Temporally Coherent Real-Time Labeling of Dynamic Scenes

Mikael Vaaraniemi
BMW Research and Technology GmbH
Munich, Germany
mikael.vaaraniemi@bmw.de

Marc Treib
Computer Graphics and Visualization Group
Technische Universität München, Germany
treib@tum.de

Rüdiger Westermann
Computer Graphics and Visualization Group
Technische Universität München, Germany
westermann@tum.de

ABSTRACT

The augmentation of objects by textual annotations provides a powerful means for visual data exploration. Especially in interactive scenarios, where the view on the objects and, thus, the preferred placement of annotations changes continually, efficient labeling procedures are required. As identified by a preliminary study for this paper, these procedures have to consider a number of requirements for achieving an optimal readability, e.g. cartographic principles, visual association and temporal coherence. In this paper, we present a force-based labeling algorithm for 2D and 3D scenes, which can compute the placements of annotations at very high speed and fulfills the identified requirements. The efficient labeling of several hundred annotations is achieved by computing their layout in parallel on the GPU. This allows for a real-time and collision-free arrangement of both dynamically changing and static information. We demonstrate that our method supports a large variety of applications, e.g. geographical information systems, automotive navigation systems, and scientific or information visualization systems. We conclude the paper with an expert study which confirms the enhancements brought by our algorithm with respect to visual association and readability.

Categories and Subject Descriptors

I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human information processing*

General Terms

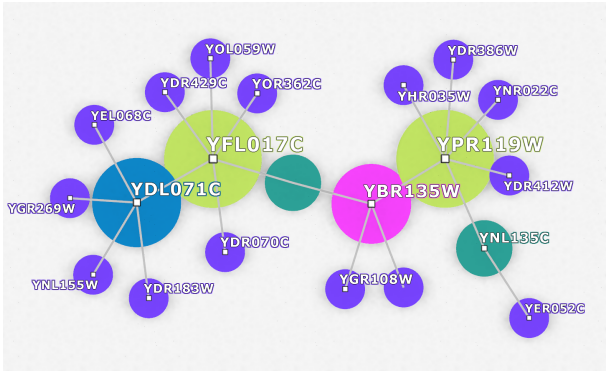
Algorithms, Design, Performance

Keywords

Labeling, force-based, real-time, annotation, graph, GIS, navigation.

1. INTRODUCTION

Information can be represented by images and textual annotations. Images give a quick overview and portray data intuitively. In contrast, textual annotations have to be actively read. However, they can precisely describe objects with selected information. Combining both types creates a very powerful tool for data exploration. These principles are used in the fields of information visualization and visual analytics. Therein, abstract data is often described by textual annotations. For instance, graphs with up to several hundred nodes have to be labeled (see Figs. 1(a) and 9). Another area of research focuses on geographic information systems (GIS). These systems visualize geospatial data such as road networks, demographic data, and their annotations (see Figs. 1(b) and 6(b)). Such systems, especially when used on large-scale display systems, must be able to provide interactive visual exploration of geospatial data with a very high number of annotations. Since in such 3D (or 2.5D) applications the camera can be moved freely, the use of unconstrained labels and the preprocessing of every possible layout constellation to avoid collisions is impractical [18]. More and more such applications have to cope with dynamic content, requiring placements of labels without prior knowledge, e.g. when loading KML files into Google Earth. Therefore, real-time computation of annotation layouts is becoming an ever important requirement in interactive data exploration. Besides interactivity, additional requirements such as Imhof's cartographic principles [13] have to be considered to improve the visual analysis process. Adhering to these principles, including readability, visual association, and classification, is extremely challenging and demands for a proper integration of the respective functionality into label placement algorithms, e.g. support for scalable rotated labels with priorities. In addition, only a stable and consistent (similar to [2]), as well as temporally coherent layout, where labels do not jitter, appear suddenly, or move unexpectedly, lets users easily track annotations in a complex scene. Labels should make slow-paced, smooth transitions to minimize distraction. Therefore, on top of the aforementioned requirements, the maintenance of a frame-coherent presentation is another major concern underlying our developments. In many applications the demand for a temporally coherent layout even supersedes the requirement for an optimal position in every frame. Based on our intended application areas and a preliminary study (see section 3), we define the following goals:



(a) Annotation of a large graph network.

(b) Labeling a GIS in Geneva, Switzerland.

Figure 1: Real-time labeling of 2D (left) and 3D (right) scenes using our force-based approach.

- Real-time labeling of point, line and area features
- Runtime placement of new labels
- Scalability to up to several hundreds of labels
- Temporal coherence and stability of the labeling layout
- Consideration of Imhof's cartographic rules (e.g. readability, visual association and classification)
- Support for rotated and scaled labels with priorities

Most existing real-time approaches follow the first two rules but do not scale well with the number of labels, have problems maintaining temporal coherence, and do not rigorously adhere to common cartographic rules (see section 2).

The major contribution of this paper is an efficient algorithm for creating a temporally coherent labeling of 3D or 2.5D objects and scenes. Our algorithm is specifically tailored to the massively parallel multi-threading architecture of graphics processing units (GPUs). Every label collision is detected with the separating axis theorem computed in parallel on the GPU. Collisions between labels are resolved with a force-based approach thus creating smooth changing label positions. In this way we can create layouts for several hundred annotations in real-time. Furthermore, our method supports scaled and rotated annotations.

The remainder of this paper is organized as follows: First, we review related work in section 2. In section 3, we present the findings of a preliminary study and define design principles for achieving an optimal readability of labels. In section 4, we introduce our real-time, force-based labeling approach with implementation details in section 5. We conclude the paper with performance measurements and a validation of our real-time prototype in an expert study.

2. RELATED WORK

Cartographic principles Imhof [13] names legibility and the graphical association of a label with its feature as characteristics of good lettering. Furthermore, he emphasizes the importance of minimal map disturbance, good label distribution, as well as fonts which indicate the spatial properties of features and their classification. He divides the labeling problem into three categories: labeling point features, line features and area features. For each category certain cartographic principles apply. Yoeli's four-position model [31] depicted in Fig. 3(a) ranks each possible position of a label around a point feature according to its degree of intuitive association. He already mentions the semi-automatic labeling of point features. Its computational complexity is NP-complete [17]. Christensen et al. [5] introduce the point-feature-labeling-problem (PFLP) and prove that it is NP-hard. Thus, heuristics are needed to label maps with a huge number of features. A broad range of strategies has been developed, for instance exhaustive rule-based, genetic, force-based or greedy approaches. A collection of related papers can be found in the bibliography by Wolff and Strijk [29].

