Labeling Points of Interest in Dynamic Maps using Disk Labels

Filip Krumpe
Department of Computer Science - University of Stuttgart, Germany
filip.krumpe@fmi.uni-stuttgart.de

Abstract

Dynamic maps which support panning, rotating and zooming are available on every smartphone today. To label geographic features on these maps such that the user is presented with a consistent map view even on map interaction is a challenge. We are presenting a map labeling scheme, which allows to label maps at an interactive speed. For any possible map rotation the computed labeling remains free of intersections between labels. It is not required to remove labels from the map view to ensure this. The labeling scheme supports map panning and continuous zooming. During zooming a label appears and disappears only once. When zooming out of the map a label disappears only if it may overlap an equally or more important label in an arbitrary map rotation. This guarantees that more important labels are preferred to less important labels on small scale maps. We are presenting some extensions to the labeling that could be used for more sophisticated labeling features such as area labels turning into point labels at smaller map scales.

The proposed labeling scheme relies on a preprocessing phase. In this phase for each label the map scale where it is removed from the map view is computed. During the phase of map presentation the precomputed label set must only be filtered, what can be done very fast. We are presenting some hints that allow to efficiently compute the labeling in the preprocessing phase. Using these a labeling of about 11 million labels can be computed in less than 20 minutes. We are also presenting a datastructure to efficiently filter the precomputed label set in the interaction phase.

2012 ACM Subject Classification Human-centered computing → Geographic visualization

Keywords and phrases Map labeling, dynamic maps, label consistency, real-time, sorting/searching

Digital Object Identifier 10.4230/LIPIcs.GIScience.2018.8

Acknowledgements I want to thank the OpenStreetMap project for the huge amount of geographic data they collected and provide to the public for free use. To have free access to a worldwide, coherent geographic data set was invaluable for this project. Further thank goes to a group of students at the university of Stuttgart for expanding the OpenLayers framework to use the proposed labeling scheme. This work would not have been possible without the open sources of the OpenLayers project. We also want to thank them for their work. Many thanks to the anonymous reviewers for their detailed comments and suggestions for improvements.

1 Introduction

If today someone wants to explore a place anywhere on earth, he uses map services like Google Maps, Here Maps or others. In addition to a classical map these services allow not only to view a static map but to interactively explore the map via zooming and panning of the displayed region. This interactive approach allows to show a rough overview as well as a detailed view of the interesting places on demand. In contrast to former static maps it is not possible to label interactive maps by hand unless zooming is restricted to a set of fixed zoom

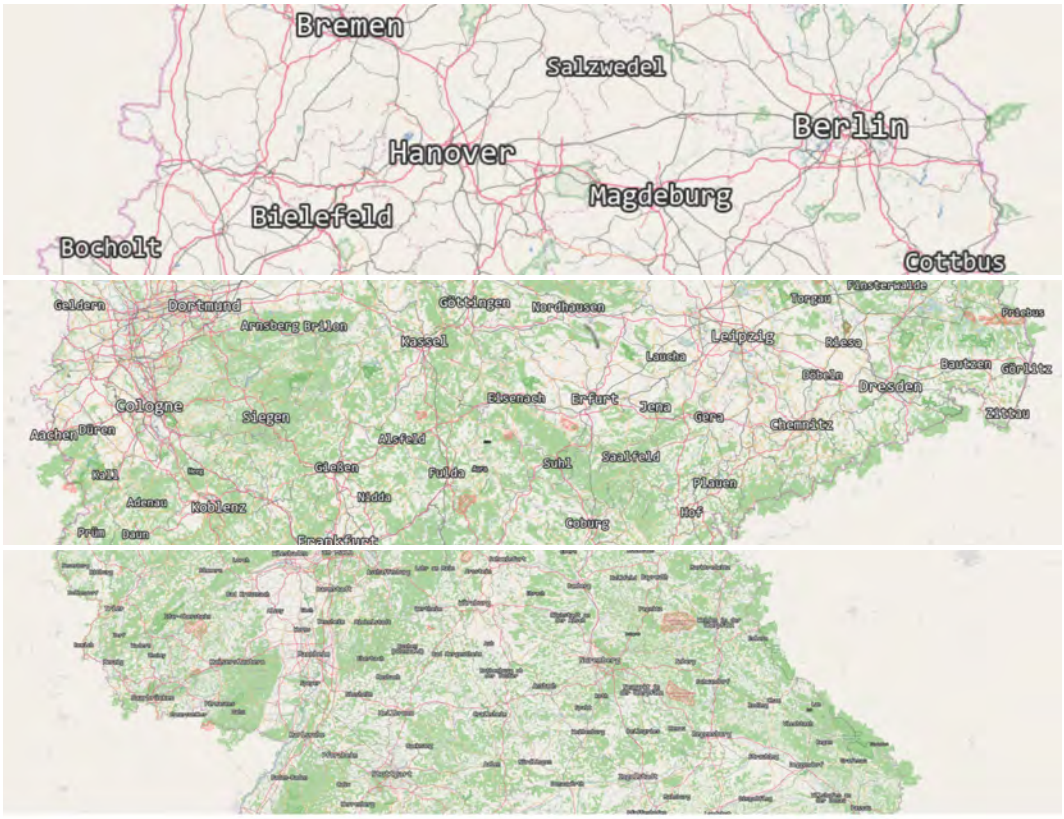

Figure 1 Germany labeled in 3 different levels of detail at a coarse, medium and high level of detail (© OpenStreetMap contributors).

