FLOATING LABELS:

IMPROVING DYNAMICS OF INTERACTIVE LABELING APPROACHES

Hilko Cords Martin Luboschik Heidrun Schumann

Computer Graphics Group
University of Rostock
Germany

ABSTRACT

The fastest existing labeling-algorithms allow the labeling of thousands of objects within a few milliseconds on today's desktop computers. Thus, it is possible to recalculate the labeling in dynamic scenes for every frame as it is demanded in interactive scenarios like information visualization. The main problem in such dynamic labeling environments is the lack of frame-to-frame coherence. Topology of label positions can change dramatically between consecutive frames — resulting in flickering and popping artifacts. Hence, visual label tracking becomes difficult and usability suffers. This short paper presents a universal approach for solving these problems by the use of animations — without manipulating the underlying labeling algorithm.

KEYWORDS

Interactive Labeling, Animated Labels.

1. INTRODUCTION

During the last years, the demand for a very fast labeling in interactive environments encouraged the development of several fast labeling techniques with different scopes. Due to decreasing labeling time, new problems arise in the presentation of the labeling solution. Whereas older approaches generate solutions within some seconds, recent techniques enable the labeling within milliseconds. Hence, labels appear and disappear within milliseconds (jittering), they appear unexpectedly during interactions like e.g. zooming (popping) or they rapidly change the screen position (jumping).

Most labeling techniques for interactive scenarios have been developed in the field of dynamic map labeling. Since the underlying dataset is *static* in such scenarios, an intense preprocessing can be applied to guarantee a *realtime interaction phase* [Petzold et al. 2003, Yamamoto et al. 2005, Been et al. 2006]. A common strategy in these techniques is to precompute all possible labeling conflicts into a conflict graph and to determine at which scale each single label should be added or removed from the solution. Since all labels have fixed precomputed positions only the visual appearing and disappearing of labels has to be handled – generally by changing opacity.

Unfortunately, the basis of labeling in information visualization is *dynamically* changing due to extensive interaction during data exploration (e.g., filtering, zooming, reordering, changing datasets…). Hence, these approaches with intense preprocessing cannot be used. More recent approaches aim at fast labeling results without the need for a time-consuming preprocessing. In [Roy et al. 2005] for example, a graph-theoretic

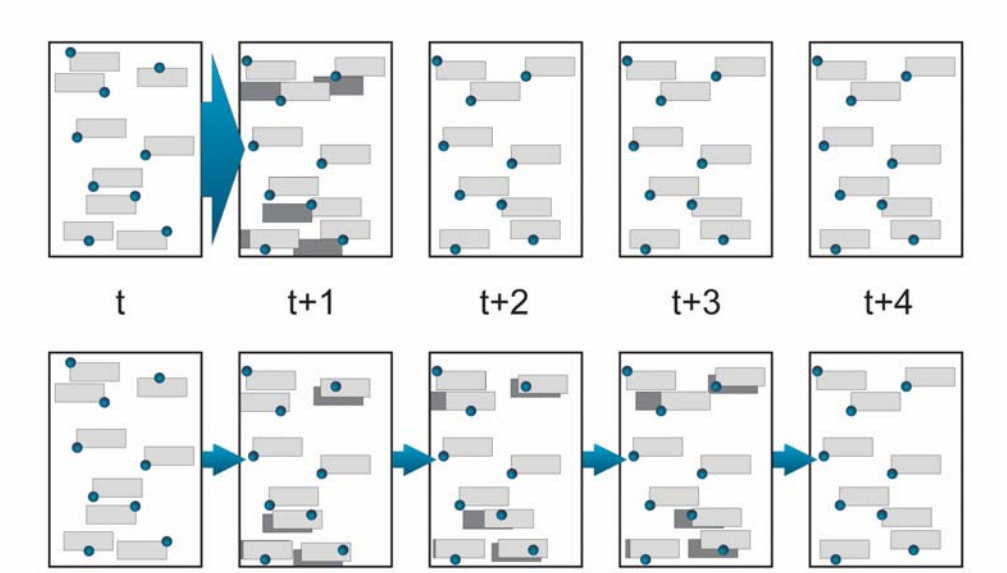

Figure 1: The proposed *floating labels* method. The upper row demonstrates rapid changes, while losing frame-to-frame coherence (dark grey: previous position, light grey: current position). Within the lower row floating labels are used, to spread instantaneous changes smoothly across several frames. Thus, the visual tracking of label positions is improved. Please take note of the accompanying video, demonstrating the technique more clearly.

algorithm possesses a runtime complexity of $O(n\sqrt{n})$ – promising realtime suitability with high quality labeling. The approaches presented in [Mote 2007, Luboschik et al. 2008] also enable a very fast global labeling of thousands of objects. Besides those global labeling methods, there exist local techniques like e.g., excentric labeling [Fekete and Plaisant 1999, Fuchs et al. 2006]. To reduce global complexity, a lens selects items that are labeled locally: Labels are arranged in lists around the lens and are connected to the corresponding items with lines.