Force-based A force-based approach was first presented by Hirsch [12]. He uses a gradient-driven heuristic to label point features. Labels are placed on a circle around their corresponding features. To resolve conflicts, vectors between overlapping labels are computed based on the intersection area. As a single label can collide with multiple other annotations, the sum of all its vectors guides its movement, thus improving the global labeling layout. Feigenbaum [9] describes a similar method to automatically annotate point features. He resolves collisions and places labels close to their anchor by a force-based approach. Attractive forces pull labels to their point feature, while repelling forces push labels away from other features. Thus, step by step, the map layout converges towards a final labeling state through a gradient descent method. Ebner et al. [7, 8] enhance this approach by simulated annealing. Consequently, local minima which usually prevail in greedy force-based methods can be avoided. Similar to the approaches of Hirsch and Feigenbaum, this technique approximates the intricate form of a label's lettering by an axis-aligned rectangle. A hybrid approach is presented by Stadler et al. [25]. They obtain an initial placement with the help of image processing. Then, iterative forces improve the chosen labeling positions. They note that force-based approaches achieve a clear label distribution and thus an aesthetical layout. However, this method does not allow for real-time placement of labels.

2D Real-time Full interaction with 2D GIS environments, which includes panning and zooming, requires dynamic map labeling. In the last decade, several real-time algorithms for 2D maps have appeared. They divide the labeling problem into a pre-processing and an interactive real-time phase [2,

18, 21, 30]. In the former, a conflict graph is generated which stores every possible conflict between labels at every possible scale. At runtime, relevant labeling positions are chosen based on the precomputed graph. Unfortunately, all these approaches only work for a top-down view and require uniformly sized, axis-aligned labels. Recently, real-time labeling of 2D maps without pre-processing has been achieved by Luboschick et al. [15, 6]. They divide the available screen space into a uniform grid wherein "conflict particles" indicate if a region (cell) is occupied. Choosing a labeling position then becomes a search for free cells. This approach uses existing position-models [31] sequentially to determine the first valid position. By using particles, this method can freely define obstacles and is not restricted to axis-aligned annotations. The layout is computed from scratch every frame. Temporal coherence is achieved by interpolating the resulting positions. However, during animation, occlusion of labels can occur. In this approach, most of Imhof's cartographic principles [13] are not addressed, even though positioning models are used.

3D Real-time Bell et al. present a real-time approach for labeling a 3D virtual world [3]. They introduce the notion of temporal continuity in a lettering. This avoids popping of annotations while panning or zooming the map. Labels are projected from 3D world space into 2D screen space. Placement is done by iterating through a set of rectangles describing unoccupied screen space. Thus, the performance of this method does not scale well and labels can only be approximated by axis-aligned rectangles. A simpler approach from Maass and Döllner [16] splits the screen space into disjoint vertical slots. Therein, labels are stacked sorted by their distance to the viewer. A line connects a feature and its label visually. A bold font with a halo makes textual annotations easily readable. However, this approach creates dense clusters of labels and long connecting lines. Thus, visual association becomes impossible. Several approaches deal with the labeling of single 3D objects for illustration purposes [1]. Göetzelmann et al. [11] describe an algorithm which uses distance fields to compute ideal labeling positions and employ agents to preserve temporal coherency. Unfortunately, during camera movements labels are hidden and jittering occurs. Stein and Décoret [26] present a GPU-based real-time approach for labeling a 3D scenery. To compensate the drawbacks of their greedy approach, an Appolonius diagram defines the label placement order. Similar to Stadler et al. [25], they use image processing to determine the initial positions. Unfortunately, only up to 20 labels can be placed at interactive framerates. Furthermore, during navigation, jittering and harsh changes in labeling positions occur. Finally, a real-time approach for annotation of virtual reality environments is described by Pick et al. [22]. They choose the initial position of a label by voxelizing the corresponding object, extracting a medial line and choosing the closest point on this line to the object's center of gravity. Unfortunately, this can lead to collisions at placement and thus unstable layouts. To resolve conflicts at runtime, they first compute a visibility volume from an object's axis-aligned bounding box. The intersection between these volumes on directed 2D planes creates a force vector which depends on the penetration depth. This force is applied at runtime on the involved labels. However, real-time labeling can only be achieved for 20-40 annotations.

Summary None of the presented approaches satisfy all the requirements stated in section 1. Several real-time algorithms suffer from visual frame-to-frame discontinuities and create too much movement in the labeling layout [16, 26]. Others only follow a minor set of Imhof's cartographic principles [6, 15, 16] or exhibit an unstructured layout [6, 16, 22]. The performance of some approaches does not scale well [3, 26, 22]. Finally, some algorithms only allow axis-aligned annotations [3, 18, 16, 21, 26, 22].

3. PRELIMINARY STUDY

A good readability is one of the most important goal of our approach. In order to define design principles, we conducted a preliminary expert study at our research facility. As a representative application, we chose the labeling of a GIS with 3D-terrain overlayed by orthoimages and a road network. We interviewed one psychologist, who has been working as a researcher in the human-machine interaction (HMI) design field for over two decades, four engineers, three of which work as project leaders for navigation components, the fourth is developing HMI concepts, and a sixth expert who has worked for over a decade as a visual designer.

3.1 Study Design

In an interview of approximately one hour we presented different labeling concepts (see Fig. 2), all adhering to Imhofs cartographic principles [13], and we provided the following questionnaire to the experts. Firstly, we asked whether the size of a label should change with respect to its depth position in a 3D landscape. Secondly, we surveyed how to annotate point features (e.g. cities): using the four-position model, centered above their anchor, or circling around its feature. Finally, we questioned the best strategy to label line features (e.g. streets): horizontally (see Fig. 2(a)), as rotated straight labels (see Fig. 2(b)), or by following the line features (see Fig. 2(c)).

3.2 Results

Depth Scaling This concept categorizes the subjects into two groups. The first group (with 4 of 6 candidates) stated that depth scaling helped spatial perception in the 3D landscape. The second group (2 of 6) stated it is acceptable if textual annotations remain readable.

