CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Force-Based Label Placement for Dynamic Data

Master's thesis in Computer Science - Algorithms, Languages and Logic

SEBASTIAN EKMAN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Force-Based Label Placement for Dynamic Data

SEBASTIAN EKMAN

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Force-Based Label Placement for Dynamic Data
SEBASTIAN EKMAN

Supervisor: K V S Prasad, Department of Computer Science and Engineering
Advisor: Mikael Rittri, Carmenta AB
Examiner: Carl Seger, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

Force-Based Label Placement for Dynamic Data
SEBASTIAN EKMAN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In this thesis a new method for placing graphical labels on moving objects in real-time maps is presented. The intended use case is command and control systems, such as software used by emergency response dispatchers. The method expands on previous work on force-based labeling. To ensure that the readability of labels is high and sudden movements are limited, new force components are included. Most notable of these are forces that act on labels in the direction of travel for another label, in such manner that space is made for it. The method presented is then tested on several relevant data sets that could make up future use cases and some simulated data sets. The tests are used to determine possible parameter values to be used, something that is shown to be dependent on the type of data and personal preferences. Also, the tests show that the method preforms sufficiently fast to be used in real-time applications.
A video demonstration can be found on `https://youtu.be/GiUzVvo0f1s`.

Keywords: Labeling, GIS, annotation, real-time, force-based, coherent, dynamic, placement.

Acknowledgements

I would like thank my supervisor at Chalmers, K V S Prasad, for his support and feedback, and my examiner Carl Seger for his feedback. Also a big thank you to my co-workers at Carmenta, and especially to my advisor Mikael Rittri, for their support, feedback, and the company they have kept. Finally, I would like to thank my family for the support and thank Carina for the proofreading of this report.

Sebastian Ekman, Gothenburg, August 2018

Contents

List of Figures

1

Introduction

1.1 Background

The business enterprise Carmenta develops software for displaying maps and supplying other geospatial information. This software is often used in real-time applications, such as command and control systems for emergency response organizations and the defense industry. One of its users is the Swedish emergency response dispatcher, SOS Alarm. They are the client where Carmenta identified the problem to be tackled by this thesis. There is also reason to believe that a good solution to the problem presented here can prove useful in similar applications.

Emergency response dispatchers use their software to display geographical information on computer screens. It is important to note that this is data updated in real-time. The information includes the location of both emergency vehicles and the actual incidents. These locations are represented by points on a map and all have some associated information. This information could be an identifier, a status, a priority etc. This is displayed in a label that is placed on the map, next to the point it represents.

For dispatchers, it is very important that the clarity of the information presented is high as their work requires them to make critical decisions in a small amount of time. They spend all of their working hours watching this map, and in certain cases, it could even be that a quick and correct decision is the difference between life and death. Thus, the information must be easy to read, and the map view must not contain any unnecessary clutter. The view should clearly convey the locations and the associated information previously mentioned. The problem becomes more prominent when a lot of information is to be displayed in a small area. An example of this could be in the close proximity of a severe accident, to where a lot of emergency vehicles are dispatched.

At present, Carmenta places labels next to point features using what they call a static method. Labels are placed in predefined locations relative to the point features. They are placed in a default location relative to the points, but if they overlap with other labels they are moved to other pre-defined positions, chosen by a heuristic. This method allows the labels to move between any of these relative locations in each update of the view. This sudden movement of labels is troublesome, it causes the dispatcher to lose track of the information, and it is what this thesis

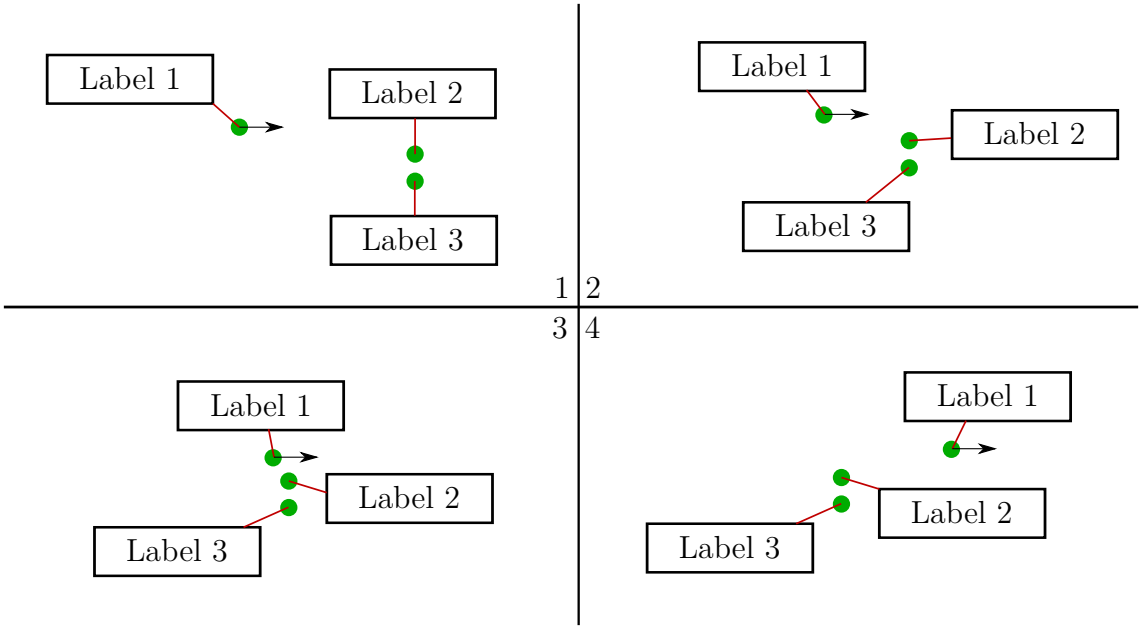

Figure 1.1: An illustration of what a smooth labeling of point features could look like. In this case, the point with label 1 is moving sideways in such way that label 2 must be moved to avoid overlap. This should be viewed as a continuous process shown in 4 discrete steps.

will address.

1.2 Purpose

The primary goal of this thesis is to present a method for placing labels next to point features, whose positions are updated in real-time, for use in geographic information systems (GIS). Real-time refers to the fact that the positions of the points are updated continuously, without any knowledge about their future position. The placement should be done in such a way that the labels are easy to read, while also moving smoothly. Figure 1.1 illustrates what this could look like.

1.2.1 Requirements

The main requirement is to find a method that ensures labels to move smoothly on the screen and limit sudden movements. At the same time, an implementation must not induce any significant delay between that of a point's position becoming known and the instance that the point is placed at that position on the screen, with its label. Under these requirements, as many labels as possible should be shown while the screen is still able to update sufficiently often for the movements to appear smooth.

Zooming and panning of the map view should be possible. When doing so, it is

still important that the movement of labels are smooth. The panning and zooming means that there are situations where new labels may suddenly come in to view. The zooming means that the size of the labels changes relative to the distances on the map, but their size on screen does not.

The overlaps between labels should be limited. During movement, minor overlaps can be tolerated, but any prolonged overlaps should not be. This means that in a situation where no point features are moving, the labels should also be placed such that all visible labels are readable, not intersecting each other.

Labels should also not overlap the point features themselves. This is especially true for points that have their labels showing and are of higher priority. For other points, overlaps can be tolerated.

1.2.1.1 Label Properties

Labels are all rectangles, with arbitrary side lengths, and are axis-aligned. However, they are not all of equal size. Labels should be able to change size at any moment, as the content of the labels changes.

All labels have an order of importance. In situations where a label cannot be placed without causing overlap, we can allow the labels with lower priority to be hidden. That action should however be kept to a minimum as we want to show as many labels as possible. Similarly, if it comes to the situation that a label cannot be moved smoothly to a new position, a sudden movement might be necessary. Alternatively, a label with lower priority must be hidden.

It should be possible to add and remove points with labels at any time. Also, the priority order of the labels can change.

The use of leading lines between the point features and labels have been identified by Carmenta to cause clutter. If used, they should be as short as possible. In situations where labels only fit far away from their feature points, it could be a better solution to hide the labels instead.

1.2.1.2 Performance

It is important that the resulting algorithm used to solve the problem can perform in such a way that the labels can be updated continuously given a typical workload and computing power. The target should be to run it on standard desktop computers.

Carmenta has expressed that there is no reason for delays to exceed one second. This is based on the fact that the real-time position data is typically updated once per second. For the actual updating of the screen, it is desirable for it to be done at a rate of 60 times per second. This is since a typical LCD-screen refresh at that rate. The update rate is, however, not a hard criterion as long as smooth movement can be achieved. If a method cannot keep up, it should rather limit the number of labels shown.

An update rate of 60 frames per second would mean that one update must not take more than 16.67 ms. It should however also be factored in, that label placement usually is only a part of a larger application, and things like map rendering also consumes time. Therefore, even lower time consumption would be desirable.

SOS Alarm typically looks at one dispatcher area at a time. If a whole area is to be displayed on the screen at once, we have a ballpark of 1000 labels to be considered for placement, if left unfiltered. To get this number, Carmenta looked at a recording of a real-world scenario, supplied by SOS Alarm. It was also pointed out that it is not necessary to consider all the labels for placement. As there is a priority order for them, a method can choose to only consider so many that the performance requirements can be met. For the developed method to be comparable to the current ones used by Carmenta, it should be able to place a minimum of 100 labels or more.

1.2.2 Limitations

There are also some limitations to the scope of the thesis. Most important is that only placing labels on point features of maps will be considered. Other typical map features are lines and areas, for example often used to represent roads and bodies of water, respectively. While it might be of interest to place labels in real-time on these kinds of features as well, they will not be considered here.

Labels should only avoid overlapping other labels and the point features they are referring to. In some applications, it might be of interest to make sure that labels do not overlap other important parts of a map. This is not a requirement given by Carmenta, and if it was, it would increase the complexity of the problem.

There should be no other user control in the actual placement of labels. Some examples of such user controls could be to highlight certain labels, so that they are always shown, and not hidden by the placer due to a lower priority. It could also be that the user is given the ability to lock a label in a specific location relative to its point so that it does not move as much, which is also not a requirement.

One goal is to improve the readability of labels, in this case by making them move more smoothly. We will not consider whether this is entirely true or not. To decide if the end result actually is something that improves the readability is a topic of human interaction and psychology. It would require different expertise, a study on its own, and is best left for future work.

2

Theory

This section presents earlier work on the topic of label placement and how it relates to the problem to be solved here. Further, a more in-depth description of the method our work is primarily based on is given. Finally, definitions and notations are put in place.

2.1 Previous Work

There are various existing methods for placing labels on maps [1]. Most of them focus on the problem of placing labels on a static map, where information will not change. This could, for example, be printed maps or pre-rendered maps to be shown on computers. In this case, the placement only has to be done once and naturally time consumption is not as critical as it would be in a real-time scenario. To place as many labels as possible, without them overlapping, is typically a problem that is NP-complete [2]. For this reason, many methods try to find an approximate solution. However, they are still not all suited for labeling ever-changing data in real-time.

