Particle-Based Labeling: Fast Point-Feature Labeling without Obscuring Other Visual Features

Martin Luboschik, Heidrun Schumann and Hilko Cords

Abstract— In many information visualization techniques, labels are an essential part to communicate the visualized data. To preserve the expressiveness of the visual representation, a placed label should neither occlude other labels nor visual representatives (e.g., icons, lines) that communicate crucial information. Optimal, non-overlapping labeling is an NP-hard problem. Thus, only a few approaches achieve a fast non-overlapping labeling in highly interactive scenarios like information visualization. These approaches generally target the point-feature label placement (PFLP) problem, solving only label-label conflicts.
This paper presents a new, fast, solid and flexible 2D labeling approach for the PFLP problem that additionally respects other visual elements and the visual extent of labeled features. The results (number of placed labels, processing time) of our particle-based method compare favorably to those of existing techniques. Although the esthetic quality of non-real-time approaches may not be achieved with our method, it complies with practical demands and thus supports the interactive exploration of information spaces. In contrast to the known adjacent techniques, the flexibility of our technique enables labeling of dense point clouds by the use of non-occluding distant labels. Our approach is independent of the underlying visualization technique, which enables us to demonstrate the application of our labeling method within different information visualization scenarios.

Index Terms—Interactive labeling, dynamic labeling, automatic label placement, occlusion-free, information visualization.

1 Introduction

Labeling graphical objects is a fundamental task in the field of information visualization to communicate the visualized data and to identify different objects. For example, the visualization of graphs profits a lot from the labeling of visual representatives. The main challenge in solving the labeling problem is to find a global labeling solution without any overlapping of other labels. This problem has been addressed comprehensively in the field of cartography, since general legibility is the most important trait of a cartographic map. The affinity of dynamic maps and interactive information visualization regarding labeling needs and the long-time expertise gained in cartography motivates the use of cartographic approaches in the field of information visualization as well.

Typically the non-overlapping labeling of general graphical objects is an NP-hard problem (e.g., [13] [15]). Therefore, current approaches generally use approximations and heuristics to decrease complexity and thus, processing time. Besides the general aim of placing a maximum number of non-overlapping labels, in general cartographic preferences have to be considered. Since the labeling of points is a widespread problem, most existing labeling techniques (cf. to [21]) focus on the point-feature label placement problem (PFLP) solving label-label conflicts. To achieve an interactive labeling in dynamic scenarios, the corresponding methods generally use intense preprocessing steps (e.g., in [3] [18] [22] [23]) or a reduction of possible label positions (e.g., in [9] [16]) and/or labeling space (e.g., in [10]).

Most information visualization techniques use non-textual visual attributes (e.g., color, shape, size) to represent information. Labeling in information visualization should regard those important visual attributes to maintain the legibility of visual representatives and thus of encoded information. If crucial information are occluded, the whole visualization may be worthless. Besides this fact, the demand for interactivity (e.g., human-computer-interaction, changing data sets) in today's information visualization systems, requires very fast labeling algorithms without preprocessing. To the best of our knowledge, a combination of both — a very fast labeling also respecting the drawn visualization elements — cannot be found in current literature.

Hence, we present a practical particle-based interactive labeling method that does not obscure existing visual features. Our main design criteria is a good tradeoff between labeling performance and labeling quality. For performance reasons, we split up the labeling process into several labeling steps arranged as a labeling pipeline. This pipeline uses increasingly more complex computations to place labels for successively fewer remaining point-features. Furthermore, our labeling approach uses particles to determine occlusions between labels efficiently. By transferring graphical information from image space to our particle system, our method additionally guarantees that labels do not obscure other graphical features. Moreover, we introduce a distant labeling method that enables us to label dense point clouds, which principally could not be labeled with adjacent labeling techniques (label-positions tangential to point-feature). As we will demonstrate, our particle-based PFLP is suitable for interactive environments with many point-features as well.

In practice, the presented approach achieves fast results comparing favorably to other interactive labeling approaches — e.g., several thousands of point-features can be labeled at interactive rates without preprocessing — being very attractive for highly interactive environments like information visualization. Finally, our solution is independent of the underlying visualization technique and extensible to the axis-aligned labeling of line- and area-features.

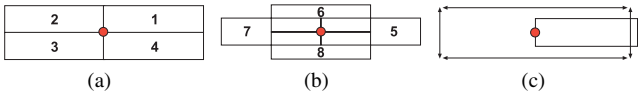
An overview over related work is given in the following Section 2. The main contribution of this paper is a fast particle-based labeling solution for point-features, respecting important visual objects (Section 3). Different application scenarios are described in Section 4. Detailed measurements and discussion are presented in Section 5 to demonstrate the potential and limits of our labeling approach. This paper is concluded in Section 6.

2 Related work

The problem of automatic label placement has been attracting researchers for decades. Applications lie in the field of cartography, computational geometry, or information visualization. Due to the complexity of the problem and different application scenarios, various strategies have been presented for automatic labeling.

- *Martin Luboschik, University of Rostock, E-mail: martin.luboschik@uni-rostock.de.*
- *Heidrun Schumann, University of Rostock, E-mail: heidrun.schumann@uni-rostock.de.*
- *Hilko Cords, University of Rostock, E-mail: hilko.cords@uni-rostock.de.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.



(a) (b) (c)

Fig. 1. The common positioning models with their possible positions (rectangles) and rankings (1: good ... 8: worst): (a) 4-position model, (b) positions 5–8 of the 8-position model, (c) 4-slider model.

Generally the labeling process can be split up into three stages [17]:

- Calculation of possible labeling positions (especially for line- and area-features).
- Rating of the found labeling positions according to cartographic preferences.
- Selection of one label position for each feature that minimizes the overlapping of other labels and simultaneously increases the rating of the global solution.