Annotation of Point Features Firstly, we asked which point labeling concept is appropriate. The majority (4 of 6) stated that a consistent approach was the most important property. Three of the candidates suggested that the labeling concept should create a clear layout and give a good overview of the situation. Finally, one of them chose the 4-position model as the best concept because it incorporated all these stated requirements.

Annotation of Line Features The concept of horizontally placed labels again splits the candidates into two groups. The first group (4 of 6) qualified it as very readable but with a higher search and visual association time. The second group (2 of 6) did not like this kind of placement. One stated reason was that a horizontal label could occlude neighbouring features. The concept of straight, rotated annotations was selected as the best alternative by almost every subject (5 of 6) as it provides a good compromise between readability and visual association to the line feature. In the

(a) Horizontal annotations. (b) Straight, screen space rotated annotations. (c) Annotations following the roads curvature.

Figure 2: Static images for the conducted expert study: each concept shows a different annotation style.

last concept, annotations follow the curvature of the corresponding line. The majority (5 of 6) liked the appearance, but questioned if it would remain readable under special circumstances (e.g. roads with tight turns). Furthermore, this group stated that this concept helps understanding the layout of a road. Some of them (3 of 6) indicated the good visual association. Finally, some (2 of 6) said that they did not see much difference compared to straight annotations.

3.3 Design Principles

The first conclusion of our study is that scaling annotations by their depth helps spatial perception in a 3D landscape. However, the scale factor should not go below a certain minimum to maintain readable labels. Hence, the first requirement for the layouting algorithm is freely scalable annotations. Secondly, labeling point features should create a clear layout and give a good overview of the situation. Therefore, we choose the four-position model because of its clear and consistent labeling. The best compromise between readability and visual association in most situations are provided with straight, rotated labels. Only at very steep angles, where the readability would be compromised significantly, we switch to horizontal labels. Thus, we concluded on the support for rotated annotations as our last requirement.

4. FORCE-BASED LABELING

In this section, we introduce our force-based, real-time labeling approach for dynamic scenes. By providing a temporally coherent layout, at the same time considering the additional requirements identified in the expert study, optimal readability of annotations is enforced in an interactive environment. However, achieving this for a huge number of labels is extremely challenging, since at runtime initial label positions have to be computed (see section 4.3), collisions have to be detected (see section 4.4), and these collision have to be resolved in a temporally coherent manner (see section 4.5).

4.1 Motivation

According to Chen et al. [4], naive search tasks are completed in less time when labels are displayed in screen space. Hence, to achieve an optimal readability and a more efficient visual exploration we handle the entire layouting in screen space. It is worth noting that this strategy also facilitates an efficient computation of conflict situation between different labels [3, 15, 26]. Textual labels and icons are represented by 2D object-oriented bounding boxes (OOBBs) to accelerate the computation of possible collisions via the separating axis theorem [24] (see section 4.4). Furthermore, OOBBs enable us to efficiently approximate rotated labels needed for line aligned annotations. For the initial placement of point features, we use the four-position model (see section 4.3). As shown in our preliminary study, and also stated by Yoeli [31], this model helps creating a visual association between a feature and its annotation. While browsing through annotated datasets, the addition and removal of labels changes the optimal labeling layout, such that in every frame a completely new arrangement might be required to again achieve optimality. These frame-to-frame changes lead to jittering effects and abrupt re-layouting in several existing approaches [15, 26]. Therefore, we do not create a discrete optimum arrangement in each frame. Instead, the current layout is always based on the labeling result of the previous frame. This is achieved by using a force-based approach which only allows continuous changes to the layout and thus, creates a temporally coherent labeling. Additionally, this results in an appropriate label distribution and avoids clustering [25] (see also Imhof's cartographic principles [13]).

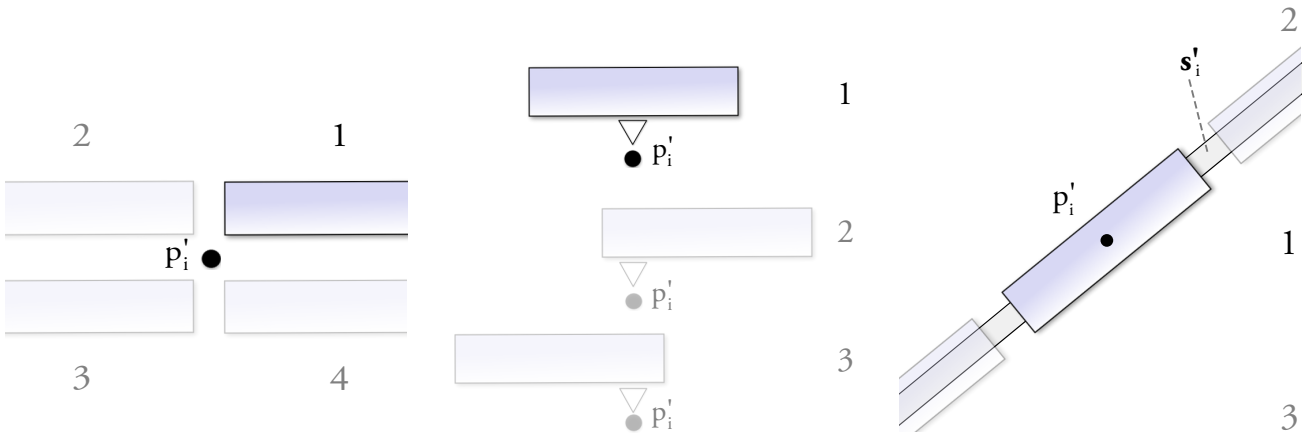
4.2 Features

Following the definition of Imhof [13], 2D/3D scenes contain point features (e.g. graph nodes, points-of-interest), line features (e.g. graph edges, streets), and area features (e.g. graph regions, land cover). Point features can be depicted by icons or horizontal labels (see Fig. 3(a)). Line features can either be labeled horizontally for better readability (see Fig. 3(b)) or using rotated text following a line segment to create a better visual association (see Fig. 3(c)). The labels of area features are always drawn horizontally (see Fig. 3(a)). When loading feature datasets, we first have to compute their labeling positions. Horizontal labels for point features use the feature's world coordinate p_{point} as the labeling anchor. Line features with horizontal annotations use the polyline's center p_{line} as the anchor. For a line feature with rotated text, we determine the longest straight segment $\mathbf{s}_{line} = (p_{line_0}, p_{line_1})$. Finally, for an area feature we compute and store its barycenter p_{area}.

