COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2013; **24**:285–295
Published online 14 May 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1511

SPECIAL ISSUE PAPER

Real-time path planning in heterogeneous environments

Norman Jaklin[1]*, Atlas Cook IV[2] and Roland Geraerts[1]

[1] Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
[2] Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, TX, USA

ABSTRACT

Modern virtual environments can contain a variety of characters and traversable regions. Each character may have different preferences for the traversable region types. Pedestrians may prefer to walk on sidewalks, but they may occasionally need to traverse roads and dirt paths. By contrast, wild animals might try to stay in forest areas, but they are able to leave their protective environment when necessary. This paper presents a novel path planning method named *Modified Indicative Routes and Navigation* (MIRAN) that takes a character's region preferences into account. Given an indicative route as a rough estimation of a character's preferred route, MIRAN efficiently computes a visually convincing path that is smooth, keeps clearance from obstacles, avoids unnecessary detours, and allows local changes to avoid other characters. To the best of our knowledge, MIRAN is the first path planning method that supports the aforementioned features while using an exact representation of the navigable space. Experiments show that with our approach, a wide range of different character behaviors can be simulated. It also overcomes problems that occur in previous path planning methods such as the *Indicative Route Method*. The resulting paths are well suited for real-time simulations and gaming applications. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS
path planning; autonomous agents; heterogeneous virtual environments

***Correspondence**
Norman Jaklin, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands.
E-mail: N.S.Jaklin@uu.nl

1. INTRODUCTION

The goal of this paper is to show how virtual characters can be steered through a *heterogeneous environment* in real time. Such an environment contains impassable obstacles plus many types of traversable regions. Our method uses customizable weights for arbitrary terrain polygons, thus making it applicable for a great variety of settings. City environments with traversable regions such as sidewalks, roads, and lawns can be modeled in the same way as weather-influenced environments such as swamps, dirt paths, deserts, and frozen lakes. Furthermore, psychological influences such as unattractive environments (narrow passages or trashy areas) or attractive areas (artsy neighborhoods or areas with a panoramic view; see Figure 1) can be modeled as well.

The ability to automatically guide virtual characters along their preferred traversable regions enhances realism in simulation applications. It also improves player immersion in digital games.

1.1. Related Work

The general path planning problem in *homogeneous* virtual environments has been widely studied in the past. Such environments have traversable regions and nontraversable obstacles. This research has led to advanced data structures and algorithms. We refer the reader to the books of Latombe [1], LaValle [2], and Choset *et al*. [3] for an overview.

Other related works have considered autonomous agent navigation in virtual environments. Shao and Terzopoulos [4] present an artificial life approach that integrates and combines motor, perceptual, behavioral, and cognitive components to emulate the complexity of real pedestrians in urban environments. Only recently, Lo *et al*. [5] showed how agents can learn from raw vision input to navigate autonomously. They introduce a hierarchical state model and a novel regression algorithm to avoid the "curse of dimensionality". Kang *et al*. [6] use a different approach to find paths for agents in virtual environments. Their

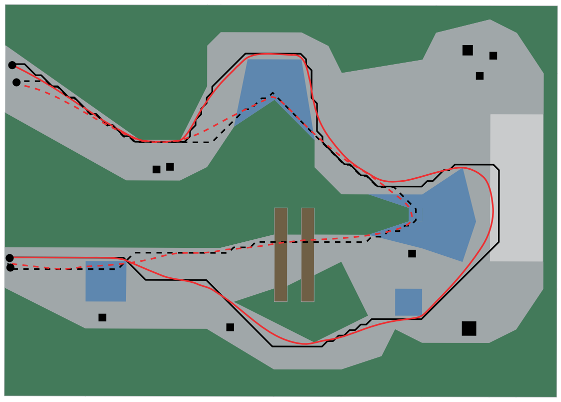

Figure 1. A path (gray) in a forest (green) with obstacle trees (black), puddles (blue), fallen trees (brown), and a spot with a panoramic view (light gray). Two characters (adult and child) follow automatically computed indicative routes (solid and dashed black). The smoothed paths (solid red for the adult, dashed red for the child) are computed with our MIRAN method. Both the indicative routes and the paths are based on the characters' terrain preferences. The adult avoids puddles and fallen trees and is attracted to the spot with the panoramic view. The child prefers to walk through puddles, climbs over the trees and is not interested in the panoramic view.

adaptive agent navigation approach collects data and learns new paths from user-controlled characters.

The problem we tackle can be seen as a variant of the *weighted region problem* (WRP) that was introduced by Mitchell and Papadimitriou [7]. They assign a weight to each type of region. The *weighted length* of a path inside any single region is the arc length of that path times the weight of the containing region. The weighted length of a path that passes through multiple regions is the sum of the weighted path lengths inside each region. An optimal path between two points is a path with the minimum possible weighted length. Solving the WRP means finding such an optimal path. The algorithm of Mitchell and Papadimitriou has $O(n^8)$ time complexity with n being the total number of vertices in a heterogeneous environment. Because this running time is too large for most practical applications, research has been carried out on finding feasible approximation algorithms [8–13]. Only recently, De Carufel *et al.* [14] showed that the WRP is unsolvable in the *Algebraic Computation Model over the Rational Numbers*.

Because of the computational complexity of the WRP, many experimental works show how to compute reasonably short paths through an environment by combining a grid or quadtree with an A* or D* search [15,16]. Harabor and Botea [17] use a grid to guide a set of uniquely sized characters through a heterogeneous environment. Each character occupies a $c \times c$ square in the grid. The value of c can be any nonnegative integer, and c can be unique for each character. They keep track of the nearest obstacle to each grid cell to guarantee that paths have good clearance with respect to the obstacles. Multiple terrain types are handled by associating each character with a set of terrain types that this character can traverse. These kinds of grid-based approaches often approximate an environment and can sometimes produce nonsmooth paths that are located very close to obstacles. This behavior can make it difficult to avoid other moving characters. Furthermore, the approaches are expensive and can yield long computation times when dealing with a large number of characters.

Geraerts [18] presented an augmented medial axis to create a navigation mesh called the *Explicit Corridor Map* (ECM). The ECM can compute paths with any desired amount of clearance to obstacles and permits each character to have any desired size. Although this navigation mesh is compact and exact, each region in the environment is strictly traversable or not traversable. Thus, the ECM does not consider heterogeneous environments. By contrast, our method can use the ECM and its concept of *corridors* as an underlying data structure to handle local collision avoidance.