In point-feature labeling, the possible labeling positions and their rankings (only considering the point-feature itself) can be determined in a straight forward way, since labels are mostly approximated by their axis-aligned bounding rectangles being tangential to the point-feature. Most PFLP approaches use the 4-position or 8-position model with their corresponding rankings which refer to [12]. This discrete positioning has been extended into a so-called slider model [11]. These strategies (see Figure 1) are the most popular ones as they provide a high legibility on different media (paper, screen...).

Over the years, several selection methods have been applied to PFLP to find cartographically good solutions in a short time: greedy, gradient descent, simulated annealing, integer programming, genetic algorithms, tabu search ... (for an overview see [7] [17] [21]). In fact, most of the mentioned methods focus on finding a solution that places a majority of labels, rather than considering the labeling time.

Dynamic and interactive cartographic or visualization applications have raised the need for fast methods. In interactive environments, which include tools like interactive zooming, the complete labeling has to be refreshed constantly at interactive frame rates. Therefore, several techniques have been developed and successfully applied to real-time-scenarios like dynamic map labeling. In such scenarios, the underlying dataset is *static* and hence, intense preprocessing can be applied to guarantee a *real-time interaction phase* [3] [18] [23]. Generally, these methods are not applicable to *dynamically* changing point-features as they are often found in information visualization.

Other approaches like simulated annealing [7], genetic algorithms (e.g., [19]) or tabu search [22] are too slow for interactive scenarios or the number of placed labels falls short of the demands (see [7]).

The graph-theoretic algorithm in [20] possesses a runtime complexity of $O(n\sqrt{n})$. The presented results promise a real-time suitability with high quality labeling, but the approach presumes a fixed height of labels. Since different label sizes (including height) are a common tool in visualization sciences to express importance or data values, this labeling method also constrains well-known mapping strategies. Another approach that enables a very fast labeling of thousands of objects without preprocessing has been presented in [16]. It is optimized for uniformly sized labels, although allowing non-uniform labels with decreasing quality. Unfortunately, some unlabeled point-features may be obscured completely by placed labels.

The need for an interactive labeling method induced the independent development of labeling techniques in the field of visualization. Besides the basic labeling techniques of *Tooltip-Labeling*, *Rapid-Label-All* and *Label-What-You-Can*, there are some more sophisticated techniques. They generally use non-adjacent labeling to address the problem of dense geometry. The approach of excentric labeling [8] for example uses a lens to select the point-features to be labeled and two attached vertical label lists. Labels and point-features are connected by lines. Unfortunately, the label lists may occlude crucial information in the lens' neighborhood. An occlusion-preventing space management technique has been introduced in [4], which is also used in labeling scenarios [5]. However, the complexity of space management and rough rectangular space approximations hinder an effective labeling of thousands of point-features simultaneously regarding other graphical elements. In [10] two approaches have been presented: Whereas the first is based upon [2] and a client-server architecture, the second combines excentric labeling and space management to compensate their respective disadvantages. Again the high effort for space management prohibits an extension of the latter, local method to screen filling real-time application.

In [2] an important real-time labeling approach is presented that is widely used in recent volume visualization systems (e.g., in [6] [14]) and offers high quality illustrative labeling. It uses a force-based approach (see [11]) to place labels onto an objects surrounding contour. This approach respects nearby graphical elements and labels by special repelling forces. Labeling a tight group of centrally placed objects — as they are often found in illustrations — avoids the problem of local minima that appears in classical force-based methods. Thus, it is not applicable in scenarios with several loosely scattered point-features.

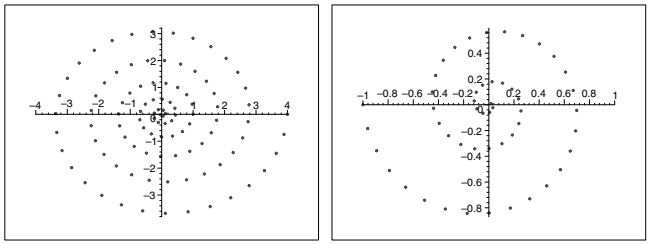
A common argument that is often used to bypass intense labeling calculations, is the interaction technique of zooming (e.g., in [1] [16]). Indeed, this technique results in additional space between point-features and thus, increases the number of placeable labels. It is successfully applied and furthermore necessary in dynamic maps. But then there are several visualization techniques that communicate the most important information without the need of zooming (e.g., scatterplots, tree visualizations...). Thus, an otherwise unnecessary zooming functionality is only needed to bypass missing labeling performance.

This paper concentrates on the fast placement of as many labels as possible while respecting important visual elements. The performance of our concept allows common interactions (i.e., zooming and panning) with complete label refreshment, if required. Our method calculates and rates label positions (also considering non-textual elements) on the fly and finally positions a large amount of labels at interactive rates.

3 A NEW APPROACH FOR INTERACTIVE LABELING

The NP-hardness of the labeling problem arises from the need for a global solution minimizing overlapping, maximizing the number of placed labels and maximizing the global labeling quality (rating). Thus, the amount of possible solutions increases exponentially with the number of features to be labeled. For this reason, in interactive labeling environments, a compromise between performance and quality has to be found. Existing concepts reduce complexity by using hard constraints. For example, the amount of possible label positions is reduced (see Section 2), resulting in an increased performance but squandered solutions and space. Other constraints like fixed label sizes contradict practical needs since labels generally have different lengths. Moreover, label size is a commonly used visual attribute in visualization. Furthermore, there is often a demand for labeling of objects other than point-features (e.g., lines or icons) and the overall aim to prevent occlusion of crucial information — mostly encoded in non-textual visual elements.