4.3 Initial Placement

The initial placement of labels consists of choosing a position in screen space when a label first appears. The priorities of labels determine the order of their initial placement. First, we compute the screen space position p_i' by projecting the 3D world coordinate p_i of a feature's anchor point. Based on the projected coordinate and the label's size, we then create an OOBB approximation.

As concluded from our expert study in section 3.3, we use the four-position model to label point features. For each of



(a) Horizontal point annotations: The four-position model [31] defines candidate positions ranked by their degree of intuitiveness, e.g. viewers associate most easily a label with its point feature p'_i, if it is placed in the top right corner.

(b) Horizontal line annotations: To label line features at their anchor p'_i we can choose between one of three candidate positions.

(c) Rotated line annotations: Three candidate positions for labeling line features along the projected segment s'_i. The first candidate position is above p'_i, the second candidate is on the right side and the third on the left side.

Figure 3: Initial placement of annotations.

the four candidates k we compute the corresponding OOBB C_k using a screen space offset $\mathbf{o_i}^{(k)}$ from the projected position p'_i (see Fig. 3(a)). The same is done for area features, using the projected barycenter as the anchor. To consistently place line features, we compute three offset OOBBs C_k from p'_i. Depending on the current view angle, the candidate OOBBs are either placed horizontally (see Fig. 3(b)) or along the projected segment $\mathbf{s}'_i$ (see Fig. 3(c)). Each of these candidates k with OOBB $C_k^{(i)}$ of the currently processed label i are tested for collision against the OOBB $C^{(j)}$ of every already placed label j (see section 4.4). If multiple candidate OOBBs $C_k^{(i)}$ are collision free, we choose the position with the best visual association as described by Yoeli [31]. If there is no free position, we do not add the label i. A timer is started and the initial placement is re-evaluated after the timer expires. To comply with the requirement of a temporal coherent layout from section 1, the new label i is smoothly alpha-blended into the layout.

As the placement is ordered by importance, the most relevant annotations are placed in the layout first. Less important labels are filtered out. Enforcing their placement would lead to problems such as collision and clustering. Collision would create rapid movements during conflict resolution, and it would lead to several unreadable labels as stated by Wolff [28;, 27, p. 3-4]. Finally, the resulting clusters would make visual association and reading difficult (see Noyes [19] and Imhof [13]). These points would contradict our goal to follow Imhof's cartographic rules [13].

4.4 Collision

We use a unified approach to layout the labels of all feature categories. Thus, we can involve all feature types to compose the final layout. First, we project all computed positions p_{point}, p_{line}, p_{area}, p_{line_0} and p_{line_1} from 3D world space to 2D screen space, resulting in the projected coordinates p'_{point}, p'_{line}, and p'_{area}, and a projected segment $\mathbf{s}'_{line} = (p'_{line_0}, p'_{line_1})$ (normalized $\hat{\mathbf{s}}'_{line}$). Second, we generate an OOBB encompassing each label. Finally, we compute pair-wise screen space conflicts between all visible OOBBs. Implementation details are discussed in section 5. Every OOBB intersection causes the creation of a force vector $\mathbf{f}_{collision}$ which aims to resolve the collision (see Fig. 5).

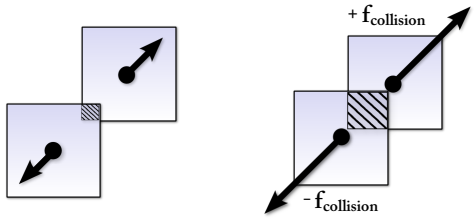


Figure 5: Collision dependent force vector $\mathbf{f}_{collision}$.

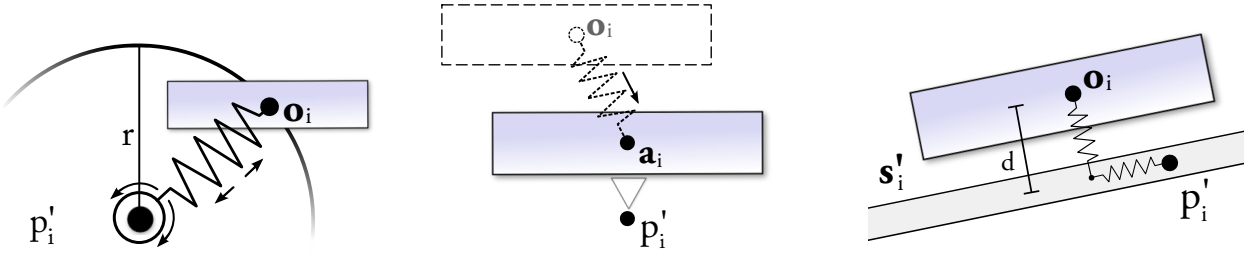
Collisions between OOBBs are computed using the separating axis theorem [10, 24]. It states that if two boxes do not overlap, there must be an axis which separates their projections. First, we normalize the edge vectors of an OOBB with corners $C = \{c_0, c_1, c_2, c_3\}$ to unit length and get the set of normalized axes $\hat{\mathbf{a}}$. Then, we project all its corners C onto every normalized axis. The result is the following interval I:

$$I = [i_{min}, i_{max}] = [min\{\hat{\mathbf{a}} \cdot c_i \in C\}, max\{\hat{\mathbf{a}} \cdot c_i \in C\}] \quad (1)$$

There is no collision between two OOBBs $C^{(n)}$ and $C^{(k)}$ if there exists a normalized axis $\hat{\mathbf{a}}$ in which the respective intervals $I^{(n)}$ and $I^{(k)}$ do not overlap. Thus, there is no collision if for $\delta_0^{(n,k)} := i_{min}^{(n)} - i_{max}^{(k)}$ and $\delta_1^{(n,k)} := i_{min}^{(k)} - i_{max}^{(n)}$:

$$\delta_0 > 0 \quad \vee \quad \delta_1 > 0 \quad (2)$$

To normalize the difference vector $\boldsymbol{\delta} = (\delta_0, \delta_1)^T$ to $[-1, +1]$, we calculate the width in relation to the current projection



(a) Circling annotations: At runtime, a label i with offset $\mathbf{o_i}$ can circle around its point feature p'_i. A spring keeps the label at a distance r from p'_i.

