Procedural Generation of Roads

Article *in* Computer Graphics Forum · June 2010

DOI: 10.1111/j.1467-8659.2009.01612.x · Source: dx.doi.org

CITATIONS
80

READS
3,156

4 authors, including:

Eric Galin
Claude Bernard University Lyon 1
98 PUBLICATIONS **1,488** CITATIONS
SEE PROFILE

Adrien Peytavie
Claude Bernard University Lyon 1
39 PUBLICATIONS **616** CITATIONS
SEE PROFILE

Eric Guérin
Institut National des Sciences Appliquées de Lyon
52 PUBLICATIONS **587** CITATIONS
SEE PROFILE

EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Procedural Generation of Roads

E. Galin[1], A. Peytavie[2], N. Maréchal[2], E. Guérin[2]

[1]LIRIS - CNRS - Université Lumière Lyon 2, France [2]LIRIS - CNRS - Université Claude Bernard Lyon 1, France

Abstract

In this paper, we propose an automatic method for generating roads based on a weighted anisotropic shortest path algorithm. Given an input scene, we automatically create a path connecting an initial and a final point. The trajectory of the road minimizes a cost function that takes into account the different parameters of the scene including the slope of the terrain, natural obstacles such as rivers, lakes, mountains and forests. The road is generated by excavating the terrain along the path and instantiating generic parameterized models.

Categories and Subject Descriptors (according to ACM CCS): [Computer Graphics]: Three-Dimensional Graphics and Realism

Keywords: Procedural modeling, road generation, discrete anisotropic shortest path.

1. Introduction

Modeling and rendering realistic images of landscapes and cities is an important problem in computer graphics. The creation of compelling models is a crucial task, not only in the entertainment industry but also in various training, planning and simulation applications.

Over the years, researchers have made considerable progress towards developing efficient techniques for generating natural landscapes covered with vegetation [DHL*98] and cities [PM01, MWH*06]. Procedural algorithms have been developed for generating large cities with complex street networks [CEW*08]. Several methods have been proposed for sketching [BN08] and editing [MS09] roads. The major limitation of editing approaches is that they require a considerable effort to carefully control the trajectories to obtain realistic roads. In contrast, the procedural generation of countryside roads and highways with tunnels and bridges remains an open area of research.

In this paper, we present an algorithm for generating a road connecting an initial and a final point that adapts to the characteristics of an input scene. Given an input scene, we compute the shortest path connecting an initial and a final point that minimizes a cost function that takes into account the slope of the terrain as well as natural obstacles such as rivers, lakes and forests. The discrete shortest path is then converted into a set of piecewise clothoid splines representing the trajectory of the road. This trajectory is further segmented according to the elevation of the terrain as well as rivers to identify which parts are surface roads and which parts should be instantiated as tunnels and bridges. Finally, we excavate the terrain along the path and rely on generic procedural road, bridge and tunnel models to create the final mesh models. Our contributions are as follows.

Figure 1: *A complex road generated by our system*

Control We present a class of parameterized and controllable cost functions that takes into account the different parameters of the scene including the slope of the terrain, natural obstacles such as rivers, lakes, mountains and forests (Section 4). Our generic cost function can also handle the evaluation of the cost of tunnels and bridges between two points in a consistent way.

Anisotropic shortest path We address the computation of the weighted anisotropic shortest path problem on a continuous domain, i.e. the creation of a path that minimizes the line integral of the cost function.

We present an algorithm that reduces the complex problem to an optimization over an implicit finite graph (Section 5). Our method restricts the search to paths formed by the concatenation of straight-line segments between grid aligned points from a uniform discretization of the continuous region. To overcome the limit-on-direction problem, we introduce k-neighborhood connectivity masks so as to generate realistic smooth paths.

We show that our anisotropic shortest path algorithm can generate tunnels and bridges in a consistent way by simply generalizing the optimization process over a more complex finite graph involving a huge number of arcs. Therefore, we propose an accelerated technique based on a stochastic sampling to speed up computations, at the expense of slightly less accurate shortest path.

Procedural generation We present a compact procedural model for representing roads, tunnels and bridges with a few parameters describing their geometrical characteristics. Our method automatically generates the smooth trajectory of the road from the piecewise segment paths, excavates the terrain around the path of the roads and generates the road mesh as well as bridges and tunnels with the appropriate size and characteristics (Section 6).

2. Related work

In this section, we briefly review research describing road generation and editing techniques, variational curve design on surfaces and shortest path algorithms.

City street modeling Several procedural techniques have been proposed to generate street networks [MWH*06] in cities in the context of large-scale urban modeling. Parish and Müller [PM01] first presented a solution to model street networks based on L-systems. Sun et al. [SYBG02] proposed a technique based on template road pattern and Voronoï diagrams. Another approach consists in using tensor fields to guide the generation of street graphs [CEW*08], which allows the user to interactively edit the street graph by either modifying the underlying tensor field or changing the graph directly. Alternatively, example based methods [AVB08, VABW09] have been proposed for interactively synthesizing urban street networks.

Interactive road editing and sketching Several techniques have been proposed for interactively editing and sketching roads. Bruneton et al. [BN08] presented a system to interactively edit very large terrains with very detailed features such as roads, rivers, lakes and fields. McCrae et al. [MS09] proposed a system for interactively editing a road network on a terrain based on user controlled piecewise clothoïd curves [MS08]. Cabral et al. [CLDD09] proposed an approach for modeling architectural scenes by reshaping and combining existing textured models, and demonstrated that this technique could be successfully applied to build complex road structures from simple initial pieces.

Variational path computation Paths can be generated using variational curve design on surfaces as presented in [HP04]. The proposed algorithm minimizes quadratic energy functionals involving first and second derivatives, but with the nonlinear side condition that the solution curves are confined to surfaces.

Anisotropic shortest path problem The solution to shortest-path problem is an active area of research in computational geometry. When strong assumptions on the cost-weighting functions are imposed, efficient algorithms can be used to compute the shortest-path.