Karamouzas *et al.* [19] introduced the *Indicative Route Method* (IRM) as a force-based approach to steer characters through a homogeneous environment. An *indicative route* is a rough estimation of the preferred path from a character's start position to a goal position. An indicative route can be manually designed or automatically computed by a higher-level path planning approach such as an A* search on a grid [20]. However, an indicative route is usually not natural enough for a character to strictly follow. Local collision avoidance is also difficult if characters strictly follow their indicative route. Instead, an indicative route can be used as a guideline to produce smooth trajectories for a character. In the IRM, an *attraction point* on the indicative route is computed to make the character follow the route while avoiding obstacles and other moving characters. Using the ECM as an underlying navigation mesh, the paths can have any desired amount of clearance from obstacles.

1.2. Contribution

The main focus of our work lies in computing visually convincing and terrain-dependent paths for characters in virtual environments. We solve a generalized variant of the WRP [7]. In this variant, we do not look for a shortest path to a goal destination while taking weighted regions into account. Instead, we compute natural-looking paths that follow an arbitrary indicative route. Such an indicative route functions as a rough estimation of the characters' preferred path. We adopt the concept from the IRM by Karamouzas *et al.* [19] of using attraction points on the indicative route to let the character follow the route until the goal position is reached. Our method supports heterogeneous environments because the character follows the indicative route based on its individual terrain preferences. To the best of our knowledge, it is the first path planning method that combines the aforementioned features with the possibility to use it on an exact representation of the environment.

Such an exact representation is provided by the ECM that was introduced by Geraerts [21]. *Modified Indicative Routes and Navigation* (MIRAN) is general enough to be used with any other type of underlying representation of the navigable space. However, using it on an exact representation such as the ECM overcomes the problems approximated representations (such as grids) suffer from. For example, narrow passages between obstacles will never be blocked, which can be the case when using a grid with large cell sizes.

MIRAN also overcomes the following problem: In the IRM, the character may skip arbitrarily large parts of the indicative route. See Figure 8 for an example. This is because the attraction point is defined as the intersection of the indicative route with a maximum clearance disk centered on the medial axis of the navigable space. Whenever there is a large area of free space around the indicative route, the radius of the maximum clearance disk is also large. This permits the attraction point to be farther away along the indicative route. If the indicative route has many turns inside the clearance disk, those turns will be skipped by the character. Our method presented in Section 3 overcomes this issue because the computation of attraction points is independent of the amount of free space around the character. It is therefore applicable to both virtual indoor and outdoor environments.

1.3. Overview

This paper is organized as follows. Section 2 introduces the terminology and a navigation mesh that is well suited for heterogeneous environments. In Section 3, we present our new MIRAN method for using an indicative route to compute the final path trajectories. We provide the theoretical background and show that a character will always reach the goal position. In Section 4, we briefly discuss how indicative routes can be automatically computed with respect to a character's traversable region preferences. We conduct experiments in Section 5 and conclude in Section 6 that MIRAN can be used to compute visually convincing paths in heterogeneous environments in real time.

2. PRELIMINARIES

We say that a two-dimensional polygonal environment is *heterogeneous* when it contains more than one type of *traversable region*. A traversable region is a two-dimensional polygon that is annotated with a *type*. Examples of types are roads, sidewalks, carpeted floors, tile floors, grasslands, snowlands, deserts, and mud pits. A region type can also represent a psychological aspect such as a dangerous area or a pleasant spot with a panoramic view. Slope information and crowd density information [22] could be used to weight the attractiveness of a traversable region as well. The union of these regions make up the free navigable space of the environment.

Our method generally works with any popular representation of the navigable space such as grids, waypoint graphs, or navigation meshes. We choose the ECM data structure of Geraerts [21] as a navigation mesh because it yields a storage space-efficient representation of the exact geometry of the environment. We construct the ECM based on the obstacle polygons and consider the union of all traversable terrain polygons to be free space. The ECM works both on 2D and multilayered 3D environments [23].

For our method, we assume that a *character* is represented by a single point in the environment. Each character has a unique set of region preferences that are given as positive numerical values. For example, consider a family that strolls through a park. A child might walk through mud and puddles, whereas the adults will avoid those spots (Figure 1). Furthermore, a person who is very late might be willing to run through muddy terrain to save time, whereas a person in a nice suit is willing to take large detours. Those kind of higher-level considerations can be modeled with MIRAN in the same way as geometrical terrain information by setting the character's preferences accordingly.

A character is steered through a heterogeneous environment by using both an indicative route and local steering methods. An *indicative route* is a curve $\pi_{ind} : [0,1] \to \mathbb{R}^2$ through the environment. This curve passes through a sequence of traversable regions that a character prefers to traverse. The character can locally diverge from the route to walk a smooth path or to avoid collisions with other characters.

3. THE MIRAN METHOD

In this section, we show how MIRAN works in detail. Given an indicative route π_{ind}, we assume that the initial character's position x_0 equals the starting point s of π_{ind}. Otherwise, we first compute a connection from x_0 to s to bridge the gap.

Algorithm 1 THE MIRAN METHOD

Input. Start s, goal g, indicative route from s to g
Output. Smooth terrain-dependent path from s to g

```
1: $i \leftarrow 0$
2: $x_0 \leftarrow s$
3: while $x_i \neq g$ do
4:     $r_i \leftarrow$ COMPUTEREFERENCEPOINT($x_i$)
5:     $\mathcal{A}_i \leftarrow$ COMPUTECANDIDATEATTRACTIONPOINTS($r_i, x_i$)
6:     $\alpha_i \leftarrow$ PICKBESTCANDIDATE($\mathcal{A}_i, x_i$)
7:     $x_{i+1} \leftarrow$ MOVECHARACTERTOWARDSATTRACTIONPOINT($x_i, \alpha_i$)
8:     $i \leftarrow i + 1$
9: end while
```

Because of smoothing and taking shortcuts along the way, the character's position will usually not be exactly on the indicative route. Furthermore, the character can be pushed away from π_{ind} by other forces (such as other moving characters). In each step i of the method, we compute a *reference point* r_i on π_{ind} to indicate how far along the route the character has come.