levels. But even in this case it's a lot of work to develop an appropriate labeling for the map on each of the fixed map scales. So people started working on algorithms to automatically perform the label selection and placement. In figure 1 you see a map of Germany in three distinct levels of detail from coarse (in the top third) to detailed in the bottom third, labeled with the proposed labeling scheme

In order to help the user to track the labeled features while continuously zooming or panning the map some best practices were presented (e.g. [9, 3]). An essential one is that two labels should not overlap each other to ensure readability of the presented map. Of course each label should be placed close to the feature it labels. During map interaction a well-known requirement is that a feature which is labeled on a specific scale should not vanish except for moving out of the viewing range. When it comes to zooming, the label of a less important feature, e.g. a street, should disappear before the label of a more important feature, e.g. the town the street is located in. All of these requirements need to be considered if a map is labeled, may it be by hand or automatically.

In navigation systems the panning and zooming of the map is often done by the system itself. So the most appropriate view is presented to the user, e.g. a car driver. If the driver needs to change roads, a detailed view is presented to him. He is presented with a coarse map view if he travels long distances on a highway. What is special to the navigation system setting is that the presented map often is not north oriented but oriented in the current direction of travel. This leads to a special demand for the map labeling. Imagine you are sitting in your car focused on traversing a road junction. When you looked at your

navigation system most recently you saw some nearby towns labeled with their names. After having passed the junction you look at the presented map again that has been rotated in the meantime. Unfortunately you are now presented with a completely different set of labels. This is because some of the former labels would intersect in the current orientation. Some labels, which were overlapping before, can now be presented without overlapping. These modifications in the presented labeling make it difficult for you to regain orientation again.

The scenario shows that a labeling in case of rotating maps especially in navigation systems need to fulfill an additional requirement. A label that is presented to the user at a specific rotation angle should not overlap with another label in any rotation angle. Of course the requirements we described above for interactive maps also need to be fulfilled in this particular use case.

In this paper we will present a new labeling scheme, which allows to compute labelings for arbitrary map scales at interactive speed. The consistency requirements, our labeling scheme guarantees to be fulfilled, are described in section 2. The scheme is based on a preprocessing phase and a filtering step during run-time. It is described below in section 3. In section 4 we describe some extensions to the model, which allow to add more sophisticated labeling features. We implemented the approach and extended the well-known OpenLayers web based map visualization framework. Some details and practical considerations are presented in section 5. Section 6 concludes the paper and sketches some further research topics.

Let us shortly create a common understanding of the basic terms we are using in the context of map interaction before describing some related work. A map may show a specific geographic region, for example Germany, Europe or the whole planet. If the map shows only the most important details, we are talking about a coarse or low level of detail. In contrast a fine grained map shows a lot of smaller features, i.e. a high level of detail (see figure 1 for an example). In practice the level of detail is interrelated to the map scale of the map visualization. So a map of a high scale (e.g. 1:1,000) shows a higher level of detail than a map of small scale, e.g. 1:1,000,000). Given an interactive map at a small scale, showing Germany for example, we may zoom into the map, i.e. increase the map scale while shrinking the view area, to get a more detailed view of our capital Berlin.

1.1 Related Work

Some of the first and well-known approaches to systematically describe general principles of static map labeling were done by Eduard Imhof in the 1980s [9]. He distinguishes three general types of features that can be labeled on a map: area, line and point features. For these features he describes best practices for the placement of associated labels to gain best visibility and readability of a map. His observation are the basis of the work at hand.

In the 1990s the problem to automatically label dynamic maps at an interactive speed arose. Kreveld et al. in 1997 published some models about how to select a suitable subset of settlements for interactive display [10]. In particular their approach of computing a ranking of the settlements that allows to efficiently obtain a solution using filtering can be found in the labeling scheme we are presenting here.

In 2006 Been et al. proposed some general consistency desiderata for dynamic map labelings [3]. Those criteria can be considered almost standard in the field of automatic labeling of dynamic maps. They also describe a framework to automatically derive map labelings fulfilling these criteria at interactive speed for map interactions like panning and zooming. Their approach can be considered a direct parent of the approach we are describing in the paper at hand. In the cited paper Been et al. also provide a nice overview of the research on this topic before 2006.