All these methods are fast enough to calculate a valid labeling for nearly every frame. Since they are not designed to include information of previous solutions into current calculations, all of these techniques are characterized by a missing frame-to-frame coherence. Only [Mote 2007] addresses this problem by using a just-in-time precalculation of different zoom levels to prevent hard popping artifacts.

In principle, classical force-based labeling approaches include a smooth transition that avoids jumping. Therefore, an additional attracting force is introduced, that is attached to the label and its previous position. But approaches like [Ali et al. 2005] are generally not applicable to loosely scattered point-features as they are endangered of a local minima problem [Hirsch 1982]. Moreover, most of them are to slow to handle thousands of objects and finally, the problem of popping or jittering cannot be addressed this way.

Hence, the presented work uses the advantage of very short labeling times in recent labeling approaches to prevent jumping, jittering and popping by smooth transitions. Thus, the visual tracking of single labels is made possible or at least easier and the overall presentation is calmed down and is more pleasing to the eye. By providing fundamentals, this paper demonstrates the potential of floating labels.

2. FLOATING LABELS

In the following we will describe our concept for handling dynamically changing labels. We assume an environment, in which all label positions are recalculated in every frame. Take note that this is not a constraint, since our approach works with recalculations every *n*-th frame as well. The resulting disadvantage within the latter case is the lag of visual transition feedback behind the actual interaction.

To keep our concept general, we focus on adjacent labeling techniques as well as distant labeling (e.g., [Fekete and Plaisant 1999, Fuchs et al. 2006, Luboschik et al. 2008]). Distant labels are an important approach to facilitate the labeling of dense features. In practice, they are the major reason for jittering and jumping effects due to dramatically changing label positions – reducing convenient usability within interactive environments immensely.

The main goal of our current work is the distribution of instantaneous changes of labeling states to smooth transitions over time – converging against the final labeling solution. Hence, we facilitate or at least support the visual tracking of single items. An example of the floating labels approach is given in Figure 1.

In dynamic label environments, there are four different possible events, which can occur to a label:

- A label is added,
- A label holds its position,
- A label changes its position,
- A label is removed.

Except the second trivial one, each event is followed by a rearrangement in the labeling solution. With each change we start an animation process, which is either finished by achieving the recent solution or interrupted by a new event. To aim on smooth transitions even the interruption case has to be handled stable and consistent.

A straight forward and well-known representation for the adding and removal process of visual items in interactive environments is the use of blending (fading in/out): If a label is added, the label's colors and intensities are faded in – vice versa if it is removed. For simplicity reasons we use a linear fading function. Each label i ($i = 1 \ldots n$, n: number of *all* labels) gets a blending value $b_i \in \mathbf{R}$ with $0 \leq b_i \leq 1$, describing the intensity of the label. There are three different cases to be considered, resulting in different variations of b_i:

- Label i is added: $b_i = 0$,
- Label i is already shown: $b_i = b_i + b^{step}$,
- Label i is not shown/removed: $b_i = b_i - b^{step}$,

where the constant $b^{step}>0$ defines the blending speed. Afterwards, b_i is clamped to $0 \leq b_i \leq 1$. Thus, added/removed labels are blended smoothly in/out. The method results in smooth transitions, even, if a label is jittering, since no hard intensity changes occur.

For instantaneous label position changes, we apply an animated translation that incorporates the following requirements:

- Converge to the determined, current position quickly,
- Smooth transitions,
- Fast calculation of the animation path.

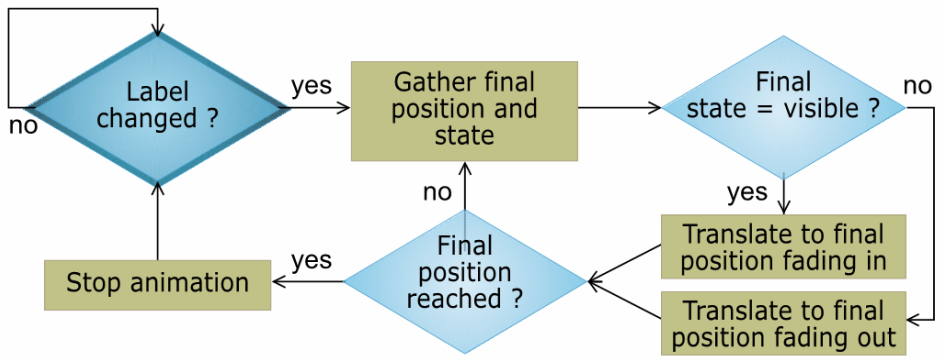

Figure 2: Flowchart of the described event handling. Starting point is the marked state in the upper left.

