Pacific Graphics 2011
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin
(Guest Editors)

*Volume 30 (2011), Number 7*

# Authoring Hierarchical Road Networks

Eric Galin[1], Adrien Peytavie[2], Eric Guérin[3], Bedřich Beneš[4].

[1]LIRIS - CNRS - Université Lumière Lyon 2, France [3]LIRIS - CNRS - INSA Lyon, France
[2]LIRIS - CNRS - Université Claude Bernard Lyon 1, France [4]Purdue University, USA

**Abstract**

*We present a procedural method for generating hierarchical road networks connecting cities, towns and villages over large terrains. Our approach relies on an original geometric graph generation algorithm based on a non-Euclidean metric combined with a path merging algorithm that creates junctions between the different types of roads. Unlike previous work, our method allows high level user control by manipulating the density and the pattern of the network. The geometry of the highways, primary and secondary roads as well as the interchanges and intersections are automatically created from the graph structure by instantiating generic parameterized models.*

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

## 1. Introduction

Procedural modeling of urban areas in computer graphics has undergone an important progress in recent years. Still, modeling and synthesizing realistic virtual worlds remains an open problem. The authoring of compelling models is a crucial task with many applications not only in the entertainment industry but also in training, urban planning and simulation applications.

Procedural and simulation techniques have proved to be very efficient for synthesizing natural landscapes covered with vegetation [DHL*98], and large cities [PM01, MWH*06] with complex street networks [CEW*08]. Recently, several methods have been proposed for sketching [BN08], editing [MS09] and generating roads [GPMG10]. Nevertheless, the automatic generation of complex road networks connecting cities of different size and featuring highways as well as primary and secondary roads, connected by interchanges and crossroads, remains an open area of research.

From a land planning and simulation perspective, the evolution of the road network can be achieved by complex simulations of different phenomena such as urban growth, transitions in land use, changes in population density and migrations, transportation policies and infrastructures development. Although this has been addressed *e.g.*, in [VABW09], the controllability and user input remain a problem. Our solution tackles the problem as a geometric optimization. The challenge stems not only from the difficulty to synthesize realistic road network patterns adapting to their geographical environment, but also from the complexity of the geometric models that should adapt to the terrain and take into account natural obstacles such as rivers or lakes.

### 1.1. Related work

Our approach builds on the early work of [PM01] who modeled street geometry in two passes: first the underlying graph was generated and second it was filled with geometry. We focus on the modeling of road networks outside cities. Moreover, we also provide the geometry of tunnels, bridges, interchanges and junctions. Our approach can be complemented with shape grammar [MWH*06] for buildings and ecosystem simulation [DHL*98] to obtain a complete system for generating virtual environments. In this section, we review research describing city street and countryside road modeling techniques.

Several techniques have been proposed to generate street networks in cities in the context of large-scale urban modeling [MWH*06, WMWG09]. Parish and Muller [PM01] first presented a solution to model street networks based on L-systems. The major limitation is that the generated street graph often requires a significant amount of editing to meet the desired patterns and suffers from a low controlability (partially addressed in [LSWW11]). An alternative technique based on template road pattern and Voronoï

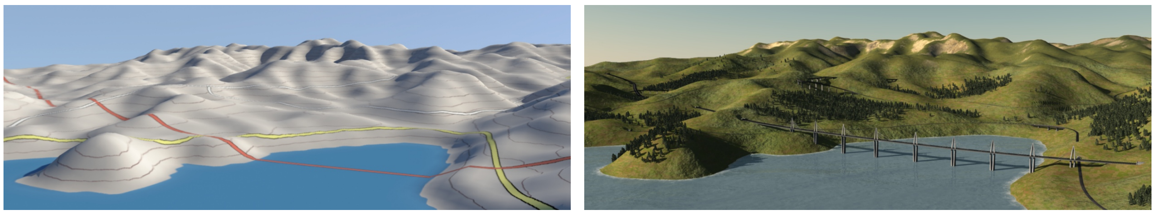

**Figure 1:** *A hierachical road network and the corresponding synthetic scenery generated by our system.*

diagrams to imitate the streets layout was proposed in [SYBG02]. Another technique based on tensor fields was proposed in [CEW*08]. The tensor field allows the user to interactively edit the street graph. Example based methods [AVB08, VABW09] have been proposed for interactively synthesizing urban street networks. While these techniques are effective for city streets, they cannot be generalized to countryside roads whose patterns and layouts are fundamentally different and heavily influenced by the terrain geometry.

Several methods have been proposed for interactive sketching [BN08] and editing [MS09] road networks. The limitation of those methods is that they require a considerable effort to carefully control the trajectories in order to obtain realistic paths. Recently, a procedural technique for generating a single road based on an anisotropic shortest path algorithm was proposed in [GPMG10]. The trajectory of the road minimizes a cost function that takes into account 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. Our work extends this model by proposing a complete framework for generating a hierarchical road network connecting cities and villages, but in a fundamentally different perspective as we address the complex problem of the generation of a complex geometric graph with Steiner points connecting an input set of cities, towns and villages.

## 1.2. Contributions

We present a procedural framework for creating a hierarchical road network connecting a set of cities. Starting with highways, we incrementally generate and merge the different layouts of the road network towards the primary and secondary roads. For every level of the hierarchy, we generate the layout of the sub-network as a proximity graph using a non-Euclidean metric that captures the anisotropy produced by the characteristics of the terrain. We create the geometry from the graph representation by instancing generic procedural models.

We propose a constrained road generation framework that creates highways and primary and secondary road networks connecting a set of input cities, towns and villages. The paths of the roads are computed by minimizing the line integral of a cost function that takes into account, not only the characteristics of the terrain (the slope of the terrain as well as natural obstacles such as rivers, lakes, mountains or forests), but also the villages, town and city areas and previously created roads. This approach enables us to control the junctions between different types of roads and create a consistent hierachical road network (Section 4).

We present a parameterized proximity graph generation algorithm based on a non-Euclidean metric which enables us to select a restricted subset of roads from the complete graph connecting all input cities. Our method generates different layouts and controls the density of the network for highways and primary and secondary road (Section 5).