While Been at al. only consider zooming and panning interaction, Gemsa et al. [7, 8] are considering map rotation. Starting with a given map labeling they defined a model to efficiently derive a subset of labels that can be visible at a specific rotation angle without overlapping each other. Their approach covers map rotation only in particular they do not target the problem of presenting a consistent labeling at several levels of details.

From the algorithmic side there is some previous works of our working group targeting the efficient computation of the so called elimination sequences of growing disks [5, 2, 6]. Given a set of points $p_i \in \mathbb{R}^2$ (or $\mathbb{R}^d$) with associated radii r_i. Each of the points induces a disk (or a hyperball) with radius $r_i \cdot t$. Starting at $t = 0$, t increases continuously and the less prioritized disk is eliminated if two disks touch. The goal is to efficiently compute the elimination sequence and the elimination times for each of the disks. In these two papers a total order for the growing disks is assumed but the results (algorithms and complexity analysis) nicely transfer to other priority functions which only have constant evaluation time. Ahn et al. in 2017 introduced another algorithmic approach that further decreases the computational complexity in [1]. If no order on the disks is given, the problem of computing optimal elimination sequences is NP-hard [3]. Some used mixed-integer programming to compute optimal solutions for these cases [4, 15]. But because of the computational complexity of the problem, in practice they were able to compute solutions only for instances of a few hundred points.

1.2 Contribution

We propose a labeling scheme for dynamic maps, which allows panning, rotating and zooming interaction. It provides consistent labelings at an interactive speed using a computationally challenging preprocessing phase that allows to use efficient and fast filtering techniques during the interaction phase. The scheme is based on the described works of Kreveld et al. and Been at al. . We go beyond the work of Kreveld et al. by additionally taking into account some consistency requirements during zooming operations. Those criteria are inspired by the ones proposed by Been at al. but are exceeding their work by respecting some sort of hierarchy of the geographic objects. In contrast to the work of Been at al. we are also considering rotation as a possible map interaction.

Our results can be extended to some more sophisticated labeling scenarios like area or line labels that turn to point labels while zooming out of the map. This idea is based on an observation of Imhof in [9], namely that area and line labels turn into point labels on smaller map scales.

The labeling scheme incorporates a preprocessing step and allows to use simple filtering and spatial search during the interaction phase to compute a consistent labeling at an adequate speed. The preprocessing of the data can be done efficiently by a proposed algorithm with a running time of $\mathcal{O}(\Delta^2 n(\log n + \Delta^2))$ where Δ is the maximum ratio between two distinct label radii (see [2]). To get an appropriate subset of labels out of the precomputed label ranking we use a priority search tree in combination with a 2-dimensional kd-tree. The implementation is open source and can be found on GitHub [11].

In an associated student project, we extended the well-known OpenLayers web framework [13] to present the capability of our new approach in practice. The so called Tile Rotating Universal Map Projection Presentation is open source and available online [14].

2 Consistency requirements

Based on the consistency desiderata defined by Been at al. in [3] we define the following requirements for interactive maps which allow panning, rotating and zooming of the map view:

(D1) *During monotonous zooming labels should not appear and disappear more than once.* This requirement covers the user expectation that during zooming an object is visible until it is no longer important enough (when zooming out). When zooming in it ensures that a label can vanish if the labeled object covers the whole view or the labeled point for example got replaced by an area or line feature label at larger scales.

(D2) *Labels should not change position or size abruptly on map interaction.* This is because abrupt label changes during map interaction may distract the user and make it difficult to track the position of the labeled objects.

(D3) *During panning and rotation labels are not allowed to appear or disappear except for moving in and out of the view area.* This implies that labels might be partly visible if the label disk is is not fully contained in the view area. Especially in navigation systems the map rotation changes automatically, for example when it is linked to a driving direction. The requirement ensures that always the same set of labels is visible even if the map rotation changed since the last time the user looked at the map. This allows the user to keep orientation with a low cognitive load.

(D4) *The label placement and selection is a function of scale and the view area. It does not depend on the interaction history.* Given this requirement, the map at a specific map setting looks like a static map labeling. So users might directly recognize the places if they look at the map.

Based on the work of Kreveld et al. [10] we add another constraint targeting the fact that there is some inherent order of precedence for geographic features to be labeled.

(D5) *During zooming a label disappears only if it is in conflict with an equally or more important label.* For example a megacity label is preferred to a label of a small rural settlement. So if those two labels are in conflict, the megacity label should be shown instead of the settlement label. Also on small map scales a street name is less important than the label of the city in which the street is located. During the label selection process these precedences need to be taken into account. Our label selection process additionally respects what Kreveld et al. called the *relative importance* of an object. For example a town might have a high relative importance if it is in the middle of nowhere compared to a city that is located directly beside a megacity with millions of inhabitants. The relative importance manifests itself in the fact that the label of the town is shown longer than the city label while zooming out of the map.