We use two user-controlled parameters, the *shortcut parameter* σ and the *sampling distance* d, which together with the reference point determine the set $\mathcal{A}_i$ of candidate attraction points. Our method ensures that each candidate attraction point in $\mathcal{A}_i$ is visible from the character's current position. Once $\mathcal{A}_i$ is computed, we pick the best attraction point with respect to a cost function. This cost function assigns a cost to each straight-line segment between the character's current position and a candidate attraction point. The costs depend on the Euclidean length of the line segment, the types of terrain it intersects, and the position of the candidate point on π_{ind}. The farther ahead the candidate point is on π_{ind}, the lower the cost. In this way, we reward picking the farthest candidate attraction point whenever there are multiple choices. We then use the same force-based steering approach that is used in the IRM [19] to move the character.

The pseudo-code on page 3 shows how the overall method works. In Section 3.1, we describe how to compute the reference point in each step of the simulation. In Section 3.2, we give details on how the set $\mathcal{A}_i$ of candidate attraction points is computed, and in Section 3.3, we explain how to choose an attraction point from $\mathcal{A}_i$ using our cost function. In Section 3.4, we discuss the force-based approach to steer the character through the heterogeneous environment. We prove the correctness of our method in Section 3.5.

3.1. Computing a Reference Point

We will now discuss how to compute a reference point r_i in each step i of the method. Let x_i be the character's current position. We define r_i as the *first closest point* from x_i to the part of π_{ind} that lies between the previous reference point r_{i-1} and the previous attraction point α_{i-1} for $i \geq 1$ (Figure 2). For the initial step $i = 0$, we have $r_0 = x_0$ because we assumed x_0 to be the starting point s of π_{ind}. Whenever the character is standing on the indicative route, the reference point r_i equals the current position x_i. We restrict the reference point to the given subpath of π_{ind} because otherwise we might refer to a point that lies too far ahead along the route, leading to undesired shortcuts. In the example shown in Figure 2, picking the closest point c as the next reference point r_i would lead to skipping the whole part of the indicative route between α_{i-1} and c, which can be arbitrarily large in general.

With the aforementioned definition of the reference point, we are now able to compute the set $\mathcal{A}_i$ of candidate attraction points. We will show next how this is performed in detail.

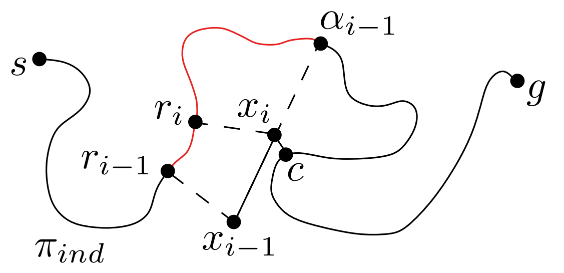

Figure 2. Only the subpath of π_{ind} between r_{i-1} and α_{i-1} (shown in red) is taken into consideration for the computation of reference point r_i. Choosing the closest point c as the reference point would lead to an undesired shortcut.

3.2. Computing the Candidate Attraction Points

As sketched previously, we introduce two parameters that can be adjusted by the user to compute the set $\mathcal{A}_i$ of candidate attraction points:

- The *shortcut parameter* σ defines the maximum allowed curve length distance from the reference point to the farthest candidate attraction point.
- The *sampling distance* d defines the maximum curve length distance between each candidate attraction point.

The shortcut parameter σ is used to control the degree of smoothing we want to allow and to prevent the character from taking undesired shortcuts. It defines the maximum curve length distance the character is allowed to skip when following the route. Because all candidate attraction points in $\mathcal{A}_i$ are not farther away from r_i than σ (with respect to the curve length of π_{ind}), we ensure that we can pick any of them without generating undesired shortcuts. By σ_i, we denote the point on π_{ind} that has curve length distance σ from r_i. Let $\pi_{ind}(r_i, \sigma_i)$ be the subpath of π_{ind} from r_i to σ_i. Our candidate attraction points are always on $\pi_{ind}(r_i, \sigma_i)$.

Now, the first step in the computation of $\mathcal{A}_i$ is to determine all parts of $\pi_{ind}(r_i, \sigma_i)$ that are *visible* from the current position x_i with respect to all static obstacle polygons. Formally, we compute the *visibility polygon* $\mathcal{P}_i$ for x_i and the union of all static obstacle polygons, and let the intersection $\mathcal{V}_i := \mathcal{P}_i \bigcap \pi_{ind}(r_i, \sigma_i)$ be the set of all points on $\pi_{ind}(r_i, \sigma_i)$ that are currently visible (Figure 3). The set $\mathcal{V}_i$ yields a division of the indicative route into a set of real intervals $V_j = [a_j, b_j] \subset \mathbb{R}$, such that $\pi_{ind}(t)$ is visible from x_i for all $t \in V_j$. We let each visible end point $\pi_{ind}(a_j)$ and $\pi_{ind}(b_j)$ of each interval be a candidate attraction point. The only exception is that we do not want the reference point r_i to become a candidate point whenever $r_i = x_i$. Because r_i equals $\pi_{ind}(a_1)$ whenever the reference point is visible, the character would be attracted to its current position. We ignore a_1 in this case and start assigning the candidate attraction

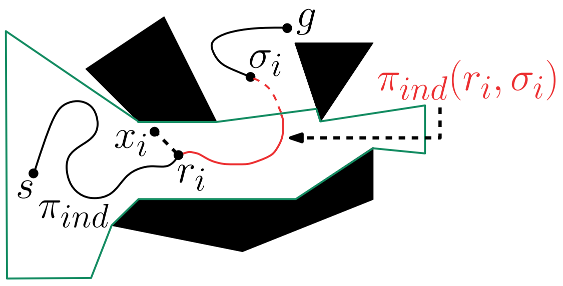

Figure 3. Visible parts of the environment and the indicative route from the current position x_i.

points with the point $\pi_{ind}(b_1)$. Note, however, that we do not exclude r_i from the set of candidate attraction points if $x_i \neq r_i$ because r_i might be the only visible and valid candidate point.