The first requirement is important for a fast recognition of label and feature affiliation as it minimizes the distance between the translating label and the corresponding feature. The second one is necessary for a smooth eye-pleasing animation – even in strongly jittered situations. We experimented with different translation functions and found the linear approach to be the most suitable according to tracking, smoothness and calculation efforts. However, functions of higher order or e.g., bezier-approaches, result in smoother translations but we did not spot a convincing benefit of smoother animation in contrast to slower convergence and decreased performance. Since the translation function is evaluated for each label and every frame, the calculation time becomes significant – especially within environments with many labels.

Our translation approach is as follows: Each label i ($i = 1…n$, n: number of *all* labels) with its current position $\mathbf{l}_i \in \mathbf{R}^2$ gets an attached label called *animated label* (with position $\mathbf{a}_i \in \mathbf{R}^2$). During the animation process the animated label replaces the current label position $\mathbf{l}_i$. Thus, all labels are rendered at positions $\mathbf{a}_i$, although the determined current position is $\mathbf{l}_i$. To smooth out the mentioned flickering effects and to animate the label smoothly, we use the following formula:

$$\mathbf{a}_i^{new} = \frac{(c \cdot \mathbf{a}_i^{old}) + \mathbf{l}_i}{c+1},$$

where the constant $c>0$ defines the animation speed. This translation function features a distant-dependent acceleration from $\mathbf{a}_i$ to $\mathbf{l}_i$. Hence, the closer the animated label to the real label is, the slower the animation runs, resulting in a spring-like behavior. Additionally, if the current label position $\mathbf{l}_i$ changes, the animation will follow immediately – without any explicit handling within the animation process. Thus, the animation has not to be finished before a label position changes.

Within dynamic labeling environments, the three events of adding, removing and position change occur permanently for different labels. Hence, an important feature is the superposition of incomplete transitions and blending effects. We implement this feature by incrementing or decrementing the blending factor of an item according to its final state (visible/non-visible) – within each label translation step. The translation function handles such situations directly, as it is based on the current labeling position $\mathbf{a}_i$. The flowchart of the described principle is given in Figure 2.

3. CONCLUSION

Our current work successfully distributes rapid changes in labeling solutions over time. Thus, it supports the visual tracking of labels and facilitates eye-pleasing interactive interfaces as found e.g., in force-based labeling approaches. By using complete labeling solutions as input, it is additionally labeling technique independent. Please take note of the accompanying video, demonstrating the technique and its benefits in practice (http://www.dosensport.de/floating/floating.html). Ongoing work investigates possibilities of translating labels not occluding each other, e.g. by using repelling forces during animation.

REFERENCES

Kamran Ali, Knut Hartmann, and Thomas Strothotte, 2005, Label layout for interactive 3d illustrations. *Journal of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'05)*. Vol. 13, No. 1, pp 1-8.

Ken Been, Eli Daiches, and Chee Yap, 2006, Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*. Vol. 12, No. 5, pp 773-780.

Georg Fuchs, Martin Luboschik, Knut Hartmann, Kamran Ali, Thomas Strothotte, and Heidrun Schumann, 2006, Adaptive labeling for interactive mobile information systems. *Proceedings of the 10th International Conference Information Visualization (IV'06),* pp 453-459.

Jean-Daniel Fekete and Catherine Plaisant, 1999, Excentric labeling: Dynamic neighborhood labeling for data visualization. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99),* pp 512-519.

Stephen A. Hirsch, 1982, An algorithm for automatic name placement around point data. *The American Cartographer*, Vol. 9, No. 1, pp 5-17.

Martin Luboschik, Heidrun Schumann, and Hilko Cords, 2008, Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics (InfoVis'08)*, Vol. 14, No. 6, pp 1237-1244.

Kevin Mote, 2007, Fast point-feature label placement for dynamic visualizations. *Information Visualization*. Vol. 6, No. 4, pp 249–260.

Ingo Petzold, Gerhard Gröger, and Lutz Plümer, 2003, Fast screen map labeling – data-structures and algorithms. *Proceedings of the 23rd International Cartographic Conference (ICC'03).* Vol. 13, No. 1, pp 1-8.

Sasanka Roy, Subhasis Bhattacharjee, Sandip Das, and Subhas C. Nandy, 2005, A fast algorithm for point labeling problem. *Proceedings of the 17th Canadian Conference on Computational Geometry (CCCG'05)*, pp 155-158.

Missae Yamamoto, Gilberto Camara, and Luiz Antonio Nogueira Lorena, 2005, Fast point-feature label placement algorithm for real time screen maps. *Proceedings of the Brazilian Symposium on GeoInformatics (GEOINFO'05)*, pp 1-13.