Our labeling approach requires no such strict limitations for the sake of acceleration. Instead, it increases the performance immensely by seeking the first local solution exploiting the surrounding area but not seeking a globally optimal solution. Therefore, we define a labeling pipeline consisting of several labeling steps (Section 3.1) with increasing complexity. Each pipeline step labels as many point-features as possible. Thus, the more complex labeling steps only have to be fulfilled for fewer elements. Using particles to detect labeling conflicts (Section 3.2) and an appropriate data structure increases performance further on and ensures high flexibility. Moreover, the use of particles facilitates the consideration of other visual elements during labeling (Section 3.3) to prevent their occlusion. Finally, our approach can be extended to the axis-aligned labeling of arbitrary objects (Section 3.4) and non-rectangular labels (Section 3.5). Our technique can roughly be described as a deterministic greedy-algorithm testing several label-



(a) (b)

Fig. 2. Space sampling function: (a) $r = 4$, $m_{max} = 100$, $c = 6$, $d = 1$ (b) $r = 1$, $m_{max} = 50$, $c = 3$, $d = -1$.

ing positions. With that it tries to maximize the solution, without trying to overcome local minima using any heuristic. This is where our approach significantly differs from known approaches like force-based techniques.

3.1 Labeling pipeline

To achieve high frame rates, we aim at fast results in the first place and according to this feature, we present the concept of a labeling pipeline. The basic idea is to label as many point-features as possible within a fast labeling step, to save CPU power and processing time for more sophisticated approaches applied to fewer points. Since the labeling pipeline is a general construct, single pipeline steps can be exchanged or ignored. We define the following labeling pipeline as a result of experimenting with different pipelines:

1. Labeling with the 4-position model,
2. Labeling with positions 5–8 of the 8-position model,
3. Labeling with the 4-slider model,
4. Labeling with distant positions.

Current techniques generally apply one or two of these steps repetitively, trying to optimize the results (cf. to [7]). Our approach uses every pipeline step only *once* — and only for unlabeled point-features. Each step is executed separately for all point-features not labeled by the previous one. In steps 1–3 we use the well-known positioning preferences found in labeling literature (see Figure 1).

Our approach seeks available labeling space nearby unlabeled features in the fourth step, if adjacent labeling (step 1–3) does not succeed in placing all labels. For that purpose, a space sampling function is defined, providing additional labeling positions of which we select the position that induces no conflict and is closest to the labeled feature. In the ideal case, the space sampling function samples the entire surrounding area. In practice, we found that more sparse defined functions are useful to increase performance, decreasing the labeling quality only marginally. We construct the space sampling function as a spiral $\mathbf{s}(m) \in \mathbb{R}^2$ with adaptive sampling:

$$\mathbf{s}(m) = \begin{pmatrix} d \cdot \cos(2\pi\sqrt{\frac{m}{m_{max}}} \cdot c) \\ \sin(2\pi\sqrt{\frac{m}{m_{max}}} \cdot c) \end{pmatrix} \cdot \frac{m}{m_{max}} \cdot r, \quad m = 1 \ldots m_{max},$$

where m_{max} is the number of sampling points, r is the maximum radius of the spiral and c defines the number of rotations within the spiral. The orientation of the spiral (left- or right-handed) is given by parameter $d \in \{1; -1\}$. Thus, the definition of sampled space is intuitive (Figure 2). Distant labels and the corresponding point-features are connected by lines to indicate affiliation.

Importance labeling. The described labeling pipeline discriminates labels in labeling order: Whereas there is a high probability to find a good adjacent labeling for the point-features labeled first, later features are endangered to be labeled with distant labels due to missing space. Since many applications use labels with different levels of importance, this is more likely an advantage than a disadvantage. E.g., in cartography, the name of a state is generally more important than the names of cities or rivers. In information visualization, the importance of a visual element or its label is difficult to determine automatically and depends on the application scenario. Assuming a given importance for each label, the labels can easily be handled with our pipeline accordingly. Each importance level is processed separately starting with the most important one: All features within one level are labeled by executing the whole labeling pipeline. Thus, more important items are more likely to get a closer label.

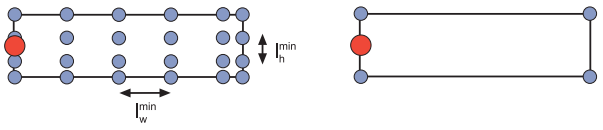
3.2 Particle-based point-feature labeling

The essential task to prevent occlusions in labeling is to detect conflicts between labels. While using axis-aligned rectangular label approximations, this generally means to detect intersections of rectangles. In our approach we use a set of so-called *conflict particles* to detect conflicts as they simplify calculations and increase flexibility. This set of particles is composed of *label particles* and *virtual particles*. The label particles represent the n 2D point-features, characterized by their positions $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^2$. The virtual particles are used to indicate occupied space. They can be created or removed dynamically.

We use conflict particles as follows: A candidate label position induces *no* labeling conflict if there is *no* conflict particle in the corresponding rectangular area — the label position is accepted. Hence, the conflict test for a given rectangular label with coordinates $(x_{left}, y_{bottom}, x_{right}, y_{top})$ and a given conflict particle with position (x_p, y_p) is reduced to the following logical expression:

$$\begin{aligned} accepted = \neg \big(& (x_p > x_{left}) \quad \wedge \quad (x_p < x_{right}) \quad \wedge \\ & (y_p > y_{bottom}) \quad \wedge \quad (y_p < y_{top}) \big). \end{aligned}$$

To prevent later occlusion of a label just added, we have to generate new virtual particles representing the space occupied by the new label. Therefore, the rectangular label area is discretized by the minimal label size l_w^{min},l_h^{min} and populated with virtual particles (Figure 3a). In the case of uniform label sizes, this results in only four particles precluding other labels from overlapping (Figure 3b). Besides the prevention of label-label conflicts, we use virtual particles to avoid the occlusion of other visual elements as well — described in Section 3.3.



(a) (b)

Fig. 3. Virtual particle placement: (a) Each label area is discretized by the minimal label size l_w^{min},l_h^{min} and populated with virtual particles. (b) Uniform label sizes result in only four virtual particles.

