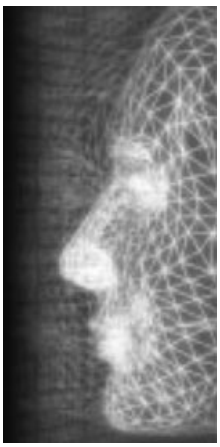
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2008; **19**: 283–293
Published online 11 August 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.242

Adding variation to path planning



By Ioannis Karamouzas and Mark H. Overmars*

Path planning in computer games, whether these are serious or entertainment games, plays an important role in the immersion of a player. Recently, the concept of path planning inside corridors has been introduced and a novel approach under the name Corridor Map Method (CMM) has been proposed. In this paper, we extend the original CMM by creating alternative paths that a character can follow within a corridor. This not only presents a more challenging and less predictable opponent for the player, but also enhances the realism of the gaming and/or training experience. We also discuss how variation in the speed of the animated characters can be integrated into our extended framework, leading to convincing characters that exhibit human like path planning. Copyright © 2008 John Wiley & Sons, Ltd.

Received: 26 June 2008; Accepted: 27 June 2008

KEY WORDS: path planning; Corridor Map Method; variation; Perlin noise

Introduction

Path planning in computer games plays an important role in the immersion of a player. Immersion enables players to feel as if they are present in the game world and live an experience in which "the dynamic form of successful play becomes beautiful and satisfying".[1]

Recently, the concept of path finding inside corridors has been introduced[2] and a general framework, under the name Corridor Map Method (CMM), has been proposed by Geraerts and Overmars.[3] The main advantage of the CMM over the traditional path finding approaches is the ability to compute smooth and aesthetically pleasant paths in real-time and in any complicated environment, which makes it ideal for interactive applications. However, a fixed path is always returned when the same query is performed. This is highly a unnatural behavior that breaks immediately the player's sense of immersion.

An individual does not always take the exact same route, when solving a typical path planning problem in real life. Thus, for a computer game character to be believable, it should follow a (slightly) different path in response to the same query. This not only improves the realism of the game's AI, but also presents a less predictable opponent for the player.

*Correspondence to: I. Karamouzas, Department of Information and Computing Sciences, Center for Games and Virtual Worlds, Utrecht University, Padualaan 14, De Uithof, 3584CH Utrecht, The Netherlands. E-mail: ioannis@cs.uu.nl

Furthermore, people usually show a preference to a specific route while moving to a desired location. Although the paths might globally be the same, locally they can vary from individual to individual. As an example, consider the paths that people follow when walking down a hallway. One person might have the tendency to stay close to the walls or to turn corners more tightly, whereas another might prefer to walk in the middle of the hallway. Therefore, alternative paths should also be generated to indicate different preferred routes that a character can follow, given a specific query.

Another critical issue that can easily destroy the suspension of disbelief and ruin the player's immersion is the speed of the animated characters. Nevertheless, this detail is usually overlooked in discussions of path planning. In particular, many computer games and simulation applications populate their worlds with a large number of characters that all have identical motions and move at the same velocity.

In this paper, we extend the CMM by creating high-quality alternative paths that a character can follow within a corridor. We also demonstrate how variation in the speed of the animated characters can be incorporated into the method, leading to believable characters that navigate through a virtual environment in a natural looking manner.

Previous Work

Over the past years, path finding has been studied extensively in the game development and animation

community, as well as in the robotics community. The main focus of the research was to compute collision-free and short paths as quickly as possible. As a result, little effort has been put to create similar, yet different paths.

In the game development community, the most popular way to plan a path is to divide the environment into a grid that can be searched using A* based algorithms.[4,5] This approach works very well in computer games as it always retrieves the shortest path, if one exists. Some variation of paths can be obtained by varying the A* cost function. However, the resulting variation is not very natural. Also, the returned paths may not belong to the same homotopic class. In addition, grid-based methods are computationally expensive, especially for very large environments, and A* based algorithms usually produce aesthetically unpleasant paths.

In the animation community the most widely used approach is the work of Reynolds on boids (group members), often referred to as flocking.[6] Flocking is a technique that tries to describe the collective behavior of the members in a group using simple local rules. Later, Reynolds enhanced his initial model by incorporating additional algorithmic steering behaviors.[7] His main goal was the creation of convincing autonomous characters. This approach works very well in open environments and generates very natural movements. We can also combine several of the proposed behaviors in order to create alternative paths. However, the local nature of the method gives no guarantees on the paths. For example, the characters can easily get stuck behind obstacles in cluttered environments.

Pedestrian simulation and traffic flow has also received a lot of attention over the past years. Interesting models have been generated that can be used to create different paths for computer-controlled characters. The most popular is the work of Helbing.[8,9] Helbing simulated the behavior of pedestrians and traffic using social rules. His work is closely related to the social potential fields method introduced in Reference [10]. The general idea is that the character is attracted toward its goal position by a strong force. At the same time repulsive forces are exerted on the character to prevent collision with the obstacles and other entities. However, the same problem as in flocking arises, because in all these models only local information is taken into account.

