used to improve the first step. If any sample generated by the constrained coordination algorithm is not statically stable, we modify the sample by using inverse kinematics (IK) so that the CoM at the new sample lies inside the approximate foot support polygon.

3 A Hybrid Method for Human Motion Synthesis

The planner based on randomized sampling described in Section can efficiently compute collision-free paths. However, it is rather hard to generate natural-looking motion. This is due to the following reasons: 1) The randomness of sample-based motion planner can cause jerky and unnatural motion. 2) Human body's complexity requires high-DOF planning, which is difficult due to the underlying complexity. 3) Complex indoor and outdoor environments contain narrow passages or tight spaces, which can be difficult to handle for the sample-based planners.

There is also extensive literature on human motion generation in computer animation. Motion blending [16] and motions graph [17] use operators such as re-sequencing and interpolation to recombine the example motion into new motions. Some recent variants [18,19] use optimization methods to satisfy the constraints like footstep. These methods can create natural long clips with variety of behaviors, but their results are restricted within the linear space spanned by example motions in mocap database. Another disadvantage shared by animation methods is that they can not handle constrained environment with complex obstacles automatically. Usually these methods need artists' manual adjustment to avoid collisions with the obstacles, which is quite tedious.

Since motion planning and motion capture methods are unable to produce natural human motion in constrained environments, the recent trend has been to combine the techniques from both fields to generate more complex and realistic motion. In particular, many authors have proposed using two-stage framework [20,21,7]. In this case, a motion planner first computes the collision-free motion by taking into account few or partial DOF of the human model. Next, in the second stage, some motion data (e.g. mocap) is retargetted by using the planned trajectory as the guiding path with appropriate constraints, e.g. inverse-kinematics. These two-stage methods typically work well in terms of generating

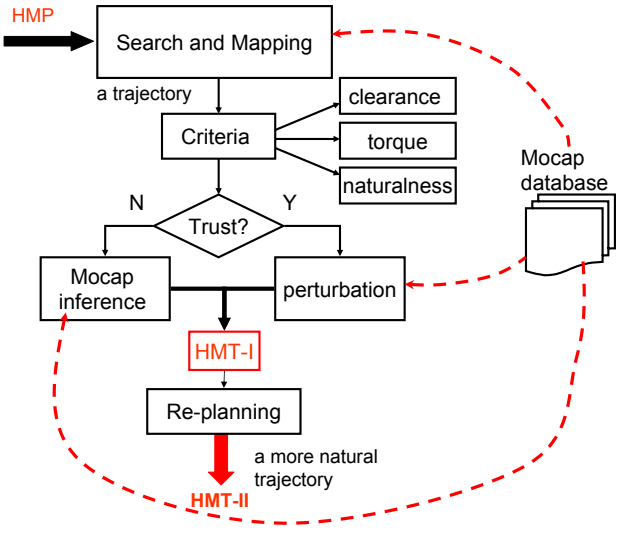


Fig. 2. An overview of our hybrid two-stage algorithm, which tries to improve the smoothness of the planner output and generate a more realistic looking motion

regular motion (e.g. locomotion) in relatively open environments. Some other mocap methods focus on simple motions that can be modeled by step-to-step dynamics model from FSM (Finite-State Machine) [22,23,24] or motions graph [25]. They use these problem-specific models to find an optimized control sequence for a collision-free path. Overall, these methods can generate natural looking motions for certain application such as crowd simulation but are limited to low-DOF models for planning and may not be able to handle constrained spaces with multiple obstacles. Shapiro et al. [26] also describe an elegant approach to combine mocap data with motion planner. The basic idea is to perform local adjustment using a RRT planner whenever the trajectory computed by mocap motion collides with any of the obstacles. However, the resulting approach may not be applicable to all kind of constrained environments. Moreover, performing whole-body path planning using RRT planners can be slow for high-DOF articulated models.

Our formulation is also based on a hybrid approach. It combines the motion planner described in Section 2 with the mocap data to generate collision-free motion that satisfies both the kinematics and dynamic constraints. The overall approach can utilize some of the advantages of both the methods.

Our framework starts with the collision-free path computed by the planner. We first use some criteria to check the quality of each posture and roughly estimate whether the environment around each posture is constrained or not [2]. Based on these criteria, we decide which part of path needs to be replaced or augmented with the motion capture data. Currently, we use three heuristic criteria: space clearance, naturalness of posture and torque of joints. The last two criteria measure the quality of path segments. For path segments of low

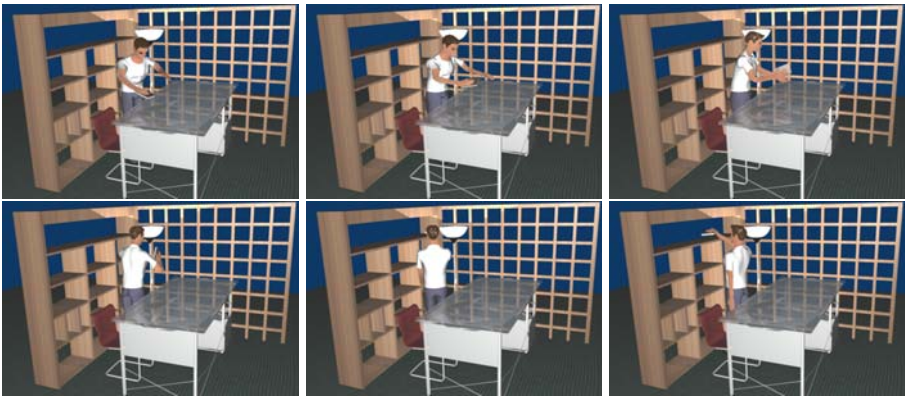


Fig. 3. Object Placement: The human model picks the book from the table and puts it on the bookshelf. This is a constrained environment with multiple obstacles and our algorithm is able to compute a plausible, smooth and collision-free path.

quality and in open spaces, we use the mocap data to generate a possibly better posture. For other segments, we just perform some perturbation of posture to locally improve the quality of the path. Finally, we perform replanning with the optimized path as starting path and obtain a more natural-looking collision-free path. An overview of our hybrid method is shown in Fig. 2.

Fig. 3 and Fig. 4 show the results of our method on some benchmarks within constrained environments. In the first benchmark (Fig. 3), human-like model begins from a knee-bending posture, tries to pick up an object and then puts it on the table. The round edge of table also results in a difficult narrow passage in the configuration space. for the human-like robot. Another aspect that makes the benchmark even more challenging: under the table, the right limb encounters obstacles in all directions except towards the right. The sample-based planner

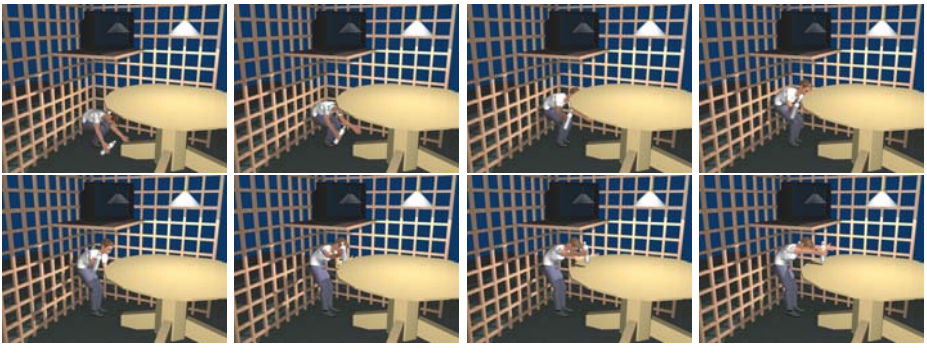


Fig. 4. Object Picking: In this benchmark, the human model stands up and places the object on the table. The overall motion involves use of many DOFs and the resulting path is collision-free.

tends to find the path with minimum RRT extension steps, instead of going through the narrow passages and this causes a unnatural motion with a large rotation.

The second benchmark corresponds to a manipulation task (Fig. 4). In this case, the human holds an object (e.g. a book) and rotates backward to put it on a shelf. The grate, lamp and the bookshelf in the environment result in a tight and constrained environment.

On both benchmarks, our hybrid method can obtain much natural motion compared with pure planner results. These are quite constrained environments.

4 Conclusions

In this paper, we give an overview of our recent work on planning and synthesis of human motion. We use a hierarchical decomposition of a high DOF articulated model and use that decomposition for constrained coordination and satisfy different constraints. Our approach is further combined with searching mocap data to generate plausible motion. We highlight the performance on generating We highlight the performance on generating motion for different tasks including object placement, object picking.

There are many avenues for future work. We would like to compute dynamically stable motions by incrementally enforcing dynamics constraints within our coordination approach. In addition, we would like to apply our approach to more complex scenarios.

Acknowledgement. We would like to thank Will Moss, Ming C. Lin, Jean-Paul Laumond and Katsu Yamane for their helpful discussions. This project was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, ONR Contract N00014-01-1-0496, DARPA/RDECOM Contract N61339-04-C-0043 and Intel.

References

1. Zhang, L., Pan, J., Manocha, D.: Motion planning of human-like robots using constrained coordination. In: IEEE-RAS International Conference on Humanoid Robots (to appear, 2009)
2. Pan, J., Zhang, L., Moss, W., Manocha, D., Lin, M.: A hybrid approach for synthesizing human motion in constrained environments. Technical Report TR09- 002, Department of Computer Science. UNC Chapel Hill (2009)
3. Kuffner, J.: Goal-directed navigation for animated characters using real-time path planning and control. In: Magnenat-Thalmann, N., Thalmann, D. (eds.) CAPTECH 1998. LNCS (LNAI), vol. 1537, pp. 171–186. Springer, Heidelberg (1998)
4. Pettre, J., Laumond, J.P., Simeon, T.: 3d collision avoidance for digital actors locomotion. In: Proceedings of International Conference on Intelligent Robots and Systems, pp. 400–405 (2003)

5. Hauser, K., Bretl, T., Latombe, J.C., Harada, K., Wilcox, B.: Motion planning for legged robots on varied terrain. The International Journal of Robotics Research 27(11-12), 1325–1349 (2008)
6. Koga, Y., Kondo, K., Kuffner, J., Latombe, J.: Planning motions with intentions. In: Glassner, A. (ed.) Proceedings of SIGGRAPH 1994, Orlando, Florida, July 24–29, pp. 395–408 (1994)
7. Choi, M.G., Lee, J., Shin, S.Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Trans. Graph. 22(2) (2003)
8. Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., Kuffner, J.: Bispace planning: Concurrent multi-space exploration. In: Robotics: Science and Systems (June 2008)
9. Drumwright, E., Ng-Thow-Hing, V.: Toward interactive reaching in static environments for humanoid robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China, October 2006, pp. 846–851 (2006)
10. Erdmann, M., Lozano-Perez, T.: On multiple moving objects. In: Proc. of IEEE Conf. on Robot. & Autom., pp. 1419–1424 (1986)
11. Saha, M., Isto, P.: Multi-robot motion planning by incremental coordination. In: Proc. of IROS, pp. 5960–5963 (2006)
12. Vukobratovic, M., Borovac, B.: Zero-moment pointthirty five years of its life. International Journal of Humanoid Robotics 1, 157–173 (2004)
13. Harada, K., Hattori, S., Hirukawa, H., Morisawa, M., Kajita, S., Yoshida, E.: Motion planning for walking pattern generation of humanoid. In: Proceedings of International Conference on Robotics and Automation, pp. 4227–4233 (2007)
14. Kuffner, J., Kagami, S., Nishiwaki, K., Inaba, M., Inoue, H.: Dynamically-stable motion planning for humanoid robots. Autonomous Robots 12(1), 105–118 (2002)
15. Yoshida, E., Belousov, I., Esteves, C., Laumond, J.P.: Humanoid motion planning for dynamic tasks. In: International Conference on Humanoid Robots, pp. 1–6 (2005)
16. Kovar, L., Gleicher, M.: Flexible automatic motion blending with registration curves. In: SCA (2003)
17. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. ACM Trans. Graph. 21(3) (2002)
18. Safonova, A., Hodgins, J.K., Pollard, N.S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. ACM Trans. Graph. 23(3) (2004)
19. Safonova, A., Hodgins, J.K.: Construction and optimal search of interpolated motion graphs. ACM Trans. Graph. 26(3) (2007)
20. Pettré, J., Laumond, J.P., Siméon, T.: A 2-stages locomotion planner for digital actors. In: SCA (2003)
21. Yamane, K., Kuffner, J.J., Hodgins, J.K.: Synthesizing animations of human manipulation tasks. ACM Trans. Graph. 23(3) (2004)
22. Coros, S., Beaudoin, P., Yin, K.K., van de Pann, M.: Synthesis of constrained walking skills. ACM Trans. Graph. 27(5) (2008)
23. Lau, M., Kuffner, J.J.: Behavior planning for character animation. In: SCA (2005)
24. Lau, M., Kuffner, J.J.: Precomputed search trees: planning for interactive goal-driven animation. In: SCA (2006)
25. Treuille, A., Lee, Y., Popović, Z.: Near-optimal character animation with continuous control. ACM Trans. Graph. 26(3) (2007)
26. Shapiro, A., Kallmann, M., Faloutsos, P.: Interactive motion correction and object manipulation. In: I3D (2007)

A Semantic Navigation Model for Video Games

Leonard van Driel and Rafael Bidarra

Delft University of Technology,
Mekelweg 4, 2628CD Delft, Netherlands
leonardvandriel@gmail.com, r.bidarra@ewi.tudelft.nl
http://graphics.tudelft.nl/

Abstract. Navigational performance of artificial intelligence (AI) characters in computer games is gaining an increasingly important role in the perception of their behavior. While recent games successfully solve some complex navigation problems, there is little known or documented on the underlying approaches, often resembling a primitive conglomerate of ad-hoc algorithms for specific situations.

In this paper we develop a generic navigation model which we call semantics-based, because it enables AI to incorporate a wide variety of navigation information into the planning of a character's behavior, including raw geometry, strategic objects, and locomotion abilities. The role of semantics in this domain is highlighted and the presentation of the main components of this semantic navigation model is illustrated with a variety of examples.

The semantic navigation model has been validated and implemented within a navigation system for Unreal Engine 3 that requires little level designer intervention, has a rich interface for AI programmers, and can be extended with other types of semantic information. It is concluded that using this navigation model delivers more natural paths, requires fewer world annotation, and supports dynamic re-planning.

Keywords: navigation, game semantics, game AI.

1 Introduction

Successful improvements in current video games have not been evenly distributed over the development areas involved. So far, the ongoing trend of enhancing visual realism has clearly dominated [2]. In contrast, there are relatively few innovative applications of artificial intelligence (AI) techniques in these games, and mostly we have seen little improvement over the last decades. It is true that a small fraction of video games have shown significant sophistication when compared to the other games at their time. While other game development areas have pushed rapidly forward, using the most state-of-the-art technologies, game AI has been said to lag at least 25 years behind the academic level of research [1]. No wonder, therefore, that in recent years there has been an increasing incentive to help the game industry focus on the challenges of improving game AI [7].

One of the areas in which game AI has been lagging concerns the navigational capabilities of so-called AI- or non-player characters (NPC). As a part

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 146–157, 2009.

of the game AI, navigation is typically concerned with spatial search and path execution (how do you go from A to B?), and it is clearly distinct from the higher-level decision making process (where to go?, why should you go to B?). One of the problems faced by navigation in current games is that it has too often been tightly built up around the A* algorithm, which has stimulated the development of graph-based navigation algorithms and released an abundance of A* variants to run optimally under specific resource constraints. Still A* is only an algorithm and by far not a general solution to navigation [6]. As the requirements on navigation become more and more complex, devising suitable heuristics for A* turns very difficult, leaving us with Dijkstra's algorithm, which isn't but a good exercise on dynamic programming. We believe that although A* or Dijkstra may be good solutions for a large number of isolated problems, they should not be the starting point of the development of a navigation system.

In our view, navigation is about understanding the direct relation between the world and a moving character. This implies a significant shift in focus, from the field of low level search, typically associated with the A* algorithm, to what we call a semantic perspective, in order to capture and understand the environment from the point of view of movement and positioning. In the context of this paper, we define navigation as "the act of guiding a body through space-time". In practical terms, navigation deals with the prediction and execution of motion. In order to predict the outcome of motion, one has to know the navigation space to look for feasible or optimal paths. Without prediction, navigation would be based on reflexes only, which is generally referred to as steering. In order to execute motion, one has to make actual moves based on the planned motion and react to discrepancies between the internal representation and the world.

The main advantages of our semantic navigation model can be summarized as follows:

- it offers less restrictions to the design process, therefore allowing more complex game AI and a potentially richer and more challenging game play;
- it is based on fairly invariant concepts, which are valid through a variety of domains and genres providing better reusability;
- from a game development point of view, it provides a better link between disciplines, whereby both level designers and AI programmers share the same concepts and can therefore enable them to work together more closely.

In this paper, we first outline the basics of our semantic approach (Section 2), proceed with the development of the navigation model (Section 3), and then describe how we have implemented this in a practical setting, together with the results we obtained (Section 4) and end up drawing some conclusions (Section 5).

2 Game Semantics

Semantics is generally defined as the study of meaning and relations among signs. In our present scope, game semantics is concerned with the structure and meaning of game elements, such as level geometry and player characters [9].

2.1 Navigation and the Current Lack of Semantic Structure

When designing game AI or a navigation system, explicit semantics become important because game mechanics needs to be explained, e.g. how should an AI character respond to a constantly changing world [4]. While human players can be very flexible in dealing with game semantics, an artificial player requires a designer that is well aware of this semantics in order to make the AI understand the game. This becomes apparent with some examples that show the link between semantics and daily game development. Here we focus on cover finding, because it is a very important and complex game play element in many modern games.

When a level designer creates the world geometry, he creates floors, walls, obstacles, and ledges based on specific interaction options between players and the environment. For example, a rock can be placed in such a way that it provides excellent cover close to the enemy base, so that an experienced human player will be able to recognize its tactical significance, while an AI character's cover finding algorithm might miss it. Here the understanding of the designer is not shared by the AI because it lacks a structure to contain such information.

Conversely, in that same level setting, we can imagine the level designer marking positions that are suitable for taking cover. This does allow the designer to accurately model the AI behavior for planning covered paths or escaping from a line of fire. However, it also reduces the AI's understanding of a covered area to a set of marks. Here, the designer's understanding of cover is explained to the AI in only a restricted form. This AI will be limited to the designers ability to spot good covered positions. Even worse, such an AI character will be completely exposed if the level geometry is changed afterward.

As a final example, imagine a player and an AI character having to cooperate during a fire fight and, say, both have a proper understanding of being exposed or in cover. At some moment, the latter decides to take cover behind a nearby rock. However, by arrival it fails to take actual cover, because the player was already standing there. The AI does not understand how its teammate's presence conforms to its understanding of cover, and it is now both fully exposed and blocking its teammate's line of fire.

What these examples have in common, is a lack of semantic structure due to a poor distribution of understanding among game development domains like level design, game play, and AI.

2.2 Our Approach

The semantics of game elements deals with the multiple domains of game development and how different building blocks of the game are understood in each domain. Here *understanding* is the relation between meaningful information and the domain where it is interpreted. Because games cope with a limited storage space, we can assume that all internal data is meaningful in some way in some domain. This includes cover markers that are meaningful to the AI, or texture data that is meaningful to the rendering pipeline, but also the game's storyline, which is mostly meaningful to the player, but often not to the AI.

An important element of a semantic model is the *concept*, which groups similar understanding of different data or in different domains. Domains can be level design, game play programming, but also the AI subsystem. For example, players base their understanding on the video output and AI on game data, but they both can have the same understanding of a bridge to cross a river. Here the bridge is the concept of an accessible surface for crossing a river.

Semantic navigation is an approach to navigation with a focus on the semantic model of the navigation AI. The key in creating a semantic navigation model lies in the identification of relevant *concepts* in the game. Concepts allow us to define navigation problems and solutions in a *domain-invariant* way. The latter is very important, because it allows us to design a navigation system without the restrictions of tools, languages and hardware resources.

As an example, we will look at how we define a spatial area over different domains. AI systems in general are very single-position oriented, mostly because a single point in 3D space is a very compact and unambiguous representation. Therefore a path often consists of a sequence of way-points starting at position A and ending at B. Instead, a level designer will rarely think of a single spot, but more of an area that is bound by all kinds of criteria like visibility, distance, and even shape. While the programmer and the designer have different *representations*, they both indicate the same area. This calls for the concept we will later on refer to as *place*. With it, instead of trying to explain each other how they see it, both can issue a place in their own language, while it is up to the implementation of place to combine these different representations.

We propose to define a semantic model based on a set of concepts and representations. A concept is defined by the relations it has with other concepts, like how one concept restricts or modifies another. A concept has a representation for each domain in which it has a relevant meaning. We will refer to the correspondence between representations of one concept in different domains as the *translation* between these representations. It is these translations that make the semantic model consistent, because it is the translation that will eventually make a concept valuable; see a schematic example of this in Figure 1.

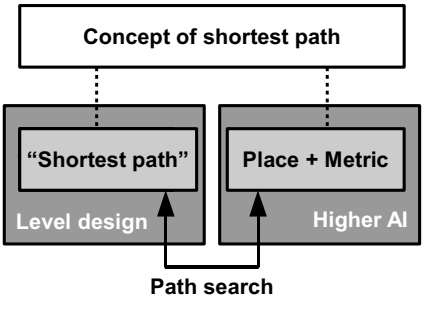


Fig. 1. Schematic representation of the relation between concept (white), domain (dark gray), representation (light gray), and translation (arrow) for "the shortest path"

Imagine, for example, an AI character that wants to move to a covered position because of an enemy approaching. The domains here are the behavior system and a locomotion system. Because the enemy is spotted, the behavior decides to go to a safe location because it wants to maximize chances of survival. The locomotion knows that a set of way-points leading to a position of low visibility is safe. It is the fact that we are able to *translate* safety in the domain of behavior to the domain of locomotion, that makes the concept of safety *valuable*. Generally such a translation exists and will probably require a way-point graph, cover evaluation, and a search algorithm.

Whether a translation can be implemented into a navigation system depends on the design and system limitations. This means that a consistent semantic model does not necessarily result in a feasible implementation model. We will refer to the feasibility or quality of such a translation as the *alignment* of two representations. In the previous example, if both representations of safety are well aligned, the locomotion system will offer the behavior system a path that does maximize the chances of survival.

Defining a general concept in the programming language of a large domain can be a complicated step, but it can be overcome by sub-dividing concepts and domains into more specific concepts for smaller domains. While a more specific concept is less relevant in general, it will have more concrete representations in a set of sub-domains. This approach leads to a concept hierarchy of general and specific concepts, which meets both a wide range of domains and the specific nature of all sub-domains. As we move down the hierarchy, the definition of concepts becomes more implementation specific. This process of exchanging generality for ease of representation, will be referred to as *mixing*. Take, for example, the concept of a path, that an AI character can use to move from one point to another. Instead of trying to directly represent a path, we first introduce the concept of a way-point, which is a position on the path. Using the concept of way-points, we can easily implement a path by moving in a straight line from way-point to way-point. Here the general path concept has been *mixed* with the domain specific use of point locations in programming languages.

The implementation of a navigation system that is based on a semantic model, is in many ways the same as one without it. We still need module interfaces, search algorithms, data models, and so on. However, by implementing the former we also create a powerful language that has been explicitly defined through concepts. Such a language can be used to define complex relations and even new concepts, without having a direct impact on the implementation. We not only define these rules of the game implicitly in programming code, but also explicitly in the semantic model, allowing them to be understood by the AI.

The semantic model provides structure to the implementation in different ways depending on the extensiveness of the concept *hierarchy*. This hierarchy can be used as a basis for the class hierarchy of an object-oriented software design. The representations of concepts can be used to derive data models for our classes. The meaning of a representation can be implemented by the functions

that operate on this data. This allows us to group functionality based on its role in the semantic model and on the domain it provides meaning for.

A part of the implementation will deal with the translation of representations. This will determine the alignment of the data. For example, for good alignment of the path concept, we should either have a static world geometry, or regularly update the navigation grid. Clearly the alignment and the frequency of this updating are related.

3 Semantic Navigation Model

Navigation is a fundamental AI component in the action-adventure games genre. It supports many AI components and plays a principal role in connecting the general AI to the game world. In general, navigation deals with the problem of guiding objects, more specifically finding paths.

Although the area of navigation originates in search algorithms and system design, we choose to approach it from an abstract, conceptual level, before we advance towards its design and implementation level. Because navigation is not a goal by itself, we start by identifying the domains that define the semantics for our model. Next we group all representations in the domain set into concepts that will form the basis of our navigation model.

3.1 Domains

Our design of a semantic navigation model starts by investigating the domains that relate to navigation, because it is within these domains that the initial need for navigation arises. This investigation is based on a simplified view on today's game development process [5] [10]. Our choice of domains covers the key domains relevant to ensure a wide applicability, rather than attempting to fit all development process details into one single model.

We will deal with two classes of domains, namely game development and system components. Within *game development*, there are two domains particularly interested in navigation: the level designer and the AI programmer.

The *level designer* creates the level geometry and places a variety of special objects and markers in this environment. From his perspective, it is important to understand how the design affects the behavior of an AI character, limiting its behavior or creating opportunities. A level designer, therefore, requires a visual representations of concepts that provide feedback during the design process.

The *AI programmer*, in turn, designs the behavior of the AI using a programming language. From his perspective, concepts will mostly be represented by a single name referring to, for example, an object-oriented class. For the AI programmer, it is important that concepts refer to real-life behavior and navigation. The AI programmer will further rely on graphical representations for both debugging purposes and the visual representation used by the level designer.

In the class of *system components*, we distinguish the domains of higher AI and motion control. The *higher AI* is a general decision making component, which

deals with decisions that are unrelated to navigation, like strategy, intuition, and emotion. The higher AI uses implicit descriptions of a path, e.g. by specifying the start and end positions. It is mostly concerned with understanding the link between its goals and such specifications. In order to plan its actions, the higher AI must also understand the cost of trying to reach these goals.

The *motion control* component coordinates a character's movement in terms of animation and physics simulation. Therefore, it has to understand how a route can be traversed based on some explicit definition like a sequence of way-points. While motion control has a subjective role compared to the higher AI, it does take part in the feedback from the world to the higher AI.

3.2 Concepts

The initial process of identifying representations and shaping concepts has been performed in previous research [3], which was geared towards the development of an action-adventure game. Based on that experience, we now derive our semantic model in a top-down fashion, starting at the top of its concept hierarchy with the trivial concept of *navigation*. From there, we look for implementation invariant concepts that describe navigation across all relevant domains.

Path and space. The first sub-division in the concept hierarchy is one rather abstract, as we define the concept of navigation based on the sub-concepts of path and space. A *path* is a way for a character to move about in the game world. As the higher AI formulates goals, it needs a way of expressing them in navigational terms.

The navigation *space* is the collection of all possible paths to navigate, including all traversible surface, open doors, and accessible switches. The explicit definition of this collection is important to test path existence and to quantify path quality. We define a navigation problem as an optimization problem over the navigation space (e.g. the safest path away from danger).

World and ability. The game world is the collection of all game elements, including level geometry, movable objects, and characters. The player is presented with a representation of this world, 'dressed' in pretty textures, particle effects, etc. For the purpose of navigation, one has to 'look through' this dressing and see the elements that matter to navigation, i.e. the navigation space. Although conceptualizing the game world is a simple step, it is important to group all views on this same world into a single concept. The concept of space describes the game world from the perspective of an AI character. This world is primarily created by the level designer, who shapes the geometry and places obstacles and passages, and it is generic in the way it will be used by both human players and AI. It is therefore the perspective of the AI character that filters the world to this concept of space.

We also introduce the concept of ability, from an AI's perspective, defined as a means of a character to change itself and the world for navigation purposes. The most common ability is walking, which changes the position of a character. An

AI character becomes more versatile by adding navigation abilities like jumping, opening doors or pushing objects. While the game world remains the same, an increase of abilities makes the AI perceive a richer world with more paths.

Place and metric. In order for an AI character to move, it has to somehow specify its intentions or desired state, for example the desire to be in a safe location. Although it might not be a specific location that the AI has in mind, there must be some way of anchoring the navigation to the world geometry. Therefore we introduce the concept of place, which is a sub-set of space. A *place* can be a single explicit location, but also a disjoint area.

A navigation solution is always restricted to some part of the environment, for example, related to the current or future position of a character. The place concept defines such a part of the environment, ranging from a single point to a large or scattered area. Using places, we can restrict the solution set by defining where the path begins and ends, and restricting the path as a whole.

The concept of place offers the AI character a large variety of paths and possibilities, through which it can roam freely using its abilities. However, when the AI has a clear goal, it typically searches for a single, optimal path. In order to evaluate places and paths, we introduce the concept of metric. A *metric* is a function that maps a place or path onto a real value. This makes places and paths comparable and allows for a definition of optimality.

In order for the higher AI to specify what navigation can do for its goals, it has to express them in terms of a metric. For example, to find the shortest path, a metric of distance is needed. In practice, these goals require much more complex metrics which measure not only distance, but also cover, visibility, and health change. Such metrics can be achieved by combining several primitive metrics into a single real value, for example using linear combinations or thresholds.

In the domains of level design and AI programming, the visualization of a metric is most important. For a level designer, it is not only important to see the possible paths to navigate, but also which ones will be preferred by the AI, and under which circunstances. For example when placing objects that provide cover, it is important to see how the AI will evaluate these places.

4 Practice

The semantic model presented in the previous section offers a generic approach to navigation in video games, transparently dealing with semantics on various domains. In this section, we develop a further refinement of our semantic model toward an actual implementation based on more specific system requirements, and provide the link between concepts and software. Finally, we briefly evaluate our semantic approach and reflect on the advantages of a semantic model for navigation.

4.1 Refining the Model

Our generic approach allowed us to think of navigation without the restrictions of system design, implementation, and hardware. It is the platform on which video

games run that enforces these restrictions, not games or AI itself. Therefore, in order to progress in the areas of games and AI, it is important to meet these restrictions in the end, instead of starting with them.

Moving from a generic semantic model to a navigation system implementation, we introduce new concepts that gradually mix generic concepts with implementation concepts. Here we respect the domains of game developers and system components, while we express their representations in a programming language. It is important here, not to confuse the system components with the actual implementation.

World and ability. Our characters move by walking or jumping, and it is thus the surface of the level geometry that is of interest to navigation. To represent both the geometry and the topology of this surface in the higher AI domain, we take a cell-decomposition approach to construct a graph of spatial nodes that can cover the curved plane of the level geometry, which is similar to a navigation mesh [8]. Here the graph captures the topology, by defining links between locations where a set of paths can be traversed. Each node is accompanied with a 2D convex area that locally represents the geometry. We will refer to this graph representation as the *mesh.*

To translate between the geometric world representation and the mesh representation, we use collision traces to sample space. First the level geometry is decomposed into 2D areas, and convex sub-areas, called tiles, are traced out. Next the edges adjacent tiles are connected though links, which are also constructed based on collision traces. Because our collision traces are relatively cheap, we were able to implement a real-time translation, which is able to cope with slow but complex changes in the level geometry.

An ability is represented by the higher AI as a set of rules to test a subspace for applicability. For example, walking has rules about the steepness, the floor and the width of the collision cylinder that can be placed on top of the floor. Besides walking, we also introduced the abilities of crouching, jumping, and walking under all angles; see Figure 2. These are all simple variations on the rules of walking.

Abilities are used to identify convex areas and links for the mesh. This is a relation between ability and world, which allows a character specific view on the world. Each application of an ability in a specific location is an instantiation of this ability, to which we will refer as an *action.* An action is represented by an ability and a specification of where and how it is performed. This leads us to another representation of the mesh, namely as a set of related actions. This representation is of interest in the domain of motion control, because it represents all possible moves a character can make in a certain location, assuming the piecewise independence of motion steps.

Place and metric. We restrict the concept of place to the representations of a single location, a mesh element, or a set of places (on which one can perform the operations union, intersection or difference). Although restrictive, this notion is sufficiently rich for navigation while avoiding complex geometric evaluation.

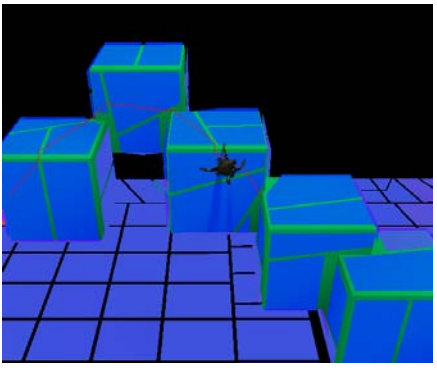


Fig. 2. Screen shot of a spider-like AI character traversing a path (red line) across connected cubes (green). The walkable areas (blue) of the mesh are based on the character's ability to walk under all angles.

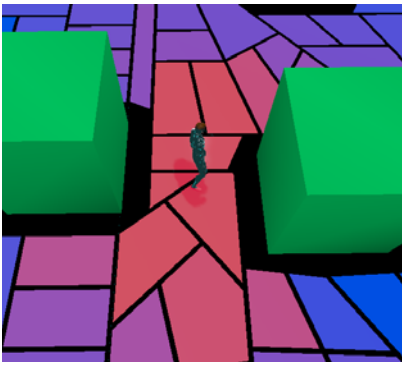


Fig. 3. Screen shot depicting the mesh nodes (red/blue) surrounding two cubes (green). The mesh is colored according to the vision of the AI character (gray) standing in the middle (red is high visibility).

Our approach offers metrics to evaluate e.g. distance, visibility, cover, and also operators on metrics. The distance metric evaluates a link by its Euclidean length and an area by its average diameter. To evaluate the visibility metric, we sample points on a link or in an area, and move a phantom character to the position of these samples to perform a set of ray casts from the eyes of an enemy character to the phantom body. Figure 3 shows a screen shot of the visual representation of the visibility metric.

Based on the visibility metric, we define cover by evaluating local visibility change, e.g. by assuming that positions with good cover are well hidden and close to other high visibility positions. To create more complex metrics, we use the operator metric to combine primitive metrics like distance and visibility. Here we found linear combination and maximum component to be most useful.

Places become a powerful tool for defining paths when we create places based on a metric. For example, an AI character that is under attack needs a covered position to recover. By providing a path that leads to a covered place, we have the AI character move to safety.

Path. The concept of path is the combining element of the navigation system. It is represented by the higher AI as a navigation problem based on a place and metric, and by the motion controller as a solution based on a sequence of actions. The translation from problem to solution is the search for a feasible path, which we approach by performing A* graph search on the mesh graph. The problem is defined by a start and end place, and a metric. The graph search start and end node are determined by evaluating start and end place. The area nodes and link edges weight are based on the metric. Finally a sequence of actions is derived by fitting an action to each area and link.

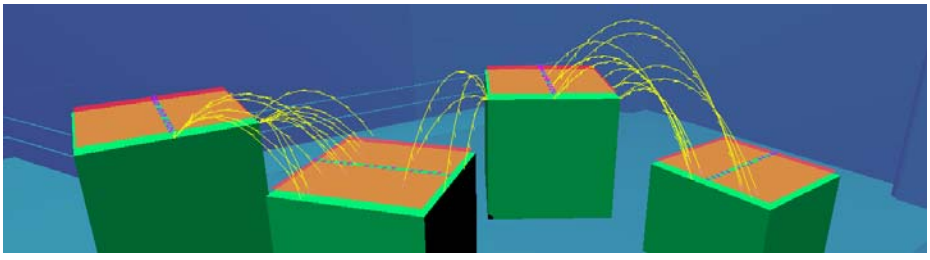


Fig. 4. While placing level components (green), the level designer is made constantly aware of the navigation space. Convex walkable areas (orange) are connected through walk links (purple) and jump links (yellow).

4.2 Results

A prototype navigation system based on our semantic navigation model has been built as an AI sub-system for the Unreal Engine 3. This prototype focuses on navigation for action-adventure games. The AI system in these games has to provide resistance in combat situations and assist in the story line. This includes on-surface navigation in a 3D world with strategic and scripted paths of a high quality. Navigation instructions were generated by a state-based system, and motion instructions were executed by a physics-based motion system.

Previously, designing a game environment did not only require the designer to place geometry, but also to iteratively add annotations for the AI followed by run-time testing of player and AI movement and interaction. By implementing real-time concept translation in our prototype, designers gain insight in the AI's abilities and understanding while shaping the game environment; see Figure 4. This allows for much less annotation and testing during the design phase.

While many concepts are already present in our engine, the explicit definition of these concepts allows a much sharper separation of scripted AI and motion control. The introduction of new concepts and the wrapping of the existing concepts made the AI system modular, so that it can be configured by simply adding and removing concepts, thus improving reusability.

In turn, navigation has been separated from the higher AI, improving robustness and making the higher AI less dependent on the low-level routines that perform the navigation. High-level AI behavior can be designed using high-level concepts that describe a characters behavior, giving programmers a natural understanding of the concepts, and supporting design of higher-level behaviors without being distracted by low-level issues.

5 Conclusion

In this paper we argued that most current navigation subsystems in games are too biased towards very particular applications, their design strongly suffering from contamination by implementation aspects as graph search, terrain specifics,

etc. We have presented a hierarchically structured semantic model that helps defining rich and clear semantics in the design of a navigation system. This model provides a set of basic concepts that apply to generic navigation systems in action-adventure games. This approach leads to better aligned representations in the domains of game development and system components. We believe that the development of game AI for navigation can significantly profit from a semantic navigation model like the one we have described. This semantic navigation model has been validated and implemented within a navigation prototype system for Unreal Engine 3 that requires little level designer intervention, has a rich interface for AI programmers, and can be extended with other types of semantic information. It is concluded that using this navigation model delivers more natural paths, requires fewer world annotation, and supports dynamic re-planning. Our prototype system confirmed that game AI development based on a semantic model is perfectly compatible with the performance requirements related to limited CPU and memory usage.