A lot of techniques focus on the isotropic case when the cost function only depend on the position. Several algorithms have been proposed for planar regions and obstacle spaces defined by polygons and when the cost function is independent of velocity [MM97, HS99]. The complexity of the problem increases when the cost-weighting function is continuous but still independent of the velocity [MP91, AMS00]. Several efficient algorithms have been proposed to solve an isotropic shortest-path problem by solving a discretized Hamilton Jacobi Bellmann equation [Tsi95, PBT98]. However, none of these algorithms seems to generalize to the anisotropic case.

Much less work has been devoted to the anisotropic case when the cost function depends on the position, velocity and acceleration. Several techniques were proposed for vehicles taking into account the grade of the climb [RR90, LMS99]. Aleksandrov et al. [AMS00, AMS05] proposed a discretization technique to solve the shortest path problem on a weighted terrain. The method involves inserting Steiner points on the edges of the polygonal terrain and then constructing an edge weighted graph. Therefore, the problem simplifies to finding a shortest path between two points in the graph. Kim et al. [KH03] proposed a method based on a non-uniform discretization of the continuous region obtained by a honeycomb sampling algorithm. An alternative technique has been presented in [JV04] using a rectangular grid to discretize the region. Because the cost is anisotropic, rectilinear paths connecting adjacent grid points may not approximate the optimal path. To overcome this limit-on-direction problem, the method searches over shifted versions of the rectilinear paths.

3. Discrete anisotropic shortest path algorithm

In this section, we present an overview of our discrete anisotropic shortest path algorithm for generating roads, tunnels and bridges over complex terrains.

Anisotropic shortest path problem Our paper addresses the weighted anisotropic shortest-path problem on a continuous domain, i.e., the computation of a path between two points that minimizes the line integral of a cost-weighting function along the path.

Consider a compact region $\Omega \in \mathbb{R}^2$ and two initial and final points denoted as $\mathbf{a}$ and $\mathbf{b}$. Our goal is to compute a continuous path ρ from $\mathbf{a}$ to $\mathbf{b}$ that minimizes the line integral over the path of a cost weighting function $c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ that depends on the position $\mathbf{p}$ and the first two derivatives denoted as $\dot{\mathbf{p}}$ and $\ddot{\mathbf{p}}$ respectively.

To formalize the problem, we denote $\mathcal{P}$ the set of all continuous paths in Ω from $\mathbf{a}$ to $\mathbf{b}$ that are piecewise twice continuously differentiable, i.e., $\mathcal{P}$ denotes the set of continuous functions $\rho : [0,T] \to \Omega$, $T > 0$ for which $\rho(0) = \mathbf{a}$ and $\rho(T) = \mathbf{b}$. Let $C : \mathcal{P} \to [0, \infty($ denote the functional characterizing the cost of a path $\rho \in \mathcal{P}$:

$$C(\rho) = \int_0^T c(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))\, \mathrm{d}t$$

The continuous anisotropic shortest path problem consists in finding a path ρ^* that minimizes the functional $C(\rho)$:

$$C(\rho^*) = \min_{\rho \in \mathcal{P}} C(\rho)$$

Discrete anisotropic shortest path To overcome the difficulties that arise in solving the weighted anisotropic shortest path problem exactly, we approximate the solution by performing a uniform discretization of the region Ω into a grid. We define the path as a concatenation of segments between points on the grid. For a finite number of grid points, this procedure converts the original continuous shortest-path problem into a shortest-path problem on a finite graph $\mathcal{G}$.

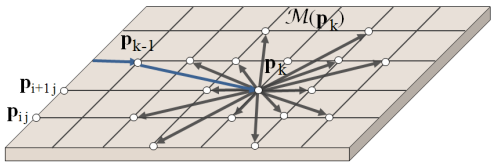

Figure 2: *Notations for the grid points* $\mathbf{p}_{ij}$*, the mask* $\mathcal{M}(\mathbf{p}_k)$ *and the path* $\rho^* = \{\mathbf{p}_k\}_{k \in [0,n]}$.

Grid sampling and graph definition Let $\mathbf{p}_{ij}$, $(i,j) \in [0, n-1]^2$ denote the grid points uniformly sampling the search domain. Those points correspond to the nodes of the graph $\mathcal{G}$. Because the cost function c is anisotropic, the segments connecting a point $\mathbf{p}_{ij}$ to adjacent grid points may not give a good approximation of the optimal path [JV04]. The originality of our approach is to consider that every grid point $\mathbf{p}_{ij}$ is implicitly connected to a large set of neighboring grid points within distance r. We define this subset as:

$$\mathcal{M}(\mathbf{p}_{ij}) \subset \{\mathbf{q}, \|\mathbf{p}_{ij} - \mathbf{q}\| \leq r\}$$

Since explicitly storing those arcs would be memory demanding, we rely on generic path segment masks, denoted as $\mathcal{M}_k$ that will store the connectivity patterns between grid points (Section 5). This technique enables us to overcome the limit on direction problem that arise when considering only 4 or 8 connectivity between the grid points $\mathbf{p}_{ij}$.

Shortest path computation We compute the discrete shortest path by applying an A* algorithm to the graph $\mathcal{G}$.

Recall that A* is a best-first graph search algorithm that finds the least-cost path from a given initial node to a goal node. We use the cost function $c(\mathbf{p})$ plus an admissible heuristic cost estimate function $e(\mathbf{p})$ to determine the order in which the search visits nodes. We define the function $e(\mathbf{p})$ as the cost of a straight-line road to the goal $\mathbf{b}$ so that it never overestimates the actual minimal cost of reaching $\mathbf{b}$. The heuristic function e is monotonic, therefore A* itself is admissible.