Another way of generating alternative paths would be to use rapidly exploring random trees (RRTs). RRTs originate from the robotics community and are particularly suited for solving single-query motion planning problems in high dimensional spaces.[11–13] Due to their random nature, RRTs seem ideal for creating different paths given a specific query. Unfortunately, the running time can become too large and the resulting paths can be of low quality, rendering this approach inappropriate for gaming applications.

The CMM can overcome the disadvantages of the path finding approaches mentioned above. In the CMM global motion is directed by high-quality roadmaps, whereas social potential fields are used to guide the local motion of the characters inside a corridor. As a result, the CMM can always guarantee that a path, if one exists, will be found, providing at the same time the flexibility of creating locally different paths by taking additional constraints into account. Bearing also in mind that most of the work is done in a preprocessing phase, the CMM is applicable for creating variation of paths in virtual environment applications.

Corridor Map Method

A brief overview of the CMM is presented in this section. For a more thorough explanation we refer the user to Reference [3]. The CMM consists of two phases, an off-line construction phase and an online query phase.

In the construction phase a high-quality roadmap graph is generated, consisting of vertices and edges. The vertices correspond to collision-free positions in a 2D or 3D workspace, whereas the edges correspond to local paths that a character can follow. From the graph, a Corridor Map is constructed by taking into account the clearance information. On every point along a path the clearance is defined as the radius of the largest circle around this point that does not intersect with the environment.

In the query phase a so-called backbone path is computed by running Dijkstra's shortest path algorithm on the graph. Assuming that the character is modeled as a disc or cylinder, the clearance at every point of the backbone path must be at least the radius of the character. This ensures that the path is wide enough so that the character can traverse it.

Next, the backbone path is enhanced with clearance information and a corridor around the path is generated, where the character can move freely. Once the corridor is created, the character plans its path using a force field technique. Forces drive the character inside the corridor and its movements are updated using Newton's laws of motion and a time integration scheme.

To be more specific, in every time step an attraction point on the backbone path is retrieved based on the

DOI: 10.1002/cav

current position of the character. The attraction point is selected as the maximally advanced point on the backbone path, such that the character still remains inside the circle centered at the attraction point and having as radius the clearance of the attraction point.

A force toward the attraction point is acted upon the character, making the character move forward and stay inside the corridor. This force becomes infinite when the character touches the boundary of the circle and zero when the character is positioned on the attraction point. Additional forces can also be incorporated. For example, a repulsive force can be added so that the character can avoid collision with additional obstacles or other entities.

Knowing the final force on the character, its new position is computed by numerically integrating the acceleration and velocity. As a result, the character updates its position and moves forward until it reaches its goal.

Overall Problem Description and Approach

Given a start and a goal position in a virtual environment we want to compute alternative homotopic paths that a character can follow. The backbone path is used to define the homotopic class for the different paths. Two paths Π_0 and Π_1, with endpoints fixed, are said to be homotopic only if one path can be continuously warped into another without intersecting any obstacles.

We will describe two different approaches for creating variation of paths. In the first, a random bias is applied in order to retrieve a path that is slightly different than the fixed path returned by the original CMM. In the second approach, a fixed bias is applied to generate alternative routes that a character may prefer to follow inside the corridor. The bias can either be directed to the right (or left) of the backbone path, leading to the formation of different lanes inside the corridor, or it can follow the direction of the path, generating more natural looking motions. Both approaches can also be combined.

Variation of Paths Using Perlin Noise

This section presents our first approach that generates slightly different paths every time the same path planning problem has to be solved. The quality of the returned paths and the effectiveness of the approach are also discussed.

Implementation

An obvious way of generating different paths would be to add in every time step a random force (bias) to the attractive force that steers the character toward the goal position. However, the resulting paths will be far from realistic, since the character may change direction almost instantaneously.

A more sophisticated approach is the creation of a "directed" random path where the direction of the character at any moment is related to the previous. We will show that a coherent noise function like Perlin noise can be used to control the direction of the random bias and ensure that it will change smoothly in every step of the integration.

Perlin Noise and Random Bias. Perlin noise is a pseudorandom noise function introduced by Perlin.[14] His main goal was to create a wide variety of natural looking textures. In our problem setting, Perlin noise is implemented as a function over a two-dimensional space. In every time step the current point of attraction is given as an input and a direction for the bias is returned that varies pseudo-randomly.

As a feature of Perlin noise, providing the same attraction point will always result in the the same value. A small change in the input point will lead to only a small change in the direction of the bias. Therefore, if the current attraction point lies very close to the attraction point at the previous time step, the new bias will still be heading in almost the same direction as the previous one.

Perlin noise allows us to control the frequency with which the output value is changed. A low frequency results in smooth changes in the direction of the bias, generating aesthetically pleasant paths. In contrast, a too high-frequency noise leads to the retrieval of unrealistic paths, as the character will change its direction at almost every successive time step.

