VISUALIZING ROUTE MAPS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Maneesh Agrawala
December 2001

Abstract

Route maps, which depict a path from one location to another, can be powerful tools for visualizing and communicating directions. Although creating a route map may seem to be a straightforward task, the underlying design of most route maps is quite complex. Mapmakers choose which information is most essential for following the route and they use a variety of cartographic generalization techniques including distortion, simplification, and abstraction to emphasize this essential information

Recently, route maps in the form of driving directions, have emerged as one of the most popular applications on the Web. In contrast to hand-designed route maps, these computer-generated route maps are more precise and contain more information. Yet, in general, they are also more difficult and frustrating to use. The main shortcoming of current route map rendering systems is that they do not distinguish between essential and extraneous information. As a result, these systems cannot apply the generalization techniques used in hand-designed maps to emphasize the information needed to follow the route.

In this dissertation we present a new set of techniques and algorithms for automatically designing and rendering route maps that are far easier to follow than standard computer-generated route maps. We begin by examining research in cognitive psychology and cartography, on how people think about and communicate routes. Based on this analysis we identify the essential information a route map must communicate to support navigation. We then examine a variety of hand-designed route maps and enumerate a new set of cartographic generalization techniques specifically designed to improve the usability of route maps by emphasizing the most essential route information.

Finally, we describe algorithmic implementations of these generalization techniques within LineDrive, a real-time system for automatically designing and rendering route maps. LineDrive designs routes maps to the constraints of the display device and can produce clear, easy-to-read maps for a variety of display devices including standard sized web pages, handheld personal digital assistants and WAP cell phones. LineDrive is publicly accessible at `www.mapblast.com` and we present feedback from over 2200 users of the LineDrive system. The feedback shows that just over 99 percent of users believe LineDrive maps are preferable to using standard computer-generated route maps alone. The response strongly suggests that LineDrive route maps support navigation tasks much better than the standard computer-generated route maps.

Acknowledgments

Over the last six years I have had the privilege of working with a variety of people who have made my time at Stanford an enjoyable and intellectually stimulating experience. I would like to thank all the people who have helped me along the way and contributed to this dissertation.

I am especially grateful to my adviser Pat Hanrahan. In working with Pat, I have learned how to pursue research problems with intellectual rigor and how to critically evaluate my work. Pat has also been incredibly patient with me as I have gone off to explore many different topics only tangentially related to my original thesis topic. I have appreciated the freedom and the work presented in this dissertation would not have been possible without this freedom.

I would like to thank the other members of my reading committee, Barbara Tverksy and Terry Winograd, who provided invaluable guidance as I performed the final stages of my research and I prepared this document. Barbara taught me that in order to produce more effective route maps we must first understand how people follow route maps and form a cognitive understanding of routes. This advice extends beyond route maps to all kinds of visualizations and I eagerly look forward to continue working with Barbara to explore more connections between cognitive psychology and visualization. Terry introduced me to the basic concepts of human-computer interaction through his introductory class on the subject. It was through this class that I learned to appreciate the difficulty of creating interfaces that are truly useful. I have tried to employ that knowledge in the work presented here.

Marc Levoy and Marti Hearst were on my oral defense committee and helped me prepare a strong presentation of my work. Marc was also the first person to teach me computer graphics. His clear presentations of the subject and his enthusiasm for it, were infectious and pushed me to pursue a Ph.D. in computer graphics. While I have admired Marti's research on information visualization for quite some time, it was only when I needed a fifth member for my oral's committee that I had a chance to meet and interact with her. Marti has subsequently become a friend and her knowledge of information visualization has been an invaluable resource.

A couple of years ago, when I first thought of the idea for the route mapping system described in this dissertation, the only person who believed in the project was Chris Stolte. Somehow he even agreed to help me work on it and I am deeply grateful that he did. My work has benefited immeasurably from my collaboration and friendship with Chris. I would also like to acknowledge Christian Chabot who has consistently championed this project and provided much of the early testing of the system. I would also like to thank the many people at Vicinity Corporation who helped integrate the system into the MapBlast! website. Mia Trachinger and John Owens provided the hand-drawn maps that appear in this dissertation.

One of the great things about being at Stanford for so long and working on a variety of different topics is that I have gotten to work closely with a lot of people. I have learned a lot from these collaborations and I would like to thank everyone I have worked with: Andrew Beers, Navin Chaddha, James Davis, Bernd Fröhlich, Pat Hanrahan, Marc Levoy, Tamara Munzner, Greg Niemeyer, Ravi Ramamoorthi, Chris Stolte and Denis Zorin. In the last couple of years I have worked most closely with the visualization group, including Robert Bosch, François Guibretière, Tamara Munzner, Chris Stolte, and Diane Tang. They have been a great sounding board for many of my ideas. They have also spent countless hours reviewing my papers and talks, and my work is much better for it.

The Stanford graphics lab has been a fun place to work because of the many friends I have made in the lab. In particular I would like to acknowledge Matt Pharr and Reid Gershbein who have been comrades through the hardships of graduate school.

None of my work would be possible without John Gerth running the computers and Heather Gentner and Ada Glucksman forging signatures as necessary in order to push all the administrative paperwork through the Stanford bureaucracy.

I am deeply appreciative of Sibyl Diver's companionship and empathy over the course of my graduate studies. In sharing her infectious laughter she has kept me smiling over all these years. Finally I would like to acknowledge the endless love of my family. My sister, Geetika has been a constant source of support. My parents Ashok and Radhika Agrawala have provided guidance, love, and encouragement throughout my life. This thesis is dedicated to them.

To Ashok and Radhika Agrawala

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Route maps, which depict a path from one location to another, are one of the most common forms of graphic communication. Various kinds of route maps have existed for centuries [MB83]. Today, route maps are often created as quick drawings to direct someone to a particular location. Such handcrafted maps are usually very easy to understand and follow.

The recent availability of detailed driving directions via the Web has led to the widespread use of computer-generated route maps. Online mapping services typically provide directions as a set of maps complemented with text descriptions. While the text descriptions work well, the accompanying maps are often very difficult and frustrating to use. Although computer-generated route maps are often more precise and contain more information than hand-designed route maps, we have found that the computer-generated route maps are completely inadequate for navigating a route. While it is usually possible to follow a route using a single hand-designed route maps, it is usually impossible to follow a route using the computer-generated route maps alone. We believe this is because current systems for automatically generating route maps disregard many of the principles and techniques that guide human mapmakers.

Mapmakers make explicit decisions about which aspects of the route are most relevant for a navigator to understand and follow the route. Based on these decisions, mapmakers use a variety of cartographic generalization techniques, including distortion, abstraction, and simplification, to improve the clarity of the map and emphasize only the most important information [Mac95, Mon91][1]. Cartographic generalization, performed either consciously or sub-consciously, is prevalent both in quickly sketched maps and in professionally designed route maps that appear in print advertisements, invitations, and subway schedules [Tuf90a, Hol91, Hol93].

The main shortcoming of automatic route map rendering systems is that they do not distinguish between essential and extraneous information. As a result, these systems cannot apply the generalization techniques used in hand-designed maps to emphasize the information needed to follow the route.

1.1 A Motivating Example

Figure 1.1 shows a standard computer-generated map, while figure 1.2 shows a hand-drawn map for the same route. The lack of differentiation between necessary and unnecessary information in the computer-generated map makes it impossible to follow the route using the map alone. Many turning points are not visible and the route itself is difficult to distinguish from other elements in the map. Thus, online mapping services usually augment such a map with text directions describing the route in detail.

The primary problem with standard computer-generated maps is that they maintain a constant scale factor. For many routes, the lengths of roads can vary over several orders of magnitude, from tens of feet within a neighborhood to hundreds of miles along a highway. When a constant scale factor is used for these routes, the shorter roads shrink to a point and essentially vanish. This phenomenon is particularly problematic near the origin and destination of routes where many quick turns are often required to enter or exit a neighborhood. Although precisely scaled roads might

[1]While cartographers use the term *generalization* to refer the visual distortions, abstractions, and simplifications that occur in maps, cognitive psychologists use the term *schematization* instead. Linguist Leonard Talmy [Tal83] originally used the word *schematization* to refer to the process of reducing and abstracting the detail of a physical scene to a sparse and sketch-like representation. While Talmy was concerned with the process of schematization in language, cognitive psychologists also use the term to refer to the similar process of abstraction and distortion that occurs within hand-crafted sketches and diagrams. In this dissertation we follow the cartographic convention and use *generalization* when referring to these techniques.

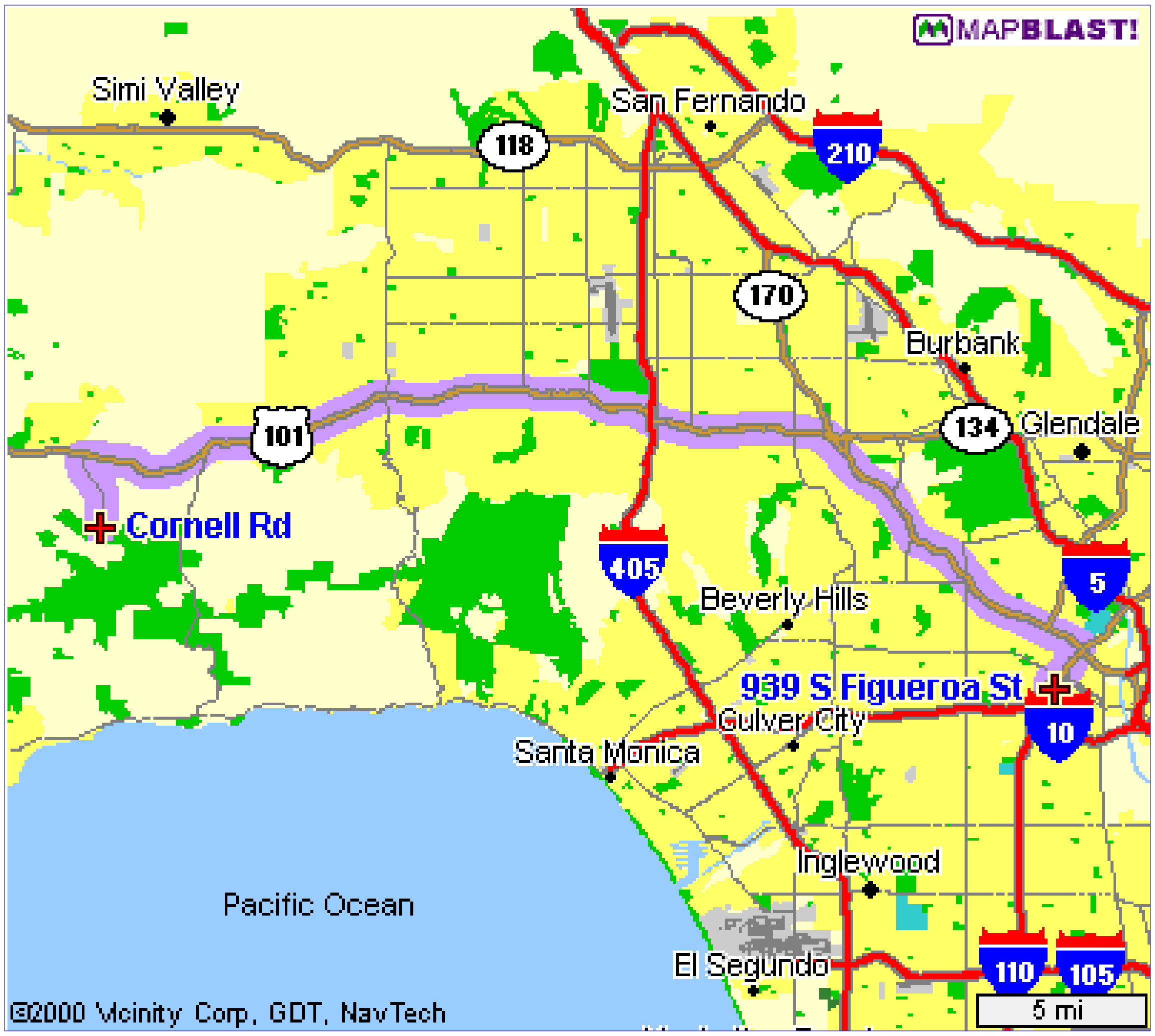


Figure 1.1: Standard computer-generated route map. The map is difficult to use because its large, constant scale factor causes short roads to vanish and it is cluttered with extraneous detail such as cities, parks, and roads that are far away from the main route.

in theory help navigators judge how far they must travel along a road, in practice it is far more important that all roads and turning points are visible. Hand-drawn maps make this distinction and exaggerate the lengths of shorter roads to ensure they are visible.

Another problem with computer-generated maps is that they are often cluttered

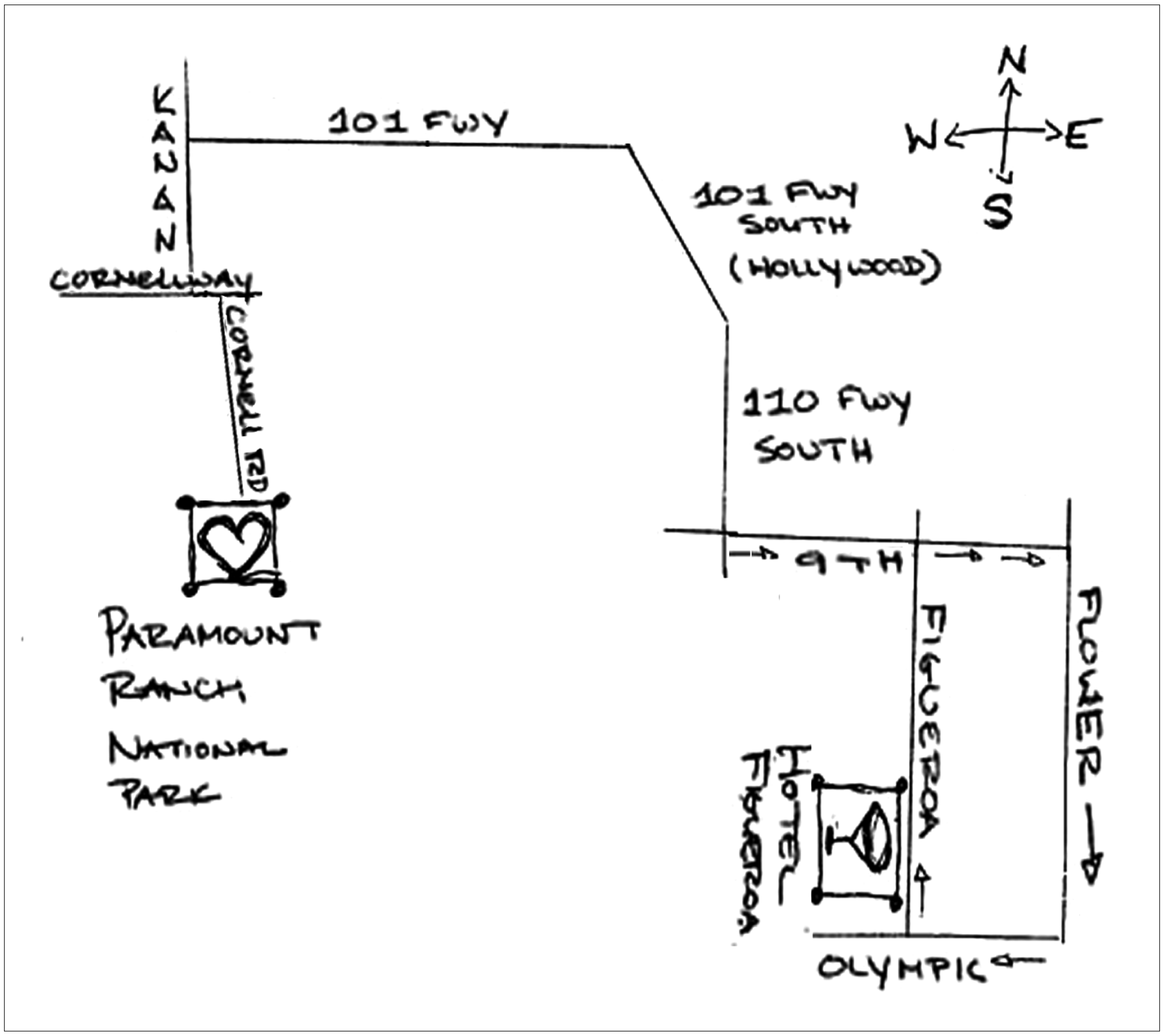


Figure 1.2: Hand-drawn route map. In contrast to the standard computer-generated route map in figure 1.1, this hand-drawn map exaggerates the lengths of short roads to ensure they are visible and it maintains a simple, clean design that emphasizes the most important information for following the route. Note that this map was created without seeing the standard computer generated map, shown in figure 1.1, or the LineDrive map, shown in figure 1.3.

with information irrelevant to navigation. This extraneous information, such as the names and locations of cities, parks, and roads far away from the route, often hides or masks information that is essential for following the route. The clutter makes the maps very difficult to read, especially while driving. Hand-drawn maps, which usually include only the most essential information, are very simple and clean. In the hand-drawn map of figure 1.2, even the shape of the roads has been distorted and simplified to improve the readability of the map. Furthermore, distorting the lengths of shorter roads and removing unnecessary information makes it possible to include helpful navigational aids such as major cross-streets or landmarks.

1.2 Improving Route Map Usability

In this dissertation we present LineDrive, a fully automated system for designing and rendering route maps that are much easier to use than the standard computer-generated maps. LineDrive automatically generalizes maps, much like a human map-maker, to highlight the information essential for following the route, while abstracting or omitting less important details. As a result, LineDrive maps contain many of the advantageous characteristics of hand-drawn maps.

Figure 1.3 shows the LineDrive map for the same route as in figures 1.1 and 1.2. The overall look and feel of the LineDrive map is similar to the hand-drawn map. All turning points along the route are clearly visible and the graphic design is free of extraneous clutter.

There are also several important differences between the LineDrive map and the hand-drawn map. The LineDrive map provides the mileage of each road on the route, with bullets to emphasize each turning point. The rendering style of the road indicates whether it is a highway, a standard residential road along the route, or a cross-street. The hand-drawn map incorrectly shows highway 110 connecting with 9th Street The correct exit sequence requires passing through 8th Place and James Wood Boulevard. as shown in the LineDrive map. Furthermore, the grid in downtown LA and the highway 110 have been reoriented to run north-south in the hand-drawn map, rather than northeast-southwest as they actually do. The LineDrive map correctly maintains the orientation of these roads.

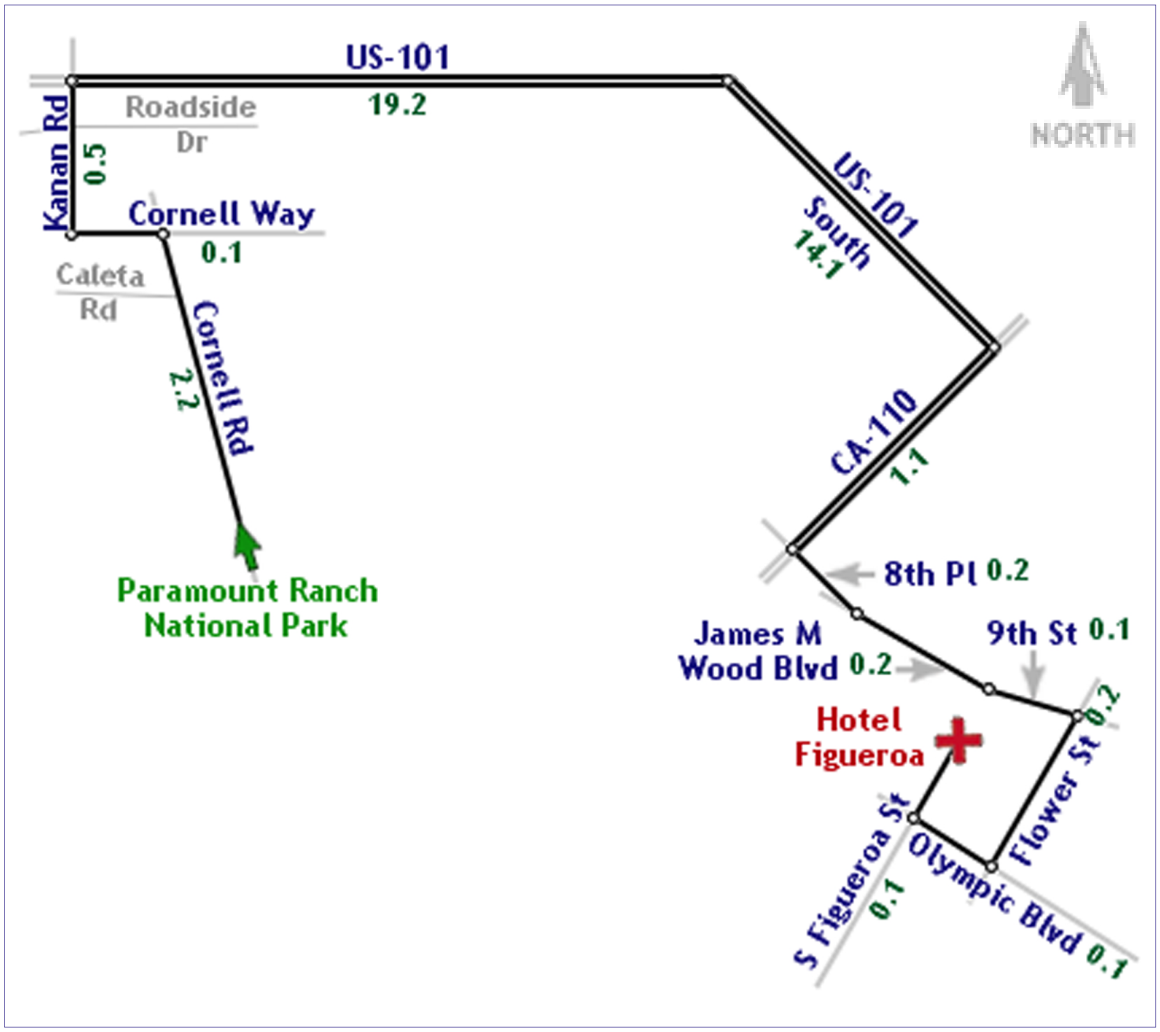


Figure 1.3: LineDrive map. As in the hand-drawn map in figure 1.2, the lengths of short roads are exaggerated and the overall design is simple and clean in this LineDrive map. These attributes make the LineDrive map much easier to use than the standard computer-generated map shown in figure 1.1.