We continue to add more values to our set A_i by sampling each visible interval of the indicative route using the sampling distance d. The closer to 0 the sampling distance, the more candidate attraction points we generate, and the more is our set $\mathcal{A}_i$ an approximation of a continuous set. A smaller sampling distance therefore generates higher accuracy while increasing computation time. If d is set too large, the resulting inaccuracy may lead to undesired output paths. In practice, however, feasible values of d can be easily set if the size of the environment and the curve length of the indicative route are known (see Section 5 for examples). Once set to a feasible value, smaller values affect the overall output paths only insignificantly. In areas near static obstacles with no change of the underlying terrain, smaller values of d do not change the output paths at all because the last visible point along the route will always be picked as an attraction point (see Section 3.3 for details).

Sampling each visible interval is performed as follows. If for any real interval V_j, the curve length distance between $\pi_{ind}(a_j)$ and $\pi_{ind}(b_j)$ is greater than the sampling distance d, we add a candidate attraction point between those two points with curve length distance d from $\pi_{ind}(a_j)$. We iterate this process until the maximum distance between any two subsequent candidate attraction points is d. Note that the curve length distance between b_j and the previous sampled candidate attraction point can be smaller than d. We then let $\mathcal{A}_i = \{\alpha_{i_1}, \alpha_{i_2}, \ldots\}$ be the final set of candidate attraction points, ordered by their positions along the indicative route. See Figure 4 for an example of the resulting set $\mathcal{A}_i$.

3.3. Choosing the Attraction Point

Now that we have computed the set $\mathcal{A}_i$ of candidate attraction points, we pick the best candidate with respect to a weight function ω as our final attraction point α_i for the current step i of the algorithm.

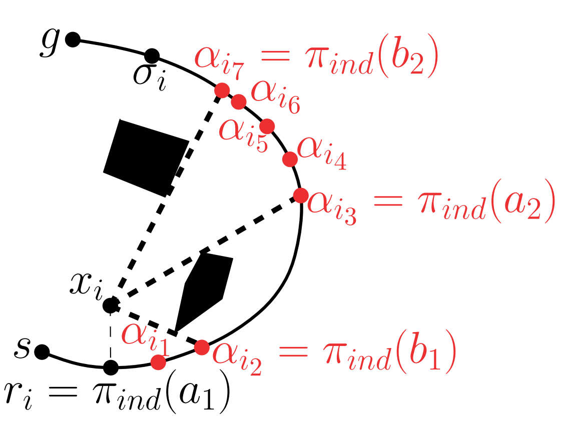

Figure 4. Example of candidate attraction points computed by our method.

Let k be the total number of candidate attraction points in the current step i. We consider the straight line segments $l(\alpha_{i_j}, x_i)$ between α_{i_j} and x_i, with $1 \leq j \leq k$. We compute a weight $\omega(l(\alpha_{i_j}, x_i))$ for each such line segment and choose α_i to be the final attraction point for which the corresponding line segment has minimum weight. The Euclidean length of the line segments, the different types of terrain the line segments intersect, as well as the corresponding character's preference values influence the weight function. Furthermore, we define the weight function such that candidate attraction points that are farther away from the current position reduce the weight of the line segments. This ensures that the character will be rewarded for picking an attraction point that is farther away.

For each line segment $l(\alpha_{i_j}, x_i)$, let $\mathcal{T}_{i_j}$ be the set of all terrain types that $l(\alpha_{i_j}, x_i)$ intersects. Let d_{i_j} be the curve length distance along the indicative route from the reference point r_i to the candidate attraction point α_{i_j}. For each terrain $T \in \mathcal{T}_{i_j}$, we let $w(T) > 0$ be the character's terrain preference. By $l_{i_j}^T$, we denote the amount of terrain type T on $l(\alpha_{i_j}, x_i)$ by summing up the length of all parts of $l(\alpha_{i_j}, x_i)$ that cross terrain type T. We define the weight $\omega(l(\alpha_{i_j}, x_i)) := \sum_{T \in \mathcal{T}_{i_j}} w(T) \cdot l_{i_j}^T / d_{i_j}$. The fraction $l_{i_j}^T / d_{i_j}$ describes the relation between the Euclidean length of the line segment with underlying terrain T and the curve length distance d_{i_j}. It ensures that we reward picking an attraction point that is farther away along the route.

After computing the weights for each one of the k line segments, we pick the final attraction point $\alpha_i := \alpha_{i,m}$ with $m = \underset{1 \leq j \leq k}{\operatorname{argmin}}\, \omega(l(\alpha_{i_j}, x_i))$. If there are several candidate points with minimum weight for the corresponding line segments, we pick the one with greatest curve length distance from the reference point along the indicative route.

3.4. Moving the Character

Once the final attraction point α_i is computed for the current step i of the algorithm, the character moves toward that point. We basically use the same local force-based steering method as described in the IRM [19]. However, other force-based approaches can be used, as well. Furthermore, we use the concept of *corridors* in the ECM structure [21] to let the character avoid static obstacles with a variable user-controlled amount of clearance.

For a fixed time step, let x be the character's current position and $||.||$ be the standard Euclidean norm. Three forces are applied to the character in each step: the *steering force* $F_s(x)$ to let the character move towards its attraction point, the *boundary force* $F_b(x)$ to push the character away from the boundary of the ECM corridor, and the *obstacle avoidance force* $F_o(x)$ to avoid small local obstacles or other moving characters.

Let α be the current attraction point and c_s be a constant specifying the relative strength of $F_s(x)$. We then define $F_s(x) := c_s \frac{\alpha - x}{||\alpha - x||}$. We use the parameters c_b and c_o to scale the effect of $F_b(x)$ and $F_o(x)$, respectively. The closer to the boundary or obstacle the character is, the stronger the repelling force. Let $b(x)$ be the closest point on the boundary of the corridor. We define $F_b(x) := c_b \frac{x - b(x)}{||x - b(x)||}$.

For n being the number of local obstacles or other characters, we let O_j, $1 \le j \le n$, be the current position of the jth obstacle. We define $F_o(x) := \sum_{j=1}^{n} c_o \frac{x - O_j}{||x - O_j||}$. We then let $F(x) = F_s(x) + F_b(x) + F_o(x)$ be the final force exerting on the character at position x.