2.1.1 Static Methods

As already mentioned, there are some methods that can place labels for real-time data. These methods typically start by defining potential placements for each label and are then followed by two stages. First, we calculate which potential positions might overlap other labels, i.e. overlaps other potential positions [3]. A cost is associated with each position based on how many other positions it overlaps. In the second stage, we can then place labels starting with the label that has a potential position with the lowest cost, choose that position, and repeat. This is the basis of the method Carmenta is currently using, which has the unwanted sudden movements.

This method is the basic principle of many placement methods which aims to be usable in a real-time application and it had its introduction in [4]. It has been expanded in following work such as [5], where a pre-processing stage was added to resolve the conflicting labels more quickly. This made the algorithm work in

interactive applications, allowing for zooming and panning of maps, but still have the unwanted sudden movements of labels.

2.1.2 Dynamic Methods

A master thesis has been done in collaboration with Carmenta before, addressing the problem, and the work was presented in a report, Dynamic Label Placement for Moving Objects. This method was mostly based on the method of predefined locations earlier mentioned and expanded the collision detection to act in time as the labels move [6]. The movement did however, require knowledge about the labels positions during a time interval and therefore induce a delay to be able to interpolate positions. This induced delay is the reason Carmenta has deemed the method unsuitable for its applications. Apart from the delay, the method produces relatively good results, according to the report.

There are also other methods of combining the traditional static labeling methods, known to work in real-time, with some method for moving the labels in a time coherent, continuous, manner [7], [8]. What these methods have in common is they make use of position data over time, and would thus need to introduce some delay in the placement.

There are however, a few methods that do not require this knowledge, such as methods for interactive label placement in 3D environments [9], [10], but the performance of such algorithms have proved too low to place much more than 20-40 labels in interactive applications [11].

2.1.3 Force-Based Methods

The concept of force-based label placement was first introduced in 1982 [12]. The idea behind force-based methods is that labels are first placed in their most desired position. In an iterative process, they are then moved apart from each other in small steps such that any overlapping is resolved. This makes it possible to have labels moving smoothly on the map.

A report from 2003 presents a forced-based method that is stated to be suitable for some dynamic real-time applications [13]. The report does not however, state to what extent this is possible, or even exactly how it would perform in a dynamic situation. All labels are given a repelling force acting on all other labels, and they are moved based on this force. The approach introduced improvements to earlier methods by solving local minimum situations by simulated annealing. This affects the placement so that it is no longer coherent in time.

The article Temporally Coherent Real-Time Labeling of Dynamic Scenes expand by Vaaraniemi et al. on the report from 2003 [11]. It presents methods for labeling both point and line features. To achieve real-time performance, for zooming and panning, the majority of calculations were done on a GPU [11]. The results suggest the method should be of great interest here.

There are some key differences compared to the problem presented in this report. For example, the method of Vaaraniemi et al. does allow labels to obstruct the features themselves. Even more important is the fact that the method is not tailored to the dynamic case where the features are moving. While it might still work in the sense that labels are not obstructing each other, it might not allow for smooth movement. Therefore, some modifications will have to be put in place.

The requirements of Vaaraniemi et al. align well with ours, which summarized, are as follows:

- Real-time labeling of point features
- Run-time placement of new labels
- Scaling up to several hundreds of labels
- Temporal coherence
- Scaled labels with priorities

They also had some further requirements which do not apply here:

- Labeling of line and area features
- Support for rotated labels

The type of labeling most aligned with what we are trying to achieve is what Vaaraniemi et al. calls circling annotations. In this case, three kinds of forces are applied to all labels in each frame update. These are a spring force, a collision force and a friction force. The forces are applied in each update of the screen, which creates the time-coherent smooth movement [11].

The purpose of the spring force is to pull the label close to the point feature it is annotating [11]. The collision force is a force acting on the label in case it is overlapping another label, in such manner that they repel each other [11]. The friction force is put in place to stabilize the system [11].

2.2 Mathematical Notation

In this paper, we assume that the area where we place our labels is a Euclidean plane. This plane is the computer screen with pixels as coordinates. Labels and point features are described using vectors.

The positions of labels can be described as vectors from an origin position to the center of the label. We denote these as $\mathbf{l}_i$. The positions of all point features is described in the same way, denoted $\mathbf{p}_i$, where $\mathbf{p}_i$ is the positions of the features annotated by the label at position $\mathbf{l}_i$.

As all labels are axis-aligned rectangles their extent can be described by only one vector. This would be a vector defined as the difference in the position of one corner of the rectangle to that of the opposite corner. To simplify calculations we only

allow this vector to have positive components, which would be the absolute value of each component in such a difference vector.

This paper uses the absolute value notation $|\cdot|$ to denote the component-wise absolute value of a vector. Further, the max function of a vector represents the largest component of the vector. Finally, the $\|\cdot\|$ is used to denote the 2-norm, i.e. the Euclidean length, of a vector.

3

Methods

In this section, we will first present the mathematical method used for our general label placement. Further, the method for solving the problem of which labels should be hidden and which should be showing, is given together with how labels are given a prioritization order. Finally, we present our evaluation methods.

3.1 Mathematical Model

The approach chosen as a basis is primarily based on the forced based method by Vaaraniemi et al. The problem statement does differ a bit between that study and ours, so some modifications are put in place. Our problem has the most in common with what was called circling annotations. This is the reference for our work.

Vaaraniemi et al. had three forces acting on each label. The first one was a collision force for avoiding label overlap. The second one was a circle, or spring, force placing the label close to its point feature. The last one was a friction force for avoiding pendulum effects caused by momentum. We will make use of the same kind of forces, some calculated in different ways.

It was not a requirement by Vaaraniemi et al. for a label to avoid other point features other than the one it is annotating [11]. Therefore, we will introduce a new force to meet that requirement. This will be done in a similar fashion as the collision force between labels.

As smoothness also is a major requirement, some additional forces are added to ensure smooth movement. This includes forces that try to predict the movement of other labels and features. In addition to that, we have some weaker forces acting at greater distances. The reason for adding all these forces is because we have moving features, unlike the problem stated by Vaaraniemi et al.

In comparison, we can also choose to hide certain labels to avoid overcrowding. This will be realized by a scoring system penalizing overlap and other unwanted behaviors. This scoring system can also make use of the prioritization order of the features.

For our method, there are some data that must be supplied for each placement update. A list of what data this is can be found in the table below. Most notable

in this list is that we must always supply a label position. For a smooth movement through consecutive updates, the reason for this is obvious, but an initial value for the first update must also be supplied. We will set the initial position of all labels to be to the upper-right of the point feature. One can imagine that it would be possible to use a more traditional static label placement method for this initial value.

Description	Symbol
Point Feature Positions	$\mathbf{p}$
Point Feature Velocities	$\mathbf{v}_p$
Label Positions	$\mathbf{l}$
Label Velocities	$\mathbf{v}_l$
Label Sizes	$\mathbf{e}$
Label Scores	-
Label Priorities	-
Label Visibilities	-

In most cases, these values will just be carried over from the last update. However, one situation where this is not the case, is when zooming and panning as described later. Also, label sizes and priorities are data that is not changed by the label placement, but rather application data only that can be changed between updates.

3.1.1 Label Collision Force

To avoid overlap between labels the first force introduced is acting on labels that get within a certain distance of each other, such that they repel. This only takes into consideration the positions of the labels, their size, and a parameter deciding at what distance the force should act.

Calculating the distance between the labels is straightforward, as it is known that all labels are rectangles, oriented in the same axis. We do however simplify the calculation even further by only finding the distance in the axis where it is the largest. This reduces the need for square root calculations, which in some cases can be considered heavy to calculate by the computer.

The formula used for the distance d between label i and j is

$$d(\mathbf{l}_i, \mathbf{l}_j, \mathbf{e}_i, \mathbf{e}_j) = \max(|\mathbf{l}_j - \mathbf{l}_i| - \frac{1}{2}(\mathbf{e}_i + \mathbf{e}_j)),$$

where $\mathbf{l}$ is the center position of a label and $\mathbf{e}$ is the extent of a label.

If this distance d is lower than 0 we have a collision between the labels. We also introduce a parameter for at what distance the force will act. Call this value v, where

$$v^{i,j}_{\text{collision}} = \min(\frac{d(\mathbf{l}_i, \mathbf{l}_j, \mathbf{e}_i, \mathbf{e}_j)}{m_{\text{collision}}} - 1, 0).$$

This means that if the distance between the labels is lower than a certain value, the force will be applied and otherwise not.

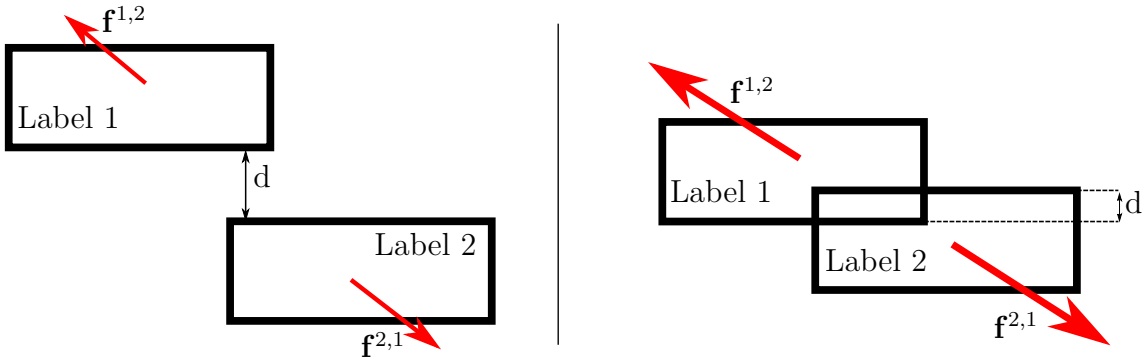

Figure 3.1: This illustrates how the collision force acts when labels are close to each other (left) and when the labels are intersecting (right). In the latter case, the value d would be negative, but in both cases, $v_{i,j}$ would be negative.

The force applied will be proportional to v and will be applied in the direction of the difference vector of the label's positions, as

$$\mathbf{f}^{i,j}_{\text{collision}} = v^{i,j}_{\text{collision}} \frac{\mathbf{l}_i - \mathbf{l}_j}{\|\mathbf{l}_i - \mathbf{l}_j\|}.$$

This is a calculation that must be done for every label against all other labels. An illustration of this force can be seen in figure 3.1.

Introduced is also a weaker force acting on a greater distance. The purpose of such force is for labels that are closing in on each other during movement to try avoiding each other early. With this force being weaker it will still be possible for the labels to be close to each other if stronger forces mandate so. How this is weighted into the total force can be seen in section 3.1.6.

This weaker force is calculated in exactly the same fashion as the main collision force, but with the parameter $m_{\text{weak collision}}$ being larger than $m_{\text{collision}}$. We call this force $\mathbf{f}^{i,j}_{\text{weak collision}}$.

3.1.2 Feature Collision Force

One force is introduced to avoid labels overlapping the features that are to be labeled. This is done in an almost identical way as the overlap between labels. The difference is that the calculation for a label is done against all point features instead.