Let us present the outline of the algorithm. For every grid point, we define its corresponding cost value $c(\mathbf{p}_{ij})$ and its predecessor as $p(\mathbf{p}_{ij})$. The cost value of all the points $c(\mathbf{p}_{ij})$ is first initialized with ∞, whereas the value of the starting point $c(\mathbf{a})$ is set to $h(\mathbf{b})$. We initialize a priority queue $\mathcal{Q}$ with the initial point $\mathbf{a}$. The main loop of the algorithm can be written as follows:

1. While $\mathcal{Q}$ is not empty, select the point $\mathbf{p}_{ij}$ from the priority queue with the smallest cost value $c(\mathbf{p}_{ij})$.
2. If destination has been found $\mathbf{p}_{ij} = \mathbf{b}$, stop the algorithm.
3. For all points $\mathbf{q} \in \mathcal{M}_k(\mathbf{p}_{ij})$, evaluate the cost $c(\mathbf{p}_{ij}, \mathbf{q})$ over the segments $[\mathbf{p}_{ij}, \mathbf{q}]$. If $c(\mathbf{p}_{ij}, \mathbf{q}) < c(\mathbf{q})$ then set the predecessor of $\mathbf{q}$ as $\mathbf{p}_{ij}$.

This algorithm generates a discrete path characterized by a set of grid points and denoted as $\rho^* = \{\mathbf{p}_k\}$, $k \in [0,n]$.

Step 3 of the algorithm involves the computation of the set of grid points $\mathcal{M}_k(\mathbf{p}_{ij})$ to define which nodes are connected together. It also requires the evaluation of the cost function along line segments connecting two grid points.This cost will be computed on the fly as follows. Let $[\mathbf{p}_k, \mathbf{p}_{k+1}]$ denote a segments of the path and t_k, t_{k+1} the parameters corresponding to $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$. The line integral is defined as:

$$c(\mathbf{p}_k, \mathbf{p}_{k+1}) = \int_{t_k}^{t_{k+1}} c(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))\, dt$$

We approximate the line integral as a finite sum by discretizing the integration domain into n intervals.

Cost functions definition We define a set of parameterized cost functions $c(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))$ that will be used to evaluate the line integral of the cost function. Those functions are defined by the user and indirectly control the trajectory of the path by constraining the shortest path research. In the next section, we address the computation of the cost functions.

4. Cost functions

In this section, we present a class of cost functions that define the influence of the slope of the terrain and the natural obstacles over the shortest path computation. We define the global cost function c as a sum of several weighting functions that evaluate relative influence of the different characteristics of the terrain and the road. We propose to define the cost function as follows:

$$c(\mathbf{p},\dot{\mathbf{p}},\ddot{\mathbf{p}}) = \sum_{i=0}^{i=n-1} \mu_i \circ \kappa_i(\mathbf{p},\dot{\mathbf{p}},\ddot{\mathbf{p}})$$

The set of functions $\kappa_i : \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ evaluate the different characteristics of the terrain and the geometric characteristics of the trajectory of the road at point $\mathbf{p}$. The functions μ_i are transfer functions that weight and combine the influence of the characteristics $\kappa_i(\mathbf{p},\dot{\mathbf{p}},\ddot{\mathbf{p}})$. Transfer functions allow the user to control the relative influence of the parameters of the scene and therefore control the shape of the minimum shortest path.

4.1. Surface roads

The trajectory of a road on the surface of a terrain should be constrained by the slope of the terrain and natural obstacles such as rivers, lakes and forests (Figure 3). The cost function also takes into account the curvature of the road so as to control and constrain whether the generated road should avoid sharp curves.

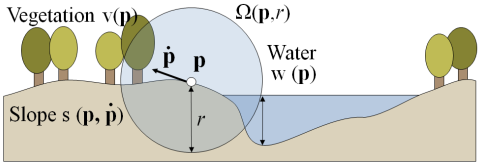

Figure 3: *Overview of the evaluation of the cost function for surface roads: we evaluate the characteristics $\kappa_i(\mathbf{p},\dot{\mathbf{p}},\ddot{\mathbf{p}})$ of the scene in the neighborhood of $\mathbf{p}$ and apply transfer functions μ_i to weight their relative influence.*

Characteristic functions Let $\mathbf{p}$ denote a point on the trajectory of a road on the surface of the terrain. We compute the following characteristic functions: the density of vegetation $v(\mathbf{p})$, the water height $w(\mathbf{p})$, the slope of the terrain in the direction of the trajectory of the road $s(\mathbf{p},\dot{\mathbf{p}})$ and the curvature of the trajectory of the road $c(\mathbf{p},\dot{\mathbf{p}},\ddot{\mathbf{p}})$.

The water height $w(\mathbf{p})$ is defined as the maximum height of water in a small area $\Omega(\mathbf{p},r)$ around the projection of $\mathbf{p}$ onto the ground. $\Omega(\mathbf{p},r)$ denotes a sphere centered at point $\mathbf{p}$ and of radius r. The density of vegetation $v(\mathbf{p})$ is computed by evaluating the number of trees that lie within $\Omega(\mathbf{p},r)$.

Transfer functions In this section, we present transfer functions that weight the influence of the characteristics of the scene and road trajectory. In our system, transfer functions are characterized by their graph which is edited interactively (Figure 4).

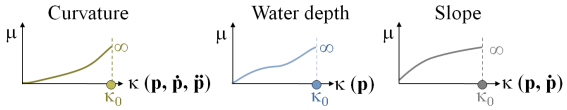

Figure 4: *Synthetic representation of the cost functions for surface roads.*

Transfer functions are characterized by a threshold value, denoted as κ_0. If the input characteristic κ is greater than κ_0, the corresponding transfer function $\mu(\kappa)$ should return infinity. This enables us to control the definition of regions where path cannot be created. Let us review two transfer functions.

Slope Let $s(\mathbf{p},\dot{\mathbf{p}})$ denote the slope of the terrain at point $\mathbf{p}$. If the slope is too steep, the transfer function should return infinity so as to prevent paths from climbing very steep slopes which would generate unrealistic roads. Otherwise, we use a function of the slope controlled by the maximum cost value $\mu(\kappa_0)$.