(b) Free annotations: At runtime, a label i can freely move away from its optimum position $p'_i + \mathbf{a}_i$. When no force is acting, the spring repositions the label from $p'_i + \mathbf{o}_i$ to $p'_i + \mathbf{a}_i$.

(c) Line annotations: At runtime, the label follows the projected screen space segment $\mathbf{s}'_i$. A spring positions the label at a distance d from the line $\mathbf{s}'_i$.

Figure 4: Force-based resolution of collisions.

axis as

$$w = -(\delta_0 + \delta_1) \quad (3)$$

$$\mathbf{g} = 2 \cdot (\frac{\boldsymbol{\delta}}{w} + 0.5). \quad (4)$$

Finally, we have to invert the vector to compute the final force with magnitude $|\mathbf{f}| \in [0, 1]$. Thus, the magnitude of the resulting force scales with the amount of overlap:

$$\mathbf{f}^{(n,k)} = \begin{cases} \mathbf{g} \cdot (\frac{1}{|g_y|} - 1) & \text{if } |\frac{g_y}{g_x}| > 1 \\ \mathbf{g} \cdot (\frac{1}{|g_x|} - 1) & \text{else} \end{cases} \quad (5)$$

Analogously to Hirsch [12], we accumulate all collision forces $\mathbf{f}^{(i,k)}$ $(i \neq k)$ for every visible label i, resulting in a label's overall repulsion force

$$\mathbf{f}^{(i)}_{collision} = \sum_k \mathbf{f}^{(i,k)} \quad \text{for } i \neq k \quad (6)$$

4.5 Forces and Movement

After calculating the collision forces $\mathbf{f}^{(i)}_{collision}$ for every visible label i, we resolve conflicts using a force-based approach. This enables the flexible definition of each label's reaction to its surrounding environment. At timestep t every visible label i stores its current screen space offset $\mathbf{o_i}$ from the projected position p'_i, and its velocity $\mathbf{v}_i$. First, we compute the total force $\mathbf{f}^{(i)}_{total}$ acting on the label i. It is composed of several forces, depending on the type of the annotation. In the following, several possible annotation behaviours are introduced: a free annotation, a line annotation and a circle annotation behaviour.

Free Annotation A free label i is positioned by an optional offset $\mathbf{a}_i$ from its projected anchor p'_i (see Fig. 4(b)). When a collision with force $\mathbf{f}_{collision}$ occurs, it can be repelled in any direction to an offset $\mathbf{o_i}$. Its new screen space position becomes $p'_i + \mathbf{a}_i + \mathbf{o_i}$, and an attracting force $\mathbf{f}_{feature}$ pulls the label back to its original position $p'_i + \mathbf{a}_i$. This force $\mathbf{f}_{feature}$ is modelled as a spring with the constant k_1 as in Hooke's law:

$$\mathbf{f}^{(i)}_{feature} = -k_1 \cdot \mathbf{o_i} \quad (7)$$

Finally, we introduce a friction force $\mathbf{f}_{friction}$ to stabilize the force-based system. Based on the current velocity $\mathbf{v}_i$ and a friction coefficient c, we get

$$\mathbf{f}^{(i)}_{friction} = -c \cdot \mathbf{v}_i \quad (8)$$

The total force acting on a free annotation i thus becomes

$$\mathbf{f}^{(i)}_{total} = \mathbf{f}^{(i)}_{collision} + \mathbf{f}^{(i)}_{feature} + \mathbf{f}^{(i)}_{friction} \quad (9)$$

Line Annotation At runtime, labels for line features can follow a straight segment $\mathbf{s}_i$ in screen space. This is achieved by adding two stiff springs which create the forces $\mathbf{f}_{normal}$ and $\mathbf{f}_{tangent}$. $\mathbf{f}_{normal}$ pushes the label displaced by d_{real_0} from its optimum distance $d_{optimum}$ along the normal back onto the line. $\mathbf{f}_{tangent}$ attracts the label along the line back onto its anchor. Using the projected and normalized line segment $\hat{\mathbf{s}}'_i$, the offset $\mathbf{o_i}$ and the line equation $p = p'_i + t\ \hat{\mathbf{s}}'_i$, we compute the nearest point $p_{nearest}$ on the line using

$$p_{nearest_0} = p'_i + t\ \hat{\mathbf{s}}'_i \qquad \text{with} \qquad t = \frac{\mathbf{o}_i \cdot \hat{\mathbf{s}}'_i}{|\hat{\mathbf{s}}'_i|^2} \quad (10)$$

$$d_{real_0} = ||p_{nearest_0},\ p'_i + \mathbf{o}_i|| \quad (11)$$

Using the normal $\hat{\mathbf{s}}'_{normal} = (-\hat{s}'_{i_y},\ \hat{s}'_{i_x})^T$, the spring constant k_2 and the displacement $(d_{real_0} - d_{optimum})$, we compute

$$\mathbf{f}^{(i)}_{normal} = -k_2 \cdot (d_{real_0} - d_{optimum_0}) \cdot \hat{\mathbf{s}}'_{normal} \quad (12)$$

The spring force $\mathbf{f}_{tangent}$ with the constant k_3, pulling the label back to its center position, is computed analogously.

$$p_{nearest_1} = p'_{line} + t\ \hat{\mathbf{s}}'_{normal} \quad \text{with} \quad t = \frac{\mathbf{o_i} \cdot \hat{\mathbf{s}}'_{normal}}{|\hat{\mathbf{s}}'_{normal}|^2} \quad (13)$$

$$d_{real_1} = ||p_{nearest_1},\ p'_i + \mathbf{o_i}|| \quad (14)$$

$$\mathbf{f}_{tangent} = -k_3 \cdot d_{real_1} \cdot \hat{\mathbf{s}}'_i \quad (15)$$

The total force acting on a line annotation i thus becomes

$$\mathbf{f}^{(i)}_{total} = \mathbf{f}^{(i)}_{collision} + \mathbf{f}^{(i)}_{normal} + \mathbf{f}^{(i)}_{tangent} + \mathbf{f}^{(i)}_{friction} \quad (16)$$

Circle Annotation Similar to Hirsch [12], we introduce an annotation which circles around its anchor. We define a spring with force $\mathbf{f}_{circle}$ keeping the label on a radius r

(a) Annotation layout created by Google Earth: not enough labels are placed, most labels can not be read properly and camera movement makes labels jitter.

(b) Labeling of a road network in a GIS. In this figure, we define a large buffer zone around the labels to enhance readability and visual association.