Several Perlin noise functions with different frequencies can also be combined in every step of the numerical integration. This will create more variety in the resulting paths, at a cost of additional computation time. Therefore, in our implementation only one function is used to compute the new direction of the force.

To initialize our implemented function a random seed is used. By changing the seed value, the output of the Perlin noise is changed, leading to the generation of a

DOI: 10.1002/cav

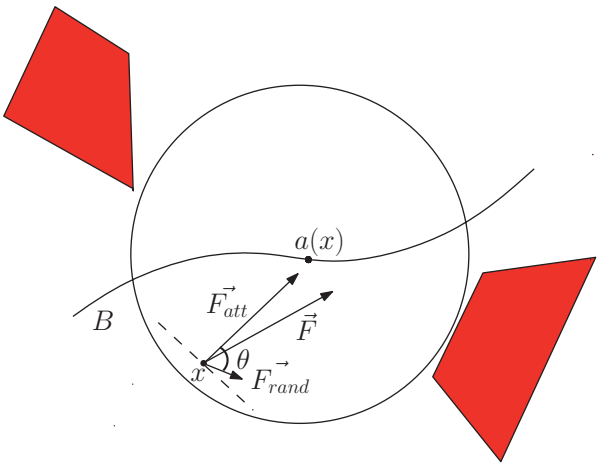


Figure 1. Adding a random force.

locally different path when the same path planning query must be solved.

Forces and Time Integration. In every step of the numerical integration two forces are used to generate the character's motion inside the corridor (Figure 1).

First, the attraction force steers the character toward a point further on the backbone path B. The character's current position is denoted by x and the attraction point by $\alpha(x)$. Let d be the Euclidean distance between the character and the attraction point, r the radius of the character, and $R[t]$ the clearance of the attraction point, where $t : t \in [0, 1]$ is the time index. Then, the force is defined in the CMM as follows:

$$\vec{F}_{\text{att}} = f\frac{\vec{\alpha}(x) - \vec{x}}{\|\vec{\alpha}(x) - \vec{x}\|}, \text{ where } f = \frac{1}{R[t] - r - d} - \frac{1}{R[t] - r}$$

Second, a random force is added to slightly change the direction of the character's movement. Let θ denote the direction returned by our Perlin Noise implementation. This direction is expressed as an angle of rotation about the current attractive vector. Thus, the random force can be defined as:

$$\vec{F}_{\text{rand}} = c_{\text{rand}}\mathcal{R}(\theta)\frac{\vec{\alpha}(x) - \vec{x}}{\|\vec{\alpha}(x) - \vec{x}\|}$$

where c_{rand} is a small constant specifying the relative strength of the force, $\mathcal{R}(\theta)$ represents the 2D rotation matrix and the angle $\theta \in [-\pi/2, +\pi/2]$. The later ensures that the character will slightly deviate from its main route, as the random force will be directed either to the left or to the right of the attraction point.

The final force $\vec{F}$ is calculated by adding the random force to the current attractive force. The resulting force vector is used to compute the new velocity vector and update the position of the character. The attractive force keeps the character inside the corridor, whereas the random force changes the steering direction.

Frequency Dependence. As mentioned above, the frequency of the noise function affects the quality of the computed random paths. In Figure 2, a typical example of varying the frequency is depicted. The first image shows the correlation between the direction of the random force and the motion time, given a low frequency. An aesthetically pleasant path is retrieved as displayed in the third image. The second image demonstrates an example of a high-frequency usage that leads to an unrealistic path, displayed in the fourth image. The simulated movement looks like a drunkard's walk and certainly is not a path that someone will take in the everyday life.

Results

We have implemented the presented approach to test its efficiency and check whether it can produce smooth paths that are similar yet different to the ones returned by the CMM. All the experiments were performed on a Pentium IV 2.4 GHz computer with 1GB memory.

The experiments were conducted for the environment depicted in Figure 3. This is a model of the McKenna MOUT (military operations in urban terrain) training center, hosted at Fort Benning, Georgia, USA. The center features an urban European village and is used by the United States Armed Forces for military training.

There are many scale differences in the test environment i.e., it contains many large open spaces and many narrow passages. This leads to the extraction of different types of corridors, which make this environment ideal for testing our path planning approaches.

To ensure low query times and high-quality paths the input roadmap was generated using the enhanced Reachability Roadmap Method,[15] which is based on the Probabilistic Roadmap Method (PRM) introduced by Svestka *et al.*[16] The resulting roadmap is shown in Figure 3(b).