References

1. Baekkelund, C.: Academic AI Research and Relations with the Games Industry. In: AI Game Programming Wisdom 3, Charles River Media (2006)
2. Buro, M., Furtak, T.: RTS Games as Test-Bed for Real-Time Research. In: Invited Paper at the Workshop on Game AI, pp. 481–484 (2003)
3. van Driel, L.: Semantic navigation in video games. MSc thesis, Delft University of Technology (2008)
4. Heckel, F.W.P., Youngblood, G.M., Hale, D.H.: Influence Points for Tactical Information in Navigation Meshes. In: Proceedings of Fourth International Conference on the Foundations of Digital Games, Port Canaveral, FL, April 26-30, pp. 79–85 (2009)
5. Mesdaghi, S., Stephens, N.: The 2005 Report of the IGDA's Artificial Intelligence Interface Standards Committee. In: IGDA (2005)
6. Reese, B., Stout, B.: Finding a Pathfinder. In: Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Computer Games (1999)
7. Savage, F.: What are the Hard Problems in Game Development? In: Proceedings of the 3rd Annual Academic Days on Game Development in Computer Science Education, Miami, FL, 28 February-3 March (2008)
8. Tozour, P.: Building a Near-Optimal Navigation Mesh. In: AI Game Programming Wisdom, Charles River Media, pp. 171–185 (2002)
9. Tutenel, T., Bidarra, R., Smelik, R.M., de Kraker, K.J.: The role of semantics in games and simulations. ACM Computers in Entertainment 6(4), a57 (2008)
10. Yue, B., de Byl, P.: The state of the art in game AI standardisation. In: CyberGames 2006: Proceedings of the 2006 international conference on Game research and development, pp. 41–46 (2006)

An Open Framework for Developing, Evaluating, and Sharing Steering Algorithms

Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman

University of California, Los Angeles

Abstract. There are very few software frameworks for steering behaviors that are publicly available for developing, evaluating, and sharing steering algorithms. Furthermore, there is no widely accepted methodology for how to evaluate results of agent steering simulations. This situation makes it difficult to identify the real underlying challenges in agent simulations and future research directions to advance the state of the art. With the hope of encouraging community participation to address these issues, we have released *SteerSuite*, a flexible but easy-to-use set of tools, libraries, and test cases for steering behaviors. The software includes enhanced test cases, an improved version of SteerBench, a modular simulation engine, a novel steering algorithm, and more. Care has been taken to make SteerSuite practical and easy-to-use, yet flexible and forward-looking, to challenge researchers and developers to advance the state of the art in steering.

1 Introduction

Steering is an important aspect of behavioral animation that allows autonomous agents to navigate through an environment, and this topic has generated a large amount of research in the fields of robotics, graphics, artificial intelligence, and even sociology and psychology. One of the most time-consuming tasks required for anyone who wants to experiment with steering behaviors is developing the infrastructure surrounding the actual steering algorithm. This includes developing a simulation framework, designing scenarios to test the steering algorithm, deciding the method of evaluating the results, and devising a way to present results

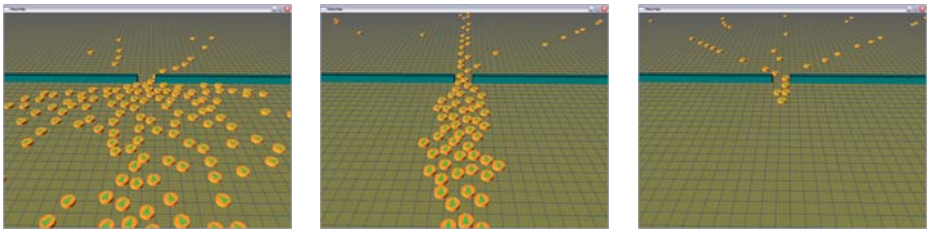


Fig. 1. Agents using the PPR algorithm to steer through the *bottleneck-evacuation* test case, shown here using SteerSim without the user-interface

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 158–169, 2009.

to others. Even in our own lines of steering research, we have come across significant practical hurdles, such as how to run tens of hundreds of simulations in a batch script, how to post-analyze them automatically, how to evaluate whether a steering algorithm is versatile and robust, and other related challenges. These tasks take weeks, even months of effort to do properly.

Over the past year, our research has accumulated into one such infrastructure. We proposed SteerBench [1], which explored the possibility of scoring the agents steering through a variety of challenging test cases. Recognizing that benchmark scores and detailed numerical metrics are not always enough, we also recently developed SteerBug [2], which uses pattern recognition techniques for recognizing user-specified behaviors of interest. Additionally, we have experimented with our own novel steering techniques: egocentric affordance fields [3] and the PPR algorithm (presented in this paper), and during the research process we developed a flexible simulation tool that provides common functionality to both algorithms.

We call the resulting framework *SteerSuite*. The source code and content are publicly available for download [4]. SteerSuite includes:

- Many diverse and challenging test cases with an open specification for creating more test cases,
- SteerBench, including several improvements,
- SteerSim, a simulation engine,
- The PPR steering algorithm,
- SteerLib, which includes functionality to make it easy to read the test case files, to record/replay agent simulations, and much more.

We chose to release this software for the following reasons:

- To make the implementation of our research works available for scrutiny and for use by others
- To propose a set of test cases as a starting point for the community to eventually create a standard suite of tests for steering algorithms
- To make it easy for users to start developing and testing their own steering experiments
- To make it easy to share results, in the form of benchmark/metrics reports and also in the form of recordings of the simulations.

To our knowledge, the only other openly available steering framework is OpenSteer [5] by Craig Reynolds. Our simulation engine, SteerSim, is inspired by the OpenSteerDemo component of Reynolds' software, however, beyond this similarity, both softwares are complementary to each other. OpenSteer provides a library of functions used for steering decisions of agents, including path following, seek, flee, and boids behaviors, and helper functions to determine the agent's state, while SteerSuite provides functionality related to the testing, evaluating, recording, and infrastructure surrounding a steering algorithm.

This paper discusses the novel aspects of SteerSuite. We first discuss the improvements made since the original SteerBench: improvements to the set of test cases are discussed in Section 2, and the generalizations made to benchmarking are

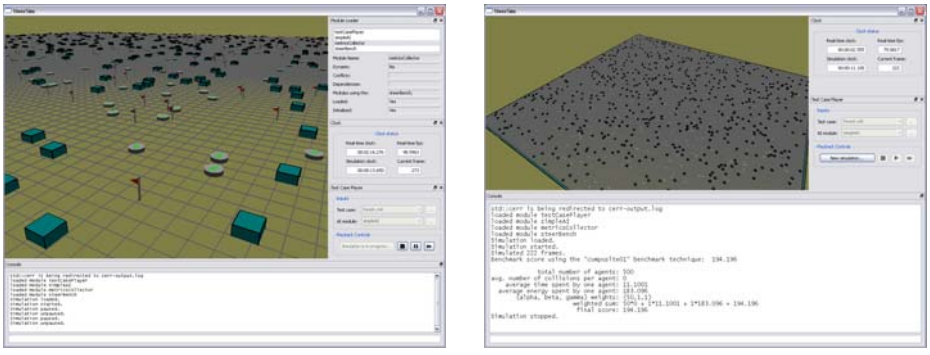


Fig. 2. Screenshots of the SteerSim user interface, benchmarking a simple steering algorithm on the *forest* test case

discussed in Section 3. Then, Section 4 describes the example steering algorithm provided with SteerSuite, called PPR (Plan, Predict, and React). The development and testing of this algorithm illustrates the various features of SteerSuite. We discuss debugging and evaluation in Section 5, and Section 6 concludes.

2 Improvements to Test Case Specifications

In the context of SteerSuite and SteerBench, a *test case* is a description of the initial conditions of agents and objects in an environment. The original set of test cases is described in [1], which focuses on testing normal everyday pedestrian behaviors. We hope to encourage the community to contribute more test cases for different application domains, which can eventually evolve into a widely accepted set of tests that steering algorithms are expected to use.

To this end, the test case format is designed to be flexible but easy to use. For portability, we migrated the test case format into XML. SteerSuite provides an XML Schema that describes the specifics of the format, so that users can easily create and validate their own test cases. We added the ability for test cases to specify "regions of agents" and "regions of obstacles," where the agents or obstacles are randomly placed; this feature was previously hard-coded into only a few test cases. SteerSuite also provides a library that can read these test cases and automatically set up all initial conditions, deterministically resolving all random regions and random targets before giving the initial conditions to the user.

We also added more elaborate goal specifications in the test cases. An agent's goal can be one or a combination of the following types:

- **Seek static location:** the agent should navigate towards a fixed location in space.
- **Flee static location:** the agent should move away from a fixed location in space. For example, agents should flee from a stationary vehicle that is on fire.

- **Seek dynamic target:** the agent should steer towards a moving target. The two common examples of this are (1) pursuit, where one agent chases another agent, and (2) meeting, where two friends want to steer towards each other.
- **Flee dynamic target:** the agent should flee a moving target, for example, when being chased.
- **Flow in a fixed direction:** the agent should progress in a particular direction, for example, when going down a hallway with no need to follow an exact planned path.
- **Flow in a dynamic direction:** the agent should follow a dynamically changing direction, which can be used for agents to advect along a potential field or velocity field, or to wander with a randomly changing direction.

Each type of goal takes additional data, such as the point location, named target, or direction vector. This additional data can optionally be declared as "random". In future work we may add support for more controlled randomness, allowing the user to specify sampling distributions and regions.

This goal specification deserves mention because it addresses an important practical consideration: a steering algorithm is only one of many components of an autonomous virtual character, and eventually it will be necessary to interface the steering algorithm with a high-level intelligent controller (i.e. the artificial intelligence of the agent). The described goal specification is our first attempt to characterize all possible ways that artificial intelligence may want to interface with steering. If the goal specification receives positive feedback, we will develop more test cases that use these new goal types, with an emphasis on normal everyday steering tasks where real human pedestrians may think in terms of these goal types instead of steering to a static target.

3 Improvements to Benchmarking

The original SteerBench benchmark process was a monolithic process that collected numerous metrics of an agent simulation and then computed a weighted sum of three primary metrics for the final benchmark score. For more information about these metrics, refer to the original SteerBench work [1]. In SteerSuite, we have generalized this process by separating the concepts of metrics collection and benchmark techniques. This separation allows users complete flexibility; users can easily experiment with their own benchmark techniques regardless of what metrics are used, or they can focus on using metrics to debug or analyze a simulation.

The metrics are updated once per frame, and can be accessed and examined by the user for any agent at any frame of the simulation. The desired benchmark technique is also updated once per frame, given access to the metrics of all agents and of the environment. The benchmark technique then provides functionality to (1) get a simple score or (2) to output details of how the score was computed, (a) for all agents in a test case or (b) for an individual agent. Benchmarking can be done on-line while running the steering algorithm, on-line with a recording of the simulation being replayed, or off-line with a command-line tool.

SteerSuite provides several basic benchmark techniques. The original SteerBench work, which used a weighted sum of three primary metrics, is called the `composite01` benchmark technique. We also developed a `composite02` technique which uses four primary metrics: the first three metrics are the original three metrics from `composite01`: (1) number of collisions, (2) time efficiency measured as seconds to reach goal, and (3) sum total of kinetic energy samples along the path (which was called effort efficiency). For `composite02`, we add (4) a sum total of acceleration samples along the path, another measure of effort efficiency.

4 PPR: The SteerSuite Steering Algorithm

In this section we describe the PPR (Plan, Predict, and React) algorithm, which is currently the main example algorithm provided with SteerSuite. As the algorithm is described, it illustrates how various features of SteerSuite can be used.

The algorithm is implemented as a plugin to SteerSim, the simulation engine that is part of SteerSuite. SteerSim has a modular architecture, so that almost all useful functionality is provided in modules. Modules have access to most of the simulation engine's data, including a spatial database, clock, camera, access to other modules. Modules can even add components to the graphical user interface. When a simulation is started, the engine uses the PPR module to create and initialize each agent, providing each agent its initial conditions (including a sequence of goals) that came from a test case. As the simulation runs, the engine automatically updates every agent. Modules have the opportunity to perform preprocessing and postprocessing at every frame, which is useful for metrics collection or for a steering algorithm that requires a global processing stage, but the PPR steering algorithm does not use this feature.

The PPR algorithm is a novel rule-based pedestrian steering algorithm that combines three (potentially conflicting) aspects of human steering into a single steering decision. The three aspects are:

- Plan: The agent selects a local target that is used to smoothly steer along the planned path.
- Predict: The agent makes predictions about other agents and determines how to steer to avoid the most imminent predicted collision.
- React: The agent steers to avoid problems interacting with other agents in its immediate surroundings.

All three aspects produce a steering decision, and the challenge of this approach is how to combine these steering decisions, or at least how to choose which steering decision to use at any given time. We address this by using a state machine and a set of rules. The implementation is divided into six main phases, described below. (There are actually more phases and states of the agent that are part of future research.)

Long-term planning phase. Given a goal target, the agent plans a path to its goal using the standard A-star algorithm [6], only planning around static obstacles in the scenario. The graph used by A-star is a rectangular grid where each

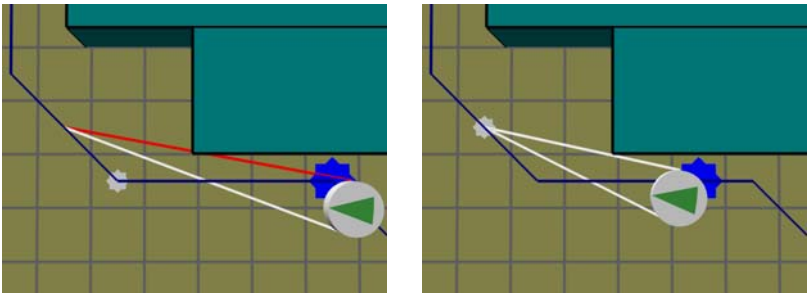


Fig. 3. Short-term planning. The local target (white star) is chosen as the furthest point such that all path nodes between the agent's closest path node (blue star) and the local target have line-of-sight to the agent.

node is connected to its eight neighbors. This type of graph is chosen because the spatial database provided by SteerSuite is a grid, and it allows traversal costs to be associated with each grid cell. A grid-based graph can result in costly A-star searches, but this choice avoids the need for users to manually create A-star graphs, and the amortized cost of path planning remains very low.

Short-term planning phase. Given the planned path, the agent chooses a local target along the path to steer towards. This local target is chosen as the furthest point along the path such that all path nodes between the agent and the local target are visible to the agent (Figure 3). This criterion smooths the agent's path while enforcing the agent to follow the path correctly around obstacles. This, combined with path planning described above, is all the agent needs to steer correctly around fixed obstacles.

To implement the short-term plan requires visibility testing, which can be done with ray tracing. The spatial database in SteerSuite provides fast ray tracing support for this purpose; the grid data structure allows us to greatly reduce the number of objects that need to be tested for ray-intersection. Users can add support for arbitrary objects in the spatial database by implementing a few geometric helper functions, including a ray-intersection routine.

Perception phase. To perform natural predictions and reactions, it is important to model what the agent actually sees. We model an agent's visual field as a 10 meter hemisphere centered around the agent's forward facing direction. The SteerSuite spatial database makes it possible to perform range queries in the spatial database to collect a list of these objects. Furthermore, objects that do not have line-of-sight are not added to the list of objects the agent sees.

Prediction phase. The agent predicts possible collisions, only with agents in its visual field, using a linear space-time predictor based on [7]. Given an agent's position, P, velocity V, and radius r, our linear predictor estimates the agent's position at time t as

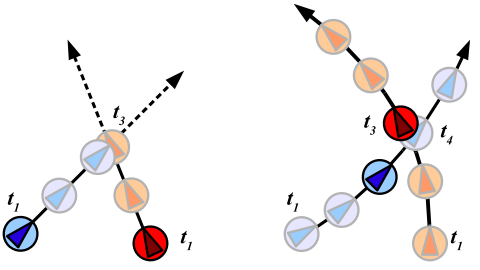


Fig. 4. Space-time prediction. Left: agents predict a collision, knowing their trajectories will overlap at the same time, t_3. Right: agents steer to avoid the collision. Note that space-time prediction correctly avoids a false prediction between the blue agent at t_4 and the red agent at t_3, because they reach that point at different times.

$$\text{Agent's future position} = P + t \cdot V. \tag{1}$$

A collision between agent a and agent b would occur at time t if the distance between their predicted positions becomes less than the sum of their radii:

$$\|(P_a + t \cdot V_a) - (P_b + t \cdot V_b)\| < r_a + r_b. \tag{2}$$

Solving this expression for time t results in a quadratic equation. The agents collide only if there are two real roots, and these two roots represent the exact time interval of the expected collision.

If collisions are predicted, they are handled similar to [8]. Each predicted threat is classified as one of three possible types: oncoming, crossing, or similar direction collisions. For oncoming threats, agents will both choose to steer to the right, or to the left, depending on which side is closer. For crossing threats, the agents first determine who would reach the intersection first. The agent that will reach the collision first will decide to speed up and turn slightly outward, and the agent that will reach the collision later will slow down and turn slightly inward (Figure 4). This behavior is very subtle, but the difference between using these predictions or disabling the predictions is very clear.

The prediction phase also updates a state machine that decides how to steer. SteerSuite provides a useful state machine helper object for this purpose. The possible states and corresponding steering actions are described in Figure 5. In most cases, if an agent needs to react to something more immediate, it will override the predictive steering decision.

Reaction phase. The reaction phase implements both reactive steering and the rules to decide which steering decision to use, and outputs a steering command to the locomotion phase. The agent traces three forward-facing rays, 1 meter to the front of the agent, and 0.1 meters to the side. If these rays intersect anything, the agent may need to react, possibly overriding steering decisions made by prediction. When reacting, the agent takes into account the relative location and orientation of the new obstructions. This results in a very long list of rules that account for

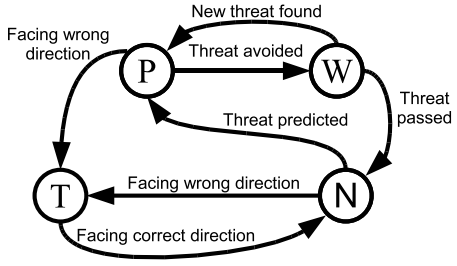


Fig. 5. State machine used to integrate plan, prediction, and reaction behaviors. The prediction updates the state machine, and the reaction phase uses the current state to help choose the final steering decision. The agent steers normally in state N, proactively avoids threats in state P, waits for avoided threats to pass in state W, and re-orients itself towards the short-term-planned target in state T. These behaviors may be overridden by reactions.

all possible configurations: there can be up to three perceived obstructions (one per forward-facing ray), each obstruction may be an agent or an obstacle, and any obstructing agents can be classified as oncoming, crossing, or facing the same direction. For efficiency, the rules are implemented using a hierarchy of conditions instead of checking each rule one after the next. This way, identifying any rule requires only a (informally) logarithmic number of conditional branches instead of linear. The top level of the decision hierarchy is illustrated in Figure 6. Once the exact rule has been identified, the agent outputs a "steering command" to the locomotion phase. As with the previous phases, this phase benefits from SteerSuite's ray tracing and geometry helper functionality.

Locomotion phase. The locomotion phase receives an abstract steering command that tells an agent to turn, accelerate, aim for target speed, and/or scoot left or right. This command is converted into a force and angular velocity that moves the agent's position and orientation using simple forward Euler integration. Note that, even though we use dynamics to move the agent, the locomotion phase constrains the output to model what pedestrians could realistically do. Most likely, in a future version of SteerSuite, we will generalize the locomotion phase into a few SteerSuite helper functions that will be available to any steering algorithm.

5 Debugging and Evaluating an Algorithm

The process of debugging and evaluating an algorithm is where SteerSuite features really become useful. Like OpenSteer, the simulation engine allows the user to visualize the simulation, draw annotations, and step through the simulation manually. This is already very useful, allowing users to annotate what the agents are thinking. Here we describe additional debugging and evaluation features of SteerSuite.

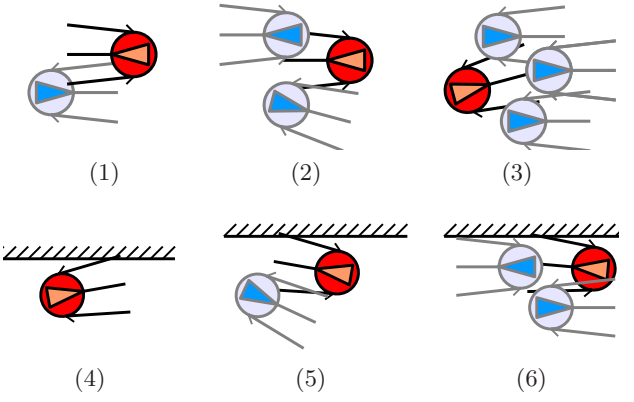


Fig. 6. Main cases of the reaction phase, determined by three short forward-facing rays traced for each agent: (1) one agent, (2) two agents, (3) three agents, (4) obstacles only, (5) one agent and obstacles, (6) two agents and an obstacle. Each case has many sub-cases depending on the position, orientation, and state of the other agents and obstacles.

Test cases. The test cases have proven to be the most important aspect of debugging and evaluating, for us as well as other users of SteerSuite. With these test cases, it is possible to test the full spectrum of expected steering behaviors for a given application. The test cases are also crucial for presenting results; when it is impractical to demonstrate the wide range of results in a presentation or paper, instead it is possible to summarize the results of an algorithm based on existing test cases.

Simulation recordings. SteerSuite further provides the ability to record simulations, using SteerLib in the user's code or using SteerSim directly. We have found this feature to be invaluable. It is often easier to visualize recordings of large simulations in real-time, instead of watching a slow simulation while it is running. At the same time, recordings retain more flexibility than a pre-recorded movie, allowing the user to move and zoom the camera while interactively visualizing the replay. In combination with command-line tools to perform batches of simulations and batches of benchmark analysis, we sometimes record hundreds of simulation experiments in mere minutes, examine the data, and then narrow down which simulations to inspect visually and to re-simulate. SteerSim can also run on an architecture simulator, which has only a command-line interface, and recordings allows us to later verify that the command-line simulation worked properly. Eventually we hope that these recordings can become a common way to report results, with two main benefits: (1) users easily can provide a large amount of results for others to see, and (2) users can benchmark other people's simulations with their own downloaded version of SteerSuite, so they can trust that the metrics and scores were not altered.

Benchmark scores. The ability to simplify an algorithm's performance into a single number has been useful when we try to compare a large number of simulations. While there is very little information in the single number, it still helps narrow down which simulations should receive closer attention – for example, while debugging PPR, we searched for the simulation with the "worst" benchmark score and then examined that simulation. One limitation is that some benchmark techniques cannot be used to compare scores across test cases. After becoming familiar with the typical range of scores for each test case, this limitation is not very significant, and other benchmark techniques do not have this limitation in the first place.

Detailed metrics. There are several positive experiences we had with detailed metrics (refer to the original SteerBench paper [1] to see a list of these metrics) while developing the PPR and egocentric affordance fields algorithms. For example:

- On several occasions during development, we caught instances where agents were oscillating unexpectedly, according to the "number of times the angular velocity changed sign" and "number of times acceleration flipped direction" metrics, but the oscillations were barely visible in the simulation. It turned out these oscillations were a result of the agent repeatedly switching between two steering decisions, and without the metrics we would not have caught these errors.
- At one point, in some scenarios such as bottleneck-squeeze, we saw a surprising number of collisions, which did not seem true from the visualization. Upon closer examination, it turned out that the agents were "scooting" (side-to-side motion) into the wall as they were walking along the wall. This was technically not an oscillation of turning or of velocity, so it was useful that we verified some obvious metrics to find this error.

Finally, we encourage interested readers to download the SteerSuite software and explore the benefits and limitations of the PPR algorithm by seeing how it performs on the test cases. The PPR algorithm is not intended to robustly solve all test cases, but rather to be a starting point for users to start using SteerSuite. There is also a large and growing number of crowd simulation and agent steering approaches in existing literature (e.g [7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23] are just a few); we encourage developers to port or implement any existing steering techniques that they find interesting, reporting their experiences with these algorithms in the form of recordings and benchmark results of our test cases.

6 Conclusion and Future Work

In this paper we described the novel aspects of SteerSuite, a publicly available software framework for developing, testing, and sharing steering algorithms. The ideas in SteerSuite are just one proposed way to answer many interesting questions: how can we evaluate steering behaviors? What types of tests should we require every

steering algorithm to pass? How should a steering algorithm be interfaced with higher-level artificial intelligence in a practical application? We hope that SteerSuite will generate interest in these questions within the community.

A major goal of SteerSuite is to make infrastructure tasks very easy for users. If such tools become widely accepted, whether it is SteerSuite or some other future software, the research community will be able to communicate and share results more easily, thus promoting more rigorous evaluation of steering algorithms, and ultimately helping to push forward the state of the art in steering.

We also discussed the PPR steering algorithm, which demonstrates a versatile set of behaviors, and along the way it also showcases how various parts of SteerSuite can be used. In the future we plan to integrate the egocentric affordance fields algorithm into SteerSuite as another example steering algorithm, and we plan to migrate the SteerBug framework [2] into SteerSuite as well.

Acknowledgements

This work was partially supported by the NSF grant CCF-0429983. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF. Parts of the user interface were contributed by Brian Hou and Tingyu Thomas Lin. We would like to thank Craig Reynolds and Ronan Boulic for interesting discussions and suggestions. We would also like to thank Intel Corp. and Microsoft Corp. for their generous support through equipment and software grants.

References

1. Singh, S., Kapadia, M., Naik, M., Reinman, G., Faloutsos, P.: SteerBench: A Steering Framework for Evaluating Steering Behaviors. Computer Animation and Virtual Worlds (2009)
2. Kapadia, M., Singh, S., Allen, B., Reinman, G., Faloutsos, P.: SteerBug: An Interactive Framework for Specifying and Detecting Steering Behaviors. In: ACM Siggraph/Eurographics Symposium on Computer Animation, SCA (2009)
3. Kapadia, M., Singh, S., Hewlett, W., Faloutsos, P.: Egocentric affordance fields in pedestrian steering. In: I3D 2009: Proceedings of the 2009 symposium on Interactive 3D graphics and games 2009, pp. 215–223 (2009)
4. Singh, S., Kapadia, M., Reinman, G., Faloutsos, P.: Steersuite, http://www.magix.ucla.edu/steersuite/
5. Reynolds, C.: http://opensteer.sourceforge.net/
6. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
7. Paris, S., Pettre, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. In: EUROGRAPHICS 2007, pp. 665–674 (2007)
8. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: SCA 2005: Proc. of the 2005 ACM SIGGRAPH/Eurographics symp. on Computer animation, pp. 19–28 (2005)

9. Brogan, D.C., Hodgins, J.K.: Group behaviors for systems with significant dynamics. Auton. Robots 4(1), 137–153 (1997)
10. Goldenstein, S., et al.: Scalable nonlinear dynamical systems for agent steering and crowd simulation. Computers and Graphics 25(6), 983–998 (2001)
11. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Papers, pp. 1160–1168 (2006)
12. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature 407, 487 (2000)
13. Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. Computer Graphics Forum 23, 509–518 (2004)
14. Loscos, C., Marchal, D., Meyer, A.: Intuitive crowd behaviour in dense urban environments using local laws. In: TPCG 2003: Proceedings of the Theory and Practice of Computer Graphics 2003, p. 122. IEEE Computer Society, Los Alamitos (2003)
15. Reynolds, C.: Steering behaviors for autonomous characters. In: Game Developers Conference (1999)
16. Rudomín, I., Millán, E., Hernández, B.: Fragment shaders for agent animation using finite state machines. Simulation Modelling Practice and Theory 13(8), 741–751 (2005)
17. Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: SCA 2007: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 99–108 (2007)
18. Boulic, R.: Relaxed steering towards oriented region goals. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, pp. 176–187. Springer, Heidelberg (2008)
19. Metoyer, R.A., Hodgins, J.K.: Reactive pedestrian path following from examples. The Visual Computer 20(10), 635–649 (2004)
20. Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., Manocha, D.: Real-time navigation of independent agents using adaptive roadmaps. In: VRST 2007: Proceedings of the 2007 ACM symposium on Virtual reality software and technology, pp. 99–106. ACM, New York (2007)
21. van den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of multiple agents in crowded environments. In: SI3D 2008: Proceedings of the 2008 symposium on Interactive 3D graphics and games, pp. 139–147 (2008)
22. Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: SCA 2007: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 109–118 (2007)
23. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphics Forum 26(3), 655–664 (2007)

Data Based Steering of Virtual Human Using a Velocity-Space Approach

Yijiang Zhang[1,2], Julien Pettré[1], Qunsheng Peng[2], and Stéphane Donikian[1]

[1] Bunraku team, INRIA Rennes - Bretagne Atlantique
Campus de Beaulieu, 35042 Rennes, France
julien.pettre@irisa.fr,
yijiang.zhang@irisa.fr
[2] CAD&CG Lab, Zhejiang University
Hangzhou, China

Abstract. In this paper, we analyze the habits of human walk. By plotting and studying some real experimental motion-captured data, we estimate the bounds of the velocity and acceleration of the pedestrian and achieve the admissible motion set. We introduce the velocity-space which means the set of the whole admissible velocities for one step in the procession of human walk. In addition, a model describing the dependence of the next movement on the previous one is designed. It provides an interior constraint for pedestrian in steering model. With two examples, we show the adaptability and efficiency of the model.

1 Introduction

Steering model of pedestrian is an essential issue with strong connection to research in the fields of computer animation, computer graphics and artificial intelligence. The applications in video games and virtual walk-through raised the level of expectation for the theory. Recently, diverse steering models of pedestrians have been emerging.

Human's bipedalism distinguishes it from general vehicle model. And the steering of pedestrian is dependent on this particular mechanism. As a result, it is reasonable to study the locomotion of pedestrian before the steering modeling. The admissible space of velocity and continuity of motion are the original topic of this paper.

As our contribution, we follow the fact that pedestrian walks step by step and analyze the motion with step as a principal element. First, subdividing the trajectory data along time by steps, we learn bounds of pedestrian velocity and acceleration. We then propose a pedestrian locomotion model, considering the bounds of velocity and continuity of motion. Last, through examples, we demonstrate that the model can be generally applied in different conditions. It could ensure the artificial trajectory plausible without considering the motion data in details.

The paper is structured as follow: in Section 2, we review prior work in autonomous steering behaviors model. Section 3 describes the procedure for

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 170–181, 2009.

acquiring experimental data. In Section 4, some properties about pedestrian locomotion are deduced and a model of Optimal Velocity Change is proposed based on these properties. Several applications of the model are given in Section 5. We draw the conclusions in Section 6 along with guidelines for future works.

2 Related Work

2.1 Locomotion Model

In a steering behavior model, locomotion represents a character's inherent movement ability. The steering motion is subject to constraints imposed by the character's physical ability. Due to the development of the steering model, more and more locomotion models emerge. The simple vehicle was adopted by Reynolds et al. [1] to simulate typical steering behaviors for simple autonomous creatures in an on-line manner, where the objects were abstracted as oriented particles with their own dynamics. Go et al. [2] attempted to pre-integrate a collection of representative local trajectories offline as locomotion samples for effective use at runtime. Kwon and Shin [3] analyzed example motion data to extract some properties of human locomotion, i.e. speed and acceleration bounds, as preprocessing. G. Arechavaleta et al. [4] proposed that human locomotion can be approximated by the motion of a nonholonomic system, which introduces the mathematical background on nonholonomic systems into the researches performed in the human physiology. In [5], it is found that the trajectory data always supports a close relationship between instantaneous velocity and curvature.

The discrete version of the locomotion is another important part. Metoyer et al. [6] and Antonini et al. [7] proposed discrete steering model, by which the physical space and the agent's motion are modeled as a dynamic and individual-based spatial discretization. Another example is the cellular automaton(CA) model [8]. In this case the local movements of the pedestrian are modeled with a matrix of preferences which contains the probabilities for a move, related to the preferred walking direction and speed, toward the adjacent directions. Meanwhile, Treuille et al. [9] processed the human motion capture data based on single walk cycles. In [9], the transition cost between two contiguous steps is computed directly by physical difference and motion direction, while in our work, the transition cost derives from some inherent properties of human walking.

2.2 Behavioural Model

Important behavioral models in the context of pedestrian movements include a target approach model, a route choice model, and a collision avoidance model. One of the most popular approaches is Helbing's social forces model [10] where an individual is subject to long-ranged forces and the dynamics follow the equation of motion, similar to Newtonian mechanics. Reynolds et al. [1] adopted a simple vector quantity, i.e. a desired steering force, as the motion control signals. Fajen

et al. [11] investigated the dynamics of steering and obstacle avoidance by experiment to reproduce the pattern of routes through a simple scene. An approach was explored in [6] for generating reactive path based on the user's examples of the desired behaviour. Treuille et al. [9] applied a global optimal method to predict the best reaction movement for each interaction.

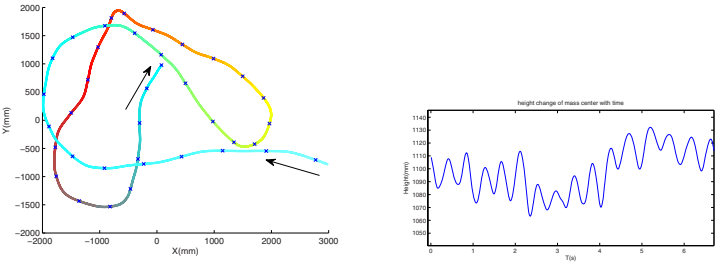
3 Data Set

To study human's natural movements during free walk, the first work is to acquire experimental data. We use the motion capture technology to record the trajectories of body movements. Subjects are asked to walk freely where there are no destinations or obstacles on the ground, and they can change their speed and orientation as they want. Because we focus on the study of human's regular walk, subjects are not permitted to walk backward, spin or run etc..

When we get the trajectories of body movement, considering the different physical condition for each participant, we need normalize the trajectory at first. The motion data are scaled to set the length of legs 1.0 meter. We use the average position of the markers fixed on both sides of the pelvis to define the position of the pedestrian, denoted as P. The 2D trajectory of the marker is shown in Figure 1(a). Meanwhile the height changing with time is displayed in Figure 1(b), which shows that in the procession of walking, the mass center of pedestrian is up and down periodically. The trajectory reaches its local minimum height when both feet are contacting the ground. All of these local minimum frames are collected as the samples of the complete trajectory. After segmenting the trajectory, the height component of the trajectory will be eliminated. We use the horizontal component information from all these samples to get a series of discrete motion states of the pedestrian. The difference of the contiguous samples is the displacement for one step. To We can estimate the velocity between two successive samples, denoted as V, as the ratio of displacement to duration time. From the data, we find the duration of each step is almost constant, the mean of the durations is 0.56519 second and the standard deviation is 0.11587. So we will set the duration for each step Δt as 0.56 approximately in the application.

4 Steering Method

Pedestrian locomotion is constrained by physical and biomechanical factors of human body. For example, pedestrian cannot walk at a high speed or acceleration. Moreover, pedestrian locomotion is dependent on the habit of human walking, for example, pedestrian would not like to walk too slowly or turn too fast. Considering that the pedestrian walks step by step, a discrete model based on step is a better alternative, which is described as above. In this section, we will investigate properties of pedestrian walk based on motion captured data. First, we study the range of the whole admissible velocities. Then, the speed change and orientation change between successive steps are detected and the boundaries



(a) trajectory example. (b) height change with time

Fig. 1. (a) A 2D visual trajectory of one trial: The colorful curve is the projected 2D trajectory of a subject's mass center; the arrows represent the orientation of movement; the blue crosses are the positions when both feet of the subject contact the floor. (b) Change of mass center height with time: The trajectory reaches its local minimum height when both feet are contacting to the floor.

are approximated by simple curves. Last, a model of optimal velocity change is proposed.

4.1 Admissible Velocities

In the foregoing section, the trajectory has been sampled with the local minimum frames. For each sample frame, expect the first and last one, two velocities are related to it, that are the previous velocity V_i and next velocity V_o. Given V_i, V_o can be transformed into the form (O, L), where O and L are respectively the angle between V_i and V_o, in degree, and the norm of V_o. We define the form (O, L) as a *motion state* for one step. Figure 2 illustrates the correlation between the speed and body rotation, the O-L samples distribute in an upside-down 'V'-shape area. Generally speaking, the larger angle the pedestrian turns, the lower speed, which is consistent with our intuitive knowledge. Because in this paper the pedestrian's turn is not acute in one step, say, less than 60 degree, according to the relationship between spin turn and step turn in [12], we approximate that the turn strategy is independent of the support foot.

The model of admissible velocities is based on quadratic curve, and the reason we adopt quadratic curve is that it's the simplest nonlinear curve. Let l_{max} and l_{min} be the speed limits when pedestrian walks straight. The average walk speed l_{av}, which is the average of l_{max} and l_{min}, is a basic property for pedestrian. A parameter depicting the relation of speed and turning is introduced here, *the flexibility of pedestrian* λ. It controls how much l_{av} decelerates when turning. In addition, we adopt β to denote the limit of turning during one step. Given these parameters, the base curve

$$f(x) = -\lambda * x * x + l_{av}, x \in [-\beta, \beta] \tag{1}$$

embodies the character of the specific admissible velocity space, where x is the turning angle during one step. The two points $\mathcal{P} : (\pm\beta, f(\beta))$ are singular points

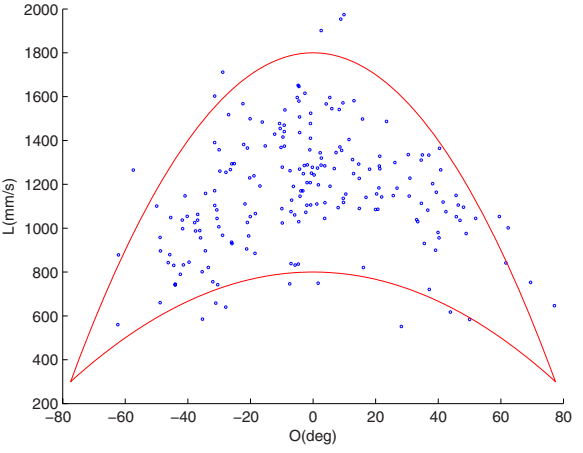


Fig. 2. Distribution of motion state samples: The horizontal axis represents the orientation change for one step and the vertical axis represents the speed for one step. The samples distribute in an upside-down 'V'-shape area bounded by two quadratic segments.

of $\mathbb{V}$, $\mathbb{V}$ denoting the set of admissible velocities. At last, the quadratic passing the singular points and $(0, l_{max})$ bounds the upper limit of $\mathbb{V}$ and similarly the one passing the singular points and $(0, l_{min})$ bounds the lower limit. For instance in Figure 2, we set l_{max}, l_{min}, λ and β respectively 1800, 800, 0.166 and 77.5, β is larger than the reachable limit of turning in order to separate the singular points from $\mathbb{V}$.