Figure 6: Comparison of annotation layouts created by Google Earth (left) and using our real-time labeling approach (right). Lettering (color, size, font) and temporal coherence directly impacts the readability and visual association of annotations.

around the projected anchor p_i'. Using the distance $|\mathbf{o_i}|$ between the current position and the anchor, the displacement from the equilibrium is $(|\mathbf{o_i}| - r)$. The restoring force $\mathbf{f}_{circle}$ with the spring constant k_4 and the normalized offset direction $\hat{\mathbf{o}}_\mathbf{i}$ is

$$\mathbf{f}_{circle}^{(i)} = -k_4 \cdot (|\mathbf{o_i}| - r) \cdot \hat{\mathbf{o}}_\mathbf{i} \tag{17}$$

The total force for a circle annotation thus becomes

$$\mathbf{f}_{total}^{(i)} = \mathbf{f}_{collision}^{(i)} + \mathbf{f}_{circle}^{(i)} + \mathbf{f}_{friction}^{(i)} \tag{18}$$

4.6 Acceleration

We define a virtual mass m_i for each label i. Its value is determined by the label's importance. The current acceleration $\mathbf{a}_i$ is computed using Newton's second law of motion:

$$\mathbf{a}_i = \mathbf{f}_{total}^{(i)} / m_i \tag{19}$$

Each label i has a screen space offset $\mathbf{o_i}$ at time t. The time difference between consecutive frames is given by Δt. We obtain the new velocity $\mathbf{v}_i'$ and the offset $\mathbf{o_i'}$ at time $t + \Delta t$ from Euler's integration method:

$$\mathbf{v}_i' = \mathbf{v}_i + \mathbf{a}_i \cdot \Delta t \tag{20}$$

$$\mathbf{o_i'} = \mathbf{o_i} + \mathbf{v}_i' \cdot \Delta t \tag{21}$$

5. IMPLEMENTATION

To achieve real-time labeling, we need an efficient computation and resolution of conflicts between labels. We therefore use the GPU for parallel processing of these tasks. Additionally, by using the GPU, we ease the central processing unit (CPU) utilization. As a consequence, the CPU is free to aid in loading, filtering and selection of annotations. The current implementation was done in C/C++, uses OpenGL 3.0 and GLSL.

The initial placement of labels is a sequential problem, as we do not want to place multiple labels on the same free spot. To minimize the computational load, this task is split over consecutive frames. In every frame we place a fixed number of annotations.

5.1 Parallelization

Every label is an independent entity, similar to a particle. Its properties include the current screen space offset, the velocity, the dimension and the mass. These are stored in a global texture buffer object (TBO). First, we use a GPU kernel to project all anchors of visible labels to screen space. Then, using each label's current offset $\mathbf{o}_i$ and its dimensions, we write updated OOBB corners in a TBO. In the second step, a GPU kernel computes the collision between every label pair (n, k) using the separating axis theorem (see section 4.4). This results in a 2D buffer containing force vectors. A cell in row n, column k contains the force $\mathbf{f}_{collision}^{(n,k)}$ created by the collision between the OOBBs n and k (see Table 1). To determine the final force $\mathbf{f}_{collision}^{(i)}$ acting on a single label i, we use a line-wise reduction operation. Finally, as described

Label i	ID 1	ID 2	ID 3	$\mathbf{f}_{collision}^{(i)}$
ID 1	x	(0.0, 0.0)	(0.5, 0.0)	(0.5, 0.0)
ID 2	(0.0, 0.0)	x	(0.1, 0.3)	(0.1, 0.3)
ID 3	(0.5, 0.0)	(-0.1, -0.3)	x	(0.4, -0.3)

Table 1: Pairwise collision creates force $\mathbf{f}_{collision}^{(n,k)}$. Accumulation gives a global collision force $\mathbf{f}_{collision}^{(i)}$ acting on a label i.

in subsection 4.5, we resolve conflicts by applying the force $\mathbf{f}_{total}^{(i)}$ onto its respective label i. The computation of $\mathbf{f}_{total}^{(i)}$ and the necessary Euler step for moving the offset $\mathbf{o_i}$ of a label is done on the GPU. Unfortunately, the precision of Euler's method is strongly tied to Δt. Large values lead to an unstable layout. Thus, labels are subject to harsh position changes. Implementing the fourth order Runge-Kutta method did not lead to significantly better results. In the end, clamping the final acceleration value $\mathbf{a_i}$ was enough to achieve stability. This leads to continuously moving labels and thus temporal coherent labeling.

5.2 Rendering Textual Annotations

After we have determined the current screen position of visible annotations, we render them in a standard way using texture mapped text [14]. To improve readability, we introduce further techniques: We add a dark outline to the font to increase the contrast to the background [23, p.416] (see Fig. 7(a)). A halo around the text clears the space around the label and makes it more readable [20] (see Fig. 7(c)).

(a) Blank. (b) Dark edge. (c) Halo.

Figure 7: Techniques for increasing the readability.

5.3 Enhancements

To further stabilize the labeling layout, we implement the following enhancements.

Enhanced OOBBs Changing the view in a scene with a tight labeling layout creates a lot of movement. We remedy this by implementing two improvements: First, we slightly increase the size of the OOBB encompassing each label to create a buffer zone. Secondly, we define an even larger zone around visible labels where no new annotations can be placed. The latter improvement greatly stabilizes the labeling layout, but has to be used cautiously to avoid filtering out important labels.

Speed-based removal To further stabilize the layout, we remove labels that would otherwise be moving at very high speed. We also remove labels where the different forces acting on them cancel each other out to a large degree, indicating that the label is constricted from multiple sides. The removed labels are inserted again after a given time if there is room for them. These actions help to meet our requirements from section 1, where we stated that labels should make only slow-paced, smooth transitions.

6. RESULTS

In this section, we analyze our force-based labeling approach with respect to scalability and cartographic principles.

6.1 Scalability

We evaluated the performance on two platforms:

- low-class hardware: Intel Core 2 Duo 1.6 GHz, 4 GB RAM, NVidia Geforce 8600M GT (256 MB)
- lower medium-class hardware: Intel Core 2 Quad 2.66 GHz, 4 GB RAM, NVidia Geforce 8800GTS 512 (512 MB)

We measured how the timings scale in respect to the number of labels (see Fig. 8). After each step we synced the GPU calls to the CPU (glFinish) to measure the total GPU processing time. We benchmarked the placement of new labels, collision computation and resolving conflicts using forces.