1.3 Producing Route Maps

Although creating a route map may seem to be a straightforward task, as we have seen from the previous example, there are many complex design choices that can affect the usability of the map. Several researchers [LHM99, Den97, DPCB99] have

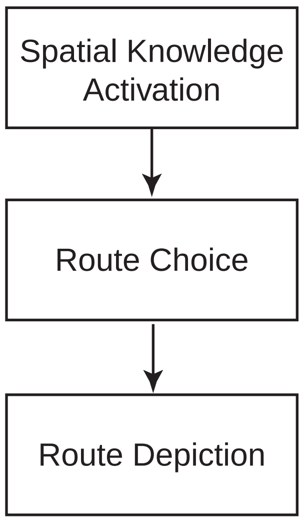


Figure 1.4: Three-stage task model for producing route maps. Producing a route map involves three main tasks as shown in the diagram. The model applies both to human mapmakers and automated mapmaking systems.

modeled the task of creating a route map as a three stage process consisting of (1) spatial knowledge activation, (2) route choice and (3) route depiction. This general three-stage model shown in figure 1.4, applies both to human mapmakers and to automated mapmaking systems. In the remainder of this section we discuss our work in the context of each of these stages.

1.3.1 Spatial Knowledge Activation

The first stage in the model, spatial knowledge activation, is the process of activating a representation of the environment at the appropriate level of detail for the route. While human mapmakers maintain this information in a mental map of the environment, automated systems store this information in a geographic database. The type of spatial knowledge and the appropriate level of detail are largely dependent on the type of transportation used to follow the route. For example, driving directions require knowledge of the road network in a region, while airline directions require knowledge of flight paths and zones in a region. A road database meant for drivers must store waypoints at several hundred foot intervals to capture the curves in a road, while a trail database meant for hikers must show waypoints at shorter intervals to capture the appropriate detail. The level of detail stored in the database is also dependent on the level of detail that will be required in the output route map. If the route map depicts each path as a straight line between each pair of turning points the database can store much less information. Our work is targeted at producing routes maps for driving and we assume that a detailed database of the road network and building-sized landmarks is available. Several companies including Navtech, GDT, and Etak specialize in producing such databases.

1.3.2 Route Choice

The second step in producing a route map is choosing a specific route through the environment. The choice usually requires considering a variety of criteria such as the length of the route, the time required to traverse the route, the time of day at which the route will be followed, the expected traffic on the roads, and the complexity of the route[2]. Mathematically, choosing the route is equivalent to performing a constrained graph search. Each criterion adds a constraint to the search that should be optimized in the final route.

Human mapmakers typically choose a route by combining a divide-and-conquer strategy with a depth-first search. They initially scan their representation of the environment for "important" roads and then perform a depth first search to connect the origin and destination to the important roads [SV86, Gol99, EL82]. The initial search for important roads breaks the problem into a set of intermediate goals. The importance of a road is related to the optimization criteria for the route. For example, when finding the fastest route, highways are often considered most important.

In computer science, route finding is called the shortest-path problem. The environment can be represented as a graph with each road represented by an edge, each road intersection represented by a node, and each optimization criterion represented

[2]Route complexity itself may be measured by several criteria including the number of turns in the route, the number of highway segments vs. residential roads, the expected level of familiarity with the environment of the navigator, as so on.

by a cost on an edge or node. The task is to find a path through the graph that minimizes the various cost functions. With this representation, classic shortest-path algorithms such as Dijkstra's algorithm [CLR90] can be applied to find the optimal route. Elliot and Lesk [EL82] have shown that depth-first search is fairly efficient because road network graphs are usually planar, and contain many nodes while edges are relatively sparse. Hierarchical approaches similar to the combined divide-and-conquer plus depth-first search approach taken by human mapmakers have been implemented more recently [JHR98, Tel99]. These hierarchical methods are much faster than the classic algorithms, but may not always find the optimal route.

Although automated route finding algorithms often produce routes that are slightly different from the route an expert mapmaker would produce, the automated routes are generally correct in that they reach the destination and optimize some basic criteria such as distance or traversal time. Our work is not aimed at finding better routes, but rather at creating better visualizations of a given route. Therefore, we assume that an automated route finding engine is available and can generate a set of roads to travel between any given pair of origin and destination addresses. While our system uses the Telcontar [Tel99] route finding engine, our visualization techniques are independent of the route finding service and can be applied to a route generated by any route finding algorithm or even a human route generator.

1.3.3 Route Depiction

Once the route has been chosen it must be translated into either an aural, haptic or visual representation to facilitate communication of the route to navigators. While aural and haptic route maps can be useful, it is difficult and expensive to produce custom aural and haptic maps for each route. In contrast, many people currently have the capability to print a visual depiction of the route, to take with them on the trip. For these reasons our work focuses on improving the effectiveness and usability of visual route maps. We begin this section with a brief discussion of aural and haptic representations and then examine the range of possible visual representations for route maps.

1.3.3.1 Aural and Haptic Depictions

Streeter et al. [SVW85] have shown that aural route descriptions can be more effective than customized route maps, when used while driving. Although some in-car navigation systems now provide aural driving directions, such systems are expensive and since they are not connected to a central server each system must be updated regularly in order to keep its road database up-to-date.

While haptic representations of a route would be difficult and cumbersome to use for most people, they have been studied in the context of generating route maps for visually impaired navigators. Several systems have been designed to convert visual route maps into tactile route maps. [Mic98]. We believe it may be possible to use the maps produced by LineDrive as input to a haptic route map rendering system. However, because haptic perception and resolution is very different from visual perception and resolution, tactile route maps must be designed optimize a different set of characteristics than visual route maps. We leave the problem of adapting LineDrive maps for haptic display as future work.

1.3.3.2 Visual Depictions

There are many ways to visually depict a route. At the top of figure 1.5 we show the range of possible visual route map depictions parameterized by the fidelity of the depiction in relation to the physical route it represents. Fidelity increases from left to right, or equivalently abstraction decreases and more information is included in the depiction.

A route can be thought of as a sequence of turns and the most abstract visual representation of a route simply presents this sequence in text as a list. The other end of the spectrum is described in Borges' short story *On the Exactitude of Science* [Bor98]. Borges takes fidelity to its logical but absurd extreme, describing the perfect map as a 1:1 replica of the Empire that is the size of the Empire[3].

Between these two endpoints we have a large range of visual representations. Starting from the most abstract depiction, the textual turn list, we can decrease

[3]We are grateful to Barbara Tversky for bringing Borges' story to our attention.

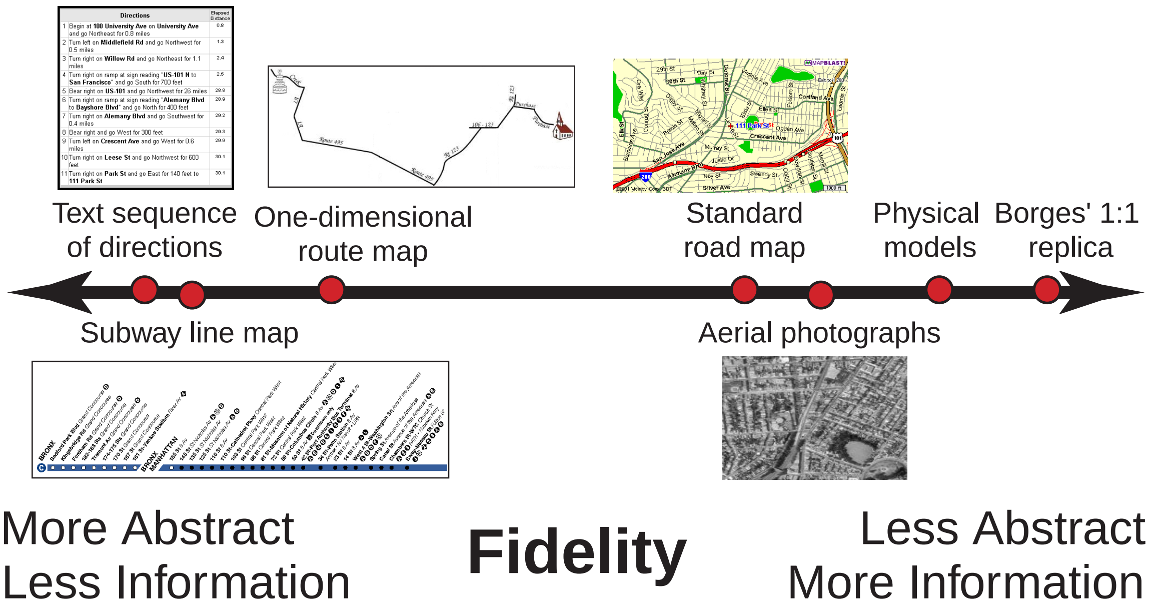


Figure 1.5: Fidelity of route maps depictions. We parameterize the range of possible route map depictions by fidelity. On our fidelity axis depictions on the left are more abstract and contain less information than those on the right.

abstraction and increase fidelity by depicting the route graphically. Maps of subway lines often show the route as a single straight line with circular nodes representing each stop. For passengers, the sequence of stops along the line is the most important information and even the turns in the route are not shown. The subway driver, however, would probably need a map explicitly showing turn information at each turning point in order to make the turns or switch tracks properly.

The simplest route maps designed for navigators typically show only the roads on the route. The route forms a one-dimensional curve in which each road is depicted as a segment, linked at its endpoint to the following road. The turn direction, left-or-right at each segment endpoint is shown implicitly in the map. To further increase fidelity, we can add more roads and landmarks surrounding the route and allow less flexibility in scale. As we increase the information and realism of the map we eventually create a standard road map of the region. Beyond the standard road maps we can increase fidelity in three ways; by creating more realistic imagery (e.g. using

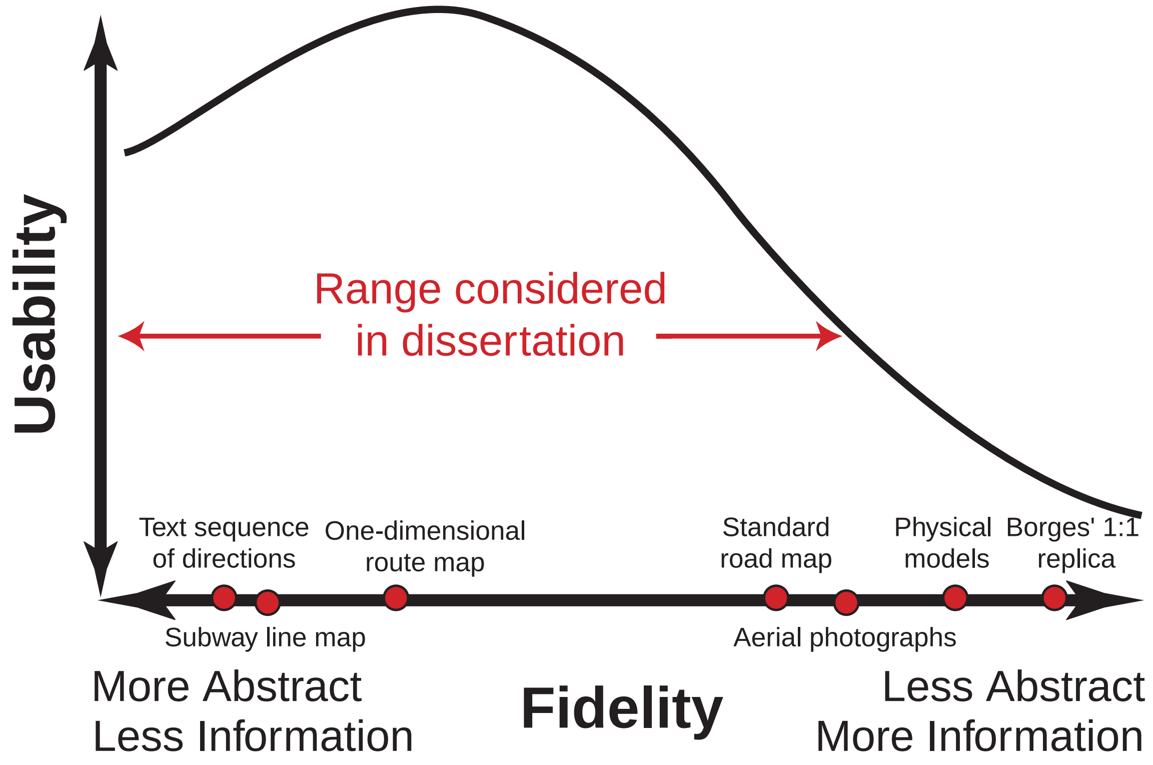


Figure 1.6: Fidelity vs. usability of route maps depictions. If we consider fidelity versus usability we assert that usability peaks near the abstract end of the spectrum. In this dissertation we examine the usability of maps within a range centered about this peak.

aerial photographs), by adding three-dimensional information (e.g. physical models and virtual reality) and by increasing scale (e.g. moving closer to a 1:1 map). The space of possible route map depictions between the simplest map showing only the route and a complete roadmap of a region is large. In this dissertation we explore this space to find the aspects of these depictions that are most relevant to navigators who are using the map while en route.

As shown in the bottom of figure 1.6, we can plot usability of the route depiction against increasing fidelity. We assert that for navigating routes usability peaks very early and the most effective route depictions are near the abstract end of the range. In this dissertation we will show that simple hand-drawn maps are near the peak of this usability function. We then describe LineDrive, a system for automatically creating route maps that maintain the successful characteristics of hand-drawn maps.

1.4 Visualization

Our initial insight for the LineDrive system came from noticing that hand-drawn maps are often far easier to follow than computer-generated maps and standard road maps. However, like much of the recent work in cartographic visualization, our research draws upon ideas from several different intellectual disciplines including cognitive psychology, cartography, graphic design, and computer science.

Two major branches of research in cognitive psychology are cognitive mapping, the study of how people develop mental representations of routes and environments, and wayfinding, the study of how people follow routes. Much of our analysis of the effectiveness of the generalization techniques found in hand-drawn maps is based on this research. Cartographers have long studied the idioms, distortions, abstractions and symbols commonly used in maps. We exploit the shared understanding of common mapping conventions to make our maps easier to understand and to reduce extraneous explanatory information. For instance, people understand that a road shown in a map may not depict all the curves the road actually takes. As a result, in LineDrive we simplify most roads to straight lines. Graphic designers have developed a variety of techniques to emphasize graphical information. We rely on such techniques to highlight the most essential information in our route maps. As a simple example, we place bullets at turning points to emphasize that a turn decision is required at that point in the route.

Within computer science, we utilize techniques from several sub-fields to automatically design and render LineDrive route maps. Search algorithms have been broadly studied and applied to all kinds of problems. In LineDrive we use efficient search techniques to quickly search the high dimensional space of possible map designs in order to find a near-optimal map layout for the roads, labels and context information constituting our route maps. We also use many basic computational geometry algorithms to compute geometric relationships such as intersection points and overlap areas between the elements of our route maps.

While traditional computer graphics provides the basic techniques for rendering the lines, polygons and textures that make up our maps, recent research in non-photorealistic rendering and visualization have also provided great inspiration for our work. Although many distortion and abstraction techniques have been developed in the context of non-photorealistic rendering, applying these techniques to visualization requires first understanding how the techniques can improve the communicative intent of an image. Earlier examples of this approach to visualization include Mackinlay's [Mac86] investigation of methods for automating chart and graph design, Seligmann and Feiner's [SF91] research on the automatic design of intent-based illustrations, and Interrante's [Int97] work on using illustration techniques to improve the perception of 3D surface shape in volume data. In this dissertation we introduce this approach to the automatic design of route maps.

1.5 Contributions

Over the last two decades there has been great interest in examining why good visualizations are so effective. Tufte [Tuf90b, Tuf90a, Tuf97], Holmes [Hol91, Hol93], and others have published collections of the best visualizations for a variety of different domains. One commonality among all the visualizations in these collections is that they were painstakingly crafted by human designers. A basic goal of visualization research is to develop completely automated systems that can design visualizations that are as effective as those created by human designers.

Although this dissertation describes how to build an automated route map design system, the high-level contribution of this dissertation is a general methodology for building an automated visualization system for any given domain. Given a particular data domain we propose the following two-step approach for creating an automated visualization design system.

- **Step 1: Identify cognitive design principles.** We analyze which aspects of the data are most important to communicate through the visualization. We then consider the best examples of hand-crafted visualizations and enumerate the techniques the human designers used to emphasize the most important aspects of the data.

- **Step 2: Algorithmically encode the design principles.** Once we understand the cognitive design principles at a low enough level we develop automated algorithms that embody the principles. Our general approach is to encode the design principles as numerical measures of effectiveness of the visualization and then algorithmically search for the most effective visualization according our these measures.

The first step in our two-step approach is an analysis step aimed at identifying why good hand-designed visualizations are so effective. Tversky et al. [TMB] have shown that two general principles from cognitive psychology apply to the design of effective visualizations:

Congruence Principle: The structure and content of the external representation should correspond to the desired structure and content of the internal representation. For example, since routes are conceived of as a series of turns, an effective external visual representation of routes will be based on turns.

Apprehension Principle: The structure and content of the external representation should be readily and accurately perceived and comprehended. For example, since people represent angles and lengths in gross categories, finer distinctions in diagrams will not be accurately apprehended. In the case of routes, exact angles of turns and lengths of roads are not important.

These basic principles suggest that diagram design can be enhanced by knowing two things: how people comprehend the content that is presented in the diagram; how people apprehend the features of the external representation. In the context of developing and automated route map design system, we begin by examining how people classify the importance of the types of information route map might convey to support navigation. This classification is in accordance with the congruence principle.

We then develop a new set of cartographic generalization techniques designed to improve the usability of route maps. The generalizations are designed in accordance with the apprehension principle.

The second step of our methodology is a synthesis step aimed at creating a system for automatically generating visualizations in the style of the best hand-designed visualizations. We implement a fully automated system for synthesizing route maps that is based on our analysis of hand-drawn route maps. As a result the maps generated by our system are much easier to use than standard computer-generated route maps.

1.5.1 Analysis: Classifying Importance of Route Information

We classify the importance of various types of information found in route maps based on their usefulness to a navigator who is following the route. Our classification is based on cognitive psychology research on wayfinding techniques which has shown that an effective route map must clearly communicate all the all the turning points on the route [Den97], and that precisely depicting the exact length, angle, and shape of each road is much less important [TL99].

1.5.2 Analysis: Enumerating New Generalization Techniques

We enumerate a new set of generalization techniques specifically designed to improve the usability of route maps by emphasizing the essential turning point information. Our techniques are based on an analysis of the generalizations commonly found in hand-drawn route maps, and an understanding how such generalizations can improve the perception and cognition of the map image. In particular we investigate how to distort road length, angle, shape while omitting unnecessary context information in order to clearly present all the turning points along the route.

1.5.3 Synthesis: Automated Map Design System

We describe algorithmic implementations of our generalization techniques within LineDrive, a real-time system for automatically designing and rendering route maps. LineDrive uses the flexibility to distort road lengths, angles and shape to produce maps that are far more usable than standard computer generated route maps. However, this flexibility also creates a large space of possible map designs. We describe how LineDrive performs a focused, randomized search over this space to quickly find a near-optimal map layout.

While the early prototype versions of the LineDrive system produced usable route maps, the system has been iteratively improved based on user feedback and our own experiences using LineDrive maps. We describe some of our major design decisions that were based on user feedback as well as user response to the current system. For example, although our initial system did not support secondary context information such as cross-streets and exit signs, user feedback indicated that this information is extremely useful and it was subsequently added to the system. In the most recent user survey, we have found that 99% of over 2200 respondents believe LineDrive maps are preferable to using standard computer-generated maps alone.

1.6 Dissertation Roadmap

The remainder of this dissertation is organized as follows. In chapter 2 we consider how people think about and follow routes. We then analyze several common examples of route maps and the generalization techniques they contain. Based on this analysis we conclude that the generalizations found in hand-drawn maps produce the most usable route map designs. Much of this design analysis previously appeared in [AS00].

In chapter 3 we describe how our work on route map generalization and automated route map design builds on previous computer science research on shape simplification and automated layout for 2D displays.

Chapter 4 presents an overview of our end-to-end route mapping system which takes an origin-destination address pair as input and produces a web page including the LineDrive map as output. LineDrive is the main component of the end-to-end system and is itself composed of five stages. We describe each of these stages in separate chapters beginning with shape simplification in chapter 5. Chapter 6 presents our algorithm for finding a layout for the roads that allows all the roads to be visible. Then in chapters 7 and 8 we describe how we add text labels, context information, and decorations to the map. We originally presented in the LineDrive system in [AS01].

In chapter 9 we describe methods for both modifying and enhancing the basic LineDrive maps based on the constraints imposed by the display device. These techniques allow LineDrive maps to remain highly effective even on limited resolution, small-screen devices such as personal digital assistants.

Finally, we present system performance and user feedback results in chapter 10. We finish with conclusions and a discussion of future work in chapter 11.

Chapter 2

Route Map Design

Understanding how people think about and communicate routes can provide great insight into what information should be emphasized in a computer-generated route map. In order to design a better route map, we begin with a brief summary of the history of route maps. We then analyze the tasks involved in building a mental representation of an environment and following a route – commonly referred to as *wayfinding*. From this analysis we identify the essential information a route map must communicate to support wayfinding. Next we analyze how several of the most common styles of route mapping often fail to emphasize the essential route information. Finally we describe how we use specific generalization techniques within LineDrive, including distortion and abstraction, to present the essential route information in a clear, concise, and convenient form.

2.1 A Brief History of Route Maps

The history of route maps begins with the earliest forms of human generated graphics. In this section we present a brief overview of the variety of guises in which route maps have appeared over their long history. Our overview is not a comprehensive review but rather presents examples that have provided inspiration for our work. Bell's [Bel95] study of strip maps is an excellent reference that provides a more complete history.

Archaeologists and cartographers studying the artifacts of ancient cultures have posited that one of the purposes of cave painting, clay markings, stone carvings and stick assemblies was to communicate routes [MB83, Thr96]. Early examples of strip maps, dating from about 2000 B.C., have been found in ancient Egyptian tombs. The Egyptians painted a basic form of strip maps on the bottoms of coffins depicting routes for the dead to follow to the afterlife [Bel95, Thr72]. The Romans created route itineraries that consisted of lists of stops along the route, such as villages, towns and cities, and the distances between them. These later evolved into graphical stripmaps [MJ87]. Today, subway routes are often depicted in the same fashion as the Roman itineraries, as shown in the thumbnail image in figure 1.5.

During the 17th and 18th centuries strip maps were used throughout Britain. John Ogilby's famous road atlas, *Britannia* [Ogi89], published in 1675, took this form. Bell [Bel95] argues that strip maps were popular at the time because there were very few routes between destinations. Tufte [Tuf90a] presents examples of strip maps created in London at the end of 18th century that are very similar to their modern day equivalents, the American Automobile Association triptik. Tufte also points out that while the Japanese strip maps from this era were similar to their European counterparts, the Japanese maps use a single, long, thin, sheet of paper to show the entire route, rather than splitting the map across multiple sheets.