The distance function in this case is

$$d(\mathbf{l}_i, \mathbf{p}_j, \mathbf{e}_i) = \max(|\mathbf{l}_i - \mathbf{p}_j| - \frac{1}{2}\mathbf{e}_i),$$

where $\mathbf{l}$ is the label position, $\mathbf{p}$ is the feature position and $\mathbf{e}$ is the extent vector of the label.

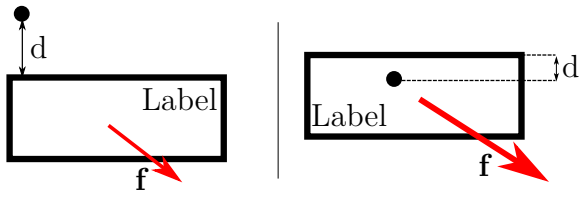

Figure 3.2: This illustrates how the feature collision force acts when a label is close to a point feature (left) and when the label is overlapping the point feature (right). In the latter case the value d would be negative, but in both cases $v_{i,j}$ would be negative.

Just as before we introduce a parameter, m_{feature}, and apply the force only if the new value is below 0. This gives the value

$$v^{i,j}_{\text{feature}} = \min(\frac{d(\mathbf{l}_i, \mathbf{l}_j, \mathbf{e}_i)}{m_{\text{feature}}} - 1, 0).$$

The force is also calculated in the same manner,

$$\mathbf{f}^{i,j}_{\text{feature}} = v^{i,j}_{\text{feature}} \frac{\mathbf{l}_i - \mathbf{p}_j}{\|\mathbf{l}_i - \mathbf{p}_j\|}.$$

This force is illustrated in figure 3.2.

We also introduce a weak version of this force, $\mathbf{f}^{i,j}_{\text{weak feature}}$, just as for the label collision force.

3.1.3 Movement Prediction

We introduce a force with a similar goal as the weak forces mentioned. To achieve the smooth movement that we want for a label moving in a certain direction, we move other labels in said direction out of the way. This is done so that when a label approaches, it tries to make room for itself where it is likely to be in the future.

In its simplest form, this is done by applying a force acting on other labels perpendicular to the velocity vector of the moving label. However, consideration must also be taken to the extent of the labels involved. When a label is larger the force must be applied on labels further away from the center point of that label. This becomes even more involved when the rectangular labels have a large difference in side length.

When doing this kind of avoidance between two labels it is only relevant to consider the relative velocity between the labels. Therefore, the relative velocity for label i and j will be defined as

$$\mathbf{v}_{\text{diff}} = \mathbf{v}^l_j - \mathbf{v}^l_i.$$

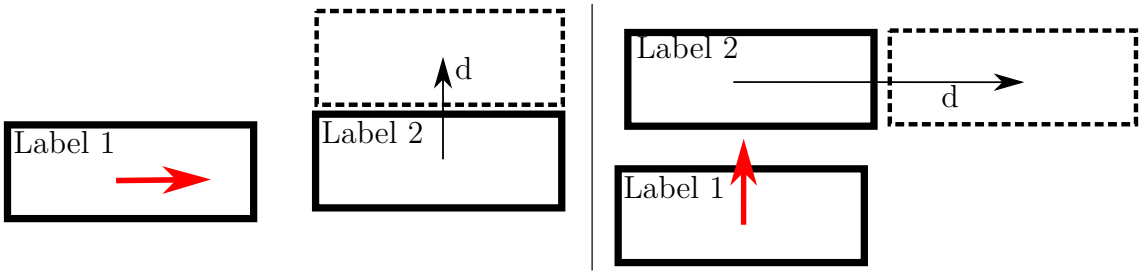

Figure 3.3: An illustration on how labels have to move further to avoid each other depending of movement direction and shape. The red arrows show the movement direction of label 1 and the dashed rectangle is the future position of label 2.

Consider labels with a width greater than their height. If such a label is moving horizontally, labels in its path only have to move a small distance to avoid a collision, i.e. the distance is the height of a label. When the label is moving vertically, labels in its path must move a greater distance, the width of a label. This is illustrated in figure 3.3. This means that the distance at which the force is applied is dependent on the direction of travel and the combined extent of two labels, given as

$$d_\perp = \frac{(\mathbf{e}_i^x + \mathbf{e}_j^x)|\mathbf{v}_{\text{diff}}^y| + (\mathbf{e}_i^y + \mathbf{e}_j^y)|\mathbf{v}_{\text{diff}}^x|}{2\|\mathbf{v}_{\text{diff}}\|^2}.$$

This is only the distance perpendicular to the velocity vector. The parallel distance will be independent of the velocity vector. Consider the same cases again. When the label is moving sideways the parallel distance should be greater as the labels occupy much of the space between the center points of the labels. One could then argue, that when a label is moving vertically, the parallel distance should be smaller as the labels occupy less space in that direction. However, as the distance the labels should be moved, is greater the force should act on a larger distance to ensure that the labels begin moving out of the way earlier. Therefore the parallel distance is defined as

$$d_\parallel = \max(\frac{\mathbf{e}_i + \mathbf{e}_j}{2}).$$

The distances will be used as a scaling matrix transform defined as

$$D = \frac{1}{m_{\text{label predict}}} \begin{bmatrix} 1/d_\perp & 0 \\ 0 & 1/d_\parallel \end{bmatrix},$$

where $m_{\text{label predict}}$ is a scaling parameter for at what distance the prediction force should act.

The movement prediction force is given by a vector function defined as

$$\mathbf{y}(\mathbf{x}) = \begin{bmatrix} 0 \\ \text{sign}(\mathbf{x}_y)\mathcal{H}(\mathbf{x}_x)\max(0, 1 - \|\mathbf{x}\|) \end{bmatrix},$$

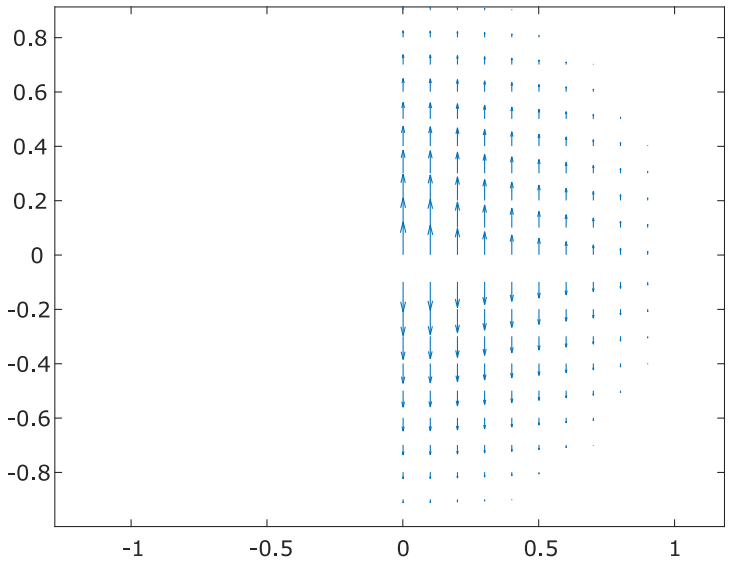

Figure 3.4: This figure visualizes the vector field for the movement prediction force. The function is defined under the assumption that the velocity of a label is $(1, 0)$.

where $\mathbf{x}$ is a correctly scaled position vector rotated to the movement direction and $\mathcal{H}$ is the Heaviside step function. The function is visualized in figure 3.4.

To transform our position vectors to the movement direction and back, the rotation transform is defined as

$$R = \frac{1}{\|\mathbf{v}_{\text{diff}}\|} \begin{bmatrix} \mathbf{v}_{\text{diff}}^x & -\mathbf{v}_{\text{diff}}^y \\ \mathbf{v}_{\text{diff}}^y & \mathbf{v}_{\text{diff}}^x \end{bmatrix}.$$

This makes the complete prediction force for label i, given that label j is moving, defined as

$$\mathbf{f}_{\text{label predict}}^{i,j} = \|\mathbf{v}_{\text{diff}}\| R\mathbf{y}(DR^{-1}(\mathbf{l}_i - \mathbf{l}_j)).$$

Observe that the strength of the force is dependent only on the relative speed of the labels. This complete function is visualized in figure 3.5.

A similar force, $\mathbf{f}_{\text{point predict}}$, is calculated for the movement of the point features as well. It is calculated in the same way with the points asserting a force on the labels that are in a feature's direction of movement. Also, we only have to consider the size of the label.

3.1.4 Pulling Force

To ensure that a label stays close to the point feature it is annotating, we introduce a pulling force. This is a force directing the label towards the feature, proportional

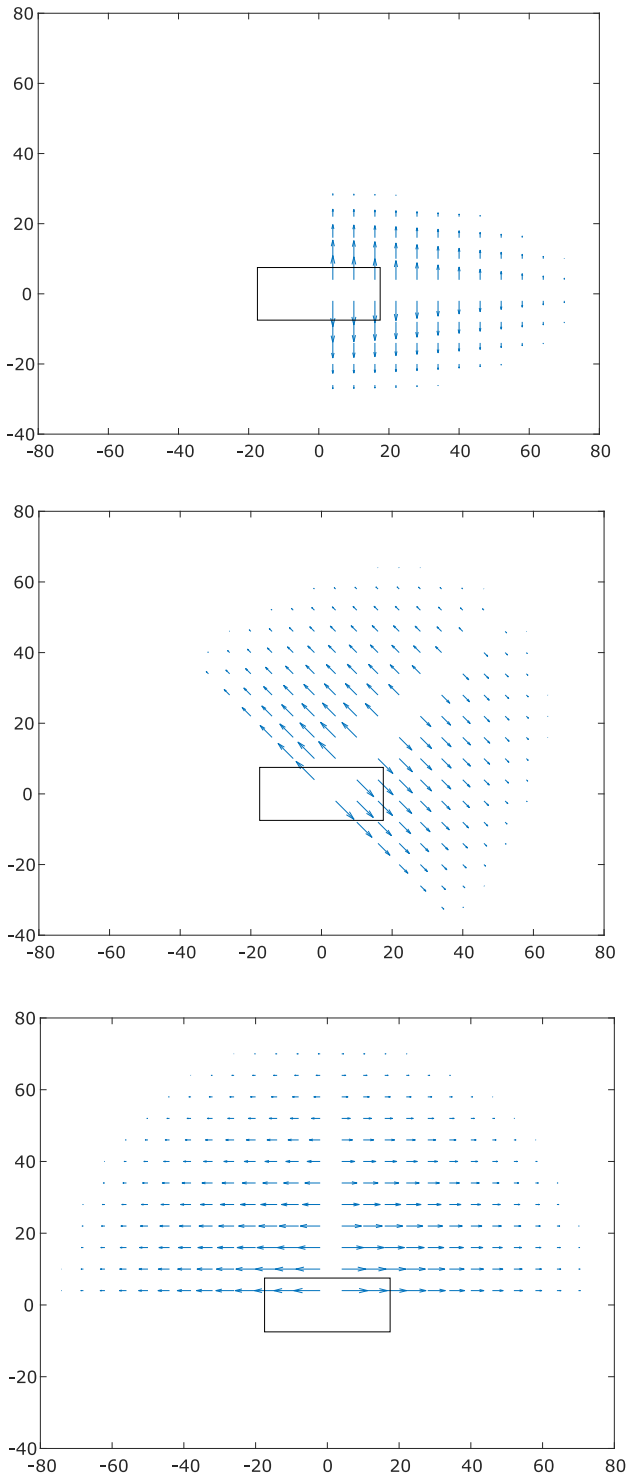

Figure 3.5: These vector fields depict the movement prediction force applied on a label depending on its center position. The three different images illustrates different movement directions, to the right, diagonally, and up. This assumes the label creating the force is placed at $(0, 0)$ and that the labels sizes are both assumed to be $(35, 15)$ and $m_{\text{label predict}}$ is 2.