To avoid an exhaustive test with all conflict particles, we use a local neighborhood data structure. To locate the neighboring particles of a point-feature, we apply a standard grid-based orthogonal range search method, where one grid element holds all particles within its area. This way, only conflict particles from affected surrounding grid elements have to be considered when testing the labeling positions of one point-feature. To reduce expensive memory access, we chose a grid spacing equivalent to the maximum label width and height.

In the following, the main steps for labeling a point-feature *pf* within one labeling pipeline step are given in pseudo code:

```
for each possible label position lp of point-feature pf:
    conflict = false
    for each conflict particle cp from grid elements affected by lp:
        conflict = conflict ∨ ¬ accepted (lp, cp)
    end for
    if ¬ conflict:
        use label position lp
        generate conflict particles for new label and save to grid
        return (pf is labeled)
end for
return (pf cannot be labeled)
```

The possible label positions *lp* are determined by the labeling method of the current labeling pipeline step (Section 3.1). If no conflict occurs, the actual position is accepted, otherwise the next label position is tested. If all positions result in conflicts, the currently handled point-feature *pf* is declared as unlabeled. Here it becomes clear that our approach selects the first solution found and is basically a greedy algorithm.

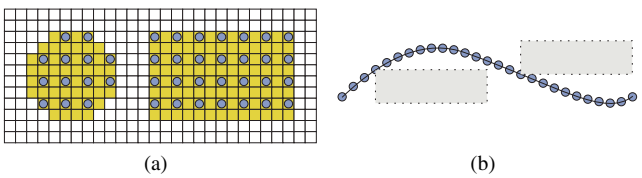
3.3 Transferring graphical data into particle space

The introduced virtual particles represent information about occupied space and act as conflict particles. Anywhere they are located, a label cannot be placed. By the use of virtual particles, any visual element can be sampled and included into the conflict test and hence, can be prevented from being overlapped by labels. In the following, we describe how we use virtual particles to define prohibited areas in screen space.

Image-based approach. In principle, the virtual particles can be generated in different ways. We chose an approach being independent of the underlying visualization technique, coping with visual elements of any shape that may also change over time. Input is an image of the rasterized elements to be regarded — the collision map. We presume a fixed color co_{empty} which defines available labeling space (e.g., background color). Now we introduce virtual particles in screen space coordinates for all color values $\neq co_{\text{empty}}$ (see Figure 4a) by sampling the collision map.

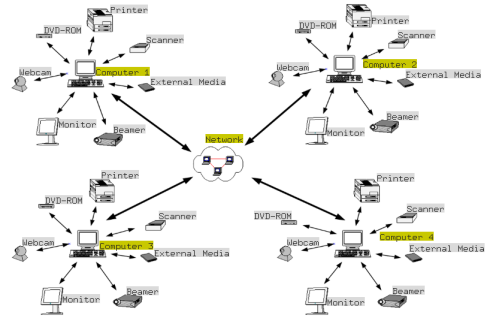
In the best case, the sampling ratio is one (equaling the pixel distance), but to increase performance, higher sampling steps can be used if no thin objects lie within the collision map. Hence, our labeling method is able to respect any visual element simply by transferring its rendered image to particle space. In many cases, the collision map can be gathered in the standard rendering pipeline without an additional rendering step. In Figures 5, 6, 7, 9, and 11 collision maps are used for virtual particle generation. Additionally, Figure 5b shows the generated virtual particles.

Vector-based approach. In addition to the proposed image sampling, a direct conversion of vector graphics into particle space is possible. Hence, the overlapping between labels and e.g., lines or icons can be prohibited without the additional image sampling step — resulting in better performance. For doing so, the contour of a vector object is sampled in screen resolution using virtual particles. A curve for instance, can be handled as shown in Figure 4b.

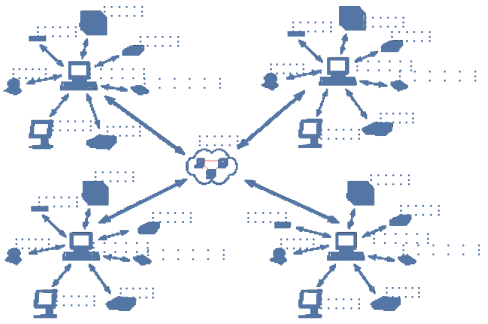


(a) (b)

Fig. 4. Virtual particle generation: (a) A rasterized image is sampled by virtual particles. (b) Vector objects (here: curve) can also be discretized directly by virtual particles. The calculated positions may also be used as candidate labeling positions for the vector object (dashed boxes).



(a)



(b)

Fig. 5. Labeling arbitrarily shaped objects. (a) The point-features are located within each object resulting in a very close distant labeling (lines omitted). (b) The shown conflict particles prevent occlusion of both graphics and other labels. The fixed color co_{empty} is white. Note that particles are drawn slightly enlarged for printing purposes visually closing existing gaps.

Importance Areas. Besides processing labels in order of importance (Section 3.1), the virtual particles themselves may hold importance information. Hence, labels of high priority may placed on illegal areas represented by virtual particles of lower importance.