Resulting Paths. In our implementation the character retains its steering direction and in every time step performs smooth random displacements. Thus, a slightly different path is expected to be computed in response to

DOI: 10.1002/cav

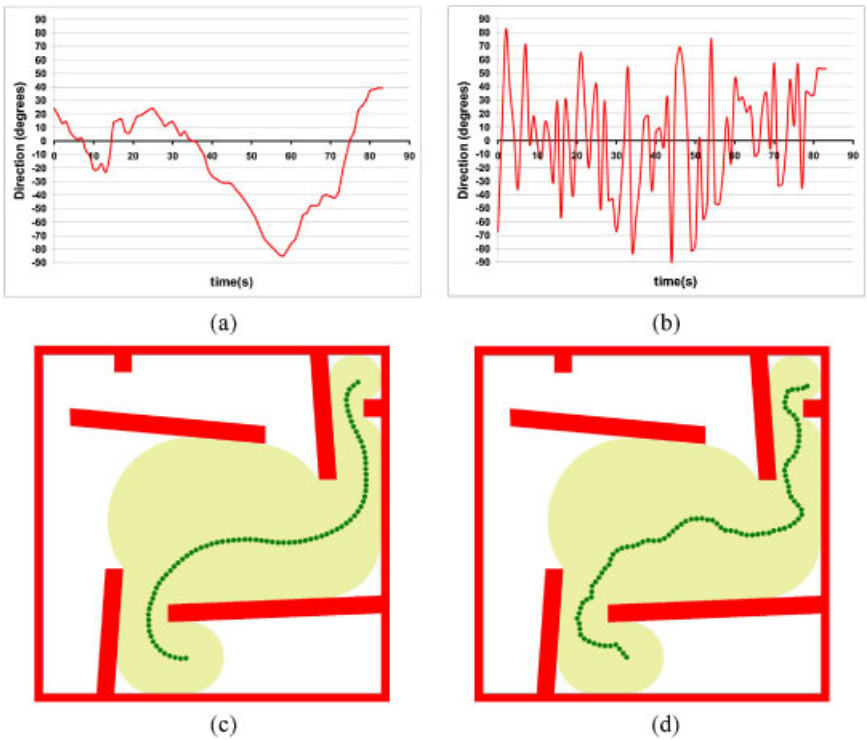


Figure 2. The frequency of the Perlin noise function affects the quality of the generated paths. (a), (b) Direction of the random force per motion time for a low and a high frequency, respectively. (c), (d) The corresponding low-frequency and high-frequency paths.

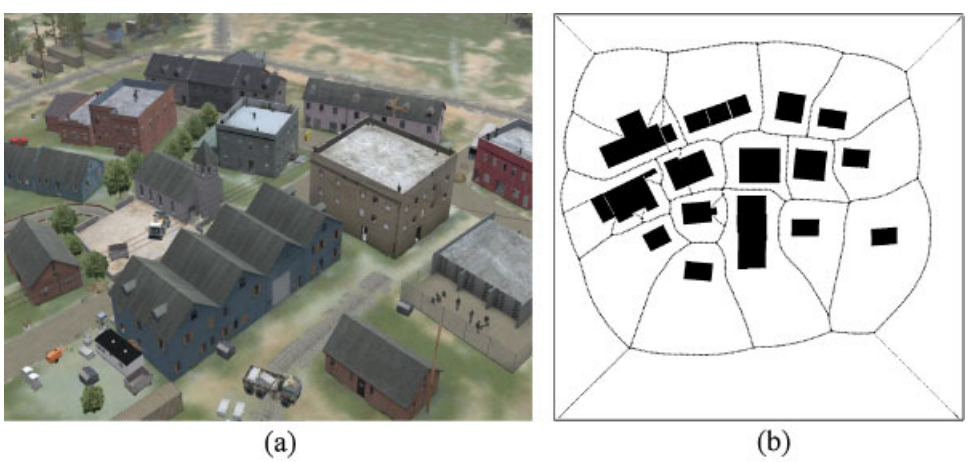


Figure 3. The McKenna MOUT training site at Fort Benning, Georgia, USA. (a) 3D Representation of the site (the image was provided by Reference [22]), (b) The input roadmap used for the experiments. Small buildings and obstacles were not included in the footprint of the test environment.

a given query. Figure 4(b) shows 100 paths generated by our technique given a start and goal position, whereas the fixed path created by the CMM is displayed in Figure 4(a). We compared the length as well as the (average, maximum, and minimum) clearance of the paths produced by the two methods.

Table 1 shows the corresponding statistics. As can be seen, the random paths keep a safe amount of clearance

DOI: 10.1002/cav

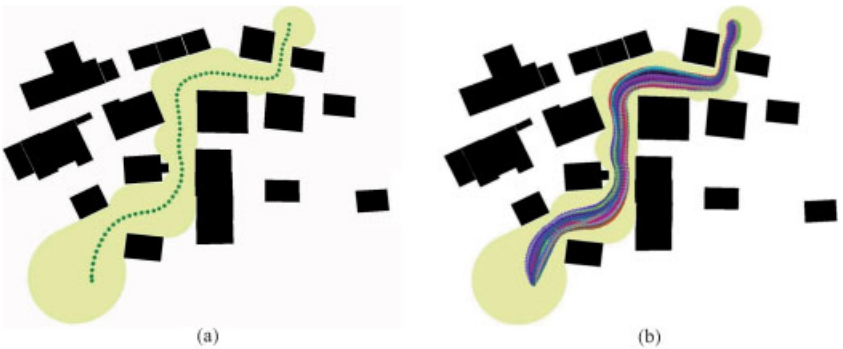


Figure 4. Using Perlin noise to create slightly different paths. (a) Fixed path returned by the CMM, (b) 100 random paths generated using the noise function.