Now that we have defined the consistency requirements we want our labeling to fulfill, we will continue describing our model in the following section.

3 The Framework

Formally we are considering a set of point of interest locations $P = \{p_1, \ldots, p_n\}$. For each point p_i we are considering its label l_i that may be a label string, an icon or the like. We also assume to have a priority function that decides for two points p and q if p is prioritized over q or vice versa or none of both. The problem we are faced with is the following: Given a

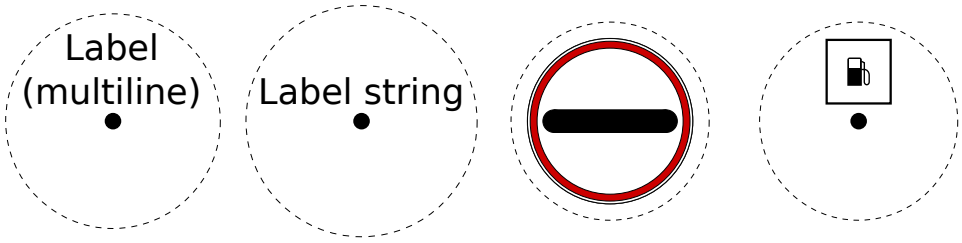

Figure 2 Example of point of interest labels with label strings (left) centered above the labeled feature and icons centered at the labeled feature and centered above the labeled feature (right - [16]). The associated label disks are depicted by the surrounding dashed circle.

specific view area, scale and rotation angle select a subset of points such that a visualization of the corresponding labels fulfills the requirements as defined in section 2. In the following we will describe a labeling model at first and a label selection process that allows to efficiently retrieve such a point set. In our case efficiently means that the label selection process can be subdivided into two phases. In a first phase the label set is preprocessed such that in a second interaction phase the actual label selection reduces to a simple and fast filtering. This allows to efficiently query the data set for a consistent labeling during the interactive visualization phase.

3.1 The label model

In order to fulfill the consistency criteria as defined in section 2 we define a label disk for each of the points to be labeled. The label disk is centered at the corresponding point location and has a specific radius r depending on the label size, i.e. the label length, the font and font size or the icon and icon size. We require a point label to be completely contained within the corresponding label disk in each rotation angle but we do not care about its actual placement. In order to fulfill requirement **(D2)** and **(D4)** the label placement must be a function of scale and rotation angle. It must ensure that the label does not change its position and size abruptly during map interaction. Except for these restrictions, the concrete label placement within the label disk is unconstrained. A fairly simple example for such a placement function, which fits the idea of the model well, is depicted in figure 2. There you see a point label that is horizontally aligned and centered above the labeled feature. During rotation the label remains horizontally aligned and keeps its absolute size on zooming. Icons can be placed e.g. centered at their location or centered above the location like in case of the text label as described before. Of course many other placements are also possible.

Using these label disks, we define a consistent labeling to be a subset of the labels such that the corresponding label disks are non-overlapping. Because each label is completely contained within its corresponding label disk by definition, this ensures that none of the labels are overlapping in any rotation of the map view. So we ensured that during rotation none of the labels need to disappear to avoid label overlap. In figure 3 you can see a visualization of Germany labeled with our scheme in two different rotation angles.

To ensure consistency during panning we come back to a concept Been et al. called an "inverted sequence" in their approach in [3]. The intuitive label selection and placement method is to first select the subset of labels in the view area and placing the corresponding labels afterwards. As Been et al. argued in their paper it is hard to achieve interactive speed and consistency with this approach. What we suggest here is to first pick a consistent labeling globally. From this restricted label set we finally display the labels intersecting our

Figure 3 A labeling of Germany in two different orientations (© OpenStreetMap contributors). The basic label set contains all human settlements extracted from the OpenStreetMap dataset [12].

view area. This selection process ensures that the only way labels appear or disappear on panning is by moving in and out of the view area.

In summary until now our label model requirement **(D3)** is fulfilled, i.e. labels only appear and disappear by moving in and out of the view area. The requirements **(D2)** and (partially) **(D4)** are fulfilled by an appropriate label placement function. For the latter we did not yet define the dependency to the map scale but we are going to make up for it right now.

The map interaction we haven't considered yet is zooming. Zooming out of the map by decreasing the map scale naturally leads to decreasing the level of detail of the map, i.e. less details get visible and labeled on the map. To support this we define the label disk radii to be dependent on the map scale. Instead of the label radius r_i we define the disk radius of p_i to be $r_i \cdot \frac{1}{s}$ where s is the current map scale. You see that decreasing the map scale s enlarges the label disks so a consistent labeling contains less labels – the level of detail decreases. By using these scale dependent label disks, the selection of a consistent labeling gets a function of scale as required in **(D4)**.