Water Let $w(\mathbf{p})$ denote the maximum water depth in $\Omega(\mathbf{p},r)$. If the water is too deep, it is impossible to create a road and the water transfer function should return infinity. Otherwise, the water height is small enough to create the road at the expense of some banking. In that case, we use a function of the water depth, and $\mu(\kappa_0)$ represents the maximum cost corresponding to the maximum water depth.

4.2. Bridges and tunnels

Bridges and tunnels are complex structures whose creation cost depends on many parameters including the geological properties of the ground. In our system, the cost for bridges is parameterized by the height of the bridge $h(\mathbf{p})$, its slope and the depth of water bodies the bridges crosses (Figure 5).

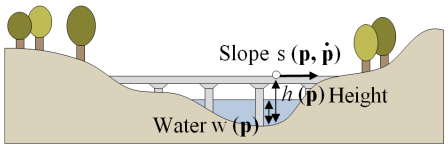

Figure 5: *Bridge cost function evaluation.*

The cost function also takes into account the curvature so as to avoid the generation of dangerous bridges with sharp curves. The function $h(\mathbf{p})$ denotes the vertical height of the

bridge at point **p** with respect to the ground. The corresponding transfer function μ_{height} is a piecewise function that returns infinity if the height is greater than maximum height value for the considered type of bridge.

In the case of tunnels, the cost is parameterized by the depth of the tunnel to the surface $d(\mathbf{p})$, its slope, and the depth of water bodies that may exist above the tunnel (Figure 6). This latter parameter enables us to simulate expensive tunnels passing under rivers or seas.

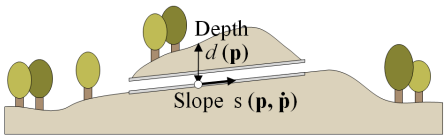

Figure 6: *Tunnel cost function evaluation.*

5. Segment path masks

In this section, we describe our method for computing the anisotropic shortest path over a uniformly sampled grid.

The limit on direction problem An important problem of uniform grid sampling is that it is not efficient as the number of grid points grows very fast if we want the shortest discrete path to approximate the position and direction of the shortest continuous path. This is because of the limit-on-direction effect (Figure 7).

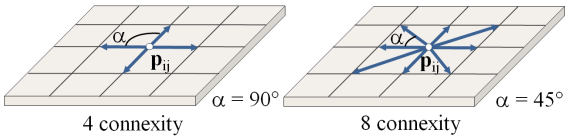

Figure 7: *The limit on direction problem when using* 4 *and* 8 *connexity between sample grid points.*

Recall that the discrete path is a rectilinear path composed of straight segments connecting sample points. Existing techniques only consider 4 or 8 connectivity between sample points. Therefore, the segments of the discrete path can only have 4 or 8 directions, with a maximal angle resolution of 45 degrees. Shift paths [JV04] can partially overcome this problem by allowing paths to shift away from the grid points, but require a computationally demanding relaxation step to shift path segments.

Overview To overcome the limit on direction problem, we propose to increase the neighborhood distance so as to introduce more directions, which enables us to improve the angle accuracy between path segments. This approach also allows us to consider very long arcs connecting distant grid points, which enables us to create bridge and tunnels.

5.1. Path segment masks

Since explicitly storing all the arcs between grid points would be memory consuming, we propose to store the connectivity information between grid points using a set of path segment masks, denoted as $\mathcal{M}_k$ (Figure 8).

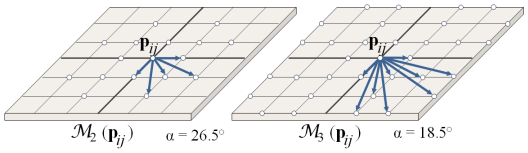

Figure 8: *Synthetic representation of the path segments for the masks $\mathcal{M}_2$ and $\mathcal{M}_3$. Only a few path segments have been drawn out of clarity.*

Definition We define the path segment masks $\mathcal{M}_k$ as the set of segments connecting the origin $(0,0)$ to a point $(i,j) \in [-k,k]^2$ such that the greatest common divisor of i and j is 1 (Figure 8). Using this definition instead of all the $(2k+1)^2$ segments with $(i,j) \in [-k,k]^2$ avoids redundant path checking when applying the discrete shortest path algorithm.

k	1	2	3	4	5	6
α	45	26.5	18.5	14.0	11.3	9.5
n_k	8	16	32	48	80	96

Table 1: *Statistics for segment path masks :* α *denotes the angle resolution and n_k the number of segment paths.*

Influence of the size of the masks Increasing the size of the segment path masks enables us to detect paths with a better angle resolution and to reduce the limit on direction effect, at the cost of an increasing number of iterations in the shortest path algorithm. Table 1 reports the number of path segments n_k as well as the maximum angle α.

Figure 9 shows the influence of the parameter k over the computation of the shortest path between an initial point in a valley and a final point at the top of a steep hill. The cost function has been set to take into account the slope. For $k = 1$, the limited connectivity between grid points results in a shortest path following an unrealistic direct path to the top, ignoring the influence of the slope. As k increases, the resulting path gets more realistic.

Table 2 reports timings (in seconds) as well as the cost (in arbitrary unit) and the length (in meters) for computing the anisotropic shortest path with different path segment masks over a 60×60 grid. Timings increase in $O(k^2)$.

Experiments demonstrate that the cost of the shortest path converges to a limit as the size of the mask increases. In general, chosing $k = 5$ proved to be a good compromise between speed and the angle resolution to generate realistic paths.