Now we estimate the boundary of this area using two quadratic curves, as shown in Figure 2:

$$y = -x * x/4 + 1800 \tag{2}$$

$$y = -x * x/12 + 800 \tag{3}$$

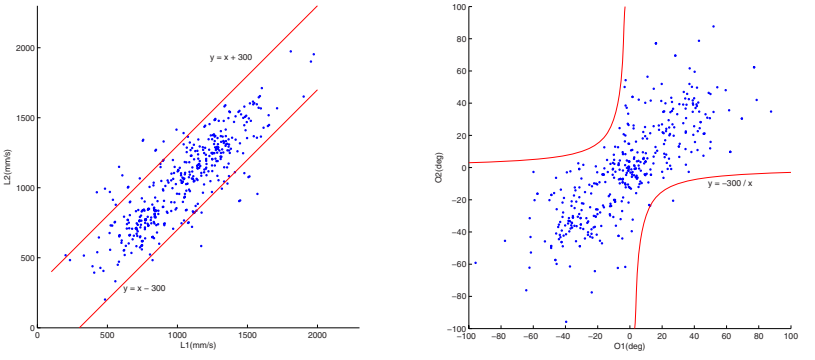
The bounded area can be covered by a bundle of quadratic curves,

$$\mathcal{C} = \{c : \ a\ conic\ passing\ (0, l)\ and\ \mathcal{P}|l \in [l_{min}, l_{max}]\} \tag{4}$$

which have no intersection in $\mathbb{V}$. In the following section, we will utilize such bundle to model the optimal velocity change.

4.2 Admissible Velocity Changes

Figure 3(a) shows the change between the successive speeds pair at one frame: The horizontal component indicates the previous speed L_1 and the vertical component indicates the next speed L_2. On one hand, most of the speed changes are bounded by some value: $L_1 - 300 < L_2 < L_1 + 300$. On the other hand, when L_1



(a) Speed change (b) Rotation change

Fig. 3. Admissible Velocity Changes:(a) The horizontal component indicates the previous speed L_1 and the vertical component indicates the next one L_2, it illustrates the dependence of L_2 on L_1;(b) The horizontal component indicates the previous turning O_1 and the vertical component indicates the next one O_2, it illustrates the consistency of motion

is low(high), L_2 is most likely high(low) to go over the limit, because it's easier to speed up(slow down) when the previous speed is low(high). Both sides will be considered in the next transition model.

Similarly, Figure 3(b) shows the change between the turning angles at successive steps: The horizontal component indicates the previous turning angle O_1 and the vertical component indicates the next one O_2. The curve $y = -300/x$ represents the boundary of the samples partially. The figure evidences the consistency of motion: when the pedestrian turn left(right) at the previous step, he will more likely turn left (right) at the next step.

These two boundaries have simple forms and they will be integrated into the model of optimal velocity change in the next section.

4.3 Optimal Velocity Change

During a human walk, the velocity at next step is related to that at the previous step. The fact that both turning angle and speed change are bounded is an important constraint. To optimize velocity change, a function $E(S_1, S_2)$ is defined as the transition cost from previous state S_1 to the next state S_2. $S_i (i = 1, 2)$ is a pair of variables (O_i, L_i) and O_i, L_i are respectively the turning angle and speed at one step. According to section **4.1**, the velocity space $\mathbb{V}$ can be covered by a bundle of quadratic curves. If S_1 and S_2 are on the same curve, we penalize orientation change only

$$E(S_1, S_2) = E_1(O_1, O_2) = \omega_1 * C_1(O_1, O_2) + \omega_2 * C_2(O_2) \tag{5a}$$

$$C_1(O_1, O_2) = \begin{cases} 1 \ , O_1 * O_2 > -300 \\ 0 \qquad , otherwise \end{cases} \tag{5b}$$

$$C_2(O_2) = \left(\frac{O_2}{60}\right)^2. \tag{5c}$$

Here, ω_1 and ω_2 are scaling constants. The continuity cost C_1 imposes the consistency of turning. The orientation deviation cost C_2 implies the bias to walk straight. Generally, when S_1 and S_2 are not on the same curve, an intermediate state S' is proposed. We set S' to lie in the same curve as S_1 and share the same turning angle with S_2, see Figure 4(b). $E(S_1, S_2)$ is divided into two parts: $E(S_1, S')$ and $E(S', E_2)$. The former is described by Equation (5a), and the latter is defined as following:

$$E(S', S_2) = E_2(L', L_2) = \omega_3 * C_3(L', L_2) + \omega_4 * C_4(L_2) \tag{6a}$$

$$C_3(L', L_2, O_2) = \left(\frac{L' - L_2}{300 * (1 - (O_2/\beta)^2)}\right)^2 \tag{6b}$$

$$C_4(L_2) = \left|\frac{L_2 - L_{av}}{500}\right|. \tag{6c}$$

Again, ω_3 and ω_4 are scaling constants. C_3 limits the change of speed and C_4 ensure the speed not deviate too much from the average speed. The denominator in the formula is to normalize the cost. In summary, the complete cost function can be defined as

$$E(S_1, S_2) = \omega_1 * C_1(O_1, O') + \omega_2 * C_2(O') + \omega_3 * C_3(L', L_2, O_2) + \omega_4 * C_4(L_2) \tag{7}$$

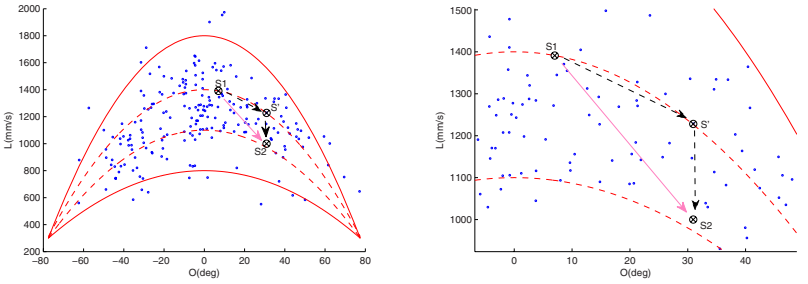
where O' and L' are the two component of intermediate state S'. In the following examples, we set ω_1, ω_2, ω_3 and ω_4 2.0, 1.0 1.0 and 0.5 respectively.

In this section, we discuss the range of the admissible velocities and changes of velocities respectively. For simplicity, we adopt some primitive analytical curves like linear and quadratic curves etc. to compose the boundary of the admissible space. As a result, a model of optimal velocity change is proposed based on the range constraint. The model provides a depiction of continuity of walk and can be applied in diverse steering problems.

5 Applications

5.1 Path Following

Path following is to move a character along the path. It is popularly applied in video games. Given a path $\mathcal{Y}$, we will collect a sequence of step decisions to follow the path. To simplify the problem without loss of realism, we adapt a discrete version of the locomotion model. For instance, the turning is sampled uniformly



(a) Transition from S_1 to S_2 (b) Transition in details

Fig. 4. Transition between two contiguous motion states: for each transition, we bring in an intermediate motion state, which lies in the same quadratic curves as the previous motion state and share the same rotation as the next motion state. We compute the cost of the transition as the sum of the two parts: $(S1 \rightarrow S')$ and $(S' \rightarrow S2)$.

distributing in the range $[-60\,°, 60\,°]$ with 13 values, and the speed is segmented into 10 samples from l_{min} to l_{max}. The samples in the admissible velocity space $\mathbb{V}$ are collected, denoted by $\mathcal{S}$, to approximate the original continuous space. At each step, to choose the best decision, the locomotion constraint and the path constraint are considered simultaneously. The locomotion constraint is the transition cost and the path constraint is described as the distance from current position to the given path, see Figure 5(a).

A strategy of three steps anticipation is adopted so that the step decisions are somewhat provident. To choose the best parameters at next step, we take the following three steps into account.

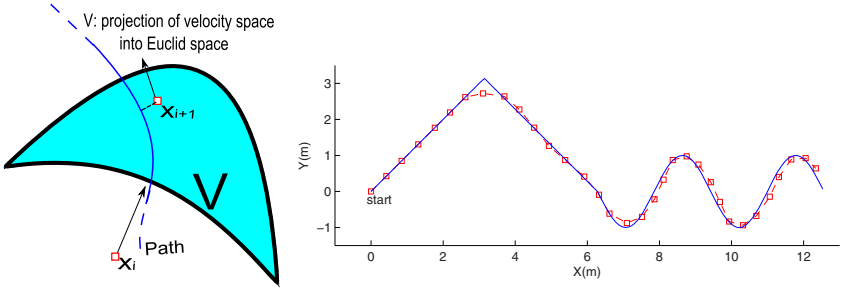
$$\Pi(x_0, s_0, \mathcal{Y}) = \underset{s_i \in \mathcal{S}, i=1,2,3}{\operatorname{argmin}} \sum_{i=1}^{3} [E(s_{i-1}, s_i) + \omega * C(x_i, \mathcal{Y})] \tag{8}$$

here x_i is the position and orientation of pedestrian which can be determined by x_{i-1}, s_i and Δt iteratively. s_0 is the current motion state and $C(x_i, \mathcal{Y})$ measures the deviation from the path $\mathcal{Y}$. ω is a weight factor. Figure 5(b) shows an example.

5.2 Walkers Interactions

An interaction occurs between two humans when they walk with converging trajectories. Simulating such interactions is complex due to the possibly high number of factors involved, both the physical factors from itself and constraint from the environment. Our work offers a formula of pedestrian locomotion. When a model of exterior constraint is provided, we can combine both to simulate the walkers' interactions.

We adopt the model proposed in [13] as the exterior constraint. Suppose two humans are walking with converging trajectories. To avoid a collision, each of



(a) (b)

Fig. 5. Path following: (a) The path constraint: given current condition x_i, a candidate of the velocity space s_{i+1} is projected into the Euclid space x_{i+1}. The deviation of x_{i+1} from the path measures the path constraint; (b) An example of path following: the blue curve represents the given path; the red squares are the series of steps following the path. Note that the speed will be low when turning and the pedestrian will skip the sharp corner.

them adjusts his velocity to keep the other out of his own personal area. In Figure 6(a), $\mathcal{R}$ and $\mathcal{P}$ are the two humans. The reference human $\mathcal{R}$ - the one actually beeing steered - has a kite shaped personal area, as shown in gray color. His velocity relative to the world is $V_{R/W}$. The perceived human is $\mathcal{P}$. $\mathcal{P}$'s position relative to $\mathcal{R}$ is P and his velocity relative to the world is $V_{P/W}$. Consequently, $\mathcal{P}$'s velocity relative to $\mathcal{R}$ is $V_{P/R} = V_{P/W} - V_{R/W}$. When the relative velocity $V_{P/R}$ orients toward $\mathcal{R}$'s personal area, there will be probably a collision between the two humans. As a result, to get rid of the collision, it is sufficient for $\mathcal{R}$ to adjust his velocity $V_{R/W}$ such that the relative velocity $V_{P/R}$ orients away from the interaction area. In Figure 6(a), the limits of interaction area T_1 and T_2 are the bounds of the relative velocities' orientation which probably results in collision. The relative velocity outside of this bound is admissible, which may belong to the "passing first" part or the "giving way" part. In order to solve the interaction in an optimal manner, the next relative velocity is desired along the limit of the interaction area, T_1 or T_2. So the next relative position is also on T_1 or T_2. Keeping $V_{P/W}$ constant and given duration Δt, each point on the limits is corresponding to a motion state in the velocity space of $\mathcal{R}$, the curves t_1 and t_2 in Figure 6(b) are the projected motion states in velocity space of $\mathcal{R}$. This is the exterior constraint for the human interaction model.

Next, our model is applied to choose the point on the limit with the lowest transition cost. Given current state S_1 of person $\mathcal{R}$, the two limits are projected to the admissible velocity space $\mathbb{V}$ of $\mathcal{R}$. By computing the transition cost E from current state S_1 to the next state, which lies on the limit t_1 or t_2, it is straightforward to get the best decision for the next step. An example about interaction between three pedestrian is given in Figure 7(b).

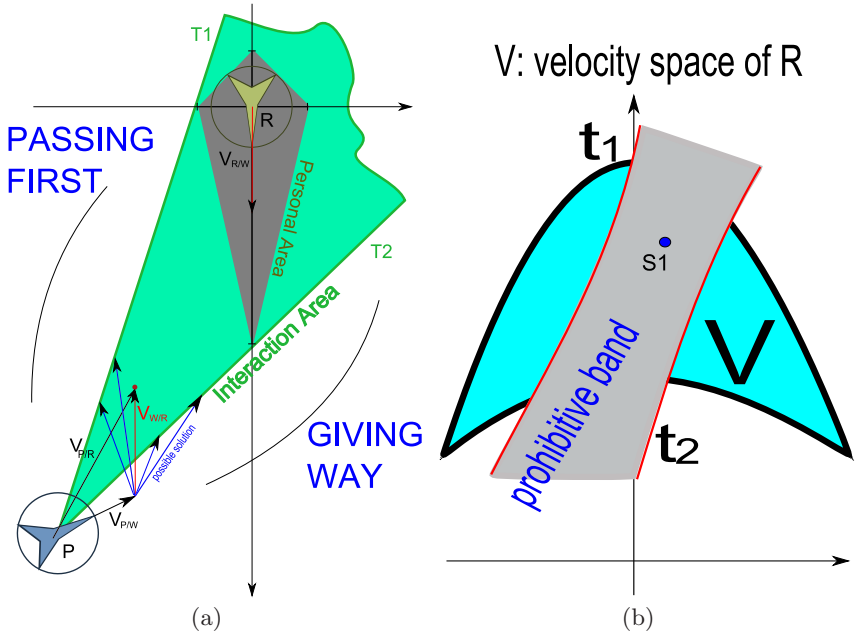


Fig. 6. (a): Illustration of the exterior constraint for the reference person $\mathcal{R}$; (b): the limit $T1$ and $T2$ are projected into the admissible velocity space $\mathbf{V}$, denoted t_1 and t_2, the projection of the interaction area is the band between t_1 and t_2. $S1$ indicates the current state of $\mathcal{R}$.

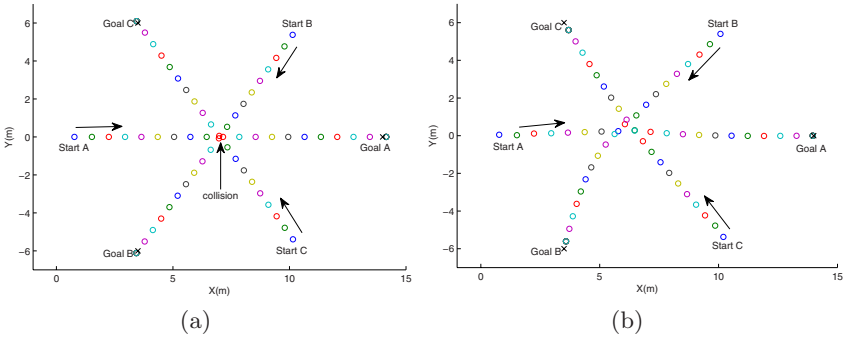


Fig. 7. This two images are an example where three pedestrians walk towards different goals. The colorful circles represent the three men's positions along time and the same color means the same time. (a): without the interaction model, the three men collide; (b): the result with our model.

6 Conclusion

This paper presents a new locomotion model of pedestrian for real-time character steering. We collect the trajectory data by motion capture system and decompose the data into single steps. Based on the analysis of step set, we take the physical ability and walking habits of human into account. We demonstrate that the admissible velocity space has a upside-down 'V' shape in some specific coordinate system, and propose a bundle of quadratic curves to represent such space. For the transition between successive steps, we give a bound of speed change and illustrate the consistency of turning. At last, all of the elements are assembled to form a transition model between successive steps. The parameters of the model can be easily tuned to characterize virtual human with different walking styles. Our model enables the virtual man to choose the naturally looking step in indeterminate situation. As an interior constraint of pedestrian, it also can be introduced into diverse autonomous behavior models.

Nevertheless, Our model is limited to the medium motions of pedestrian. For the extreme motions, especially running or spinning, it is invalid. One point which can be improved is that the current model is derived only from the estimation of the bounds of velocity space, bounds of speed changes, etc. A prospective method is to analyze the transitions between contiguous steps with some statistical technique such as Markov model. Moreover, as Arechavaleta proposed in [4], the direction of the pedestrian's trunk almost aligns with the instantaneous velocity all the time. In this paper, we adopt this conclusion and determine the trunk's orientation by associating the orientation with the velocity at each step. However, it occurs occasionally that the pedestrian walks side. With view of such case, the trunk orientation will be introduced into the current model in future work.

Acknowledgment

Yijiang Zhang's PhD work is granted by the Eiffel program of the French Egide Association, as well as by the joint PhD program of the French Embassy in China. Authors would also like to thank Richard Kulpa, Armer Cretual and Anne-Helene Olivier for their precious help on motion capture data acquisition.

References

1. Reynolds, C.W.: Steering behaviors for autonomous characters. In: Game Developers Conference 1999 (1999)
2. Go, J., Vu, T., Kuffner, J.J.: Autonomous behaviors for interactive vehicle animations. In: SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, Aire-la-Ville, Switzerland, Switzerland, pp. 9–18. Eurographics Association (2004)
3. Kwon, T., Shin, S.Y.: A steering model for on-line locomotion synthesis. Comput. Animat. Virtual Worlds 18(4-5), 463–472 (2007)

4. Arechavaleta, G., Laumond, J.P., Hicheur, H., Berthoz, A.: The nonholonomic nature of human locomotion: a modeling study. In: Biomedical Robotics and Biomechatronics, pp. 158–163 (2006)
5. Hicheur, H., Vieilledent, S., Richardson, M.J.E., Flash, T., Berthoz, A.: Velocity and curvature in human locomotion along complex curved paths: a comparison with hand movements. Experimental Brain Research 162(2), 145–154 (2005)
6. Metoyer, R.A., Hodgins, J.K.: Reactive pedestrian path following from examples. In: CASA 2003: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003), Washington, DC, USA, p. 149. IEEE Computer Society, Los Alamitos (2003)
7. Antonini, G., Bierlaire, M., Weber, M.: Discrete choice models of pedestrian walking behavior. Transportation Research Part B: Methodological 40(8), 667–687 (2006)
8. Blue, V.J., Adler, J.L.: Cellular automata microsimulation for modeling bi-directional pedestrian walkways. Transportation Research Part B: Methodological 35(3), 293–312 (2001)
9. Treuille, A., Lee, Y., Popović, Z.: Near-optimal character animation with continuous control. In: SIGGRAPH 2007: ACM SIGGRAPH 2007 papers, p. 7. ACM, New York (2007)
10. Dirk, H., Peter, M.: Social force model for pedestrian dynamics. Physical Review E 51, 4282 (1995)
11. Fajen, B.R., Warren, W.H.: Behavioral dynamics of steering, obstacle avoidance, and route selection. Journal of Experimental Psychology: Human Perception and Performance 29, 343–362 (2003)
12. Olivier, A.H., Fusco, N., Cretual, A.: Local kinematics of human walking along a turn. Computer Methods in Biomechanics and Biomedical Engineering 11(suppl. 1), 177–178 (2008)
13. Pettré, J., Ondrej, J., Olivier, A.H., Crétual, A., Donikian, S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2009)

Path Abstraction for Combined Navigation and Animation*

Ben J.H. van Basten and Arjan Egges

Center for Advanced Gaming and Simulation, Utrecht University
The Netherlands
{basten,egges}@cs.uu.nl

Abstract. Natural motion of virtual characters is crucial in games and simulations. The quality of such motion strongly depends on the path the character walks and the animation of the character locomotion. Therefore, much work has been done on path planning and character animation. However, the combination of both fields has received less attention. Combining path planning and motion synthesis introduces several problems. A path generated by a path planner is often a simplification of the character movement. This raises the question which (body) part of the character should follow the path generated by the path planner and to what extent it should closely follow the path. We will show that enforcing the pelvis to follow the path will lead to unnatural animations and that our proposed solution, using path abstractions, generates significantly better animations.

Keywords: path planning, motion synthesis, filtering.

1 Introduction

Natural movement of characters is crucial in games and simulations. In many virtual environments characters need to walk from a point A to a point B. The quality of the total movement is highly dependent on both the path the character chooses and the walking animation itself. In many systems this process is typically a 2-step process [1]. First, a path planner generates a collision-free path P. Second, an animation system generates a locomotion animation that drives the character to follow this path as closely as possible. Figure 1 shows an example of such a 2-step process. The Corridor Map Method [2] is used as path planner and a motion graph approach is used as motion synthesizer [3].

Combining a path planner with a motion synthesizer seems like a trivial task. However, there are some major issues that none of the existing methods address. For example, a generated animation can not always be guaranteed to exactly

* This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 182–193, 2009.

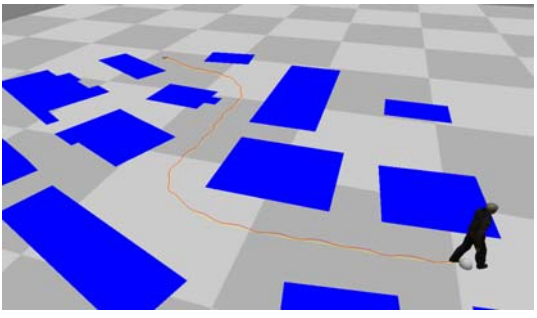


Fig. 1. Motion synthesis techniques such as a motion graph can generate locomotion along a smooth obstacle free path

follow a planned path. Errors may occur due to curvature limits in the motion database or alignment mismatches. A possible solution for this type of problem is to apply a path editing technique [4]. However, this assumes that we know what a path actually represents: the path generated by a path planning algorithm is a parametric curve that represents a simplification of the path the character needs to follow. Generally it is not clear which (body) part of the character should follow this path. In many systems the path is interpreted as the desired trajectory of the projection of the pelvis on the plane. However, when one takes a closer look at the trajectory of the pelvis during locomotion (see Figure 2) it appears that the pelvis oscillates during such a motion. Early experiments in biomechanics already showed that the center of mass oscillates both horizontally and vertically during locomotion [5]. When the motion synthesizer forces the pelvis to closely follow the desired path one loses this natural oscillation or *wiggle*, resulting in an unnatural motion. Not enforcing the path constraint will lead to path deviation, but over-enforcing the path constraint leads to unnatural motion.

In this paper, we propose a solution to this problem by using a *path abstraction*. We will show that using an abstract path instead of the traditional pelvis trajectory will lead to more natural motions while still being able to enforce path constraints. We propose several ways of obtaining an abstract path from an existing animation and we will compare their advantages and limitations. Path abstractions can easily be incorporated in motion synthesizers. Because it can be done in the preprocessing phase, it will not affect the performance of the motion synthesizer. This paper is organized as follows. In Section 2 we will give an overview of related work in both the animation and path planning fields. In Section 3 we will present a number of different path abstraction techniques. Then, we will detail our path abstraction experiment in Section 4. We will show some results of our method in Section 5 and end with the conclusion and future work.

2 Related Work

A straightforward way of doing path planning is by building a grid over the environment and then applying an A* search algorithm to find a path. Unfortunately, grid-based techniques do not result in smooth paths and the quality of the output path and computation time are dependent on the resolution of the grid. Potential field methods use attractive and repulsive forces to lead the character to its goal. An advantage of these methods is that they are able to avoid dynamic obstacles and generate smooth paths. However, most potential field methods suffer from local minima. Other techniques construct a graph over the environments. The Probabilistic Roadmap Method [6] creates a roadmap of the environment in the offline phase by trying to connect random samples in the environment. Although this technique is probabilistically complete it will generate non-smooth path that might take detours. Recent algorithms make use of corridors. A general framework for corridor-based path planning, called the Corridor Map Method, was proposed by Geraerts and Overmars [2]. The CMM uses corridors that encapsulates the global path of the character and an attraction point to guide the local path.

The second part of the 2-step process is to produce a realistic animation of a character following the generated path. In order to get realistic animations motions are often recorded using motion capture systems. It is widely known that the recorded data is difficult to adapt at a later stage: if a clip of motion data does not fit the animators needs, it has to be rerecorded. To reuse motion captured motion, several techniques have been developed to adapt the motion, such as time warping [7] and displacement mapping [8]. This allows for changing the timing and shape of the motion. Unfortunately only small adaptations are possible. One can also concatenate motions by transitioning from one motion to another motion using interpolation. In motion graph approaches [3] animation clips are structured in a graph where all edges correspond to clips of motion. The nodes are the frames where the transitions take place. The graph is extended by adding new nodes and edges in places where the frames resemble each other. The higher the transition threshold is, the more transitions (or: edges) are added. Several distance metrics have been used to identify pairs of resembling frames [9]. A new animation can be constructed by doing a graph search according to a set of constraints. An example of such a constraint is an input path that the generated animation should follow.

Some research has been done on combining path planning and character animation. Choi et al. [10] create a map of footprints based on a probabilistic path planning technique. They search this roadmap to find a collision-free path after which displacement mapping is used to adjust the motions to fit the target footprints. Sung et al. [11] also plan a path using probabilistic path planning techniques and generate a motion (using a motion graph variant) that approximately follows the path. This motion is then adjusted to follow the path more accurately. Lau and Kuffner [12] create a high-level motion finite-state machine and search over this graph-like structure for navigation. This method implicitely uses the animation data to define the navigation space. Srinivasan et al. [13]

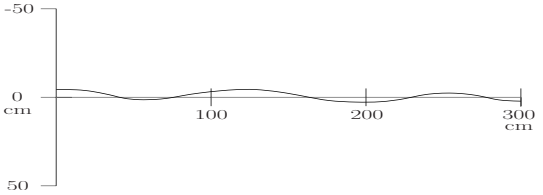


Fig. 2. The trajectory of the pelvis during locomotion along a straight line

present a conceptually similar work. Efforts have been made to speed up the search process by precomputing the possible positions a character can get to using a motion graph [14]. In Pettre et al. [1] the path planning and animation techniques are separated which means another path planner can be used to achieve similar results. They use the PRM to compute the path and interpolate motion clips to generate the character animation. Kamphuis et al. [15] also separate the path planner from the animation system. They assume the existence of a collision-free path and used the animation method based on [1] to generate an animation along the path. However, very specific motion clips are required to generate natural looking motions.

3 Path Abstraction

As already mentioned, using the pelvis trajectory as representation for the path will result into an unnatural motion because we will loose the natural pelvis oscillation. We can approach this problem from two directions. We could incorporate the pelvis oscillation in the paths of a path planner, but this has some major drawbacks. The oscillation is highly dependent on the character and we would need to know in which phase of the locomotion cycle we want to begin (do we start with a left or right foot?). Therefore we do not change the path but we change the representation of the character motions in the database. We do not try to force the pelvis to follow the presented path, but an abstraction of the character motion. In this section we will elaborate on the path abstraction techniques. First, we will elaborate on the motion graph search algorithm. This is the only component of the path planning and animation system that needs to be changed, not the motions, motion graphs or paths.

We use a standard search algorithm for motion graphs [3]. On-line path synthesis is done using an incremental branch-and-bound algorithm. While the traversed distance of the motion is smaller than the length of the path, this branch-and-bound algorithm is employed, checking the search space n frames around the current node looking for the motion clip that follows the path P best. For a more elaborate explanation on the motion graph and the branch-and-bound search algorithm, we refer the reader to the original article [3]. In order to find the motion clip that follows the path best often the projected trajectory of the pelvis on the plane is compared to the query path P. This results in losing the natural oscillation of the pelvis as explained in the previous sections. In our

method, when selecting a new motion clip from the graph, we compare the the query path P with a path abstraction instead of the pelvis trajectory.

3.1 Path Abstraction Techniques

In this section we will present the 4 path abstraction techniques. Although most of them act as a low-pass filter, the resulting path abstraction does not need to be the same. We will denote P_{wiggle} as the original oscillating pelvis trajectory projected on the ground plane, P_{global} is the trajectory of the path abstraction.

Joint Combination (JC). When examining recorded motion, it appears that there is no joint without oscillation, so we can not choose a particular joint as indication of the character's global motion. Although the combination of several joints might be interesting. It is already known that the *trunk* is already a good indication of the characters direction [16]. Unfortunately, the position of the trunk still suffers from an oscillation. Fortunately, some joints appear to follow a path that resembles the mirrored version of the pelvis path. These are the feet, ankles and knees. We will determine P_{global} by interpolating these joints.

Joint Orientation Extrema (JOE). We have observed that the global path roughly follows the midpoints of two consecutive local extrema of the joint paths. Since extrema are easiest determined in monotone functions we will transform the non-monotone path such that it is x-monotone. We can then determine extrema by finding zero-crossings of the first derivative. P_{global} is then determined by connecting the midpoints of the segments between consecutive extrema.

Gaussian smoothing (GS). The third path abstraction method is Gaussian smoothing. It is suitable for smoothing arbitrary curves and is used in image processing to filter out high frequency distortions. The wiggle of the pelvis can be viewed as such a distortion of the global path of the character. An approximation of P_{global} can be obtained by applying Gaussian smoothing to the path of the pelvis. The pelvis trajectory is represented as parametric curve. These are then convolved with a one-dimensional Gaussian kernel $G_\sigma(t)$ of standard deviation σ

$$G_\sigma(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-t^2}{2\sigma^2}} \tag{1}$$

$X(t)$ is defined as the convolution of $x(t)$ with Gaussian kernel $G_\sigma(t)$ for some value of σ: $X(t) = G_\sigma(t) \otimes x(t)$. $Y(t)$ is defined similarly.

Differentiation commutes with convolution and so $X(t)' = G'_\sigma(t) \otimes x(t)$, $X(t)'' = G''_\sigma(t) \otimes x(t)$. Where $G'_\sigma(t)$ and $G''_\sigma(t)$ are the first and second order derivatives of the Gaussian kernel. $X(t)$ and $Y(t)$ represent a smoothed version of P_{wiggle} with a smoothing factor of σ. $X(t)$ and $Y(t)$ can be used as an approximation for P_{global}.

A problem with any averaging filter is that the resulting path is shrunk towards the center of its supporting circle. Each point is filtered using its neighbors

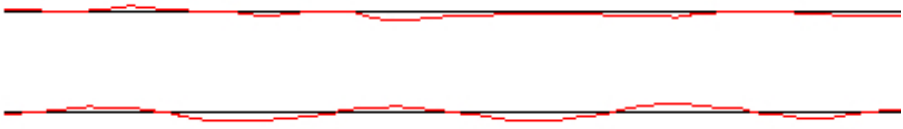


Fig. 3. Two resulting pelvis trajectories (red) for the query paths (black). As can be clearly seen, the Gaussian smoothing path abstraction (below) preserves the natural oscillation of the pelvis whereas the standard method (top) does not.

that in both directions curve towards the center of the supporting circle. A solution to this problem was provided by Lowe [17]. He states that the shrinkage is dependent on the amount of smoothing and the local curvature. We can use the known value of σ and the measured curvature of the smoothed curve to compensate for the degree of shrinkage that must have occurred. We do not actually know the value of the original curve radius r, therefore we need to use the second derivative of the smoothed curve X. Given σ and the measured curvature of the smoothed path the shrinkage at a point t is defined as

$$r(1 - e^{\frac{-\sigma^2}{2r^2}}) \tag{2}$$

where r is given by

$$X''(t) = \frac{e^{\frac{-\sigma^2}{2r^2}}}{r} \tag{3}$$

For each point on the path the appropriate error value is numerically determined and subtracted from the smoothed coordinate value. Several solutions exist to convolve the Gaussian kernel when it extends beyond the end of the path. We will use the method of Rosin [18] where the path is extended by 3σ at both ends by duplicating and reflecting the path. We use a σ of 0.75 meter, corresponding to one step cycle.

B-Spline (BS). For the last path abstraction method, we follow the idea of Gleicher [4]. He uses a uniform cubic B-Spline with 6 knots located at uniform intervals in the trajectory of the pelvis. It can be interesting to see how well this method performs in comparison to the already proposed methods. Given a set of control points P, for each segment i we determine the trajectory of P_{wiggle} using:

$$\mathbf{P}_i(t) = [t^3\ t^2\ t\ 1]\frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}\begin{pmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{pmatrix} \text{ for } t \in [0,1] \tag{4}$$

4 Experiment

We will perform an experiment to evaluate the result of each path abstraction technique. The performance is measured by comparing generated animations

with real recorded motions. We have motion-captured a human walking along three paths: a straight path, a low curvature path and a high curvature path. We manually extracted the test paths and used them as input to the motion graphs. This allows us to compare the generated animations with the original recorded motions. We will generate motions using the 5 different path abstraction techniques (including the standard pelvis-based method). Note that we do not change the motions in the database, but we change the representation. During the branch-and-bound algorithm (see Section 3) we do not compare the pelvis trajectory to the query path, but to the path abstractions. In total we have 5 different versions of the search algorithm each corresponding to a different path abstraction technique. Since the transition threshold of the motion graph greatly influences the quality of the resulting animation, we will use 7 different threshold values. We will use a common joint-angle based distance metric [9]. For each threshold a different motion graph is constructed. The properties of the graphs for the 7 thresholds are described in Table 2. These thresholds vary from a really small graph that barely allows for following a path at all, to a large graph that is able to strongly enforce the pelvis to follow the path. For all thresholds each technique will generate 40 animations per test path. We build up a motion graph using 5 recorded walking motions of varying curvature (done by the same human as the test paths). The total number of frames was 683.

4.1 Quality Measures

In this section we will elaborate on the evaluation of the generated motions. We will present five different measures to evaluate the generated motions. Note that for a natural animation these measures need to be as close as possible to original recordings.

Foot skating. We will measure the amount of foot skating because it is an indicator for the quality of the motion. We will measure foot skating using the method presented by [9]. Since the traversed distance per animation can slightly differ, we will normalize the total foot skating by dividing it with the length of the traversed distance of the generated animation. Because real recorded motion also has foot skating due to noise and post-processing, we can compare the foot skating of the generated motions with that of real motions.

Wave measures. The problem of using the pelvis as path abstraction is that, especially for larger graphs, the movement of the pelvis loses the natural oscillation. This oscillation of the pelvis can be considered a wave. A wave is defined by specific properties, such as *wavelength* and *amplitude*. Note that the phase is not a good measure, for it depends on the foot that the animation starts with, which is dependend on the starting node in the motion graph.

The goal of the path abstraction techniques is that the pelvis in the generated motion will wiggle the same way as in the recorded motions. Therefore these techniques should yield motions with similar wavelength and amplitude. In

contrast, the standard method implicitely tries to minimize the wiggle and thus should have a different outcome of wavelength and amplitude. We will determine the average wavelength and amplitude of the curvature function κ of the pelvis trajectory. This curvature function is monotone and therefore it is straightforward to determine the average wavelength and amplitude of the wiggle of κ. Let $\mathbb{E}$ be the set of local extrema of the curvature function $\kappa(t)$. Then the wavelength w of two successive points $e_i, e_{i+1} \in \mathbb{E}$ is determined as two times the distance between those points $w(e_i, e_{i+1}) = 2d(e_i, e_{i+1})$. Then the average of all measured wavelengths is calculated by $\mu_{wave} = \frac{1}{n-1} \sum_{i=1}^{n-1} w(e_i, e_{i+1})$ where n is the number of extrema in $\mathbb{E}$. The amplitude of the wiggle is calculated in a similar way. First, the amplitude of a local extrema e_i is defined as $\text{amp}(e_i) = |\kappa(e_i)|$. Then the average of all amplitudes is calculated: $\mu_{amp} = \frac{1}{n} \sum_{i=1}^{n} amp(e_i)$ where n is the number of extrema in $\mathbb{E}$.

Ideally, one would expect that the wavelength and amplitude of the pelvis in a generated motion should have an equal variance as in original motion. Therefore we will also measure the standard deviation σ_{wave} and σ_{amp} of the wavelength and amplitude. In Table 1 the measurements of the recorded motions corresponding to a straight path, low curvature path and high curvature path motion are shown.

Table 1. The measurements of the recorded motions for the three test paths

Measure	Straight	Low curv.	High curv.
Foot skating (cm)	0.20	0.28	0.39
Amplitude (cm)	3.91	3.55	4.51
Wavelength (cm)	57.83	55.31	49.06
Amplitude dev (cm)	0.55	0.78	1.10
Wavelength dev (cm)	1.33	3.73	6.44

As said, we will use 7 different transition thresholds, ranging from 0.5 to 3.5. Each threshold will lead to a different motion graph. The higher the threshold the more transitions are allowed and the denser the resulting motion graph will be. In Table 2 the number of edges, nodes and frames per graph are shown.

Table 2. The complexities of the graphs for different thresholds

Threshold	#nodes	#edges	#frames
0.5	79	114	653
1.0	295	528	2874
1.5	475	925	5202
2.0	566	1300	8046
2.5	599	1468	9403
3.0	611	1563	10235
3.5	613	1584	10425

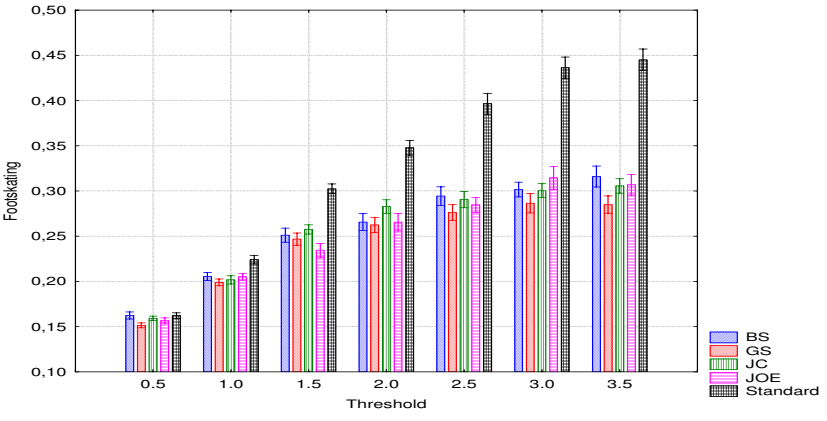


Fig. 4. The average foot skating for each method and technique over all paths. Vertical bars denote standard error of mean.

5 Results

In this section we will discuss the results of the techniques separately for each quality measure. To analyse the techniques statistically, we employ 3 repeated measures ANOVA, one for each quality measure. Whenever significant effects have been found, post-hoc analysis (Tukey, $p = 0.05$) was performed for pairwise comparison.

Foot skating. We found an effect of technique on foot skating ($F(4, 156) = 367.40$, $p < 0,001$). The foot skating for each method and technique over all paths is depicted in Figure 4. On the horizontal axis the 7 thresholds used during the graph creation ranging from 0.5 to 3.5 are shown. Pairwise comparison showed that the standard method generates significantly more foot skating (over all paths and thresholds) than all other techniques (all $p < 0.001$) and the GS method generates significantly less (all $p < 0.01$). We have observed that for the high curvature path, the foot skating of all techniques is lower than the foot skating of recorded motion. This is mainly because the motion graph is not able to generate animations with such a high curvature. It is known that there are not many transitions in the graph going to animations with such a high curvature [9].