The defined label model now allows to have a consistent map view for map interactions at arbitrary map scales. What we have not yet taken into account are the consistency criteria concerning the zooming process itself. As defined in the consistency requirements **(D1)** and **(D5)** for the zooming we have some requirements telling us that labels should not appear and disappear more than once during monotonous zooming. Additionally we require a label to be removed from the view only if it conflicts with a more or equally prioritized label. We will target this in the following section.

3.2 The label selection

In the previous section we defined a labeling model that ensured some requirements to be fulfilled when panning and rotating on an arbitrary map scale. We now want to focus on the process of selecting consistent labelings such that the remaining requirements are

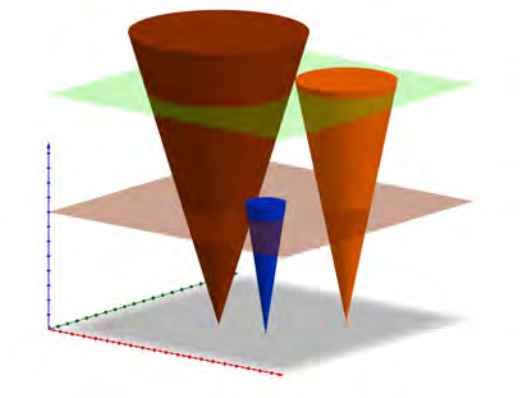

Figure 4 Label cones and two planes (brown, light green) that correspond to label selections at different map scales.

fulfilled. These are concerning the process of zooming in and out of the map, i.e. increasing or decreasing the map scale. There are two requirements left: **(D1)**: 'During monotonous zooming a label should not appear and disappear more than once' and **(D5)**: 'During zooming a label disappears only if it is occluded by an equally or more important point label'

For the sake of simplicity we only consider the process of zooming out of the map, i.e. decreasing the map scale. It is straightforward to transfer the following observations to the process of zooming in. Furthermore we will look at the label disks only and not rely on the actual label placement but assume that this is done in a suitable way as described in the previous section.

In order to find a proper consistent labeling for a target map scale S we use the following process: Starting with a sufficient large s we know that all the corresponding label disks are free of intersections. We continuously decrease s until two of the label disks touch. Now the priority function comes into play. If one of the corresponding points is prioritized over the other, we remove the less prioritized one. Otherwise we remove one of the two. We continue with the process until $s = S$, i.e. our target scale is reached. In the case that we are free to decide which of two equally important point labels to remove, the decision we make influences the further process and so the quality of the labeling. We will come back to this point in the conclusion and further research section.

The process immediately ensures requirement **(D5)** to be fulfilled as a label is removed only if its label disk is in conflict with a label disk of an equally or more important label. Requirement **(D1)** is also fulfilled by design of the label selection as a label never reenters the process after being removed once.

As you can see the label selection process always leads to the same label "elimination sequence" if we assume the label set to be unchanging and the breaking of the ties to happen deterministically. Each label can be assigned a specific map scale where it is removed from the label set during the process. This opens space for our promised precomputation phase. Because the label elimination sequence and the elimination scales for the labels do not change, we can compute them separately in advance. Having computed them for a set of labels we can derive a consistent labeling as follows. For a given map scale S we choose the subset of labels having an associated elimination scale smaller than S and restrict the subset to those labels intersecting the view area. This allows to retrieve a consistent labeling at an interactive speed.

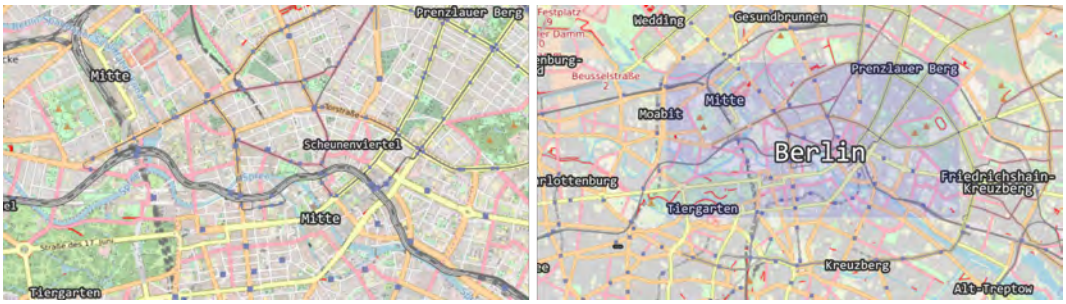

Figure 5 Labeling of a map using a map labeling with popup scales at a larger (left) and smaller map scale (right) where the "Berlin" label popped up (© OpenStreetMap contributors).

Looking at the process from a more abstract point of view we are presented with the following so called space-scale cube: The labels are located in a 2-dimensional plane for example when using the Mercator projection. Using $\frac{1}{s}$ as a third dimension, we see that each label is associated to a cone (see figure 4). The elimination scale of a label determines the height of the cone and the label cones do not intersect. In this view a labeling of a map on a specific scale S corresponds to the intersection of the label cones with the plane at the height oh $\frac{1}{S}$. In the referenced drawing you see two planes corresponding to two different map scales in brown and bright green. The intersection of a cone with the plane corresponds to the label disks of the label at the specific map scale.