In the early 20th century, with the widespread use of the automobile, AAA's triptik gained mass popularity in the United States. The most important property of these maps is that they are customized to emphasize the particular route chosen by the navigator. The complexity of the overall road network is hidden, thereby making the map easier to follow. An example of a triptik is shown in figure 2.2.

In 1931 Harry Beck designed an extremely influential map for the London's subway, the London Underground. His map used a variety generalization techniques to emphasize the routes. Beck locally distorted the scale and shape of the subway routes while preserving the overall topology of the route network. He color-coded the routes and used intuitive icons to represent stations and crossover points. The resultant map is simple, clean and easy-to-read. It has heavily influenced map design in general and subway map design in particular. The design of almost all current subway maps can be traced back to Beck's London Underground map.

Route maps are commonly found in large-scale tourist destinations such as museums, amusement parks and monuments. Tufte [Tuf90a] presents a map from Japan's Ise Shrine that extends the generalization techniques used by Beck in the London Underground map to include multiple perspectives. The Ise Shrine map contains scale distortions and employs two distinct perspective systems, an oblique bird's eye view for the main part of the image showing the pedestrian route and an orthographic view on the left side of the image showing train routes and how they connect with the pedestrian routes.

Today, Web-based route mapping services are commonly used to obtain driving directions. These services typically provide directions as an overview map complemented with text directions. In some cases they also provide turn-by-turn focus maps. An example of a Web-based overview map appears in figure 1.1 and an overview/focus map collection appears in figure 2.3. While the text directions generally work well, it is our experience that the accompanying maps are often difficult and frustrating to use. Although such computer-generated route maps are largely negative examples, they provided an initial impetus for the work presented in this dissertation. Our frustration with these maps led to primary goal of this dissertation, which is to develop an automated system for creating more usable route maps.

2.2 Mental Representations of Routes

Cognitive psychologists commonly refer to the mental representation of spatial information as a *cognitive map* [Tol48], while the act of following a route is called *wayfinding* [Gol99]. Building a cognitive map of an environment and wayfinding within it are tightly coupled processes that often occur concurrently. When navigators encounter a new environment they either travel through it (e.g. wayfinding) or view it from a survey perspective (e.g. through a map or some high point in the environment) and then integrate the information into a cognitive map. Once a cognitive map is built navigators rely on it to find their way through the environment. In this section we consider the relationship between cognitive map and wayfinding.

2.2.1 Cognitive Maps

Cognitive maps encode three basic categories of features: points, lines and areas[1]. For navigators, the most important point features in an environment are *landmarks*, the most important linear features are *routes* and the most important area features are *networks*. A commonly accepted theory [Lyn60, Gol99, Cho99] is that cognitive maps are continuously built up from these elementary features in three stages. Navigators first learn individual landmarks as disconnected points in the environment. Navigators then learn routes as a linked sequence of landmarks and finally they learn the network of routes covering a region.

Many studies have shown that cognitive maps are quantitatively inexact and contain many distortions [Tve81, Tve92, But86, Gol99]. Even an expert, such as a cab driver, who is extremely familiar with an environment will generally have trouble estimating Euclidean distances between locations [Pai69, Cha83]. Yet, even though cognitive maps lack quantitative accuracy, they maintain topological consistency and can provide reliable information on topological relations such as inclusion, exclusion, and connectedness. Chown [Cho99] argues that the topological information is enough to properly navigate the environment and the distortions allow for a more compact representation of the cognitive map.

2.2.2 Wayfinding

Navigators travel through an environment for three reasons: to reach a familiar destination, to explore the environment or to find a novel destination. Allen [All99] describes several of the most common wayfinding techniques that are used to accomplish these wayfinding tasks. Table 2.1 shows each wayfinding technique and the wayfinding tasks it supports. We summarize Allen's descriptions of the techniques.

Oriented search is the simplest wayfinding technique, but is also the least efficient. Starting at an origin the navigator visually (or aurally in the case of the visually

[1]These categories can be refined further. For example Golledge [Gol99] adds surfaces to this list, while we group surfaces into the area category. Lynch [Lyn60] splits the points category into landmarks and nodes, and the lines category into paths and edges. He also refers to areas as districts.

Wayfinding techniques	Wayfinding tasks: Travel to familiar destinations	Exploratory travel	Travel to novel destinations
Oriented search	X	X	X
Following a marked trail	X	X	X
Habitual locomotion	X		
Path integration	X	X	
Piloting between landmarks	X	X	X

Table 2.1: Wayfinding tasks and techniques. The set of wayfinding tasks that can be achieved with Allen's [All99] set of wayfinding techniques. *(After table 2.1 in Allen [All99])*

impaired) locates a destination and then searches for a path to the destination. Oriented search is most useful for exploratory travel, especially when the navigator has little prior knowledge of an environment.

Following a continuously marked trail is the most reliable wayfinding technique and can be applied to all three wayfinding tasks. While this technique reduces cognitive demands on the navigator, continuously marked trails can be very expensive to create and are therefore extremely rare.

Habitual locomotion occurs when the navigator becomes extremely familiar with a route. The repetitive pattern of the route is imprinted on the navigator and the movements required to complete the trip become somewhat automated. For most commuters the daily route between home and work is habitual and they pay minimum attention to the details of the trip.

Path integration requires navigators to self-monitor their velocity and acceleration to compute the distance and direction to their destinations via dead reckoning. Although humans are not well-equipped to precisely estimate velocity and acceleration path integration can be used to travel to familiar destinations and for some exploratory travel.

Piloting between landmarks requires the the navigator to follow a sequence of landmarks to reach the destination. Each landmark is associated with a path leading to the next landmark and therefore the route is completely defined by the sequence of landmarks along it. Given the sequence of landmarks this wayfinding technique can be used for all three wayfinding tasks[2].

Of these wayfinding techniques only *oriented search* and *piloting between landmarks* apply to all three wayfinding tasks. While oriented search is useful in some situations, piloting between landmarks is far more efficient and it is the most commonly used wayfinding technique. Therefore we focus our work on this approach.

Piloting between landmarks requires the navigator to know the sequence of landmarks from origin to destination. Each landmark is essentially a turning point on the route and wayfinding consists of two alternating activities: following a road until reaching a turning point and then changing orientation to follow another road [Den97]. As described in the previous section the sequence of turning points (or landmarks) is precisely the sequence that is built in the second stage of cognitive mapping. When the sequence of turning points is not stored in the navigator's cognitive map some form of a route description such as text directions or a map, is required in order to determine the turning point sequence.

2.3 Information Conveyed by Route Maps

The research on cognitive mapping and wayfinding has shown that routes are often thought of as a sequence of turns [Tve92, Mac95, All99]. Furthermore, verbal route directions are typically structured as a series of turns from one road to the next with emphasis on communicating turn directions and the names of the roads [DPCB99]. Tversky and Lee [TL99] have shown that hand-drawn maps maintain a similar structure with emphasis on communicating the roads and turn direction at each turning point.

It follows then that computer-generated route maps should also emphasize the turning point information. Yet, even though it is possible to follow a route map that only indicates the road names and turn direction at each turning point, additional information can greatly facilitate navigation. The information that can be depicted

[2]Our definition of landmarks refers both to physical structures such as building as well as other uniquely marked points along the route. For example intersections between roads are uniquely marked by the names of the roads at the intersection.

in route maps falls into three broad classes: turning point information, local context, and overview context. We consider each of these in order of importance for the navigator.

2.3.1 Turning Point Information

A turning point can be defined by a pair of roads (the road entering and the road exiting the turning point) and the turn direction (left or right) between those two roads. Route maps depict this information visually, so navigators can quickly scan the map to find the road they are currently following and look ahead to determine the name of the next road they will turn onto. Once the name of the next road is known, the navigator can search for the corresponding road in the physical world. The turn direction specifies the action navigators must take at the turning point. The turning point information is essential for a route map to be useful. It would be nearly impossible to follow a route without it.

2.3.2 Local Context

Local context consists of information about the route itself as well as the environment immediately surrounding the route. For example, if the map labels each road with the distance to be traveled along that road, navigators can use their odometer to determine how close they are to the next turn. Cross-streets and local landmarks along the route, such as buildings, bridges, rivers, and railroad tracks, can also be used for gauging progress. Navigators can also use this information to verify that they are still following the route and did not miss a turn. The cross-street immediately before a turn is especially useful because it can warn the navigator that the next turn is approaching. Similarly, providing the first cross-street after each turning point can cue navigators that they missed a turn. However, local context is not essential for following the route and is usually included in the map only when it does not interfere with the primary turning point information.

2.3.3 Overview Context

Overview context consists of large scale area landmarks as well as global properties of the route. Depicting large scale landmarks such as cities and bodies of water near the route can make it easier to correlate the map with the physical world. Navigators can use these landmarks to orient the route with respect to the local geography. Moreover, such landmarks can help navigators quickly locate their position in the map. For example, navigators who know they are in San Francisco can quickly narrow in on the streets shown near the city. Similarly, maintaining the overall shape and heading of the route (e.g. north-south vs. east-west) can also make it easier to place the route in the larger geographic context. However, overview context is the least important of the three classes of route map information. Like local context, it is not required for traversing the route.

2.4 Ranking the Information Classes

Although route maps may be used before a trip for planning purposes, they are most commonly used while actually traversing the route. Our ranking of the three classes of route map information reflects this fact. For a navigator, en route to a destination, finding the next turn on the route is often the most important task. In many cases, navigators are also drivers and their attention is divided between many tasks. As a result, they can only take quick glances at the map. Therefore, maps must convey the turning point information in a clear, easy-to-read manner and must have a form-factor that is convenient to carry and manipulate. Hand-drawn maps achieve all of these requirements and emphasize the turning point information by omitting some of the local and most of the overview context information. For these reasons we treat turning point information as essential, while local context is less important and overview context is least important.

Route maps created based on this ranking are not ideal in all situations. In particular, if the navigator strays from the route, context information such as nearby roads is far more important than the original turning point information for the route.

Similarly, once navigators reach their destinations they often need context information to find local parking areas. Hand-drawn route maps that emphasize turning point information by omitting context may not be very useful in such situations. A general purpose road map of the area, which emphasizes all regions of the map equally and therefore contains the appropriate context information, would be much more helpful.

However, we are most interested in improving the usability of route maps designed for navigators who are on the main route. The additional context information in a standard road map map is distracting and makes it difficult to find the route being traversed. Although it is can be useful to include some context information in a route map, it should only be included when it does not reduce the clarity of the turning point information.

2.5 Analysis of Current Route Mapping Styles

Most current styles of route maps fail to present the essential turning point information of the route in a clear, easy-to-read manner within a convenient form-factor. We analyze five of the most common route mapping styles and consider the design choices made in each.

2.5.1 Route Highlight Maps

Route highlight maps simply highlight the route on a general road map of the region. Since the purpose of a road map is to provide an understanding of the entire road system in a region, they typically employ constant scale factors and display extraneous context throughout the map.

An example of this route mapping style appears in figure 2.1. The primary problem with this style is the constant scale factor which makes it impossible to see short roads and their associated turning points. Since general road maps are not optimized to show any particular route, a route highlight map will often suffer from both a large scale factor that hides important information and an inconvenient form-factor that is too large to easily manipulate while driving.

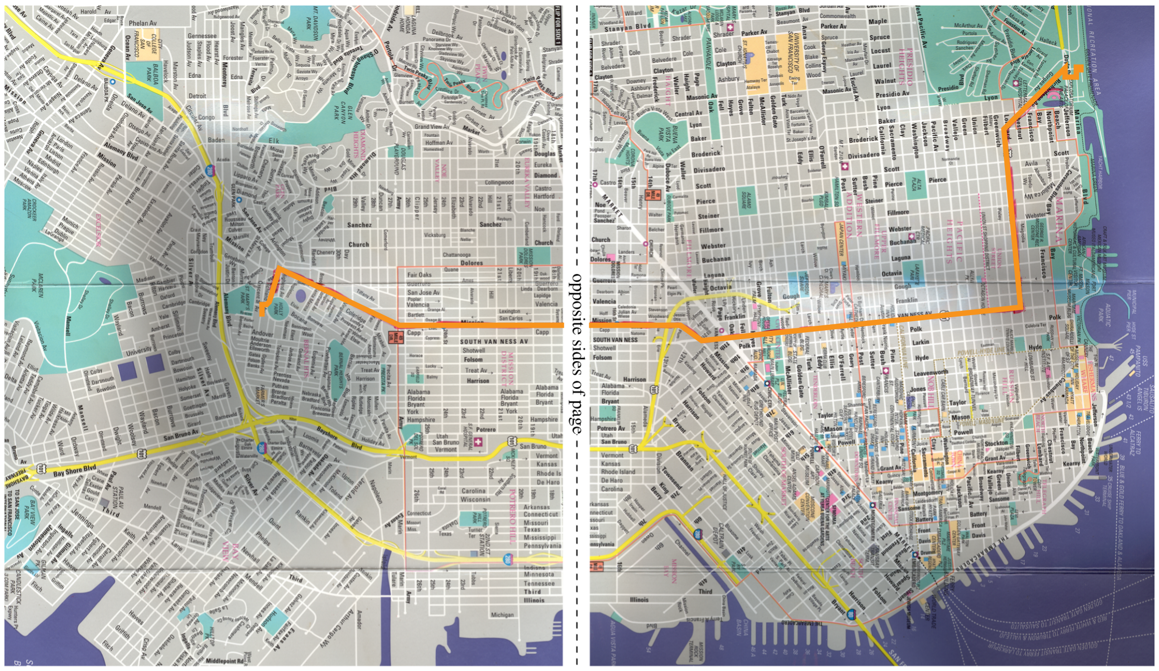


Figure 2.1: A route highlight map. Although the route is highlighted in orange, the map shows the entire city of San Francisco and therefore turns at the beginning and end of the route are barely visible. The extraneous context information and the variety of colors can make it difficult to find the route with a quick glance. The form-factor of the map is inconvenient because the route is on both sides of the map and therefore the navigator must search both sides when looking for the route.

The only property differentiating the route from the other roads is the highlighting. Therefore the clarity of the route depends on the highlighting technique. Usually the route is distinctively colored, but because general road maps provide context information over the entire map and often make liberal use of color to encode properties of this extraneous information, it can sometimes be difficult to distinguish the route from the other elements of the map.

2.5.2 Strip maps

Strip maps, or triptiks, are similar to route highlight maps, but are specifically designed to communicate a particular route. As shown in figure 2.2, strip maps address

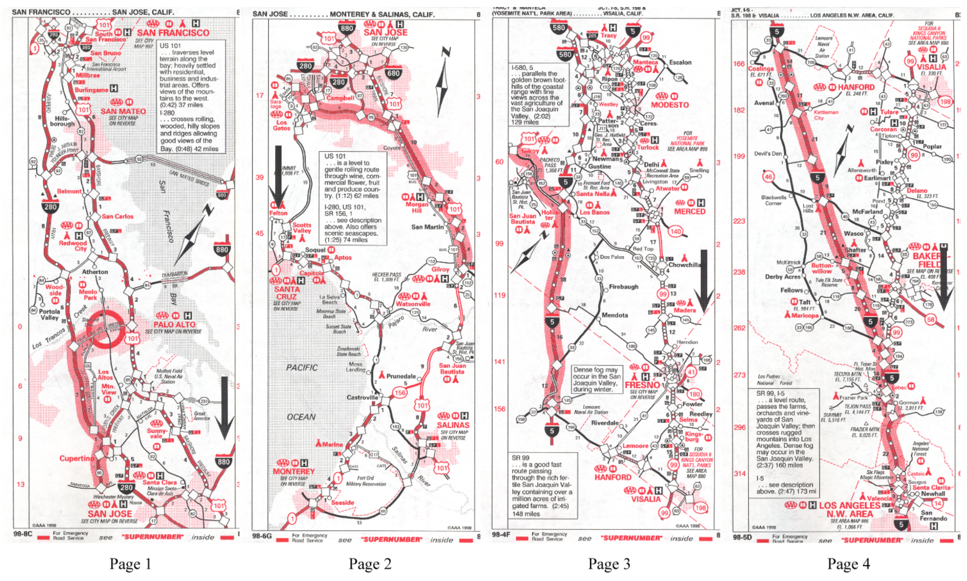


Figure 2.2: A strip map. A strip map depicting the first part of a route, highlighted in red, from Palo Alto to Los Angeles. The scale factor and orientation vary from page to page in order to center the route making it difficult to form a general understanding of the overall route.

the issue of varying scale by breaking the route up onto multiple pages. Each page is oriented so that the route runs roughly down its center and although the scale factor is fixed for each page, the scale factor can differ across pages. The differing scale factors allow strip maps to depict more detailed turning point information where needed. However, because the map stretches over many pages and the orientation and scale factor varies from page to page, forming a general understanding of the overall route can be difficult.

2.5.3 Overview/Focus Map Collections

Overview/Focus map collections consist of a set of maps rendered at different scales that present a single route. This the most common route mapping style available through the Web-based route mapping services and an example of such an overview/focus map collection is shown in figure 2.3. A constant scale factor is used within each map, but the scale factor differs across the maps. One of the maps is

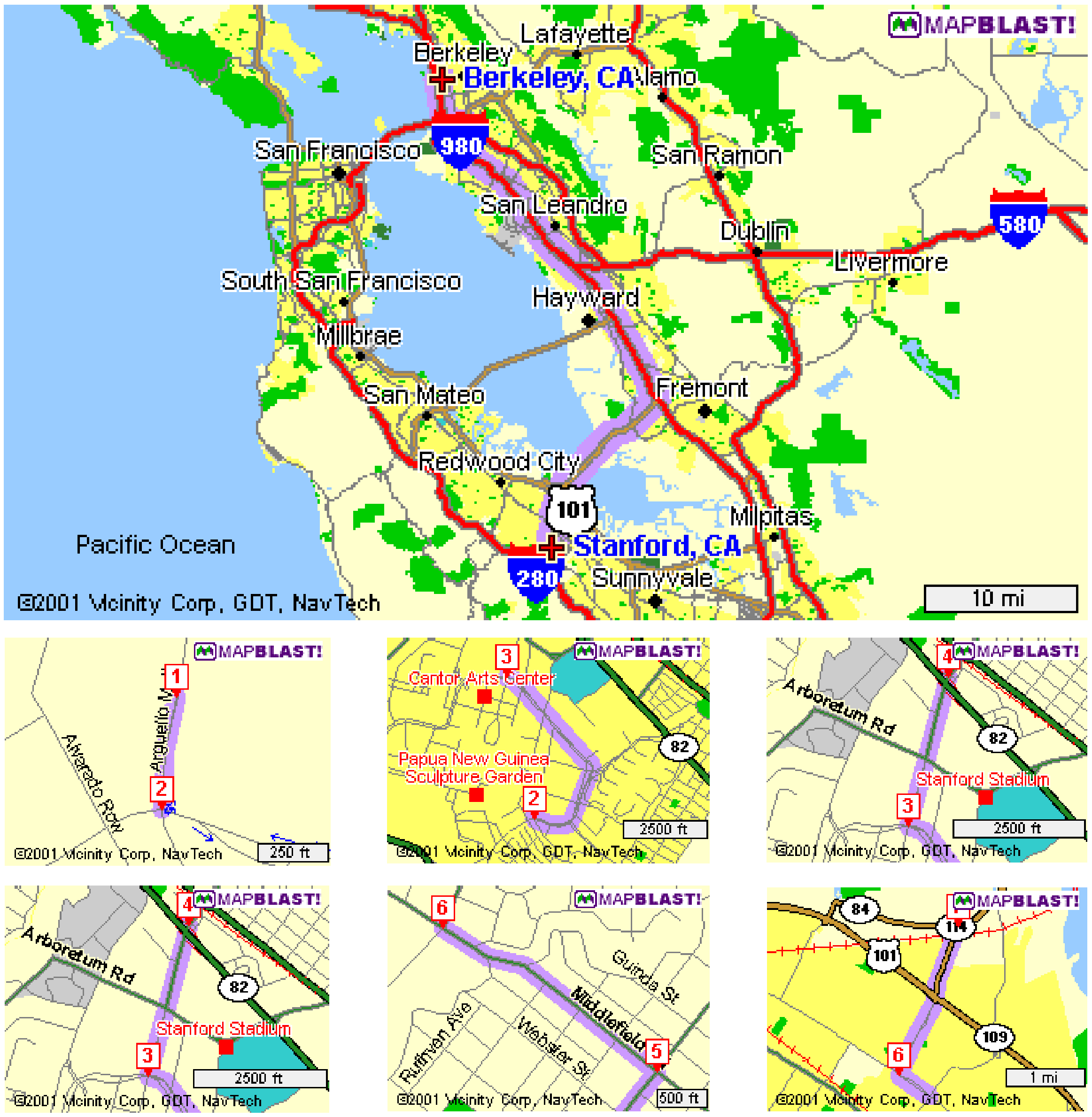


Figure 2.3: An overview/focus map collection. A route from Stanford to Berkeley depicted with an overview/focus map collection. The overview show the entire route, but turning point information is not visible. The turn-by-turn focus maps depict each turn individually, but the varying scale factors and orientations make it difficult to understand how the maps correspond to one another.

scaled so that it provides an overview of the entire route. This overview map is essentially a route highlight map and suffers many of the problems associated with that route mapping style. Since the scale of the overview map often reduces the readability of local turn information, focus maps showing turn-by-turn steps are also provided.

While this may seem like an effective combination, in practice the two sets of maps can be extremely difficult to use. The overview map rarely presents more than the overall direction, and placement of the route within the larger geographic context. Although turn-by-turn maps provide detailed turning point information, the use of distinct maps for each turn, often with differing orientation and scale, makes it difficult to understand how the maps correspond to one another. No single map provides a complete description of the route at the appropriate level of detail. Thus, just as with strip maps, the navigator may have difficulty forming an overall understanding of the route, leading to frustration and confusion.

2.5.4 2D Nonlinear Distortion Maps

To ensure clear communication of the turning point information, different regions of the route often need to be depicted at different levels of detail. Recently several researchers [CCF95, Kea98] have attempted to use image warping techniques on general road maps and subway route maps in order to emphasize turning point information while showing all of the surrounding context. These methods allow users to choose regions of the map they wish to focus on and then apply nonlinear distortions, such as spherical magnification to enlarge these focus regions.