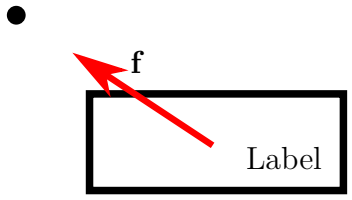

Figure 3.6: Illustration of how the pulling force acts on a label.

to the distance between them. It is defined as

$$\mathbf{f}^i_{\text{pull}} = -\ln(\|\|\mathbf{l}_i - \mathbf{p}_i| - \frac{\mathbf{e}_i}{2}\| - m_{\text{pull}} + 1)\frac{\mathbf{l}_i - \mathbf{p}_i}{\|\mathbf{l}_i - \mathbf{p}_i\|},$$

where $\mathbf{p}$ is the position of the point feature, $\mathbf{l}$ is the position of the label, and m_{pull} is a parameter to adjust the minimum distance at which the force will be applied. The force will only be applied if

$$\max|\mathbf{l}_i - \mathbf{p}_i| - \frac{\mathbf{e_i}}{2} - m_{\text{pull}} > 0,$$

i.e. when the label is sufficiently far away from its point feature. When it is too close the feature collision force will be in effect instead.

The reason for using the logarithmic function is an attempt to avoid a potential problem when the label is far from the feature. This is when a label is far away from its feature point, and there are other labels between them. If the pulling force is too strong in this case, there is a possibility that labels are squeezed together, risking labels being hidden. This was a problem experienced in early testing, using a linear scaling. Making the force weaker at greater distances might allow the label to find a way around the other labels instead.

The force is illustrated in figure 3.6.

3.1.5 Friction

To stabilize the system, a friction force is applied to each label. It depends only on the velocity of the label and the point feature. It is defined as

$$\mathbf{f}^i_{\text{friction}} = -(\mathbf{v}^l_i - \mathbf{v}^p_i),$$

where v is the velocities. This should have a dampening effect on the labels except when they are moving in the same direction as their point features, allowing them to follow their points without lagging behind due to friction.

3.1.6 Combining the Forces

When all forces have been calculated for all labels we will combine them into one total force acting on each label. While doing this we introduce a constant scaling factor for each force such that we can weigh them against each other. We call these c with matching subscript for each force.

The summarized force for label i will be

$$\begin{aligned}\mathbf{f}^i_{\text{total}} = &\sum_{j\neq i}(c_{\text{collision}}\mathbf{f}^{i,j}_{\text{collision}} + c_{\text{weak collision}}\mathbf{f}^{i,j}_{\text{weak collision}} \\ &+c_{\text{label predict}}\mathbf{f}^{i,j}_{\text{label predict}} + c_{\text{point predict}}\mathbf{f}^{i,j}_{\text{point predict}}) \\ &+\sum_{j}(c_{\text{feature}}\mathbf{f}^{i,j}_{\text{feature}} + c_{\text{weak feature}}\mathbf{f}^{i,j}_{\text{weak feature}}) \\ &+c_{\text{pull}}\mathbf{f}^{i}_{\text{pull}} + c_{\text{friction}}\mathbf{f}^{i}_{\text{friction}}.\end{aligned}$$

The acceleration $\mathbf{a}_i$ for a label is then said to be equal to $\mathbf{f}^i_{\text{total}}$. This could be thought of as using Newton's second law of motion with a mass of 1.

The next step in the calculation is to calculate the velocity for each label. This is done using the Euler integration method. A new velocity for a label, $\mathbf{v}^{l'}_i$, is calculated as

$$\mathbf{v}^{l'}_i = \mathbf{v}^{l}_i + \mathbf{a}_i\Delta t,$$

where $\mathbf{v}^{l}_i$ is the current velocity and Δt is the time step, i.e. the elapsed time between label placement updates.

From this, we can also calculate the new positions $\mathbf{l}'_i$ of all labels as

$$\mathbf{l}'_i = \mathbf{l}_i + \mathbf{v}^{l'}_i\Delta t.$$

With this kind of integration, the system will never reach an equilibrium, but it will only reach closer by each step. To make sure the system will reach a static state we introduce a threshold under which the force will not be applied at all. This is if the condition

$$\max(\|\mathbf{v}^{l'}_i\|, \frac{\|\mathbf{f}^i_{total}\|}{c_{\text{friction}}}) < c_{\text{static}}$$

holds true, where c_{static} is an adjustable parameter. Having this threshold could be important to ensure that unnecessary movement of labels is kept to a minimum.

3.2 Hiding Labels

There are situations where overlap between labels are unavoidable if all labels are to be placed on the screen. The extreme case of such a situation would, for example,

be when the total area of all labels are larger than the screen area. It could also be that the screen is covered with point features, giving no room for the labels.

In these situations, we must choose which labels should be shown and which are to be hidden. Therefore, we introduce a scoring system, where each label gets its own score. A label is penalized when it causes obstruction and other unwanted behaviors. It gains score over time.

The penalties given for obstruction is directly proportional to the distances calculated when finding the collision and feature forces. A penalty is given to a label when the calculated distance $d_{i,j}$ is negative, as this means that the labels overlap.

The score has an upper and a lower limit that can be adjusted. The lower limit can trivially be set to 0, but a value for the upper limit, $s_{\max}$, must be found. There are also two trigger levels, one when a label is to be hidden, s_{low} and one when it is to be shown, s_{high}. The former is, of course, lower than the latter. Finding suitable levels for the scoring system will be an important part of evaluating the method.

Another situation where a label might become hidden is if it has been moved too far away from its point feature. The limit is here referenced as $m_{\max}$. When this limit is reached for a label, it is hidden by setting its score equal to the hide trigger level, s_{low}. Its position is also reset to that of the point feature so that a possible new position for the label can be found closer to this point.

When a label is hidden it does not exert the same forces on other labels. Let us say that label i is showing, but label j is hidden. In this case the label collision force, $\mathbf{f}^{i,j}_{\text{collision}}$, and the weak collision force $\mathbf{f}^{i,j}_{\text{weak collision}}$ is not applied to label i. However, the opposite forces $\mathbf{f}^{j,i}_{\text{collision}}$ and $\mathbf{f}^{j,i}_{\text{weak collision}}$ are still applied to the hidden j. This is so that we can find a new location for the hidden label, where it can be shown again, if such a position exists.

The label movement prediction forces $\mathbf{f}^{i,j}_{\text{label predict}}$ and $\mathbf{f}^{j,i}_{\text{label predict}}$ is not applied at all if any of the labels i and j are hidden.

3.3 Label Priority

One of the requirements is to ensure the labels can have different priorities, where certain labels are prioritized to be placed before others. Therefore, each label has an integer number associated with it, where a higher number represents a higher priority. To enforce this, two approaches are used in conjunction.

We let a label with a high priority ignore forces applied by label with a lower one. Let us say label i has a higher priority than label j. In this case the collision force $\mathbf{f}^{i,j}_{\text{collision}}$ and the label prediction force $\mathbf{f}^{i,j}_{\text{label predict}}$ will not be applied to label i. The forces $\mathbf{f}^{j,i}_{\text{collision}}$ and $\mathbf{f}^{j,i}_{\text{label predict}}$ will still be applied to label j. This makes it so that the label j will make room, avoiding, label i, without the position of label i being affected.

We do still, however, keep the feature forces, such that the labels with high priority

do not overlap the features with lower priority. It might be desirable that this is not the case, but it could easily be adjusted.

The weak forces are also kept. The goal of these are only to make movement smoother, and they do not affect the overall placement. Therefore, it should not be a problem to keep them as long as their contribution to the total force, $\mathbf{f}_{\text{total}}$, is small.

The second approach is to let labels with higher priority not take score penalties when they are only overlapping features with lower priority. Lower priority labels still take a penalty from higher priority labels and those with the same priority. This makes it so only the lower priority label is hidden if labels with different priorities are overlapping. We choose to still apply penalties if a high priority label is overlapping a low priority point feature, but this could of course be changed, if desired.

3.4 Panning and Zooming

As panning and zooming of the map are important requirements put forward, some considerations have been made to handle this. It could however, be argued that this only is an implementation problem, that a placement method really only should work with the data that is associated with the labels and the features to be labeled. In the case of zooming and panning, it should be up to the application to transform the positions of the labels accordingly.

The only action required when panning the map, is to apply the same translation to all labels as is applied to the map, i.e. moving them in the same way. This is easy when the labels positions are kept in screen space.

When the map is zoomed, relative coordinates are used instead. It is made sure that all labels have the same relative positions to their point features after the transformation as they had before it. As the labels positions are kept in screen space, this requires a bit more of calculation. Zooming also requires the velocity vectors to be scaled accordingly.

3.5 Evaluation Tools

To evaluate the placement method three tools was developed. These are a demo application, a parameter test application, and an integration into a real-world application. Also, the placement method was used to create an overlay in a video clip.

3.5.1 Demo Application

A demo application was the first evaluation tool to be developed. Its purpose during development was to show the visual representation of what the label placement does, such that the impact of different ideas for the placement method could be seen. This application used Carmenta Engine, which is Carmenta's software development kit for creating GIS software. This allowed us to quickly build an application that renders a map on the screen and provided us with common interaction abilities such as panning and zooming in the map.

On top of this, we created simulated data sets that supplied point feature data in real-time to the application. This is the data that is to be labeled by our method. The first data set was one where a point starts at a random position in the current view, moving in a circle with a randomized speed. A second simulated data set was where the points move from the edge to the center of the view, pause in the center for a while, chose a random direction and move to the edge again. The simulated data sets also decided the sizes of the labels to be placed. The simulated data set could change the label's sizes on the fly to test a placement method on how well it would manage such situations.

In addition to the simulated data set some real data sets were also integrated. These were three publicly available, web-based, application programming interfaces (API), providing live data, and one large static data set. The first API was ADB-S Exchange[1], which provides position and other data for air traffic. Its data is updated as often as every second. The second API was the Digitraffic Marine Traffic API[2]. It is an API by the Finnish Transport Agency that supplies position and other data of sea vessels in and around Finnish waters. The position data is feed from the ships Automatic Identification System transponders in real-time. The third was Västtrafik Reseplaneraren Livemap[3], which provide positions of public transport vehicles in the south-western parts of Sweden. The static data set was the names of cities around the world with a population greater than 5000 people. This was provided by Geonames.org. The reason for integrating these was that they are believed to be possible future use cases for the method developed.