The resulting geometric graph often contains arcs with closely lying parts. We introduce an algorithm based on the evaluation of the non-Euclidean Fréchet distance, that selects and merges such cases and generates Steiner nodes in the graph for newly created nodes (Section 6).

The geometric models are created by analyzing the graph and applying selection rules that replace the edges as roads, tunnels or bridges. New nodes corresponding to Steiner nodes in the graph are instantiated as junctions or crossings according to the attributes stored at the edges and nodes of the graph (Section 7). Our procedural models automatically adapt their shape to the local geometry of the terrain and the models seamlessly match the terrain.

## 2. Workflow

Our system is a three-stage pipeline (Figure 3). First, terrains with water and vegetation maps are either procedurally generated, painted, or extracted from real data sets. Next, the user creates a set of cities on the terrain and defines their relative sizes. Additionally, the user can also control the network generation process either by painting an influence map or by using procedurally defined primitives that indicate which parts of the terrain should be privileged or avoided during the road network generation.

At the end of this step, our algorithm automatically generates a hierarchical road network as a graph whose arcs store the geometric paths of highways, major and minor roads connected together and connecting the cities (Figure 2). This

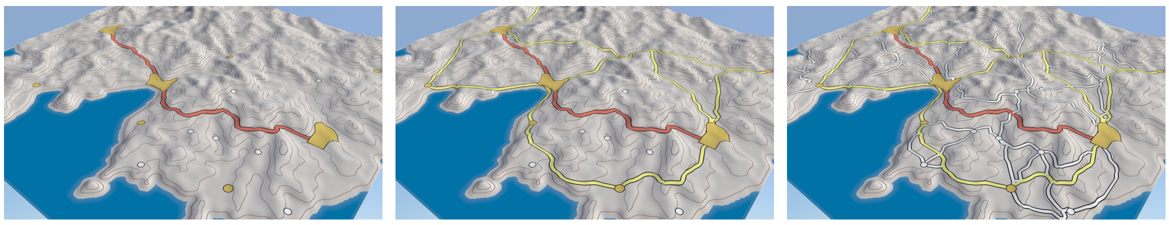

**Figure 2:** *Overview of the hierarchical road network generation process: starting from the highways, we incrementally compute the major and minor roads and merge them together.*

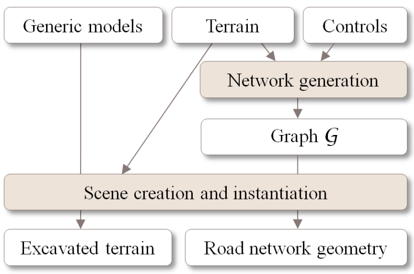

**Figure 3:** *Overview of the workflow of our method.*

graph is then used as input to our procedural modeling system to excavate the terrain along the paths of the roads and create three-dimensional geometry for roads, tunnels, bridges crossings and interchanges.

## 3. Overview and notations

We propose to model the road network as a three-level hierarchy: highways, major roads and minor roads. The road network is represented as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ denotes a set of nodes and $\mathcal{E}$ refers to the set of connecting edges. Nodes may be either cities or intersections, and edges represent the roads.

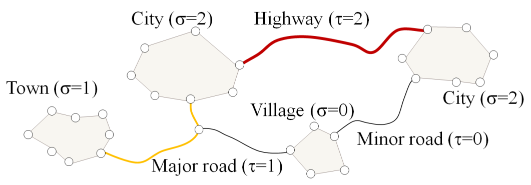

**Figure 4:** *Notations for cities and road edges with the corresponding $\sigma$ and $\tau$ attributes.*

Formally, nodes representing cities are defined by a two dimensional area $A$ representing the urban layout with buildings and houses (Figure 4). Cities are characterized by their relative size which will be denoted as $\sigma$. In our implementation, we define the following three different types: cities ($\sigma = 2$), towns ($\sigma = 1$) and villages ($\sigma = 0$). Without loss of generality, we determine $\sigma$ from the extent of the urban area and we linearly quantize it into the three categories.

An edge connecting two nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ will be denoted $\mathcal{E}_{ij}$. Edges have an attribute, denoted $\tau$, defining their type: highway ($\tau = 2$), major roads ($\tau = 1$) and minor roads ($\tau = 0$). In out system, the type of the road $\tau_{ij}$ connecting two nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ is determined from the size of the parameter $\sigma_i$ and $\sigma_j$: $\tau_{ij} = \min(\sigma_i, \sigma_j)$. Thus, highways connect cities, major roads connect cities and towns, and minor roads connect towns and villages. The characteristics of the roads, such as their width, markings, type and number of lanes, are derived from their type. The edges also store the path of the road, denoted as $\mathbf{p}$. Intersections, are characterized by their type, which ranges from highway interchanges to simple road intersections.

Note that our paper does not address the generation of the city street network. Even though several techniques have been proposed such as [CEW*08], the generation of a complete street network within the domain of a city $A_i$ and compatible with the connecting paths of the roads $\mathbf{p}_{ik}$ is a challenging problem beyond the scope of this paper.

A key observation is that the road patterns and layouts are mostly influenced by the proximity of the different cities and their relative size. The characteristics of the terrain, natural obstacles such as rivers or mountains have a major impact on the path of the roads connecting two cities. Therefore, the proximity graphs based on the Euclidean distance metric do not provide an effective measure to generate realistic network patterns (Figure 5).