Such 2D distortion allows detailed information to be displayed only where relevant and often produces route maps that can be conveniently displayed on a single page. However, as shown in figure 2.4, a major problem with 2D image distortion techniques is that there are areas of extreme distortion at the edges of the focal points which make the overall route difficult to understand and follow. The severe distortion of the text labels makes the map particularly difficult to read. One way to improve this image warping approach would be to apply the warp only to the underlying lines representing the network of routes. The unwarped text labels could then be placed

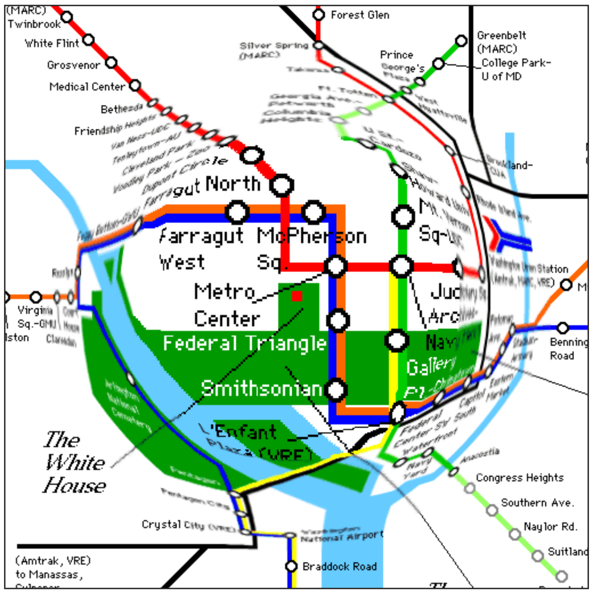


Figure 2.4: A 2D nonlinear distortion map. Spherical magnification is applied to a subway map to emphasize the important stations on the route, but extreme distortion at the edges of the spherical region makes it difficult to understand the surrounding context. Moreover, the severe distortion of the text labels makes the map very difficult to read.

appropriately in the warped image so that the text would remain readable. This approach is common in graph layout systems such as Lamping and Rau's [LR94] hyperbolic tree browser. They apply warps to the node-edge structure of the graph and then overlay unwarped text labels. However, to our knowledge approach has not been applied to visualizing routes.

2.5.5 Hand-drawn Route Maps

One existing route mapping style, the hand-drawn map, manages to display each turning point along the route clearly and simultaneously maintain simplicity and a convenient form factor, as exemplified in figures 1.2 and 2.5. Instead of using a constant scale factor hand-drawn maps only maintain the relative ordering of the roads by length. Although each road is scaled by a different factor, longer roads

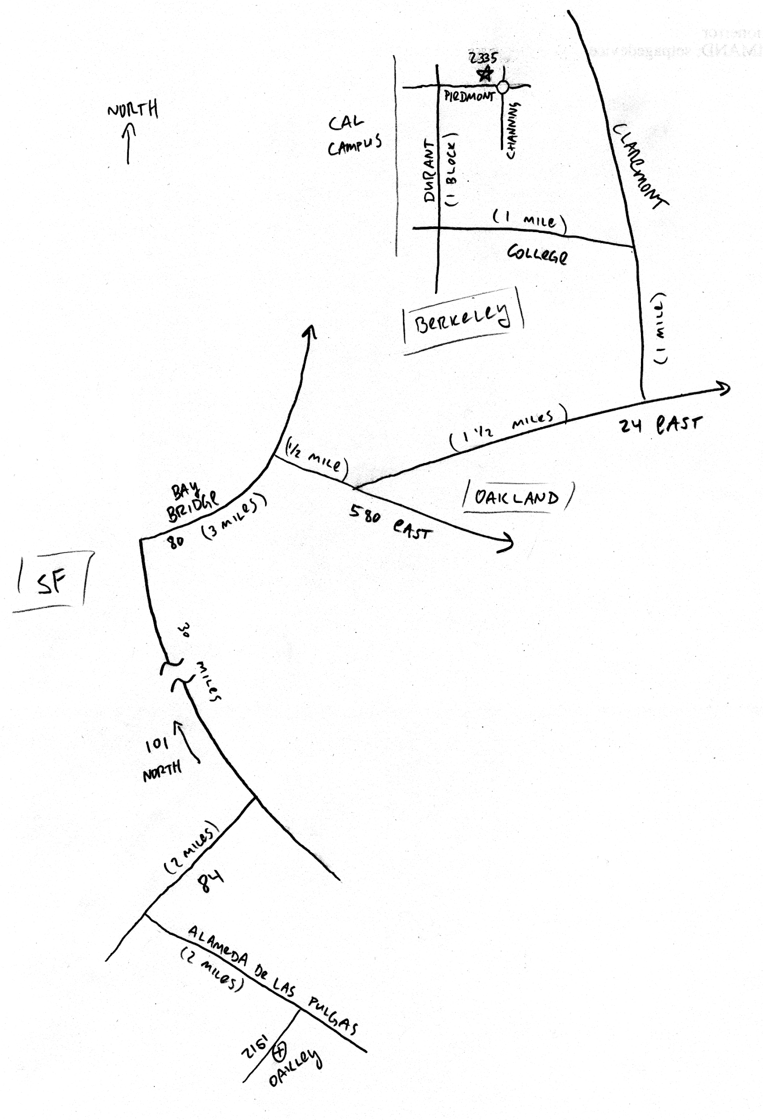


Figure 2.5: A hand-drawn route map. Roads lengths, angles and shape are distorted in order to emphasize the turning point information. Some context information such as distances and city names near the main route facilitate navigation without reducing the clarity of the overall map. Note that the mapmaker created this map from memory, without the aid of any reference maps.

appear longer than shorter roads.

Hand-drawn maps also omit most contextual information that does not lie directly along the route. This strategy reduces overall clutter and improves clarity. As shown in figure 2.5, at most a sparse set of local context information such as distances along each road and nearby city names are depicted because they greatly facilitate navigation with little impact on the overall readability of the map. The angles formed by the roads are regularized and road shape is generalized. Roads are often depicted as generically straight lines or simple curves. These distortions make the map simpler and thereby help to emphasize the essential turning point information.

2.6 Generalizing Route Maps

The generalization techniques used in LineDrive are based on those found in hand-drawn route maps. As we saw in the previous section, hand-drawn route maps use four basic types of generalization techniques: (1) the lengths of roads are distorted, (2) the angles at turning points are altered, (3) the shapes of the individual roads are simplified, and (4) extraneous context is reduced and the graphic representation of the roads and turning points are carefully chosen to emphasize the turning points.

Length Generalization: Hand-drawn maps often exaggerate the lengths of shorter roads on the route while shortening longer roads to ensure that all the roads and the turning points between them are visible. Often the map designer does not know the exact length of the roads [Tve92] and only knows their lengths relative to one another. The flexibility of relative scaling allows hand-drawn route maps containing roads that vary over several orders of magnitude to fit within a conveniently sized image (i.e. a single small sheet of paper)and remain readable. The distortion is usually performed in a controlled manner so that shorter roads remain perceptually shorter than longer roads, and the overall shape of the route is maintained as much as possible.

Angle Generalization: Mapmakers often alter the angles of turns to improve the clarity of the turning points. Very tight angles are opened up to provide more space for growing shorter roads and labeling roads clearly. Roads are often aligned with the horizontal or vertical axes of the image viewport, to form a cleaner looking, regularized map [Tve81, Byr82]. Such angular distortions are acceptable because reorienting correctly requires knowing only the turn direction (left or right), not the exact turning angle.

Shape Generalization: Since a navigator does not need to make active decisions when following individual roads, knowing the exact shape of a road is usually not important. Simplifying the road shape removes extraneous information and places more emphasis on the turning points, where decisions need to be made. Roads with simplified shape are perceptually easier to differentiate as separate entities and are also easier to label clearly.

Graphic Generalization: The main technique for emphasizing turning point information in hand-drawn maps is to reduce extraneous context information by simply omitting it from the map. The route, depicted as a one-dimensional curve, is then the primary graphical element in the map. The negative space surrounding the route creates a simple, clean design that is clear and easy-to-read.

Some hand-drawn route maps use standard graphic design techniques to enhance the turning point information. For example, roads may be color-coded or their drawing style might change depending on the importance of the road. Highways might be drawn as double lines while residential roads are drawn as a single thin line. Professionally designed route maps sometimes place bullets at each turning point in order to emphasize that a decision is required at each of those points. Finally, the context information that does appear in hand-drawn maps is often de-emphasized. For example, cross-streets may be sketched in a lighter color to reduce interference with the main route.

2.7 Errors Due to Excessive Generalization

While these generalization techniques can increase the usability of the route map, they can also cause confusion and mislead the navigator if carried to an extreme.

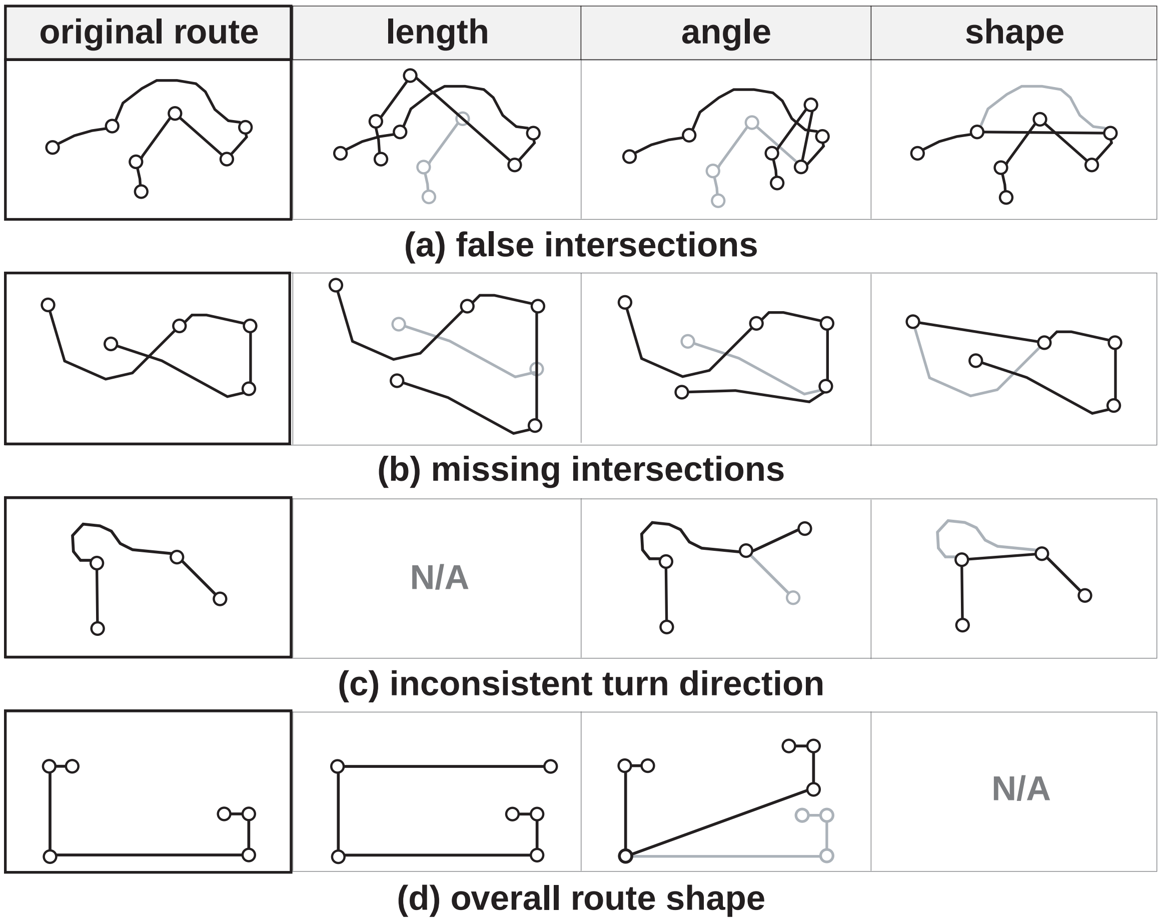


Figure 2.6: Errors due to excessive generalization. Excessive generalization can cause four types of errors in the topology and shape of the route. Each column shows the route after generalizing road length, angle or shape. For comparison, the undistorted route is shown in gray. While humans mapmakers are careful not to over-generalize a map to the point of introducing such errors in the map, automated mapmaking systems must explicitly check for these errors. (a) The original route does not contain an intersection but generalization causes a false intersection. (b) The original route contains an intersection but after generalization it is missing. (c) Generalization changes the turn direction so that a right turn appears to be a left turn or vice versa. Note that distorting road length cannot cause this error. (d) Generalization causes drastic changes in the overall shape of the route. Note that shape generalization cannot generate this error because each road is simplified individually and road endpoints are never removed.

By simplifying road shape and distorting road lengths and angles, it is possible to drastically change the topology and overall shape of the route. We consider four types of errors due to excessive generalization, as shown in figure 2.6: false intersections, missing intersections, inconsistent turn directions and incorrect overall route shape.

False Intersections: A false intersection occurs in a route map when two roads are drawn as intersecting, even though they do not in fact intersect. All three forms of road generalization can generate false intersections. False intersections are topological errors that can deceive navigators into thinking that the route contains a loop or a shortcut when no such shortcut really exists.

Missing Intersections: A missing intersection occurs when the original route contains an intersection, but the intersection is missing in the generalized route map. This usually occurs when one road passes over another road as is often the case at highway interchanges. Although missing intersections, like false intersections, are topological errors that can deceive navigators they are less misleading than false intersections.

Inconsistent Turn Direction: Generalization can cause a right turn to appear as a left turn or vice versa. Inconsistent turn direction is always generated by either angle or shape generalization. Length generalization only affects the length of each road and therefore cannot create an inconsistent turn direction. Because turn direction is a fundamental component of turning point information such errors can cause severe problems for navigators.

Overall Route Shape: Generalization can also create drastic changes in the overall shape of the route. For example, growing one road while maintaining all the others at their original size via length generalization can alter the overall direction between the origin and destination of the route. Similarly angle generalization could cause an east-west route could become a north-south route. Shape generalization essentially requires simplifying road shape. The most extreme form of road simplification replaces each road with a straight line connecting its endpoints. Since each road is simplified individually and the endpoints of each road are never removed, extreme changes in overall route shape are not possible due to shape generalization. Although errors in overall route shape can lead to a distorted cognitive map of the route and make it difficult to estimate distances or orientation, such errors are the least detrimental of the four errors we consider.

Human mapmakers rarely generalize routes to the extreme required to induce these errors[3]. Generalization is performed almost subconsciously, with constant error-checking to ensure that such misleading effects are not generated. Automated mapmaking systems, however, must be explicitly perform such error-checking. The bulk of LineDrive's road layout algorithm, as described in chapter 6, is designed to perform these checks. Like human mapmakers, LineDrive carefully generalizes road length, angle and shape, to dramatically improve the usability of the route maps.

[3]However, it is fairly common for human mapmakers to reorient sections of a route so that they align with the cardinal directions. In figure 1.2 the region in downtown LA has been reoriented in this manner. Compare this map to the LineDrive map in figure 1.3 in which the orientation of downtown LA is closer to the true orientation.

Chapter 3

Related Work

Over the last two decades cartographers, graphic designers and computer scientists have started to work together to create automated map generalization algorithms. However, despite the fact that the distortion techniques used in hand-drawn route maps improve usability, there has been surprisingly little work on developing automatic generalization techniques based on these distortions. Most of the existing research has focused on developing simplification and abstraction techniques for standard road, geographical and political maps [BM91, Mac95, Imh82, Mon91]. Unlike route maps, these general purpose maps are designed to convey information about an entire region without any particular focus area. Thus, these maps cannot include the specific types of distortion that are used in route maps.

However, route maps can exploit some of the generalization techniques originally designed for standard regional maps. In this chapter we present two threads of related work on shape simplification and automated layout techniques for 2D displays. We consider how simplification and automated layout techniques can be applied to the problem of designing and rendering route maps.

3.1 Shape Simplification

Techniques for curve smoothing, interpolation, and simplification have been well-studied in a variety of contexts including cartography [DP73, VW93, WM98, BLR00]

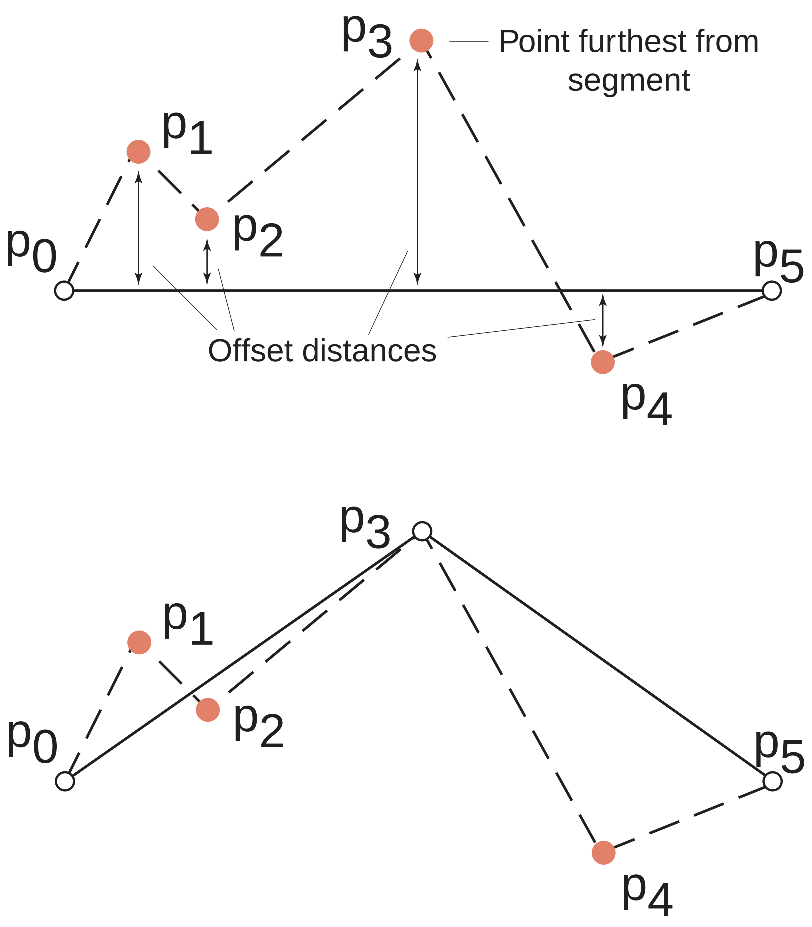


Figure 3.1: Douglas-Peucker algorithm. Initially only the extreme shape points p_0 and p_5 of the original piecewise linear curve (dotted) are retained forming a single segment $\overline{p_0p_5}$. The algorithm computes the offset distance between every interior shape point and the segment and then retains the point furthest from the segment, in this case p_3. The algorithm proceeds recursively on both subsegments (in this case $\overline{p_0p_3}$ and $\overline{p_3p_5}$) until a given error threshold is achieved.

and computer graphics [Ram72, Far88, HS92, dBvKOS97]. Today, the most common simplification technique used in geographic information systems (GIS) and automated mapmaking systems, is the Douglas-Peucker algorithm which was independently developed in the early 1970's, first by Ramer [Ram72] and then by Douglas and Peucker [DP73].

Given a piecewise linear curve, specified as a set of *shape points* connected with line segments, the Douglas-Peucker algorithm begins by throwing away all but the two extreme shape points of the curve and then iteratively attempts to add back only the most important interior shape points. On each iteration the algorithm forms a segment between the extreme shape points, computes the offset distance from each interior shape point to the segment and then adds back the shape point furthest from the segment. The curve is now broken into two segments and the algorithm performs the same test on each sub-segment recursively until an error threshold (e.g. the maximum offset distance is less than some pre-defined tolerance) is achieved. The procedure is illustrated in figure 3.1.

The offset distance is essentially a relevance metric for each shape point, and on each iteration the Douglas-Peucker algorithm retains on the most relevant shape point based on this metric. This approach can be thought of as an *additive* algorithm because it adds shape points back to the curve on each iteration. In contrast, a *subtractive* simplification algorithm begins with all the shape points and on each iteration throws away the least relevant shape points. Both approaches are similar and in fact the Douglas-Peucker algorithm can be easily framed as a subtractive algorithm; start with all the shape points and on each iteration throw away the interior shape point with the smallest offset distance, as long as the offset distance is smaller than the error threshold.

This basic approach to simplification has been applied with a variety of relevance metrics to simplify all types of one-dimensional curves found in maps, including roads, rivers and boundaries between regions such as state lines. One of the main directions of current research in curve simplification is designing relevance metrics that maintain important perceptual characteristics of the curve. Visvalingam and Whyatt [VW93] and Barkowsky et al. [BLR00] have shown that the effective area of a shape point, defined as the area subtended by the point and its two neighbors, is a better metric than offset distance for capturing the overall shape of the curve. This metric tends to eliminate the irrelevant small-scale bends in the curve, while preserving significant large-scale kinks. Wang and Müller [WM98] have designed a metric that finds and evaluates the relevance of sets of adjacent shape points that form a significant bend in the curve. Their elimination operator them removes all points within bends that fall below a given threshold.

One drawback with this class of simplification techniques is that they may introduce false intersection in the simplified curves. Several groups have proposed simplification algorithms that produce topologically consistent simplifications. However, these techniques tend to be complex and computationally expensive[dBvKS98, JBW95]. Recently, Saalfeld [Saa99] described a simple modification to the Douglas-Peucker algorithm for maintaining topological consistency using efficient local tests. These tests compute which shape points to add to the simplified curve in order to guarantee that the simplified curve will not contain false intersections. However this approach relies on properties of the Douglas-Peucker algorithm's offset distance relevance metric and therefore cannot be applied to simplification algorithms that use other types of relevance metrics.

One similarity among all the algorithms we have considered so far is that they only remove unnecessary shape points from the original curve. These algorithms never change the position of a shape point or introduce new shape points. Although a variety of smoothing and simplification algorithms that remove these restrictions have been developed [PBR98, LPL97, PAF96, ZB96, GHMS93, Far88], Guibas et al. [GHMS93] have shown that maintaining topological consistency in this more general case is NP-hard.

As we will describe in chapter 5, we use a relevance metric approach to simplifying roads in LineDrive. Like the previous approaches we only remove shape points from the original curve. However, we introduce several new constraints on the simplification process to avoid the topological and shape errors shown in figure 2.6.

3.2 Automated Layout

We define *layout* as the process of creating an arrangement for a set of visual elements. From a geometric standpoint, we must choose a position, orientation and scale for each element. Planning a layout is a common problem in a variety of domains that involve visual communication, including the design of user interfaces, architecture, web pages, documents, newspapers, magazines, posters, billboards and maps.