3.5. Proof of Correctness

Now, we show that our method ensures that the character will always find a path to the goal position g—provided that the goal can be reached and the character is not pushed away by other forces (e.g., other moving characters) such that the indicative route becomes invisible. Note that the latter can always happen whenever the character is pushed behind an obstacle. In this case, our method ends, and we continue recomputing a part of the indicative route from the character's new position to its goal.

To prove the correctness, we assume that we have a finite number of polygons that are not infinitesimally small. Note that this does not imply any restrictions for most practical applications. In addition, we assume that the character is moving towards each attraction point directly. Although in practice, one can use a velocity-based integration scheme such as Euler integration to compute paths that are proven to be C^1-continuous [24], we do not assume such an approach in theory. This is because in theory, the character might be pushed behind an obstacle due to a too large step size of the integration scheme. In practice, however, this is unlikely and can be avoided by adjusting the step size accordingly.

First, we prove that there is at least one candidate attraction point we can choose from in each step of the algorithm.

Lemma 1. *Let i be the current step of the algorithm. It holds that $\mathcal{A}_i \neq \emptyset$.*

Proof. We prove this by induction on i.

For $i = 0$, we have $x_0 = r_0 = s$. The character's initial position is the starting point s of the indicative route. Because of the definition of the *visibility intervals* $V_j = [a_j, b_j]$ in Section 3.2, it immediately follows that $\pi_{ind}(a_1) = x_0$. Because the character's position x_0 cannot be the only visible point on the route up to $\pi_{ind}(b_1)$, we conclude that $a_1 \neq b_1$ and also $\pi_{ind}(a_1) \neq \pi_{ind}(b_1)$ (note that the latter does not necessarily follow from $a_1 \neq b_1$ in general, as the route can have self-intersections). By definition of the set $\mathcal{A}_i$, it follows that $\pi_{ind}(b_1) \in \mathcal{A}_0$. Let $i > 0$. By the induction assumption, we have $\mathcal{A}_{i-1} \neq \emptyset$. Therefore, an attraction point α_{i-1} has been chosen in step $i-1$ and the character moved from position x_{i-1} to position x_i, with x_{i-1}, x_i and α_{i-1} being collinear. It immediately follows that the point α_{i-1} is still visible in step i. So there must be an index j such that $\alpha_{i-1} \in V_j = [a_j, b_j]$. By definition of $\mathcal{A}_i$, the points $\pi_{ind}(a_j)$ and $\pi_{ind}(b_j)$ are both valid candidate attraction point. □

Next, we show that the sequence of reference points moves forward along the indicative route.

Lemma 2. *Let i be the current step of the algorithm. It holds that there is a future step $j > i$ in which the reference point r_j is ahead of r_i along the indicative route.*

Proof. Assume by way of contradiction that the opposite holds. Because we define the reference point to be a point between the last reference point and the last attraction point, the opposite assumption would mean that there is a reference point that does not change for all future steps. So we assume that there is a step i in which the reference point r_i equals r_j for all $j > i$. To simplify the notation, we skip the index and denote the fixed reference point as r. It immediately follows that the character does not reach the goal position g. Otherwise, there would be a step $j > i$ in which x_j is closer to g than to r, thus making g the new reference point in step $j+1$.

Because the character does not reach g but moves forward in each step because of Lemma 1, it follows that we have an infinite sequence of character positions $x_i, x_{i+1}, x_{i+2}, \ldots$. The Euclidean distance between each x_j and x_{j+1} in this sequence always equals the relative strength c_s of the attraction force $F_s(x)$ (see Figure 5 and Section 3.4). When the character moves from x_j to x_{j+1}, there is a corresponding attraction point α_j colinear with x_j and x_{j+1} that has been picked as the best point among all candidate attraction points in $\mathcal{A}_j$. This means that the weight $\omega(l(\alpha_j, x_j)) = \sum_{T \in \mathcal{T}_j} w(T) \cdot {l_j}^T / d_j$ is minimal among all candidate line segments in step j.

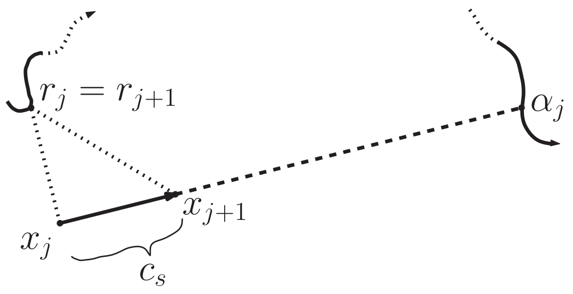

Figure 5. The distance between x_j and $\boldsymbol{\alpha}_j$ differs from the distance between x_{j+1} and $\boldsymbol{\alpha}_j$ by the amount of the strength c_s of the steering force.

Now, in the following step $j + 1$, the next candidate attraction point is computed. The former attraction point α_j has the same curve length distance $d_j = d_{j+1}$ from the reference point r along the indicative route as in step j because r stays at the same point because of our assumption. So, the weight $\omega(l(\alpha_j, x_{j+1}))$ differs from $\omega(l(\alpha_j, x_j))$ only in the Euclidean distance between the corresponding points and the terrain segments along the line. Since the Euclidean distance is smaller (it has been reduced by $c_s > 0$; see Figure 5), the weight for α_j in step $j + 1$ is smaller than the weight for α_j in step j. It follows that the attraction point picked in step $j + 1$ must have a smaller weight than the one picked in step j. Therefore, following the sequence $x_i, x_{i+1}, x_{i+2}, \ldots$ of character positions, the weight for picking the corresponding attraction points becomes smaller in each step by an absolute amount.