The creation of GPU buffers, updating the dataset, and the readback from GPU to CPU generates a constant overhead of 2 ms on low-class hardware (1 ms on middle class). The total time for up to 512 features is nearly constant and stays below 5.5 ms on low-class hardware (below 2.5 ms on middle class). Starting from 512 labels, collision takes more than 50% of the total time. With a realistic time budget of 10 ms for real-time labeling, we achieve interactive framerates for up to 1024 labels on low-class hardware (2048 labels on middle class). Also, without syncing the GPU to the CPU, the layout computation time becomes 10% to 25% faster.

A better scalability to display a much higher number of labels (>2048) can be achieved by limiting the collision search space to the label's neighbourhood. It would require to partition all labels in screen space, e.g. with a uniform grid or a quadtree. This was not deemed necessary in our current applications, as we display at most several hundred labels.

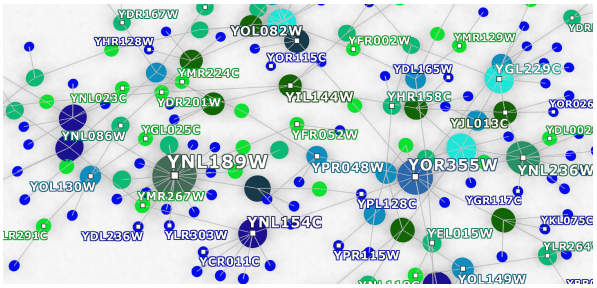


Figure 9: Labeling of a protein interaction network. A small buffer zone around the labels enables the placement of a huge number of labels, at the cost of a more difficult visual association.

6.2 Concluding Expert Study

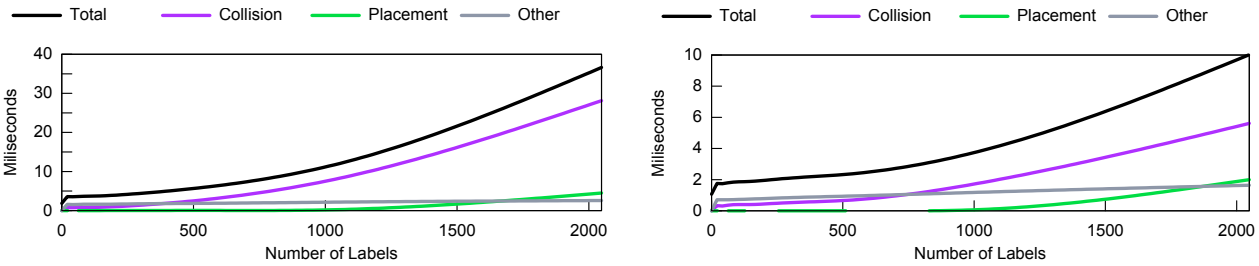
Based on our prototype implementation, we performed a concluding expert study. We invited the same experts as in section 3 into our research facility. Our goal was to validate the domain experts' first recommendations and our subsequent choices.

6.2.1 Study Design

In a similar manner to our past study, the interview lasted one hour and we chose the labeling of a GIS as a representative application. Supported by our real-time prototype, we first ask if the four-position model for labeling point features is appropriate in creating a consistent and clear layout. Secondly, we ask if its application helps associating the label to its feature. As stated in section 3, when labels are at steep angles, we switch from line-aligned (straight, rotated) to a horizontal annotation of line features. We analyze the annotation of both approaches with respect to readability and visual association. We query if the scaling of labels based on their depth creates a better spatial perception. We also study if labels are still readable despite their scaling. We compare our force-based resolution of collisions to the labeling in Google Earth (v6.1.0.5001). In their approach, when collisions occur, labels are removed and replaced in the layout. Also, we check if our solution creates too much movement. To conclude the study, we ask if they approve our force-based approach for real-world scenarios.

6.2.2 Results of the Study

Annotion of Point Features Almost all (5 of 6) experts liked the 4-position-model used for cities. The last expert



(a) Benchmark on low-class hardware: Intel Core 2 Duo 1.6 GHz, 4 GB RAM, NVidia Geforce 8600M GT (256 MB).

(b) Benchmark on lower medium-class hardware: Intel Core 2 Quad 2.66 GHz, 4 GB RAM, NVidia Geforce 8800GTS (512 MB).

Figure 8: Benchmarks of our force-based approach. The plot shows synced timings for the layout computation steps: total labeling time (black), collision computation and resolution (violet), placement of new labels (green) and the overhead generated by other steps (grey). The computation of layouts for several hundred labels remains interactive on every hardware.

suggested that switching positions 2 and 4 (see Fig. 3(a)) would create a better model. Four subjects said its application creates a consistent labeling layout. In their opinion, the labels are easily associated to the corresponding point features. One expert stated that the visual association is difficult when too many labels are on the screen at once. Another expert mentioned that the association depends on the viewing angle.

Annotion of Line Features Roads with horizontal and line-aligned labels could be easily read by all experts. The majority (5 of 6) stated that horizontal annotations allowed an easy visual association. In contrast, only half of the experts could associate line-aligned labels to their corresponding feature. Of these three experts, two noted a difficult association for labels further away from the viewer. Of the other half, one person mentioned that aligned labels hide the underlying road.

Depth Scaling of Annotations All candidates stated that scaling labels depending on their depth helps spatial perception and that all labels are still easily readable.

Comparison Every expert deemed our labeling approach superior to Google Earth. Three of them disliked the suddendly dissapearing labels. They described the approach as confusing and agitated.

Force-Based Collision Resolution All experts were pleased by the alpha blending of labels. The majority of the experts (5 of 6) liked the smoothly changing label positions and were not distracted by moving labels. One stated reason was the aesthetics and two liked the calm layout. The remaining expert described the force-based method as a gimmick. He also mentioned that the labels following a line create too much movement.

6.2.3 Discussion

Depth scaling was unanimously accepted because spatial perception was improved while all labels remained readable. The acceptance of the 4-position model was even higher than in our preliminary study. Horizontal labels for line features were determined to be easily readable and were also rated higher than before. However, two experts rejected the idea of line-aligned labels and two others mentioned cases where it fails. In total, almost all experts approved of the application of our force-based approach for real-world scenarios.