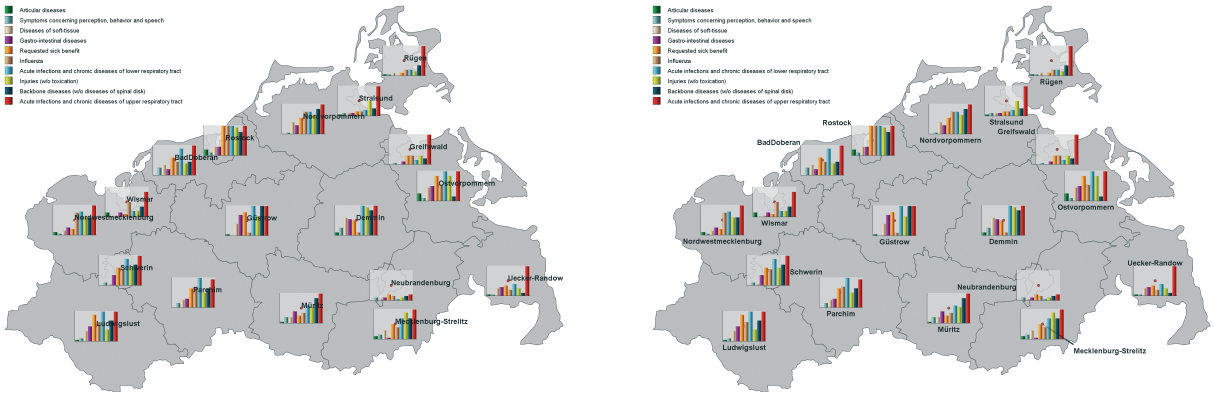
3.4 Labeling arbitrarily shaped objects

The presented labeling approach is not only suitable for the point-feature labeling problem, but also for labeling differently shaped objects (including line- and area-features). In such cases, the determination of possible labeling positions plays the major role — label selection is similar to the methods described in the previous sections. The main idea is to project the labeling of line- and area-features to our point-feature labeling method to determine valid labeling positions. Therefore, in every case the first step is to place virtual particles inside the object to be labeled as described in the previous section.

Within the *image-based* approach, we can simply place a point-feature holding the labeling information of the arbitrary object within the defined prohibited area. Hence, a possible labeling position near the contour is determined by our distant labeling, since steps 1–3 of our labeling pipeline do not succeed. If the found position is close to the object, the connecting line is omitted (see Figure 5).

In the *vector-based* approach, we can use vector graphics information: Each virtual particle discretizing the contour of the object can be used as a candidate for point-feature labeling. In Figure 4b the dashed boxes show such candidate positions. With this approach, we are able to consider the visual extent of point-features (rectangular/circular shape) as well as any other object — mainly resulting in label positions adjacent to the object.

To label area features internally (i.e., labels are not attached to an object, but placed inside), the collision map simply has to be inverted.



(a) (b)

Fig. 6. Labeling a 2D visualization showing 18 data items (health data set northern Germany 2000) with our method. This figure simply demonstrates the usefulness of considering visual elements. Whereas the standard labeling would cover information presented by barchart icons (a), our approach regards those areas (b).

3.5 Placing arbitrarily shaped labels

So far, just the special case of rectangular labels has been discussed. Supplementary, our approach can also be extended to the use of arbitrarily shaped labels. In this case, providing an efficient function that detects conflict particles inside a labels area is of capital importance to achieve high performance. Furthermore, the discretization of a placed label with virtual particles has to be handled efficiently. Hence, replacing these components of Section 3.2 is sufficient to enable the labeling with arbitrarily shaped labels. Obviously, performance decreases when complexity of label shapes increases. To alleviate this, we suggest using bounding boxes reducing the number of complex comparisons.

3.6 Summarizing the general approach

The main steps of our overlap-free labeling respecting other visual elements is summarized as follows:

1. A raster graphics and/or a vector graphics represents the current space occupied by other visual elements. The input is sampled on the fly in screen space to generate a set of conflict particles.
2. The point-features to be labeled are translated to label particles. They are also added to the set of conflict particles.
3. The set of conflict particles is processed in the labeling pipeline to determine labeling positions very fast, respecting or optionally ignoring occupied space (according to importance).

The image-based approach in (1) guarantees independence from the image generating visualization technique whereas the vector-based approach offers a faster and more precise generation of conflict particles. Both methods provide information about occupied space that is considered during labeling. (2) supplies information that is used to avoid the occlusion of point-features (not of their labels). The conflict particles (label particles *and* virtual particles) guarantee an overlap-free labeling, whereas the labeling pipeline (3) provides fast execution and high labeling quality.

4 Application

As described in Section 3, the basic input of our approach is a list of point-features and an image containing visual elements to regard. Hence, our approach is easy to apply in very different scenarios. Figure 6 shows the application of our method to a visualization of health data in northern Germany. Ten different data attributes are visualized by barchart-icons placed on corresponding districts. Although this example is easy to label in principle, it demonstrates the difference of standard point-feature labeling techniques and our approach: In either case, the labels are placed without an occlusion of neighboring labels. But even in this small example, a standard labeling occludes information encoded into the icons, whereas our new method respects these visual elements. For legibility the districts borders are regarded, too.

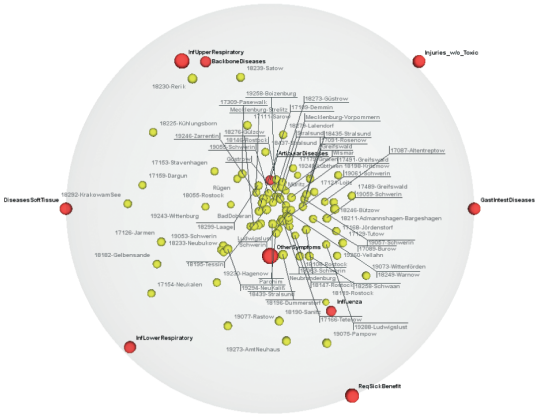
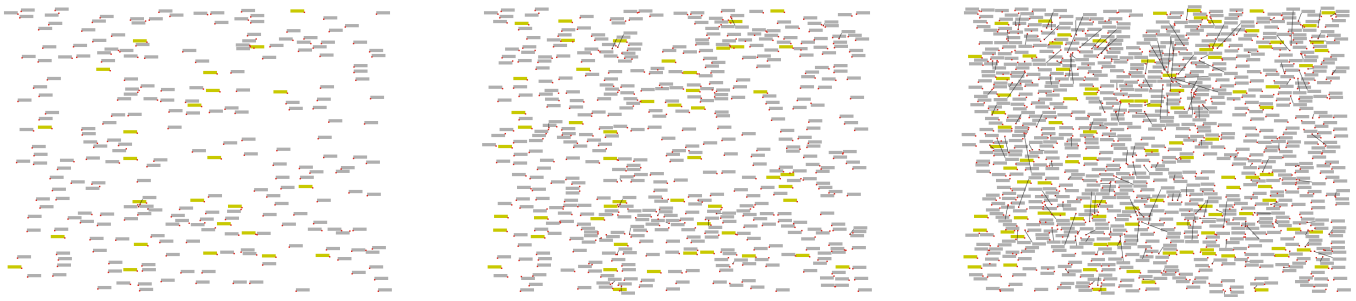


Fig. 7. Labeling a 3D spring-based visualization showing 100 data items (health data set of northern Germany). Distant labeling allows the labeling of very dense clusters. This way all 74 visible items are labeled legible.