4 Extending the model

The labeling model we developed in the previous section opens up opportunities to some extensions. In the following we will discuss some of these.

What we did not take into account yet is the point where a label occurs while zooming out of the map. In the basic labeling model we described before, all the labels are visible at the largest map scale A straightforward approach to extend the labeling model is to introduce a "popup scale" for a label. It means that the label becomes visible at this specific map scale. At larger map scales the label does not exist and also does not occlude any of the existing labels. This concept for example allows to add a label of a city on coarser map views only such that the label does not occlude details of the city while being in a zoomed in map view. This extension does not violate the requirement **(D1)** as the label only appears once during a monotonous zooming operation. An example of a labeling of Berlin is depicted in figure 5.

Having in mind the concept of popup scales as described above, we introduce another extension. As Imhof pointed out in [9] line or area feature labels turn into point labels on smaller map scales. For example a church might be displayed as an area on larger map scales but on a coarser map view its area degenerates to a single point. Analogously the label needs to turn from an area label to a simple point feature label. In figure 6 you see a map in two different map scales where the area of Berlin is labeled as an area on the left hand side while it is labeled as a point object on a smaller map scale (right). The same can be applied to line segment labels. By using the concept of popup times we are able to include this fact into our point labeling scheme. Simply setting the popup time for the point label to the map scale where the area turns into a point in the visualization allows us to support this visualization feature.

Obviously the labeling scheme is not focused on maximizing the number of labels visible but on visibility and readability of the map with low cognitive load. For example for simple

Figure 6 Labeling of an area using an area label (left) and a point label at smaller map scale (right)(© OpenStreetMap contributors).

horizontally aligned labels a lot of the available disk space remains unused. To further increase the number of labeled objects one can think of a multilayer approach as follows. Each label which is removed while zooming out is moved to a second label layer, which is overlayed by the main labeling layer. In this second layer labels might use another font type, color or opacity to make clear it is a background layer. Technically the second layer uses the same labeling model, i.e. none of the disks of labels in the second layer overlap. For each label the elimination scale in the first layer is the popup scale in the second layer. So for each of the layers the consistency requirements are fulfilled but the labels of the different layers might overlap in the visualization. In figure 7 you see an example of a labeling with two layers.

Now that we described the labeling scheme from theoretical point of view and discussed some extensions to the model, we will briefly have a look at the practical details of our work.

5 Computing labelings in practice

As described in the previous section the labeling computation subdivides into two phases. In a preprocessing phase an elimination order or label ranking is computed. The general process and some optimizations to do the precompution more efficiently are discussed in the following section 5.1. When it comes to the visualization and interaction phase the precomputed elimination order needs to be filtered efficiently. Section 5.2 describes an approach to do this by using a combination of a priority search tree and a 2-dimensional kd-tree. To finally visualize the label set we extended the OpenLayers visualization framework. The changes we made are described in section 5.3.

5.1 Preprocessing

A basic algorithm to compute the elimination order is given in the following bottom up approach. For each label p_i we compute the next upcoming label disk collision, i.e. the collision at the largest scale s_i. For the collision at the largest scale we decide which label to remove and start over again with the shrinked label set until only one label is left. This basic algorithm has a computation complexity of $\mathcal{O}(n^3)$ as we need for each label to check the remaining labels for conflicts in a total of $n-1$ iterations. The crucial observation is that

Figure 7 Labeling of Stuttgart that uses a second label layer (filled gray font) to increase the number of labeled points. In this particular rotation the labels "Stuttgart" and "Stuttgart-Ost" in the first layer are occluding the label of "Uhlandshöhe" in the second layer (© OpenStreetMap contributors).

a label may collide with the most far away label if the disk radii are suitable chosen. This observation forces us to really check each other label when checking for the next collision.

In the following we describe some observations which allow to speed up the algorithm. This is a brief sketch of two of our previous results published in [5] and [2].

A first observation allowing us to reduce the computational complexity is that in each iteration we only need to enforce that the globally next collision needs to be correct. All other computed next collisions might not be correct but the overall sequence of eliminations nevertheless is computed correctly. We see that a collision may be computed twice from each of the two colliding labels. In fact it is sufficient if only the label with the larger radius correctly computes a collision. A second observation guiding to a more efficient algorithm affects two subsequent iterations. Consider an iteration at scale s and a label disk which cover less than half the distance to the nearest neighbor of the associated label. We can argue that we do not need to search for the next collision of this labels until its label disk covers half the distance to the corresponding nearest neighbor. In the meantime some labels might be removed from the label set shrinking the set of labels we need to check for possible conflicts. These two observations allow us to conclude the following: When searching for a next collision of a label we only have to consider a subset of the labels. The maximum size of this particular subset depends on $\Delta = \frac{r_{max}}{r_{min}}$, i.e. the ratio between the largest and the smallest of all radii r in the instance.