Amplitude. We found an effect of technique on the absolute difference of the amplitude of the resulting animations and the original recordings ($F(4, 156) = 630.32$, $p < 0,001$). The amplitude difference for each technique over all paths and thresholds is depicted in Figure 5. It is clear that the path abstraction techniques are able to preserve the wiggle with a similar amplitude as the recorded motion, as their difference is significantly smaller (all $p < 0.001$) (See Figure 3). Overall, the amplitude differences from the GS and JC method are significantly smaller than the other techniques (all $p < 0.004$). We have observed that for

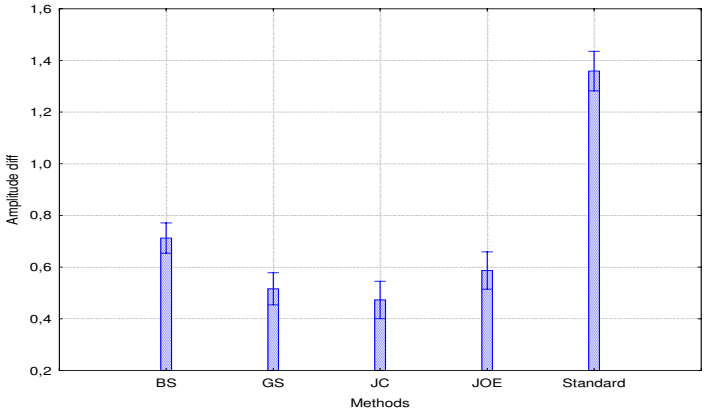


Fig. 5. The overal average amplitude difference for each technique. Vertical bars denote standard error of mean.

the standard method the amplitude decreases as the threshold increases and becomes much smaller than the amplitude of recorded motions. This typically illustrates the problem of loss of oscillation. For the high curvature path we have observed that the amplitudes of all the path abstraction techniques are at a similar level as the recorded motion, except for the threshold of 0.5. This is because the motion graph is quite small and is unable to correctly follow the path and often generates straight locomotion, which has a smaller amplitude than curved locomotion.

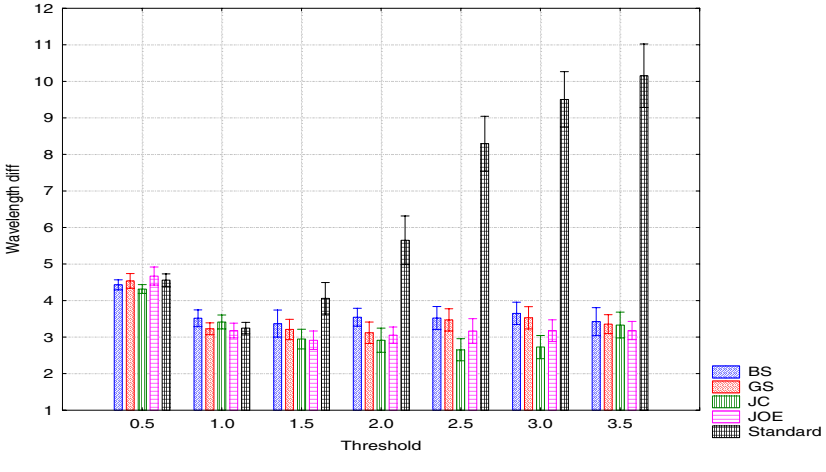


Fig. 6. The wavelength difference for each technique over all paths. Vertical bars denote standard error of mean.

Wavelength. We found an effect of technique on the absolute difference of the wavelength of the generated animations and recorded motions $F(4, 156) = 317.35$, $p < 0.001$. Overall, the path abstraction techniques result in motions whose wavelength is closer to natural values than the standard method (all $p < 0.001$), as is illustrated in Figure 6. For the standard method the wavelength decreases as the threshold increases for straight and low curvature paths. In contrast, the wavelength for the path abstraction techniques remains stable and close to natural values. For the high curvature path we see different results: in contrast to the straight and low curvature path the path abstraction techniques do not perform significantly better.

We observed that the standard deviation of amplitude and wavelength for all path abstractions differs from the standard deviations of recorded motions. However, the standard deviation of the wavelength of the standard method continuously increases as the threshold increases, whereas for the path abstraction it remains stable for all thresholds.

6 Conclusion and Future Work

In this paper we have combined path planning and animation synthesis methods to generate an animated character walking from a start to a goal position. Doing so, we have seen that defining what the path of the character means is very important. Like many motion synthesizers, the standard version of the motion graph uses the pelvis trajectory as an approximation of the path of the character. The standard motion graph method induces significantly more foot skating. Also the levels of the amplitude and wavelength of the oscillation show that the standard method is not able to preserve the oscillation of the pelvis. This translates to a far less natural looking animation for the standard method in comparison to the path abstraction techniques. The results presented in this paper are not only useful for motion graph based animation systems but for any motion synthesizer that depends on a representation of a path that a character follows.

Of the different abstraction techniques, the joint combination method is the easiest to implement, but it gives less good results than other path abstraction techniques. The results could be improved by further investigating possible joint combinations. The Joint Extrema method is useful because there are no parameters that need to be set. However, this method will only work for motions where a clear oscillation is present. The B-spline based method might suffer from corner cutting. This obviously depends on the placement of the knots. Overall, we found that the Gaussian smoothing filter gives the best results for foot skating and oscillation preservation.

There is still a lot of work to do in this field. The limitation of our current approach is that we only look at planar locomotion. For other types of (loco)motions, the oscillation will be different or even absent. Most of the path abstraction techniques we presented might therefore not work anymore. Also, our database consisted of animations recorded from a single person. It might

be interesting to incorporate humans of varying gender and height. Finally, we would like to perform more thorough evaluations of our results, by looking at the perceived quality of the motions, as done in [9].

References

1. Pettré, J., Laumond, J.P., Siméon, T.: A 2-stages locomotion planner for digital actors. In: Proc. of the 2003 Symposium on Computer animation, Aire-la-Ville, Switzerland, pp. 258–264. Eurographics Association (2003)
2. Geraerts, R., Overmars, M.: The corridor map method: Real-time high-quality path planning. Computer Animation Virtual Worlds 18, 107–119 (2007)
3. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: SIGGRAPH, pp. 473–482. ACM Press, New York (2002)
4. Gleicher, M.: Motion path editing. In: Proc. of the 2001 symposium on Interactive 3D graphics, pp. 195–202. ACM, New York (2001)
5. Saunders, J., et al.: The major determinants in normal and pathological gait. The Journal of Bone and Joint Surgery 35(3), 543 (1953)
6. Barraquand, J., Kavraki, L., Latombe, J.C., Li, T.Y., Motwani, R., Raghavan, P.: A random sampling scheme for path planning. Int. Journal of Robotics Research. 16, 729–744 (1997)
7. Witkin, A., Popović, Z.: Motion warping. In: SIGGRAPH, pp. 105–108. ACM Press, New York (1995)
8. Bruderlin, A., Williams, L.: Motion signal processing. In: SIGGRAPH, pp. 97–104. ACM Press, New York (1995)
9. van Basten, B.J.H., Egges, A.: Evaluating distance metrics for animation blending. In: Proc. of the 4th Int. Conf. on Foundations of Digital Games, pp. 199–206. ACM, New York (2009)
10. Choi, M.G., Lee, J., Shin, S.Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Trans. Graph. 22(2), 182–203 (2003)
11. Sung, M., Kovar, L., Gleicher, M.: Fast and accurate goal-directed motion synthesis for crowds. In: Proc. of the 2005 Symposium on Computer animation, pp. 291–300. ACM, New York (2005)
12. Lau, M., Kuffner, J.J.: Behavior planning for character animation. In: Proc. of the 2005 Symposium on Computer animation, pp. 271–280. ACM, New York (2005)
13. Srinivasan, M., Metoyer, R.A., Mortensen, E.N.: Controllable real-time locomotion using mobility maps. In: Proc. of Graphics Interface, pp. 51–59 (2005)
14. Lau, M., Kuffner, J.: Precomputed search trees: Planning for interactive goal-driven animation. In: Proc. of ACM SIGGRAPH, pp. 299–308 (2006)
15. Kamphuis, A., Pettre, J., Overmars, M., Laumond, J.P.: Path finding for the animation of walking characters. In: Poster Proc. of Eurographics, pp. 8–9 (2005)
16. Arechavaleta, G., Laumond, J.P., Hicheur, H., Berthoz, A.: The nonholonomic nature of human locomotion: a modeling study. In: Int. Conf. on Biomedical Robotics and Biomechatronics, Pisa, Italy, pp. 158–163 (2006)
17. Lowe, D.: Organization of smooth image curves at multiple scales. In: Proc. 2nd ICCV, pp. 558–567 (1988)
18. Rosin, P.: Representing curves at their natural scales. Pattern Recognition 25(11), 1315–1325 (1992)

Camera Planning in Virtual EnvironmentsUsing the Corridor Map Method

Roland Geraerts*

Institute of Information and Computing Sciences, Utrecht University
3508 TA Utrecht, the Netherlands
roland@cs.uu.nl

Abstract. Planning high-quality camera motions is a challenging problem for applications dealing with interactive virtual environments. This challenge is caused by conflicting requirements. On the one hand we need *good motions*, formed by trajectories that are collision-free and keep the character that is being followed in clear view. On the other hand, we need *frame coherence*, i.e. the view must change smoothly such that the viewer does not get disoriented. Since camera motions dynamically evolve, good motions may require the camera to jump, leading to a broken frame coherence. Hence, a careful trade-off must be made. In addition to this challenge, interactive applications require real-time computations, preventing an exhaustive search for 'the best' solution.

We propose a new method for planning camera motions which tackles this trade-off in real-time. The method can be used for planning camera motions of NPC's and first-person characters. Experiments show that high-quality camera motions are obtained for both scenarios in real-time.

1 Introduction

In interactive virtual environment applications, such as games, training systems and architectural applications, a virtual camera needs to be steered. In contrast to first-person games, in which the camera is steered by the user, we focus at applications which require automatic planning of camera motions. In such applications, manual camera control would require too much mental energy or would be too difficult for inexperienced users [1,2].

A camera motion consists of two paths: a camera path, which describes the camera positions in the environment over time, and an aim path, which describes the corresponding positions the camera looks at. We aim at creating good camera motions which feature a high *spatial awareness* of the viewer and coherent motions preventing the viewer from getting *motion sick*.

* This research has been funded by the Dutch BSIK/BRICKS Project and the Metaversel Project.

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 194–206, 2009.

The viewer's spatial awareness, i.e. the knowledge of the viewer's location and orientation within the environment, can be maintained if the following constraints are satisfied. The camera and aim positions must not collide with obstacles in the environment. In addition, the character should always be in view. Finally, the camera should not be too close to the character or obstacles (otherwise a large part of the view would be blocked), nor should it be too far (because the camera needs to stay behind the character to maintain the understanding of its orientation). This can be accomplished by keeping some minimal amount of clearance from the camera to the obstacles in the environment. Also, the camera should be placed behind the character at a certain preferred distance.

A viewer can get motion sick when its view changes too abruptly, and, hence, the positions on the paths should change smoothly and the angular velocity of the camera view should be minimal. Also, motion-sickness can be reduced by anticipating the direction in which the view is going. Hence, keeping frame coherence is of major importance.

Automated camera planning is a difficult task because a careful trade-off must be made between satisfying the constraints and frame coherence [3]. That is, favoring the constraints (e.g. keeping the character always in view) can lead to a broken frame coherence. Likewise, favoring frame coherence can lead to situations where the character blocks the user's view.

In related work, a distinction can be made between methods that deal with camera planning for *known* input paths [3,4,2] versus *unknown* paths [5,3,1]. Applications that deal with known input paths are games and architectural applications. In a game, the known path could be a character's path which is replayed or an NPC's (Non-Player Character) path which is being followed. In an architectural application, the path could be a walk-through in a virtual environment. A typical application in which the character's positions are created dynamically (so the path is unknown in advance) is a third-person game.

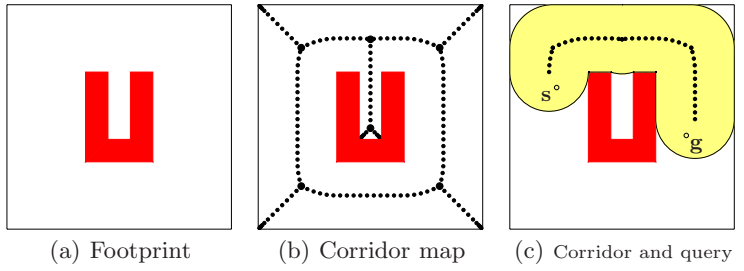
We will present a new method for planning good camera motions in real-time. We induce collision-free motions by using the Corridor Map Method, which is a global path planning technique [6,7]. The character is kept in clear view by anticipating on the future trajectory. Next, frame coherence is maintained by modeling the motions as forces applied to the camera and its view.

The paper is organized as follows. In Section 2 we will review the Corridor Map Method which is the framework we use for the new method. Its key ingredient, i.e. creating smooth collision-free paths, is discussed in Section 3. We will then introduce our new method in Section 4 which creates smooth camera motions, composed of a smooth camera and aim path, for a known character path. We will adjust the method in Section 5 such that we can deal with paths whose positions are unknown in advance. Next, we will conduct experiments in Section 6 to test the quality and performance of the method and conclude in Section 7 that high-quality camera motions are computed in real-time.

2 The Corridor Map Method

The *Corridor Map Method* (CMM) has been designed for path planning in real-time interactive environments and games [8]. The strength of the CMM is that it combines a fast global planner with a powerful local planner, providing real-time performance and the flexibility to handle a broad diversity of path planning problems. The CMM consists of an off-line construction phase and a on-line query phase (see Fig. 1). In the construction phase, we build a *Corridor Map* which is a data structure that represents the free space, i.e. the 2D walkable space, in a virtual environment. The free space is defined as the space that is not occupied by the *footprint* of the environment. This footprint is made up of a collection of geometric primitives, such as points, lines, polygons, and disks, all lying in the ground plane. The underlying graph of the map is the Generalized Voronoi Diagram [7]. The edges of this graph are sampled. We assign to each sampled point on an edge the radius of the largest empty disk. Each point/radius combination forms a maximum clearance disk. Then, a sequence of disks forms a *Corridor*, and a system of corridors is referred to as the *Corridor Map*.

In the query phase, we use the Corridor Map to compute the shortest corridor which encloses the future path of the character. We connect the start and goal positions of the character to the map and retrieve the shortest path in the map connecting these positions. We refer to this path as the *backbone path* of the corridor. The corresponding sequence of disks forms the corridor which guides the global motions of the character (or camera). An example of such a corridor, backbone path (small disks), and query is displayed in Fig. 1(c).



(a) Footprint (b) Corridor map (c) Corridor and query

(d) Control path (e) Smooth path (f) Camera & aim path

Fig. 1. The construction phase (a–b) and query phase (c–f) of the CMM

The local motions of a character are created by following an *attraction point* which moves along a *control path* toward the goal. In [6], we used the backbone path as control path. If a character is attracted by this path, its final path will have much clearance because the backbone path is composed of maximum clearance points. Nevertheless, the final path might be longer than is necessary. We are interested in a short control path that has some amount of minimum clearance to the environment. Such a path is created by shrinking the corridor with the preferred amount of clearance. The control path is then the shortest path inside the shrunk corridor [9]. We refer the reader to Fig. 1(d) for an example of such a path. It might be tempting to use the control path as final path for the character. However, this path is not smooth. In addition, traversing the fixed control path would decrease the flexibility of the method. Because the character is able to deviate from the control path, the character's motions can become smooth as is shown in Fig. 1(e).

In the next section we will discuss the technique for creating a smooth path for the character. This technique will form the basis for creating smooth camera motions such as visualized in Fig. 1(f). The two open disks denote the character's start and goal positions. The black closed disks denote the camera positions and the arrows denote the aim positions.

3 Creating a Smooth Path

Given a start position $\mathbf{s}$ and goal position $\mathbf{g}$, we extract a corridor from the corridor map which encloses the future path connecting the start to the goal. Next we extract a control path which is represented by a continuous sequence of points. This control path is then used to create the smooth path.

The algorithm computes a path Π which is defined as a continuous sequence of two-dimensional positions. Each position is annotated with a time stamp. We require that the character, modeled by a disk with radius r, is inside the corridor for any position along the path. More formally, we define a path as follows:

Definition 1 (Path Π)**.** *A path Π inside a corridor $\mathcal{C}$ is a continuous map $\Pi \in [0, t_{max}] \rightarrow \mathbb{R}^2$ such that $\forall t \in [0, t_{max}] : \Pi[t] \in \mathcal{C}$.*

By $\Pi[t]$ we denote the position at time t. For example, the start and goal position are equal to $\mathbf{s} = \Pi[0]$ and $\mathbf{g} = \Pi[t_{max}]$, respectively. Of course, the value for t_{max} will be known after the computation of the path.

The algorithm iteratively computes positions and time stamps that make up the path. The algorithm finishes when the Euclidean distance between the last computed position and the goal is smaller than a small threshold $\epsilon = 0.01$.

The character is initially placed at position $\mathbf{x} = \mathbf{s}$ and is set into motion and steered by receiving steering and boundary forces [8]. The steering force $\mathbf{F}_s(\mathbf{x})$ guides the character toward the goal and is defined by the difference between the character's attraction point $\alpha(\mathbf{x})$ and the character's current position $\mathbf{x}$. For computing the attraction point, we first need to find the closest point on the backbone path. Given its associated disk, we compute $\alpha(\mathbf{x})$ as the furthest

intersection point between this disk and the control path. The line segment connecting $\mathbf{x}$ with $\alpha(\mathbf{x})$ is free of collisions, ensuring a collision-free path.

The boundary force $\mathbf{F}_b(\mathbf{x})$ pushes the character away from the corridor's boundary, and, hence, it keeps the character inside the corridor. If the character's clearance to the boundary is larger than its preferred safe distance then this force is $\mathbf{0}$. (We set the safe distance to the character's radius r.) Otherwise, the force is directed away from the boundary and its magnitude goes to infinity when the character touches the boundary.

The combined force is defined as $\mathbf{F}(\mathbf{x}) = \mathbf{F}_s(\mathbf{x}) + \mathbf{F}_b(\mathbf{x})$. When this force is applied to the character, it starts accelerating. This phenomenon is summed up by Newton's Second Law, i.e. $\mathbf{F} = M\mathbf{a}$, where M is the mass of the character and $\mathbf{a}$ is its acceleration. Without loss of generality, we assume that $M = 1$. Hence, the force can be expressed as $\mathbf{F}(\mathbf{x}) = \frac{d^2\mathbf{x}}{dt^2}\ m/s^2$. Combining the two expressions for $\mathbf{F}(\mathbf{x})$ gives us $\frac{d^2\mathbf{x}}{dt^2} = \mathbf{F}_s(\mathbf{x}) + \mathbf{F}_b(\mathbf{x})$, which is an equation providing the positions for the character. Because this equation cannot be solved analytically, we have to revert to a numerical approximation, such as Euler integration [10], to compute the positions of the character's path.

In Euler's integration scheme, the position $\mathbf{x}$ is updated by moving it during Δt time with velocity $\mathbf{V}$. The velocity is updated similarly. If its magnitude becomes larger then the maximum velocity v_{max}, then $\mathbf{V}$ is scaled such that its magnitude becomes equal to v_{max}. Consequently, the character's speed is limited to the maximum velocity. The new position is added to the path and is annotated with the current time stamp. Finally, the time is increased with the step size Δt and the loop continues.

The algorithm produces a smooth path, i.e. the path is C^1-continuous when $\Delta t \rightarrow 0$, because the positions are the result of a function that was doubly integrated [6]. We use such a smooth path as input for creating camera motions.

4 Following a Known Path

In this section we will present an algorithm which computes a *camera motion* for a path whose positions are known (or can be computed) in advance. Applications such as games (replay a path, follow an NPC) and architectural applications (create a virtual walk-through inside a city) can benefit from the approach.

A camera motion consists of two paths which comply to Definition 1, i.e. a camera path Π_{cam}, which describes the camera positions in the environment over time, and an aim path Π_{aim}, which describes the corresponding positions the camera looks at. The algorithm consists of the following steps.

The first step is to ensure that we have the right input. The next step is to find a proper initial camera placement if this is not given. Then, the camera and aim paths are computed. Next, additional constraints are incorporated into the algorithm. Finally, we handle the case in which a character stops moving, e.g. when the character reaches its destination.

Initialization. As input we have a known path which may be traversed by a character or may be a path observed by the camera. We call this the character

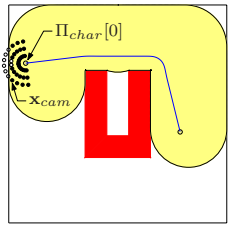


Fig. 2. Computing a collision-free initial placement of the camera. The camera can be placed on the black disks but not on the white disks.

path Π_{char}. In an architectural application, such a path may not be given, and, hence, we need to create a path from a given start to a given goal. We construct this path (and corresponding corridor) by the approach from the previous section. By constructing a path that has enough clearance, we can create camera motions that will not run too close to the obstacles. Also, having enough clearance will decrease the need for sharp turns and prevents views that are blocked by the obstacles. Since the path is also short and smooth, the final camera motions will be efficient and smooth too. If path Π_{char} is given in advance, we smooth it first for the same reasons mentioned above. Next, we compute a corridor $\mathcal{C}$ containing this path. Details on this procedure can be found in [8].

Besides path Π_{char} and corridor $\mathcal{C}$, we have the following input. The parameter d_{cam} denotes the preferred distance from the camera to the character. Next, the camera looks at a future position of the character. The corresponding look ahead time is denoted by t_{la}. Finally, we are given the maximum velocity v_{max} of the camera and aim. This value should be at least at large as the character's maximum velocity to assure that the camera can keep the character in view.

Before we create the camera path, we need to set the initial camera and aim positions. The camera initially looks at the character's start position. Since the camera stands still, its aim and camera velocity is set to $\mathbf{0}$.

Placement of the camera. If the camera position $\mathbf{x}_{cam}$ is not provided in advance, we need to choose a proper camera placement (see Fig. 2). The camera, modeled as a disk with radius r, does not have to be placed inside the corridor. We try to place the camera behind the character at the preferred distance d_{cam}. By 'behind' we mean a position on the line segment which starts at $\Pi_{char}[0]$ and is extended in the direction $\mathbf{v} = \Pi_{char}[0] - \Pi_{char}[\Delta t]$, where Δt is a small time step. The initial preferred placement is then given by $\mathbf{x}_{cam} = \Pi_{char}[0] + d_{cam} * \mathbf{v}/||\mathbf{v}||$. If this camera placement or the line through $\mathbf{x}_{cam}$ and $\Pi_{char}[0]$ is not inside the free space, we need to try other placements. We first consider a series of placements on the half circle with center $\Pi_{char}[0]$, radius d_{cam}, behind the line that is perpendicular to direction $\mathbf{v}$. If all placements collide, we decrease the radius (e.g. by $d_{cam}/3$) and test a new series of placements. We continue this procedure until we find a collision-free placement $\mathbf{x}_{cam}$. After finding $\mathbf{x}_{cam}$, we test whether $\mathbf{x}_{cam}$ is inside the corridor $\mathcal{C}$. If it is outside the corridor, we extract a new one enclosing the path from $\mathbf{x}_{cam}$ to $\Pi_{char}[0]$ and add it to $\mathcal{C}$.

Creating the camera and aim paths. Like in Section 3, we compute a smooth camera and aim path by iteratively updating their positions and time stamps until the last position of the character's path is being viewed by the camera. For the aim as well as the camera path, we define corresponding attraction points.

For the aim's attraction point $\alpha(\mathbf{x}_{aim})$ we take a future position on the character's path determined by the current time stamp t plus the time lapse t_{la}. Since the camera starts looking at a location in which the character will be in t_{la} time, the viewer gets a hint about the possible changes in direction, leading to a smaller chance of getting motion sick. The force, which will be applied to the aim position $\mathbf{x}_{aim}$ is computed like in Section 3. This force is also composed of a steering force (i.e. $\alpha(\mathbf{x}_{aim}) - \mathbf{x}_{aim}$) and boundary force. Next, we integrate the force function, compute the new aim position, and add it to the aim path.

The attraction point for the camera $\alpha(\mathbf{x}_{cam})$ is computed differently. The camera is attracted to its optimal position, which is located behind the character on the line which runs through the character and has the same direction as the current aim direction $\mathbf{v}$, where $\mathbf{v} = \Pi_{aim}[\Delta t] - \Pi_{aim}[t - \Delta t]$. Hence, $\alpha(\mathbf{x}_{cam}) = \Pi_{char}[t] - d_{cam} * \mathbf{v}/||\mathbf{v}||$. If $t = 0$, then the aim position $\Pi_{aim}[t - \Delta t]$ does not exist; we then use $\alpha(\mathbf{x}_{cam}) = \mathbf{x}_{cam}$ instead. We need to ensure that the camera can 'see' its attraction point, otherwise the camera could get stuck behind an obstacle. This can be checked by determining whether the line segment through $\mathbf{x}_{cam}$ and $\alpha(\mathbf{x}_{cam})$ is inside the corridor. If they cannot see each other, we iteratively decrease the preferred camera distance d_{cam} and update $\alpha(\mathbf{x}_{cam})$ correspondingly until the line segment is inside $\mathcal{C}$. The camera's steering and border forces are then computed like the aim's forces. These forces are applied to the camera, resulting in a new camera position which is added to the path.

Satisfying additional constraints. We have now described a procedure for creating camera motions which satisfy almost all criteria mentioned in the introduction. That is, the camera and aim paths are collision-free because they are kept inside the corridor by the boundary force. Moreover, this force assures a minimum clearance to the obstacles so that the view is blocked less by the obstacles. Next, the camera is placed as much as possible at a certain preferred distance behind the character to keep a good view. This criterion, together with the fact that the two paths change smoothly prevent motion-sickness.

Two constraints however may not have been satisfied yet. First, when the character is inside a narrow passage, it may not be in the camera's view. If the camera cannot see its corresponding aim point, we anticipate on the blocked view by temporarily increasing the camera's maximum velocity and decreasing the preferred distance d_{cam} to the camera. We increase the velocity v_{max} in each iteration until they see each other. Hence, the camera's speed will change smoothly. Besides, we iteratively decrease d_{cam}. When the character is in the camera's view again, we decrease v_{max} and increase d_{cam} until they have reached their initial values. As a result, the camera speeds up before the character would get out of view and the character stays (longer) in view. Nevertheless, increasing the camera's velocity may increase the camera's *angular velocity*, which may cause the view to change too fast. The angular velocity v_{rot} is defined by the

angle between the current camera direction and the previous one. If v_{rot} becomes larger than some predefined maximum angular velocity v_{rot_max}, we decrease the camera's maximum velocity, like we described in the previous paragraph, and increase the velocity again when $v_{rot} \leq v_{rot\_max}$. Note that these two constraints contradict each other. Depending on the application, we can favor one particular constraint. We will discuss this trade-off further in Section 6.

Reaching the destination. A nice property of modeling motion by forces is that the camera and aim smoothly accelerate until a they reach a certain maximum velocity. When the character (suddenly) stops moving, they both can have a high velocity, and, hence, they continue moving for a short while, which may result in oscillating motions near their steady attraction points. We handle this problem by appropriately slowing down the camera and aim during a short amount of time t_s (e.g. $t_s = 1$ second). We set the aim's attraction point to the character's current location (if it has not reached this location yet). The camera's attraction point is left unchanged. Next, during t_s time, we lower the maximum velocity like we did in the previous section. Such a scheme assures that the camera (and aim) smoothly decelerate within t_s seconds.

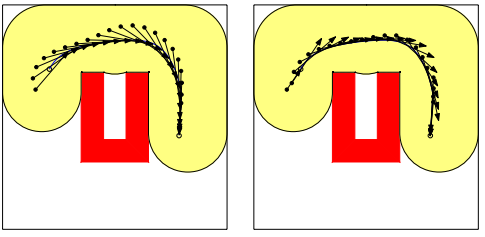
5 Following an Unknown Path

In third person games, a camera needs to follow a character which is controlled by the player. Consequently, the character's path is being created dynamically. We will adjust our approach such that the camera motions are also computed on the fly. This means that we cannot refer anymore to future character positions. This happens twice in our approach. First, the algorithm that finds the initial camera position assumes that the character's initial direction is known. Since we think this is a reasonable assumption, we require that this direction is given as input to the algorithm. Second, we need to adjust the computation of the aim's attraction point $\alpha(\mathbf{x}_{aim})$. We place $\alpha(\mathbf{x}_{aim})$ on its *estimated* future position. Starting from the character's current location $\Pi_{char}[t]$, we move in the aim's direction $\mathbf{v} = \Pi_{char}[t] - \Pi_{char}[t - \Delta t]$ with displacement $t_{la} * v_{max}$. If $t = 0$, we set $\alpha(\mathbf{x}_{aim})$ to the character's start position.

We have observed that good estimations are obtained when a smoothed version was used of the character's path instead of the original path. Each position (with time stamp t) is smoothed by computing an averaged position over a small time window t_w (e.g. $t_w = 0.25$ seconds).

Finally, the algorithm makes use of a corridor which is unknown in advance. We dynamically update the corridor while the character moves. That is, if the character moves out of the corridor (which initially is computed by the sequence of disks that connect the initial camera position with the initial character position), we add the new disk that encloses the character's new position.

We refer the reader to Fig. 3 which visualizes the effect of having knowledge (or not) about the character's future positions. To compare these effects, we used the same input character path. If the character's positions are *known* in advance (left picture), then the camera's motions anticipate well to changes of



(a) Known path (b) Unknown path

Fig. 3. Camera motions induced by knowing (or not) the character's path in advance

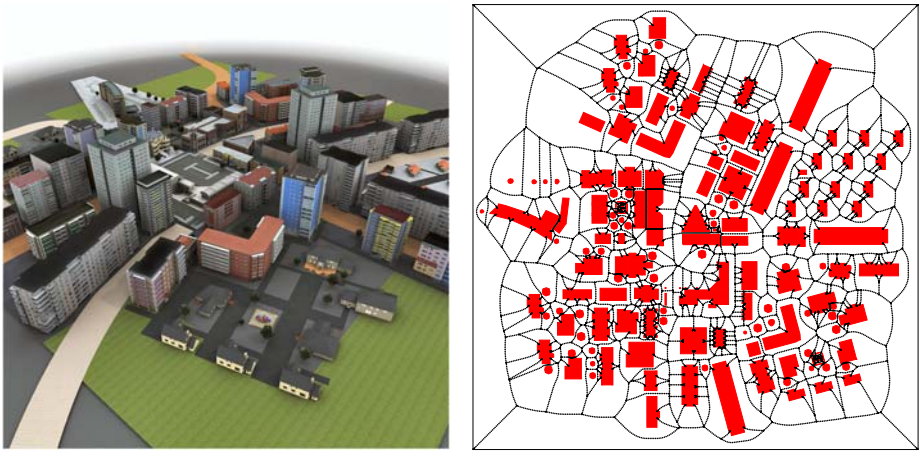
the character's future positions. That is, the character's positions (disks) are located behind the character as is required and its aim (arrows) is targeted at the character's near future positions. If the character's positions are *unknown* in advance, its future positions are estimated. Consequently, the camera motions react later on a change in the character's direction then the case in which the path is known in advance.

6 Experiments

We have tested our method inside a large virtual city. We experimentally verified whether the method can create high-quality camera motions in real-time. We implemented two applications in Visual C++ under Windows XP. The first one generated the Corridor Map of the city [7]. The second one was our CMM framework [6] in which we integrated camera planning. All the experiments were run on a PC with a NVIDIA GeForce 8800 GTX graphics card and an Intel Core2 Quad CPU (2.4 GHz) with 4 GB memory. Our application used one core.

We conducted experiments with the city environment depicted in Fig. 4. The city measured 500x500m. Its footprint (2,122 triangles) formed the input primitives for the Corridor Map. Creating the map took 0.64s, resulting in 1,434 vertices, 1,606 edges and 25,863 samples (i.e. the number of disks in all corridors). The characters' radius was set to $r = 0.25$ and the time step to $\Delta t = 0.1s$.

We performed two experiments. In the first one, we extracted 1,000 uniform random collision-free queries, yielding 1,000 character paths. Each path was respectively treated as a path that was known or unknown in advance. We recorded the average running times for both cases. To view these times in the right perspective, we defined the CPU-load, which is the spent CPU time / averaged traversed time $*$ 100%. We set the character's maximum velocity to $2m/s$ which corresponds to a fast walking speed [11]. We considered a CPU-load of 1% to be real-time. In the second experiment, we selected one representative query. Its start was located in the lower part while its goal part was located in the upper part of the city. Then, a corridor, enclosing the query, was extracted from the Corridor Map [7]. The first half of the corridor corresponded to places with much clearance and little curvature while the second half corresponded to places with



(a) City environment (b) Footprint and Corridor Map

Fig. 4. The test environment, its footprints and corridor map

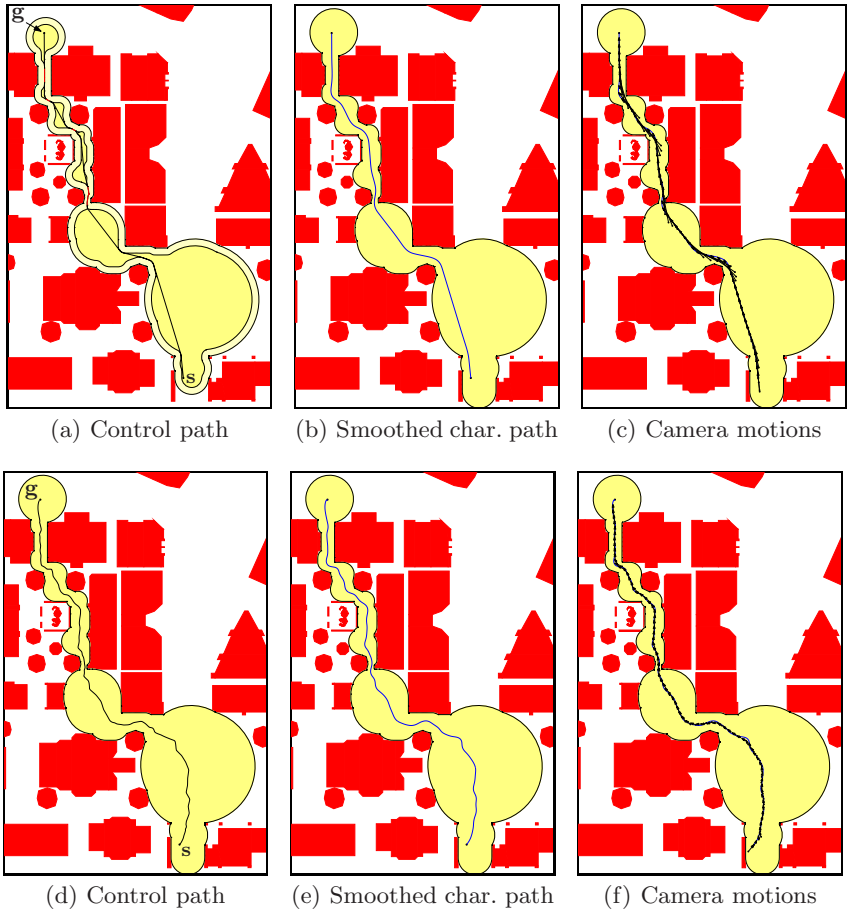
relatively little clearance and much curvature. Clearly, creating good camera motions will be more difficult in the second half.

We considered two cases. In the first case, we created camera motions for a virtual walk-through (so the character's path is known in advance). Its control path was constructed by extracting the shortest minimum clearance path inside this corridor [9]. By using much clearance (i.e. $3m$), a good overview of the environment was to be expected. We set the camera distance d_{cam} to $5m$, the time lapse t_{la} to $2s$ and maximum velocity v_{max} to $2m/s$. These settings favored looking ahead and anticipated well to future directional changes. In addition, we used a low maximum angular velocity (i.e. $v_{rot\_max} = 2m/s$) at the expense of higher velocities at straight parts of the path. These settings favor a slow change of the user's rotating view.

In the second case, we created camera motions for following a character inside a third-person game (so the character's path is unknown in advance). We simulated a user-controlled path by creating a control path with many local changes [8]. In addition, the path sometimes ran close the obstacles, leading to an increased chance for a bad view. We used a shorter camera distance (i.e. $3m$) to keep a good third-person view and used a smaller time lapse (i.e. $0.5s$) to minimize the errors induced by wrongly estimating the future directions. We used $v_{rot\_max} = 5m/s$ to favor fast reactions on the character's motions.

Both control paths were smoothed. The algorithm created an aim and camera path, annotated with time stamps. The two paths were composed of x- and y-coordinates. A z-coordinate was added (i.e. $z = 1.8m$) to place the camera above the ground. We will now discuss the results.

For the known character path, the 1,000 queries were computed in $27.3ms$ on average. The averaged traversed time was $105.2s$. Hence, the CPU-load was 0.026%. For the unknown character path, the queries were computed in $29.6ms$



(a) Control path (b) Smoothed char. path (c) Camera motions

(d) Control path (e) Smoothed char. path (f) Camera motions

Fig. 5. Creating camera motions. A virtual walk-through is constructed for a *known* path (first row). A character's *unknown* path is followed (second row).

and the traversed time was 105.2s, resulting in a CPU-load of 0.028%. Consequently, the running times can be considered as real-time.

The top row of Fig. 5 shows the results for the virtual walk-through. We used the smoothed control path to create the camera motions. The camera (disks) and aim path (arrows), as well as the view (i.e. the line connecting the camera with its aim) was collision-free as none of these glyphs intersected with the obstacles. The camera motions were smooth, i.e. there were no sudden changes in positions or orientations, even not at the locations with little clearance and high curvature. Since the latter is hard to see, we created a movie which can be viewed at `http://people.cs.uu.nl/roland/motion_planning/camera`. The movie confirms that the motions were smooth. The maximum angle constraint reduced the camera's speed wherever required, leading to coherent motions. After a sharp

corner, the camera's velocity was temporarily increased to make up for the delay. Finally, by anticipating on the character's positions, it was not lost out of sight.

The bottom row Fig. 5 shows the results for a dynamic third-person camera. The control path was rather bumpy. In each iteration, a small part of this path was smoothed. The final smoothed path is displayed in Fig. 5(e). By using smoothed character positions, relatively smooth camera motions were created as can be seen in Fig. 5(f). The figure also shows that no collisions occurred. We refer the reader to the corresponding movie. Is makes clear that the motions are indeed smooth. However, the view moved faster than in the case in which the character's path was known in advance, because the character's future positions were being estimated while constructing the camera motions. However, these estimations and a small camera distance still made sure that the character was centered in the view. Consequently, the viewer would not get disoriented.