But furthermore, our approach can also used to label 3D data within a 2D projection (as in [2]). In Figure 7 for example, our approach is used within a 3D spring-based visualization of the same data set. Since labels are placed in 2D, a 2D projection of 3D positions is used as input. Hence, only the point-features visible in screen space are labeled.

Figure 9 shows our approach within an interactive digital map environment, respecting border contours. Even in cases of common interactions (i.e. rotating, zooming, panning) — as found in today's car GPS navigation systems — the labeling result remains legible. With our approach, a map can be labeled dynamically without an occlusion of important topological information.

We applied our approach successfully to other visualization scenarios like graph visualization and we also implemented an interactive labeling lens (Figure 9c). The lens is realized by generating virtual particles within the lens area. The results are similar to [8] but occlusion is prevented at high frame rates. Using virtual particles of different importance opens up new labeling perspectives. For example, the definition of a semantic lens that reveals important labels inside the lens area and moves unwanted items to its border is easy to implement.



(a) (b) (c)

Fig. 8. Labeling examples of different numbers of randomly located point-features: (a) 250, (b) 500, (c) 1000. Green labels indicate point-features of higher importance which are labeled first. Parameters and results are given in Table 2 and 5.

The good performance of our approach (as we will discuss in the following Section) allows rapid changes to the underlying point-features. Hence, interaction techniques like filtering (appearing and disappearing point-features), the replacement of data sets (e.g., for comparison reasons), and different parameterizations (e.g., weights in a spring based visualization) are easily handled. The performance also permits the instantaneous inclusion, omission, and dynamic change of a collision map and other importance information. Thus our approach definitely supports the interactive exploration of information spaces. Other applications we think of, are the labeling in power wall environments and the labeling in a preview mode of non-real-time applications.

5 Results & Discussion

The presented experiments were performed on a dual-core desktop PC including a 2,6 GHz AMD Athlon 64 CPU, 2GBs of RAM, and a graphics card based on a GeForce 8800 GTX GPU. Using a *single* CPU-core, the times measured include screen space rendering as well as the computations required for labeling.

The comparison of our point-feature labeling method with previously reported approaches is complicated. Most existing approaches use only one adjacent labeling strategy, do not include distant labeling, and do not consider further screen elements. Some fast methods even assume fixed label sizes. Nevertheless, we performed measurements for comparison reasons on randomly distributed point-features with parameters found in literature [7] [23] — making our results at least roughly comparable. All results presented in the following paragraphs represent average measures of 100 uniformly randomized point-feature configurations (examples are shown in Figure 8).

5.1 Number of placed labels

The primary aim of labeling quality is to label as many point-features as possible without any conflict. Our approach labels almost every point-feature as it uses a labeling pipeline testing a large amount of possible label positions. At the same time, the majority of labels are placed as adjacent labels due to pipeline order. Only if the number of point-features compared to available screen space increases, more distant labels are needed.

Labeling approach	Labels to be placed			
	500 in %	750 in %	1000 in %	1500 in %
with Point-Sel.				
Sim. Annealing	99	95	88	74
w/o Point-Sel.				
FALP	100	97	90	—
Tabu Search	99	97	90	—
Sim. Annealing	98	92	82	—

Table 1. Labeling results found in [7] [23] with and without point-selection. The labeling configuration is shown in Table 2.

PC	Labels to be placed							
	500		750		1000		1500	
	%	ms	%	ms	%	ms	%	ms
1	91.54	1	82.57	1	72.93	2	55.66	2
A	95.50	1	89.40	2	81.61	4	65.52	7
D	100.00	2	100.00	4	99.96	10	85.21	59
$\Delta_{D,A}$	4.50	1	10.60	2	18.35	6	19.69	52

Table 2. Percentages of placed conflict-free labels and corresponding labeling times (in ms) for different amounts of point-features. Pipeline configurations (PC) are as follows: (1) labeling results after the first pipeline step, (A) adjacent labeling results after pipeline step 1–3, (D) all pipeline steps including distant labeling, ($\Delta_{D,A}$) indicates the difference of rows (D) and (A). Spiral parameters of (D) are: $r = 150$, $c = 20$, $d = -1$, $m_{max} = 500$. Label size is 30×7 and screen size is 792×612 [7].

In literature the counting of conflict labels is distinguished between with and without point-selection (see [7] for details) — resulting in different conflict numbers. Table 1 shows reference values found in [7] and [23]. Up to 750 point-features, our method results in 100% labeled features and thus no conflict at all (see Table 2). Hence, our results are directly comparable in both counting strategies and moreover, superior to all published approaches.

In case of 1000 and 1500 point-features our approach is superior to the simulated annealing method (upper row in Table 1). A worst-case estimation for our method without point-selection (1000 point-features) is 99% labeled features. Hence, our method outperforms even the approaches in [23] (lower row in Table 1).

Table 3 shows the results of labeling the standard data set *German railway stations* [21] (see Figure 9). As illustrated, our distant labeling facilitates the labeling of all point-features — even under consideration of a collision map. Labeling the standard data set *Munich drill holes* [21] containing 19,461 point features, we achieve a conflict-free solution that is only 3% less than [20](56.8%) and [16](57.3%) — using only adjacent labels (54.3%). Our distant labeling even achieves superior results in that benchmark (77.2%).