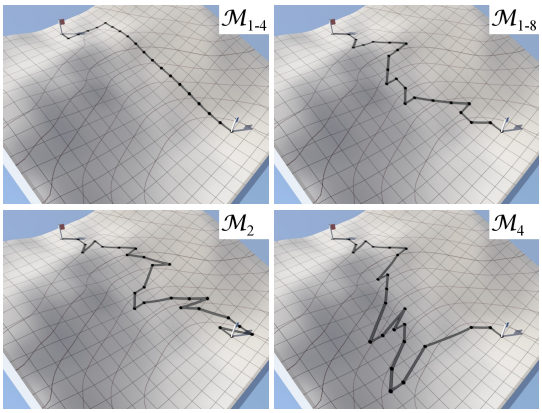

Figure 9: *Influence of the size of the masks $\mathcal{M}$.*

k	Cost	Time	Length
1	29265	0.015	477
2	26441	0.031	808
3	26242	0.062	1225
4	26096	0.110	1757

Table 2: *Statistics for different sizes of masks*

5.2. Curvature

While segment path masks enable us to overcome the limit on direction problem, the generated path often have too many shape turns (Figure 9). Therefore, taking the curvature into account is crucial to obtain realistic paths. Instead of using a two-dimensional discretization of the domain $\Omega \subset \mathbb{R}^2$, we perform a three-dimensional discretization of the continuous domain $\Omega \times [0,2\pi]$ which represents all the positions in Ω with all the possible orientations (Figure 10). We discretize the angular domain $[0,2\pi]$ into m intervals. Thus, our approach consists in computing the discrete anisotropic shortest path over a $n^2 \times m$ three-dimensional grid of oriented points, denoted as $\mathbf{p}_{ija}$, $(i,j) \in [0,n]^2$, $a \in [0,m-1]$. The segments path masks $\mathcal{M}$ generalize to extended masks, denoted as $\mathcal{E}$, such that $\mathcal{E}(\mathbf{p}_{ija})$ refers to the set of all the points $\mathcal{M}(\mathbf{p}_{ij})$ replicated for all $a \in [0,m-1]$.

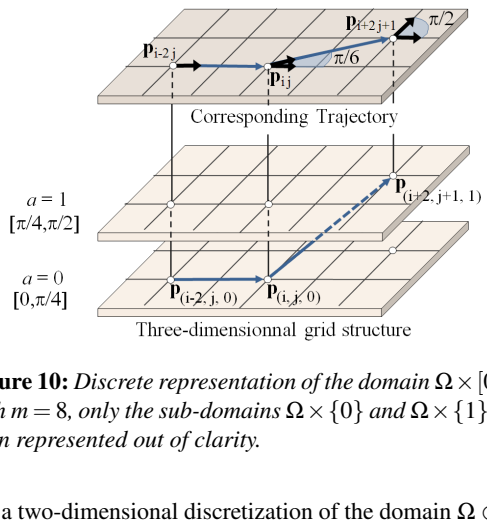

Figure 10: *Discrete representation of the domain $\Omega \times [0,2\pi]$ with $m = 8$, only the sub-domains $\Omega \times \{0\}$ and $\Omega \times \{1\}$ have been represented out of clarity.*

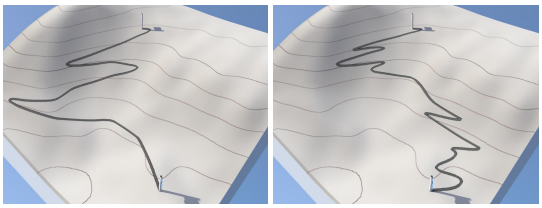

Figure 11: *An uphill road with (left) and without (right) curvature constraints.*

This technique, combined with cost functions taking account the curvature of the path, enable us to keep track of the influence of the curvature cost and to generate more realistic paths, at the expense of more computations (Figure 11). Table 3 reports the corresponding statistics for a 60×60 grid with an angle discretization set to $m = 12$.

Technique	Cost	Time	Length
Without orientation	38884	0.110	1464
With orientation	37157	1.750	0998

Table 3: *Curvature constrained shortest path: cost (in arbitrary unit), time (in seconds) and road length (in meters).*

5.3. Tunnels and bridges

Tunnels and bridges can be processed in a consistent way using an extension of path segments mask. The fundamental concept is to consider bridges and tunnels as long path segments connecting two distant grid points.

Without loss of generality, let us first consider tunnels. As for surface roads, we introduce generic tunnel masks, denoted as $\mathcal{T}$, that represent which paths connecting two grid points should be processed as tunnels by the shortest path algorithm (Figure 12).

Those path segments have a minimum and a maximum distance, denoted as r_i and r_e, which correspond to the minimum and maximum length a given type of tunnel or bridge can have. Therefore, we define:

$$\mathcal{T}(\mathbf{p}_{ij}) = \{\mathbf{q} \neq \mathbf{p}_{ij} \,|\, r_i \leq \|\mathbf{p}_{ij} - \mathbf{q}\| \leq r_e\}$$

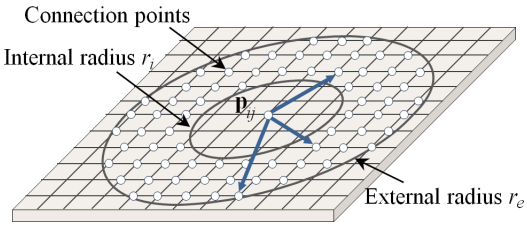

Figure 12: *Tunnel path segment masks $\mathcal{T}(\mathbf{p}_{ij})$. Only three segments have been drawn out of clarity.*

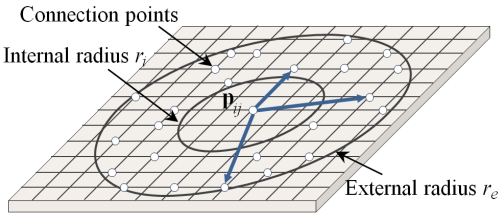

Figure 14: *Stochastic tunnel path segment masks $\mathcal{S}(\mathbf{p}_{ij})$. Only three segments have been drawn out of clarity.*

Step 3 of the graph search algorithm is modified as follows. When processing a candidate grid point $\mathbf{p}_{ij}$, we compute and update the value of the grid points in two sets $\mathbf{q} \in \mathcal{M}(\mathbf{p}_{ij})$ and $\mathbf{q} \in \mathcal{T}(\mathbf{p}_{ij})$.