Today, the majority of layouts are created by hand, often by an expert graphic designer, and usually require many hours of experimentation to fully develop. At these rates, it is impossible to hand-craft layouts for time-critical applications requiring the communication of visual information. Moreover, novice users without a formal background in graphic design may not know the necessary heuristics to create the most effective layouts. Therefore, developing efficient tools for automatically creating effective, high-quality layouts for graphical presentations has become a major area of research.

In this section we consider previous approaches to automated layout that have influenced the automated layout algorithms used in LineDrive. Our review focuses on systems that design layouts meant to communicate information to human viewers. In particular, we do not cover automated VSLI design techniques [Len90] because the layouts created for VSLI chips are constrained by the peculiarities of a fabrication process and are therefore usually not meant to communicate information to humans. Similarly we do not present packing algorithms that are designed to generate minimum area or volume layouts [MMD97, Hof95, MDL92]. While many of the systems we describe are designed to solve layout problems in domains outside of cartography, we only present systems that are most relevant to automated route map layout. Lok and Feiner [LF01] as well as Hower and Graf [HG96] have compiled comprehensive surveys of research on automated layout techniques for graphical presentations.

3.2.1 Formulating Layout as Constraint-Based Optimization

In almost any layout problem there are restrictions on how the information can be laid out, and there are a set of criteria that can be used to evaluate the quality of the layout. A standard restriction for example is that the final layout can not exceed the dimensions of given display device. Evaluation criteria are often composed of domain specific rules that assess the usability and aesthetics of the layout. In our case, they include issues such as whether roads are large enough to be visible and the desirability of a label's placement relative to the object it is labeling.

Layout constraints can be used to encode both the restrictions on the layout and the evaluation criteria for the layout. In general constraints provide a natural abstraction for specifying the spatial relationships between visual elements. It is usually much simpler to specify a local constraint among a small subset of elements, such as element A should appear above element B, than it is to specify a complete procedure for creating a final layout for all the elements. Constraints allow a designer to describe the layout locally at a high level of abstraction without having to specify exactly how to achieve the final layout. Given a set of constraints, the goal of constraint-based optimization is to find a layout that best meets all of the constraints.

Many general-purpose automated layout systems require the user to specify the spatial constraints on the visual elements. In some cases the constraints are specified as text directives such as `(element A ABOVE element B)` [WW94]. Another approach is to provide visual interfaces for specifying constraints. A number of systems provide graphical user interfaces that allow users to graphically specify the relationships between elements [Gra92]. This approach is particularly common in user interface design toolkits [Mic97, Ous94, MASC85]. The most prevalent example of this form of constraint specification is in page layout systems such as Microsoft Word [RR99] that allow users to anchor images within documents.

While these approaches allow the users to specify a layout at a high level of abstraction, creating an effective layout requires some design knowledge. The user must decide that element A should appear above element B. Recently some researchers have applied machine learning techniques to automatically extract constraints for creating effective layouts. Most of these systems learn by watching expert users interactively specify constraints [Mas94, MMK93, BD86]. Zhou and Ma [ZM00, ZM01] have explored machine learning techniques for extracting these constraints directly from a set of example layouts.

When the layout problem is restricted to a particular domain, expert graphic designers can use domain knowledge to specify the spatial constraints as a pre-defined set of rules. For example, expert cartographers such as Imhof [Imh75], have developed a set of placement constraints for map labels. A common constraint is that labels should appear above the features they are labeling rather than below. Almost all automated map label placement systems [Zor97, ECMS97] define this set of constraints a priori so that novice end-users do not have to design the constraints themselves. Such systems eliminate the layout design work an end user must perform. If the computed layout is not optimal, interactive tools may be provided so the user can guide the system or modify the final result [AAL$^+$00, RMS97] For domain specific applications the fully automated method with predefined constraints produces excellent results and this is the approach we take in LineDrive.

3.2.2 Resolving Constraints

The constraint satisfaction problem appears in a wide variety of forms and the general problem has been well-studied in computer science [Mac92]. Although there are a wide variety of techniques for resolving a system of constraints, automated layout systems usually take one of four approaches; (1) grid-based geometry management, (2) linear programming with constraint propagation, (3) dynamic simulation and (4) search. We consider examples of each of these in turn.

3.2.2.1 Grid-Based Geometry Management

Axis-aligned grids are a common tool for organizing the layout of two-dimensional presentations such as newspapers. Several automated layout systems simplify constraint resolution by only allowing the specification of constraints that conform to an underlying grid. Each visual element is treated as content for a grid cell and constraints are specified relative to other grid cells. Positional constraints are used to specify the position of one cell with respect to another, while hierarchical constraints are used to specify that one cell should appear within another cell. Initially the entire page forms a single cell at the base of the hierarchy and all the other cells are placed within it.

Given a set of constraints specified in this grid-based manner, a geometry manager then expands the constraints from the top down to determine the absolute position of each grid cell. Such systems often rely on the user to solve any inconsistencies among the constraints. Feiner's GRIDS [Fei88] system was one of the first to use this approach to automating layout. Today this grid-based approach is especially popular among user interface toolkits including Tk [Ous94], the X toolkit [MASC85], and Microsoft's Foundation Classes [Mic97].

3.2.2.2 Linear Programming With Constraint Propagation

The widespread interest in the constraint satisfaction problem has led to the development of general-purpose systems for resolving systems of linear constraints [BB98, SMFBB93]. Each constraint must be specified as either a linear equation or an inequality, and each constraint equation is given a priority so that higher priority constraints are resolved in favor of lower priority constraints if there is an inconsistency in the constraint system. The solvers then apply linear programming in combination with constraint propagation in order to find the best solution to the system.

In order to use a general-purpose constraint solver to design a graphical presentation, higher-level constraints must be translated into a set of linear equations and inequalities. Weitzman and Wittenburg [WW94] have developed a full relational grammar for specifying the high-level constraints necessary to generate multimedia presentations. Users enter a set of text rules using the grammar and their system converts these into a set of low-level constraint equations which are fed into a back-end constraint solver that generates the final layout. Graf's LayLab [Gra92] system performs a similar transformation of high-level constraints into low-level constraint equations. However, unlike Weitzman and Wittenburg's system, LayLab allows users to visually specify the high-level constraints though a graphical user interface.

3.2.2.3 Dynamic Simulation

Another technique for resolving layout constraints is to treat the visual elements as masses and specify the constraints as forces acting on the masses. Often the forces are described using the standard physics model of spring in which the force is proportional to the distance between the element center of mass and its desired position. Once a mass-spring system has been defined in this manner, standard physically-based constrained dynamics simulators [WGW90] can be used to find the rest positions, and hence the layout for the visual elements

This approach has been widely explored in the domain of graph drawing [BETT99, FR91]. Its popularity may lie in the fact that it is very intuitive to think of each node in the graph as a mass and each edge between nodes as a spring that links them. The nodes are placed in an initial position and the system is relaxed until it converges to an energy-minimizing state. One drawback of this approach is that minor tweaking of layout parameters can result in major changes in the final layout.

In computer graphics, Gleicher and Witkin [GW94] have applied constrained dynamic simulation to two-dimensional drawing programs. Their system can simultaneously enforce layout constraints while allowing the user to drag, thereby allowing the user to interactively explore the configurations of the model consistent with the layout constraints. More recently, Harada et al. [HWB95] have used constrained dynamics to provide an interactive user interface for an architectural layout system.

The text layout engine used in TeX [Knu99] builds a representation similar to a mass-spring system. Each word within a paragraph represents a mass and the spaces between them represent springs. The goal of the layout engine is to determine the spacing between the words so that the line-breaks in the formatted text appear aesthetically pleasing. Although the problem is set up to resemble a mass-spring system, the TeX layout engine uses dynamic programming techniques rather than physically-based dynamic simulation.

3.2.2.4 Search

Many layout problems can be posed as a search for an optimal layout over a space of possible layouts. To frame the layout problem as a search we need to define an initial layout and two functions: a *score* function that assesses the quality of a layout based on the evaluation criteria, and a *perturb* function that manipulates a given layout to produce a new layout within the search space. Both the score and the perturb functions are defined by the set of constraints on the layout. Given these two functions the search can be performed using any search technique including A*, tabu search, gradient descent and simulated annealing [MF00].

In the graph drawing domain a variety of search techniques have been explored. Kamada and Kawai [KK89] have used gradient descent while Davidson and Harel [DH96] exploit simulated annealing. Brandenburg et al. [BHR95] and Tunkelang [Tun93] describe more complicated algorithms that combine several search techniques. In the domain of cartography, search techniques are widely used to automatically label point and line features in traditional geographic maps. Marks and Shieber [MS91] have shown that finding optimal label placements is NP-complete and several previous systems have used randomized search to find near-optimal label placements [Zor97, ECMS97]. Simulated annealing is the most commonly used randomized search algorithm because it efficiently covers the search space and is simple to implement. For these reasons, we also use simulated annealing, not only for placing labels, but also for laying out the roads and context information in LineDrive route maps. The simulated annealing algorithm is described by the following pseudo-code:

```
procedure SimAnneal()
1  InitializeLayout()
2  $E \leftarrow$ ScoreLayout()
3  while(! termination condition)
4    PerturbLayout()
5    $newE \leftarrow$ ScoreLayout()
6    if (($newE > E$) and (Random() $< (1.0 - e^{-\Delta E/T})$))
7      RevertLayout()
9    else
10     $E \leftarrow newE$
11   Decrease($T$)
```

Implementing the algorithm is straightforward and requires the specification of four functions. The InitializeLayout() function defines the initial placement for each of the visual elements and thereby provides a starting point for the search. The PerturbLayout() function provides a method for changing a given layout into a new layout, while the RevertLayout() functions inverts the actions of PerturbLayout() to go from the new layout back to the previous layout. Finally the ScoreLayout() function computes how close to optimal the current layout is. By convention, scores are defined to always be positive and the lower the score the better the layout. Therefore, the goal of simulated annealing is to minimize the score.

As shown in the pseudo-code, the simulated annealing algorithm accepts all good moves within the search space and, with a probability that is an exponential function of a temperature T, accepts some bad moves as well. As the algorithm progresses, T is annealed (or decreased), resulting in a decreasing probability of accepting bad moves. Accepting bad moves in this manner allows the algorithm to escape local minima in the score function.

We can divide our constraints into two sets; 1) *hard constraints* consist of characteristics required of any acceptable layout and therefore bound the space of possible layouts, while 2) *soft constraints* consist of characteristics desired in the final layout but not required. In designing the constraints, it is important not to impose too many hard constraints or the layout problem will be overconstrained making it impossible to find any acceptable layout. The hard constraints are typically imposed through the perturb function which is designed to only generate layouts that meet the hard constraints. The score function checks how well a given layout achieves the soft constraints.

The difficult aspects of characterizing the layout problem as a search are designing a numerical score function that efficiently captures all of the desirable features of the optimal layout and defining a perturb function that covers a significant portion of the search space. In the following chapters as we discuss the different layout stages of LineDrive, we will focus on explaining these aspects of our algorithm design.

Chapter 4

System Overview

Given a route, the goal of a route map design system is to produce an image that visually emphasizes the most important information required to follow the route. LineDrive is a fully automated, real-time, route map design system that achieves these goals by exploiting the generalization techniques commonly found in hand-drawn maps.

From a usability perspective, however, even the best route map design system would be difficult to use without a front-end route finding service that can take an origin-destination address pair and compute a route between them. Such a route finding service requires direct access to a comprehensive geographic database of all the roads within a region. Lacking such access we originally developed a prototype for LineDrive as a stand-alone application, that took a human-readable text description of the route as input as shown in this example:

1 Starting at 652 Folsom St, begin on FOLSOM ST heading northeast
2 Turn right on HAWTHORNE ST heading southeast for 0.1 miles
3 Turn right on HARRISON ST heading southwest for 0.8 miles
4 Turn right on 7TH ST heading northwest for 0.5 miles
5 Turn left on MARKET ST heading southwest to 2059 Market St

Each line described a turn direction, the name of the road being turned onto the heading of the road (N, S, E, W, NE, NW, SE, SW) and the distance of the road in

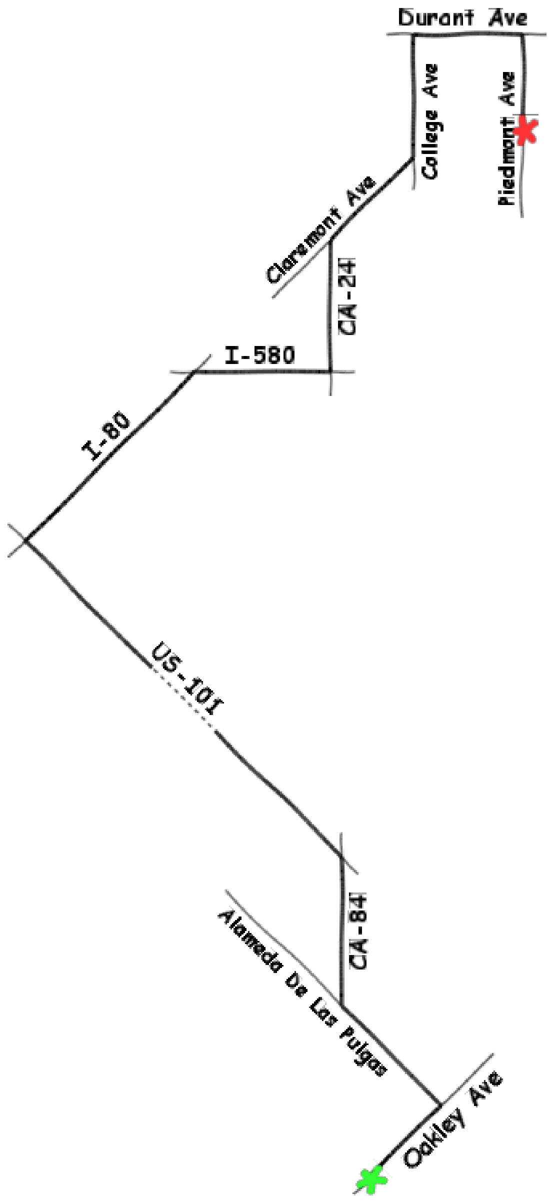


Figure 4.1: An early prototype LineDrive map. The initial prototype version of LineDrive took a human readable text description of the route as input and produced this type of route map containing many of the generalizations found in hand-drawn maps. Note that this is the same route as depicted in figure 2.5. While these maps were much more effective than standard computer-generated route maps, the lack of road shape information could cause serious topological errors.

miles. The user could either hand-specify specify the route or use the text directions produced by a standard Web-based driving direction service. A map produced by the prototype LineDrive system is shown in figure 4.1. Note that this is a map for the same route depicted in figure 2.5.

Even with such a simple input description of the route, the prototype LineDrive system could produce maps that were much more effective than those created by previous automated mapping systems. We believe this is largely because these simplified instructions, which indicate only heading and distance along each road, capture the topology of the route. As Byrne [Byr82] has shown, in many hand-drawn maps, roads are simply drawn as straight lines in one of the eight cardinal directions.

However, the lack of shape information about roads caused two types of errors with these early LineDrive maps. First, the turn direction would sometimes conflict with the road heading. Second, the lack of shape information along a road could generate false or missing intersections that did not occur in the original route. Since the prototype system did not have access to the original road shape it could not alleviate such errors. Moreover, entering text directions into the LineDrive system could be a tedious process. Users generally create maps for routes they are not familiar with and even when they have some knowledge of the route they usually do not know distances for each road segment.

For these reasons, the current version of LineDrive is embedded within a complete end-to-end route mapping system as part of Vicinity Corporation's MapBlast! website[1]. In this section we present the overall system architecture. We begin with an overview description of the end-to-end system, followed by a more detailed description of the computational stages in LineDrive.

4.1 End-To-End Route Mapping System

A block diagram of the end-to-end route mapping system is shown in figure 4.2. All of the geographic data is stored in the database in the standard latitude/longitude geographical coordinate system. The system takes an origin-destination address pair

[1]Located at www.mapblast.com

as input and the route finding service computes the sequence of roads required to go from the given origin to the given destination. Each road is represented as a piecewise linear curve described by a sequence of latitude/longitude shape points.

Once the route is computed it is passed into a image size oracle which determines the image size for the route map. The size is based on an estimate of the aspect ratio of the route and the size of the output display device. While LineDrive can design a route map to fit within any given image size[2], the system can make better use of the given space in the image if it conforms to the aspect ratio of the route. For example, routes that are predominantly north-south might be given more vertical space than horizontal space to accommodate the vertical orientation. However, the image size oracle is not a central component of the route visualization system. It is further described in chapter 9, after we present the core LineDrive system.

The route and image size are passed into LineDrive which generates a route map image. The page designer takes this route map as well as several other elements describing the route and constructs a final webpage that is delivered back to the user. This final step is crucial to proving an effective solution. Users often want to see several aspects of the route at once. In particular a multimodal description of the route with text directions near the route map image can vastly improve usability. As described in chapter 9, the page designer places the both the LineDrive route map and the text directions well as a zoomed-out overview map and a zoomed-in destination map in the final webpage. Examples of webpages created by our end-to-end system appear in figure 9.2.

4.2 LineDrive Map Design Stages

The space of all possible route map designs and layouts is extremely large and contains many dimensions. LineDrive reduces the dimensionality of this space by performing the map design in five independent stages as shown in the gray shaded LineDrive portion of the block diagram in figure 4.2. The layout stages implement many of the generalization techniques found in hand-drawn maps. These stages were originally

[2]However, at small image sizes LineDrive may not produce a very legible map.

developed as part of the prototype LineDrive system and evolved to exploit additional geographic information in the current version.

The first stage of the LineDrive system is shape simplification, which removes extraneous shape detail from the roads. The next three stages, road layout, label layout, and context layout, each deal with automating a layout problem. We use simulated annealing, a randomized search, in all three stages to efficiently find near-optimal layouts. The basic simulated annealing algorithm is described in section 3.2.2.4 of the previous chapter.

The road layout stage computes a length and orientation for each road, which ensures that all the roads can be clearly labeled and that each turning point is visible and well-emphasized. The label layout stage labels each road with its name so that navigators can quickly identify each turn. The context layout stage then attempts to add additional information, such as cross-streets and local landmarks, to the map. Since the roads and their labels constitute the essential information in route map, the additional context information is only added sparingly around turning points in order to avoid adding clutter to the map. The details of these three layout stages are presented in chapters 6, 7 and the first half of chapter 8. The second half of chapter 8 presents the decoration stage, which adds elements such as road extensions and an orientation arrow to the map to enhance its overall usability.

Figure 4.3 depicts a route rendered using the current version of LineDrive The evolution of the system is evident in a number of graphical differences between this map and the map rendered with the early version of LineDrive shown in figure 4.1. For example, the prototype map does not contain distance labels, cross-streets, road-type indicators, bullets at turning points, or even an orientation arrow. Moreover, access to the geographic database allows the system to depict more road shape and truer turning angles. While the initial version of LineDrive produced usable route maps and provided a good test-bed for trying various diagrammatic generalization techniques, the current version vastly improves usability by providing an end-to-end solution, depicting several secondary classes of context information and delivering increased robustness to topological errors.

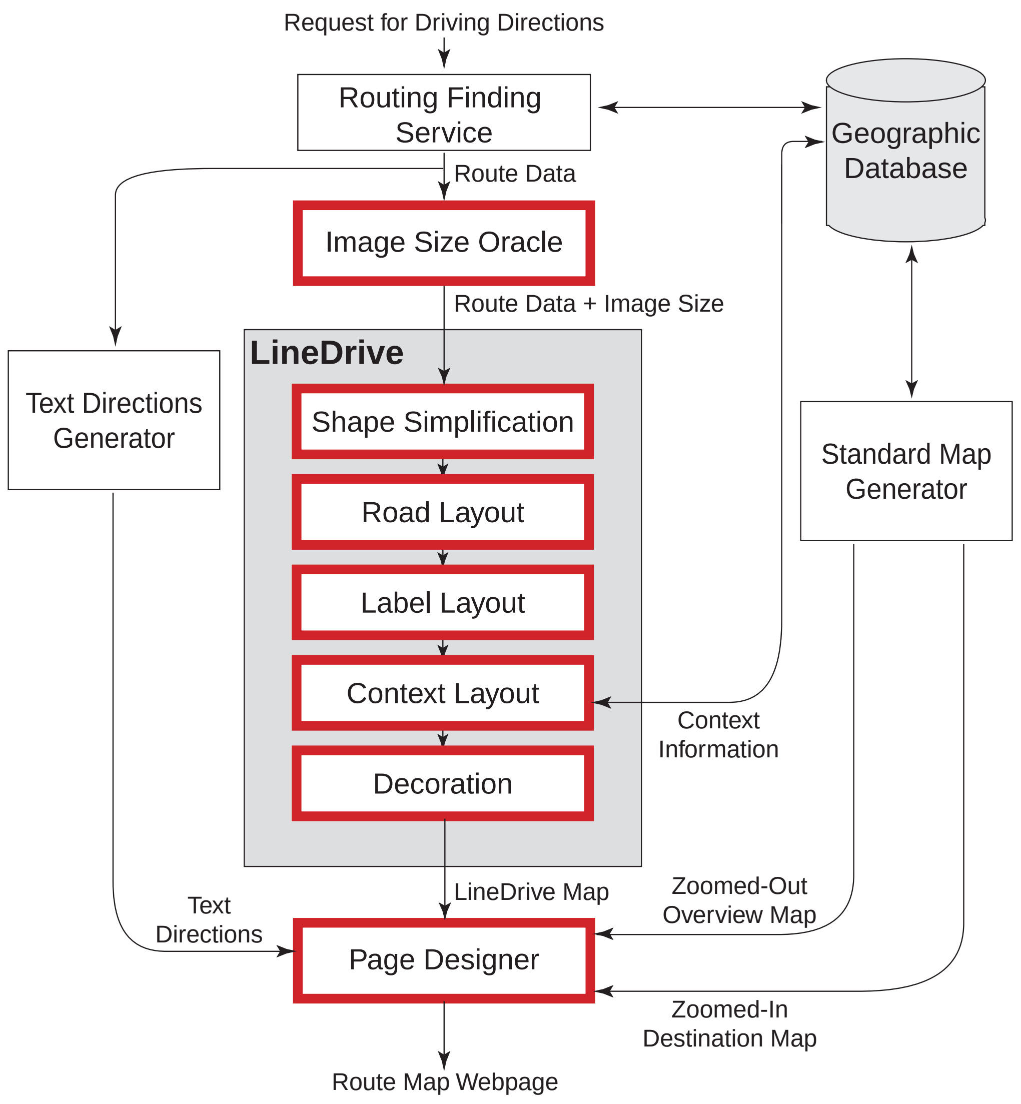


Figure 4.2: Block diagram of end-to-end route mapping system. The user requests driving directions by specifying an origin-destination pair of addresses and the system produces a webpage containing four elements; (1) a LineDrive map, (2) text directions, (3) a zoomed-out overview map and a (4) zoomed-in destination map. Each box represents a computational stage in the system and each link represents data flow through the system. The LineDrive route map designer is itself comprised of five independent stages. In this dissertation we will describe the functionality of each of the stages marked with thick red outlines. Examples of webpages created by the complete system are shown in figure 9.2.