Our paper focuses on the generation of a geometric graph connecting a set of points over a continuous domain using a non-Euclidean metric. We consider a compact domain $\Omega \subset \mathbf{R}^2$ and a direction dependent cost function $c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ that depends on the position $\mathbf{p}(t) : [0,1] \rightarrow \Omega$ along the trajectory and the first two derivatives denoted $\dot{\mathbf{p}}$ (speed) and $\ddot{\mathbf{p}}$ (acceleration) respectively [GPMG10]. Under the given constraints, the path between two points is a curve that minimizes the line integral of a cost-weighting function $c$ along the trajectory:

$$\mathcal{I}(\mathbf{p}) = \int_0^1 c(\mathbf{p}(t), \dot{\mathbf{p}}(t), \ddot{\mathbf{p}}(t))\, \mathrm{d}t$$

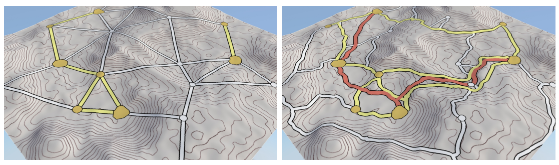

**Figure 5:** *Motivation for using a non-Euclidean metric: we compare a simple road network generated with a Gabriel Graph algorithm (left), and the network generated by our algorithm with a hierarchical network generation using a non-Euclidean metric (right). The non-Euclidean metric adapts better the road network to the terrain geometry.*

We propose a parameterized geometric graph generation algorithm based on this global geodesic metric to generate a spectrum of networks with a varying density of roads.

**Control cost functions** The paths of major and minor roads should be constrained by the highways so that paths do not overlap and do not have frequent intersections. In many cases, major roads should be redundant with toll highways. In contrast, minor roads should reuse the existing major road network as much as possible.

At every step of the hierarchical algorithm, we constrain the creation of the sub-graphs $\mathcal{G}_k$ by the shape of the paths of the previously created graphs $\mathcal{G}_i$, $0 \leq i < k$. Our approach consists in defining a control cost function, denoted $h$, which is used to locally modify the cost $c$ (Figures 9, 10). By defining a positive control cost function in the neighborhood of the previously computed road path, we limit the creation of new paths in the neighborhood of a previously created one.

## 4. Road network generation

Our hierarchical road network generation algorithm proceeds as follows (Figure 7): first, we initialize the graph $\mathcal{G}$ to an empty graph. Then, for all road types, starting from highways with decreasing importance (note that we also could use a bottom-up algorithm, leading to a different result), we perform the following steps:

1. Compute the control cost functions $h$ from the graph $\mathcal{G}$ and generate the road graph by computing the geodesic path for all the edges of the graph.
2. Generate the road layer $\mathcal{G}_k$ by computing the proximity graph (Section 5) $\mathcal{G}_k$ with the city subset $S_k$.
3. Merge the graph $\mathcal{G}$ with $\mathcal{G}_k$ by detecting which road paths in the two graphs should be merged and creating new intersection nodes if needed.

The first step generates a complete graph connecting all city nodes $\mathcal{N}_i$ by computing the geodesic paths between all pairs of nodes in the graph. For every edge, we compute the path of the road on the terrain and the modified cost function $c$ constrains the anisotropic shortest path generation process and prevent the highways from traversing towns or preserved areas such as national parks. Note that the geodesic path between two nodes is not guaranteed to be unique under the anisotropic metric. When several geodesics exist, we simply chose one of the candidate paths randomly. However, this case has never occurred in our experiments.

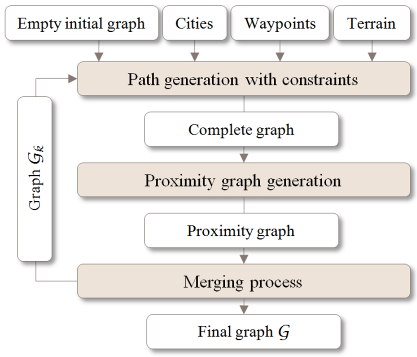

**Figure 7:** *Overview of the graph generation process.*

The second step computes the proximity graph $\mathcal{G}_k$ connecting the cities by employing an empty region criterion adapted to a direction dependent metric that defines which arcs should be preserved.

Finally, the third step cleans up the geometric graph $\mathcal{G}_k$. The previous step often generates arcs whose paths may have close parts (Figure 7). Having two roads going close to each other is not a casual case, therefore, we merge these parts by evaluating the non-Euclidean Fréchet distance to simplify the geometric graph. The results of this step are new nodes that represent road intersections.

### 4.1. Constrained road path generation algorithm

The road path connecting two cities is defined as the shortest anisotropic path that minimizes the line integral of the cost-weighting function $c$ along the path. Without loss of generality, the cost function $c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ is defined as the weighted sum of the following cost sub-functions: the density of vegetation cost $v(\mathbf{p})$, the water height cost $w(\mathbf{p})$, the slope of the terrain in the direction of the road cost $s(\mathbf{p}, \dot{\mathbf{p}})$, the curvature of the road cost $g(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ and the control cost function $h(\mathbf{p})$:

$$c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}) = v(\mathbf{p}) + w(\mathbf{p}) + s(\mathbf{p}, \dot{\mathbf{p}}) + g(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}) + h(\mathbf{p})$$

We use the cost functions for vegetation, water height, slope and curvature from [GPMG10]. Our contribution is the control cost function $h$ which can be arbitrarily defined to increase or decrease the overall cost over some regions of the domain $\Omega$. In our system, the control function $h$ only depends on the position $\mathbf{p}$. In the general case, $h$ could also take into account the direction $\dot{\mathbf{p}}$ and the acceleration $\ddot{\mathbf{p}}$.

By adding a positive cost $h(\mathbf{p}) > 0$ within a given region

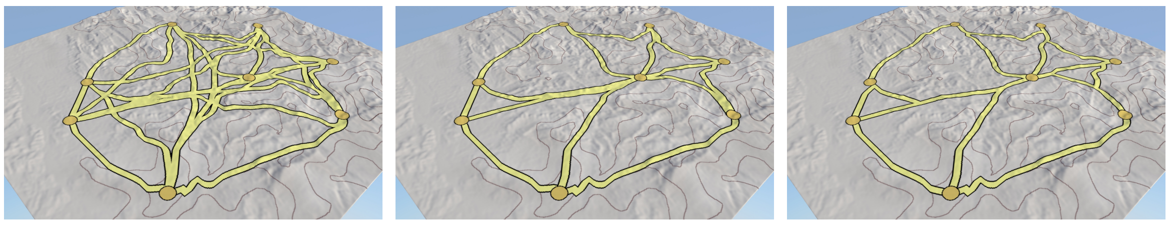

**Figure 6:** *Overview of the road network generation algorithm for a given road type: the complete graph connecting cities (left), the generated proximity graph (center) and the final road network obtained after merging the close parts of the different road paths (right).*

$\mathcal{R} \in \Omega$, we limit the road generation process within $\mathcal{R}$ since the shortest path algorithm tries to find the least cost path. In contrast, if $h(\mathbf{p}) < 0$, the cost of the line integral decreases when the path traverses the region $\mathcal{R}$ and we favor the road generation within $\mathcal{R}$. Note that $h(\mathbf{p})$ should not have high negative values however, so that the overall cost function $c(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ remains positive everywhere.