3. For all points $\mathbf{q} \in \mathcal{M}(\mathbf{p}_{ij})$, evaluate the surface road cost $c(\mathbf{p}_{ij}, \mathbf{q})$, and for all points $\mathbf{q} \in \mathcal{T}(\mathbf{p}_{ij})$, evaluate the tunnel cost $c(\mathbf{p}_{ij}, \mathbf{q})$. If $c(\mathbf{p}_{ij}, \mathbf{q}) < c(\mathbf{q})$ then set the ancestor of $\mathbf{q}$ as $\mathbf{p}_{ij}$.

Bridges are defined in the same way, the corresponding masks for bridges will be referred to as $\mathcal{B}$.

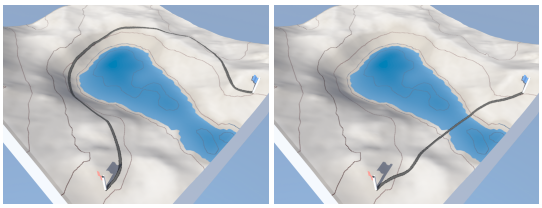

Figure 13: *Left image shows a long road without bridge, whereas right image shows a shorter path obtained by enabling the bridge generation.*

Figure 13 presents two different roads generated with and without bridges. The corresponding statistics (Table 4) demonstrate that enabling bridges effectively reduces the cost of the path, at the expense of demanding computations.

Technique	Cost	Time	Length
Bridge enabled	38 740	3.016	461
Road only	41 778	0.078	1 043

Table 4: *Statistics for two different roads: cost (in arbitrary unit), time (in seconds) and road lenth (in meters).*

5.4. Stochastic sampling

Detecting all the possible tunnels and bridges in a scene involves evaluating the corresponding cost function for a huge number of arcs. The number of arcs becomes quadratic as the external radius of the search region r_e increases.

Therefore, we propose a stochastic sampling technique to speed up computations. Instead of evaluating the tunnel and bridge functions values for all the grid points in the set $\mathcal{T}(\mathbf{p}_{ij})$ at every step of the algorithm, we only perform computations for a restricted subset of segment paths $\mathcal{S}(\mathbf{p}_{ij}) \subset \mathcal{T}_k(\mathbf{p}_{ij})$. Candidate grid points are obtained by stochastically selecting some of the grid points inside the hollow disc with a low discrepancy sequence (Figure 14) at every iteration of the algorithm.

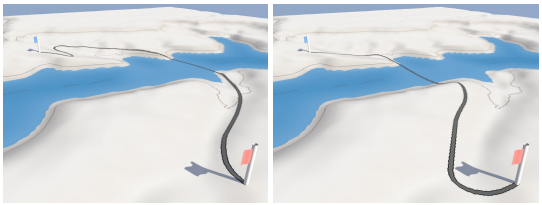

Figure 15: *Using stochastic path masks may result in slightly different road trajectories.*

Table 5 reports statistics corresponding to the shortest paths illustrated in Figure 15 and demonstrating the efficiency of the sampling technique. The tunnel and bridge mask areas were set with $r_i = 50\,\text{m}$ and $r_e = 300\,\text{m}$ over a 300×300 grid with a sampling grid size of 10 m. The corresponding number of grid points visited at every iteration was equal to $\#\mathcal{T} = 2728$. In contrast, the number of sample grid points in the stochastic approach was set to $\#\mathcal{S} = 50$. Timings demonstrate that the speed up is proportional to the ratio $\#\mathcal{S}/\#\mathcal{T}$.

Technique	Cost	Time	Length
Bridge	21 351	25.8	1 120
Stochastic bridge	28 952	3.2	752

Table 5: *Statistics for segment path masks: cost (in arbitrary unit), time (in seconds) and road length (in meters).*

Figure 16: *A mountain road following the path of a river.*

Timings demonstrate that our stochastic algorithm dramatically speeds up the computation of tunnels and bridges, with a very small cost overhead.

6. Procedural generation of road models

In this section, we present our method for generating the road, tunnel and bridge models from the discrete shortest path. Our method proceeds in three steps. First, the discrete shortest path is converted into a set of piecewise clothoid splines representing the trajectory of the road. We rely on a piecewise clothoid splines as these curves matches real world curvature constraints. The trajectory is further segmented to identify which parts should be instantiated as roads, tunnels or bridges. We modify the terrain and the vegetation along the trajectory, performing excavations and embankments, and generate the mesh representation of roads, tunnels and bridges from generic parameterized procedural models.

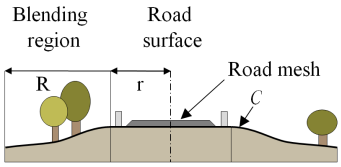

Figure 17: *Cross section of our surface road model.*

Generic models Roads, tunnels and bridges are implemented as procedural geometric models. This technique enables us to adapt the geometry to the constraints of the trajectory. Roads are characterized by a profile curve $\mathcal{C}$ parameterized by the width r of the asphalt section and the width R of a blending region (Figure 17). The blending region will be used to characterize the shape of excavations and embankments performed along the trajectory of the road.

6.1. Trajectory computation

Recall that our discrete anisotropic shortest path algorithm generates a path defined as $\rho^* = \{\mathbf{p}_k\}_{k\in[0,n]}$ where the points $\mathbf{p}_k$ are located on the grid points $\mathbf{p}_{ij}$ discretizing the search domain Ω. We create a piecewise clothoid curve, denoted as Γ, from the control points $\mathbf{p}_k$ as proposed in [WM05]. This enables us to generate smooth and realistic road trajectories.

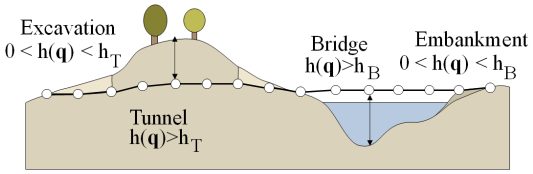

Figure 18: *Segmentation of the trajectory of the road and corresponding terrain modifications.*

Because of the smoothing process, some parts of the curve Γ corresponding to surface path segments may lie slightly inside or above the terrain. Therefore, we perform a segmentation of the trajectory with respect to the elevation of the terrain so as to identify which parts should be generated using the generic road, tunnel or bridge models (Figure 18).