7 Conclusions

We introduced a new method for planning *good* camera motions in real-time. Good motions are composed of camera and aim positions which are collision-free and keep the character in clear view. Camera planning is a challenging problem because a sequence of good camera motions may not be continuous while continuity of the user's view (i.e. *frame coherence*) is of major importance. By smoothing the character's positions, looking ahead, slowing down when the view's angle gets too large, and by modeling the camera motions by forces, we have obtained good camera motions while keeping frame coherence. These motions were generated for both paths that were known as well as paths that were unknown in advance. Real-time performance was achieved by using the Corridor Map for answering collision and visibility queries.

We tested our approach on a static flat environment. We will extend the method such that a clear view is maintained as much as possible when dynamic changes occur, e.g. when another character blocks the user's view. Then, we would also like to handle 2.5D environments which include terrains and bridges.

References

1. Li, T.Y., Cheng, C.C.: Real-time camera planning for navigation in virtual environments. In: Butz, A., Fisher, B., Krüger, A., Olivier, P., Christie, M. (eds.) SG 2008. LNCS, vol. 5166, pp. 118–129. Springer, Heidelberg (2008)
2. Nieuwenhuisen, D., Overmars, M.: Motion planning for camera movements. In: IEEE International Conference on Robotics and Automation, pp. 3870–3876 (2004)
3. Halper, N., Helbing, R., Strothotte, T.: A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. Eurographics 20, 174–183 (2001)
4. Li, T.Y., Yu, T.H.: Planning tracking motions for an intelligent virtual camera. In: IEEE International Conference on Robotics and Automation, pp. 1353–1358 (1999)

5. Bourne, O., Sattar, A.: Evolving behaviours for a real-time autonomous camera. In: Australasian conference on Interactive entertainment, pp. 27–33 (2005)
6. Geraerts, R., Overmars, M.: The corridor map method: A general framework for real-time high-quality path planning. Computer Animation and Virtual Worlds 18, 107–119 (2007)
7. Geraerts, R., Overmars, M.: Enhancing corridor maps for real-time path planning in virtual environments. In: Computer Animation and Social Agents, pp. 64–71 (2008)
8. Overmars, M., Karamouzas, I., Geraerts, R.: Flexible path planning using corridor maps. In: Halperin, D., Mehlhorn, K. (eds.) Esa 2008. LNCS, vol. 5193, pp. 1–12. Springer, Heidelberg (2008)
9. Geraerts, R.: Planning short paths with clearance using explicit corridors. In: IEEE International Conference on Robotics and Automation (submitted, 2010)
10. Butcher, J.: Numerical Methods for Ordinary Differential Equations. Wiley, Chichester (2003)
11. Knoblauch, R., Pietrucha, M., Nitzburg, M.: Field studies of pedestrian walking speed and start-up time. Transportation Research Record, 27–38 (1996)

Adaptive Physics–Inspired Facial Animation

Lihua You, Richard Southern, and Jian J. Zhang*

National Center for Computer Animation
Bournemouth University, United Kingdom
jzhang@bmth.ac.uk

Abstract. In this paper, we present a new approach for facial animation. We develop a mathematical model from the physical properties of skin deformation which incorporates the action of externally applied forces and the material properties of the skin's surface. A finite difference mesh which uses this model is generated automatically using a harmonic parametrization and interpolating nodes on the original surface. We determine the forces at these nodes of various face poses. By blending these forces at we can generate new intermediate shapes. In the interests of computational efficiency, we present a novel adaptive finite difference method which limits the calculation of surface constants to regions where significant deformation occurs.

1 Introduction

Facial animation is an active topic in computer animation attracting significant research attention. According to the survey of Deng and Neumann [3], facial modeling and animation can be divided into blend shapes, parametrization, facial action coding, deformation–based approaches, physics–based muscle modeling, 3D face modeling, performance–driven facial animation, MPEG–4 facial animation, facial animation editing, and facial transferring.

Ersotelos and Dong [6] divide face modeling into two large groups: generic model and example-based face modeling, and divide facial animation into three main approaches: simulation–based, performance-driven, and blend shapes. Partial differential equations (PDEs) based approaches have also been introduced into facial modeling and animation. Since the partial differential equations used are homogeneous, the forces and the related physics of skin deformations were not introduced[18].

Physics-based approaches consider physics of facial deformations and have a potential to create more realistic shapes. However, current physics-based approaches are computationally expensive and are not ideal in facial animation which demands high computational efficiency. The deformation of the skin's surface is affected by the underlying anatomical structures, such as fatty tissue, muscle and skin. The actions of these structures can be described with the forces which are determined by investigating physical deformations of these anatomical structures.

* Correspondence to: Jian J. Zhang, The Media School, Bournemouth University, Poole BH12 5BB, UK.

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 207–218, 2009.

As the first step towards improving efficiency, we investigate physics–based skin deformations, an essential element in physics–based facial animation considering real anatomical structure. We intend to integrate the actions of other anatomical structures in our future research.

Our approach has the following advantages:

1. Our method introduces a physical model of skin deformation to the problem of facial animation, including material properties and the forces driving deformation.
2. In facial animation using shape blending, the interpolation of a number of topologically conforming shape primitives does not follow any underlying physical properties. In order to generate physically meaningful facial animation we blend forces using our physics–based approach.
3. We introduce an adaptive analysis method to significantly reduce the computation time. This approach decomposes a very large finite difference analysis problem into a series of very small ones which can be solved separately.
4. The physics–based model is computed automatically by parameterizing the input model into $\mathbb{R}^2$. The finite difference mesh is determined by sampling the 2D domain and interpolating these nodes on the 3D face model.

2 Method Overview

Given an input surface mesh, we first deduce finite difference mesh from the input triangulation by a regular sampling in the parametric domain (see Section 5.1). This mesh is adaptively refined using the adaptive analysis technique of Section 5.3. Forces affecting nodes of the finite difference mesh are computed by the method in Section 4.2. By interpolating these forces we can generate a facial animation by using force blend shapes (see Section 6). The surface mesh is reconstructed from the finite difference mesh by using an anisotropic interpolation technique (see Section 5.2).

3 Related Work

We limit our discussion of related work to references which relate to works directly related to ours. For a more complete discussion of facial animation techniques, the reader is referred to the survey by Deng and Neumann [3].

Platt and Badler [16] initiated the work of physics–based facial modeling and animation. In their work, forces are applied to elastic meshes through muscle arcs to generate various facial expressions. Waters [22] linked the movement of each pseudo muscle to mouth area of a face model which potentially avoids the facial distortion created by Platt and Badler [16]. By applying the discrete Lagrange equations of motion, Terzopoulos and Waters [20] presented a new model which uses three deformable mesh respectively corresponding to skin, fatty tissue, and muscle tied to bone.

Wu et al. [23] presented a simplified mesh system to reduce the computation time. The physics-based model proposed by Lee et al. [11] considers five different layers: epidermis, dermis, sub-cutaneous connective tissue, fascia and muscles. Sifakis et al. [19] developed a highly non-linear volumetric modeling method involving controllable anisotropic muscle activations based on fiber directions. Zhang et al. [25] presented a novel multi-layer deformation approach to reconstruct animatable, anatomy-based human facial models. Most physically–based methods of facial animation are dependent on finite element meshes[13] or mass–spring systems which are computationally expensive and difficult to set up. Our approach in comparison is completely automatic, and because deformation is force–based, can easily incorporate the effects of underlying anatomical structures such as muscle and bone.

For shape blending, linear interpolation is most popular[14, 24], although bilinear interpolation can create a wider range of facial expression changes[1]. Morphable face modeling[2] and multi-linear face modeling[21] are more powerful and can generate more face models from an example set of 3D face models with minimal user input. Unlike shape blending, our proposed approach creates intermediate shapes by blending forces which are physically meaningful and therefore more realistic.

Partial differential equations (PDEs) were introduced into facial modeling and animation by Gonzalez Castro et al. [9], and was later extended in several papers[5, 17, 8]. All these publications use homogeneous fourth order partial differential equations without involving the physics of facial skin deformation. In comparison with existing PDE–based methods, our approach is based on the physical model of skin deformation which considers the effects of externally applied forces and material properties of facial skin on facial animation.

4 Finite Difference Model for Facial Animation

In this section we will investigate the physical model and mathematical model of facial animation and present the finite difference solution of the mathematical model.

4.1 Physical Model

While not physically accurate, for the purpose of deriving the deformable model for facial skin we assume that the deformation of each of position components x, y and z relative to the parametric (u, v) plane is similar to a thin plate under bending. Since skin surfaces will deform in all x, y and z directions, this physical model can describe not only bending but also stretching deformations of facial skin.

Considering an arbitrary 3D parametric skin surface which has positional components x, y and z. Each such component is the function of parametric variables u and v. Taking the x component as an example and considering an infinitesimal element of $du \times dx$ taken from a surface under deformation as indicated in Figure 1, the externally applied force is represented by F_x. This force will causes internal forces which are shear forces Q_u and Q_v, bending moments M_u and M_v, and twisting moment $M_{uv} = M_{vu}$.

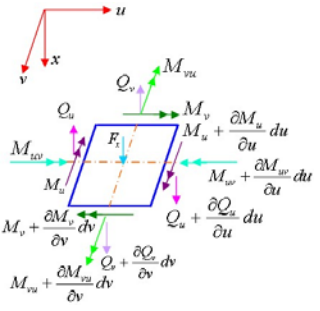


Fig. 1. External and internal forces acting on an infinitesimal element of a skin surface

4.2 Mathematical Model

From the physical model described above, we can derive the mathematical model of facial animation. For doing this, we take the equilibrium of moments around the u and v axes, respectively, and obtain

$$Q_u = \frac{\partial M_u}{\partial u} + \frac{\partial M_{vu}}{\partial v} \quad \text{and} \quad Q_v = \frac{\partial M_v}{\partial v} + \frac{\partial M_{uv}}{\partial u} \tag{1}$$

In addition to the equilibrium of moments, we should also consider the force equilibrium along x-axis which generates the following equation

$$\frac{\partial Q_u}{\partial u} + \frac{\partial Q_v}{\partial v} + F_x = 0 \tag{2}$$

Differentiating the shear force Q_u with respect to parametric variable u , and the shear force Q_v with respect to parametric variable v, and substituting them into Equation 2, we obtain

$$\frac{\partial^2 M_u}{\partial u^2} + 2\frac{\partial^2 M_{uv}}{\partial u \partial v} + \frac{\partial^2 M_v}{\partial v^2} + F_x = 0 \tag{3}$$

Although the skin is a visco–elastic, anisotropic material, its deformation can be well predicted by isotropic elastic properties. According to the theory of bending of isotropic elastic plates, the bending moments M_u and M_v, twisting moment M_{uv} and position function x hold the following relations

$$\begin{aligned} M_u &= -D\left(\frac{\partial^2 x}{\partial u^2} + \mu \frac{\partial^2 x}{\partial v^2}\right) \\ M_v &= -D\left(\frac{\partial^2 x}{\partial v^2} + \mu \frac{\partial^2 x}{\partial u^2}\right) \\ M_{uv} &= -(1-\mu) D \frac{\partial^2 x}{\partial u \partial v} \end{aligned} \tag{4}$$

where $D = Eh^3/\left(12(1-\mu^2)\right)$, E and μ are Young's modulus and Poisson's ratio respectively which are the material properties of the skin surface, and h is the thickness of the skin surface.

Substituting (4) into (3), the position function x is related to the externally applied force F_x, Young's modulus E and Poisson's ratio μ. Then we introduce shape control parameters S_{1x}, S_{2x} and S_{3x} such that $S_{1x} = S_{3x} = D$ and $S_{2x} = 2D$ which are determined by the material properties of the skin surface and influence the shape of a skin surface.

Equation 3 is therefore transformed into the following fourth order partial differential equation

$$S_{1x}\frac{\partial^4 x}{\partial u^4} + S_{2x}\frac{\partial^4 x}{\partial u^2 \partial v^2} + S_{3x}\frac{\partial^4 x}{\partial v^4} = F_x \tag{5}$$

Similar equations can be derived for y and z components. By putting the fourth order partial differential equations for the x, y and z components in a vector-valued form, we reach the following equation

$$\boldsymbol{s}_1\frac{\partial^4 \boldsymbol{x}}{\partial u^4} + \boldsymbol{s}_2\frac{\partial^4 \boldsymbol{x}}{\partial u^2 \partial v^2} + \boldsymbol{s}_3\frac{\partial^4 \boldsymbol{x}}{\partial v^4} = \boldsymbol{f} \tag{6}$$

where $\boldsymbol{x}$ is a vector-valued position function which has three positional components x, y and z, u and z are parametric variables, $\boldsymbol{s}_1$, $\boldsymbol{s}_2$ and $\boldsymbol{s}_3$ are vector–valued shape control parameters, and $\boldsymbol{f}$ is a vector-valued force function which has three components F_x, F_y and F_z.

We represent the surface of the human face as a regular grid of four–sided patches. In the applications of surface modeling, positional and tangential continuities are most frequently required. A four-sided patch satisfies these boundary constraints, which can be written as

$$\begin{aligned}
u = 0 \qquad & \boldsymbol{x} = \boldsymbol{g}_0(v) \qquad & \frac{\partial \boldsymbol{x}}{\partial u} = \boldsymbol{g}_1(v) \\
u = 1 \qquad & \boldsymbol{x} = \boldsymbol{g}_2(v) \qquad & \frac{\partial \boldsymbol{x}}{\partial u} = \boldsymbol{g}_3(v) \\
v = 0 \qquad & \boldsymbol{x} = \boldsymbol{g}_4(u) \qquad & \frac{\partial \boldsymbol{x}}{\partial v} = \boldsymbol{g}_5(u) \\
v = 1 \qquad & \boldsymbol{x} = \boldsymbol{g}_6(u) \qquad & \frac{\partial \boldsymbol{x}}{\partial v} = \boldsymbol{g}_7(u)
\end{aligned} \tag{7}$$

where $\boldsymbol{g}_0(v)$, $\boldsymbol{g}_2(v)$, $\boldsymbol{g}_4(u)$ and $\boldsymbol{g}_6(u)$ are the boundary curves, and $\boldsymbol{g}_1(v)$, $\boldsymbol{g}_3(v)$, $\boldsymbol{g}_5(u)$ and $\boldsymbol{g}_7(u)$ are the boundary tangents.

Deformation of the face model is achieved by applying a force $\boldsymbol{f}$ and solving Equations 6 and 7. This process represents our model for facial animation.

4.3 Finite Difference Solution

There are three methods to calculate partial derivatives using finite differencing: *forward*, *central* and *backward*. Among them, central difference algorithm is most common. Using central differencing with the mesh of Figure 2, the first and fourth partial derivatives can be written as

$$\begin{aligned}
\left(\frac{\partial \boldsymbol{x}}{\partial u}\right)_0 &= \frac{\boldsymbol{x}_1 - \boldsymbol{x}_3}{2\delta} \\
\left(\frac{\partial \boldsymbol{x}}{\partial v}\right)_0 &= \frac{\boldsymbol{x}_2 - \boldsymbol{x}_4}{2\delta} \\
\left(\frac{\partial^4 \boldsymbol{x}}{\partial u^4}\right)_0 &= \frac{6\boldsymbol{x}_0 - 4(\boldsymbol{x}_1 + \boldsymbol{x}_3) + (\boldsymbol{x}_9 + \boldsymbol{x}_{11})}{\delta^4} \\
\left(\frac{\partial^4 \boldsymbol{x}}{\partial u^2 \partial v^2}\right)_0 &= \frac{4\boldsymbol{x}_0 - 2(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3 + \boldsymbol{x}_4)}{\delta^4} \\
&\quad + \frac{\boldsymbol{x}_5 + \boldsymbol{x}_6 + \boldsymbol{x}_7 + \boldsymbol{x}_8}{\delta^4} \\
\left(\frac{\partial^4 \boldsymbol{x}}{\partial v^4}\right)_0 &= \frac{6\boldsymbol{x}_0 - 4(\boldsymbol{x}_2 + \boldsymbol{x}_4) + (\boldsymbol{x}_{10} + \boldsymbol{x}_{12})}{\delta^4}
\end{aligned} \tag{8}$$

where the subscripts $0, 1, 2, \ldots, 12$ are the index of nodes, and δ is the interval between two adjacent nodes. Substituting the last three lines of Equation 8 into Equation 6, the finite difference equation at node 0 is found to be

$$\begin{aligned}
&2(3\boldsymbol{s}_1 + 2\boldsymbol{s}_2 + 3\boldsymbol{s}_3)\boldsymbol{x}_0 - 2(2\boldsymbol{s}_1 + \boldsymbol{s}_2)(\boldsymbol{x}_1 + \boldsymbol{x}_3) \\
&-2(\boldsymbol{s}_2 + 2\boldsymbol{s}_3)(\boldsymbol{x}_2 + \boldsymbol{x}_4) + \boldsymbol{s}_2(\boldsymbol{x}_5 + \boldsymbol{x}_6 + \boldsymbol{x}_7 + \boldsymbol{x}_8) \\
&+\boldsymbol{s}_1(\boldsymbol{x}_9 + \boldsymbol{x}_{10} + \boldsymbol{x}_{11} + \boldsymbol{x}_{12}) = \delta^4 \boldsymbol{f}_0
\end{aligned} \tag{9}$$

Substituting the first two lines of Equation 8 into Equation 9, the boundary constraints are transformed into

$$\begin{array}{lll}
u = 0 & \boldsymbol{x}_0 = \boldsymbol{g}_{0,0} & \boldsymbol{x}_1 - \boldsymbol{x}_3 = 2\delta \boldsymbol{g}_{1,0} \\
u = 1 & \overline{\boldsymbol{x}}_0 = \boldsymbol{g}_{2,0} & \overline{\boldsymbol{x}}_1 - \overline{\boldsymbol{x}}_3 = 2\delta \boldsymbol{g}_{3,0} \\
v = 0 & \boldsymbol{x}_0 = \boldsymbol{g}_{4,0} & \boldsymbol{x}_2 - \boldsymbol{x}_4 = 2\delta \boldsymbol{g}_{5,0} \\
v = 1 & \overline{\boldsymbol{x}}_0 = \boldsymbol{g}_{6,0} & \overline{\boldsymbol{x}}_2 - \overline{\boldsymbol{x}}_4 = 2\delta \boldsymbol{g}_{7,0}
\end{array} \tag{10}$$

where the overbar is used to identify different boundaries.

For each inner node, we can compute the partial derivatives using Equation 8. Putting Equation 9 for all inner nodes and Equation 10 together the resulting linear algebraic equations can be posed in a matrix form

$$\mathbf{KX} = \mathbf{F}. \tag{11}$$

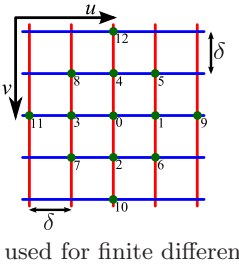


Fig. 2. The mesh used for finite difference approximation

We can determine all the unknown constants by solving Equation 11. We use this result to determine the shape of the skin surface.

5 Determining the Finite Difference Mesh

In this section, we discuss how to generate the finite difference mesh from a 3D polygonal facial model through surface sampling and how to transfer the deformation of the finite difference nodes to that of vertices of the polygonal model.

5.1 Finite Difference Mesh Sampling

Here we present a method to determine the quadrilateral patches used by the finite difference method. Given an input surface mesh, we embed this into $\mathbb{R}^2$ using a harmonic mapping. Nodes on our quadrilateral grid are determine by a regular sampling in the parametric domain, and the corresponding position in $\mathbb{R}^3$ is deduced using interpolation from the parametric domain.

An input surface with a boundary (such as the one in Figure 3(a)) is embedded into $\mathbb{R}^2$ using the harmonic map parametrization of Eck et al. [4]. While not conformal, harmonic mappings minimize the discrete Dirichlet energy and are fast to compute [7]. It is easily formulated using the discrete Laplace-Beltrami operator:

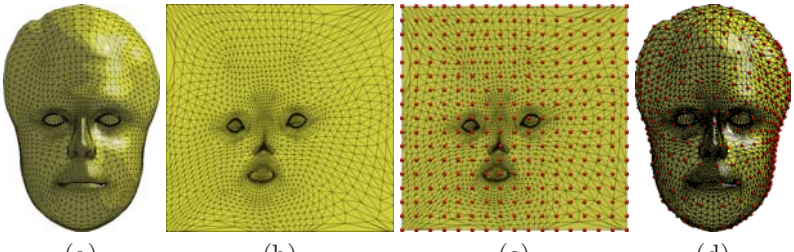
$$\delta_i = \boldsymbol{p}_i - \sum_{j \in \mathcal{N}(i)} w_{ij} \boldsymbol{p}_j, \tag{12}$$

where $\boldsymbol{p}_i$ is a one of n vertices in the surface mesh, $\mathcal{N}(j)$ is the set of vertex indices within the 1-ring neighborhood of vertex i, and w_{ij} is a weight assigned to the edge (i, j). The vector δ_i encodes a *local feature*.

We use the popular cotangent weights [15]

$$w_{ij} = \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) \tag{13}$$

where α_{ij} and β_{ij} are the angles opposite the edge (i, j).



(a) (b) (c) (d)

Fig. 3. A face model (a) is flattened into $\mathbb{R}^2$ (b) using a harmonic mapping. In (c) A 16×16 grid is sampled in the 2D domain and then interpolated on the original surface using barycentric coordinates (d).

Embedding a triangulation into $\mathbb{R}^2$ is achieved by building a system from Equation 12 for the unconstrained vertices with the local feature $\delta_i = [0,0]^T$, and $\boldsymbol{p}_i = c_i$ for nodes that are fixed along some predefined boundary. We use a unit square in $\mathbb{R}^2$ for our boundary constraints c_i.

Using this method we embed the input surface into $\mathbb{R}^2$ (see Figure 3). Our finite difference mesh is then defined by a regular grid in this 2D parametric domain. We refer to the corners of this grid as our $m \times m$ *nodes*. The corresponding positions of these nodes in $\mathbb{R}^3$ are the 3D nodes $\boldsymbol{x}$. These are computed using the barycentric coordinates of the node points in the 2D triangulation, shown in Figure 3(d).

5.2 Surface Reconstruction

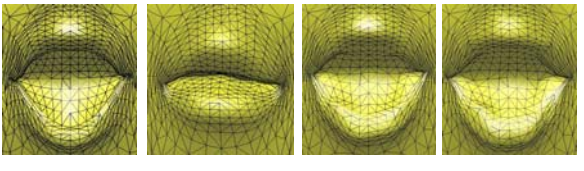
With finite difference mesh obtained in Section 5.1 and the finite difference solution developed in Section 4.3, we determine the deformations of the nodes on a 3D human face model. The remaining task is to determine how these deformations can be transferred to the vertices of the 3D polygonal model.

Using the parametrization of Section 5.1 we can define the barycentric coordinates of the original vertices in the finite difference mesh patches in $\mathbb{R}^2$. To avoid problems arising from non–planar quadrilaterals, we use a regular triangulation of the mesh. The positions of the reconstructed vertices can be written in matrix form as

$$\hat{\mathbf{P}} = \mathbf{BX} + \mathbf{D} \tag{14}$$

where $\mathbf{B}$ is a $n \times m$ matrix containing the barycentric coordinates for $\boldsymbol{p}_i$ on each row, $\mathbf{X}$ contains the positions of the nodes of the finite difference mesh, and $\mathbf{D}$ contains the displacement vectors $\boldsymbol{d}_i = [\mathbf{BX}]_i - \boldsymbol{p}_i^0$, $\boldsymbol{p}_i^0$ is the rest position of the vertex.

It is important to note that because $\mathbf{D}$ is directional, the surface reconstructed using Equation 14 is isotropic. In order to ensure an anisotropic reconstruction, we apply a rotation to each individual row of $\mathbf{D}$ based on the rotation of the containing face in the finite difference mesh. The results of this operation are



(a) Ground truth (b) Undersampling (c) Isotropic (d) Anisotropic

Fig. 4. A comparison of facial reconstruction. In (b) the surface is reconstructed using the same sample points as in Figure 3(d), where the finite difference mesh is not sufficiently dense to represent the deformation. The results of (c) and (d) were reconstructed using a 16×16 grid sampled just in the region of the mouth. Note that while anisotropic reconstruction better preserves the shape of the mouth region, isotropic reconstruction generally produces a more regular mesh sampling.

demonstrated in Figure 4. While anisotropic reconstruction does better match the overall shape of the original surface, it can negatively effect the regularity of the resulting mesh (for example, the meshing of the upper lip appears more regular without using rotation). In addition, with severe undersampling, rotating the deformation vector may cause fold–over on the mesh.

5.3 Adaptive Analysis

For a polygonal human face, many vertices are required to describe the details of the eyes, nose and mouth. The finite difference calculation of all vertices on a human face would be a time consuming process, as a large number of constants need to be determined. To address this issue, we introduce adaptive finite difference analysis. For each finite difference calculation, only a few nodes are involved and high computational efficiency is achieved. We will introduce the method below.

The face model in Figure 3(a) has 2464 vertices. Initially we choose the whole face as a deformation region and divide the parametric domain into a $m \times m$ grid. Usually, m is a small number (in Figure 3, $m = 16$). We apply forces on the nodes of the grid and use the finite difference algorithm to determine the deformation of these nodes. The surface is reconstructed, and the L^2 error between the original and the reconstructed surface is computed. If the error is above a user specified threshold, the subregion is further sampled and the process is repeated.

A problem arises when "stitching" irregularly sampled surfaces together along the boundary. In order to ensure the patch boundaries are aligned, we reconstruct the surface from coarse to fine, and compute the node positions on fine detail patches by interpolating node positions on the coarse reconstruction. This ensures that there are no discontinuities in the reconstruction at the patch boundaries.

6 Results and Discussion

Blend shapes (also called *morph targets*) is the most common and intuitive technique used in facial animation. It creates a blend shape by using a linear weighted sum of topologically conforming shape primitives. A particular problem with this approach is that as the transitions between targets are linear, they do not conform to any underlying physical laws, and in–between shapes may not be realistic.

In Figure 5 we interpolate between the mouth closed and mouth open shapes. The forces applied to the nodes in each example was determined using Equation 6. A linear blend was used to interpolate between the forces in each frame. As our blending model is based on the forces applied to nodes of the face, our approach is applicable to anatomical structures or colliding objects which may exert external forces.

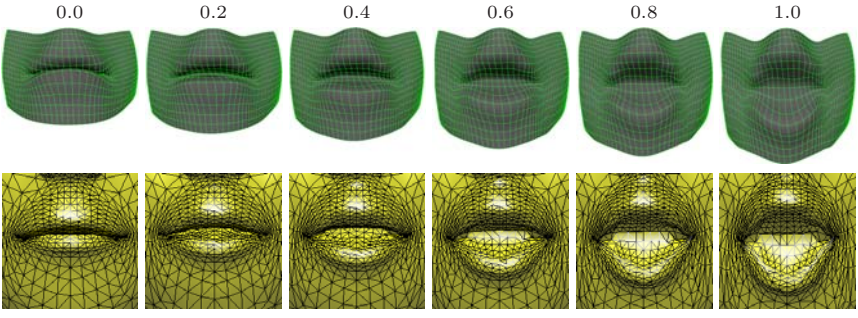


Fig. 5. Force blend shapes of the finite difference mesh and the corresponding surface reconstruction for the mouth region. The L^2 error for the reconstructed mesh is negligible ($\varepsilon < 10^{-16}$).

The finite difference solver was implemented in C++ using unoptimized linear solvers. The automatic finite difference mesh calculation, including the parametrization, adaptive sampling and reconstruction was implemented as a set of Matlab scripts. As our system was not implemented as a single system timing results are difficult to produce. However, we will show that the performance of this system is real–time.

The mesh sampling is performed as a preprocess, but the only step of significance is the solving of the harmonic parametrization which requires the solution of a sparse linear system. The matrix $\mathbf{K}$ is computed from the regular grid, and the force matrix $\mathbf{F}$ is precomputed for each input shape using Equation 11. These steps only require simple matrix multiplications, which can be implemented on graphics hardware using CUDA [12].

At run time the force blend shapes are determined by first computing the weighted sum of the forces of the input shapes to find $\mathbf{F}$, after which the positions of the nodes are computed using Equation 11, where $\mathbf{K}$ is a sparse, $m \times m$ matrix, (m is the number of nodes in the finite difference mesh). This can be quickly solved on the GPU using the conjugate gradient method [10], although we have not implemented this feature. Finally, the locations of the vertices of the reconstructed mesh are computed using Equation 14. While this is represented as a sparse matrix multiplication, it is easy to implement the reconstruction stage as a vertex program on the GPU, since each of the reconstructed vertices are represented as the sum of a weighted sum of the nodes at the face corners and a single (possibly rotated) displacement vector.

7 Conclusions

In this paper we have presented a novel adaptive finite difference method for human facial deformation. By incorporating the physical properties of skin deformation into facial animation we derive a mathematical model which can be used to drive facial deformation. We compute the solution to this physics–based skin model by deriving numerical equations using the finite difference method.

The finite difference mesh used in the above equations is determined by flattening an arbitrary input model into the 2D parametric domain. Nodes on a regular grid in this domain are sampled on the original surface using barycentric interpolation. The deformations which are applied to these nodes are deduced by using the same interpolation values on deformed examples. In order to improve the efficiency of physics–based facial animation based on the numerical finite difference solution, we present an adaptive analysis method. Each patch is adaptively subdivided based on the error between the reconstructed surface and the original mesh, removing the need to perform unnecessary finite differencing operations in regions where there is little or no deformation.

There are a number of areas of future work which can improve the results presented in this paper. A particularly important problem is the influence of bones, fatty tissue and muscles on the facial deformation, and how this can be incorporated into our current approach.

Acknowledgements

Many thanks to Gavin Lewis for providing the head model used in our examples. We are grateful to the anonymous reviewers for their insightful comments. This research is in part supported EPSRC grant EP/F030355/1.

References

[1] Arai, K., Kurihara, T., Anjyo, K.: Bilinear interpolation for facial expression and metamorphosis in real-time animation. The Visual Computer 12, 105–116 (1996)
[2] Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH), New York, pp. 187–194 (1999)
[3] Deng, Z., Neumann, U.: Data-driven 3D Facial Animation. Springer, London (2008)
[4] Eck, M., DeRose, T.D., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution analysis of arbitrary meshes. In: Proceedings of SIGGRAPH, pp. 173–182 (1995)
[5] Elyan, E., Ugail, H.: Reconstruction of 3d human facial images using partial differential equations. Journal of Computers 2(8), 1–8 (2007)
[6] Ersotelos, N., Dong, F.: Building highly realistic facial modelling and animation: a survey. The Visual Computer 24, 13–30 (2008)
[7] Floater, M.S., Hormann, K.: Surface Parameterization: a Tutorial and Survey. In: Dodgson, N.A., Floater, M.S., Sabin, M.A. (eds.) Advances in Multiresolution for Geometric Modelling, pp. 157–186. Springer, Heidelberg (2005)
[8] Gonzàlez Castro, G., Ugail, H., Willis, P., Sheng, Y.: Parametric representation of facial expressions on PDE-based surfaces. In: Proc. of the 8th IASTED Int. Conf. on Visualization, Imaging, and Image Processing, pp. 402–407 (2008)
[9] Gonzalez Castro, G.G., Ugail, H., Willis, P., Palmer, I.: Shape morphing using PDE surfaces. In: Villanueva, J. (ed.) Visualization, Imaging and Image Processing, pp. 553–558. ACTA Press (2006)

[10] Ian, J.B., Farmer, I., Grinspun, E., Schrder, P.: Sparse matrix solvers on the gpu: Conjugate gradients and multigrid. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 22, 917–924 (2003)
[11] Lee, Y., Terzopoulos, D., Waters, K.: Realistic modeling for facial animation. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH), New York, pp. 55–62 (1995)
[12] NVidia: CUDA Zone (September 2009), http://www.nvidia.com/object/cuda_home.html
[13] Pieper, S., Rosen, J., Zeltzer, D.: Interactive graphics for plastic surgery: a task-level analysis and implementation. In: SI3D 1992: Proceedings of the 1992 symposium on Interactive 3D graphics, pp. 127–134. ACM, New York (1992)
[14] Pighin, F., Hecker, J., Lischinskiy, D., Szeliskiz, R., Salesin, D.H.: Synthesizing realistic facial expressions from photographs. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 75–84 (1998)
[15] Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental Mathematics 2(1), 15–36 (1993)
[16] Platt, S., Badler, N.: Animating facial expression. Computer Graphics 15(3), 245–252 (1981)
[17] Sheng, Y., Willis, P., Castro, G.G., Ugail, H.: PDE face: A novel 3d face model. In: Proc. of the 8th IASTED Int. Conf. on Visualization, Imaging, and Image Processing, Palma de Mallorca, Spain, September 2008, pp. 408–415 (2008)
[18] Sheng, Y., Willis, P., Castro, G., Ugail, H.: PDE-based facial animation: Making the complex simple. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part II. LNCS, vol. 5359, pp. 723–732. Springer, Heidelberg (2009)
[19] Sifakis, E., Neverov, I., Fedkiw, R.: Automatic determination of facial muscle activations from sparse motion capture marker data. ACM Transactions on Graphic 24(3), 417–425 (2005)
[20] Terzopoulos, D., Waters, K.: Physically-based facial modeling, analysis, and animation. Journal of Visualization and Computer Animation 1, 73–80 (1990)
[21] Vlasic, D., Brand, M., Pfister, H., Popović, J.: Face transfer with multilinear models. ACM Trans. Graph. 24(3), 426–433 (2005)
[22] Waters, K.: A muscle model for animating three-dimensional facial expression. Computer Graphics 22(4), 17–24 (1987)
[23] Wu, U., Thalmann, N., Thalmann, D.: A plastic-visco-elastic model for wrinkles in facial animation and skin aging. In: Proc. 2nd Pacific Conference on Computer Graphics and Applications (Pacific Graphics), pp. 201–213 (1994)
[24] Zhang, Q., Liu, Z., Guo, B., Terzopoulos, D., Shum, H.: Geometry-driven photorealistic facial expression synthesis. IEEE Trans. Vis. Comput. Graph. 12(1), 48–60 (2006)
[25] Zhang, Y., Sim, T., Tan, C.L., Sung, E.: Anatomy-based face reconstruction for animation using multi-layer deformation. Journal of Visual Languages and Computing 17, 126–160 (2006)

Evolved Controllers for Simulated Locomotion

Brian F. Allen and Petros Faloutsos

University of California, Los Angeles CA 90024, USA
vector@cs.ucla.edu

Abstract. We present a system for automatically evolving neural networks as physics-based locomotion controllers for humanoid characters. Our approach provides two key features: (a) the topology of the neural network controller gradually grows in size to allow increasingly complex behavior, and (b) the evolutionary process requires only the physical properties of the character model and a simple fitness function. No *a priori* knowledge of the appropriate cycles or patterns of motion is needed.

1 Introduction

Natural character motion is closely tied to the character's physical makeup and environment. As interactive entertainment rapidly adopts physical simulation as a core aspect, the problem of physical control for characters comes to the fore.

There are three main hurdles to the adoption of physical controllers in real-time systems. First, controllers, even for seemingly simple motions, have proven surprisingly difficult and time-consuming to engineer by-hand. Second, real-time environments allow for limited computational resources, and finally, of prime consideration for animation and games, the resulting motion should appear fluid and natural.

This work takes a significant step toward a long-standing goal in the graphics literature: "An ideal automated synthesis system would be able to design an efficient locomotion controller given only the mechanical structure of the creature (including actuators) and no other *a priori* information" [1]. By forgoing acquired real-world data (e.g., motion capture), our method becomes applicable to a much wider variety of characters. Acquiring motion data from animals at extreme scales, such as an ant or an elephant, is problematic and expensive. Further, imaginary characters, although quite common to game settings, are generally unavailable for motion capture sessions. We evaluate the method described in this paper using human-like characters because we believe human motion is generally held to the highest standard of quality of motion, and because bipedal locomotion is among the more difficult gaits to control.

Our approach creates controllers for physically simulated characters in a biologically inspired manner: by evolving networks of connected neurons. The resulting controllers are consistent with physical laws and provide fluid motion. Changes to the characters' size and shape results in changes to the motion trajectories. A sequence of frames from an animation of five morphologically distinct characters is shown in figure 1. The animation begins with all characters standing upright, feet together, and shows each character initiating a walk cycle.

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 219–230, 2009.

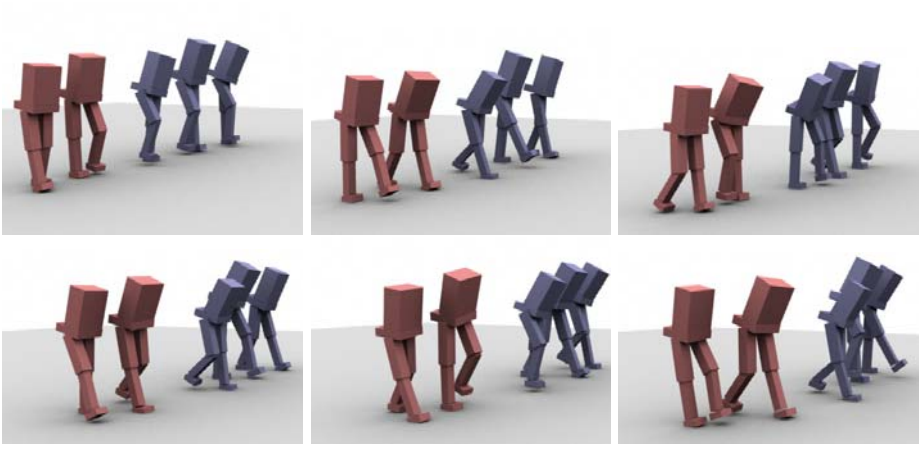


Fig. 1. Five human-sized characters with uniquely evolved walking controllers

2 Related Work

Controller design for physics-based articulated systems has been approached from a variety of perspectives. Several researchers have engineered controllers manually to perform a variety of ballistic or high-energy motions (such as running, vaulting and diving) [2,3,4]. Since low-energy motions, like walking, are poorly constrained by ballistics or energy minimization alone, controllers can rely on potentially unnatural or contrived higher-level constraints [5], or on generalizations from motion captured data [6]. Alternatively, the control problem can be simplified at the expense of realism in the physical simulation [7].

Our evolutionary approach is, at the core, an optimization problem. The space-time constraint approach also frames the problem of physically valid control in terms of search– minimizing a function of the energy expended by the virtual character [8,9]. This approach has been quite successful for finding motion paths in high-energy or ballistic domains, such as jumping or diving, however, low-energy motions with many physically correct solutions tend to be difficult to optimize and often yield unnatural motion.