Furthermore we observe that many predicted collisions remain the same in two subsequent iterations. So we only need to compute them once. We proved that it is sufficient to maintain the computed collisions in a priority queue. When removing a label we only need to recompute the next collision for a subset of direct neighbors of this label.

The described improvements heavily depend on efficient implementations of two spatial operations: "Finding the nearest neighbor in a point set for a query point" and "Finding all

Table 1 Peak memory consumption (space) and execution times (time) for the precomputation of real-world data sets from the OpenStreetMap project [12].

Dataset	#items [10^3]	time [mm:ss]	space [MB]
Germany	1,308	2 : 05	1,715
Europe	6,468	12 : 23	7,947
Planet	11,006	19 : 33	13,852

points within a distance of d around a query point".

We implemented the preprocessing algorithm with its various optimizations for point sets with a total order. This allows us to decide the priority function in constant time. In our implementation we used Delaunay triangulations to implement the spatial queries. Our implementation allowed us to compute label rankings for millions of labels in a reasonable time (see table 1). These fast computation times enabled us to recompute labelings for quickly changing point sets, which for example are containing live traffic information.

5.2 Label Selection

In the interaction phase we are provided with a set of labels and the corresponding elimination scales, i.e. a location, a scale value and the label information. For a given query consisting of a range given in maximum and minimum latitude and longitude coordinate and a map scale we need to report all label points located in the region with an elimination scale smaller than the requested scale. To efficiently answer such queries we use an approach that combines a priority search tree with a 2-dimensional kd-tree. The data structure is a 2-dimensional search tree of the label positions. The tree fulfills the min heap property on the associated elimination map scales.

The root node of the tree contains the label with minimum elimination scale. The remaining labels are split into two equally sized subsets according to their *longitude* coordinate. For each of the subsets a subtree is constructed rooted at the label with the minimum elimination scale and the remaining labels are split according to their *latitude* coordinate. The left subtree contains all the labels with smaller coordinate and the right subtree all the labels with larger coordinate. In the third layer the labels are again split according to their *longitude* coordinate and so forth. To each of the tree nodes we also append the value of the coordinate value, which separates the labels in the two subtrees.

Now at query time we traverse the data structure starting at the root node. If the current node has an elimination scale which is less or equal to the requested map scale we need to further explore the subtree. If the current node is contained within the query rectangle the label is to be reported. We need to further explore the left subtree only if the split value of the node is larger than the corresponding minimum value of the requested range. If the maximum of the corresponding query dimension is larger than the split value, we need to explore the right subtree.

5.3 Visualization

To evaluate the labeling result and to provide them to the public there was a student project associated with the research project. An extension to the well-known OpenLayers framework for web based map visualization [13] was developed. Screenshots of the testing instance can be seen in the figures 1 and 3. We created a REST based web service that allows to query a

precomputed label set for the active labels in the given display setting, i.e. displayed map range and the current map scale. On the client side the modified OpenLayers framework uses map tiles without point of interest labels and adds a layer displaying our label set on top of it. The labels are placed centered at the label position and remain horizontally aligned when the map is rotated by the user. One of the harder parts in implementing the framework extensions was to present the continuous zooming capability of our approach. Therefore we needed to modify some parts of the original framework. First we needed to interpolate the map scale between the fix zoom levels, which are provided by the framework itself, in order to visualize the correct subset of labels. Second we had to adopt the framework such that during the zooming between the levels a refresh of the label layer was triggered.

We also implemented a caching mechanism for the labeling data that allowed to reduce the number of required server requests. This mechanism shows an important property of our labeling scheme. When requesting the labeling for a specific map setting, we in fact enlarge the requested query range and increase the requested map scale. The provided labeling now contains additional labels that do not need to be displayed in the current setting. But the required labeling is a subset of the provided one and we can filter out the additional labels by simply skipping elements with a larger elimination scale or those which are not contained within the displayed region. The crucial point is that provided label set allows us to directly handle small zooming or panning interactions locally without the need to immediately request new data from the server.

6 Conclusion

We introduced a new model for labeling interactive maps, which allows panning, zooming as well as rotating the map. The labeling ensures some consistency criteria to be fulfilled. During continuous zooming a label appears and disappears at most once. If a label disappears while zooming out, it is because there exists a map rotation where it overlaps a label of equal or higher importance. When panning or rotating a map a label only appears or disappears if moving out of the view area.

The labeling model depends on a precomputation phase that allows to efficiently derive labelings for arbitrary map scales in an interaction phase using filtering. The precomputation can be done in less than 20 minutes for labelings with around 10 million labels on a standard desktop computer if a total order on the labels is given. This allows to quickly recompute labelings for example to add traffic information labels.