The segmentation is performed by uniformly sampling Γ so as to generate a piecewise linear curve denoted as $\Gamma^* = \{\mathbf{q}_k\}$, $k \in [0,n]$. For every point in the discrete curve, we compute $h(\mathbf{q}_k)$ as the difference between the height of the point $\mathbf{q}_k$ and the height of its projection on the surface of the terrain. Let $h_\mathcal{T}$ the minimum height value for creating a tunnel, and $h_\mathcal{B}$ the minimum height value for creating a bridge. The segmentation is performed as follows: if $h(\mathbf{q}_k) < h_\mathcal{T}$: label $\mathbf{q}_k$ as a tunnel point, if $h_\mathcal{T} < h(\mathbf{q}_k) < h_\mathcal{B}$: label $\mathbf{q}_k$ as a road point, otherwise label $\mathbf{q}_k$ as a bridge point.

At the end of this process, the trajectory Γ^* is consistently defined as a piecewise linear curve created from clothoid splines and every point $\mathbf{q}_k$ is unambiguously labeled as road, tunnel or bridge.

6.2. Road generation

Recall that R denotes the width of the road model, we define the neighborhood of the curve Γ^* as:

$$\Omega_\Gamma = \{\mathbf{p} \in \Omega \,|\, d(\mathbf{p},\Gamma) < R\}$$

The road generation proceeds in two steps. First, we remove the existing vegetation in the vicinity of the road by removing tree instances that lie in Ω_Γ. The terrain is also modified

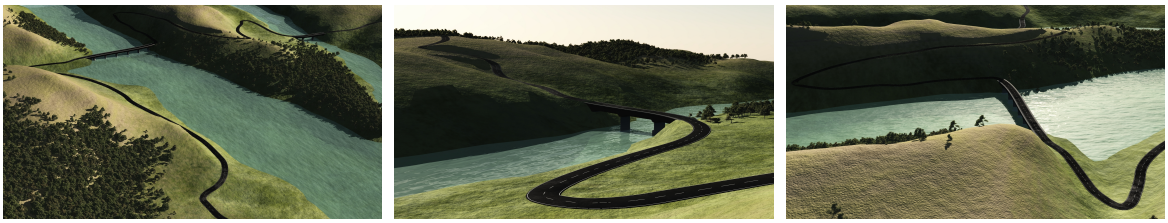

Figure 19: *Different road paths obtained by modifying the relative influence of the cost functions.*

in the neighborhood of the trajectory as follows. First, we perform an adaptive refinement of the mesh of the terrain in the region Ω_Γ (Figure 20). For every vertex $\mathbf{v}$ of the refined mesh, we compute the distance $d(\mathbf{v},\Gamma)$ to the trajectory. Excavation and embankment are performed as follows:

1. If $d(\mathbf{v},\Gamma) < r$, set the height of vertex $\mathbf{v}$ to the height of the corresponding nearest point on the curve.
2. If $r < d(\mathbf{v},\Gamma) < R$, compute the height of the vertex by interpolating the profile curve of the road $\mathcal{C}(d(\mathbf{v},\Gamma))$ with the height of the terrain.

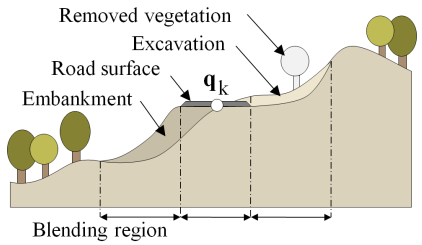

Figure 20: *Cross section representation of terrain modifications performed after the road segmentation process.*

Finally, we generate the road, tunnel and bridge models from their parameterized definition and the discrete segmented trajectory which enables us to adapt to the characteristics of the terrain automatically.

7. Results

We have implemented our procedural road generation technique into a modeling application coded in C++. We have applied our method to create the different images shown throughout this paper. Renderings were performed by using Mental Ray on the textured meshes produced by our method.

Realism Our method creates realistic trajectories and compares favorably in terms of efficiency and quality to existing editing and sketching techniques [MS09]. The main reason for this is that our algorithm generates an optimal path that satisfies a combination of realistic cost constraints. In particular, combining slope and curvature based costs enables us to automatically create very realistic mountain roads (Figure 16). Our algorithm also finds the best locations to create bridges and tunnels with a view to avoiding long detours.

Control A very interesting and powerful feature of our approach is its simplicity and control. The road generation process can be controlled very efficiently by tuning a few transfer functions (slope, curvature, vegetation, water height). In our system, the curves of those functions can be interactively modified.

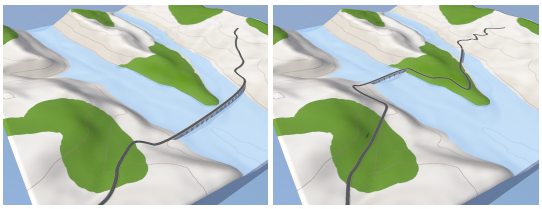

Figure 21: *Influence of the water and forest cost functions.*

Figure 21 illustrates the influence of the forest and water cost functions over the trajectory of the road. When the cost function of the forest is high and the bridge cost is low, our algorithm generates a long bridge crossing the river (left image). In contrast, when the impact of the forest over the global cost function is low and when the bridge cost is high, the method generates a longer road passing through forests with two small bridges (right image).

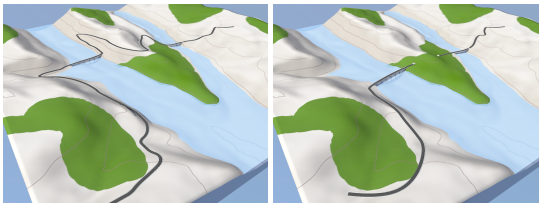

Figure 22: *Influence of the slope cost function.*

Figure 22 shows the influence of the slope cost function over the trajectory. By setting the maximum authorized slope to a high value and lowering the influence of the slope cost function over the global cost, the generated path is a long

surface road (left image). In contrast, if the maximum authorized slope is very low, it is no longer possible to create a road going up the hills and the algorithm creates a tunnel (right image).