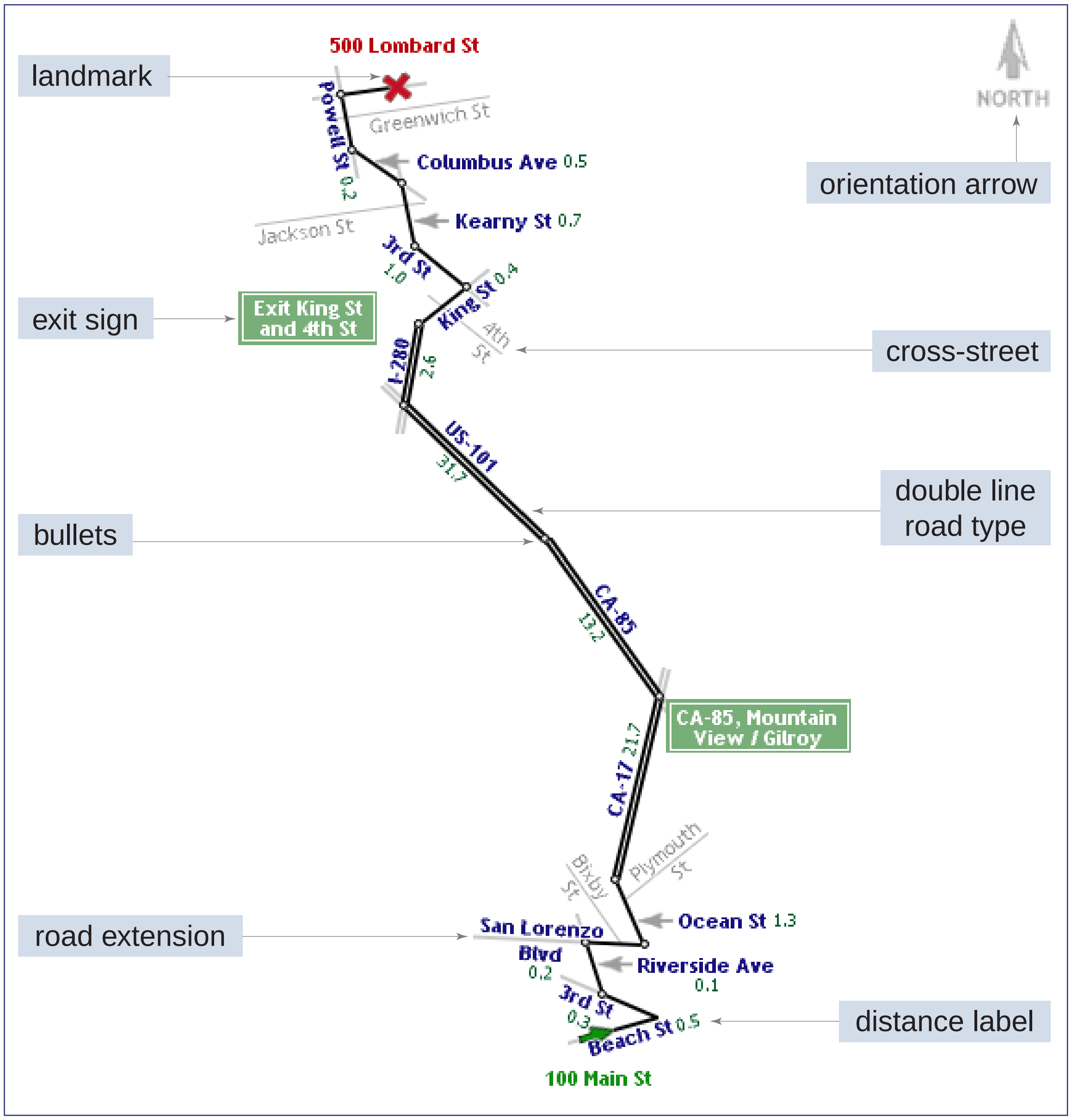


Figure 4.3: Current LineDrive map. The current version of LineDrive is embedded within an end-to-end route mapping system. Access to a comprehensive geographic database allows the system to depict more road shape and truer turning angle. Comparison with a map generated by the LineDrive prototype shown in figure 4.1 shows that the maps have graphically evolved in a number of ways.

Chapter 5

Shape Simplification

Although each road on the route is initially specified as a detailed, piecewise linear curve, navigators rarely need to know the exact shape of a road in order to follow it. LineDrive's shape simplification stage reduces the number of segments in each road to smooth out unnecessary wiggles, while leaving the overall shape of the route intact. Shape simplification not only yields a cleaner looking map but also increases the speed and memory efficiency of the subsequent layout stages of the LineDrive system.

LineDrive employs a standard simplification approach that ranks the relevance of all the shape points of the curve and then removes all interior shape points that fall below a given threshold. Figure 5.1 shows the same route before and after simplification. The high-frequency curve detail in roads such as CA-17 is unnecessary for following the route. Even relatively smooth, low-frequency curves such as the curve along US-101 does not add essential information to the map. In fact, for this route all the roads have been simplified to straight lines. Figure 5.3 shows an example of a route in which some shape information is retained after simplification to better capture the roughly 90.0 degree turn from Nita Avenue onto Betlo Avenue.

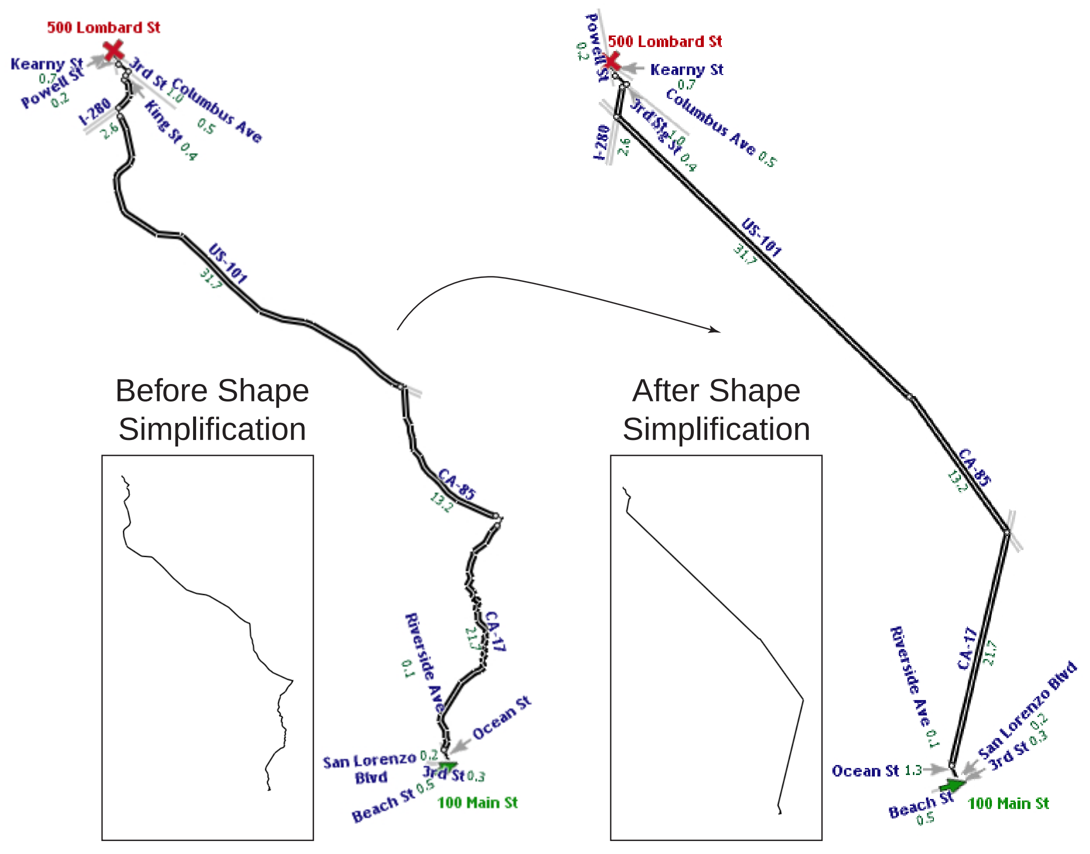


Figure 5.1: Before and after shape simplification. A route rendered directly from its input form, before simplification and after simplification. The simplification removes unnecessary shape detail and in this route all the roads become straight lines. Simplification also makes it easier to label roads so that the association between each road and its label is visually clear. Note that both maps are rendered using a constant scale factor, which causes relatively short roads at the beginning and end of the route to effectively shrink to a point. The insets depict the route map immediately before and after the shape simplification stage of the LineDrive pipeline. To show how shape simplification affects labeling we have processed the route through the all the stages of the LineDrive pipeline following road layout in the non-inset maps.

5.1 Preventing Errors

While simplifying road shape we also have the additional requirement that the algorithm must not introduce the three types of undesirable effects shown in the rightmost column of figure 2.6: false intersections, missing intersections and inconsistent turn directions. We include three tests during simplification to prevent these errors.

5.1.1 Missing Intersection Test

To ensure that the simplification process does not introduce false or missing intersections, we initially compute all the intersection points between each pair of roads. Suppose roads r_1 and r_2 initially intersect at point p_1. We add the intersection point p_1 to the set of shape points for both r_1 and r_2 and mark p_1 as *unremovable*. Since the simplification algorithm cannot remove these unremovable intersection points, a missing intersection cannot be generated.

5.1.2 False Intersection Test

We also create a separate list of the true intersection points between each pair of roads. In the subsequent phases of simplification we only accept the removal of a shape point if its removal does not create a new intersection point between any pair of roads, thus ensuring that the simplification will not introduce any false intersections.

5.1.3 Inconsistent Turn Direction Test

Finally, we check for inconsistent turn direction at the turning points between each road r_i and the roads r_{i-1} and r_{i+1} adjacent to it. First, we check the turn direction between the current road r_i and the previous road r_{i-1}. Our test detects inconsistent turn direction with respect to the coordinate system oriented along the last segment of r_{i-1}. As shown in figure 5.2, we loop through the set of shape points for r_i from start to end and determine whether or not to mark each one unremovable based on the heading directions of two vectors; vector v_1 running between the endpoint of r_{i-1} and the current shape point, and vector v_2 running between the current shape point and the endpoint of r_i. If we let $c_{r_{i-1}}$ be the vector oriented along the last segment of r_{i-1} then we can test whether or not v_1 and v_2 are in the same half-plane with

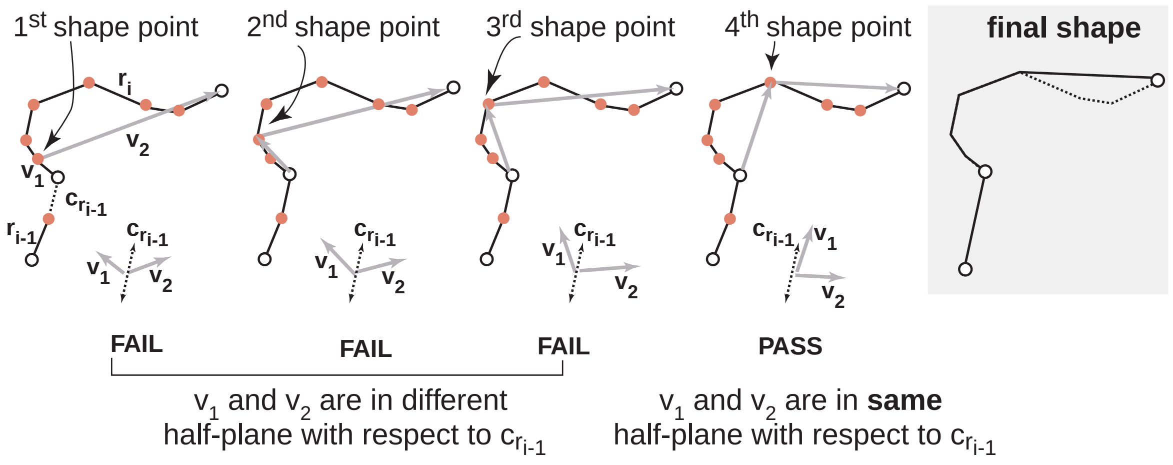

1st shape point
2nd shape point
3rd shape point
4th shape point
final shape
r_i
v_1
v_2
c_{r_{i-1}}
r_{i-1}
FAIL
FAIL
FAIL
PASS
v_1 and v_2 are in different half-plane with respect to c_{r_{i-1}}
v_1 and v_2 are in same half-plane with respect to c_{r_{i-1}}

Figure 5.2: Inconsistent turn direction test. To check turn direction consistency between roads r_i and r_{i-1} we step through the shape points of r_i, forming two vectors v_1, between the endpoint of r_{i-1} and the current shape point, and v_2, between the current shape point and the endpoint of r_i. If v_1 and v_2 are not in the same half-plane with respect to the coordinate system oriented along the last segment of the r_{i-1}, we mark the current shape point as unremovable. The test continues until a shape point is not marked as unremovable.

respect to $c_{r_{i-1}}$ by checking the following equality:

$$sign(c_{r_{i-1}} \times v_1) = sign(c_{r_{i-1}} \times v_2) \tag{5.1}$$

Since all of our vectors lie in the xy-plane, the vector cross products will point along either the positive or negative z-axis. If the two cross products have different signs[1] v_1 and v_2 are in different half-planes with respect to $c_{r_{i-1}}$ and therefore the current shape point must be marked unremovable. The test continues until a shape point is not marked as unremovable. While we have described the turn direction test for the turning point between roads r_i and r_{i-1}, the test for the turning point between r_i and the next road r_{i+1} is similar.

Even if turn direction consistency is properly maintained the map can be confusing if a sharp turning angle appears as if it is a shallow turning angle. An example of

[1]Our equation contains a slight abuse of notation. We are passing a vector to the $sign()$ function instead of a scalar. The function returns the sign of the z-component of the vector.

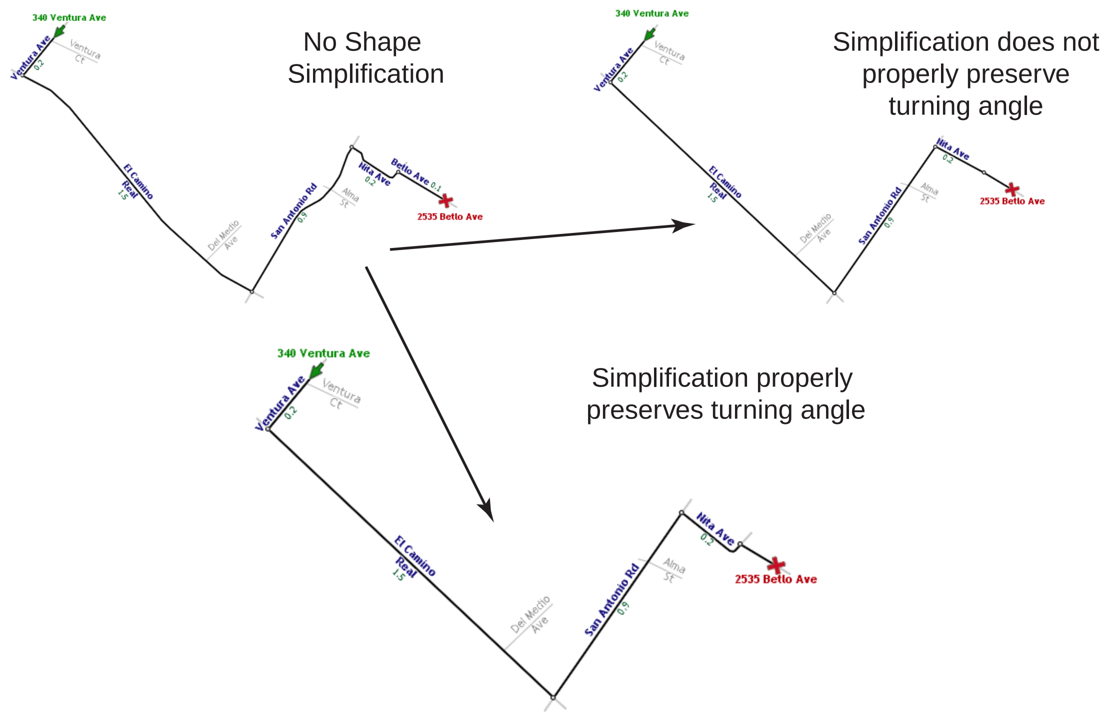


Figure 5.3: Preserving sharp turning angles. The turning angle at the intersection between Nita Avenue and Betlo Avenue is about 90.0 degrees in the unsimplified map. The inconsistent turn direction test described in figure 5.2 performed at the turning point between Nita and Betlo allows Nita to be simplified to a straight line. As shown in the map at the top right, the turn angle after simplification is just about 0.0 degree even though the the turn direction is consistent. To avoid such drastic changes in turn angle we extend the previous inconsistent turn direction test to mark shape points as unremovable if the angle between v_1 and v_2 is larger than a given threshold angle.

this problem appears in figure 5.3. In the unsimplified route the angle between Nita Avenue and Betlo Avenue is roughly 90.0 degrees. If Nita is simplified to a straight line, the turn direction between Nita and Betlo remains consistent but the turn angle reduces to just about 0.0 degrees. Instead of a turning onto Betlo it appears as if Nita becomes Betlo. To avoid this problem we extend the inconsistent turn direction test to mark shape points as unremovable if the angle between v_1 and v_2 is larger than a given threshold angle. In practice we have found that a threshold angle of

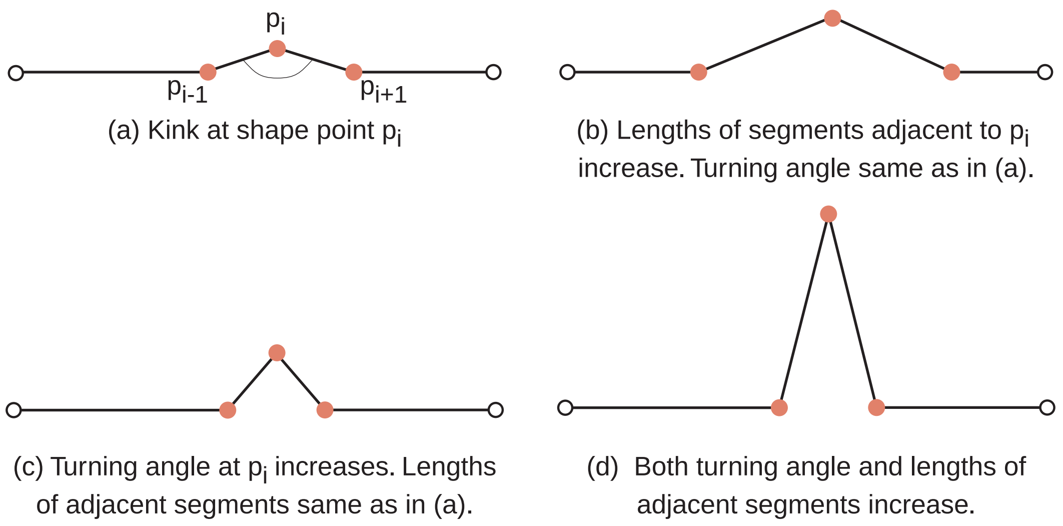

p_i
p_{i-1}
p_{i+1}
(a) Kink at shape point p_i
(b) Lengths of segments adjacent to p_i increase. Turning angle same as in (a).
(c) Turning angle at p_i increases. Lengths of adjacent segments same as in (a).
(d) Both turning angle and lengths of adjacent segments increase.

Figure 5.4: Shape point relevance. The goal of shape simplification is to remove irrelevant kinks in a road by removing shape points causing the kinks. Therefore, the relevance of a shape point is dependent on the turning angle at the shape point and the lengths of the segments adjacent to the shape point. Increasing the segment lengths as in (b) or the turning angle as in (c) increases the relevance of the shape point. The shape point in (d) is most significant because both the turning angle and the adjacent segment lengths increase. *(After figure 2 in Barkowsky et al.[BLR00])*

65.0 degrees produces good results.

5.2 Relevance Metric

For most roads we are very aggressive about simplification. Our relevance metric allows the removal of all shape points that are not marked as unremovable by the previous tests. As a result, most roads are simplified to a single line segment. Depicting roads as simple line segments helps to create the clean, easy-to-read look of LineDrive maps.

For some roads, such as highway entrance and exit ramps, depicting more realistic shape can be useful. Knowing whether a ramp curves around tightly to form a cloverleaf or only bends slightly can make it easier to enter or exit the highway. Thus, when simplifying ramps we use a more conservative simplification relevance metric to retain more shape. Our relevance metric is based on a metric described by Barkowsky et al. [BLR00]. As shown in figure 5.4, the relevance of a shape point is dependent on the turning angle at the shape point and the lengths of the segments adjacent to the shape point. In general, sharp turning angles and long segments, increase the importance of the shape point.

Mathematically we can define the relevance metric as follows. Suppose v_1 and v_2 are vectors running from shape point p_i to its neighboring shape points p_{i-1} and p_{i+1} respectively. Let $\beta(p_i)$ be the turning angle at p_i between v_1 and v_2. The relevance metric K is then computed as:

$$K(p_i) = \frac{\beta(p_i)\|v_1\|\|v_2\|}{\|v_1\| + \|v_2\|} \tag{5.2}$$

$$v_1 = p_{i-1} - p_i \tag{5.3}$$

$$v_2 = p_{i+1} - p_i \tag{5.4}$$

Higher values of K indicate more important shape points. To simplify a piecewise linear curve using this metric we initially compute relevance for all interior shape points. We then remove the least relevant point, update the relevance values for the shape points adjacent to the removed point and iterate.

5.3 Dropping Roads from the Map

Highway ramps and traffic circles are two classes of roads that are usually very short in comparison to other roads on the route. Some long routes between distant cities require traversing many highways and therefore the route will contain many entrance and exit ramps. Similarly in Europe, a main thoroughfare may pass through many traffic circles. In fact, for some roads in Europe, passing through each major intersection requires traversing a traffic circle.