	Fixed path	Random path
Min clearance	3.25	2.87
Avg clearance	6.21	5.94
Max clearance	10.81	10.01
Path length	144.08	145.22

Table 1. Comparison between fixed paths and random paths

from the edges of the corridor and have almost the same average clearance as the fixed path. In addition, no significant difference in the length of the paths is observed. As other queries also led to similar results, we can conclude that our proposed method can successfully generate smooth paths that are very similar but still different from the ones returned by the CMM.

Performance. Apart from the quality of the generated paths, we are also interested in the performance of our proposed approach. The paths must be computed in real-time so that the method can be applicable to interactive applications like computer games.

In Reference [3], Geraerts and Overmars have already shown that the CMM produces high-quality paths in real-time. As in our implementation we only extend the force function of the original CMM, we expect that the running times would not be increased much.

To test the performance, 100 paths from random start and goal positions were generated. We measured the average time required to retrieve a path, as well as the average time that the character needs to traverse it. Consistent with the results presented in Reference [3], it took less than a half millisecond CPU time to calculate a second of movement for a character, which implies a CPU load less than 0.05% per character. This clearly demonstrates that our Perlin noise implementation does not add any significant overhead to the CPU usage and it is very efficient in producing random smooth paths in real-time.

Alternative Paths Using Fixed Bias

As individuals can have a preference for certain paths over others, a second approach is proposed to generate different preferred paths that a character can follow inside the corridor. Similar to the previous approach, the force function is extended to bias the character's motion. Two different implementations are presented and evaluated.

Lane Formation

Our goal is to generate smooth paths that lie either to the left or to the right of the backbone path. This leads to the formation of different lanes inside the corridor that the character can follow in order to reach its goal position.

One could achieve this by creating a sub-corridor inside the original corridor. However, a much simpler approach is to update the force that guides the character through the corridor. An additional advantage of our method is that the characters would still be able to use the other side of the corridor, if constraints (like avoiding other entities) would enforce this.

Forces and Time Integration. In every time step, a force perpendicular to the main attractive force is exerted

DOI: 10.1002/cav

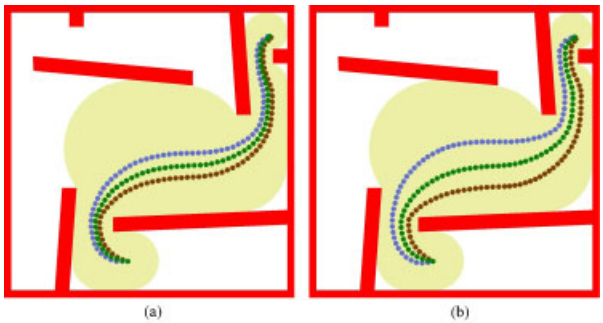


Figure 5. Lane formation inside the corridor for different values of k. (a) k = 0.25, (b) k = 0.5.

in order to change the steering direction of the character. Let d be the Euclidean distance between the character's position x and the attraction point $\alpha(x)$, $R[t]$ be the radius of the circle that corresponds to $\alpha(x)$ and r be the radius of the character. Let also θ denote the angle of rotation about the current attractive force, $\mathcal{R}(\theta)$ represent the 2D rotation matrix and $k : 0 \leq k < 1$ be a constant. Then the force can be defined as

$$\vec{F}_{\text{perp}} = c_{\text{perp}}\mathcal{R}(\theta)\frac{\vec{\alpha}(x) - \vec{x}}{\|\vec{\alpha}(x) - \vec{x}\|}, \text{ where } c_{\text{perp}} = k\frac{R[t] - 2r}{d}$$

In contrast to our previous approach, in the current implementation the angle θ can only take two values: $\theta \in \{-\pi/2, +\pi/2\}$. A positive angle of rotation steers the character to the left of the attraction point, whereas a negative θ to the right. Therefore, by keeping the angle fixed, a new (alternative) path will be generated either to the left or to the right of the backbone path.

The scalar c_{perp} controls the relative strength of the force. The closer the character is to the attraction point (i.e., the smaller the distance d), the stronger the force will be. In addition, the constant k is used to specify how close to the boundary of the corridor the character moves.

It must be pointed out that since the attraction point is selected as the furthest advanced point on the backbone path, it always lies in front of the character, and therefore, $d > 0$. The distance becomes zero when the character reaches its goal position.