More relevant to our approach are several attempts to produce locomotion cont rollers through stochastic optimization and genetic algorithms. Two early approaches were [10] and [11]. Sims referred to his control networks as "neural", but used a variety of functions favorable to cyclical output, such as sinusoids, in a manner more similar to genetic programming [12] than neuroevolution. However, as with our approach, evolving the topology of the control system was a key element in obtaining the complex behavior seen in Sims' virtual creatures. Gritz [13] also used genetic programming to create controllers, though with a focus on providing keyframe-like control of the resulting motion.

Laszlo [14] addressed the more constrained and difficult problem of bipedal walking. Their approach was to use limit-cycle control to reduce the problem

to a search of parameters that would stabilize a walk cycle. This approach yielded stable and controllable walks for two different character models, one humanoid and one non-humanoid. However, the resulting motion was clearly robotic.

More recently, evolved neural networks have become a central approach to control problems in the nascent field of evolutionary robotics [15]. For example, [16,17] use fixed-topology neural networks without sensory inputs to evolve biomimetic "central pattern generators" (CPG). The evolved artificial CPG's generate open-loop patterns that drive joint angles of a bipedal model to show very "natural motion" through a walk cycle. In these works, continuous-time neurons are arranged and connected in a human-designed, fixed topology designed to produce cyclical patterns suitable for a locomotion controller. The connection weights and neural parameters are evolved to generate walking motions. The design of the fixed topologies used in [17] were based on recent findings in neurobiology [18]. However, in the absence of guiding biological knowledge, choosing the best fixed topology for a given problem is difficult, sometimes requiring "expert experience and a tedious trial-and-error process" [19].

In contrast, our approach builds on recent advances in the evolution of the neural topology [20], in addition to evolving the connection weights. This both relieves the controller designer of the task of choosing the proper topology, and drastically expands the range of possible behaviors. This is an important advance in the pursuit of a generic controller-creator.

3 Overview of Our Approach

We propose an evolutionary approach that produces neural networks that are used to control physically simulated, humanoid characters. In the context of evolutionary processes, it is useful to appropriate analogous terms from biology. A *genome* in our system is information transmitted from parent to offspring, and is implemented as a weighted, directed graph. This graph is used to construct a neural network. The mapping of a genome to a neural network is the encoding scheme. We employ a simple, direct encoding: genome nodes become neurons and edges become interneuron connections.

A collection of such genomes makes up the population that undergoes an evolutionary process. Each genome is evaluated, and the best-performing genomes are then either duplicated with slight modification (*mutation*) or combined with other successful genomes to produce new genomes (*cross-over*). The genomes resulting from reproduction form the population of the next generation. This process is analogous to biological evolution and, over the course of many evaluate-select-reproduce generations, genomes better suited to performing the evaluation task are likely to emerge.

To perform the generational selection, each genome is evaluated by how well its generated neural network performs as the controller of a virtual character.

4 Artificial Neural Networks

Artificial neural networks(ANN's) are composed of (a) neurons with a single scalar internal state y, referred to as the neuron's activation level, and (b) weighted, directed connections between the neurons. The structure of the connections is known as the network's topology. Though simple, the ANN model is able to exhibit complex behavior due to the non-linear manner in which each neuron aggregates the inputs from its incoming connections.

Each neuron is updated once per cycle by linearly combining its incoming connections' activations and applying the activation function, $\sigma(x)$, to that sum to determine its activation level for the next cycle (as in Equation 1).

$$y_i = \sigma\left(\sum_{j=1}^{N} w_{ji} y_j\right), \qquad \sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

where σ is the sigmoid function and w_{ji} the weight of the connection from the jth neuron to the ith.

To achieve state and time dependent behavior, an artificial neural network must rely on cycles in its topology, in a manner analogous to cross-linked XOR gates forming an electronic flip-flop. Note that in the sigmoidal neurons used here, such cycles are the only way an ANN can store state internally. To manage complex timing-based behavior, as is needed for a motion controller, an ANN generally requires a complex topology of interneuron connections.

5 Neuroevolution

Evolving a fixed-topology neural network involves mapping the scalar connection weights to a one-dimensional vector. If the topology of the network is fixed, then the correspondence between position in the vector and the neural connection can be fixed. Then the evolutionary algorithm need only manipulate a fixed-length vector of scalars. To our knowledge, this general approach has been used by all evolved neural network motion controllers for 3D bipeds (e.g., [16,17,21,22,23,24]).

Although this traditional approach has the advantage of simplicity, using a fixed topology neural network categorically limits the type of behavior that can be exhibited (e.g., a network lacking recurrent connections cannot maintain memory). In contrast, by allowing the network to evolve its structure as well as the connection weights, the space of possible solutions is unrestrained. Our approach evolves neural topology using a version of the NEAT (NeuroEvolution of Augmenting Topologies) algorithm, described in detail in [25].

Unfortunately, evolving neural topology is not a panacea. Direct encoding from genetic representation to neural network topology may have scalability problems as the size of the graph grows [26]. Several indirect encodings (i.e., a genetic representation that requires some procedure to produce the neural

topology) have been proposed, such as cellular encoding [27], to alleviate these scalability concerns, though they are not without problems [28]. In the absence of either biological analogue to or accepted practice of an indirect encoding scheme, our approach uses direct encoding.

5.1 Mutation

Mutations to the directed-graph genotype are of two kinds: structural and non-structural. The latter alter the weight of a connection or a parameter of a neuron (see Section 4). Small alterations ($\mu = 0$, $\sigma^2 = 0.1$) are most likely, though occasionally entirely new values are chosen (at random uniformly in the range $(-0.1, 0.1)$) as replacements. In an effort to bias mutational changes toward newer structures, parameters that are relatively new features in the genome are more likely to be replaced than older ones.

Two types of structural mutation are allowed: splitting an existing edge into two edges and a node, and adding a directed edge between two previously unconnected nodes, or between a node and itself. All structural mutations increase the size of the genome, so any given genetic linage is monotonically increasing.

5.2 Crossover Using Historical Markers

The NEAT algorithm labels each node and edge in a genome with a unique *historical marker*. During reproduction, these markers are preserved and passed to the offspring. During sexual reproduction, these historical markers are used to determine genetic homology. The guiding assumption is that genes with the same historical origin (and therefore the same historical markers) will perform the same function in the phenotype. Although this is, in essence, an ad hoc approach whose assumption can, and may often, fail, it has been shown to be the best known neuro-evolutionary method for simple benchmark control problems such as double-pole balancing and predator-prey simulations [29].

5.3 Speciation to Protect Innovation

In addition to enabling a more productive cross-over operation, NEAT-style historical markers can also be used to estimate the chance of mating success between individuals. A distance metric $\delta = \frac{G}{N^k} + cW$ groups the individuals of a population into species, allowing sexual reproduction within a species containing similar individuals. In this expression, G is the number of genes without a corresponding historical marker in the genes of the other parent. The constant c is a normalization factor based on the magnitude of connection weights. N is the number of genes in the larger genome, and k is a term allowing a scaling of the effect of normalization based on the number of genes N. Such speciation greatly improves the likelihood that the next generation will be viable.

Note that previous implementations of the NEAT algorithm have used $k = 0$, while our implementation uses $k = 1$. The effect of $k = 1$ is to measure genomic distance by the ratio of differing genes to the total number of genes, in contrast

with $k = 0$, which considers the absolute number. Comparing the ratio, rather than absolute number better supports genomic distance calculations for large networks.

New *structural* innovations will likely differ significantly (as measured by δ) from existing genomes, meriting classification in a new species. Individuals only compete directly with other members of their own species, providing protection to new innovations until they have time to optimize their structure. This process is further assisted by giving new species an artificial fitness bonus of 20% for their first few generations.

6 Neural Networks for Control

Artificial neural networks (ANN's) have two attributes that suggest their applicability to dynamic controllers. First, for even large feedback-based tasks, neural networks require little memory and can be efficiently evaluated. Second, neural networks produce output behaviors that tend to be smooth and natural. For animation applications (as opposed to, e.g., robotics), the quality of the resulting motion is of great importance. Many approaches to physical animation control that originated in robotics have yielded stable motion, but have lacked fluidity and naturalness. Although admittedly difficult to quantify, evolved neural networks tend to yield smooth, natural-looking motion, especially in comparison with techniques based on discrete state-spaces.

7 Anthropomorphic Character Model

Our evolutionary system is not provided with any information specifying the particular gait to use. This approach improves generality, but means the character morphology plays an significant role in the quality and kind of resulting locomotive gait. Since we are interested in controllers that provide human-like motion, we use character models whose physical parameters mimic human morphology. In addition, the generality of the approach is an important measure of success. To test the generality of our method in finding human-style gaits, we use a sample of humanoid character models from the normal human range of height, weight and bi-iliac (hip) breadth (BIB) covering the 5^{th} to 95^{th} percentile of each men and women, which encompasses the variation of slightly less than 95% of the total population.

Body segment sizes and weights are scaled linearly according to the three measures (height, weight and BIB) obtained from anthropometric data aggregated from over 30,000 individuals [30]. The relative proportions of segment sizes and weights are shown in Table 2.

7.1 Sensors

The controller is provided a set of eleven sensory inputs. Proprioceptive sensors from each actuated joint angle are provided, as well as haptic sensors for each

foot that register −1 when the foot is touching the ground plane, and +1 at any other time. The X and Z components (ranging from −1 to +1) of the torso's normalized up-vector are also given.

7.2 Actuation

The controller specifies, at each control time-step, the target angle the joint will be driven toward. The corresponding actuator applies torque to drive the joint to the desired angle θ_d. The output signal of the controller ranges within $[0, 1]$, of which the central 0.8 range is linearly scaled to the full range of the joint, leaving 0.1 units at each extreme clamped to 0.0 and 1.0, respectively. The applied torque τ is computed from the total moment of inertia at the joint I using a proportional-derivative (PD) controller as

$$\tau = I(k_p(\theta - \theta_d) - k_d\dot{\theta}). \tag{2}$$

Table 1. The allowed ranges for each joint of the agent. The Control column indicates if the joint's target angle is set by the controller. If not, the joint applies torque to center itself in its range.

	Range		Neural Control	
Joint, axis	High	Low	Walk	Balance
Spine, transverse	$-\frac{\pi}{12}$	$\frac{\pi}{12}$	Yes	No
Spine, coronal	$-\frac{\pi}{16}$	$\frac{\pi}{16}$	No	No
Spine, sagittal	$-\frac{\pi}{16}$	$\frac{\pi}{16}$	No	No
Hip, transverse	0	0	No	No
Hip, coronal	$-\frac{\pi}{48}$	$\frac{\pi}{24}$	Yes	Yes
Hip, sagittal	$-\frac{\pi}{16}$	$\frac{\pi}{16}$	Yes	Yes
Knee	0	$\frac{3\pi}{4}$	Yes	Yes
Ankle, transverse	$-\frac{\pi}{12}$	$\frac{\pi}{12}$	No	No
Ankle, coronal	$-\frac{\pi}{12}$	$\frac{\pi}{12}$	No	Yes
Ankle, sagittal	$-\frac{\pi}{4}$	$\frac{\pi}{4}$	No	Yes

Table 2. The distribution of the physical proportions of the agent's body. The mass proportion of bilaterally repeated segments shows the summed mass proportion of both sides.

Body Segment	Height	Mass
Trunk	31.2%	51.6%
Waist	6.2%	10.3%
Thigh (2)	31.0%	19.3%
Shank (2)	25.3%	9.3%
Foot (2)	6.2%	9.3%

7.3 Bilateral Symmetry

The character models used are bilaterally symmetric both in their body shape and in their neural controllers. Using simple, fixed-topology reactive neural controllers, [21] demonstrated that two identical, uncoupled networks could be used, instead of one monolithic neural network, to decrease the size of the evolutionary search space. Our system uses a similar approach by building two identical neural networks from a single evolved system.

Human locomotion is approximately bilaterally symmetric (although asymmetry has been shown to be important to human perception of motion as realistic [31]). It is reasonable then to use the same controller duplicated to each side of the agent. This approach was suggested for future work in [16] and [22] used simple, fixed-topology reactive neural controllers to demonstrate that two identical, uncoupled networks could successfully decrease the size of the evolutionary search space compared to evolving a single, monolithic controller.

In this work, we evolve a single controller genome, which is then used to build two initially identical neural networks, one for each side. The outputs of each network drive their side's actuated joints, with the central waist joint taken as the average of the two corresponding output nodes' activations. The inputs of each network are likewise set from per-side information.

Each controller is given two bias nodes, one has a constant activation of $+1$, while the other has an activation of $+1$ for the right controller and -1 for the left. This per-side bias (along with asymmetric sensory input) allows for asymmetrical behavior. In addition, two haptic foot-contact sensors yield $+1$ for the same-side foot, and -1 for the opposite foot.

8 Implementation

Evolutionary runs use a population of 512 individual genomes, which are clustered into thirty species based on the genetic similarity metric δ (see Section 5.3). Each generation, every genome is evaluated by creating two neural networks, each set to control one side of the character.

These neural networks are supplied with sensory data and updated once every 0.07 s of simulated time. This delay was chosen to be within the observed range of spinal reflex response times in humans [32]. Additionally, in the context of animation, Zordan [33] described similar response delays as subjectively believable.

8.1 Fitness Measure

Evolutionary selection is based on a simple fitness measure f calculated as

$$f = c_d \max(\|\text{proj}_{\boldsymbol{j}}\, \boldsymbol{d}\|,\ \epsilon) + f_b, \tag{3}$$

where $\boldsymbol{d}$ is the vector from the starting position to the hindmost foot, and $\boldsymbol{j}$ is a fixed unit vector pointing in the direction the animator wishes the character to walk. Throughout this work, $\boldsymbol{j}$ is fixed to the character's starting facing direction, rewarding simple forward walking. c_d is a constant scaling factor, and ϵ a fixed small positive value. f_b is proportional to the time spent upright, $f_b = c_t t$, where t is the elapsed time for the trial and c_t is a constant scale factor. We also experimented with more complex fitness functions for locomotion, however Equation 3 was the most effective and is used for the results presented.

Early Termination. The evaluation of an individual controller halts if an early termination criteria is met. This is a useful way to improve the overall speed of the evolutionary process, and also provides a powerful means to shape the resulting behavior by specifying outcomes the animator deems undesirable.

Waist height. If the z-coordinate of the waist segment's center of mass (CoM) falls below a minimum height (70% of the waist segment's starting height), the simulation is terminated and the genome's final fitness is that computed using the state of the previous time-step. By forcing the CoM to maintain a minimum height, locomotion gaits such as crawling and walking on the knees (instead of the feet) are prohibited.

Instability. If significant numerical instability or joint divergence is detected, the simulation is terminated and the genome's final fitness is set to the minimal allowed value, ϵ.

Stagnation. A trial is terminated if the fitness fails to improve by $\epsilon + \frac{c_t n}{\Delta t}$ over the n seconds of simulated time since the previous improvement. This forces the system to gain fitness by some means other than time, preventing, for example, simply balancing in place for the course of the run.

8.2 Support Harness

During the first 100 generations, a virtual harness provides lateral, posterior and vertical support to the character. A similar harness was described in [17], though our variation offers no resistance to forward or upward motion. Although not found to be strictly necessary, a supporting harness was helpful in reducing the total number of evaluations needed to find a walking controller.

9 Results

Bipedal locomotion is a difficult control problem, partly due to the inherent instability. However, demanding natural-looking motion compounds the difficulty because, as a system, a walking humanoid is relatively unconstrained by either physical limitations (i.e., there are many physically feasible forms of moving forward besides the normal human gait, e.g., hopping, jumping or skipping) or energy minimization (though the human walk is characteristically low-energy). Our method explores an aesthetic subset of these possible motions due to two key factors. First, we use smoothly varying neural networks, and second, we use human-scale models with human-scale actuators.

Example joint angles for a typical successful walk controller are shown in Figure 2. This particular individual is of medium female height, weight and BIB.

9.1 Reliability of Walking Controller Generation

Although evolutionary processes are stochastic by nature, it is desirable that the system be able to reliably generate quality controllers. Over the entire range

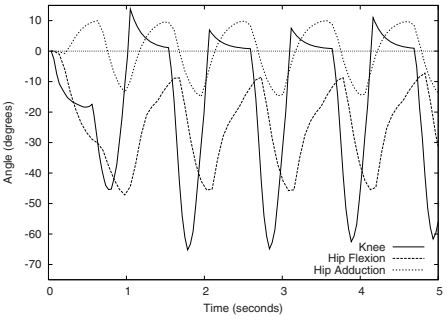


Fig. 2. Joint angles observed for average (50^{th} percentile) female model's walk cycle, starting from a standing position

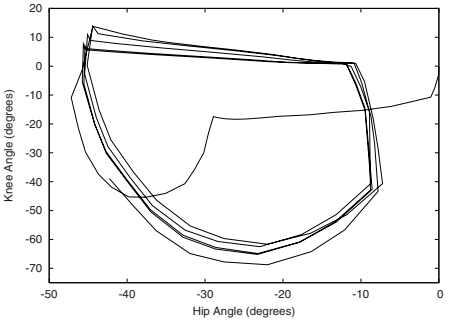


Fig. 3. Phase-space diagram of the hip and knee joints during the gait initiation and subsequent walking cycle

of human body shapes evaluated at walking, 93% of the runs found controllers capable of walking upright for more than two meters (28 out of 30 runs). Both failures were with heavy (95^{th} percentile weight) characters with short stature (5^{th} percentile height), once for a male character and once for a female. Successful walking controllers were found by simply repeating the runs with the same parameters. It is worth commenting that humans with such opposite extremes in body size and weight are likely well outside the normal range.

10 Conclusions

In this work, we have shown that the proposed system can produce walking controllers for a wide range of anthropomorphic bodies. The system is able to do so without the need for *a priori* knowledge of the correct neural topology. The resulting controllers show smooth and believable human-like motion, without the stiffness and phase artifacts associated with methods based on robotics techniques. The controllers are closed-loop, using proprioceptive, tactile and vestibular sensors to maintain balance and improve their performance.

However, the evolutionary process is by nature unpredictable, and may not result in useful controllers for any given character morphology. Also, none of the evolved controllers were capable of sustained walking– generally walks were stable for 5-10 meters before toppling. For the neural networks to evolve sufficient complexity to initiate locomotion and walk for a few meters, they have genome sizes in the several hundred real parameters. Mutations that only effect the stability in the circumstances of the final step are likely to be quite rare, leaving the system fragile and unable to improve. Constraining to or enforcing for stable cycles in neural output may be possible, but may also introduce unwanted visual or aesthetic artifacts in the motion.

A promising next step is to build on the task-specific controllers evolved with our system by composing many together, as described with hand-engineered controllers in [4].

References

1. van de Panne, M., Kim, R., Fiume, E.: Virtual wind-up toys for animation. In: Proceedings of Graphics Interface 1994, Banff, Alberta, Canada, pp. 208–215 (1994)
2. Raibert, M.H., Hodgins, J.K.: Animation of dynamic legged locomotion. In: SIGGRAPH 1991: Proceedings of the 18th annual conference on Computer graphics and interactive techniques, pp. 349–358. ACM Press, New York (1991)
3. Hodgins, J.K., Wooten, W.L., Brogan, D.C., O'Brien, J.F.: Animating human athletics. In: SIGGRAPH 1995: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 71–78. ACM Press, New York (1995)
4. Faloutsos, P., van de Panne, M., Terzopoulos, D.: The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. Computers and Graphics 25(6), 933–953 (2001)
5. Yin, K., Loken, K., van de Panne, M.: Simbicon: Simple biped locomotion control. In: Proceedings of the 2007 SIGGRAPH conference, vol. 26. ACM, New York (2007)
6. da Silva, M., Abe, Y., Popović, J.: Interactive simulation of stylized human locomotion. In: International Conference on Computer Graphics and Interactive Techniques. ACM, New York (2008)
7. Shapiro, A., Chu, D., Allen, B., Faloutsos, P.: A dynamic controller toolkit. In: Sandbox 2007: Proceedings of the 2007 ACM SIGGRAPH Symposium on Video Games, pp. 15–20. ACM, New York (2007)
8. Witkin, A., Kass, M.: Spacetime constraints. Computer Graphics 22(4), 159–168 (1988)
9. Liu, C.K., Hertzmann, A., Popovic, Z.: Learning physics-based motion style with nonlinear inverse optimization. ACM Transactions on Graphics (TOG) 24(3), 1071–1081 (2005)
10. van de Panne, M.: Sensor-actuator networks. In: SIGGRAPH 1993: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 335–342. ACM Press, New York (1993)
11. Sims, K.: Evolving virtual creatures. In: SIGGRAPH 1994: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pp. 15–22. ACM Press, New York (1994)
12. Koza, J.: Genetic Programming: On the Programming of Computers by means of Natural Selection. MIT Press, Cambridge (1992)
13. Gritz, L.: Evolutionary Controller Synthesis for 3-D Character Animation. PhD thesis, George Washington University (1999)
14. Laszlo, J., van de Panne, M., Fiume, E.: Limit cycle control and its application to the animation of balancing and walking. In: Computer Graphics. Annual Conference Series, vol. 30, pp. 155–162 (1996)
15. Nolfi, S., Floreano, D.: Evolutionary Robotics. In: Intelligent Robots and Autonomous Agents. MIT Press, Cambridge (2000)
16. Reil, T., Husbands, P.: Evolution of central pattern generators for bipedal walking in a real-time physics environment. IEEE Trans. Evolutionary Computation 6(2), 159–168 (2002)
17. Reil, T., Massey, C.: Morpho-Functional Machines: The New Species: Designing Embodied Intelligence. Springer, Heidelberg (2003)
18. Golubitsky, M., Stewart, I., Buono, P.L., Collins, J.: Symmetry in locomotor central pattern generators and animal gaits. Nature 401, 693–695 (1999)

19. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87, 1423–1447 (1999)
20. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
21. Paul, C.: Bilateral decoupling in the neural control of biped locomotion. In: Proc. 2nd International Symposium on Adaptive Motion of Animals and Machines (2003)
22. Paul, C.: Sensorimotor control of biped locomotion based on contact information. In: Proc. International Symposium on Intelligent Signal Processing and Robotics (2004)
23. McHale, G., Husbands, P.: From Animals to Animats 8. In: Proceedings of the 8th International Conference on Simulation of Adaptive Behavior. MIT Press, Cambridge (2005)
24. Vaughan, E.D., Paolo, E.D., Harvey, I.R.: The evolution of control and adaptation in a 3d powered passive dynamic walker. In: Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems (Alife9). MIT Press, Cambridge (2004)
25. Stanley, K.: Efficient Evolution of Neural Networks Through Complexification. PhD thesis, University of Texas, Austin (2004)
26. Hornby, G.S., Pollack, J.B.: Creating high-level components with a generative representation for body-brain evolution. Artificial Life 8(3), 223–246 (2002)
27. Gruau, F.: Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm. PhD thesis, Laboratoire de l'Informatique du Parallilisme, Ecole Normale Supirieure de Lyon, France (1994)
28. Hornby, G.S.: Shortcomings with tree-structured edge encodings for neural networks. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 495–506. Springer, Heidelberg (2004)
29. Stanley, K.O., Miikkulainen, R.: Competitive coevolution through evolutionary complexification. Journal of Artificial Intelligence Research (21), 63–100 (2004)
30. Department of Defense, U.S.o.A.: Anthropometry of U.S. Military Personnel (DOD-HDBK-743A). Department of Defense, United States of America (1991)
31. Bodenheimer, B., Shleyfman, A., Hodgins, J.: The effects of noise on the perception of animated human running. Computer Animation and Simulation (1999)
32. DJ, D.: Neuromuscular control system. IEEE Transactions on Biomedical Engineering 3, 167–171 (1967)
33. Zordan, V.B., Majkowska, A., Chiu, B., Fast, M.: Dynamic response for motion capture animation. ACM Trans. Graph. 24(3), 697–701 (2005)

Integrated Analytic and Linearized Inverse Kinematics for Precise Full Body Interactions

Ronan Boulic and Daniel Raunhardt

VRLAB, Ecole Polytechnique Fédérale de Lausanne, EPFL
1015 Lausanne, Switzerland
{Ronan.Boulic,Daniel.Raunhardt}@epfl.ch
http://vrlab.epfl.ch

Abstract. Despite the large success of games grounded on movement-based interactions the current state of full body motion capture technologies still prevents the exploitation of precise interactions with complex environments. This paper focuses on ensuring a precise spatial correspondence between the user and the avatar. We build upon our past effort in human postural control with a Prioritized Inverse Kinematics framework. One of its key advantage is to ease the dynamic combination of postural and collision avoidance constraints. However its reliance on a linearized approximation of the problem makes it vulnerable to the well-known full extension singularity of the limbs. In such context the tracking performance is reduced and/or less believable intermediate postural solutions are produced. We address this issue by introducing a new type of analytic constraint that smoothly integrates within the prioritized Inverse Kinematics framework. The paper first recalls the background of full body 3D interactions and the advantages and drawbacks of the linearized IK solution. Then the Flexion-EXTension constraint (FLEXT in short) is introduced for the partial position control of limb-like articulated structures. Comparative results illustrate the interest of this new type of integrated analytical and linearized IK control.

1 Introduction

For a movement-based game to be successful a good match is needed between the mapping of sensed user gestures in the game space and a sufficiently challenging goal to achieve within that space. There is no need of highly precise full-body mapping in the game space as long as the 'flow' and 'engagement' of the user are maintained. These two concepts are central to the analysis and design of movement-based interactions as extensively reviewed in [NvDR08]. They help to understand that the player can accommodate large discrepancies between their egocentric space and the virtual space of the game. For example, Loke et al. analyze multiple framework for interaction design that aim to identify the best exploitation of bodily interactions with a given technology ; the paper is illustrated with two Sony Eyetoy games [LLRE07].

Conversely, a highly precise spatial correspondence of the user and the avatar is a basic requirements of virtual prototyping applications for which important design decisions have to be made based on 3D immersive interactions. However, even now few motion capture hardware and software can provide a robust and precise real-time

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 231–242, 2009.

participant registration with virtual objects and obstacles. The problem is made even more complex whenever the participant wishes to feel immersed as a virtual counterpart with a differing body size as analyzed in [BMT09]. We believe that methods and algorithms tackling such precision issues will find their way in the game sector in the near future.

In this paper we focus on ensuring a precise spatial correspondence between the user equipped with a relevant posture sensing equipment and the avatar. We first review the state of the art on motion capture and Inverse Kinematics (IK) approaches. Our contribution builds upon our past effort in human postural control with a Prioritized Inverse Kinematics framework [BB04]. Such a framework eases the dynamic combination of constraints, e.g. to avoid collisions while performing a reach task [PMMRTB09]. However its reliance on a linearized approximation of the problem makes it vulnerable to the well-known full extension singularity of the limbs illustrated in Figure 1. In such context the convergence performance is degraded and/or less believable intermediate postural solutions are produced.

We address this issue by introducing a new type of constraint that smoothly integrates within the prioritized Inverse Kinematics framework. The proposed Flexion-EXTension (FLEXT) effector takes advantage of the knowledge of the desired 1D displacement along the direction linking the limb root to the controlled effector (chain tip square on Figure 1). Comparative results illustrate the interest of this new type of IK effector.

2 Background in Full Body 3D Interactions

The field of applications relying on full body interactions traces back to the sixties as reviewed by Sturman [S98]. In the early days mechanical exoskeleton were (tediously) adjusted to the performer for live performance in various shows. However such technology is too intrusive to be accepted by engineers or gamers. Nowadays the trend is clearly towards transparent and ubiquitous systems such as marker-free vision-based approaches. For example Michoud et al. report performances compatible with real-time full-body interactions [MGBB07]. Yet the approach is limited to the identification of eleven body joints due to the coarse resolution of the underlying voxel grid ; besides its reliance on a static background for silhouette processing limits its application in more versatile contexts. In the nineties a seducing alternative to vision-based techniques was to exploit magnetic sensors as they are not subject to occlusion [A96]. Molet et al [MBRT99] proposed an analytic IK algorithm exploiting one 3D orientation measurement per body segment and one 3D position measurement for the whole body. It has been exploited in a real-time full-body tennis game demonstrated at the international exhibition Telecom Interactive‘97 in Geneva [M*99]. Its major drawback was however the heavy calibration effort to obtain homogeneous position measurement over the capture area [MBRT99]. A true improvement came with the advent of active optical motion capture systems that could automatically recognize the markers even after some periods of occlusion (e.g. [P]). Although not as precise as some optical systems exploiting high tech cameras with passive markers [V] we tradeoff the extra precision for the real-time marker identification. In the remainder of the section we review various approaches of on the fly postural reconstruction from a set of 3D locations, further denoted as position *effectors*.

Analytic methods have always been the most efficient approaches for reconstructing the body posture from the input of desired locations of body parts. One key reference is the work of Korein [K85] from which successive variants and improvements have been proposed. Tolani et al. describe a position and orientation control of the limbs [TGB00] with a special focus on the control of the *swivel* angle along the direction linking the limb root to the controlled effector (visible as a dotted line in Figure 1a). Kulpa et al. build upon the Cyclic Coordinate Descent approach to combine postural and balance control [KMA05]. Kallmann takes advantage of a database of reach postures [K07]. Unzuetta et al. exploits a reduced set of effector locations (pelvis, head, hands and feet) and compares its performances with numerous other analytic and linearized IK approaches [UPBS08]. Despite their low computational cost analytical solutions often suffer from temporary instability. Indeed most of them rely on simplifying assumptions that are violated from time to time. For example, the limb full extension posture is prone to produce discontinuities of the swivel angle.

Linearized Inverse Kinematics approaches are more versatile as they can combine a large variety of constraints in a redundant context. On the down side they are much slower due to the iterative nature of the convergence towards a posture minimizing the constraint errors [BMW87][PB91][ZB94][BB04][UPBS08]. Nevertheless the regular increase in computing power has made it possible to exploit them for full-body 3D immersive interactions [MBT08]. The next section focuses on the numerical problems due to the human postural singularities.

3 The Limb Singular Postures

The limb singular postures (Figure 1) are especially annoying for linearized IK method. The full extension posture (Figure 1b) is very frequent in standing and reaching poses. We have identified two recurrent problems in such singular posture.

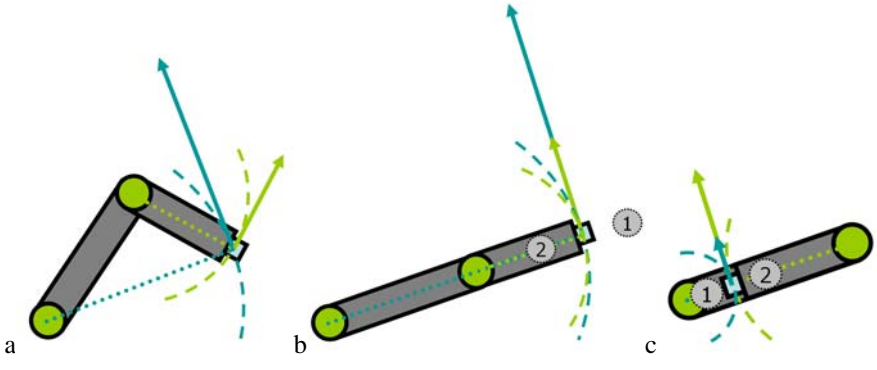


Fig. 1. In the general case the linearized Inverse Kinematic control can move the chain tip effector in any direction (a) but this is no more the case when the two controlled segments are aligned such as in the full extension (b) or the full flexion (c) ; no linearized solution can be found to move the chain tip in the direction of goal 1 or 2 although there exist an analytic solution for goal 2.

The flexing blindness syndrome: when a singularity occurs the limb joints cannot theoretically contribute to move the chain tip effector towards the limb root because the linear influences of the joints are collinear and span only a 1D translation space orthogonal to the desired translation (visible on Fig 1b and c). We call this artefact the flexing blindness syndrome to distinguish it from the second side effect described later. In practice the posture is seldom exactly on the singularity but in the close neighborhood. The use of a damped least square pseudo-inverse ensures that the solution norm remains bounded [M90][BB04] by trading off the convergence speed for stability. This type of context leads to a very slow limb flexing. In the meantime the effector inward movement can be produced by other joints such as the spine or whole body movements. Needless to say that the movement produced by this type of convergence is much less believable than simply flexing the limb.

The rank decrease syndrome: this alternate type of full extension side effect occurs when the posture happens to be considered as theoretically singular by the IK algorithm. In such a context the rank decreases from 2 (Fig 1a) to 1 because the linearized joint influences are collinear and span a 1D space (Fig 1b, c). The consequence of such a rank decrease is the corresponding rank increase of the Null space in the joint variation space. It could be interpreted positively as an opportunity for lower priority constraints to achieve their own goal. In fact the consequence is rather negative as what happens is a posture variation discontinuity introduced by these lower priority constraints. Most of the time the posture variation induced by the low priority constraints leads to a new posture that restores the full rank of the high priority constraint. Therefore this ends the convergence opportunity for the low priority tasks. This convergence pattern can reproduce repeatedly as long as the constraints' goal remain constant. As a consequence, it is highly desirable to prevent such a rank decrease.

The contribution of this paper addresses both kinds of side effects. We now briefly recall the architecture of the linearized Inverse Kinematics with multiple levels of priority prior to describe how to integrate an analytic solution of the limb flexion-extension control into that framework.

4 Basic Concepts of Prioritized Inverse Kinematics

The concept of priority in the linearized IK solution was introduced by Liégeois [L77] and generalized for an arbitrary number of priority levels in [SS91] ; Baerlocher proposed a formulation with a lower computation cost in [BB04]. An extended illustration of the concepts and the algorithm can be found in [BLP06]. In the remainder of this section we simply recall the key aspects and notations related to this approach.

Linearized IK and validity of the solution: a linearized Inverse Kinematics approach replaces the resolution of a general non-linear problem of finding the posture $\boldsymbol{\theta}$ such that

$$\mathbf{x}_{goal} = \mathbf{f}(\boldsymbol{\theta}) \tag{1}$$

by the resolution of a first order approximation equation

$$\Delta\mathbf{x} = \boldsymbol{J}\,\Delta\mathbf{q} \tag{2}$$

that is valid only in the neighborhood of the current state of the system $(\mathbf{x}_c, \boldsymbol{\theta}_c)$**.** The Jacobian $\boldsymbol{J}$ is the matrix of the partial derivatives $\delta\mathbf{x}/\delta\boldsymbol{\theta}$ for the current state.

For example, the two vectors displayed for Figure 1a are the partial derivatives of the chain tip position with respect to the limb joints. Inverting equation 2 consists in expressing a desired position variation $\Delta\mathbf{x}$ as a linear combination of those vectors. As one can see from Figure 1 the vectors are in fact only the approximation of the true joints influence shown as arcs. The validity of the approximation degrades as $\Delta\mathbf{x}$ norm increases. Hence equation 2 has to be solved for successive small variations $\Delta\mathbf{x}$ proportional to $(\mathbf{x}_{goal} - \mathbf{x}_c)$. As a rule of thumb, the maximum norm of $\Delta\mathbf{x}$ can be expressed as a fraction of the total length of the articulated chain, e.g. 10%. In our case we also scale it by a damping factor between 0 and 1 that prevents instabilities in the close neighborhood of the goal.

It is now easy to see why the limb full extension is singular for flexing the limb (Fig 1b); it is not possible to express a desired $\Delta\mathbf{x}$ in the direction of the limb root as a linear combination of the two collinear vectors stored in the Jacobian $\boldsymbol{J}$. Similarly one can also see the 2 to 1 rank reduction of the space spanned by these two vectors in Figure 1b and 1c.

Redundancy, priority and projection operator: the articulated structure to control is generally redundant with respect to the constraints to achieve, i.e. the dimension **N** of the joint space is much larger than the dimension **M** of the constraint space. The first consequence is the potentially infinite number of posture variations $\Delta\boldsymbol{\theta}$ that achieve a desired constraint variation $\Delta\mathbf{x}$. The linearized IK architecture retains the one with the smallest (and bounded) norm as provided by a damped pseudo-inverse of the jacobian $\boldsymbol{J}$. For 3D full-body interactions multiple constraints have to be enforced simultaneously ; it is possible to associate a distinct priority level **k** to each of them, or to obtain a compromise solution by integrating them in a single Jacobian. The exploitation of priority levels is preferred to guarantee important properties first, e.g. the avatar feet remains in contact with the floor, etc… In such a case a projection operator

$$\boldsymbol{P}_k = \boldsymbol{I} - \boldsymbol{J}_{Ak}^{+}\,\boldsymbol{J}_{Ak} \tag{3}$$

is built to avoid low priority solutions to interfere with all higher priority ones gathered in the augmented Jacobian $\boldsymbol{J}_{Ak}$[SS91].

5 The 1D Analytic FLexion-EXTension (FLEXT) Constraint

The proposed Flexion-EXTension (FLEXT) constraint is intimately associated with a limb-like articulated structure corresponding to those present in the human body for the arms and the legs. The considered limb articulated structure is constituted by a ball & socket joint for the root joint and, for the middle joint, the succession of a flexion-extension rotation followed by a twist rotation along the second limb segment. Only the Flexion-extension degrees of freedom of the middle and the root joints are exploited for the proposed constraint (Figure 2a,b). The other degrees of freedom remain free to be exploited by other constraints. The FLEXT constraint concept is illustrated in the ideal case where both limb segments belong to the plane orthogonal to the middle joint flexion-extension axis. A more general case is presented in the appendix A.

The rationale for introducing the new FLEXT constraint is the following :

- The flexibility of the Prioritized IK framework is highly desirable to preserve
- The linearized approximation is satisfactory except in singular contexts
- The singular context occurs frequently in some application such as human motion capture (standing, reaching)
- It is necessary and sufficient to solve ***analytically*** for the posture variation $\Delta\theta$ corresponding to a desired *scalar displacement* Δx along the line linking the effector to the limb root (Figure 2a,b) to prevent the two identified side effects.
- As a consequence, the control of the other effector dimensions (position and orientation) can be left to standard linearized constraints.

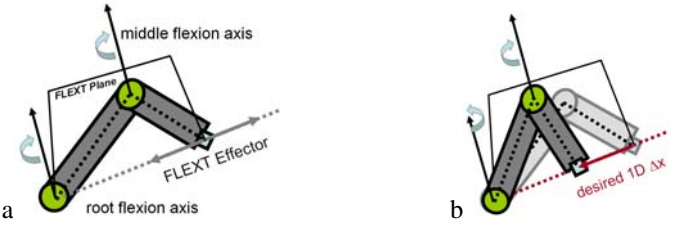


Fig. 2. (a) Definition of the Flexion-Extension 1D controlled translation and the FLEXT plane passing through the root joint whose normal is the flexion axis, (b) solving for a desired scalar displacement Δx resulting in a coupled variation of the two flexion degrees of freedom ; note that the limb remains within the FLEXT plane

The principle of the proposed approach is to reframe the 1D analytical solution as a linear equality constraint so that it can be transparently integrated in the linearized IK framework.