Efficiency The performance of our algorithm depends on the resolution of the underlying sampling grid. Our experiments show that our optimized algorithm can generate realistic trajectories in less than 1 second when using a 100×100 grid with both curvature control and tunnel and bridge detection activated.

8. Conclusion

We have proposed a complete framework for generating realistic roads in complex scenes. Our approach relies on a discrete anisotropic shortest path algorithm applied to a graph whose nodes are obtained by a uniform sampling of the scene and whose arcs are implicitly defined using generic segment path masks. The trajectory of roads can be easily controlled by adjusting the parameters of the transfer functions that weight the relative influence of the characteristics of the terrain.

This work is the first step towards a solution to the much more complex and general problem of modeling a complete hierarchical road network connecting cities. We are currently investigating this research field.

Acknowledgments

This work was supported by Agence Nationale de la Recherche as ANR-06-MDCA-004-01 project. We wish to credit Etienne Duranton, Virginie Perrier-Cornet and Guillaume Hayette (LIRIS) for their assistance in the implementation of the framework.

References

[AMS00] ALEKSANDROV L., MAHESHWARI A., , SACK J.-R.: Approximation algorithms for geometric shortest path problems. In *Proceedings of the 32nd ACM symposium on Theory of Computing* (2000), pp. 286–295. 2

[AMS05] ALEKSANDROV L., MAHESHWARI A., , SACK J.-R.: Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM 52* (2005), 25–53. 2

[AVB08] ALIAGA D., VANEGAS C., BENEŠ B.: Interactive example-based urban layout synthesis. *ACM Transaction on Graphics 27*, 5 (2008), 1–10. 2

[BN08] BRUNETON E., NEYRET F.: Real-time rendering and editing of vector-based terrains. *Computer Graphics Forum (Proceedings Eurographics) 27*, 2 (2008), 311–320. 1, 2

[CEW*08] CHEN G., ESCH G., WONKA P., MÜLLER P., ZHANG E.: Interactive procedural street modeling. *ACM Transactions on Graphics 27*, 3 (2008), 1–10. 1, 2

[CLDD09] CABRAL M., LEFEBVRE S., DACHSBACHER C., DRETTAKIS G.: Structure preserving reshape for textured architectural scenes. *Computer Graphics Forum (Proceedings Eurographics) 27*, 2 (2009), 469–480. 2

[DHL*98] DEUSSEN O., HANRAHAN P., LINTERMANN B., MĚCH R., PHARR M., PRUSINKIEWICZ P.: Realistic modeling and rendering of plant ecosystems. In *Proceedings of SIGGRAPH* (1998), pp. 275–286. 1

[HP04] HOFER M., POTTMANN H.: Energy-minimizing splines in manifolds. *ACM Transactions on Graphics 23*, 3 (2004), 284–293. 2

[HS99] HERSHBERGER J., SURI S.: An optimal algorithm for euclidean shortest paths in the plane. *SIAM Journal on Computing 28*, 6 (1999), 2215–2256. 2

[JV04] JIA Z., VARAIYA P.: Grid discretization based method for anisotropic shortest path problem over continuous regions. In *IEEE Conference on Decision and Control* (2004), pp. 3830–3837. 2, 3, 5

[KH03] KIM J., HESPANHA J. P.: Discrete approximations to continuous shortest-path: Application to minimum-risk path planning for groups of UAVs. In *42nd IEEE Conference on Decision and Control* (2003), vol. 2, pp. 1734–1740. 2

[LMS99] LANTHIER M., MAHESHWARI A., SACK J.-R.: Shortest anisotropic paths on terrains. In *26th International Colloquium on Automata, Languages and Programming* (1999), vol. 1644, Lecture Notes in Computer Science, pp. 524–533. 2

[MM97] MAHESHWARI S., MITCHELL J.: An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete Computational Geometry 18* (1997), 366–383. 2

[MP91] MITCHELL J., PAPADIMITRIOU C.: The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM 38* (1991), 18–73. 2

[MS08] MCCRAE J., SINGH K.: Sketching piecewise clothoid curves. In *Sketch-Based Interfaces and Modeling* (2008), pp. 1–8. 2

[MS09] MCCRAE J., SINGH K.: Sketch-based path design. In *Proceedings of Graphics Interface* (2009), pp. 95–102. 1, 2, 9

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. In *Proceedings of ACM SIGGRAPH* (2006), pp. 614–623. 1, 2

[PBT98] POLYMENAKOS L. C., BERTSEKAS D. P., TSITSIKLIS J. N.: Implementation of efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control 43*, 2 (1998), 278–283. 2

[PM01] PARISH Y., MÜLLER P.: Procedural modeling of cities. In *Proceedings ACM SIGGRAPH* (2001), pp. 301–308. 1, 2

[RR90] ROWE N., ROSS R.: Optimal grid-free path planning across arbitrarily contoured terrain with anisotropic friction and gravity effects. *IEEE Transactions on Robotics and Automation 6*, 5 (1990), 146–152. 2

[SYBG02] SUN J., YU X., BACIU G., GREEN M.: Template-based generation of road networks for virtual city modeling. In *Proceedings of the ACM symposium on Virtual reality software and technology* (2002), pp. 33–40. 2

[Tsi95] TSITSIKLIS J.: Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control 40*, 9 (1995), 1528–1538. 2

[VABW09] VANEGAS C. A., ALIAGA D. G., BENEŽ B., WADDELL P.: Visualization of simulated urban spaces: Inferring parameterized generation of streets, parcels, and aerial imagery. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 424–435. 2

[WM05] WALTON D., MEEK D.: A controlled clothoid spline. *Computers & Graphics 29*, 3 (2005), 353–363. 8