However, there is a lower bound for the weight. Let $d_{\min} := \min_{i \le j} ||x_j - r||$ be the minimal distance between the fixed reference point r and all character positions $x_i, x_{i+1}, x_{i+2}, \ldots$. Then, it holds that $||l(\alpha_j, x_j)|| \ge d_{\min}$. Otherwise, the corresponding attraction point α_j would be closer to x_j than r, thus becoming the new reference point in step $j + 1$. This contradicts our assumption that r stays the same point for all future steps. Furthermore, the curve length distance d_j between the reference point and the attraction point is never greater than the shortcut parameter σ, that is, $d_j \le \sigma$. If we let $w_{\min} := \min_{T \in \mathcal{T}} w(T)$ be the character's minimum preference value for all terrain types, the following lower bound for the weight $\omega(l(\alpha_j, x_j))$ applies:

$$\omega(l(\alpha_j, x_j)) = \frac{1}{d_j} \sum_{T \in \mathcal{T}_j} w(T) \cdot l_j{}^T \ge \frac{w_{\min}}{\sigma} \sum_{T \in \mathcal{T}_j} l_j{}^T$$
$$= \frac{w_{\min}}{\sigma} \cdot ||l(\alpha_j, x_j)|| \ge \frac{w_{\min}}{\sigma} \cdot d_{\min}.$$

Now, we know that the weights become smaller in each step, we have an infinite number of such steps, and there is a lower bound for the weight. It follows that the weights must asymptotically approximate the lower bound. This corresponds, however, to an infinite number of asymptotically small portions of terrain polygons that the character crosses. Because we assume that there are a finite number of polygons that are not infinitesimally small, we have a contradiction to our assumption, which proves the lemma. □

Lemmata 1 and 2 ensure that our method makes the character move forward in each step of the algorithm. Now, we prove that the character will always reach the goal position g.

Theorem 1. *There is an index $i \in \mathbb{N}$ such that $x_i = g$.*

Proof. By Lemma 2, it holds that the curve length distance along π_{ind} from the reference point to g becomes smaller over time.

Assume by way of contradiction that the character does not reach its goal, that is, for all steps i, the character's position x_i does not equal g. Because the reference point becomes closer to g over time, it follows that the sequence $(r_i)_{i \in \mathbb{N}}$ of reference points has a limit $l \in \pi_{ind}$. This limit l cannot be reached. Otherwise, if there was a step j such that $x_j = l$, by Lemma 1, there would be at least one candidate attraction point we could choose from, thus making the character go beyond l, a contradiction.

Because l is a limit point on the indicative route and the character moves forward in each step, it follows that there is a step i in the algorithm where the curve length distance along π_{ind} between r_i and l is smaller than the sampling distance d. By Lemma 1, we know that there is at least one candidate attraction point α_i to choose from. By the definition of the set $\mathcal{A}_i$ and because of the sampling distance d, this point α_i either lies beyond l or equals the reference point r_i. If the latter case holds and $r_i = \alpha_i$, the character is attracted to its reference point until it reaches that point or a different attraction point beyond l will be picked. In any case, an attraction point α_j beyond l will finally be picked. The attraction force lets the character go towards α_j and beyond l, yielding a contradiction. □

4. COMPUTING AN INDICATIVE ROUTE

We now briefly discuss how to compute an indicative route with a higher-level path planning approach. The general idea is to tessellate all navigable space and to perform an A* search [20] on the dual graph of the tessellation. Note that the resulting routes are neither smooth nor natural-looking. They are therefore not used as final paths. However, they give a rough estimation of the character's preferred route and can be used as input for our MIRAN approach.

The indicative route can be computed by using a small grid to represent the environment, or by computing a constrained triangulation of all traversable regions. We then let $G = (V, E)$ be a *weighted dual graph* of the tessellation $\mathcal{T}$. Each tessellated polygon (grid cell, triangle or

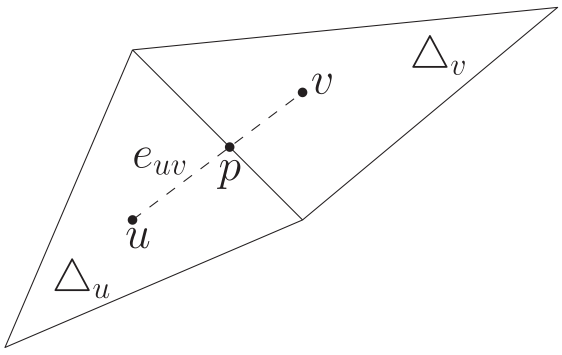

Figure 6. Example of how to weight edges in the dual graph G. Here, we show triangles $\Delta_u, \Delta_v \in \mathcal{T}$ and the straight-line edge e_{uv}.

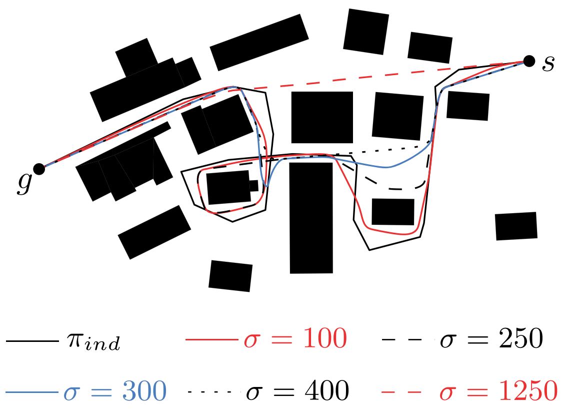

Figure 7. Output paths for different values of σ. The sampling distance d used in all examples is 20.

other) is associated with both a *vertex* $v \in V$ and with the traversable region type. A pair of nodes u, v is connected by an *edge* $e_{uv} \in E$ if the associated polygons are adjacent in the tessellation $\mathcal{T}$.

We assign a weight $w(e) > 0$ to each edge $e \in E$, based on the character's terrain preferences, as follows. Each edge $e_{uv} \in E$ is associated with the two polygons $P_u, P_v \in \mathcal{T}$ that correspond to the vertices u and v. Each one of the polygons has a terrain type T_u, T_v in the environment. There is a point p on e_{uv} where the underlying polygon in $\mathcal{T}$ changes from P_u to P_v. See Figure 6 for an example with triangles. We can split e_{uv} at p into two curve segments e_u and e_v. The curve length $\|e_{uv}\|$ of e_{uv} equals $\|e_u\| + \|e_v\|$. Let $w(T_u), w(T_v) \in \mathbb{R}$ be the character's preferences for terrain types T_u and T_v. We then define the weight $w(e_{uv}) := w(T_u) \cdot \|e_u\| + w(T_v) \cdot \|e_v\|$. We can then use an A* search in this weighted dual graph to compute the indicative route.