The actual implementation of the placement method was done as a plugin component in the processing chain of Carmenta Engine. The positions of the point features are translated from their geographical positions to screen positions and then sent to our algorithm to be labeled. When positions for the labels have been calculated, this information is given to Carmenta Engine. It is then the engine that is given the task to render the point features and the labels.

We will only do time measurements on the actual placement. The fact that the data is simulated in the test application, and that rendering is also left as external software, means that the placement method implementation is a stand-alone component. This makes the time measurements rather simple.

[1]ADB-S Exchange: `https://www.adsbexchange.com`
[2]Digitraffic: `http://digitraffic.liikennevirasto.fi`
[3]Västtrafik: `https://developer.vasttrafik.se`

3.5.2 Test Program

A simple test program was developed in order to test how different parameters values affect the placement in a consistent manner. It contains variations of the placement method implementation and the simulated data sets that is not required to be running in real-time. It rather emulates real-time such that calculations can be done faster. The simulated data sets was also modified such as they can simulate the exact same scenario multiple times and consistent results are generated.

The program outputs four different measures of how the placement method performs during a simulation. These are the total overlap of labels with other labels and point features, the total distance labels are placed from their point features, a total of how long each label is hidden and, finally, the absolute value of all forces applied to labels. These measurements should provide useful data in the process of finding good parameter values.

To analyze the results, an interface to interact with Matlab was added. This was to allow an easy way to test many different scenarios, with different parameter values, and comparing the results.

3.5.3 Real-World Application

As the main focus is to find a placement method for labeling in software used by emergency response the method was implemented in one such application. This was an administration interface for the system called ResQMap, used by the Swedish emergency response.

The application contains a typical map, which allows for normal zooming and panning. This map shows the location of all rescue vehicles such as ambulances and fire trucks. The positions are retrieved from a recording supplied by emergency response dispatcher SOS Alarm.

3.5.4 Sports

To test whether the label placement method is also suitable in the field of sport analysis, it was tested on a short video sequence of basketball. Placing labels on a sports game such as basketball is interesting as there is a lot of movement, with sudden direction changes, and players tend to converge towards the ball.

The data set used here is the APIDIS dataset[4]. It provides 7 cameras from a basketball game, and position of all players, for a one minute period. The data has an update rate of 25 times per second. To ensure that the labeling was performing well in terms of placement quality, two updates of the labels are done for each video frame. The velocities of the players feed to the labeling was based on the distances traveled between the two latest updates.

[4]APIDIS: `https://sites.uclouvain.be/ispgroup/Softwares/APIDIS`

4

Results

This section presents the results gathered using the evaluation tools. First the implementation of the method used is presented, followed by the possible parameter settings. Finally we show the performance of the method, both run time performance and how well it suits the requirements.

4.1 Implementation

We have made two different implementations of our method. They are both written in C++, are only based on CPU calculations, and run using a single thread. This should be compared to Vaaraniemi's implementation which was running on a GPU [11].

The first implementation is a naive one where each label is tested against all other labels and features. It required that forces were calculated for each pair. A second, smarter, implementation, places all point features and labels in a spatial data structure at the start of each update. Using this kind of data structure allows us to only calculate forces between pairs of labels and features that are sufficiently close to each other to have an effect.

The performance results for the second implementation includes the building of this data structure. It should be noted here that the point features were given random positions on the screen, with a uniform distribution. If all points would have the same position on the screen, the time results would be about as long as that of the first implementation. This is due to the fact that labels sufficiently close to said point must calculate the force contribution from each point feature.

The performance of the second implementation also depends on the parameters given. If the set parameters result in forces acting on larger distances, more forces will have to be calculated for each label.

4.2 Parameters

As the method chosen has 21 adjustable parameters, as summarized in appendix A, the knowledge gained from the demo application and the evaluation program have

been used to reduce this number. This has been accomplished by first identifying relationships between the parameters, such that they can be combined with each other, and trying to find good default values for others.

4.2.1 Margin Parameters

Some parameters were identified to be application specific or dependent on personal preferences. The most obvious here is the distance between the labels and the features. The pulling force distance m_{pull} and the feature collision margin m_{feature} should have the same value. This means that the corresponding forces should both be zero when a label is exactly this distance from its point feature, granted that it is not colliding with other features.

The similar margin parameter $m_{\text{collision}}$ is also left as an adjustable parameter. This controls how close labels should be placed next to each other, which can depend on the application.

Having forces acting on great distances may limit the effectiveness of certain implementations of the method. One such implementation would be the above-mentioned implementation using a spatial data structure. The forces that act on the greatest distances are the weak forces and the movement prediction forces. Therefore, these should be acting on an area as small as possible, while still accomplishing their purpose.

The weak forces should be effective if they act on a distance equal to the size of a label. This is motivated by looking at a situation where two labels move apart due to this force, leaving just enough room for a label to be placed in between them. This allows a hidden label to shown again in that space. The same goes for the weak point feature collision force, leaving room for a new label between a point feature and the label. This means that the weak collision reach parameters, $m_{\text{weak collision}}$ and $m_{\text{weak feature}}$, is set to that of the largest label side length.

The movement prediction forces have little reason to act on a larger distance than for the labels to avoid a total collision. This would represent a parameter value of 1. Some extra margin could however be given to avoid a situation where the stronger collision forces come into effect. Therefore appropriate values for these parameters, $m_{\text{label predict}}$ and $m_{\text{point predict}}$, should be 1.5.

4.2.2 Force Scaling Parameters

The force scaling parameter for collision between labels and features, $c_{\text{collision}}$ and c_{feature}, is set to have the same value. This was decided as the forces have the same purpose, avoid overlap, and their definitions are near to identical.

To find good reference values for these parameters, the evaluation program was used. We looked at how the total acceleration of all labels and the total overlap of labels in one simulation varied based on the parameter value. The result of one

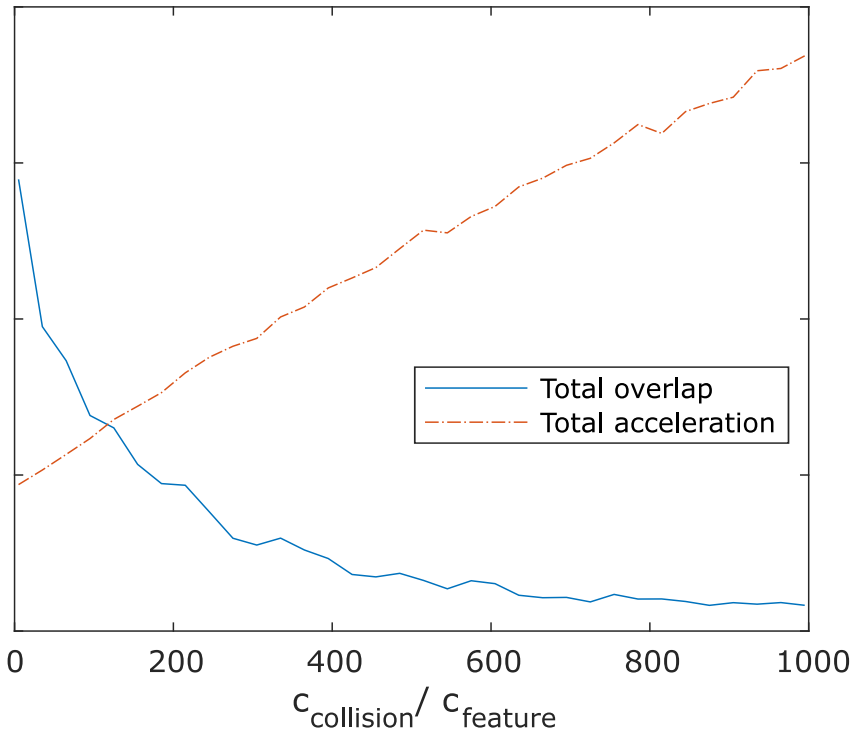

Figure 4.1: The figure shows how the total overlap between labels and features in a simulation depend on different values of the force scale parameters $c_{\text{collision}}$ and c_{feature}, presuming they are set to the same value. It also shows that the acceleration of labels are linearly dependent on these parameters.

such simulation can be seen in figure 4.1. Different label sizes, number of labels, and different label speeds were tested, but results did not variate much.

In the analysis of the results, it was found that the total acceleration has a near to linear relationship with the parameter setting. Based on this, the parameter value should be minimized to make sure the movement of the labels is as smooth as possible.

Looking instead on the total label and feature overlap, we find that it has an inverse relationship to the parameter setting. It appears that there is a point where the overlap does no longer become much smaller when increasing the force scale. For most situations this is where $c_{\text{collision}} > 700$, which constitutes an upper limit. A lower bound on the parameter is set as 300. From our tests, there are some situations where this lower value is enough, but it depends on the application and, therefore, this is left as an adjustable parameter.

For the weak collision forces the proposed value is at about 5% of the normal collision forces. The motivation for this can be found in figure 4.2 as a small plateau in the beginning of each curve.

For the movement prediction forces there is no decisive answer for at what level these should be set. Though it has been found that it could be dependent on the

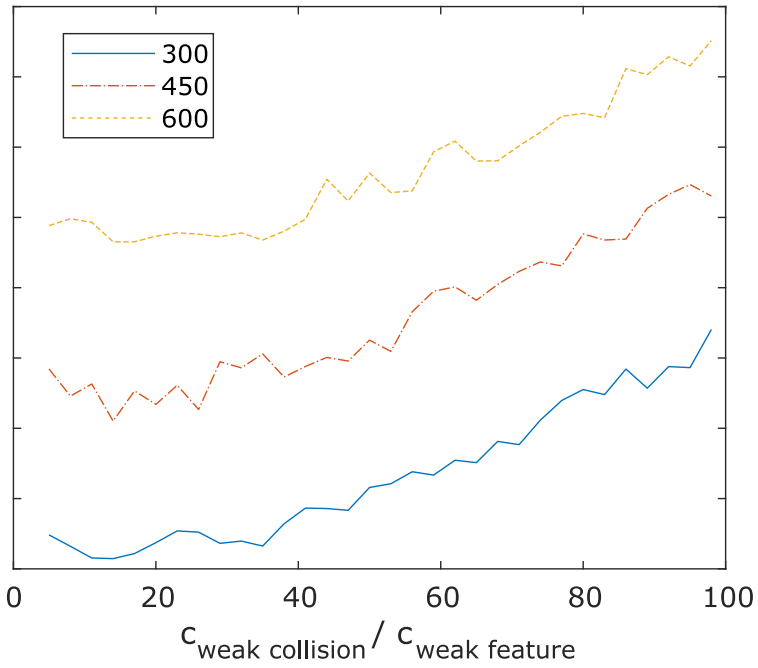

Figure 4.2: The figure shows the total acceleration of labels in a simulation depending of the force scale for the weak forces. The different curves show different settings for the main collision forces.

application. Applications with very low update rate of the position data, lower than a few seconds apart, does not benefit much from the prediction force. Where positions are given at a higher frequency or are extrapolated/interpolated at small intervals, reasonable values has been found to be approximately 6.