We refer the reader to Figure 5 for a graphical representation of the described technique. Different right/left paths are returned when varying the value of the constant k.

Our Perlin noise implementation can also be used to create slightly different lanes in response to a given query. Depending on whether we want to follow a right or a left path, the angle θ will lie within the range of $[-\pi/2, 0]$ or $[\pi/2, 0]$, respectively. The noise function ensures that the character will stay on one side of the corridor, performing at the same time smooth random movements.

Path Following

In this second method our goal is to generate paths that either take tight or wide corners. To achieve this in every time step we dynamically compute the direction to which the backbone path is headed. The estimated direction is then used to bias the character's motion. As a result, if the path turns to the right or left, rather than waiting until the very last moment, the character will start the turn in advance. This leads to a more natural and visually interesting movement. The character looks intelligent, in the sense that it anticipates the direction of the path it has to follow and takes a more direct path, minimizing the time needed to reach its goal position.

Direction Estimation and Time Integration. Let $\alpha(x)$ be the attraction point that corresponds to point $B[t]$ on the backbone path, where $t : t \in [0, 1]$ is the time index. Then, in every time step a point that lies ahead of the current attraction point is selected. The point $B^f[t]$ is defined as the maximum point on the backbone path that remains inside the circle, centered at the attraction point and having as radius its clearance $R[t]$ i.e.,

$$B^f[t] = \max\left\{\bigcup_{t' \in (t,1]} B[t'] \,\middle|\, \text{dist}\,(\text{B[t']}, \text{B[t]}) \leq \text{R[t]}\right\}$$

Having extracted the maximum valid $B^f[t]$ point, the resulting force that has to be added to the attractive force equals to

$$\vec{F}_{\text{dir}} = \mathbf{c}_{\text{dir}}\mathcal{R}(\theta)\frac{\vec{\alpha}(x) - \vec{x}}{\|\vec{\alpha}(x) - \vec{x}\|}$$

where $\mathbf{c}_{dir}$ specifies the relative strength of the force, $\mathcal{R}(\theta)$ represents the 2D rotation matrix, and θ is the angle between the current attractive vector $(\vec{\alpha}(x) - \vec{x})$ and the estimated direction of the backbone path $\left(\vec{B}^f[t] - \vec{B}[t]\right)$.

Similar to the previous approaches, the direction of the additional force (i.e., the angle θ) can be controlled by a Perlin noise function, leading to the generation of different shorter paths. The closer this direction is to the direction of the backbone path, the more tightly the character will turn the corners.

The final force $(\vec{F} = \vec{F}_{\text{att}} + \vec{F}_{\text{dir}})$ drives the character toward its goal position and at the same time ensures

DOI: 10.1002/cav

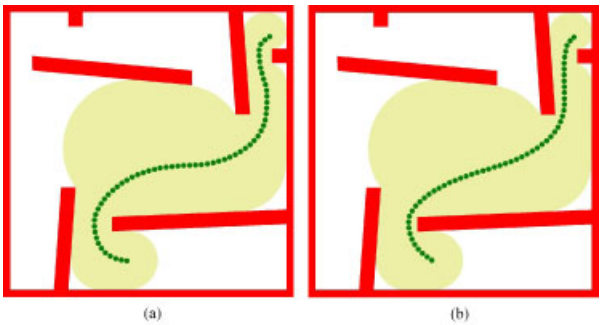


Figure 6. The "Path Following" method leads to shorter and more convincing paths. (a) Smooth path returned by the original CMM, (b) Shorter path generated by our technique.

that any unnecessary detour will be avoided. Therefore, a shorter and more believable path is returned as can be examined in Figure 6.

Results

We implemented the two techniques described in this section. Once again the McKenna environment was used to test the performance of the two methods and the quality of the generated paths.

Resulting Paths. In the first part of the experiments, we defined 100 random queries and compared the clearance and the length of the paths produced by our two proposed techniques, as well as by the original CMM. Table 2 shows the corresponding statistics for our example query depicted in Figure 4(a).

By applying a left and a right perpendicular bias ($k = 0.5$) a three-lane corridor was obtained. As both the left and right paths are closer to the boundary of the corridor, they have less (minimum, maximum, and average) clearance than the fixed path. However, as it can be inferred by the Table 2, no significant difference in the length of the paths is observed.

In the "Path Following" method the character predicts the direction of the path it has to follow and starts turning in advance. Thus, a more natural looking movement is generated. As it is expected this path is the shortest one.

The results were similar for the other queries. Therefore, we can conclude that the two techniques can successfully generate alternative routes that are aesthetically pleasant to the observer. The first method forms different lanes that a character can follow inside the extracted corridor, whereas the second method leads to a smooth and shorter path by creating shortcuts through the turns of the backbone path.

Performance. In the second part of our experiments we tested the performance of the two presented techniques. Consistent with the results of our first approach, it took less than a half millisecond per second traversed time to compute a path, either using the "right/left bias" method or the "path following" method. As a result, both implementations are efficient in planning alternative paths inside a corridor.