7. CONCLUSION

In this paper, we have presented a real-time force-based labeling approach. It allows the flexible definition of forces to create appropriate layouts. We have presented several force behaviours for labeling point, line and area features. Our method follows Imhof's cartographic principles. Classification of labels is done by font scaling, coloring and by choosing appropriate anchor icons. Visual association is achieved by an appropriate color encoding, distance-dependent scaling and by defining attractive forces pulling the label back to its anchor. Almost all of our domain experts approve the force-based approach. Every expert deems it superior to Google Earth, where labels disappear at collision. They like our smooth transitions, the excellent readability and the good visual association. This is achieved with a temporal coherent layout which enables the user to keep track of annotations during visual exploration. As the entire method uses parallel GPU computations, we achieve excellent performance scalability. On medium-class hardware, our approach can layout up to 2000 annotations in real-time, consuming about 10 ms per frame. This enables the labeling of vast information graphs, GIS on powerwalls, and even allows real-time layout computation on embedded hardware, e.g. for automotive navigation systems. Furthermore, as no precomputation is necessary, it is possible to include dynamic and online annotations. Hence, our force-based approach can be applied to a broad range of applications such as GIS and scientific and information visualization. In further research, we plan to develop more intelligent selection and filtering techniques. Also, we will evaluate the management and visualization of occluded labels in 3D cities and terrains. Using the terrain's depth and silhouette buffer we could appropriately place labels in 3D terrains.

8. ACKNOWLEDGMENTS

We would like to thank Michael Genau for the first implementation of the force-based algorithm. Also, we thank Christopher Roelle for his extensive help during the design of this approach. Finally, we thank Philipp Promesberger for developing huge parts of the map rendering framework.

9. REFERENCES

[1] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3d illustrations. *Journal of the WSCG*, 13(1):1–8, 2005.

[2] K. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Transactions on visualization and computer graphics*, 12(5):773–780, 2006.

[3] B. Bell, S. Feiner, and T. Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 101–110. ACM, 2001.

[4] J. Chen, P. Pyla, and D. Bowman. Testbed evaluation of navigation and text display techniques in an information-rich virtual environment. In *Virtual Reality, 2004. Proceedings. IEEE*, pages 181–289. IEEE, 2004.

[5] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics (TOG)*, 14(3):203–232, 1995.

[6] H. Cords, M. Luboschik, and H. Schumann. Floating labels: Improving dynamics of interactive labeling approaches. In *Proceedings of MCCSIS (IADIS Multi Conference on Computer Science and Information Systems)*, pages 235–238, 2009.

[7] D. Ebner, G. Klau, and R. Weiskircher. Force-based label number maximization. Technical Report TR-186-1-03-02, Vienna University of Technology, 2003.

[8] D. Ebner, G. W. Klau, and R. Weiskircher. Label number maximization in the slider model. In J. Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 144–154. Springer Berlin / Heidelberg, 2005.

[9] M. Feigenbaum. Method and apparatus for automatically generating symbol images against a background image without collision utilizing distance-dependent attractive and repulsive forces in a computer simulation, Oct. 11 1994. US Patent 5,355,314.

[10] S. Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, University of North Carolina at Chapel Hill, 2000.

[11] T. Götzelmann, K. Hartmann, and T. Strothotte. Agent-based annotation of interactive 3d visualizations. In *Smart Graphics*, pages 24–35. Springer, 2006.

[12] S. Hirsch. An algorithm for automatic name placement around point data. *Cartography and Geographic Information Science*, 9(1):5–17, 1982.

[13] E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[14] M. Kilgard. A simple opengl-based api for texture mapped text. 1997.

[15] M. Luboschik, H. Schumann, and H. Cords. Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics (TVCG) / Proceedings of IEEE Information Visualization (InfoVis' 08)*, 14(6):1237–1244, November-December 2008.

[16] S. Maass and J. Döllner. Efficient view management for dynamic annotation placement in virtual landscapes. In A. Butz, B. Fisher, A. Krüger, and P. Olivier, editors, *Smart Graphics*, volume 4073 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2006.

[17] J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard University Center for Research in Computing Technology, Cambridge, Massachusetts, 1991.

[18] K. Mote. Fast point-feature label placement for dynamic visualizations. *Information Visualization*, 6(4):249–260, 2007.

[19] L. Noyes. The Positioning of Type on Maps: The Effect of Surrounding Material on Word Recognition Time. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 22(3):353–360, 1980.

[20] J. O'Beirne. Blog 41latitude. Electronic Blog, December 2010.

[21] I. Petzold, G. Gröger, and L. Plümer. Fast screen map labeling - data-structures and algorithms. In *Proc. 23rd Internat. Cartographic Conf.(ICC' 03)*, pages 288–298, 2003.

[22] S. Pick, B. Hentschel, M. Wolter, I. Tedjo-Palczynski, and T. Kuhlen. Automated positioning of annotations in immersive virtual environments. In *Proceedings of the Joint Virtual Reality Conference of EuroVR - EGVE - VEC*, 2010.

[23] A. H. Robinson. *Elements of cartography*. John Wiley & Sons, Inc, New York, 2 edition, 1960.

[24] P. Schneider and D. Eberly. *Geometric tools for computer graphics*. Morgan Kaufmann Publishers, 2003.

[25] G. Stadler, T. Steiner, and J. Beiglbock. A practical map labeling algorithm utilizing morphological image processing and force-directed methods. *Cartography and Geographic Information Science*, 33:207–215(9), July 2006.

[26] T. Stein and X. Décoret. Dynamic label placement for improved interactive exploration. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, NPAR '08, pages 15–21, New York, NY, USA, 2008. ACM.

[27] S. Van Dijk, M. Van Kreveld, T. Strijk, and A. Wolff. Towards an evaluation of quality for names placement methods. *International Journal of Geographical Information Science*, 16(7):641–661, 2002.

[28] A. Wolff. *Automated label placement in theory and practice*. PhD thesis, Freie Universität Berlin, Universitätsbibliothek, 1999.

[29] A. Wolff and T. Strijk. The map-labeling bibliography. Electronic Bibliography, 2011.

[30] M. Yamamoto, G. Camara, and L. Lorena. Fast point-feature label placement algorithm for real time screen maps. In *Proceedings of the Brazilian Symposium on GeoInformatics (GEOINFO' 05)*, pages 1–13, 2005.

[31] P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9(2):99–108, 1972.