We define $h$ by summing the influence of a set of compactly supported primitives, denoted as $h_i$:

$$h(\mathbf{p}) = \sum_{i=0}^{i=n-1} h_i(\mathbf{p})$$

The primitives are defined as $h_i(\mathbf{p}) = f_i \circ d_i(\mathbf{p})$ where $d_i$ denotes the Euclidean distance to a skeleton and $f_i : \mathbf{R} \to \mathbf{R}$ is a smooth distance decreasing function with a compact support: $f_i(r) = 1 - (r^2/r_i^2)^2$ if $r < r_i$ and $f_i(r) = 0$ otherwise. $r_i$ will be referred to as the radius of influence of the primitive. In our system, primitives can be points, curves and compact areas defined by a closed non intersecting curve (Figure 8).

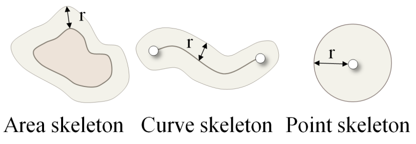

**Figure 8:** *Primitives used for generating the control cost function h.*

The control cost functions $h$ are used at two different steps of the road network generation pipeline. First, they provide the designer with a high level control to prevent generation of roads over some regions by painting a positive penalty cost over them. Another application in the automatic computation of a penalty cost in the neighborhood of an existing network to control the generation of new roads. Curve primitives define the control cost function for roads (using the road path as the skeleton) whereas compact areas primitives are for used to prevent roads from traversing cities (using the city region as skeleton).

## 4.2. Local control

The road generation process can be efficiently controlled by defining local regions limiting or favoring the generation of roads. An important feature is that different control cost function can be used during the hierarchical generation of the different types of roads. In our system, we use three specific parameterized functions with a decreasing radius of influence for highways, major and minor roads. Figure 9 shows an example in which we limit the number of intersections between the primary road network and the highway network by increasing the cost function $h(\mathbf{p})$ in the neighborhood of the previously generated highway paths.

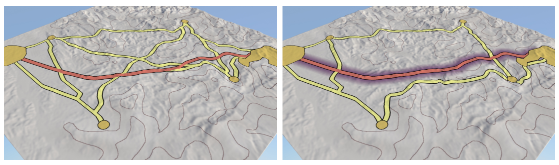

**Figure 9:** *Comparison between two networks, without (left) and with (right) limiting the generation of the major road network in the neighborhood of the highway.*

By constraining the creation of roads within some controlled bounded regions, our method can produce realistic road networks conforming to some land use policies. An example in Figure 10 shows that our method can generate a road network that does not traverse a park.

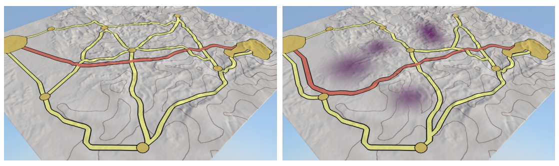

**Figure 10:** *Impact of an influence region over the road network generation process. The input model (left) is constrained by painting parks (right) and the road network automatically adapts to the new constraints.*

### 4.3. Waypoints

Waypoints are an efficient technique for controlling the road generation process. We define two different types of waypoints: persistent waypoints that have an impact over the proximity graph generation process, and road specific waypoints that constrain the generation of a specific road path.

Persistent waypoints are defined as virtual nodes of the graph and are characterized by an attribute that defines which kind of road may pass through them (Figure 11). The virtual nodes are added to the list of nodes $\mathcal{N}$ and processed like city nodes during the road generation process and also used later in the proximity graph generation step. At least one road will always pass through a persistent waypoint.

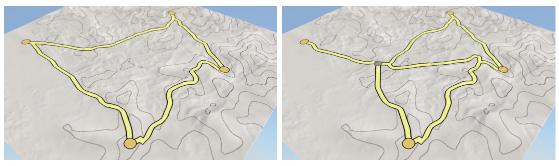

**Figure 11:** *Two different constrained road networks generated with a single persistent waypoint.*

In contrast, road specific waypoints are defined as control points $\mathbf{c}_k$, $k \in [0, n-1]$ attached to a given edge $\mathcal{E}_{ij}$ connecting two nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ prior to the proximity graph generation. When generating the road path of the edge $\mathcal{E}_{ij} = (\mathbf{a}, \mathbf{b})$, we generate $n+1$ connected sub-paths $(\mathbf{a}, \mathbf{c}_0)$, $(\mathbf{c}_i, \mathbf{c}_{i+1})$ with $i \in [0, n-2]$ and $(\mathbf{c}_{n-1}, \mathbf{b})$. The generated road can be discarded later by the proximity graph generation step if it does not meet the proximity criterion. Therefore, contrary to persistent waypoints, roads controlled with some specific waypoints might not appear in the final network. In our system, we allow the user to flag some roads so that they should be preserved, whatever the result of the proximity graph.

## 5. Graph generation algorithm

An important class of layout algorithms is based on Voronoï Diagrams [OBSC00] of (randomly) distributed points. Proximity graphs [JT92], also referred to as neighborhood graphs, are defined on a finite set of vertices in the plane such that there exists an edge between any two vertices if they are close in some sense. The proximity can be measured by the Euclidean distance between these vertices, the distance to other vertices of the graph, or the number of other vertices in a given neighborhood. Many definitions and algorithms have been provided, such as the relative neighborhood graphs [Lan69, Tou80], Gabriel graphs [GS69], β-Skeletons [KR85] and γ-neighborhood graphs [Vel91]. When the underlying metric is Euclidean those graphs generate line segments between the vertices. We propose an original proximity graph definition based on a non-Euclidean metric which lends itself for the road network generation problem.