When a route contains many such short roads it can be difficult to show all the roads on the route at the appropriate level of detail. If a non-constant scale factor is used to grow short roads, the ramps and traffic circles grow much larger than the other roads and all the roads end up about the same size. Moreover, depicting all the

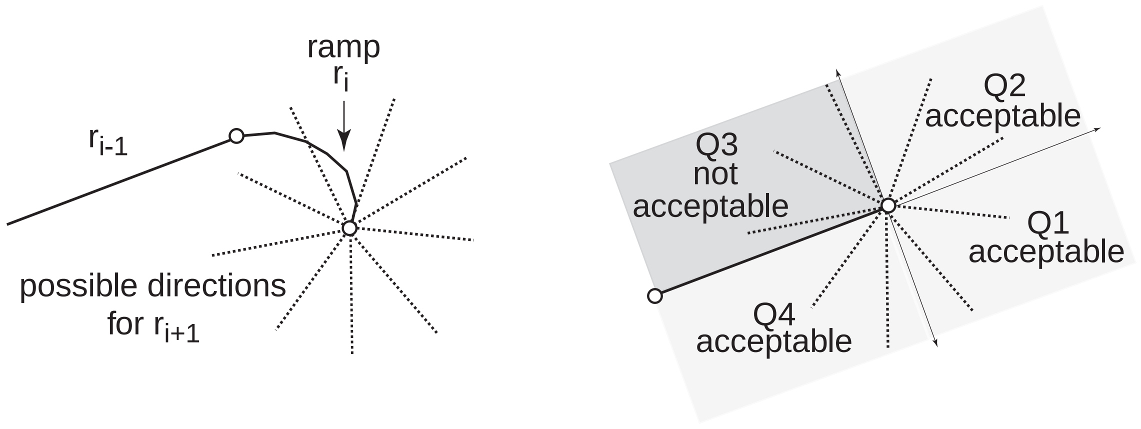


Figure 5.5: Inconsistent turn direction test when dropping ramps. The ramp r_i can only be removed if the next road r_{i+1} does not head back towards the previous road r_{i-1}. Here, the ramp turns to the right of r_{i-i}. The dotted lines represent possible directions for the road r_{i+1} following the ramp. In this case we do not drop the ramp if r_{i+1} lies in the third quadrant of the coordinate system oriented along the last segment of r_{i-1}.

short ramps and traffic circles can clutter the map with unnecessary detail. In this section we describe strategies for dropping such short roads from the map entirely. Dropping these roads makes it easier to design the map layout, and adds to the overall simplicity of the map thereby increasing usability as well.

5.3.1 Dropping Ramps

While highway entrance and exit ramps can help navigators understand how to enter or exit the highway, it is usually possible to follow the map even if they are not depicted. On long routes the ramps tend to clutter the map and therefore if the route is longer than a given threshold we remove all ramps from the map that can be removed without creating a false or missing intersection or inconsistent turn direction. The threshold is based both on the total mileage of the route and the number of turns in the route.

As shown in figure 5.5, the test for checking turn angle consistency is slightly different when dropping ramps than when simplifying road shape. If the road r_{i+1}

immediately after a ramp r_i, heads towards the road r_{i-1} preceding the ramp, the ramp cannot be dropped from the map. In such cases dropping the ramp would cause r_{i+1} to appear on the "wrong" side of r_{i-1}.

To determine the length threshold for dropping ramps we generated a set of 25 routes with lengths varying from a couple of miles to several thousand miles. The number of turns varied from two turns to 27 turns. We created two LineDrive maps for each of the routes, one maintaining all the ramps and the other dropping as many ramps as possible without generating topological errors or inconsistent turn directions. We then asked a small group of users to look at corresponding pairs of maps and tell us whether or not the ramps cluttered the maps containing them. Through this informal experiment we found that it is usually best to drop ramps for any route longer than 30.0 miles or contains more than 11 steps.

5.3.2 Dropping Traffic Circles

Unlike highway ramps we always drop all traffic circles from the route regardless of the length of the route. We have found that traffic circles have a consistent topology in which a set of roads all meet at the circle itself, like the spokes on a wagon wheel. Therefore dropping the circle from the route simply forces the roads before and after the circle to meet at a point rather than going around the circle. Topological and shape errors are unlikely. In fact we have yet to encounter a case in which dropping a traffic circle causes a false intersection, a missing intersection or an inconsistent turn direction. Therefore we do not require these tests when dropping traffic circles.

However, completely eliminating any representation of a traffic circle from a map can be misleading. To avoid such confusion we replace the original traffic circle road r_i, with a fixed-size circle icon at the intersection between the roads r_{i-1} and r_{i+1}. An example of this simplification is shown in figure 5.6. We describe how such icons are along the route in chapter 8 on context layout. In this way, even though the traffic circle is no longer a separate road and does not need to be laid out in the subsequent road layout stage, a visual representation for the traffic circle appears in the final LineDrive map.

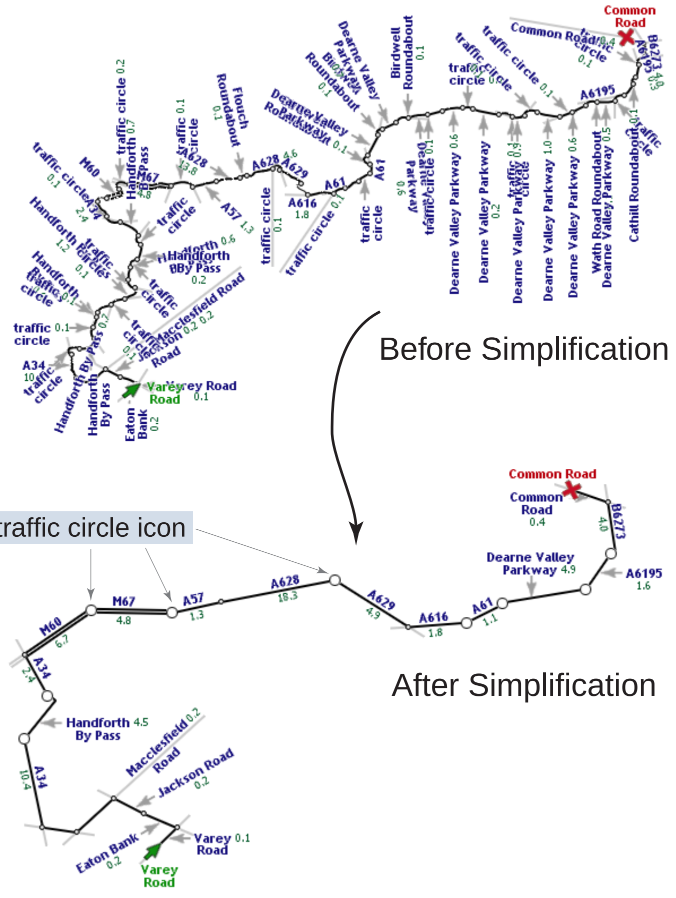


Figure 5.6: Dropping traffic circles. This European route contains many traffic circles as shown in the unsimplified map. After we drop all the traffic circle road, and then consolidate adjacent roads with the same name the map becomes much simpler. We use traffic circle icons to represent any remaining traffic circles between roads with different names.

5.3.3 Consolidating Roads

Once we have dropped ramps and traffic circles we check whether adjacent pairs of roads have the same name and merge these into a single road. In such cases it is likely that the route simply passed through a ramp or a traffic circle while continuing on the same road. For example, as shown in the unsimplified map in figure 5.6 a short section of Dearne Valley Parkway, contains a traffic circle at every major intersection. Taking

Dearne Valley Parkway for a five mile requires passing through five different traffic circles. While shape simplification replaces each traffic circle road with a circular icon, without road consolidation each part of Dearne Valley Parkway would appear as a separate segment with its own label. Consolidating adjacent roads with the same name reduces clutter and ensures that road which people usually think of as a single unit appear as a single segment in the LineDrive map.

If a traffic circle icon appears between two roads that should be consolidated we simply remove the icon from the map. Like many handcrafted route maps we only show a traffic circle icon when the navigator must turn from one road onto another road after passing through the circle. If the navigator does not turn onto a new road at the traffic circle we do not show it. While this might seem misleading, we have found that the reductions in clutter far outweigh the cognitive errors generated by inaccurately depicting such traffic circles.

Chapter 6

Road Layout

The goal of road layout stage of LineDrive is to determine a length and an orientation for each road such that all roads are visible and the entire map image fits within a pre-specified image size. Moreover, the layout must avoid the problems shown in the second and third columns of figure 2.6 and preserve the topology and overall shape of the route. In particular, longer roads should appear longer than shorter roads and the layout should not introduce false or missing intersections.

Our approach is to specify a set of constraints that define the properties we would like in the final road layout. We then use simulated annealing to find a layout for the roads that best realizes our constraints. Figure 6.1 shows a route before and after the road layout stage.

As described in section 3.2.2.4 to apply the simulated annealing search algorithm we must specify an initial road layout, an invertible function for perturbing a layout and a function for scoring a layout. We describe each of these components of the road layout search in sections 6.1- 6.3. In section 6.4 we describe how we set the simulated annealing search termination parameter and cooling function to achieve efficient running times while finding near-optimal road layouts. We conclude this chapter in section 6.5, by describing the final phase of road layout which deterministically processes each road to fine-tune their orientations.

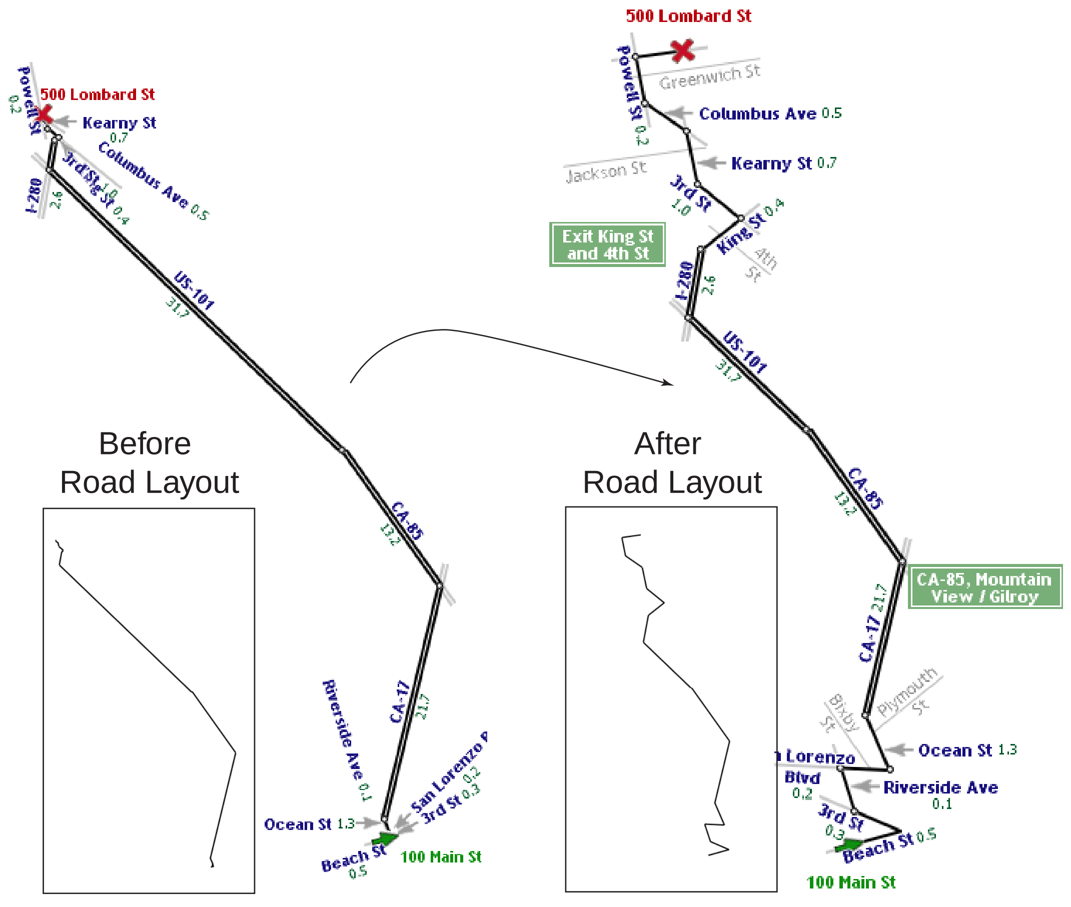


Figure 6.1: Before and after road layout. The road layout stage of LineDrive is responsible for determining the length and orientation of each road such that all the roads are visible and no topological or shape errors are introduced into the map. Before road layout many roads and turning points especially near the origin and destination of the route are difficult to see. After road layout this is no longer the case and there is enough space to easily label each road as well as provide some context information in the form of cross-streets. The insets depict the route map immediately before and after the road layout stage of the LineDrive pipeline. To show how road layout affects labeling and context layout we have processed the route through all the stages of the LineDrive pipeline in the non-inset maps.

6.1 Initial Layout

To generate an initial layout for the search, we first build an axis-aligned bounding box for the original route and compute a single factor to scale the entire route to fit within the given image viewport. Such a uniformly scaled route will usually contain roads that are too small to see or label. While this uniformly scaled route layout could be used as a starting point for the search, we have found that it is not ideal. Like most search techniques, simulated annealing performs better the closer the initial solution is to the optimal solution. A layout containing roads that are too small to see is far from our optimal solution and therefore we modify the uniformly scaled route to push our initial layout closer to the optimal layout.

Once we have constructed the uniformly scaled layout, we find all the roads that are shorter than a predefined minimum pixel length, L_{min}, and grow them to be L_{min} pixels long. Since we initially scaled all the roads to fit exactly within the bounds of the image, growing the short roads may extend the map outside the viewport. We finish the initial layout phase by again scaling the entire route to fit within the image viewport.

The initial layout created in this manner may contain topological errors. In most cases the errors will be resolved during the road layout search. However, it may be argued that since an optimal layout will not contain topological errors, an initial layout containing topological errors is no closer to the optimal layout than a uniformly scaled layout containing extremely short roads.

Based on this argument, we have experimented with an alternative approach for generating an initial layout which attempts to both grow all short roads so they are visible but also ensure that topological errors are not introduced. Given a subset of roads on the route, we can maintain the topology between them by growing all of them by the same factor. In particular if two roads r_i and r_{i+m}, intersect in the original route, we consider the set of roads between them $[r_i, r_{i+1}, ..., r_{i+m}]$ as forming an *intersection interval*. The idea behind our second approach is to partition the route into subsets of such intersection intervals that must be grown together to maintain the route topology. The details of this topological error-free initial layout algorithm are presented in appendix A.

The main advantage of the topological error-free approach is that it may provide a better starting point for the subsequent search phase of road layout. The simulated annealing algorithm is more likely to converge to a solution that does not contain topological errors if it starts from an error-free initial layout. However, the drawback of this error-free approach is that it requires entire groups of roads to be scaled by the same factor. Therefore, the initial layout may contain roads that are far too short to be visible. This problem usually occurs if roads within an intersection interval have vastly different lengths. For example, a set of short residential roads may circle around to pass under a long highway. Growing all of these roads by the same factor while fitting them within the given viewport will ensure that the residential roads are not initially visible. Instead of searching for a layout that resolves topological errors, this approach often requires searching for a solution that will grow short roads. In practice we have found that this approach performs marginally better than the simple initial layout scheme which individually grows all short roads up to L_{min} pixels.

6.2 Perturb

To perturb a road layout during the search, we randomly choose a road r_i and either scale its current length $l_{curr}(r_i)$ by a random factor between $0.8x$ and $1.2x$, or change its orientation by a random reorientation angle between ± 5 degrees. However, we dynamically update the bounds on the reorientation angle of each road to ensure that an inconsistent turn direction is not introduced. In particular, if the angle α formed by the first(last) segment of road r_i is within ± 5 degrees of the last(first) segment of the previous(next) road we tighten the bounds on the reorientation angle so that it will be less than α. After modifying a road, we rescale the route to fit within the image viewport. By disallowing perturbations that cause inconsistent turn directions and forcing the route to always fit the viewport, we limit our search space to maps that meet our turn direction and image size constraints.

6.3 Score

All other constraints on road layout are enforced through the score function. Our score function examines four main aspects of the road layout; road length, road orientation, intersections between roads, and the shape of the overall route. The goal of each of these score categories can be described as follows:

- **Length:** ensure all roads visible and preserve ordering of roads by length.
- **Orientation:** preserve the original orientations of the roads.
- **Intersections:** ensure that topological errors are not introduced.
- **Overall Shape:** preserve original shape and orientation of entire route.

The scores represent soft constraints on the road layout. The simulated annealing algorithm uses the scoring functions to find a layout that achieves as many of these goals as possible. Unlike hard constraints which must be realized in the final layout, it is possible that the final layout may not attain all the of soft constraints. The constraints enforced in our perturb function, such as ensuring that the map fits within the given viewport, represent hard constraints on our road layout.

As shown in table 6.1 we further refine each of these score categories into a number of component scores. The overall road layout score is calculated as the sum of each of these component scores. We discuss the computation of the component scores in the next three subsections. A summary of the formulations for each component score appears in table 6.2. Once we have developed the scoring function we must balance their effects on the final layout with respect to one another. We describe the balancing process in section 6.3.4.

6.3.1 Road Length and Orientation

Each road r_i is scored on two length-based criteria. First, we penalize any road that is shorter than L_{min} using the following formula:

$$score(r_i) = \left(\frac{l_{curr}(r_i) - L_{min}}{L_{min}}\right)^2 * W_{small} \tag{6.1}$$

where W_{small} is a predefined constant used to control the weight of the score in relation to the other scoring criteria[1]. The function is quadratic rather than linear,

[1]Each of our component scores uses a similar weighting constant

Score Name	Description	Category
1. Small	Ensure roads at least L_{min} pixels long.	Length
2. Shuffle	Preserve ordering of roads by length.	Length
3. Orientation	Preserve original orientation of roads.	Orientation
4. Missing	Penalize missing intersections.	Intersection
5. Misplaced	Penalize misplaced intersections.	Intersection
6. False	Penalize false intersections.	Intersection
7. Extended	Penalize extended intersections.	Intersection
8. Endpt Direction	Preserve overall orientation of route.	Overall Shape
9. Endpt Distance	Preserve overall shape of route.	Overall Shape

Table 6.1: Road layout component score descriptions. To score a road layout we the nine component scores described in this table. Each component score evaluates a particular aspect of the current road layout. The overall score for the road layout is computed as the sum of these component scores.

so roads that are much shorter than L_{min} are given a higher penalty than roads that are just a little shorter than L_{min}. Recall that simulated annealing decides whether to accept the current layout based on the difference between the current score and the previous score. By using a quadratic function, we increase the probability of accepting perturbations which grow the shortest roads because such perturbations yield the largest change in score per pixel length. If we used a linear function growing any of the roads shorter than L_{min} by a fixed amount x would yield the same change in score with no preference for growing the shortest roads.

The second length-based scoring criteria considers the relative ordering of the roads by length. During the initialization of the road layout search we build a list of the roads sorted by their original lengths. After each perturb we build a current version of this length-ordered list and we then add a constant penalty to the score for each pair of roads whose length ordering has shuffled between the original map and the current layout. More precisely, if the current ordering by length between roads r_i and r_j differs from the current ordering by length, then:

$$score(r_i, r_j) = W_{shuffle} \tag{6.2}$$

The purpose of this score is to encourage layouts in which the longer roads *appear* longer than shorter roads in the final map. Therefore, we only consider roads as being shuffled when the difference in their lengths is greater than a predefined perceptual threshold (usually 5-10 pixels).

We also penalize each road r_i, by a score proportional to the difference between its current orientation $\alpha_{curr}(r_i)$ and its original orientation $\alpha_{orig}(r_i)$ using the following formula:

$$score(r_i) = |\alpha_{curr}(r_i) - \alpha_{orig}(r_i)| * W_{orient} \tag{6.3}$$

Since this orientation score is minimized when the current orientation is closest to its original orientation, we only introduce substantial changes to road orientation if the change helps minimize some other road layout score. For example, a substantial change in orientation may be introduced to resolve a false intersection.

6.3.2 Intersections

Both missing and false intersections can be extremely misleading, so we severely penalize any proposed layout containing these problems. While the false and missing intersections scores are essential for maintaining the overall topology of the route, they do not consider the spacing between roads. It is possible for the perturb function to generate road layouts in which non-intersecting roads pass so close to one another that they incorrectly appear to touch. Therefore we also consider *extended intersections* between the roads. We extend the endpoints of each road by a fixed pixel length and penalize the layout if any of the resulting roads intersect.

The scoring function should guide the layout algorithm to the desired layout. One approach is to add a fixed constant penalty when either of these conditions exists. However, this type of scoring function does not provide adequate guidance because the same penalty is always added to the score no matter how severe the false, missing or extended intersection. Suppose our route contains a missing intersection. If we perturb the layout and the missing intersection points end up closer to one another but do not exactly match, the intersection score for this map will not change. The algorithm will not know that moving the missing intersection points closer together generates a better layout. In other words the annealing algorithm is less likely to converge.

In order to guide the layout algorithm to a desirable layout we construct a score that reflects the severity of the intersection problems in a manner that suggests how they might be resolved. We begin by explaining how simple missing and false intersections are resolved independently and then show how scoring must change when a layout contains both missing and false intersections. We then develop a scoring function for extended intersections and describe how the extended intersection score may interact with the false intersection score.

6.3.2.1 Simple Missing Intersections

There are two forms of missing intersections. A true *missing* intersection occurs when two roads should intersect but don't, while a *misplaced* intersection occurs when two roads should intersect and do, but at the wrong point. As shown in figure 6.2, in both cases if roads r_i and r_j are supposed to intersect at points p_i and p_j, we compute a score that is proportional to the Euclidean distance $d(p_i, p_j)$ between these two points using the following formulae:

$$score(r_i, r_j) = d(p_i, p_j) * W_{missing} \tag{6.4}$$

$$score(r_i, r_j) = d(p_i, p_j) * W_{misplaced} \tag{6.5}$$

The proper intersection points p_i and p_j along each road are computed from the parametric value of the intersection in the original unscaled route. Since it is more important for the proper pair of roads to intersect than it is for the point of intersection to be placed exactly, we set the scoring weight for a misplaced intersection to be much lower than for a missing intersection.