5.1 Evaluating the Posture Variation $\Delta\theta$ for a Desired Scalar Displacement Δx

First let us express the minimal angle variation $(\Delta\alpha, \Delta\beta)$ in the FLEXT plane that achieves the signed scalar displacement Δx as illustrated in Figure 3 (Δx is negative in this example). The distances **L1, L2, L3** are constant known quantities or can be evaluated from the current limb state. The angles are obtained from the cosine rule:

$$\cos\alpha = (L1^2 - L2^2 + L3^2) \quad /(2\,L1\,L3) \tag{4}$$

$$\cos\alpha' = (L1^2 - L2^2 + (L3 + \Delta x)^2) \,/(2\,L1\,(L3 + \Delta x)) \tag{5}$$

$$\cos\beta = (L1^2 + L2^2 - L3^2) \quad /(2\,L1\,L2) \tag{6}$$

$$\cos\beta' = (L1^2 + L2^2 - (L3 + \Delta x)^2) \,/(2\,L1\,L2) \tag{7}$$

and

$$\Delta\alpha = \alpha' - \alpha \qquad \Delta\beta = \beta' - \beta \tag{8}$$

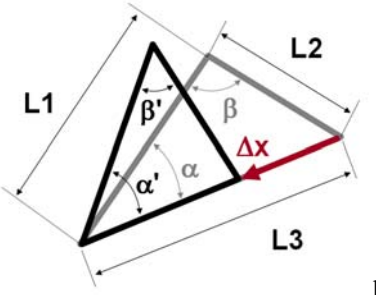


Fig. 3. The two triangles highlighting the posture variation of Figure 2b in the FLEXT plane. **L1** is the distance from the root to the middle joint, **L2** is the distance from the middle joint to the effector, **L3** is the initial distance of the effector to the root ; in this example it is decreased by a negative Δx displacement.

The limb posture variation $\Delta\theta$ is composed of a component for the root joint and a component for the middle joint ; they can be obtained from the angle variations in the FLEXT plane $(\Delta\alpha, \Delta\beta)$ as follows:

- By construction of the FLEXT constraint $\Delta\beta$ corresponds to the scalar variation of the middle joint flexion-extension degree of freedom. The middle joint twist variation is null.
- $\Delta\alpha$ has to be converted into a parameter variation of the ball & socket root joint. We exploit the exponential map representation of rotation [G98] but the principle of the conversion remain the same for other choices. It work like this:
  - Express the middle flexion axis in the coordinate system of the first limb segment that takes into account the current root joint transformation. The result is the root flexion axis.
  - Build the rotation matrix $\boldsymbol{R}_{\alpha}$ of amplitude $\Delta\alpha$ along the root flexion axis.
  - Concatenate $\boldsymbol{R}_{\alpha}$ to the current joint rotation matrix $\boldsymbol{R}$ to obtain a new rotation matrix $\boldsymbol{R'}$.
  - Convert both $\boldsymbol{R'}$ and $\boldsymbol{R}$ into your chosen rotation parameter ; the difference is the root joint component of the posture variation $\Delta\theta$

5.2 Evaluating the FLEXT Coupling Jacobian J_{FLEXT} and the Task Δx_{FLEXT}

In order to be transparently integrated within the linearized IK framework, the FLEXT constraint has to provide its Jacobian matrix $\boldsymbol{J}_{FLEXT}$ and its task vector $\Delta\boldsymbol{x}_{FLEXT}$. Given these two entities it can be associated with a priority and integrated in the loop evaluating the prioritized IK solution [BB04]. The FLEXT constraint has one fundamental difference compared to other linearized IK constraints : it exploits the knowledge of the desired scalar displacement $\Delta\boldsymbol{x}$ to build its associated Jacobian line matrix $\boldsymbol{J}_{FLEXT}$ whereas other constraints evaluate separately their Jacobian and their task vector.

Given the desired scalar displacement Δx and the corresponding posture variation $\Delta\theta$ (section 5.1) the FLEXT line matrix Jacobian J_{FLEXT} and the task Δx_{FLEXT} are given by:

$$J_{FLEXT} = (1 / \| \Delta\theta \|) \Delta\theta^T \quad (9)$$

$$\Delta x_{FLEXT} = \| \Delta\theta \| \quad (10)$$

J_{FLEXT} being a unit vector its pseudo-inverse is given by:

$$J_{FLEXT}^+ = J_{FLEXT}^T \quad (11)$$

Hence the product of the pseudo-inverse J_{FLEXT}^+ by the task Δx_{FLEXT} gives the expected $\Delta\theta$. Most likely the task Δx_{FLEXT} will be scaled down by the linearized IK framework to suit the small variation requirement (section 4) but in any case the solution will be proportional to $\Delta\theta$. The key point to note is that, once the coupled joint variations are expressed in J_{FLEXT}, the corresponding one dimensional joint variation subspace cannot be exploited by lower priority constraints owing to the projection operator (equ. (3)).

5.3 Handling Singular Contexts

The description provided in the previous section has to be completed to handle the following singular contexts:

- **Null desired displacement Δx**: this frequent context is singular as it produces a null $\Delta\theta$ that prevents the normalization of the Jacobian (equ. (9)). It would be a mistake to consider that no constraint exists ; instead there IS a constraint of keeping the same location for the effector. So the solution is not to specify the Jacobian as a null line matrix but instead to evaluate it for a ***small virtual displacement*** Δx_V. This allows to build a non-null jacobian J_{FLEXT} that expresses and locks the one-dimension joint variation subspace acting on this type of displacement. The task itself Δx_{FLEXT} is set to zero to be consistent with Δx.
- **Limb full extension or full flexion :** two contexts are limit cases for which the desired displacement Δx may not be totally achievable. For this reason any desired displacement Δx must always be checked against potentially reaching these two limits before applying equations (4) to (7). We must have, respectively for the full extension and the full flexion:

$$L3 + \Delta x < L1 + L2 \quad (12)$$

$$L3 + \Delta x > | L1 - L2 | \quad (13)$$

  In case one of these limits is violated the desired displacement is clamped to the limit value. The case of a resulting null clamped displacement has to be treated according to the virtual displacement method described in the previous point. All other cases can be treated with the standard method (sections 5.1 and 5.2).

The methodology described in these two points eradicates the two problems described in section 3. Flexing can always be computed if it is within the limits as it is solved analytically. If the goal is unreachable, out of the limits, the desired displacement is clamped and possibly null. But owing to the virtual displacement method we can

identify the joint variation subspace that produces a flexion-extension and we can lock it to prevent its use by lower priority constraints. This second mechanism prevents the rank decrease syndrome

6 Results

We have first performed comparative tests of the convergence performance for two cases of difficult flexing for linearized IK (Fig. 4 a,b,c,e,f). The error amplitude show a regular and much faster decrease with the FLEXT constraint. Figure 4d shows an example of combining the FLEXT constraint with an orientation constraint of the hand.

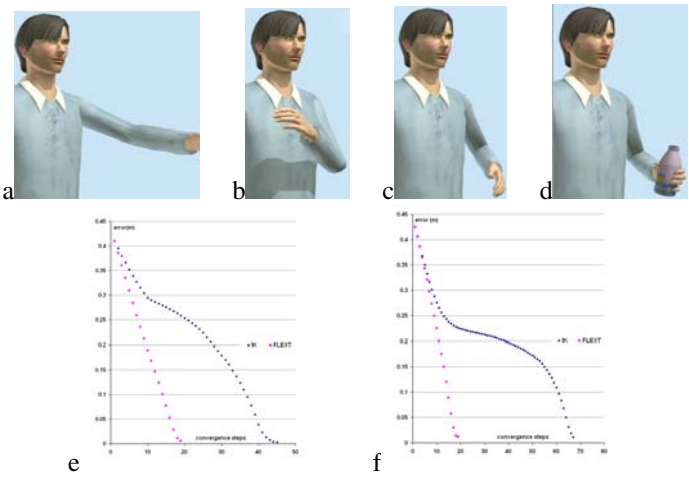


Fig. 4. (a)initial posture, (b) final front flexing, (c) final side flexing (d) final side flexing combined with orientation constraint, (e) front flexing error, (f) side flexing error, IK position constraint (dark diamond), FLEXT constraint (light square)

We have also tested the FLEXT constraint for real-time interactions in which a reduced set of 24 active optical markers were used to track the full body. The critical difference with the previous tests is the nature of the leg control ; although leg and arm have the same limb structure they do not have the same type of task. In a context of "standing reach" the arm effector moves whereas the feet remain static with respect to the floor. Instead the leg root is moving due to the movement of the pelvis and the upper body. Fortunately the integration of the FLEXT constraint into the prioritized IK framework allows to use it transparently for both type of use.

Fig. 5. reduced set of marker used for the real-time full-body interactions

7 Conclusion

Although exploiting a fully analytic approach would allow to solve the postural control in a single computing step, we prefer to comply with the requirement of the small task variation of the linearized IK. Indeed a too large posture variation resulting from the FLEXT constraint would induce a very low validity of the task compensation stage (see [BB04]). Nevertheless the convergence of the overall postural control is improved as shown in the comparative tests. Additional full-body exploitation of this approach will be done in complex environment with obstacles and in conjunction with the use of data-based methods, such as the "motion constraint" introduced in [RB09].

Acknowledgements

This work has been supported by the Swiss National Foundation under the grant N° 200020-117706. Thanks to Mireille for the character design and to Achille Peternier for the MVisio viewer and the CAVE software environment.

References

[NvDR08] Nijholt, A., van Dijk, B., Reidsma, D.: Design of experience and flow in movement-based interaction. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, pp. 166–175. Springer, Heidelberg (2008)

[LLRE07] Loke, L., Larssen, A.T., Robertson, T., Edwards, J.: Understanding movement for interaction design: frameworks and approaches. Pers. Ubiquit. Comput. 11, 691–701 (2007)

[BMT09] Boulic, R., Maupu, D., Thalmann, D.: On Scaling Strategies for the Full Body Interaction with Virtual Mannequins. Journal Interacting with Computers, Special Issue on Enactive Interfaces 21(1-2), 11–25 (2009)

[BB04] Baerlocher, P., Boulic, R.: An Inverse Kinematic Architecture Enforcing an Arbitrary Number of Strict Priority Levels. The Visual Computer 20(6), 402–417 (2004)

[PMMRTB09] Peinado, M., Meziat, D., Maupu, D., Raunhardt, D., Thalmann, D., Boulic, R.: Full-body Avatar Control with Environment Awareness. IEEE CGA 29(3) (May-June 2009)

[S98] Sturman, D.: Computer Puppetry. IEEE CGA 18(1), 38–45 (1998)

[MGBB07] Michoud, B., Guillou, E., Briceño, H., Bouakaz, S.: Real-Time and Marker-Free 3D Motion Capture for Home Entertainment Oriented Applications. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part I. LNCS, vol. 4843, pp. 678–687. Springer, Heidelberg (2007)

[A96] Ascension, The Flock of Birds Installation and Operation Guide, Ascension Technology Corporation POB 527 Burlington, Vermont 05402 (802), pp. 860–6440 (1996)

[MBRT99] Molet, T., Boulic, R., Rezzonico, S., Thalmann, D.: An architecture for immersive evaluation of complex human tasks. IEEE TRA 15(3) (1999)

[M*.99] Molet, T., Aubel, A., Capin, T., Carion, S., Lee, E., Magnenat-Thalmann, N., Noser, H., Pandzic, I., Sannier, I., Thalmann, D.: Anyone for tennis? Presence 8(2), 140–156 (1999)

[P], `http://www.phasespace.com`

[V], http://www.vicon.com
[K85] Korein, J.U.: A Geometric Investigation of Reach. MIT Press, Cambridge (1985)
[TGB00] Tolani, D., Goswami, A., Badler, N.I.: Real-Time Inverse Kinematics Techniques for An-thropomorphic Limbs. Graphical Models 62(5), 353–388 (2000)
[KMA05] Kulpa, R., Multon, F., Arnaldi, B.: Morphology-independent Representation of Motions for Interactive Human-like Animation. Computer Graphics Forum 24(3) (2005)
Kallmann, M.: Analytical inverse kinematics with body posture control. Comp. Anim. Virtual Worlds 19(2), 79–91 (2008)
[UPBS08] Unzueta, L., Peinado, M., Boulic, R., Suescun, A.: Full-Body Performance Animation with Sequential Inverse Kinematics. Graphical Models 70(5), 87–104
[BMW87] Badler, N., Manoochehri, K.H., Walters, G.: Articulated figure positioning by multiple constraints. IEEE Comput. Graph. Appl. 7(6), 28–38 (1987)
[PB91] Phillips, C.B., Badler, N.: Interactive behaviors for bipedal articulated figures. Comput. Graph. 25(4), 359–362 (1991)
[ZB94] Zhao, J., Badler, N.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. ACM Trans. Graph. 13(4), 313–336 (1994)
[MBT08] Maupu, D., Boulic, R., Thalmann, D.: Characterizing full-body reach duration across task and viewpoint modalities. On-line Journal of Virtual Reality and Broadcasting 5(15) (November 2008)
[M90] Maciejewski, A.A.: Dealing with the ill-conditioned equations of motion for articulated figures. IEEE CGA 10(3), 63–71 (1990)
Liégeois, A.: Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. IEEE Transaction on Systems, Man and Cybernetics SMC-7(12), 868–871 (1977)
[SS91] Siciliano, B., Slotine, J.J.: A General Framework for Managing Multiple Tasks in Highly Redundant Robotic Systems. In: Proc. of ICAR 1991, vol. 2, pp. 1211–1215 (1991), ISBN 0-7803-0078-5
[BLP06] Boulic, R., Le Callennec, B., Peinado, M.: Challenges in Exploiting Prioritized Inverse Kinematics for Motion Capture and Postural Control. In: Gibet, S., Courty, N., Kamp, J.-F. (eds.) GW 2005. LNCS (LNAI), vol. 3881, pp. 176–187. Springer, Heidelberg (2006)
[G98] Grassia, F.S.: Practical parameterization of rotations using the exponential map. Journal of graphics tools 3(3), 29–48 (1998); AK Peters
[RB09] Raunhardt, D., Boulic, R.: Motion Constraint. The Visual Computer 25, 509–518 (2009)

Appendix A: General Case

In the general case the human anatomy may deviate from the ideal case used in the prior illustrations of the approach. It may happen that the limb articulated structure does not stand exactly in the flexion-extension plane, for example when the middle flexion axis is not exactly perpendicular to the vector linking the root origin $\mathbf{O_r}$ to the middle joint origin $\mathbf{O_m}$. Another frequent deviation comes from the choice of the controlled effector that may not belong to the FLEXT plane, especially if it corresponds to the location of an optical marker.

Figure 6 illustrates the general definition of the FLEXT plane (i.e. passing through the root origin $\mathbf{O_r}$ with the middle flexion axis as normal) in such a non-ideal case to summarize how the key lengths **L1**, **L2**, **L3** and the offset angle $\boldsymbol{\gamma}$ are computed. The

angle γ can be used to detect whether the limb configuration is within the standard limits as illustrated in figure 6 or whether the limb is "overextended", a rare case in which the desired displacement produces a posture variation with opposite sign as the standard case. The overextended case is generally quickly limited by the middle joint limit.

It is important to note that **L1** remains constant as it depends only on the skeleton structure ; in particular it is independent of the twist angle along the first segment linking the limb root to the middle joint because the FLEXT plane rotates together with this twist angle. On the other hand **L2** needs to be updated whenever the twist rotation along the second limb segment changes if the effector is not *on* the twist axis.

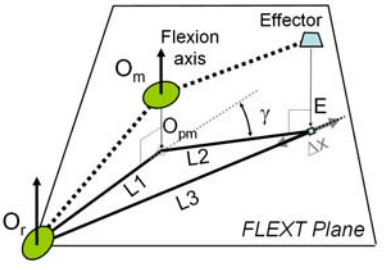


Fig. 6. General case of setting the FLEXT plane and the controlled effector E

Light Space Cascaded Shadow Maps for Large Scale Dynamic Environments

Shang Ma, Xiaohui Liang, Zhuo Yu, and Wei Ren

State Key Lab. of Virtual Reality Technology,
School of Computer Science, Beihang University, Beijing, 100191, PR China
{mashang,lxh}@vrlab.buaa.edu.cn

Abstract. Cascaded shadow maps (CSMs) is an efficient real-time shadow rendering method for large scale dynamic environments. When view direction and light direction are not perpendicular, CSMs unnecessarily render objects to different shadow map textures. This paper presents the light space cascaded shadow maps (LiSCSMs) algorithm to solve CSMs' redundant rendering problem. We propose a light space scene splitting scheme which divides the scene into non-intersecting layers in light space, and a fast shadow map rendering method with irregular frustum clipping and scene organization to generate a shadow map for each layer, ensuring that any shadow sample point never appears in multiple shadow maps. Finally, a succinct shadow determination method is given to produce higher quality shadows when rendering scene. The results show that LiSCSMs effectively improves the efficiency and shadow quality of CSMs by avoiding redundant rendering, and can produce high-quality shadow rendering in large scale dynamic environments with real-time performance.

Keywords: Shadow mapping; Light space; Redundant rendering; Large scale dynamic environment; Image-based rendering.

1 Introduction

Shadows are important to virtual environment. They can not only provide geometry and position information of objects, but also improve realism of the environment. Since high quality shadows require lots of computational resources, research on real-time shadow rendering is still significant.

Shadow mapping [1] is one of the most popular shadow rendering algorithms. Due to its generality and high speed, shadow mapping can create shadows for large scenes and has been widely used in rendering-intensive applications like video games. Like all image-space algorithms based on discrete buffers, shadow mapping suffers from aliasing errors at shadow edges. The aliasing problem is mainly caused by insufficient shadow map sampling. When using shadow maps for large environments with high depth range, aliasing errors become worse since a less number of samples in the shadow map is used to cover larger regions.

Recently, several approaches have been tried to reduce those aliasing artifacts. Cascaded Shadow maps (CSMs) [2] is one of them. In order to fix the aliasing problems, CSMs provide higher resolution of the depth texture near the viewer and lower

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 243–255, 2009.

resolution far away by splitting the camera view frustum into separate partitions and creating a depth-map for each partition. Furthermore, CSMs avoid limitation of the texture maximum resolution on graphic hardware and does not require special tricks to handle extreme cases in warping algorithms [3, 4]. One of the most widely used kind of CSMs is parallel split shadow maps(PSSMs) [5], which splits the view frustum along the view axis with planes parallel to the near and far clip planes.

However, with CSM algorithm in dynamic scenes, when the direction of view and the direction of light are not perpendicular, light sub-frustums are intersected with each other. When generating shadow maps, objects in intersected spaces will be redundantly rendered into different depth buffers. This redundant rendering can cause two major drawbacks: First, because just one shadow map is enough for pixel's depth comparison, redundant sampling of one object into different shadow maps is unnecessary. In large-scale environment, redundant rendering will cost so much time comsuption that could dinger the real-time requirement. Second, redundant rendering leads to non-optimized split selection than renders sub-optimal shadow quality.

In this paper, we present a light space cascaded shadow maps (LiSCSMs) algorithms to resolve redundant rendering in CSMs. First, under the analysis of the causes of redundant rendering, we propose a light space scene splitting scheme that divides the scene into multiple non-intersecting layers. Then, we develop a fast shadow map generation method with an irregular frustum clipping based on geometry shaders to avoid redundant rendering. An object organization method is used to accelerate the shadow map rendering step. Finally, a fast and succinct shadow determination method in pixel shader is given to render scene shadow.

The rest of this paper is organized as follows. In Section 2, we briefly discuss previous work related to shadow mapping. In Section 3, we describe our LiSCSMs scheme for large-scale dynamic virtual environments, including view frustum and light frustum split, light space scene split, shadow maps rendering and scene-shadows synthesis. Section 4 presents the experimental results of our proposed scheme in comparison with CSM algorithms in large-scale dynamic environments. Finally, the conclusion and further work can be found in Section 5.

2 Related Work

Literatures on real-time shadow rendering in computer graphics are vast, and far beyond the scope of this paper. So here we just focus on those most relevant to our algorithm in solving the aliasing problem of shadow mapping. More details about real-time shadow rendering in general are given in the survey [6].

Warping algorithms redistribute shadow map sampling density by using a projection transformation. Perspective Shadow Maps (PSMs) [3] is one popular warping algorithm, in which both the scene and the light are transformed to the post-perspective space. Then the shadow map is generated in this space. Since the scene is warped by perspective projection, perspective aliasing can be significantly decreased. Light Space Perspective Shadow Maps[4] improves perspective reparameterization of PSMs in light space to make a better aliasing distribution over the whole depth range, and avoids some inconvenience of PSMs by converting point light sources to directional light. Logarithmic Perspective Shadow Maps[7] provide the best aliasing

distribution, but logarithmic reparameterization changes lines into curves whose rasterization is not supported by current graphic hardware.

Filtering algorithms filter shadow boundaries to decrease aliasing problems of shadow mapping. Percentage Closer Filtering (PCF)[8] filters the results of the depth comparisons during shadow determination, to help anti-aliasing around shadow boundaries. As recent improvements of PCF, Variance Shadow Maps (VSMs)[9], Layered Variance Shadow Maps[10] and Exponential Shadow Maps[11] reconstruct shadow determination function allowing pre-filtering to speed up shadow filtering. Combination of CSMs and filtering algorithms is a very popular technique to obtain realistic shadows in game engines, such as CryEngine2.

Partitioning algorithms split the scene into multiple parts and generate a shadow map for each part. Adaptive shadow maps(ASMs)[12] store shadow maps as a quadtree structure and removes aliasing errors by refining leaves at shadow boundaries adaptively. However, the refinement operations require many rendering passes and are not feasible for real-time applications. Resolution matched shadow maps [13] and Queried virtual shadow maps [14] improve the efficiency of ASMs by using data structures more suitable for graphic hardware acceleration. CSMs[2,5] is a z-partitioning algorithm that splits the scene along the view axis. Zhang[15,16] improves CSMs' efficiency by reducing rendering passes based on Directx10 level GPUs. Practical Cascaded Shadow Maps (PCSMs) [17] amend the non-optimized shadow map selection in shadow determining step to improve shadow quality.

3 Light Space Cascaded Shadow Maps

3.1 Motivation

As shown in Figure 1, CSMs first split the view frustum V into multiple parts $\{ V_i \}$ with planes parallel to the near and far clip planes, and the light frustum W into multiple smaller ones $\{ W_i \}$, each of which covers the objects potentially casting shadows into V_i . And then $\{ W_i \}$ are rendered to shadow maps $\{ T_i \}$ respectively.

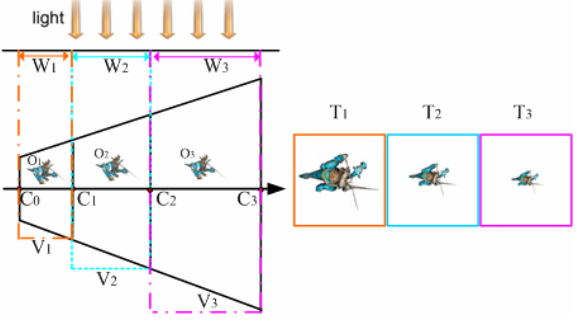


Fig. 1. CSMs' view frustum splitting and light frustum

With CSMs' splitting scheme, when the direction of view and the direction of the light are not perpendicular, light sub-frustums intersect with each others. Since these intersected sub-frustums are used to create shadow maps, objects which belong to multiple sub-frustums must be rendered into corresponding different shadow maps. This causes redundant rendering. Figure 2 illustrates how the include angle θ between view and light causes redundant rendering. There is only one especial case when θ equals to 90°that redundant rendering does not happen. As soon as θ is decreasing to 0°or increasing to 180°, redundant rendering occurs. When view direction and light direction are parallel, the light sub-frustums are embedded with each other, resulting in most objects being rendered redundantly.

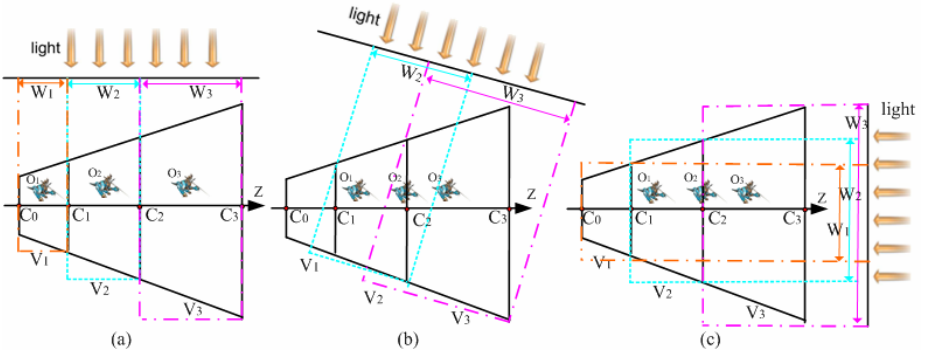


Fig. 2. Redundant rendering cases under different include angle θ between view and light: (a).θ=90°there is no redundant rendering. (b) θ=120°light sub-frustums W_2 and W_3 intersected ,object O_2 and part of object O_3 are rendered twice. (c) θ=180°the worst case that most redundant rendering occurs.

Samples of redundantly rendered objects exist in multiple shadow maps. When rendering scene shadows, CSM algorithm uses the eye's view-space depth to select the shadow map for every pixel. This selection might not be optimal, for example, consider a sphere O which casts shadows in the second split V_2 as shown in Figure 3. It is rendered in all three shadow maps (T_1, T_2, T_3). In order to get the highest shadow quality at the shadow positions, we should choose the first shadow map rather than the second one.

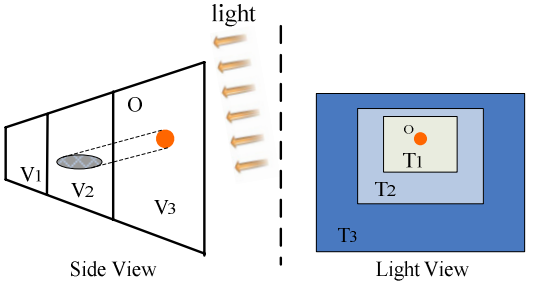


Fig. 3. The best shadow map selection is T_1 rather than T_2 in CSMs

In dynamic environment where both position and direction of view point and light are changing, CSMs' redundant rendering problem widely exists. So resolving it can effectively improve the efficiency and quality of CSM algorithm.

3.2 LiSCSMs' Overview

The processing steps of the LiSCSMs scheme are outlined in the following:

STEP 1 Split the view frustum V into multiple depth parts $\{ V_i \}$.

STEP 2 Split the light frustum W into multiple sub-frustums $\{ W_i \}$ by using $\{ V_i \}$.

STEP 3 Light space scene splitting that splits scene into non-intersecting layers $\{ L_i \}$ by using $\{ W_i \}$.

STEP 4 Render a shadow map for each L_i.

STEP 5 Render scene shadows for the whole scene.

The first two steps follow the CSM algorithm as described in [5] for splitting of view frustum and light frustum. We describe each step's details in next parts of this section.

3.3 Splitting View Frustum and Light Frustum

The CSM algorithm splits the view frustum along the view axis using planes parallel to the near and far clip planes. If the view frustum is divided into m splits, the depth value c_i of the split positions on the view axis in view coordinate can be computed by this equation:

$$
\begin{aligned}
&C_i = \lambda C_i^{\log} + (1-\lambda) C_i^{\text{uni}} \\
&C_i^{\log} = n\left(\frac{f}{n}\right)^{\frac{i}{m}} \qquad 0 \le \lambda \le 1, 0 \le i \le m \\
&C_i^{\text{uni}} = n + \frac{(f-n)i}{m}
\end{aligned}
\tag{1}
$$

where f, n represent the far and near clip plane of view frustum V and the weight λ adjusts the split positions according to practical requirements of the application. $c^{\log}$ and c^{uni} are two classic splitting schemes. You can get the details of view frustum splitting scheme from reference [5].

The principle of light frustum splitting is that each sub-frustum must cover all objects potentially casting shadows into corresponding view frustum part. [15] presents two methods: the scene independent method and the scene dependent method. The scene independent method calculates light sub-frustums by using axis-aligned bounding boxes of the splits of the view frustum in the light's clip space. The scene dependent method creates light sub-frustums that only contain shadow castors and therefore optimize the usage of the texture resolution.

In this paper, we use directional light sources and the scene independent method to illustrate the problems and our ideas for clarity. And they fit for point light sources too, because point light sources can be unified as directional light sources in the light's post-perspective space [4]. Scene dependent method can reduce a small part of redundant rendering, and we use this method in ours experiments.

3.4 Light Space Scene Splitting

To solve redundant rendering problem, we present a light space scene splitting scheme based on CSMs light frustum splitting, as shown in Figure 4 (a). In light space, we split the scene using light sub-frustums created like in the CSM algorithm to make sure layers do not intersect each other whatever the include angle between eye view and light is. Let L_i be the ith layer splitted by LiSCSMs, L_i can be computed by

$$\begin{aligned} L_1 &= W_1 \\ L_i &= W_i - (W_i \cap W_{i-1}) \quad (1 < i \le m) \end{aligned} \tag{2}$$

Where $W_i \cap W_{i-1}$ is the intersected space of W_i and W_{i-1}.

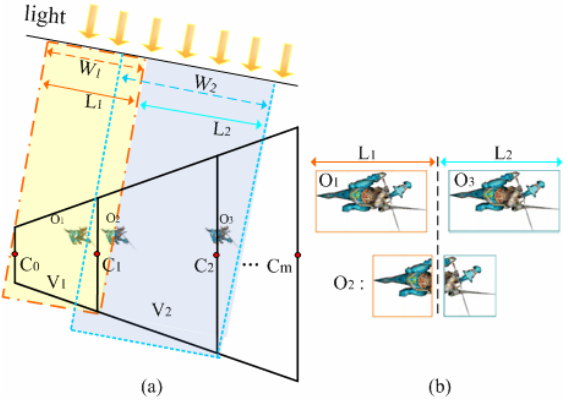


Fig. 4. Light space scene splitting. CSMs generate shadow maps for each Wi. In situation (a), O_1 and O_2 are redundantly rendered in CSMs, as W_1 crosses W_2. For we generate shadow maps based on non-intersecting layers { Li }, redundant rendering does not occur as shown in (b).

We use Li in LiSCSMs as both Vi and Wi in CSMs. In shadow maps rendering step, only objects included in L_i are rendered into shadow map Ti, which ensures that no sample of the scene is rendered multiply. When rendering scene shadows, the pixels in Li are combined with the information in shadow map Ti to perform the best shadow test. As shown in figure 4 (b), objects $O1$, $O2$, $O3$ will be redundantly rendered by CSMs. However, in LiSCSMs no redundant rendering occurs.

3.5 Rendering Shadow Maps

[16] proposes two object-cloning methods to create multiple shadow maps through one pass scene rendering: instancing method clones objects' vertexes in the vertex shader and geometry shader cloning method reproduces polygons in the geometry shader. In this paper, we improve CSMs' instancing method to render shadow maps faster based on our light space scene splitting scheme.

3.5.1 Irregular Frustum Clipping

Our light space scene splitting method produces irregular splits, such as L_2 in figure 4(a). To eliminate redundant rendering, we should use this irregular space area as light frustum to clip polygons in render pipeline. Figure 5 illuminates the pipeline of shadow map generating in comparison with CSMs' method.

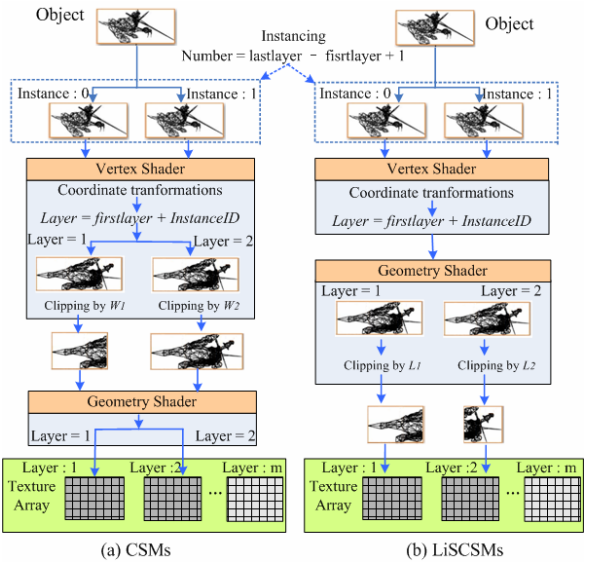


Fig. 5. Comparison of shadow map rendering pipelines

Objects crossing multiple light sub-frustums are first copied by technique of instancing and then rendered into corresponding shadow maps. In the geometry shader, we discard the part of the object which is located in previous light sub-frustum with higher shadow map resolution, and render the remaining part, as shown in figure 6.

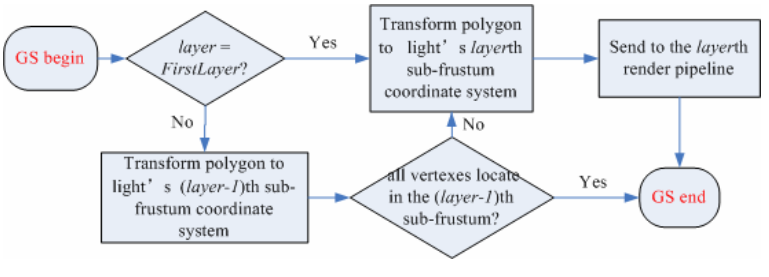


Fig. 6. Irregular frustum clipping in geometry shader

In order to avoid excessive complex computation in geometry shader, we simplify the clipping to only discard polygons that are entirely contained in the previous layer, which means polygons crossing the boundary of layers are still rendered multiple

times. However, since this kind of polygons is a very small part in the whole scene, our simplification does not affect shadow quality in practical applications.

3.5.2 Optimizing Rendering by Object Organization

We find that only objects crossing the boundary of the $\{ L_i \}$ need additional processing on the GPU. On the contrary, objects exclusively contained in one layer can be just rendered into that layer's shadow map directly. So to further improve the efficiency of our algorithms, we can reduce the number of objects, which shader programs applied on, by organizing objects with their $FirstLayer$ and $LastLayer$.

Let S_{ij} be the set of all objects whose attributes $FirstLayer = i$ and $LastLayer = j$. We only apply shader programs to objects in S_{ij} $(1 \le i < j \le m)$. Since objects in S_{ij} share same parameter on the GPU, this classification can accelerate parameters passing to GPU. Main procedure of optimized shadow maps rendering works as follows:

Step 1. Compute each object's attributes $FirstLayer$ and $LastLayer$, and put it into S_{ij} $(i = FirstLayer, j = LastLayer)$.

Step 2. Render objects in S_{ii} $(1 \le i \le m)$ to T_i, the i-th layer of shadow maps.

Step 3. Render objects in S_{ij} $(1 \le i < j \le m)$:

Step 3.1. Compute objects' instance numbers and pass parameters to GPU.

Step 3.2. In vertex shader, translate each vertex into light coordinate system, and compute its $Layer$ of light sub-frustums.

Step 3.3. In geometry shader, clip polygons by light space scene splitting.

3.6 Rendering Scene Shadow

The CSMs use the eye's view-space depth to select the shadow map for every pixel. To address non-optimized selection problem, Practical Cascaded Shadow Maps (PCSMs) [17] gives a method that always selects the shadow map with the highest resolution. However, finding the best shadow map in PCSMs uses many condition branches in pixel shader that are very time consuming. In LiSCSMs, the shapes of light space scene layers are irregular, so more computations are needed to select shadow map for each pixel.

Based on light space light splitting, we present a succinct method to render scene shadows with highest shadow quality in pixel shader. The main procedure is as follows:

Step 1. Translate each pixel to all shadow maps' coordinate systems and perform a shadow test that returns 0 if the pixel is in shadow and otherwise returns 1.

Step 2. Multiply all returns and use the result as the final indication whether the pixel is shadowed or not.

List 1 illustrates how to achieve our method in the pixel shader, and Listing 2 shows the PCSMs scheme. Compared to PCSMs, without shadow map selection step, our method simplifies computations in pixel shader. And we attest its efficiency by experiments in next section.

```
                Constant:   m           the number of layers
                uniform :   smArray     shadow map texture array
                Input:      TexCoord[m] pixel' s texture  coordinate in each layers
                Output:     Value       0 represents pixel is shadowed
// Pixel Shader  PCSMs                          //Pixel Shader  LiSCSMs
Program:                                        Program:
FOR(i=1 to m)                                     FOR(i=1 to m)
   vCoord = TexCoord[i].xyzz /TexCoord[i].w;        Coord = TexCoord[i].xyzz /TexCoord[i].w;
   IF(vCoord.x >= 0 && vCoord.x <=1                 vCoord.z = float(i);
      && vCoord.y >=0 && vCoord.y <= 1) THEN        Value *= shadow2DArray(smArray, vCoord).x;
      vCoord.z = float(i);                        END FOR
      Value = shadow2DArray(smArray, vCoord).x;
      break;
   END IF
END FOR
```

List 1. Shadow determination methods in pixel shader

Since our method will always query all shadow maps, it will increase memory bandwidth on GPU. So we recommend using VSMs that needs only two samples around the projected position instead of PCF to generate soft shadows.

4 Experiments and Analysis

We have developed the Light Space Cascaded Shadow Maps by using OpenGL and GLSL. We run the shadow rendering tests using 800*600 image resolution, and FPS is measured by an Intel Core 2 2.6GHz CPU with 2G RAM, and NVidia GeForce 8800GTX GPU with 512M video memory.

We compare our LiSCSMs with standard shadow maps (SSMs), CSMs (or PSSMs)[16], and latest CSMs based technique , Practical Cascaded Shadow Maps (PCSMs)[17]. We adapted models from [16] as our large scale dynamic environments. In experiment, split count M is 4 and shadow map resolution is 1024*1024.