In an interactive visualization phase the labeling computation is a filtering step only. An algorithm to efficiently do that even for large datasets was introduced. The resulting label set has some nice property namely that a labeling for smaller map scales is a subset of the current label set. So labelings for smaller map scales can be derived by filtering the current label set. This idea was also sketched in the paper at hand.

Further research should target the computation of labelings for point sets which do not have a total order but some kind of hierarchy levels the points of interest are belonging to. A top level might for example contain all the megacities. The next levels of decreasing priority might contain other city, town, street and point of interest labels and so forth. In such instances heuristics might be used to solve conflicts between labels of the same hierarchy level. The outcome of such heuristics could be compared with optimal results, e.g. computed by a linear program solver.

Regarding the quality of the computed labeling it is of interest to compare the computed labeling with a maximum subset of labels whose associated disks are free of intersections at

this particular scale. This would be suitable to show the influence of the zooming consistency requirement to the quality of the labeling, i.e. the number of labels.

Further development of our visualization might contain the labeling of area and line features that turn into point labels on smaller map scales. An initial approach how to do that was sketched in the section about extensions to the label model.

References

1 Hee-Kap Ahn, Sang Won Bae, Jongmin Choi, Matias Korman, Wolfgang Mulzer, Eunjin Oh, Ji-won Park, André van Renssen, and Antoine Vigneron. Faster Algorithms for Growing Prioritized Disks and Rectangles. *arXiv:1704.07580 [cs]*, 2017. arXiv: 1704.07580. URL: http://arxiv.org/abs/1704.07580.

2 D. Bahrdt, M. Becher, S. Funke, F. Krumpe, A. Nusser, M. Seybold, and S. Storandt. Growing balls in $\mathbb{R}^d$. In *2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, Proceedings, pages 247–258. Society for Industrial and Applied Mathematics, 2017. DOI: 10.1137/1.9781611974768.20. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611974768.20.

3 K. Been, E. Daiches, and C. Yap. Dynamic Map Labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):773–780, 2006. doi:10.1109/TVCG.2006.136.

4 Ken Been, Martin Nöllenburg, Sheung-Hung Poon, and Alexander Wolff. Optimizing active ranges for consistent dynamic map labeling. *Computational Geometry*, 43(3):312–328, apr 2010. doi:10.1016/j.comgeo.2009.03.006.

5 Stefan Funke, Filip Krumpe, and Sabine Storandt. Crushing Disks Efficiently. In *Combinatorial Algorithms*, Lecture Notes in Computer Science, pages 43–54. Springer, Cham, 2016. doi:10.1007/978-3-319-44543-4_4.

6 Stefan Funke and Sabine Storandt. Parametrized runtimes for ball tournaments. In *Proc. 33rd European Workshop Comput. Geom.(EWCG)*, pages 221–224, 2017.

7 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Consistent Labeling of Rotating Maps. *arXiv:1104.5634 [cs]*, apr 2011. arXiv: 1104.5634. URL: http://arxiv.org/abs/1104.5634.

8 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Evaluation of Labeling Strategies for Rotating Maps. *J. Exp. Algorithmics*, 21:1.4:1–1.4:21, apr 2016. doi:10.1145/2851493.

9 Eduard Imhof. Positioning Names on Maps. *The American Cartographer*, 2(2):128–144, jan 1975. doi:10.1559/152304075784313304.

10 Marc Van Kreveld, Rene Van Oostrum, and Jack Snoeyink. Efficient settlement selection for interactive display. In *In Proc. Auto-Carto 13: ACSM/ASPRS Annual Convention Technical Papers*, pages 287–296, 1997.

11 Filip Krumpe. Runtime datastructure project, 2017. [Online; accessed 10-February-2018]. URL: https://github.com/krumpefp/runtime_datastructure.

12 OpenStreetMap. Openstreetmap project, 2017. [Online; accessed 4-February-2018]. URL: https://www.openstreetmap.org.

13 OpenLayers Project. Openlayers project page, 2017. [Online; accessed 10-February-2018]. URL: http://openlayers.org/.

14 TRUMP Project. Tile rotating universal map projection project, 2017. [Online; accessed 10-February-2018]. URL: https://trump-fmi.github.io/#/.

15 Nadine Schwartges, Dennis Allerkamp, Jan-Henrik Haunert, and Alexander Wolff. Optimizing Active Ranges for Point Selection in Dynamic Maps. In *Proceedings of the 16th ICA Generalisation Workshop (ICA'13)*, 2013.

16 OpenStreetMap Wiki. Map icons/proposed icons — openstreetmap wiki, 2017. [Online; accessed 4-February-2018]. URL: http://wiki.openstreetmap.org/w/index.php?title=Map_Icons/Proposed_Icons&oldid=1463667.