The pulling force is also left as an adjustable parameter. Suggested values are in the range of 20 to 40, but is down to personal preferences.

The friction force is set to 6. This ensures that the system is generally stable and labels do rarely overshoot their optimal positions, while still allowing them to move quite freely. This was set by visual inspection.

4.2.3 Score Parameters

The investigation of the scoring system parameters yielded no conclusive results. The suggested values for these are mostly based on subjective grounds. There is however, one important relationship between them to consider, the shortest possible time from that of a label being hidden, to when it can possibly be shown again. This time t_{state} is given as

$$t_{\text{state}} = \frac{s_{\text{high}} - s_{\text{low}}}{s_{\text{recover}}}.$$

Letting this value be too low might cause unnecessary flickering, where a label might

be in a suitable spot for this amount of time, only to be causing overlap again shortly after.

We suggest the following parameter values for the scoring system:

Description	Symbol	Value
Score Hide Trigger	s_{high}	1
Score Show Trigger	s_{low}	3
Max Score	s_{max}	6
Score Recover	s_{recover}	5

4.3 Performance

The first naive implementation is able to place 100 labels in 0.9 ms per screen update and 1000 labels take about 47 ms per screen update. The second implementation, using the spatial data structure, can place 100 labels in under 0.3 ms per update and 1000 labels in about 4.2 ms. All these time measurement results were done on a modern laptop, using an Intel Core i7-7820HQ, on the first simulated data set with uniform distribution of point features.

4.4 Complexity

We must calculate $\mathbf{f}^i_{\text{total}}$ for each label i. When calculating on such $\mathbf{f}^i_{\text{total}}$ forces are summarized from all labels and features. So if all forces $\mathbf{f}^{i,j}$ can be calculated in constant time the complexity will be $\mathcal{O}(n^2)$.

Calculating those forces requires at most the current positions of the labels and the features, together with the extent of the labels. If these can be found in constant time the calculation of the individual forces can be done in constant time.

This means the worst case computational complexity of our method is $\mathcal{O}(n^2)$, where n represents the number of point features. This is based on the case where forces must be calculated for each pair of label and points.

4.5 Requirement Fulfillment

The label placement method presented only rely on positions and velocities of the features to be labeled in one given moment. Therefore, it need not introduce any delay waiting for data to be available. The only delay introduced is that of the computation time required for the placement to update.

Overlap of labels and point features is allowed as there are no strict rules preventing this. The scoring system introduced does require that labels overlap somewhat

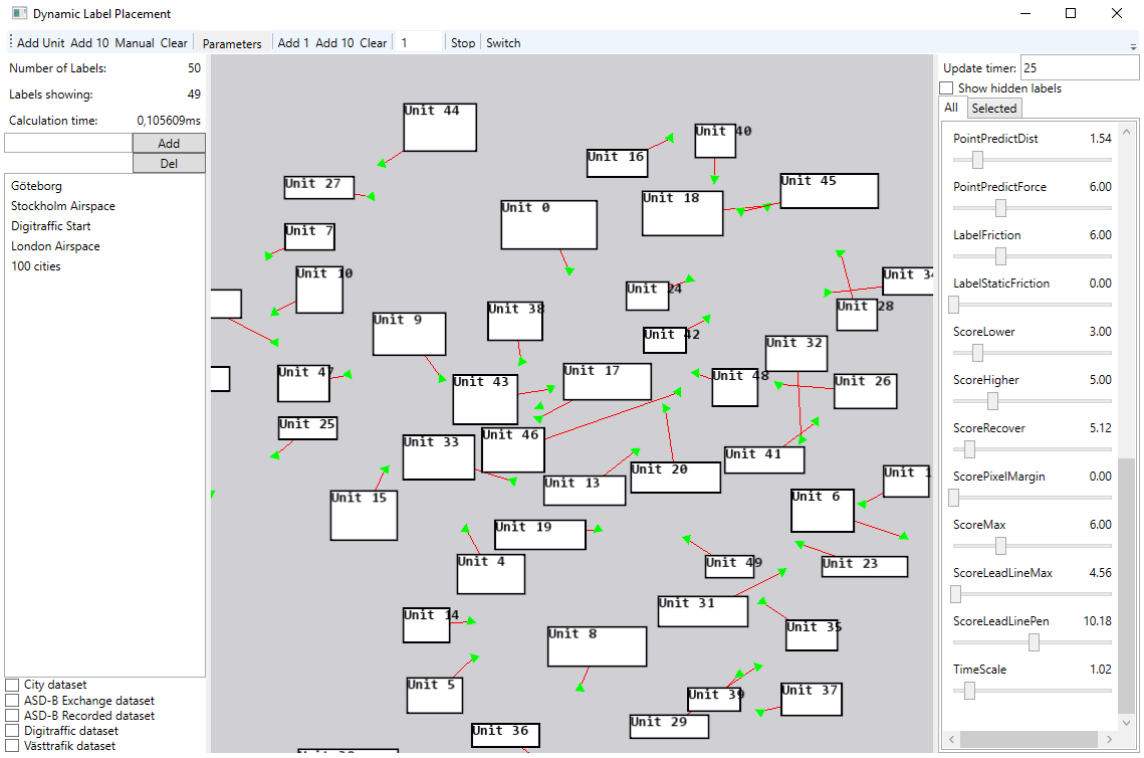

Figure 4.3: A screen shot of the demonstration application showing one of the simulated data sets.

before deciding to hide one of the labels. However, the forces acting on labels make sure that labels stay separated when possible.

The method does allow labels to be placed some distance away from their point features, which creates situations where the use of lead lines may be necessary. There is however, a method presented that allows the user to prevent the lead lines to become too long by hiding such labels and finding new positions for such labels.

It is possible to pan and zoom the map with the labels moving smoothly and staying close to their point features. Also the label priorities were taken into consideration.

Carmenta has given positive feedback on the placement method. The team working with the development of the products used by emergency services has expressed interest in integrating the method in the future.

4.6 Demo Application

To get an idea of how the demonstration application came to look like, a screenshot can be viewed in figure 4.3. Additional screen shots are found in appendix B. A video demonstration of the application can be found on `https://youtu.be/GiUzVvo0f1s`.

The method seems to perform well on all the data sets tested, with one exception.

The labels move smoothly regardless of the update frequency of the data. However, the quality is bound by the update frequency of the placement. A slow running application can of course not ensure the smoothness. It can also induce some oscillations in the labels. This means that working with very large data sets pose a problem.

The data set providing position data for public transport vehicles did not work well with the label placement. The reason is that the position data was discontinuous, i.e. vehicles would make some sudden jumps on the map. As the method was created to move the labels smoothly, they had a hard time staying sufficiently close to their point features to make the information shown easy to comprehend.

5

Conclusion

In this section we analyze the development of the placement method and the results found. We also look at where the method can be used, and how it can be improved or expanded upon.

5.1 Discussion

5.1.1 Development

The development of the method was an iterative process. The choice fell on making it a force-based method, as this had earlier proved to not need to introduce any delay and to be able to perform in real-time. From the method presented by Vaaraniemi et al., the point feature collision and weak forces were added quite early in the process. Later, the prediction force and the static friction were added out of necessity.

Early on in the development, we were only using the simulated data set. This proved to be enough to decide on how the forces should be calculated, what relationship to use between distances and the forces. However, the real data sets played an important role in the exploration of the parameters. The exploration was what showed that different kind of data would require different parameter setups. Also, it was the integration in the real-world application that resulted in the inclusion of static friction.

5.1.2 Force-Distance Relationship

All forces applied to the labels use a linear relationship between distances and the magnitude of the forces, with the pulling force being the exception. This choice is not obvious. In nature forces such as gravity and electromagnetism work according to the inverse-square law and it could have been an alternative here as well. Other possibilities could have been higher exponents or logarithmic relationships.

All of these alternatives have been tested. What was found was the fact that the Euler integration method did not cope well with the alternative functions. It often created unstable situations, something that might have been avoidable with

shorter time steps or other numerical integration methods, but would require more computation time.

5.1.3 Problems

There is one major downside of our forced based method. It is possible that local minimums exist, where labels may not be placed in an optimal manner. One example of this is when a label or point feature is placed exactly between another label and its point feature. In such a situation, it would often be preferred that the labels would switch places.

This problem was known since previous work. It was mentioned already when the force-based method for label placement was introduced in [12], and it was the problem addressed in [13].

Another problem with the force-based methods is that they are dependent on the time step length. While we were able to keep an update rate of 60 updates per second the placement method performed well. In a situation where such update rate is not possible the quality of the placement could be suffering.

5.1.4 Parameters

Analyzing and finding parameter values proved to be a difficult part of the work conducted. When first looking at them, it was obvious that some of them had no reason to be different, but could instead be combined, such as the scaling of the collision forces. It was the next step that proved more difficult.

Many of the parameter values proved to be dependent on personal preferences or to be application specific. Applications working with more static data, slow-moving objects and low update rates seemed to require different settings from applications with more movement and higher update rates. We could at least present some reference values that can be used as a start when implementing the method.

The parameters that proved to be the most troublesome were those for the scoring system. They seem quite arbitrary and the values presented are based on nothing else than that they seemed to work fine in our demo application. There is however, one problem with them that was not addressed. The parameters are dependent on the size of the labels. When a label is given a score penalty due to overlap, the penalty is proportional to the magnitude of the overlap. On a small label, a 10 pixels overlap might be a complete overlap, while it on a much larger label could be just a fraction of the label that is overlapping. It may be possible to solve by making the scoring system dependent on the label size as well. The question is if this is suitable for all applications.

5.1.5 Performance

We managed to reach our performance target as 1000 points in around 4 ms is well below our target of around 16 ms. There should be few problems integrating the method in interactive systems requiring update rates of 60 times per second.

The performance numbers were gathered on an, at the time, quite powerful laptop processor. With less computational power it might be difficult to keep the interactive speed. However, the only real optimization we did was to use the spatial data structure, so there is a reason to believe that more can be done to improve the performance.

The most obvious is to parallelize the computation. The computation for each label only depends on the input data and is independent of each others calculations. This means that each label can be computed in parallel. With modern multi-core processors, much more performance could be gained. With the method being parallelizable, it would also be possible to make use of GPU to do the computation as we saw in [11].

5.1.6 Possible Use Cases

The primary use cases are, as the purpose stated, labeling of real-time geographical data. Carmenta is already considering to implement the method in their software for emergency dispatchers. Other command and control systems are good candidates as use cases, such as traffic control demonstrated by the demo application.

There are however, certain aspects of the method could make it less suited in some cases. For example, hiding labels might not always be an alternative. Also, the quality of the method depends on the computation power available. Lastly, the method is not suited for applications where movements in the data are very sudden.

The method could of course also be used in non-real-time applications as well. It is suitable for all interactive and non-interactive applications with dynamic data. However, it could be that for non-real-time data, other methods exist that make use of knowledge of the position data over a time period, allowing them to provide better quality placement.