Consider a compact region $\Omega \subset \mathbf{R}^2$ and a set of points $S$. We address the generation of a geometric skeleton connecting a set of points using a non-Euclidean metric, i.e., the computation of a graph such that the arcs connecting two points $\mathbf{a}$ and $\mathbf{b}$ should be defined as the continuous shortest path between those points that minimizes the line integral of a cost-weighting function along the path.

### 5.1. Fundamental concepts

Let us first recall some of the properties of the Gabriel graph [GS69] and relative neighbor graphs [Lan69, Tou80]. Let $S = \{\mathbf{p}_i\}$, $i \in [0, n]$ denote a set of points in a convex domain $\Omega \subset \mathbf{R}^2$. The neighborhood graph of $S$ is created from the definition of an influence region, denoted $\Omega(\mathbf{p}_i, \mathbf{p}_j)$, which is associated to candidate edges: $(\mathbf{p}_i, \mathbf{p}_j)$ forms an edge in the graph if and only if $\Omega(\mathbf{p}_i, \mathbf{p}_j) \cap S = \varnothing$. Note that the Gabriel graph is a sub-graph of the Delaunay triangulation since the Delaunay cell $\mathbf{p}_i\mathbf{p}_j\mathbf{p}_k$ is included in $\Omega(\mathbf{p}_i, \mathbf{p}_j) \cup \Omega(\mathbf{p}_i, \mathbf{p}_k) \cup \Omega(\mathbf{p}_j, \mathbf{p}_k)$.

The simplest graph connecting a set of points is the nearest neighbor graph which often yields an unconnected graph with many spatial subsets. The relative neighbor graph creates edges between two points $\mathbf{p}_i$ and $\mathbf{p}_j$ if the region of influence, referred to as lune, formed by the intersection of two discs of radius $\|\mathbf{p}_i\mathbf{p}_j\|$ and centered at $\mathbf{p}_i$ and $\mathbf{p}_j$ respectively is empty. Conceptually similar, the Gabriel graph links two points if the disc of diameter $\|\mathbf{p}_i\mathbf{p}_j\|$ and passing through $\mathbf{p}_i$ and $\mathbf{p}_j$ is empty (Figure 12).

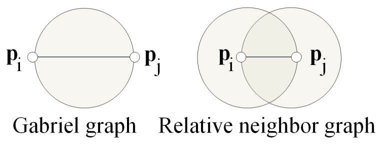

**Figure 12:** *Characterization of the regions $\Omega(\mathbf{p}_i, \mathbf{p}_j)$ for the Gabriel graph and relative neighbor graph.*

We propose a characterization of the neighborhood region of an arc $(\mathbf{a}, \mathbf{b})$ so as to create a spectrum of skeletons connecting the points in $S$ using a non-Euclidean metric. As in differential geometry, we define the metric over the domain $\Omega$ by using a global geodesic characterization: the distance between two points $\mathbf{a}$ and $\mathbf{b}$ and denoted as $d(\mathbf{a}, \mathbf{b})$ is equal to the length of the path $\mathcal{I}(\mathbf{p}(\mathbf{a}, \mathbf{b}))$ which minimizes the line integral of the cost function along the path.

We define the ball centered at a given point $\mathbf{a}$ with radius $r$ as the set of points $\mathbf{p}$ in $\Omega$ such that the length of the geodesic between $\mathbf{a}$ and $\mathbf{p}$ is less than $r$ (Figure 13):

$$\mathcal{B}(\mathbf{a}, r) = \{\mathbf{p} \in \Omega, d(\mathbf{a}, \mathbf{p}) < r\}$$

Let $\mathbf{a}$ and $\mathbf{b}$ two points in $\Omega$ and $(\mathbf{a}, \mathbf{b})$ the geodesic path

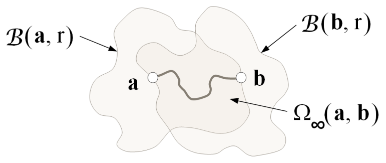
$\mathcal{B}(\mathbf{a}, r)$
$\mathcal{B}(\mathbf{b}, r)$
a
b
$\Omega_\infty(\mathbf{a}, \mathbf{b})$

**Figure 13:** *Geometric representation of* $\Omega_\infty(\mathbf{a},\mathbf{b})$.

between $\mathbf{a}$ and $\mathbf{b}$. Let $\gamma \in [1,+\infty[$, we define the parameterized neighborhood region of $(\mathbf{a},\mathbf{b})$ as:

$$\Omega_\gamma(\mathbf{a},\mathbf{b}) = \left\{\mathbf{p} \in \Omega,\, d(\mathbf{a},\mathbf{b})^\gamma < d(\mathbf{a},\mathbf{p})^\gamma + d(\mathbf{b},\mathbf{p})^\gamma\right\}$$

Note that as $\gamma \to \infty$, the definition is equivalent to the relative neighbor graph:

$$\Omega_\infty(\mathbf{a},\mathbf{b}) = \left\{\mathbf{p} \in \Omega,\, d(\mathbf{a},\mathbf{b}) < \max(d(\mathbf{a},\mathbf{p}), d(\mathbf{b},\mathbf{p}))\right\}$$

From a geometrical point of view, $\Omega_\infty(\mathbf{a},\mathbf{b})$ can be characterized as the intersection of the two balls $\mathcal{B}(\mathbf{a}, d(\mathbf{a},\mathbf{b}))$ and $\mathcal{B}(\mathbf{b}, d(\mathbf{a},\mathbf{b}))$ (Figure 13). When $\gamma$ decreases, the neighborhood region $\Omega_\gamma(\mathbf{a},\mathbf{b})$ smoothly shrinks and the number of edges in the resulting geometric graph increases (Figure 14). When $\gamma = 2$, the definition of $\Omega_2$ creates the same graph as the Gabriel graph created with a non-Euclidean metric:

$$\Omega_2(\mathbf{a},\mathbf{b}) = \left\{\mathbf{p} \in \Omega,\, d(\mathbf{a},\mathbf{b})^2 < d(\mathbf{a},\mathbf{p})^2 + d(\mathbf{b},\mathbf{p})^2\right\}$$

