Boundary Labeling for annotated documents

Jakob Klinger
Matr.: 1125755
Curriculum: 033 534
e1125755@student.tuwien.ac.at

August 23, 2017

1 Introduction

Annotating a document is usually solved by adding footnotes or figures in an appropriate position and adding a simple reference in the text, leaving the reader to find the referenced content by themselves. Sometimes however, if a more obvious connection between the text and the referenced content is required, the reference is visibly connected to the text by drawing a straight line or a more complex path between them.

In this paper, we will look at ways to use Boundary Labeling, which means that all annotations will be placed outside of the text they are referencing and will be visually connected to the feature they are referencing. (See also [1])

The guidelines on how to create an optimal labeling are as follows: the connections should be as direct as possible, no important information should be obscured, and it should be easily discernable which Label belongs to which site. These three easily come into conflict with one another, especially when labeling documents, as the text usually is very dense and leaves little space for lines in between, yet one shouldn't allow them to pass through the text, as this makes the text harder to read.

While there are many papers discussing Boundary Labeling in general, only very few exist that apply this concept to written text. Generally, this approach isn't used very often, and tends to use simplistic algorithms which produce mediocre results. However, the papers that do discuss boundary labeling in text offer interesting contributions.

For example, the paper about the Luatodonotes-Package[2] illustrates some of the different styles of drawing these connecting paths, and came to the conclusion that paths without bends are easier to follow. However, most solutions proposed in that paper don't care whether a path overlapped with text or not, which results in a decrease in readability.

The paper by Loose[3] on the other hand is based around only using the free space between lines and words, which produces longer paths, and forces curves, but doesn't obscure any part of the text.

2 Terminology and Fundamentals

While Boundary Labeling (or an equivalent concept) can be applied to a space with different geometry or more dimensions, this paper will only concern itself with two-dimensonal, euclidean space. To easily reference important concepts, some additional terminology will be introduced as well. (See Fig. 1 for a visual explanation)

- **Graph:** A Set of Vertices and Edges (see below), often used to represent routing problems. Can be further classified depending on its properties.
- **Edge:** A path connecting two vertices, usually represented as a line. Can be directional, limiting its usage, or have a cost ("Weight") associated with its use.

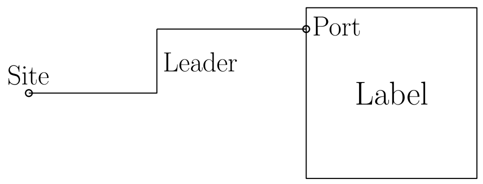

Figure 1: Illustrated guide to the labeling terminology

- **Vertex:** A featureless object that can be connected to any number other vertices via edges. Usually depicted as a point or small shape. Commonly also referred to as **"Node"**, as will be the the case in this paper.
- **Label:** Additional information, usually represented as a box containing the additional information. Will also be referred to as **"Annotation"** in this paper.
- **Site:** The point, object or word that is labeled - without it, there wouldn't be any labeling necessary.
- **Leader:** The path connecting the site to the label - depending on its shape, it can be described further using several subgroups which can be freely combined to form new leader types:
  - *S-Leader:* This Leader connects site and label in a straight line.
  - *O-Leader:* This leader runs orthogonally to the border between text and label area.
  - *P-Leader:* This Leader runs parallel to the border between text and label area. It must be combined with other leader styles, as it won't reach the label area otherwise.

  For example, the leader from Fig. 1 would be classified as an opo-Leader.
- **Port:** The location where the leader connects to the label. It may be restricted pre-determined locations, like only at the corners.

3 The program

In our program, the focus was put on keeping the text as readable as possible. This means that leaders are only allowed in the space between words or lines, which can make certain locations for sites unavailable, as there might already too many leaders routed through that area to insert yet another for the new site. We also wanted the leaders to be monotonous, which means that they will be one of the shortest possible connection between port and site locations. Our final priority was to efficiently use the space for the labels, which meant that each label should initially be placed as close to the available area's border as possible. While this allows for more space in which labels can be placed, it also means that the distance between Label and site often increases, which leads to longer leaders, and often more curves as well.

3.1 Implementation

The program was written in Java, using only JGraphT as additional Library. Similar to Loose's Approach[3], we used a Graph to model the available routing space for the algorithm. As a result of this approach, it is important to know exactly where each word is placed. While Java offers a function that automatically places text in lines of given width, it does so in a way that doesn't allow the content of each line to be read as a String afterwards, which makes it unusable for our purposes. Therefore each word must be placed manually, and to since it isn't the main focus of the program, each series of characters between a space is considered one word that has to be fully placed within one line.

Using the dimensions of each word, we create the graph by creating and connecting nodes in the space between lines at the start and end of each word, and at the end of the text area. We also create special nodes for each annotated word, they are located above the word in a central position. Each node also is located at set co-ordinates, which allows the routing algorithms to easily fulfill the monotonicity constraint. We left some space in between the text area and the label area to allow the labels to be placed more freely. While the routing graph only reaches up to the border of the text area, the connection between the graph and the label is done with an OPO-Leader. As there is the possibility of intersecting with other leaders, this means some extra adjustments have to be done. For demonstration purposes, we also included another algorithm that uses S-Leaders for this section.

References

[1] Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry*, 36(3):215 – 236, 2007.

[2] Philipp Kindermann, Fabian Lipp, and Alexander Wolff. *Luatodonotes: Boundary Labeling for Annotations in Texts*, pages 76–88. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[3] Amrei Loose. Annotation von texten unter berücksichtigung von textzwischenräumen. Master's thesis, Karlsruhe Institute of Technology, 2015.