Variation of Speed

The approaches described above can be easily implemented to obtain variation in the paths that a character can follow. This leads to more realistic motions of characters. Another important but often overlooked factor that contributes to the realism of virtual environment applications is the speed of the animated character. A character moving at an unnatural speed would immediately break our suspension of disbelief. Therefore, in this section, we report some known facts about human speed and describe how to integrate them in our approach.

According to the U.S. Manual of Uniform Traffic Control Devices (MUTCD) the normal walking speed is 1.2 meters/second,[17] while the Manual of Uniform Traffic Control Devices for Canada (MUTCDC) indicates that the normal speed can vary from 1.1 to 1.4 meters/second.[18] These facts can be easily incorporated into the CMM in order to obtain a more believable path planning approach.

	Fixed path	Right path	Left path	Direct path
Min clearance	3.25	1.36	1.68	1.31
Avg clearance	6.22	3.94	4.48	4.89
Max clearance	10.81	8.05	8.58	8.99
Path length	144.08	145.56	147.71	131.69

Table 2. Clearance and length statistics for the fixed path and the alternative paths

DOI: 10.1002/cav

	Anger	Sad	Neutral	Joy	Content
Velocity (m/second)	1.41 ± 0.22	1.10 ± 0.21	1.19 ± 0.13	1.42 ± 0.23	1.29 ± 0.19
Normalised Velocity (1/second)	0.83 ± 0.12	0.66 ± 0.13	0.71 ± 0.08	0.84 ± 0.13	0.76 ± 0.11

Table 3. Gait velocities for the basic emotions according to Reference [21]

In every time step of the numerical integration the velocity of the character can be constrained to a maximum value that lies within the range of 1.1 and 1.4. Although, this modification will lead to a more natural motion, it is unrealistic to assume that the character will move at exactly the same speed toward its goal.

In particular, various factors can influence the walking speed such as the terrain, the weather conditions, and the number of pedestrians. Fritz and Carver mention that winter conditions can remarkably change the speed, whereas a strong headwind can more than halve the velocity of the walker.[19] The morphology, as well as the steepness of the terrain has also a significant effect on the speed. For example, a standard walking pace is unlikely to be achieved on a densely vegetated terrain, a rocky field or an ascent. Furthermore, the presence of many people in an area can also make walking difficult and lead to a decrease in the walker's speed.

An obvious way to simulate these behaviors is to add an extra force (drag force) to the final force that steers the character. The drag force is directly related to the speed of the character and is headed in the opposite direction of the character's motion i.e., $\vec{F}_{\text{drag}} = -k \times \vec{V}_{t-1}$, where $k > 0$ controls the strength of the force and $\vec{V}_{t-1}$ is the velocity of the character at the previous time step.

Emotion can also significantly affect the speed of an individual. Over the past years the relationship between walking and emotion has received a vast amount of attention. In particular, qualitative measures for emotion-related gait characteristics have been devised in Reference [20]. For instance, "heavy-footedness" can indicate an angry gait, whereas a sad gait can be distinguished by a small amount of arm swing.

Furthermore, a number of researchers from the area of psychology and biomechanics have recently managed to quantitatively describe the effects of emotion on gait.[21] Their results indicate that the walking speed of an individual slows significantly with sadness. On the other hand, for emotions with high levels of arousal (i.e., anger, joy) the velocity is faster than the normal walking speed.

These findings are summarized in Table 3. The velocities presented in the table can be used to modify the speed of a virtual character and make it consistent with its emotion and/or state. This ensures that a more natural motion will be generated for each animated character.

Conclusions

In this paper, we extended the CMM in order to generate a more realistic path planning approach. In particular, we studied how alternative paths can be created within a corridor. Two different approaches were proposed. The first uses a Perlin noise function to generate slightly different paths in response to a given query. In the second approach, a bias toward a fixed direction influences the movements of the character. The bias can be directed either to the left or to the right of the backbone path, which leads to lane formation, or it can follow the direction of the backbone path resulting in a shorter and more natural looking motion.

Experiments showed that our techniques can compute in real-time alternative paths for a character that are aesthetically pleasant to the observer. Due to the nature of the CMM (the paths are computed using social forces) our framework is also suited for planning alternative paths of a large number of moving characters. In addition, it can be easily extended to take into account static and dynamic obstacles. Repulsive forces must be added so that each character can avoid collision with the other entities and/or obstacles as described in Reference [3].

Besides planning smooth and convincing paths, virtual characters should also move in a naturally looking manner. A natural motion enhances the realism of the virtual environment and results in an interesting game play and/or better training transfer. Therefore, in the last section of this paper, we discussed how variation in the speed of the animated characters can be incorporated into the CMM, leading to characters that exhibit human like behavior.