## 5.2. Time complexity of the algorithms

In Euclidean space, the Gabriel and the relative neighborhood graphs are subgraphs of the Delaunay triangulation, therefore they can be computed in $O(n \ln n)$ time. Because we rely on a non-Euclidean metric, our proximity graph cannot be computed as efficiently as Gabriel or relative neighbor graphs. Given a set of points $\mathbf{p}_k$, $k \in [0, n-1]$, we need to first compute the paths of all the arcs $\mathcal{E}_{ij}$ using the anisotropic shortest path algorithm. Then, we can filter the edges by applying the empty region criterion. Therefore, the overall complexity of our algorithm would be $O(n^3)$.

In practice, the topography of the terrain is usually sufficiently regular to avoid the expensive computation of all the paths connecting all cities. Therefore, we use a heuristic, in which for every city, we only compute the paths connecting the city with a restricted subset of neighboring cities. In our experiments, using the 20 nearest neighboring cities proved to be enough to yield the same proximity graph as the one produced with the initial complete graph.

## 5.3. Global control

Our parameterized definition of the proximity graph enables the user to globally control the overall road density of the network by tuning the exponent $\gamma$ in the definition of the neighborhood of a road path $\Omega(\mathbf{a},\mathbf{b})$.

When $\gamma \in [1,2]$, the resulting road network is dense and has many redundant roads connecting different cities. As $\gamma$ increases, more edges are removed in the graph generation process and the road network simplifies (Figure 14).

| | Highways | | Major roads | | Minor roads | |
|---|---|---|---|---|---|---|
| $\gamma$ | $n$ | Length | $n$ | Length | $n$ | Length |
| 1.2 | 3 | 194 | 19 | 700 | 70 | 269 |
| 2.0 | 2 | 96 | 11 | 383 | 39 | 371 |
| 8.0 | 2 | 96 | 9 | 290 | 29 | 368 |

**Table 1:** *Road network statistics for some* $\gamma$ *values*

Table 1 reports the corresponding number of highways, major and minor roads as well as the length of the corresponding sub-networks for a set of increasing $\gamma$ values. The network with the fewer arcs is obtained when $\gamma \to \infty$, which corresponds to a relative neighbor graph generation.

## 6. Path merging

The proximity graph generation algorithm often yields to a set of paths $\mathcal{P}_{ij}$ with parts that are mutually close (Figure 15). The path merging step aims at merging some parts of the paths by inserting Steiner points in the graph which will be instantiated as intersections and interchanges.

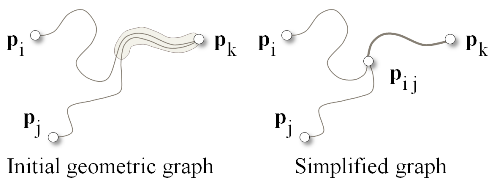

**Figure 15:** *Two road paths* $(\mathbf{p}_i, \mathbf{p}_j)$ *and* $(\mathbf{p}_i, \mathbf{p}_k)$ *partially merging and resulting in a new Steiner point.*

The road merging process proceeds as follows. For all pairs of arcs in the graph $\mathcal{G}_k$, we compute the Fréchet distance [AG92] between the paths to identify which parts should be merged.

Recall that a curve in $\mathbf{R}^2$ can be represented as a parametric function $f : [0,1] \to \mathbf{R}^2$. A monotone reparameterization $\alpha$ is a continuous non-decreasing function $\alpha : [0,1] \to [0,1]$ with $\alpha(0) = 0$ and $\alpha(1) = 1$. A matching between $f$ and $g$ is a pair of monotone reparameterizations $(\alpha, \beta)$ of $f$ and $g$ respectively, where the point $f(\alpha(x))$ is matched to the point $g(\beta(x))$, for any $x \in [0,1]$. Let $\alpha$ and $\beta$ range over all monotone re parameterizations. Given two curves $f$ and $g$, their Fréchet distance under the Euclidean norm is defined as:

$$d_\mathcal{F}(f,g) = \inf_{\alpha,\beta} \max_{t \in [0,1]} d\Big(f(\alpha(t)), g(\beta(t))\Big)$$

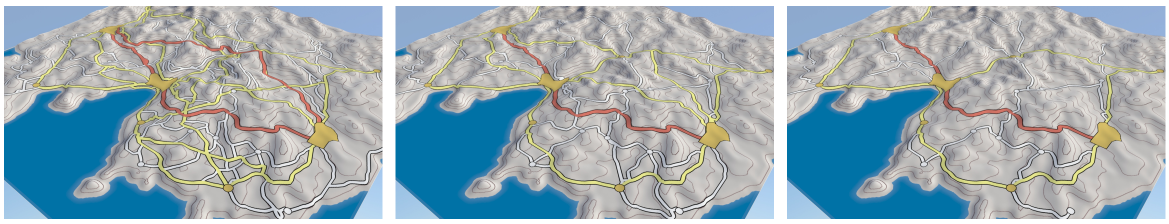

**Figure 14:** *Different road networks connecting* 3 *cities,* 6 *towns and* 16 *villages: dense* $\gamma = 1.2$ *(left), average* $\gamma = 2.0$ *(middle) and sparse* $\gamma = 8.0$ *(right). The networks were created on a* $128 \times 128\,\mathrm{km}^2$ *terrain discretized into* $256 \times 256$ *resolution.*

Computing the exact Fréchet distance in general is a complex problem and even for simple polygonal curves, it requires algorithms using parametric search techniques [AG92]. When two curves are embedded in a more complex metric space, the distance between two points on the curves is most naturally defined as the length of the shortest path between them. The definition of geodesic Fréchet distance allows the distance to switch discontinuously, without penalty, from one side of an obstacle or a mountain to another.