In the forest example in Figure 1, we use an A* search on a grid to compute the indicative routes. For the adult character, each grid cell is weighted with one of the following weights: *forest* = 30, *path* = 2, *puddles* = 10, *trees* = 30, and *panoramic view* = 1. For the child, the weights are as follows: *forest* = 30, *path* = 2, *puddles* = 1, *trees* = 1, and *panoramic view* = 10. The used MIRAN parameters to follow the routes are $\sigma = 80$ and $d = 20$ for the adult. For the child, we set $\sigma = 100$ and $d = 20$.

5. EXPERIMENTS

We performed experiments in a framework using the ECM data structure [21]. To demonstrate specific features of the method, we used indicative routes that were manually created in 2D polygonal environments containing multiple terrain types. All our experiments were performed on a PC running Windows 7, with a 3.2 GHz AMD Phenom™II X2 CPU and 4 GB memory. We used one CPU core for the computations. In this implementation, we discretized the environment. We built the ECM using graphics hardware [21] and extracted all terrain and obstacle information from the color buffer during this building process. We used Bresenham's line algorithm [25] for checking visibility for all candidate attraction points. We also used the same algorithm for computing their weights and to let the character keep clearance from obstacles. In addition, we use Euler integration in each step after computing the current attraction point to generate smooth paths. The ECM structure was also used to implement the IRM [19] and compare it with our method.

First, we tested the method with a 2D footprint of the McKenna MOUT training site at Fort Benning, Georgia, USA (Figure 7). This scene spans an area of 200 × 200 units. It contains one type of terrain and a set of 23 convex obstacle polygons. These polygons represent buildings, and they have a total of 96 vertices. Using a fixed sampling distance of 20 and different settings for the the shortcut parameter σ, we computed five output paths for the same indicative route π_{ind}. As expected, the smaller the value of σ, the closer the character follows π_{ind}. With higher values of σ, large parts of π_{ind} can be skipped. If the size of a scene and the length of π_{ind} is known, appropriate values of σ can be easily set to produce the desired behavior. This is because σ corresponds to the exact distance the character may look ahead along π_{ind}. With $\sigma = 300$, the character neither follows the loop contained in π_{ind} nor does it skip it entirely. This results in the path shown in Figure 7 (light blue). One might argue that this is an unnecessary detour. However, the MIRAN approach does not tackle the problem of how to reach a goal destination along a shortest path. Instead, it generalizes this idea and shows how to follow an arbitrary route while taking terrain information into account. Furthermore, the shown behavior can be easily changed if needed by adjusting the value of σ accordingly.

We compared MIRAN with the IRM in the McKenna scene with an additional road, sidewalk, and some muddy terrain next to the sidewalk (Figure 8). This scene spans the same area and contains the same obstacle polygons as the original scene, but it has 10 additional convex terrain

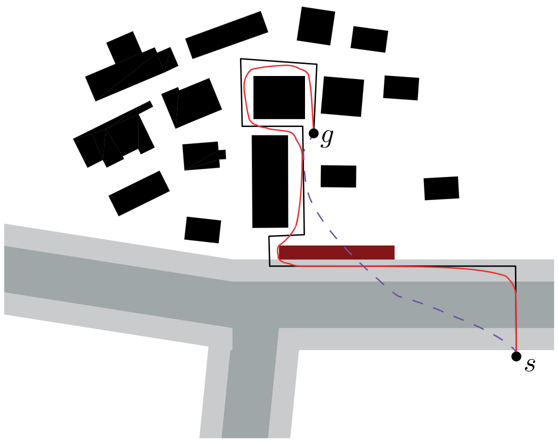

Figure 8. Comparison of MIRAN (solid red path) and IRM (dashed purple path) for a pedestrian character.

polygons with 39 vertices. The character is assumed to be a pedestrian that has a preference for sidewalks. Sidewalks are weighted with a cost of 0.5. The road terrain is weighted with a cost of 4.0, and the mud has a high cost of 40.0. The remaining terrain has a weight of 1.0. For MIRAN, we set the shortcut parameter σ to 40 and the sampling distance d to 10. By running both methods on the same indicative route, the experiment shows that MIRAN lets the character stay on the sidewalk as long as possible. Mud is completely avoided, and no larger parts of the route are skipped. By contrast, the IRM lets the character stay on the road for a long time, crosses the mud, and always skips the last part of the route.

The computation time needed to compute the shown paths can be seen in Table 1. It shows that the number of candidate attraction points has a great impact on the computation time. With a shortcut parameter of 1250 and a sampling distance of 20, the number of candidate attraction points is large, and this yields higher computation times. By contrast, with a shortcut parameter of 40 and a sampling distance of 10, the number of candidate attraction points is small. The amount of time used for computing the path in the *McKenna* + *Road* scene is therefore small as well.

Table 1. Computation times.

Scene	Method	σ	d	Time (ms)
McKenna	MIRAN	100	20	7.13
McKenna	MIRAN	250	20	16.50
McKenna	MIRAN	300	20	16.04
McKenna	MIRAN	400	20	20.90
McKenna	MIRAN	1250	20	63.33
McKenna + Road	MIRAN	40	10	4.66
McKenna + Road	IRM	—	—	4.48
Forest (adult)	MIRAN	80	20	8.00
Forest (child)	MIRAN	100	20	10.52

6. CONCLUSION

We introduced MIRAN, a novel algorithm that enables advanced path planning in environments that contain weighted regions. It can be used to compute visually convincing and terrain-dependent paths in real-time applications, and it solves a variant of the WRP [7]. Those regions can describe a great variety of terrain types or regions of variable attractiveness with respect to psychological influences. The method also overcomes some issues of the IRM [19] by giving the user control over the desired amount of smoothing. We presented the details of the algorithm, proved its correctness, and conducted experiments. The forest example as well as the comparison with the IRM are also illustrated in the video accompanying this paper.

MIRAN can be used by level designers to create individual predefined routes in heterogeneous virtual environments. Furthermore, it can be used as part of an artificial intelligence system when applied to virtual agents or robots that compute their indicative routes automatically with a higher-level path planning approach.

One open research question is how to modify MIRAN to handle a character that is represented as a disk with variable radius. Furthermore, the automated computation of indicative routes can be improved. Up until now, we use a higher-level path planning algorithm such as an A* search [20] on a grid. Applying MIRAN to the corresponding indicative routes already produces convincing results. However, improving this preprocessing step could enhance MIRAN output paths even more.