Figure 7 compares LiSCSMs with CSMs concerning the shadow quality and shadow maps created by them. CSMs' shadow maps illustrate that redundant rendering happens, like house models existing in multiple shadow maps. On other hand, LiSCSMs' shadow maps show that we have resolved redundant rendering problems. Furthermore, LiSCSMs improve shadow quality, for instance the shadow area marked

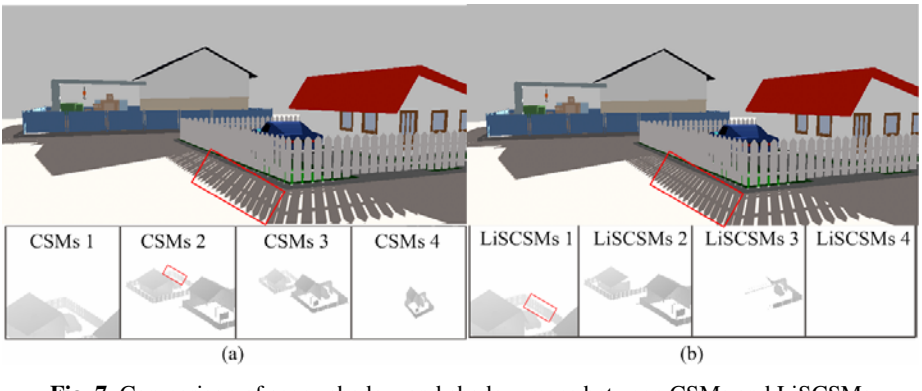
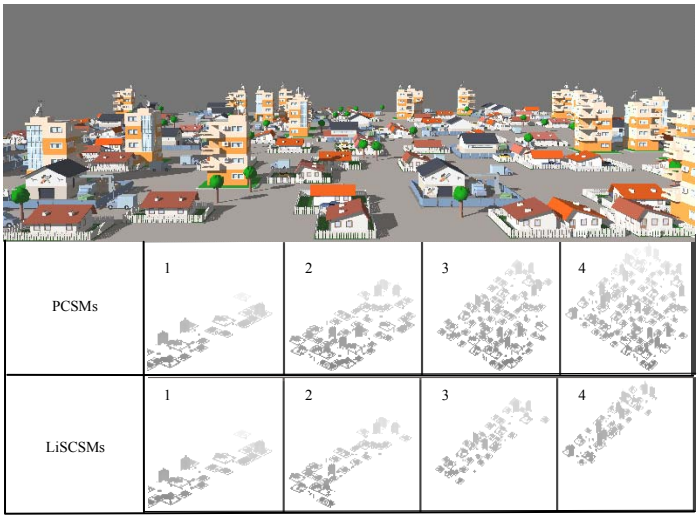
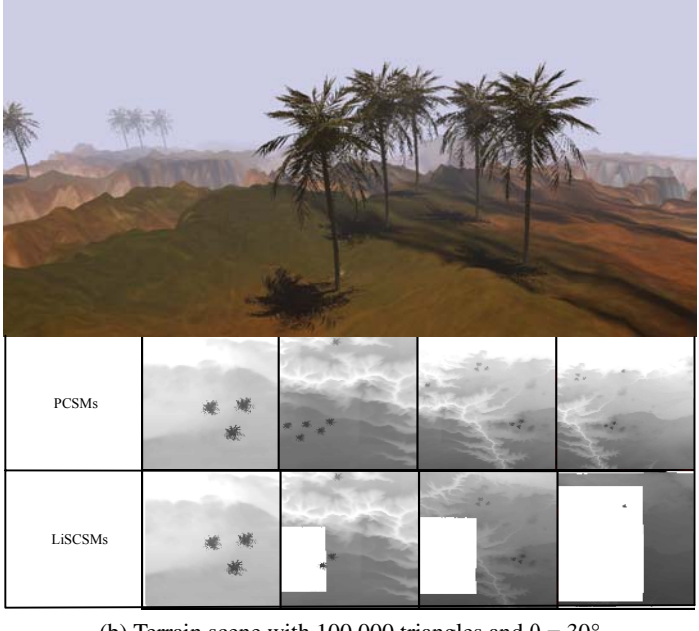


Fig. 7. Comparison of scene shadow and shadow maps between CSMs and LiSCSMs



(a) Town scene with 500 objects and 300,000 triangles in total and θ = 45°



(b) Terrain scene with 100,000 triangles and θ = 30°

Fig. 8. Scene shadows and shadow maps of PCSMs and LiSCSMs

by red rectangle In this case, the fence model is redundantly rendered, but CSMs select the second layer of shadow maps not the optimized first layer, to produce scene shadows. On the contrary, LiSCSMs store only model's highest resolution in one certain shadow map and use it for shadow determination. So our method creates a higher quality of the fence's shadow than CSMs does.

Figure 8 demonstrates pictures rendered using LiSCSMs and PCSMs in different type large scale scenes.As shown in Figure 8(a), LiSCSMs and PCSMs render the same quality of scene shadows. But comparing the shadow maps, LiSCSMs render far less objects than PCSMs, and will save a lot of rendering time. Dealing with the terrain scene that not completely be made up of objects, shown in Figure 8(b), we consider the terrain as one big object and apply LiSCSMs as usual. In Figure 8(b), we can see that LiSCSMs still render less geometries.

Figure 9 further illustrates how the scale of scene influence FPS of SSMs , LiSCSMs and PCSMs. The experiment results show that LiSCMSs speed up CSMs significantly, especially in large scale environments. As augmenting scene's scale, the speedup of LiSCSMs on CSMs is increasing from 1.3 to 1.7. FPS of SSMs can be seen as the upper limit of any shadow mapping algorithm in our experiment situation. As the number of triangles increasing, the speedup of SSMs on LiSCMSs is decreasing and gradually closes to 1.3.

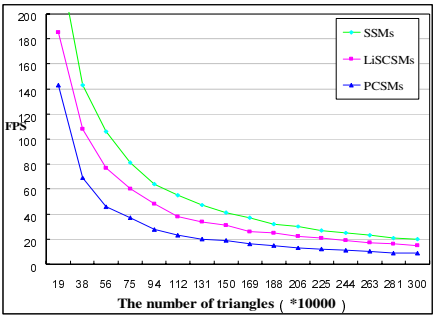


Fig. 9. FPS of SSMs, LiSCSMs and PCSMs in terms of the number of triangles in the scen

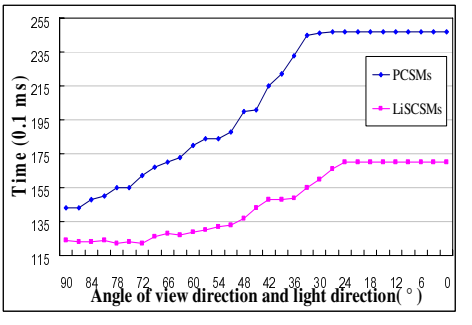


Fig. 10. Comparison of FPS of LiSCSMs and PCSMs in terms of θ

In figure 10, the fixed scene contains 500,000 triangles. The diminishing of θ slows down both algorithms. The reason for this concerning PCSMs is that the smaller θ is, the larger space light sub-frustums intersect and more redundant rending happens. In LiSCSMs, the slowdown occurs because more objects belong to S_{ij} $(1 \le i < j \le m)$ which result in more computations on the GPU. Without redundant rendering, LiSCSMs is always faster than CSMs when θ is changing.

Table 1. Time comparision of each step between LiSCSMs and PCSMs

Steps' running time (ms) / Alogrithms	Split view frustum & find shadow receivers	Split light frustum & find shadow castors	Objects Oragnization	Pass parameters to GPU	Render shadow maps	Scene shadow rendering	Total
LiSCSMs	3.2	7.9	0.2	0.1	1.3	1.8	14.5
PCSMs	3.2	7.9	-	1.1	5.1	3.4	20.7

Finally, we focus on how LiSCSMs speed up CSMs, Table 1 gives the time that each algorithm spends in one of the computational steps in figure 10 when θ is 45°. In shadow map rendering step, we reduce time consumption mainly by eliminating redundant rendering. Besides, we apply shader programs only to objects that cross multiple layers, and reduce parameter translations from CPU to GPU. In scene shadow rendering step, the experiment data can prove that our shadow determination method utilizes GPU more efficiency than PCSMs.

5 Conclusion

This paper developed the Light Space Cascaded Shadow Maps algorithm to resolve redundant rendering problem of CSMs. First, we split the scene into multiple non-intersecting parts using light sub-frustums. And then we proposed a fast method to generate shadow maps for each layer by an irregular frustum clipping based on GPU and objects organization. Finally, a succinct shadow determination method in pixel shader is given to rendering scene shadows. Experimental results have shown that LiSCSMs can improve the efficiency and shadow quality of CSMs, and produce high-quality shadows in large scale dynamic scenes with real-time performance.

In our future work, we can consider combining filtering shadow algorithms, like VSMs, with LiSCSMs to create soft shadows. Since we organize scene by scene graph, we are also interested in accelerate performance by space partition technique, like Octree, on complex models that are less easily subdivided into separate objects.

Acknowledgement

This paper is supported by National Science Foundation of China (Grant No. 60873159),NCET(Grant No. NCET-07-0039), National High-Tech Research &Development Program of China (Grant No. 2006AA01Z333).

References

1. Williams, L.: Casting Curved Shadows on Curved Surfaces. In: Computer Graphics (Proceedings of SIGGRAPH 1978), vol. 12(3), pp. 270–274 (1978)
2. Engel, W.: Cascaded shadow maps. In: Engel, W. (ed.) ShaderX5, pp. 197–206. Charles River Media (2007)
3. Stamminger, M., Drettakis, G.: Perspective shadow maps. In: Proceedings of ACM SIGGRAPH, pp. 557–562 (2002)
4. Wimmer, M., Scherzer, D., Purgathofer, W.: Light space perspective shadow map. In: Proceedings of the Eurographics Symposium on Rendering, pp. 143–152 (2004)
5. Fan, Z., Sun, H., Xu, L., Lee, K.-L.: Parallel-Split Shadow Maps for Large-Scale Virtual Environments. In: Proceedings of ACM International Conference on Virtual Reality Continuum and Its Applications, pp. 311–318 (2006)
6. Hasenfratz, J.M., Lapierre, M., Holzschuch, N., Sillion, F.: A survey of real-time soft shadows algorithms. In: Eurographics, Eurographics, Eurographics (2003)

7. Lloyd, D.B., Govindaraju, N.K., Quammen, C., Molnar, S.E., Manocha, D.: Logarithmic perspective shadow maps. ACM Trans. Graph. 27(4) (2008)
8. Reeves, W., Salesin, D., Cook, R.: Rendering anti-aliased shadows with depth maps. In: Computer Graphics (ACM SIGGRAPH 1987 Proceedings), vol. 21, pp. 283–291 (1987)
9. Donnelly, W., Lauritzen, A.: Variance shadow maps. In: SI3D 2006: Proceedings of the 2006 symposium on Interactive 3D graphics and games, pp. 161–165 (2006)
10. Donnelly, W., Lauritzen, A.: Layered variance shadow maps. In: Proceedings of graphics interface 2008, pp. 139–146 (2008)
11. Salvi, M.: Rendering filtered shadows with exponential shadow maps. In: ShaderX6, pp. 257–274 (2008)
12. Fernando, R., Fernandez, S., Bala, K., Greenberg, D.: Adaptive shadow maps. In: Proceedings of ACM SIGGRAPH 2001, pp. 387–390 (2001)
13. Lefohn, A.E., Sengupta, S., Owens, J.D.: Resolution matched shadow maps. ACM Transactions on Graphics 26(4), 20:1–20:17
14. Giegl, M., Wimmer, M.: Queried virtual shadow maps. In: Proceedings of ACM SIGGRAPH 2007 Symposium on Interactive 3D Graphics and Games, pp. 65–72. ACM Press, New York (2007b)
15. Fan, Z., Sun, H., Xu, L., Lee, K.-L.: Hardware-Accelerated Parallel-Split Shadow Maps. International Journal of Image and Graphics 8(2), 223–241 (2008)
16. Nguyen, H.: Parallel-Split Shadow Maps on Programmable GPUs. In: GPU Gems 3. Addison-Wesley, Reading (2007)
17. Engel, W.: Practical cascaded shadow maps. In: Engel, W. (ed.) ShaderX7, pp. 305–330. Charles River Media (2009)

Practical and Scalable Transmission of Segmented Video Sequences to Multiple Players Using H.264

Peter Quax, Fabian Di Fiore, Panagiotis Issaris,
Wim Lamotte, and Frank van Reeth

Hasselt University - tUL - IBBT
Expertise Centre for Digital Media
Wetenschapspark 2
BE-3590 Diepenbeek
Belgium
{peter.quax,fabian.difiore,panagiotis.issaris,
wim.lamotte,frank.vanreeth}@uhasselt.be
http://www.edm.uhasselt.be

Abstract. We present a practical way to distribute viewports on the same video sequence to large amounts of players. Each of them has personal preferences to be met or is limited by the physical properties of his/her device (e.g., screen size of a PDA or processing power of a mobile phone). Instead of taking the naïve approach, in which sections of the video sequence are decoded and re-encoded for each of the clients, we have exploited advanced features offered by the H.264 codec to enable selection of parts of the video sequence by directly manipulating the encoder-generated bitstream. At the same time, we have overcome several practical issues presented by the fact that support for these features is sadly lacking from the state-of-the-art encoders available on the market. Two alternative solutions are discussed and have been implemented, enabling the generation of measurement results and comparison to alternative approaches.

Keywords: Video coding, Transmission, Remote Rendering.

1 Introduction

Motivation. The integration of multimedia streams of various nature is an important feature of many of today's games. While audio communication is widely supported — at least for some genres — the same is not yet true for video. The causes for this are many, but the required processing power and efficient distribution methods are probably the main culprits. Nevertheless, it opens the door for many interesting applications. The successful integration of video sequences for games is not limited to inter-person communication: various other applications, such as omnidirectional video [1], the replacement of traditional computer-generated background images by real-life captured sequences

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 256–267, 2009.

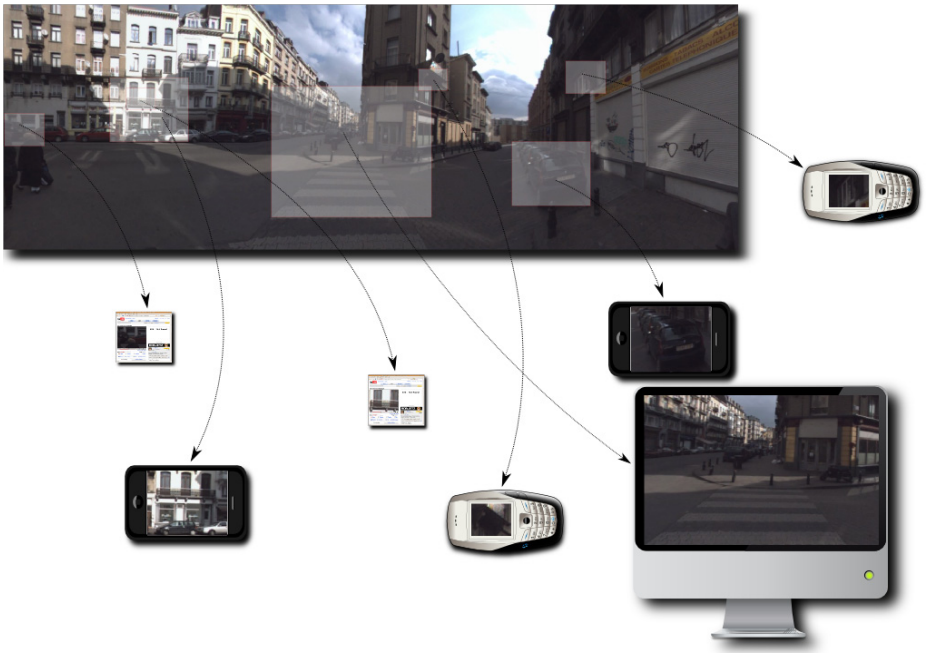


Fig. 1. Concept of distributing viewports on the same video sequence to large amounts of players

or enabling low-power devices to play state of the art games through remote rendering can be envisaged. This paper considers the case in which parts of the same video sequence(s) need to be transmitted to a number of players (see figure 1). Such a condition exists, for example, in remote rendering setups [2,3,4] — in which a high number of individual viewports is generated on-the-fly by a high-performance server setup. It would be beneficial if a single encoding step could be used to generate a (large) stream, instead of maintaining multiple instances of the video encoding software to generate the individual viewports. A similar setup can be created for video communication purposes, in which a video conference is used during gameplay (e.g., for FPS or MMO games). In case a centralised server is responsible for generating the video sequences for individual participants (depending on their bandwidth capacity or display size for example), it would again be beneficial to generate a single video sequence that can be segmented at a later time. It should be clear that a system that takes a single (large) sequence and segments it through encoding/decoding parts of the frames would not scale, thus the need arises for being able to crop the desired regions directly from the encoded bitstream.

Contribution. As stated before, this paper focuses on sending parts of the same video sequence to a multitude of devices, each having their own set of characteristics and player preferences. Our technique focuses on selecting areas in a video sequence for individual transmission directly in the encoded bitstream, without

requiring a separate decoding/encoding phase for each of them. In this paper we elaborate on exploiting advanced features that exist in the H.264 standard (e.g., slices) in order to target our applications. In addition, as there is currently little to no support for these features in the available codecs we present a way to work around these practical limitations and to enable an investigation into the efficiency of the approach.

Approach. In this section, we will briefly explain the general outline of the approach. For a more detailed description, we refer to the appropriate parts of the paper. Consider the case in which a single large video sequence is generated by an application, which is subsequently provided as input for a viewport selection service (based on player preferences or device limitations). A naïve way to perform the segmentation would be to decode a specific part of the sequence and provide it as input for an encoding instance. It should be clear that such an approach would not scale to the number of participants that is representative for today's games. Instead, one could provide some additional information in the bitstream representing the large video sequence. Through the use of this information, specific sections can be cropped simply by selecting those elements of the bitstream that are useful, eliminating the need for separate decoding/encoding phases. Of course, there are some requirements that need to be fulfilled in order to be able to just cut out specific sections of the bitstream, one of them being the requirement to remove dependencies between candidate viewpoint areas. This is accomplished by using slices, one of the features offered by the H.264 standard. In case we limit the motion estimation to these slices, they become self-contained and can be cut from the encoded bitstream in an efficient manner. By following through on this approach, we can combine several slices to re-generate complete video sequences, representing different viewports.

Paper Organisation. In section 2, we provide a short introduction to the features of the H.264 standard that are relevant for this paper. Some of the applications are also described in more detail. Section 3 details our specific approach to the problem and discusses two alternative solutions. The advantages and disadvantages of both are presented. A comparison between our approach and others is presented in section 4. Some pointers to possible future work are presented in section 5.

2 Related Work

A video encoder takes a sequence of frames as its input and spits out a — typically much smaller — bitstream conforming to a certain specification. To do so, block-based video codecs first divide each input frame into a grid structure of which each individual area is called a macroblock. The resulting output bitstream consists of a highly compressed representation of these macroblocks. As the representation of each macroblock in the compressed form is of variable length, a bit error in for example the first macroblock leads to the failure to decode any macroblock in the frame.

The H.264 specification [5] provides a feature called a "slice" which allows these macroblocks to be grouped. H.264, also known as MPEG-4/AVC, is the most recent ITU & MPEG video coding standard widely used (Blu-ray, DVB-S2, QuickTime, ...). Because H.264 allows macroblocks to be grouped into multiple slices, each slide becomes individually en/decodable. A coded video bitstream can have a very simple structure using one slice per frame, however, having multiple slices is advantageous for parallel processing and network transmission where errors in a slice will not prevent the correct decoding of the other slices.

In our work, we exploit this feature in order to ignore unnecessary slices without having to re-encode the cropped region.

3 Approach

By regarding each frame of a video as being composed of many independent areas, we could – for each client – select the areas needed according to his viewport. As explained in section 2 the H.264 standard provides a way to divide frames in smaller units by means of slices which are groups of macroblocks belonging to the same frame and contain the actual encoded pixel data. In general, the H.264 specification describes the slice structure as being totally irregular, allowing each frame to be encoded in a different number of slices.

For our approach, we force our encoder to use a more regular slice subdivision with each frame containing the same number of slices, and each slice occupying the same area over all frames. Encoding a video stream using these constraints allows us to instantly know the visual area a specific slice represents, without needing to decode the actual contents.

Figure 2 illustrates several client requests for a particular view and the corresponding selected area of slices enclosing each view request.

Whenever a client wants to view just a particular part of the video, we somehow need to remove the areas of the image falling outside the selected area. The naïve approach of just sending only the selected slices imposes the client with

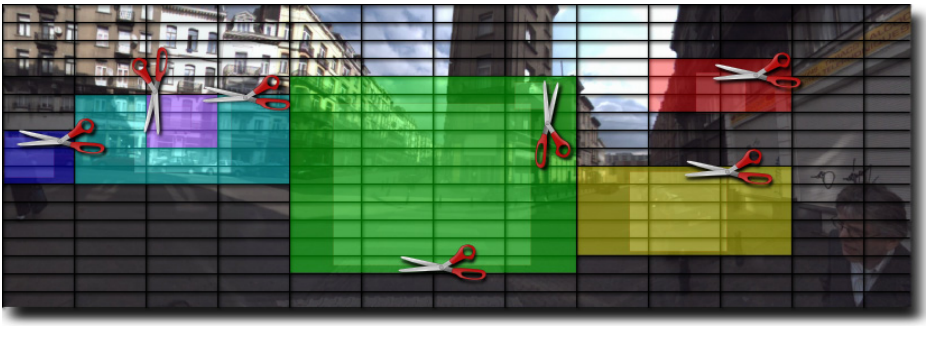


Fig. 2. Each frame is subdivided into a same number of slices. For each client, the areas depicted are needed to enclose the requested view (shown by the smaller area).

a corrupt bitstream which regular decoders can not cope with. Also, cropping out pixel data as is common in known photo and videoprocessing tools would imply compressing the cropped out region for each client individually, since each client might be looking at a different segment of the video (i.e. a different area). Recompressing the cropped out region could be feasible for a few clients, but it cannot be considered scalable.

In the following subsections we elaborate on our specific approach to this problem and discuss two alternative solutions.

3.1 Removing Slices

If we want to view just a particular part of the video, we need to remove the areas of the image falling out the selected area. Ideally, this can be accomplished by removing all unneeded slices (See figure 3).

Unfortunately, deleting one slice affects all following slices. This is because each slice starts with a "first_mb_in_slice" field which contains the macroblock number of the first macroblock coded in this slice. Therefore, as we delete entire slices containing macroblocks, many macroblock numbers disappear and all slices following a deleted slice need to be altered. Moreover, H.264 uses variable bit length encoding meaning that different values are possibly encoded using codes of different length. As a result, changing one single value could imply changes in all bytes that follow.

One further problem is related to the fact that macroblocks are implicitly identified by their number. The data that motion compensation uses to reconstruct the frame is also identified by the same number. When the cropped out area is static (i.e. the client's view frustrum does not move) this causes no harm as the renumbering due to the cropping is the same for each frame, and motion vectors point to the correct data. However, when the cropping window moves (due to a moving view frustrum), problems arise as the numbering of macroblocks is different before and after the movement of the cropping window. Consequently, the

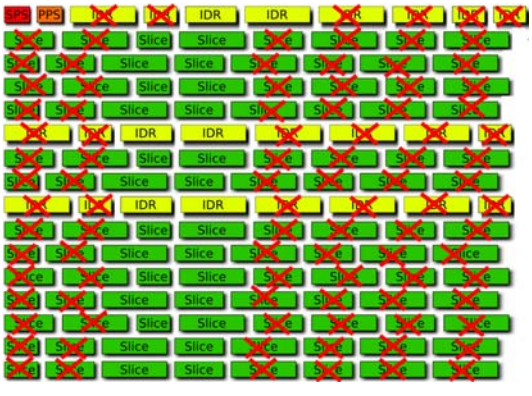
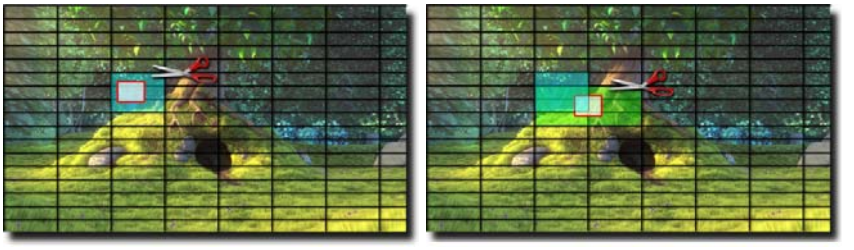


Fig. 3. Removing blocks removes part of a frame. In this case, each row depicts a single frame.



(a) (b)

Fig. 4. Moving the view frustrum can cause the original (a) frame size to change (b)

motion compensation process in the decoder will start using the wrong image data to reconstruct the image. The only way to avoid this problem, is to stall the cropping window until the next intra-frame. This is a full frame used in MPEG-4 AVC encoding (comparable to I-frames) that does not need any information from the frame before it or after it to be played back properly.

Furthermore, moving the view frustum might cause the video frame size to change (see Figure 4). Although not restricted according to the specs, this causes problems with current decoders including VLC, Xine and MPlayer.

To summarise, removing the unnecessary blocks results in the smallest possible stream where the receiving end receives a simple stream with a resolution matching its request without any modifications needed on the client side. The mentioned issues, however, restrain this approach from being usable in real cases.

3.2 Replacing Slices

A different solution, which alleviates the aforementioned issues, uses replacement rather than removal of slices. Slices which would have been removed in the previous solution are now instead replaced by an artificially generated slice of minimal size. Because of this, no slice numbers need to be altered and motion compensation uses the correct numbers. Furthermore, no shifting of bits is needed as the existing and needed slices are sent as is. No header modification is needed either.

As the slices need no modification, this approach is more scalable and each client can be sent the slices it requires, unaltered. The only compensation of this approach is the larger decoded picture buffer size at the decoder side and the need for the decoder to be able to handle a larger number of slices.

The replacement slices can be generated on the fly (but can be cached too). For intra-frames (comparable to I-frames), the replacement slices contain the encoded slice data consisting of a number of gray blocks. Concerning inter-frames (comparable to B- and P-frames), the replacement slices contain encoded slice data using skip-bits, representing data which can be skipped as they are supposedly identical to the same data in the previous frame. This is also illustrated in Figure 5(a).

Replacing each unneeded slice with a generated slice is suboptimal and results in high overhead of headers. Furthermore, each slice might be sent individually which

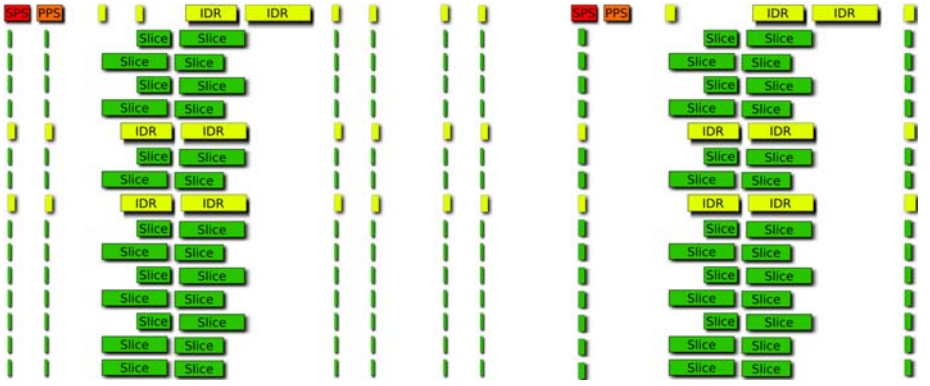


Fig. 5. a) Replacing slices empties part of a frame. b) Grouping empty slices.

would add more overhead due to RTP, UDP, IP and data link layer headers. To this end, we also restructure the bitstream by generating slices containing spans of macroblocks from one unaltered slice to the next unaltered slice (see figure 5(b)).

We can conclude that this approach is very close to the space-efficiency-optimal case (when using the slice-deletion technique), but without the computational overhead and addressing issues.

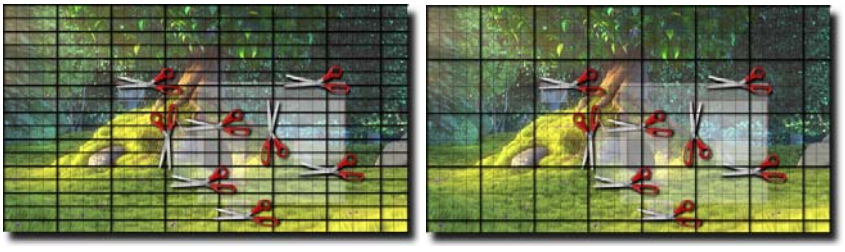
3.3 General Concerns

In this section we elaborate on some general concerns and workarounds, applicable to both approaches.

Frame Dependency. Although slices are independently decodable, they do depend on the previous frame being entirely available. So, if we look at figure 6 we can see the motion vectors pointing out of various slices to the same slices in previous frames. However, as slices can be removed or replaced over frames motion vectors could arise which don't refer to the correct data in the previous frame. To this end, we modified our encoder to restrict motion vectors to the current slice area in the previous frame.

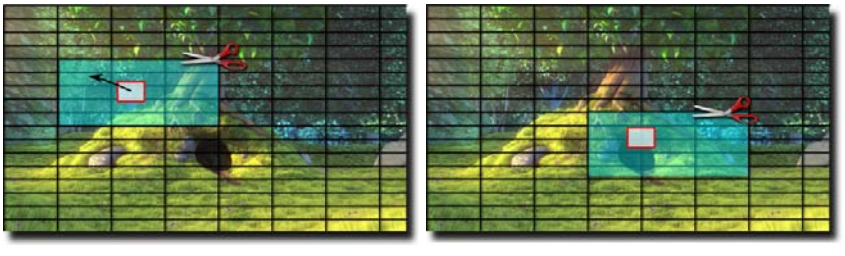
Grid Structure. Note that in the current approach, our slices are rectangular. As H.264 forces grouping of macroblocks to be consecutive in raster scan order our slices cannot contain more than one row of macroblocks. Furthermore, because each macroblock has a height of 16 pixels, the height of our slices is constrained to 16 pixels as well. Therefor, in order not to end up with too many slices each frame is divided into rectangular slices measuring 16 x 192 pixels. It might also be worth investigating the FMO (Flexible Macroblock Ordering) feature of the H.264 specification which allows ordering macroblocks in a more flexible way. This could allow us to implement the grid structure more efficiently by reducing the number of slices needed. Unfortunately, using FMO requires it

Fig. 6. Problematic motion vectors when slices are removed or replaced



(a) (b)

Fig. 7. Comparison of grid structures. a) Currently supported by H.264 encoders and decoders. b) Hypothetical case when employing Flexible Macroblock Ordering (FMO).



(a) (b)

Fig. 8. Moving the view frustrum can cause motion compensation to use data we cropped out in previous frames

to be supported by both the H.264 encoder at the server side and the H.264 decoder at the client side which is not the case at present. Figure 7 depicts our rectangular grid structure and the hypothetical case when employing FMO.

Moving View Frustrum. Moving the view frustum might also cause motion compensation to use data we cropped out in previous frames (see figure 8). The motion vectors could point to a area which is no longer within the cropped area. We can send more slices then strictly needed. If during movement we reach the border of the available area, we need to change the cropping area.

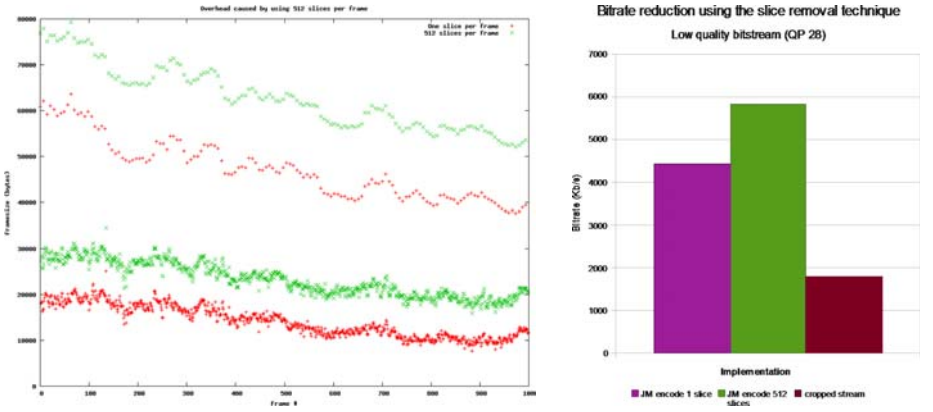
Prediction Artifacts. If we would only send the slices needed to fill the current viewport, any sudden and/or large movement could cause visual artifacts. This is because we would be looking at an area for which we suddenly start receiving

predicted slices for which we do not have the basis of the prediction. In other words, we start receiving predicted slices for that area, but we either never received the initial slices or either received artificial slices which are plain gray. As these artifacts are distracting, we send out more slices (i.e. located around the viewport) than strictly needed to fill the current viewport.

4 Measurement Results and Comparison

In this section we elaborate on some results measuring our optimal proposed approach of replacing unneeded slices by empty ones. All comparisons are made against the official reference encoder (called JM or Joint Model) [6]. Our implementation is also built on the reference encoder as this is the only encoder

Fig. 9. Top row) Snapshot of the input stream. Medium row) Snapshots of two clients with a different view on the input stream. Bottom row) Snapshot of the actual videostream sent to a third client.



(a) (b)

Fig. 10. a) Grid structure overhead caused by using 512 slices per frame. b) The purple bar shows the size of an encode with JM using 1 slice per frame (i.e. a full frame). As indicated by the green bar, using JM to encode using 512 slices per frame causes considerable overhead to represent the same frame. However, the data our cropping server actually sends is significantly lower, which is shown in the dark red bar.

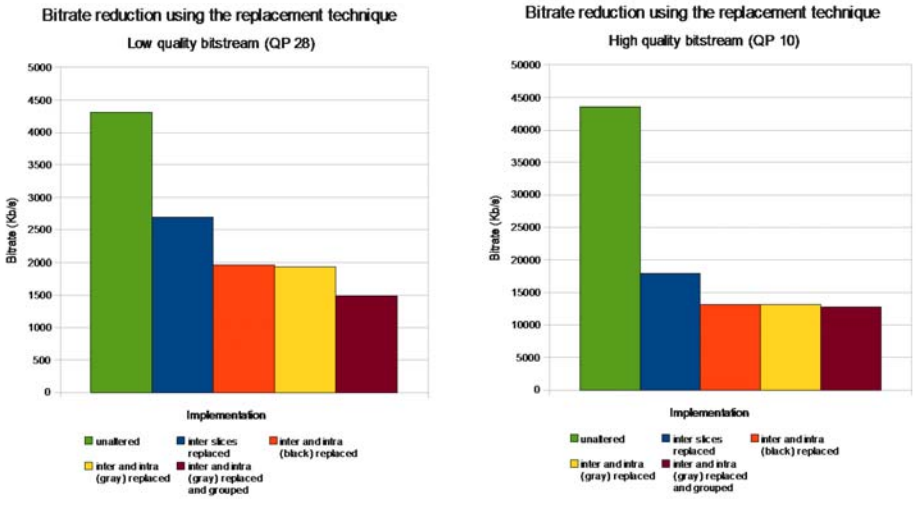
available that fully implements the H.264 standard. Concerning the clients, no specific decoder has to be employed.

All measurements were performed on the omnidirectional video sequence shown in the top row of figure 9. The medium row depicts snapshots of two clients with a different view on the input stream while the bottom row illustrates the actual videostream sent to a third client.

Figure 10(a) shows the grid structure overhead caused by using 512 slices per frame compared to 1 slice per frame (i.e. a full frame). Figure 10(b) depicts the same information (first two bars) but also the significantly lower bitrate (dark red bar) when applying our replacement approach to the 512 slices. Note that once H.264 encoders and decoders support the FMO (Flexible Macroblock Ordering) feature, the overhead due to the amount of slices will be reduced and, hence, the final bitrate as well.

Starting again from an unaltered stream (Figure 11(a), green bar) and applying our approach to just the inter slices (blue bar) already a reduction of 1.6Mbit for a 4.3Mbit stream is achieved. For the used samples, intra slices only represent 1/7 of the data but they are relatively large compared to inter slices. Additionally, using the replacement technique (replacing slices with black colour) on both inter and intra slices yields a considerable advantage (orange bar). The advantage of using the slightly more compact gray block representation is negligible (yellow bar). Finally, regrouping the replaced slices partially alleviates the overhead of the slices' headers (dark red bar).

For higher quality streams (Figure 11(b)), our technique gives even better results. Now, with only replacement of inter-slices we see a reduction of a 43Mbit stream to 18Mbit (blue bar). Further replacement of intra-slices results in a



(a) (b)

Fig. 11. Bitrate reduction using the replacement technique. Green bar = unaltered stream; blue bar = inter slices replaced; orange bar = inter and intra slices replaced (black); yellow bar = inter and intra slices replaced (gray); dark red bar = inter and intra replaced and regrouped. a) Low quality stream. b) High quality stream.

13Mbit stream (orange and yellow bar). Regrouping results in a stream just below 13Mbit (dark red bar).

5 Conclusion and Future Work

We have shown a practical way to distribute parts of a video sequence to large amounts of users. By adapting existing techniques from the H.264 standard, we have shown that it is possible to perform the frame extraction directly from the compressed bitstream, eliminating the need for additional decoding and encoding phases. Test results have shown that the overhead associated with the slice structures is mitigated by the bandwidth reduction in transmitting high-quality video streams.

Additional benefit would be gained if FMO was practically usable and performable with respect to both the encoder and decoder. As this is not yet the case, we will look at the integration in open-source codecs (e.g., libavcodec).

Acknowledgements

Part of the research at EDM is funded by the ERDF (European Regional Development Fund) and the Flemish government. The research presented in this paper is part of the IWT-Teleon project.

References

1. Boult, T.E., Micheals, R.J., Eckmann, M., Gao, X., Power, C., Sablak, S.: Omnidirectional video applications. In: Proceedings of the 8th International Symposium on Intelligent Robotic Systems (2000)
2. Cohen-Or, D., Noimark, Y., Zvi, T.: A server-based interactive remote walkthrough. In: Proceedings of the sixth Eurographics workshop on Multimedia 2001, pp. 75–86. Springer-Verlag New York, Inc., New York (2001)
3. Diepstraten, J., Ertl, T.: Remote Line Rendering for Mobile Devices. In: Proceedings of Computer Graphics International (CGI 2004), pp. 454–461. IEEE, Los Alamitos (2004)
4. Quax, P., Geuns, B., Jehaes, T., Vansichem, G., Lamotte, W.: On the applicability of remote rendering of networked virtual environments on mobile devices. In: Proceedings of the International Conference on Systems and Network Communications, p. 16. IEEE, Los Alamitos (2006)
5. H.264: Advanced video coding for generic audiovisual services. World Wide Web (2009), http://www.itu.int/rec/T-REC-H.264
6. H.264/AVC JM Reference Software. World Wide Web (2009), http://iphome.hhi.de/suehring/tml/

Author Index