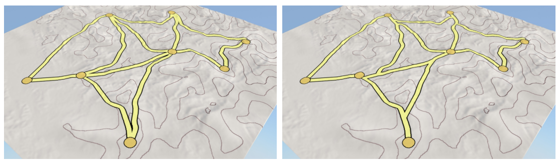

**Figure 16:** *Simplification process showing an initial network (left) and the resulting simplified network (right).*

We approximate the Fréchet distance by its discrete variation from [EM94] applied to the polygonal curves produced by the anisotropic shortest path [GPMG10]. The variation is called the coupling distance $\delta_{\mathcal{F}}$ and can be computed efficiently by looking at all possible couplings between the end points of the line segments of the polygonal curves. The distance $\delta_{\mathcal{F}}$ provides a good approximation of the Fréchet distance: it is an upper bound for $d_{\mathcal{F}}$, and the difference between these measures is bounded by the length of the longest edge of the polygonal curves [EM94].

Different strategies may be applied to decide whether roads should be merged. In our system, primary and secondary roads do not merge with toll highways, but can merge with toll free highways. In contrast, secondary roads can always merge with primary roads (Figure 16). The type $\tau_{ij}$ of a road obtained by merging two arguments roads is defined as $\tau_{ij} = \max(\tau_i, \tau_j)$.

## 7. Geometry instantiation

Our approach for generating the geometric models follows the outline presented in [GPMG10]. In this section, we present the several improvements and the specific steps that are required for handling a whole complex road network. The main challenge stems from the fact that geometric models should seamlessly match together and with the terrain.

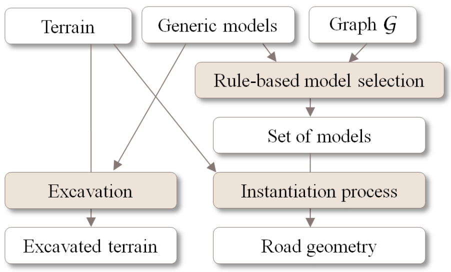

**Figure 17:** *Overview of the instantiation process. The inputs are the terrain and the set of generic models that are adapted into the scene using the graph* $\mathcal{G}$.

We use a set of procedurally defined parameterized models, denoted $\mathcal{M}$, representing roads, bridges, tunnels as well as junctions, and a set of selection rules that define which models should be instantiated according to the information carried by the road network graph (Figure 17).

A geometric model $\mathcal{M}$ is a three-dimensional object oriented model characterized by a procedurally defined support region, denoted as $\Omega_{\mathcal{M}}$, and two specific methods for excavating the terrain and generating its geometry.

We developed two different types of generic procedural models: road models (including bridges and tunnels) that correspond to the edges $\mathcal{E}$ and junction models (including crossings and interchanges) corresponding to the junction nodes $\mathcal{N}$ added to the graph during the merging process (Figure 18). The instantiation of the city nodes $\mathcal{N}$ is beyond the scope of the paper and could be achieved by previously published methods such as combining procedural street generation techniques [CEW*08] with shape grammar based building generation methods [MWH*06] or more recently in [KK11].

Recall that edges representing the roads are defined by their type $\tau_{ij}$, which stores the attributes about their geometric characteristics such as road width or number of lanes.

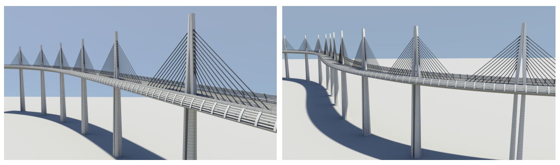

**Figure 18:** *Two different instances of the same generic bridge model with different parameters.*

Similarly, the nodes representing junctions are characterized by the type of their neighboring edges. The overall algorithm for generating the geometry proceeds in three steps:

1. For all the nodes $\mathcal{N}_k$ and the edges $\mathcal{E}_{ij}$ of graph $\mathcal{G}$, apply a set of selections rules to define their corresponding model $\mathcal{M}(\tau_k)$ and $\mathcal{M}(\tau_{ij})$ respectively. This step selects and generates a set of generic procedural models denoted as $\mathcal{M}_k$ and $\mathcal{M}_{ij}$.
2. Create the excavations and the embankments on the different regions of the terrain defined by their support region $\Omega_{\mathcal{M}_k}$
3. Finally, generate the geometry of every model $\mathcal{M}_k$ according to its own attributes and to the geometry of the terrain inside the support region $\Omega_{\mathcal{M}_k}$.

## 8. Results

Our algorithm has been implemented in C++ and runs on a Core *i*7 computer clocked at 3 GHz with 8 Gb of memory. The images of the road networks presented throughout this paper were generated by applying our algorithm to different input scenes featuring cities, towns and villages. The realistic images (Figures 1, 20) were created by generating the textured mesh models of the roads, tunnels and bridges as well as the different junctions using procedurally defined models. Those generic models were instantiated with the parameters computed during the road generation process and stored in the graph. The renderings were performed with Mental Ray.

Our method can create a wide variety of scenes with highly varying models that resemble realistic scenes. A fundamental feature of our method is the road merging step which automatically creates junctions outside of cities, which plays an important part in the overall realism.

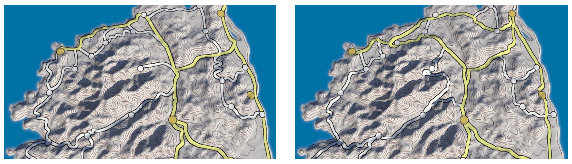

**Figure 19:** *Comparison between the real (left) and synthesized (right) road networks of Corsica.*

We carried out several experiments to compare our synthesized road networks with real ones. Figure 19 shows a comparison of the global road network of Corsica. The synthetic network was generated automatically from a set of 10 cities and 39 towns (we removed the villages for clarity). The coefficient $\lambda$ was found after a few trial and error steps. Figure 19 shows that even without waypoints, our algorithm produces very similar hierarchical layouts. The differences between our results and reality often occur in urbanized regions where some roads were created to connect many small villages and towns along the coast, probably because of the fact that a purely geometric algorithm does not take into account land use policies and infrastructure development. By inserting a few persistent waypoints in the map, we can generate a road network which closely conforms to reality.