Due to an exhaustive testing of adjacent labeling positions and the inclusion of distant labels, we expand the solution space immensely compared to other methods. This way, our method is actually able to label *all* point-features in a stress-test with up to 400,000 point features

PC	without CM		with CM	
	%	ms	%	ms
1	90.71	9	85.25	21
A	93.71	9	88.25	23
D	100.00	15	100.00	33

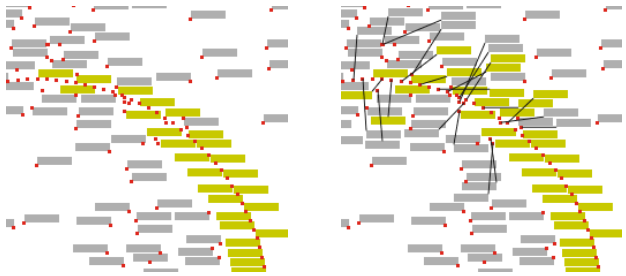
Table 3. German railway stations (Figure 9): Placed conflict-free labels (in %) and performance (in ms) with and without considering a collision map (CM). 366 point-features, resolution: 1752×2148, label size: (characters · 8)×13. Pipeline configurations (PC) as described in Table 2.

(a) (b) (c)

Fig. 9. German railway stations: Applying the complete labeling pipeline (a) without and (b) with respecting border contours. The labeling lens in (c) produces results similar to [8]. It is implemented by simply adding virtual particles into the lens area.

PC	Labels to be placed							
	10,000 Res: 3k×3k		100,000 Res: 8k×8k		200,000 Res: 12k×12k		400,000 Res: 15k×15k	
	%	ms	%	ms	%	ms	%	ms
1	89.9	17	79.5	196	83.6	400	75.6	855
A	94.8	24	87.6	370	90.4	667	84.3	1695
D	100.0	36	100.0	769	100.0	1220	100.0	4167

Table 4. Stress-test: Placed conflict-free labels (in %) and corresponding labeling times (in ms) for large random point-feature sets at different resolutions. Pipeline configurations (PC) and label size as described in Table 2.



(a) (b)

Fig. 10. Comparison of adjacent labeling only and the inclusion of distant labeling. (a) pipeline configuration A and (b) pipeline configuration D as described in Table 2.

at the given resolution (Table 4). At the same time, more than 84% are labeled with adjacent labels.

Results of an importance-driven labeling are given in Table 5. 10% of points-features are declared to be of higher importance (green labels in Figure 8). The distant labeling approach labels all point-features within the higher importance level. Thus, our approach is very practical for e.g., hierarchical datasets or clustered graphs.

The presented results show that our labeling pipeline usually leads to more adjacent labels than distant labels. As a matter of fact, the number of distant labels increases with the number of point-features per screen space (Table 2). The increased screen usage and the additional lines (sometimes crossing) raise clutter but overall legibility is still guaranteed (see Figure 8c). Whereas the adjacent pipeline steps 2 and 3 improve the labeling results only slightly, the presented distant labeling brings the major improvement towards 100% labeled point-features. The difference of adjacent labeling and the additional distant labeling is demonstrated in Figure 10. In distant labeling, the layout is still legible and useful and 100% of point-features are labeled. Eventually, our technique is the only existing interactive approach considering other visual elements without any preprocessing in general PFLP.

No. of important point-features	Important features labeled with	
	adjacent labels	distant labels
10 of 100	100.00 %	0.00 %
25 of 250	99.84 %	0.16 %
50 of 500	99.64 %	0.36 %
75 of 750	99.29 %	0.71 %
100 of 1000	98.33 %	1.67 %
150 of 1500	95.63 %	4.37 %

Table 5. Importance labeling: The table shows how many of the important labels (in %) are placed as favorable adjacent labels compared to distant labels. In either case, all important point-features are labeled. Labeling configuration is given in Table 2.

5.2 Performance

The primary aim to achieve an interactive labeling is to label as quickly as possible. Our approach is very fast and thus allows for a complete label refreshment at interactive frame rates without any preprocessing. The performance decreases if a collision map is included and if many distant labels are used but is still adequate for interactive scenarios.

Performance measurements of our method are given in Table 2. As shown, the performance for labeling up to 1000 point-features lies within a few milliseconds — including distance labeling. To the best of our knowledge, faster results at comparable labeling ratios have yet not been published without requiring any preprocessing.

Note that the distant labeling step is the most expensive of our pipeline. In case of 1500 point-features the distant labeling test has to be conducted for $\approx 35\%$ of all point-features. Hence, labeling performance for 1500 point-features is dropping to 59 ms at pipeline configuration D. However, even this extreme example guarantees interactive frame rates. In very large environments, we outperform the approach in [16]: We label more point-features (200,000 versus 130,000) in less time (1.2 s versus 1.3 s).

Collision map	VP	%	ms
without	1587	100.0 (4)	2
grid	8339	100.0 (25.4)	20
small lens	3353	100.0 (7.4)	17
large lens	7109	98.2 (19.8)	34
grid + small lens	9760	100.0 (29.2)	24

Table 6. Results of examples shown in Figure 11 including an importance map: Number of virtual particles (VP), percentage of placed labels, percentage of distance labels in parenthesis and processing time in ms. Labeling configuration is given in Table 2.