Sport analysis is also a possible use case. Our method preform well on the data set used, and it is obvious that the movement prediction force place an important role in avoiding sudden label movements in this case. However, our method is primarily designed for higher number of labels than what is typically required for such applications. Another problem is that it is not really a case of point labeling, but rather placing labels next to humans on a screen, and our method has no ability to avoid obscuring them. As mentioned, it already exists methods for dynamic label placement in the numbers of 20-40 [11]. It is possible that those methods are even better suited for sport tracking.

5.2 Future Work

As already stated in the limitations, a human interaction study is not a part of this thesis. To be completely certain that a developed method is good, one should do such a study. We saw that such studies had been done on existing methods, even on the method by Vaaraniemi et al. [11] that our work is based on. It is however, best left for future work due to time constraints and lack of expertise in the field.

While the developed method can avoid the point features that are to be labeled, no other part of the map is considered in the placement. Such placement considerations could be of interest in some applications, and future work could try to expand the method to do so. That other point features can be avoided as well is easy to see, but it is not obvious how one would avoid other types of map features, such as line and area features. Some interesting candidate methods for achieving this can be found in works by Luboschik et al. [14] and Stadler et al. [15].

In a report by Duverger and Hering [16] it is mentioned that labels can have preferred placements relative to the point feature. In that particular case, it is preferred to place labels to the left or to the right of the direction of movement. They present a weighted formula that may be compatible or adaptable to our method to achieve the same goal.

The same report also mentioned the problem of crossed and obscured leading lines, and that it would be of interest to find a solution to that problem [16]. This was not addressed in here but would be an interesting continuation of development.

Our method performs well enough to place the number of labels given as initial requirements, in real-time. It could however, exist situations where you might want to consider even more labels to be placed. We believe this could be achieved by placing time restrictions on the computations, and only consider so many time allows, granted there is a way to prioritize the labels.

Finally, it could be interesting to extend our method to work with labels which are not axis-aligned and rectangular. As long as it is possible to calculate a distance between two labels it should not be a problem to calculate most of the forces. However, the movement prediction force might need some more work to make it work with more general label shapes.

5.3 Conclusion

Our early results show that it is in fact very possible to do label placement on moving objects in real-time. We can place a reasonable amount of labels, such that a typical computer screen can be filled, with a reasonable update rate. While earlier work suggested it was possible using a GPU to achieve such results, we have here shown that it is also possible on a CPU.

Compared to earlier work we have also expanded on what such a placement method

can do. What is new here is that we allow the point features to be associated with real-time data, and this without a need to introduce any significant delay in the label placement. Furthermore, we also allow a condition that labels must not overlap the point features that are to be labeled.

Bibliography

Bibliographic notes: The most relevant reference is the force-based method in [11]. [1] contains an extensive bibliography on label placement. [2] is a proof that some label placement problems are NP-complete. [6] is the previous master thesis done at Carmenta, which in turn use much of the static method in [3]. Some interesting ideas that can be used to extend our method is found in [14] and [15].

[1] A. Wolff and T. Strijk, *The Map-Labeling Bibliography*, http://i11www.ira.uka.de/map-labeling/bibliography, 1996. [Online]. Available: http://i11www.ira.uka.de/map-labeling/bibliography/.

[2] S. Shieber and J. Marks, "The Computational Complexity of Cartographic Label Placement", Harvard Computer Science Group, Tech. Rep. TR-05-91, 1991. [Online]. Available: http://nrs.harvard.edu/urn-3:HUL.InstRepos:24019781.

[3] K. Mote, "Fast Point-Feature Label Placement for Dynamic Visualizations", *Information Visualization*, vol. 6, no. 4, pp. 249–260, 2007. DOI: doi:10.1057/palgrave.ivs.9500163. [Online]. Available: http://journals.sagepub.com/doi/10.1057/palgrave.ivs.9500163.

[4] F. Wagner, A. Wolff, V. Kapoor, and T. Strijk, "Three rules suffice for good label placement", *Algorithmica*, vol. 30, no. 2, pp. 334–349, 2001.

[5] I. Petzold, G. Gröger, and L. Plümer, "Fast screen map labeling—data-structures and algorithms", in *Proc. 23rd Internat. Cartographic Conf. (ICC'03)*, Durban, South Africa, 2003, pp. 288–298.

[6] K. Hallqvist, "Dynamic label placement for moving objects", Independent thesis Advanced level, KTH Royal Institute of Technology, 2017. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-201632.

[7] M. De Berg and D. H. P. Gerrits, "Labeling moving points with a trade-off between label speed and label overlap", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, ISBN: 9783642404498. DOI: 10.1007/978-3-642-40450-4_32.

[8] H. Cords, M. Luboschik, and H. Schumann, "Floating labels: Improving dynamics of interactive labeling approaches", *Proceedings of the IADIS International Conference Computer Graphics, Visualization, Computer Vision and Image Processing 2009, CGVCVIP 2009. Part of the IADIS MCCSIS 2009*, 2009. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.162.2485&rep=rep1&type=pdf.

[9] T. Stein and X. Décoret, "Dynamic Label Placement for Improved Interactive Exploration", *Proceedings of the Sixth International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2008, June 9–11, 2008, Annecy, France)*, vol. 1, no. 212, pp. 15–21, 2008. DOI: 10.1145/1377980.1377986. [Online]. Available: http://portal.acm.org/citation.cfm?id=1377986&dl=GUIDE&coll=GUIDE&CFID=65084944&CFTOKEN=30232914.

[10] S. Pick, B. Hentschel, I. Tedjo-Palczynski, M. Wolter, and T. Kuhlen, "Automated Positioning of Annotations in Immersive Virtual Environments", in *Joint Virtual Reality Conference of EGVE - EuroVR - VEC*, T. Kuhlen, S. Coquillart, and V. Interrante, Eds., The Eurographics Association, 2010, ISBN: 978-3-905674-30-9. DOI: 10.2312/EGVE/JVRC10/001-008.

[11] M. Vaaraniemi, M. Treib, and R. Westermann, "Temporally Coherent Real-Time Labeling of Dynamic Scenes", in *COM.Geo Proceedings on Computing for Geospatial Research and Application*, 2012.

[12] S. A. Hirsch, "An algorithm for automatic name placement around point data", *The American Cartographer*, vol. 9, no. 1, pp. 5–17, 1982. DOI: 10.1559/152304082783948367. [Online]. Available: https://doi.org/10.1559/152304082783948367.

[13] D. Ebner, G. W. Klau, and R. Weiskircher, "Force-Based Label Number Maximization", Institute of Computer Graphics and Algorithms, Vienna University of Technology, Tech. Rep., 2003.

[14] M. Luboschik, H. Schumann, and H. Cords, "Particle-based labeling: Fast point-feature labeling without obscuring other visual features", *IEEE Trans. Visualization & Comput. Graphics*, vol. 14, no. 6, pp. 1237–1244, 2008. DOI: 10.1109/TVCG.2008.152. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2008.152.

[15] G. Stadler, T. Steiner, and J. Beiglböck, "A Practical Map Labeling Algorithm Utilizing Morphological Image Processing and Force-directed Methods", *Cartography and Geographic Information Science*, vol. 33, no. 3, pp. 207–215, Jan. 2006, ISSN: 1523-0406. DOI: 10.1559/152304006779077327. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1559/152304006779077327.

[16] A. Duverger and H. Hering, "Development of a Mathematical weighted Formula to eliminate the overlapping of Aircraft Labels on the ATC Radar Display", Eurocontrol Experimental Centre, Bretigny sur Orge, Tech. Rep., 2005, pp. viii+36.

A

Adjustable Parameters

This table presents all parameters for the label placement. If a parameter is marked as adjustable it means that its suggested value is dependent on personal preferences or the application.

Description	Symbol	Adj[1]	Value	Unit
Label-label collision force scale	$c_{\text{collision}}$	Yes	300-700	-
Weak label-label collision force scale	$c_{\text{weak collision}}$	No	$.05c_{\text{collision}}$	-
Label-point feature collision force scale	c_{feature}	No	$c_{\text{collision}}$	-
Weak label-point feature collision force scale	$c_{\text{weak feature}}$	No	$.05c_{\text{feature}}$	-
Label movement prediction force scale	$c_{\text{label predict}}$	No	6	-
Point feature movement prediction force scale	$c_{\text{point predict}}$	No	6	-
Pulling force scale	c_{pull}	Yes	20-40	-
Friction force scale	c_{friction}	No	6	-
Static friction threshold	c_{static}	Yes	0-6	pixels /sec
Label-label collision margin	$m_{\text{collision}}$	Yes	-	pixels
Weak label-label collision reach	$m_{\text{weak collision}}$	No	$\max_i(\max(\mathbf{e}_i))$	pixels
Label-point feature margin	m_{feature}	Yes	-	pixels
Weak label-point feature reach	$m_{\text{weak feature}}$	No	$\max_i(\max(\mathbf{e}_i))$	pixels
Label movement prediction reach	$m_{\text{label predict}}$	No	1.5	-
Point movement prediction reach	$m_{\text{point predict}}$	No	1.5	-
Pulling force margin	m_{pull}	No	m_{feature}	pixels
Label-point maximum distance	$m_{\max}$	Yes	-	pixels
Score hide trigger	s_{low}	No	1	score

Description	Symbol	Adj[1]	Value	Unit
Score show trigger	s_{high}	No	3	score
Max score	s_{max}	No	6	score
Score recover	s_{recover}	No	5	score /sec

[1] Adjustable

B

Data Set Illustrations

Below are illustrations of the data sets labeled with our method.

All maps are attributed to Esri, Here, Garmin, © OpenStreetMap contributors, and the GIS user community. More information on openstreetmap.org/copyright.

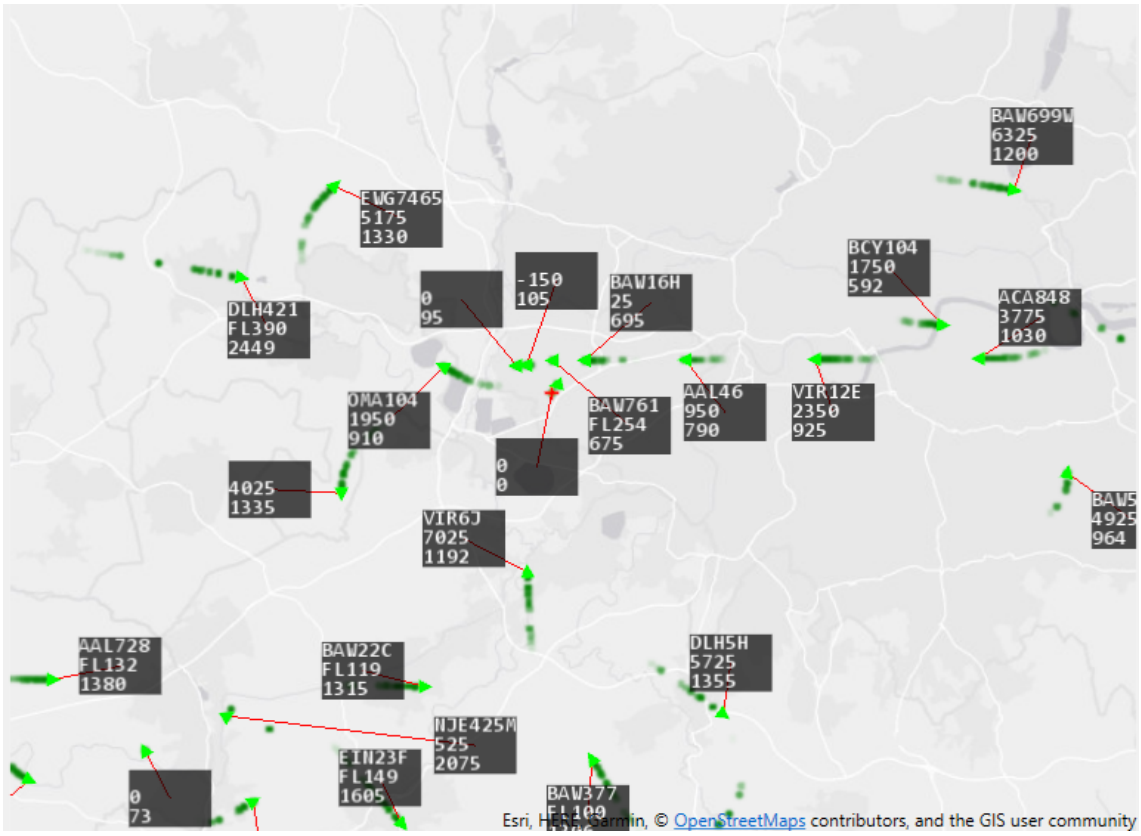

Figure B.1: Live air traffic above London from ADS-B Exchange, adsbexchange.com.