| Scene | Grid size | Time | Length |
|---|---|---|---|
| Sea shore | $128 \times 128$ | 23.2 | 4 640 |
| Mountains | $256 \times 256$ | 112.7 | 5 248 |
| Foothills | $512 \times 512$ | 625.3 | 5 206 |

**Table 2:** *Time (in seconds) and total network length (in kilometers) on a $200 \times 200\,\text{km}^2$ terrain discretized at different resolution (4 cities, 16 towns and 70 villages).*

An interesting and powerful feature of our approach is its simplicity and control. The generation process can be controlled in several ways. The trajectories of the different roads can be intuitively controlled by modifying the parameters of the cost functions. The global road density can be adapted by tuning the coefficient $\gamma$ in the geometric graph generation, whereas the road network pattern can be controlled by defining arbitrary constraints over the terrain.

| Scene | Cities | Towns | Villages | Time |
|---|---|---|---|---|
| Sea shore | 3 | 10 | 37 | 89.1 |
| Mountain | 5 | 45 | 97 | 119.2 |
| Foothill | 10 | 107 | 204 | 146.8 |

**Table 3:** *Number of cities, towns and villages, time (in seconds) for a grid size of $256 \times 256$.*

The most computationally demanding part of our algorithm is the anisotropic shortest path generation and the overall performance mostly depends on the resolution of the underlying sampling grid that is used for computing the road paths of the initial complete graph (Table 2). Increasing the grid size to get a better accuracy results in a more expensive path generation process, which creates more detailed paths with more control points, which in turn results in a more computationally demanding road merging step. Nevertheless, our method can produce large road networks within times ranging from a few seconds to a couple of minutes (Table 3).

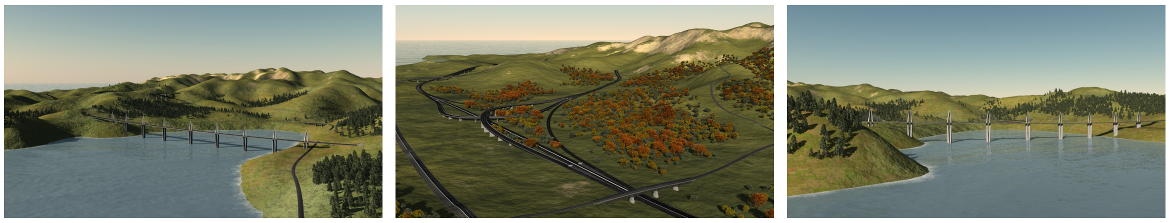

**Figure 20:** *Some synthetic sceneries generated by our system.*

## 9. Conclusion

We have introduced a procedural framework for generating a hierachical road network with highways, major and minor roads connecting cities, towns and villages. The overall road network pattern is obtained by combining different geometric graphs created using a non-Euclidean metric that reflects the anisotropy produced by the terrain as well as the natural obstacles such as rivers or mountains. The geometry of the paths of the different road types can be easily controlled by adjusting the parameters of the cost function which weight the relative influences of the terrain characteristics.

This work opens several research problems worth investigating. We would like to put the emphasis on the generation of the rural-urban fringe which is a particularly challenging research topic. While several techniques exist for generating large cities with complex road networks, the generation of the region at the interface between the interior and the exterior of a city remains a difficult problem. In particular, there is a need for specific techniques for generating non planar graphs that would simulate the highway and free way networks around cities. In terms of geometry, this would require the development of specific parameterized models for generating stacks of interchanges.

## References

[AG92] ALT H., GODAU M.: Measuring the resemblance of polygonal curves. In *Symposium on Computational Geometry* (1992), pp. 102–109. 7, 8

[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, 3, 8

[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

[EM94] EITER T., MANNILA H.: Computing discrete Fréchet distance. In *Technical Report CD-TR* 94/64, *TU Vienna, Austria* (1994). 8

[GPMG10] GALIN E., PEYTAVIE A., MARÉCHAL N., GUERIN E.: Procedural generation of roads. *Computer Graphics Forum (Proceedings Eurographics) 29*, 2 (2010), 429–438. 1, 2, 3, 4, 8

[GS69] GABRIEL K., SOKAL R.: A new statistical approach to geographic variation analysis. *Systematic Zoology 18*, 3 (1969), 259–270. 6

[JT92] JAROMCZYK J., TOUSSAINT G.: Relative neighborhood graphs and their relatives. *Proceedings of IEEE 80*, 9 (1992), 1502–1517. 6

[KK11] KRECKLAU L., KOBBELT L.: Procedural modeling of interconnected structures. *Computer Graphics Forum (Proceedings Eurographics) 30*, 2 (2011), 335–344. 8

[KR85] KIRKPATRICK D. G., RADKE J. D.: Framework for computational morphology. *Computational Geometry* (1985), 217–248. 6

[Lan69] LANKFORD P. M.: Regionalization: Theory and alternative algorithms. *Geographical Analysis 1* (1969), 169–212. 6

[LSWW11] LIPP M., SCHERZER D., WONKA P., WIMMER M.: Interactive modeling of city layouts using layers of procedural content. *Computer Graphics Forum (Proceedings Eurographics) 30*, 2 (2011), 345–354. 1

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

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. In *Proceedings of SIGGRAPH* (2006), pp. 614–623. 1, 8

[OBSC00] OKABE A., BOOTS B., SUGIHARA K., CHIU S. N.: *Concepts and Application of Voronoï Diagrams*. John Wiley, 2000. 6

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

[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

[Tou80] TOUSSAINT G. T.: The relative neighbourhood graph of a finite planar set. *Pattern Recognition 12* (1980), 261–268. 6

[VABW09] VANEGAS C., ALIAGA D., BENEŠ 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. 1, 2

[Vel91] VELTKAMP R.: The gamma-neighborhood graph. *Computational Geometry 1* (1991), 227–246. 6

[WMWG09] WEBER B., MÜLLER P., WONKA P., GROSS M.: Interactive geometric simulation of 4d cities. *Computuer Graphics Forum 28*, 2 (2009), 481–492. 1