Inspired by our current research, we believe that our extended CMM framework has a lot of potential and can form the basis for further research. We are

DOI: 10.1002/cav

currently investigating ways to integrate it with existing agent-based models, aiming to obtain realistic crowd motions. Preliminary results look very promising. We are also planning to exploit motion capture data in order to validate our presented models, as well as propose new ones. Last but not least, we are researching quantitative quality measures that will allow the users to evaluate how well the proposed solutions can capture real-life behavior and influence their sense of immersion.

ACKNOWLEDGEMENTS

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. Poole S. *Trigger Happy: Videogames and the Entertainment Revolution*. Arcade Publishing: New York, 2000.
2. Kamphuis A, Overmars MH. Finding paths for coherent groups using clearance. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. 2004; 19–28.
3. Geraerts R, Overmars MH. The corridor map method: a general framework for real-time high-quality path planning. *Computer Animation and Virtual Worlds* 2007; **18**(2): 107–119.
4. Russell S, Norvig P. *Artificial Intelligence: A Modern Approach*. Prentice Hall: Englewood Cliffs, New Jersey, 1995.
5. DeLoura M. *Game Programming Gems 1*. Charles River Media, Inc.: Rockland, MA, USA, 2000.
6. Reynolds CW. Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics* 1987; **21**(4): 24–34 (SIGGRAPH'87 Conference Proceedings).
7. Reynolds CW. Steering behaviors for autonomous characters. In *the proceedings of Game Developers Conference*, Miller Freeman Game Group:San Jose, California, 1999; 763–782.
8. Helbing D. A mathematical model for the behavior of individuals in a social field. *Journal of Mathematical Sociology* 1994; **19**(3): 189–219.
9. Helbing D, Molnar P. Social force model for pedestrian dynamics. *Physical Review E* 1995; **51**: 4282–4286.
10. Reif J, Wang H. Social potential fields: a distributed behavioral control for autonomous robots. In *International Workshop on Algorithmic Foundations of Robotics (WAFR)*, Goldberg K, Halperin D, Latombe J-C, Wilson R (eds). A. K. Peters: Wellesley, MA, USA, 1995; 431–459.
11. LaValle SM, Kuffner JJ. Randomized kinodynamic planning. In *Proceedings IEEE International Conference on Robotics and Automation*, Detroit, MI, May 1999; 473–479.
12. Kuffner JJ, LaValle SM. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000; 995–1001.
13. LaValle SM, Kuffner JJ. Rapidly-exploring random trees: progress and prospects. In *Proceedings Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, 2000.
14. Perlin K. An image synthesizer. *SIGGRAPH Computer Graphics* 1985; **19**(3): 287–296.
15. Geraerts R, Overmars MH. Creating high-quality roadmaps for motion planning in virtual environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006; 4355–4361.
16. Kavraki LE, Švestka P, Latombe J-C, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 1996; **12**(4): 566–580.
17. Federal Highway Administration, U.S. Department of Transportation. *U.S. FHWA Manual on Uniform Traffic Control Devices*, 2003 with revisions 1 and 2 incorporated edition, AASHTO: Washington, DC, 2003.
18. The Transportation Association of Canada. *Manual of Uniform Traffic Control Devices for Canada*, 2002 updated edition, Transportation Association of Canada: Ottawa, 2002.
19. Fritz S, Carver S. Accessibility as an important wilderness indicator: Modelling naismith's rule. In *GISRUK'98*, Edinburgh, Scotland, April 1998.
20. Montepare JM, Goldstein SB, Clausen A. The identification of emotions from gait information. *Journal of Nonverbal Behavior* 1987; **11**(1): 33–42.
21. Cluss MB, Crane EA, Gross MM, Fredrickson BL. Effect of emotion on the kinematics of gait. In *American Society of Biomechanics*, Blacksburg, VA, September 2006.
22. MetaVR Inc. Metavr real-time pc-based 3d visual simulation. http://www.metavr.com (Accessed 23 June 2008).

Authors' biographies:

Ioannis Karamouzas is a PhD student in the Department of Computer Science at Utrecht University, under the supervision of Professor Mark H. Overmars. He received his MSc degree in Advanced Computer Science in 2005 from the University of Manchester and his BS degree in Applied Informatics in 2004 from the University of Macedonia, Greece. His research interests include path planning, crowd simulation, and computer games. His doctoral thesis involves natural path planning for virtual entities.

DOI: 10.1002/cav

Mark H. Overmars received his PhD degree in Computer Science in 1983 from Utrecht University in the Netherlands. Currently, he is a full professor at the Department of Computer Science at the same university. Here, he is scientific director of the project Game Research for Training and Entertainment (GATE; http://gate.gameresearch.nl).

His main research interests include path planning, crowds modeling, animation, virtual environments, and game design. Over the past years he published over 250 papers in refereed journals and conferences, and he is author of one of the prime textbooks on computational geometry.

Comp. Anim. Virtual Worlds 2008; **19**: 283–293
DOI: 10.1002/cav