We believe that MIRAN is a promising and flexible method that can form the basis for solving many challenging path planning problems for future simulation, animation, gaming, and robotics applications.

ACKNOWLEDGEMENTS

This research has been supported by the COMMIT project (http://www.commit-nl.nl/) and the COMMANDS project in cooperation with the European Design Center (http://www.edc.nl/) and the Netherlands Forensic Institute (http://www.forensicinstitute.nl/). Norman Jaklin and Roland Geraerts are part of the Institute of Information and Computing Sciences, Utrecht University, 3584 CC Utrecht, the Netherlands. Atlas Cook IV is part of the Institute for Computational Engineering and Sciences at the University of Texas at Austin, USA.

REFERENCES

1. Latombe J-C. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.
2. LaValle SM. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

3. Choset H, Burgard W, Hutchinson S, *et al. Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, Cambridge, MA, USA, 2005.
4. Shao W, Terzopoulos D. Autonomous pedestrians, In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation,* SCA '05, New York, NY, USA, 2005; 19–28. ACM.
5. Lo W-Y, Knaus C, Zwicker M. Learning motion controllers with adaptive depth perception, In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '12*, Aire-la-Ville, Switzerland, Switzerland, 2012; 145–154. Eurographics Association.
6. Kang S-J, Kim Y, Kim C-H. Live path: adaptive agent navigation in the interactive virtual world. *The Visual Computer: International Journal of Computer Graphics* 2010; **26**(6-8): 467–476.
7. Mitchell JSB, Papadimitriou CH. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM* 1991; **38**(1): 18–73.
8. Aleksandrov L, Djidjev HN, Guo H, Maheshwari A, Nussbaum D, Sack J-R. Algorithms for approximate shortest path queries on weighted polyhedral surfaces. *Discrete & Computational Geometry* 2010; **44**: 762–801.
9. Mata CS, Mitchell JSB. A new algorithm for computing shortest paths in weighted planar subdivisions (extended abstract), In *Proceeding of the 13th Annual ACM Symposium on Computational Geometry*, Nice, France, 1997; 264–273. ACM Press.
10. Aleksandrov L, Lanthier M, Maheshwari A, Sack J-R. An ϵ-approximation algorithm for weighted shortest paths on polyhedral surfaces, In *Proceedings of the 6th Scandinavian Workshop on Algorithm Theory*, Stockholm, Sweden, 1998; 11–22.
11. Aleksandrov L, Maheshwari A, Sack J-R. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM* 2005; **52**(1): 25–53.
12. Reif J, Sun Z. An efficient approximation algorithm for weighted region shortest path problem, In *Algorithmic and Computational Robotics: New Directions - The Fourth Workshop on the Algorithmic Foundations of Robotics*, Dartmouth College, Hanover, NH, USA, 2000.
13. Sun Z, Reif J. Bushwhack: An approximation algorithm for minimal paths through pseudo-euclidean spaces, In *Proceedings of the 12th Annual International Symposium on Algorithms and Computation*, Christchurch, New Zealand, 2001; 160–171. Springer.
14. De Carufel J-L, Grimm C, Maheshwari A, Owen M, Smid M. Unsolvability of the weighted region shortest path problem, In *European Workshop on Computational Geometry (EuroCG)*, Assisi, Perugia, Italy, 2012; 65–68.
15. Guo Y, Parker LE, Jung D., Dong Z. Performance-based rough terrain navigation for nonholonomic mobile robots. *IEEE Industrial Electronics Society* 2003; **3**: 2811–2816.
16. Yahja A, Singh S, Stentz A. An efficient online path planner for outdoor mobile robots. *Robotics and Autonomous Systems* 2000; **32**: 129–143.
17. Harabor D, Botea A. Hierarchical path planning for multi-size agents in heterogeneous environments. *Computational Intelligence and Games* 2008: 258–265.
18. Geraerts R, Overmars MH. Enhancing corridor maps for real-time path planning in virtual environments. *Computer Animation and Social Agents* 2008: 64–71.
19. Karamouzas I, Geraerts R, Overmars MH. Indicative routes for path planning and crowd simulation, In *4th International Conference on Foundations of Digital Games*, On-board the Disney Wonder cruise ship, departing from Port Canaveral, Florida, USA, 2009; 113–120.
20. Hart PE, Nilsson NJ, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 1968; **4**(2): 100 –107.
21. Geraerts R. Planning short paths with clearance using Explicit Corridors, In *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, 2010; 1997–2004.
22. van Toll WG, Cook IV, AF, Geraerts R. Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds (CAVW)* 2012; **23**: 59–69.
23. van Toll WG, Cook IV, AF, Geraerts R. Navigation meshes for realistic multi-layered environments, In *Proceedings of the International Conference on Intelligent Robots and Systems*, San Francisco, California, USA, 2011; 3526–3532.
24. Karamouzas I. Motion Planning for Human Crowds: From Individuals to Groups of Virtual Characters, *Ph.D. Thesis*, Utrecht University, 2012.
25. Bresenham JE. Algorithm for computer control of a digital plotter. *IBM Systems Journal* 1965; **4**(1): 25–30.

AUTHORS' BIOGRAPHIES

Norman Jaklin received his diploma in Computer Science in 2011 from the Rheinische Friedrich-Wilhelms-Universität Bonn, Germany. He is a PhD candidate at the Department of Information and Computing Sciences at Utrecht University since 2011. His research interests cover computational geometry, path planning, discrete mathematics, graph theory, and complexity theory.

Atlas F. Cook IV is a web designer for the Institute for Computational Engineering and Sciences at the University of Texas at Austin. Atlas received his PhD in Computational Geometry in 2009 from the University of Texas at San Antonio. He loves path planning research, gaming, dancing, and smiling.

Roland Geraerts is an assistant professor at the Games and Virtual Worlds group in the Department of Information and Computing Sciences at Utrecht University in the Netherlands. There, he obtained his PhD on sampling-based motion planning techniques. In addition, he studied quality aspects of paths and roadmaps. His current research focuses on path planning and crowd simulation in games and virtual environments. Furthermore, he teaches several courses related to games and crowd simulation. Roland has organized the Creative Game Challenge and is one of the cofounders of the annual Motion in Games conference.