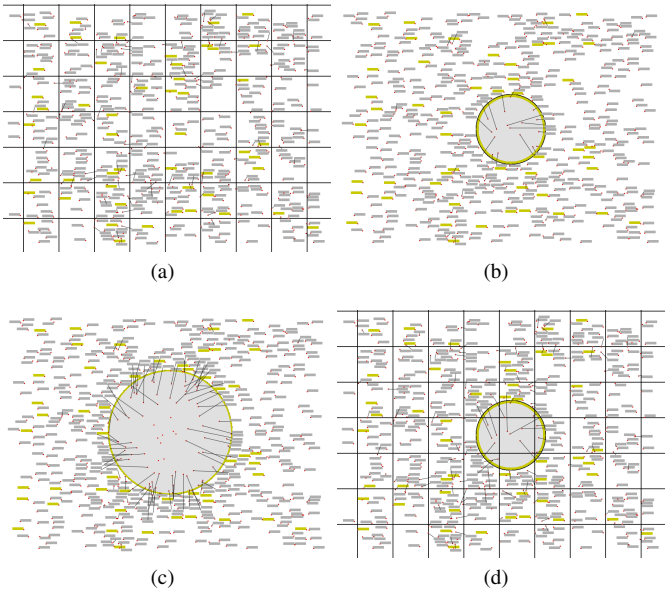


Fig. 11. Labeling 500 point-features (configuration is shown in Figure 8b) under consideration of different collision maps: (a) grid, (b) small lens, (c) large lens, (d) grid & small lens. Results are given in Table 6.

The inclusion of different visual elements decreases performance. Figure 11 shows the exemplarily consideration of different collision maps during labeling of 500 point-features. Measurements are given in Table 6. For the used examples, the collision maps reduce performance by a factor of ten, but still being well suited for interactive applications. The *German railway stations* data set can be labeled entirely within 33 ms (Table 3) — resulting in interactive frame rates of 30 FPS with occlusion-free contours (see Figure 9).

Optionally labeling only the user field of view defined by the available 2D screen space, reduces the amount of necessary calculations significantly. This would speed up our labeling even further.

Overall, our method is fast enough for interactive scenarios with thousands of point-features comparing favorably to known approaches. Even if distant labeling and collision maps are included, convincing labeling performances are achieved.

6 Conclusion & Future Work

The presented particle-based approach allows for a very fast labeling of thousands of point-features without any preprocessing. In combination with a collision map, arbitrary visual elements may be marked in order not to be covered by labels. Thus, important information encoded in non-textual elements may be preserved of occlusion. This trait cannot be achieved by any existing interactive labeling approach that is free of preprocessing. Our method can easily handle arbitrary label sizes and label shapes. Additionally, our method allows the labeling of objects that are no point-features (e.g., line- and area-features). Summing up, our approach is very attractive for highly interactive and dynamically changing environments.

Due to the use of distant labels, our method achieves high labeling ratios and can label point-features in situations where adjacent labeling methods would fail in principle. However, the distant labeling introduces additional lines between labels and point-features to show their relation. To alleviate the resulting clutter, further investigations include the minimization of distant labels, e.g., by using additional heuristics. To reduce crossing lines (Figure 9c), e.g., an improved space sampling function could be used. For example, parameter d of our spiral function allows the definition of elliptical spirals.

If CPU time is left, our labeling result may also be refined successively using enhanced heuristics. For example, genetic algorithms may merge results of different labeling pipelines.

Ongoing work also investigates the reduction of flickering effects in interactive environments resulting from per-frame-labeling (e.g., by animations or coherence-thresholds between frames)

References

[1] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'94)*, pages 313–317, 1994.

[2] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3d illustrations. *Journal of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'05)*, 13(1):1–8, 2005.

[3] K. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):773–780, 2006.

[4] B. A. Bell and S. K. Feiner. Dynamic space management for user interfaces. In *Proceedings of the 13th annual ACM Symposium on User Interface Software and Technology (UIST '00)*, pages 238–248, 2000.

[5] B. A. Bell, S. K. 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 (UIST '01)*, pages 101–110, 2001.

[6] S. Bruckner and E. Gröller. Volumeshop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization (VIS'05)*, pages 671–678, 2005.

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

[8] J.-D. Fekete and C. Plaisant. Excentric labeling: Dynamic neighborhood labeling for data visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'99)*, pages 512–519, 1999.

[9] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proceedings of the 7th Annuual ACM Symposium on Computational Geometry (SCG'91)*, pages 281–288, 1991.

[10] G. Fuchs, M. Luboschik, K. Hartmann, K. Ali, T. Strothotte, and H. Schumann. Adaptive labeling for interactive mobile information systems. In *Proceedings of the 10th International Conference Information Visualization (IV'06)*, pages 453–459, 2006.

[11] S. A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*, 9(1):5–17, 1982.

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

[13] K. G. Kakoulis and I. G. Tollis. On the edge label placement problem. In *Proceedings of the Symposium on Graph Drawing (GD'96)*, pages 241–256, 1996.

[14] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin. Interactive cutaway illustrations of complex 3d models. *ACM Transactions on Graphics*, 26(3):31, 2007.

[15] J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS, 1991.

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

[17] I. Petzold. *Beschriftung von Bildschirmkarten in Echtzeit – Konzept Und Struktur*. PhD thesis, University of Bonn, Germany, 2003.

[18] I. Petzold, G. Gröger, and L. Plümer. Fast screen map labeling – data-structures and algorithms. In *Proceedings of the 23rd International Cartographic Conference (ICC'03)*, 2003.

[19] G. Raidl. A genetic algorithm for labeling point features. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, pages 189–196, 1998.

[20] S. Roy, S. Bhattacharjee, S. Das, and S. C. Nandy. A fast algorithm for point labeling problem. In *Proceedings of the 17th Canadian Conference on Computational Geometry (CCCG'05)*, pages 155–158, 2005.

[21] A. Wolff and T. Strijk. Map labeling [website]. *http://i11www.iti.uni-karlsruhe.de/~awolff/map-labeling/, (accessed 8-1-2008)*.

[22] M. Yamamoto, G. Camara, and L. A. N. Lorena. Tabu search heuristic for point-feature cartographic label placement. *GeoInformatica*, 6(1):77–90, 2002.

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