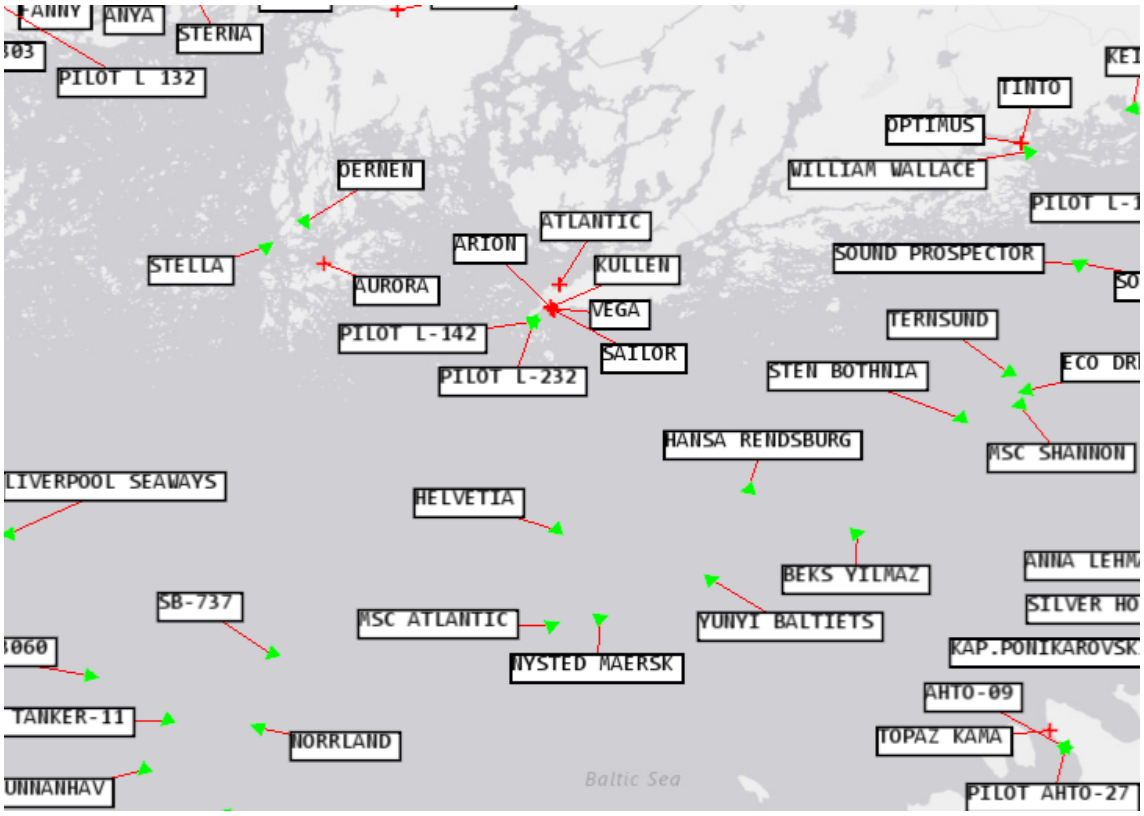

Figure B.2: Live sea vessel traffic from the Digitraffic data set. Data source: Finnish Transport Agency / digitraffic.liikennevirasto.fi, CC 4.0 BY

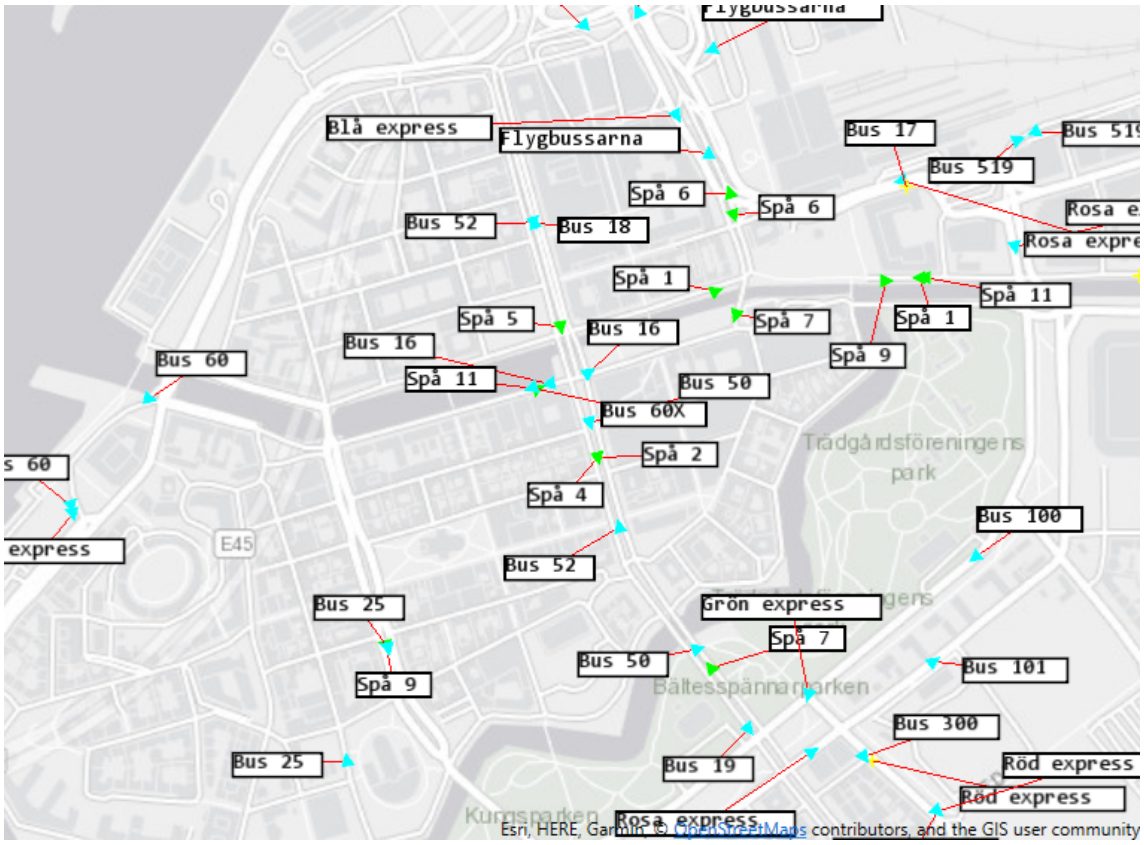

Figure B.3: Public transport traffic from Västtrafik.

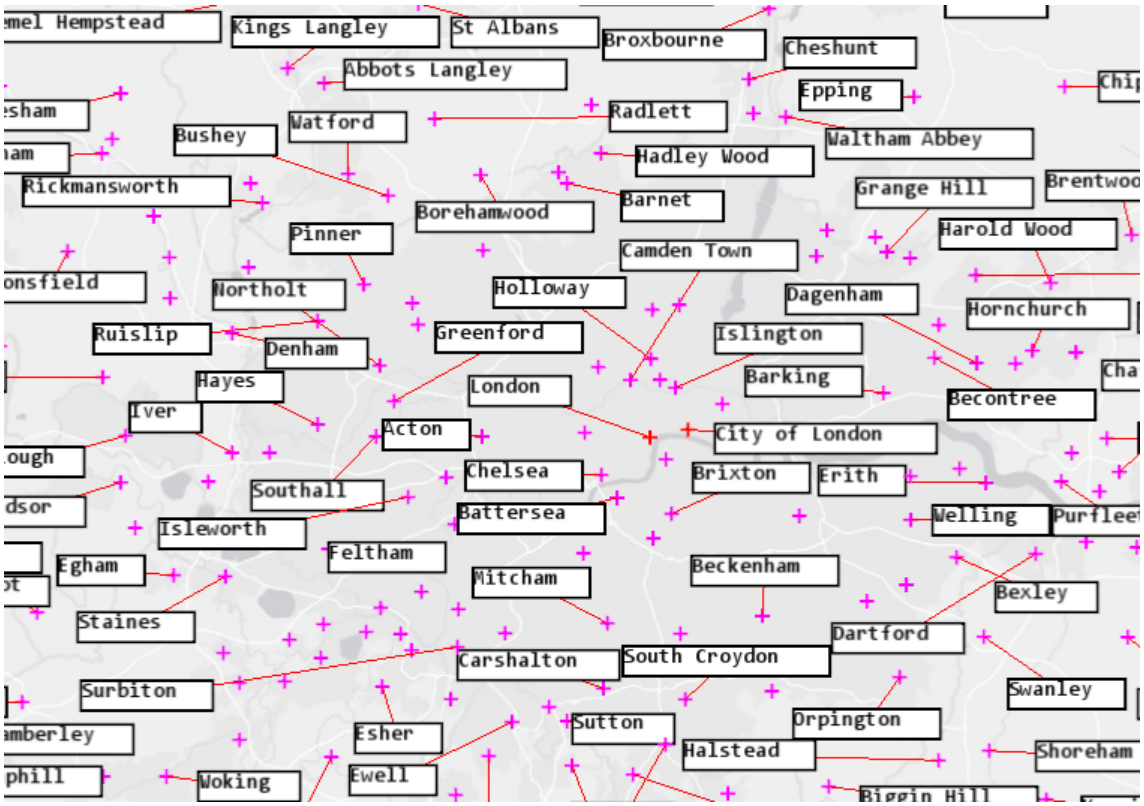

Figure B.4: A data set showing cities with a population greater than 5000. This shows the London area. Data source: GeoNames / geonames.org, CC 4.0 BY

Figure B.5: Labeling of a basketball game found in the APIDIS data set.