6.3.2.2 Simple False Intersections

False intersections occur when the path incorrectly folds back on itself, forming a loop or knot. One way to remove an individual knot is to move the route endpoint closest to the intersection (measured in pixels along the route) towards the intersection

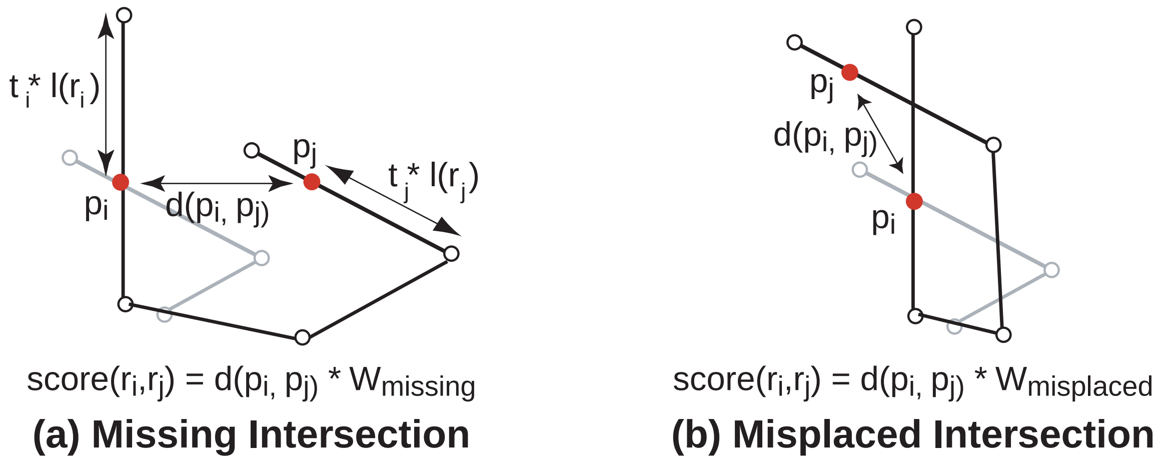

t_i * l(r_i)
p_i
d(p_i, p_j)
p_j
t_j * l(r_j)
score(r_i,r_j) = d(p_i, p_j) * W_missing
(a) Missing Intersection
p_j
d(p_i, p_j)
p_i
score(r_i,r_j) = d(p_i, p_j) * W_misplaced
(b) Misplaced Intersection

Figure 6.2: Scoring missing and misplaced intersections. In both cases the score is proportional to d, the Euclidean distance between the two points p_i and p_j that should intersect (marked in red). Initially for each pair of intersecting roads r_i and r_j we compute the parametric values t_i and t_j of the intersection point. Multiplying these parameters by the current lengths of the roads $l_{curr}(r_i)$ and $l_{curr}(r_j)$ gives us the current position of p_i and p_j. For comparison, the original route is shown in gray.

point. Figure 6.3(a)-(c) illustrates several false intersection scenarios, showing for each intersection point which direction the closest endpoint must move to remove the knot.

For each false intersection we compute a score proportional to the distance in pixels along the route to the nearest endpoint, as shown in figure 6.3(d). The closer the false intersection is to the center of the route, the higher the score. This approach is conceptually equivalent to building a scoring hill along the route that guides the closest endpoint towards the intersection point, thereby unravelling the knot. Therefore if roads r_i and r_j form a false intersection at point p_{int}, we can compute the false intersection score between them as follows:

$$score(r_i, r_j) = min(dRte(p_{int}, p_{origin}), dRte(p_{int}, p_{dest})) * W_{false} \tag{6.6}$$

where p_{origin} and p_{dest} are the point of origin and and destination for the route respectively and $dRte(p_a, p_b)$ is the distance between two points on the route p_a and p_b in pixels along the route (i.e. the arc length between the two points).

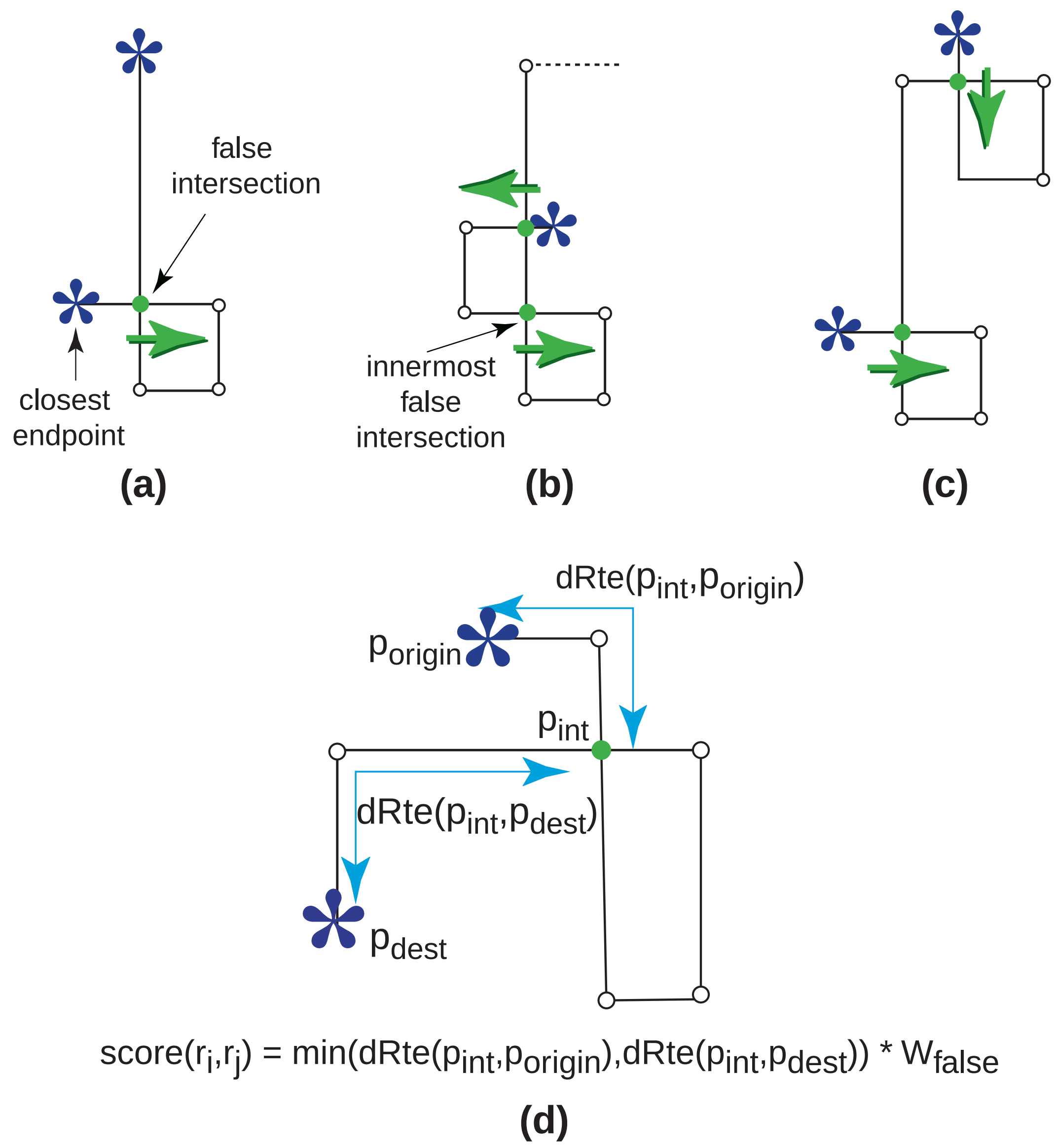


Figure 6.3: Scoring false intersections. (a),(b),(c) The direction the route endpoints should move to independently resolve each false intersection is indicated by the large green arrows. (b) The two false intersections pull the endpoint in opposite directions. This is addressed by counting only the innermost false intersection score. (c) The innermost false intersection is scored for each endpoint independently, so in this case both false intersections are included in the final score. (d) The score for a simple false intersection at point p_{int} is proportional to the distance from p_{int} to the closest endpoint of the route, either p_{origin} or p_{dest}, measured in pixels *along the route.*

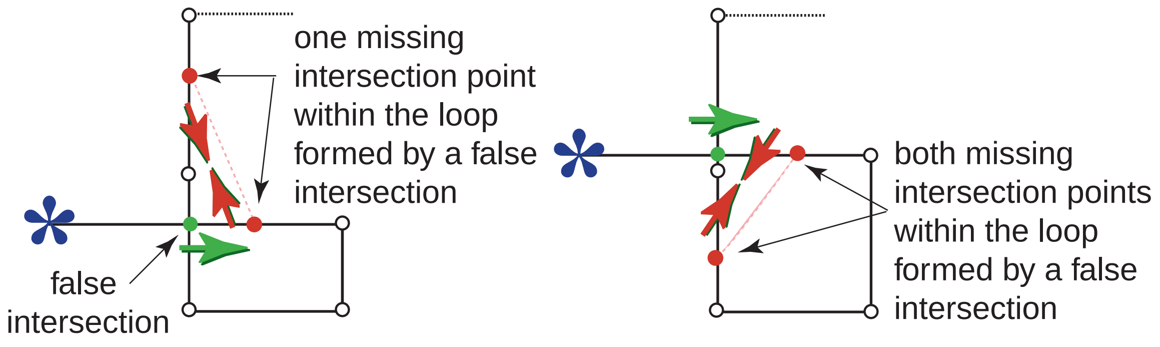
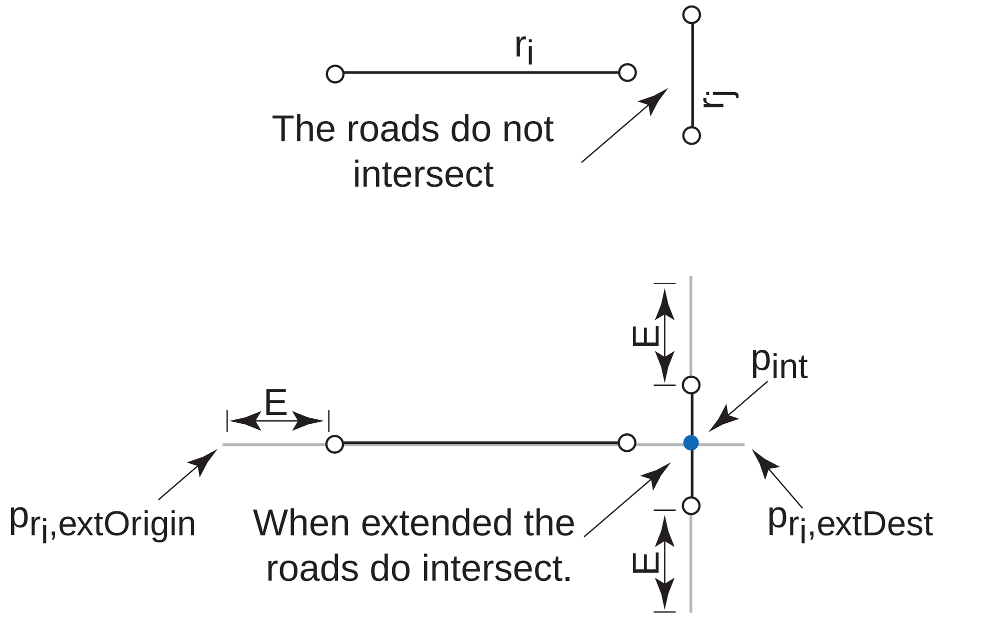


Figure 6.4: Interactions between false and missing intersections. In both these cases, the false and missing intersection scores push points on the route in conflicting directions, as indicated by the arrows. To resolve the conflict, we add a constant penalty for the false intersection and allow the missing intersection score to pull the intersection to the desired location.

When a route contains multiple false intersections, the false intersection scores may conflict and push the endpoint in opposite directions, as shown in figure 6.3(b). We address this problem by counting only the score for the innermost false intersection (working inwards from the endpoint to the center of the route). By penalizing the layout for only the innermost false intersection, we guide the endpoint towards the desired direction and eventually resolve both false intersections. Counting the innermost intersection is done for each endpoint separately. In situations such as 6.3(c), where there are two false intersections but each is closer to a different endpoint, both scores are counted.

6.3.2.3 False Intersections and Missing Intersections

In most cases when false and missing intersections occur in the same map, the scores interact properly to resolve both problems. There is one exceptional situation that occurs when the loop formed by a false intersection contains a missing intersection. As shown in figure 6.4, one score may push in one direction and the other score in the other direction, resulting in a stalemate in which neither problem can be resolved. In both of these cases there is supposed to be an intersection; it is just occurring between the wrong roads. It is often the case that when a missing intersection occurs



r_i
r_j
The roads do not intersect
E
E
E
p_{int}
$p_{r_i,extOrigin}$
When extended the roads do intersect.
$p_{r_i,extDest}$

$$score(r_i,r_j) = \min(d(p_{int}, p_{r_i,extOrigin}), d(p_{int}, p_{r_i,extDest})) * W_{ext}$$
$$score(r_j,r_i) = E * W_{ext}$$

Figure 6.5: Scoring extended intersections. If the extended intersection occurs on the extended portion of the road as for r_i, the score is proportional to the distance between the intersection point and the nearest extended endpoint, either $p_{r_i,extOrigin}$ or $p_{r_i,extDest}$, of the road. If the extended intersection occurs within the main extent of the road as for r_j, the score is set to the largest possible penalty for intersection with the extended portion of the road.

within the loop of a false intersection that the false intersection is simply the missing intersection misplaced.

We resolve the situation with an additional rule: if either point of a missing intersection is inside the loop formed by a false intersection, we add a constant penalty for the false intersection, rather than a hill-based score. With the false intersection score fixed, the missing intersection score can guide the intersection to the desired location, since there is no longer a conflict. Both of the cases shown in figure 6.4 will use a constant penalty for the false intersection, as both contain at least one point of a missing intersection within the false intersection loop.

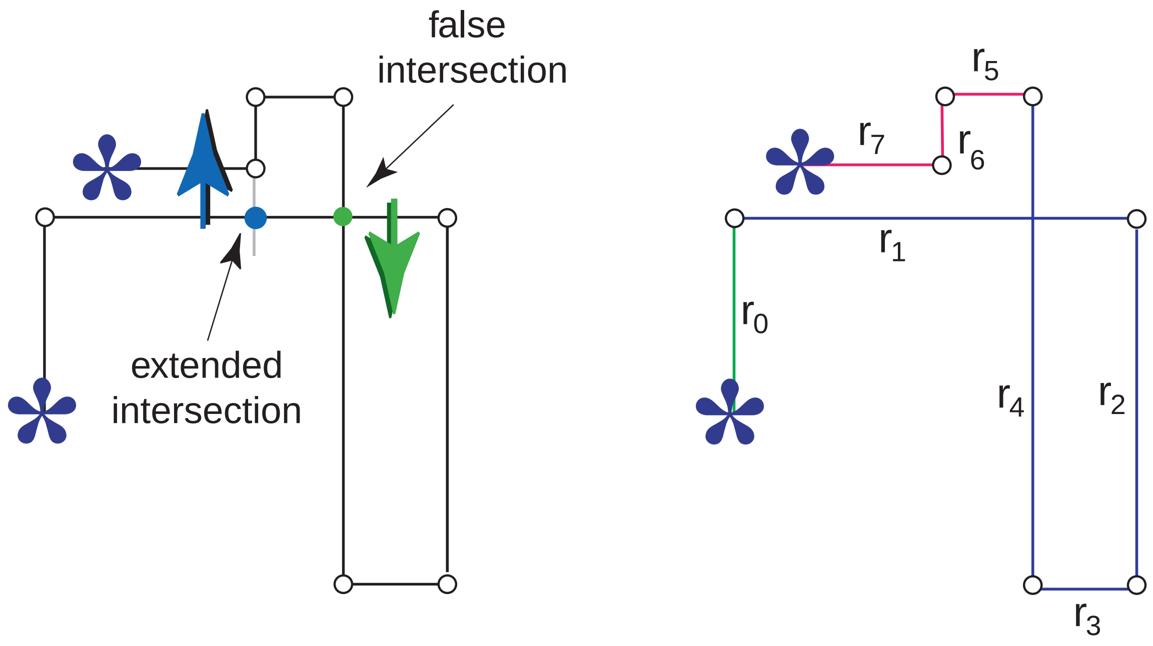


Figure 6.6: Interactions between extended and false intersections. (a) The extended intersection and false intersection scores conflict and push the layout in opposite directions. (b) All roads between an endpoint of the route and a false intersection or between a pair of false intersections are considered to be in the same *false intersection interval*. In this case, there are three intervals $[r_0]$, $[r_1, r_2, r_3, r_4]$, and $[r_5, r_6, r_7]$. We resolve the conflicting scores by only counting extended scores between roads in the same false intersection interval. Since r_1 and r_6 are in different intervals, their extended intersection score is not counted.

6.3.2.4 Extended Intersections

If two roads are not supposed to intersect, we would also like to avoid having them pass close enough to each other that they appear to touch. We therefore compute extended intersections between each pair of roads. We extend the endpoints of each road by a fixed pixel length E and then check if the resulting roads intersect.

Extended intersections are scored as shown in figure 6.5. If the extended intersection occurs on the extended portion of the road as for r_i, the score is proportional to the distance between the intersection point and the nearest extended endpoint, either

$p_{r_i,extOrigin}$ or $p_{r_i,extDest}$, of the road. In this case the score is computed as follows:

$$score(r_i, r_j) = min(d(p_{int}, p_{r_i,extOrigin}), d(p_{int}, p_{r_i,extDest})) * W_{ext} \tag{6.7}$$

As the extended intersection point moves further from the main extent of the road, the score decreases. If the extended intersection occurs within the main extent of the road as for r_j, the score is set to the largest possible penalty for intersection with the extended portion of the road. Since the extension length E is known a priori, the score in this case is simply:

$$score(r_j, r_i) = E * W_{ext} \tag{6.8}$$

As shown in figure 6.6, it is possible for an extended intersection score to conflict with a false intersection score. To reduce such conflicts, we include extended intersection scores only when the extended intersection occurs between two roads in the same *false intersection interval*, as shown in figure 6.6(b). All roads between an endpoint of the map and a false intersection, or between a pair of false intersections, are considered to be in the same false intersection interval. Since roads r_1 and r_6 are in different false intersection intervals we do not include the extended intersection score between them.

When a false intersection is resolved, roads that were in two different false intersection intervals are merged into the same false intersection interval. This can result in a number of extended intersections being included in the road layout score that were not being included before the false intersection was resolved. If the increase in score due to additional extended intersections is larger than the decrease due to resolving the false intersection, the search can get stuck in a local minimum in which the false intersection is almost resolved and the two roads just touch. We eliminate this problem by always adding the maximum possible extended intersection score to each false intersection score. This guarantees that the resolution of a false intersection will result in a decrease in score.

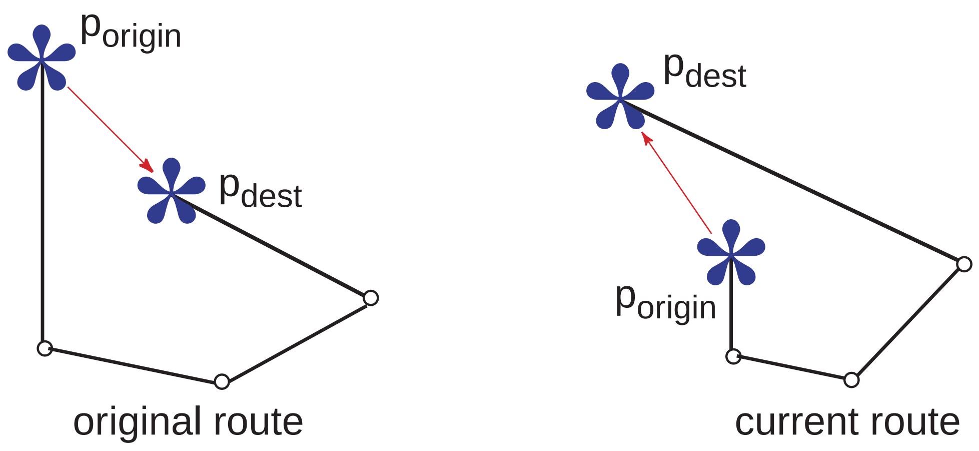


Figure 6.7: Maintaining overall route shape. To maintain the overall shape of the route we compute two scores based on the vector between the endpoints of the route in the original unscaled route and the current route. This endpoint vector is shown in red. The endpoint direction score penalizes maps which alter the direction of this vector. This example would be severely penalized since the direction has changed almost 180 degrees and the origin currently appears southeast of the destination rather than to the northwest of it as in the original route. If the length of the endpoint vector is less than half the original endpoint vector length we add the endpoint distance penalty to the road layout score as well. This ensures that the endpoints do not get too close to one another.

6.3.3 Overall Route Shape

The final road layout score considers the overall shape of the route. Although growing short roads and altering road orientations may be essential for creating a usable route map, such distortions can drastically alter the overall shape of the route. In some cases, it is possible for a destination that should appear to the west of the origin to end up appearing to the east of the origin. Non-uniformly scaling the roads can also cause the origin to appear much closer to the destination than it actually is.

We compute two road layout scores based on the vector between the origin, p_{origin} and destination p_{dest} of the route. We compare the direction and distance of this vector in the current route to its direction and distance in the original unscaled route. The endpoint vector is illustrated in figure 6.7. The endpoint direction score

Score Name	Formulation
1. Small	if $l_{curr}(r_i) < L_{min}$ $score(r_i) = \left(\frac{l_{curr}(r_i) - L_{min}}{L_{min}}\right)^2 * W_{small}$
2. Shuffle	if $order(l_{orig}(r_i), l_{orig}(r_j)) \neq order(l_{curr}(r_i), l_{curr}(r_j))$ $score(r_i, r_j) = W_{shuffle}$
3. Orientation	$score(r_i) = \lvert\alpha_{curr}(r_i) - \alpha_{orig}(r_i)\rvert * W_{orient}$
4. Missing	if missing intersection between r_i and r_j $score(r_i, r_j) = d(p_i, p_j) * W_{missing}$
5. Misplaced	if misplaced intersection between r_i and r_j $score(r_i, r_j) = d(p_i, p_j) * W_{misplaced}$
6. False	if false intersection between r_i and r_j $score(r_i, r_j) = min(dRte(p_{int}, p_{origin}), dRte(p_{int}, p_{dest})) * W_{false}$
7. Extended	if extended intersection between r_i and r_j in same interval if extended intersection on extension of r_i $score(r_i, r_j) = min(d(p_{int}, p_{r_i,extOrigin}), d(p_{int}, p_{r_i,extDest})) * W_{ext}$ if extended intersection on main extent of r_i $score(r_i, r_j) = E * W_{ext}$
8. Endpt Direction	$score = \lvert\alpha_{curr}(p_{dest} - p_{origin}) - \alpha_{orig}(p_{dest} - p_{origin})\rvert * W_{endptDir}$
9. Endpt Distance	if $d_{curr}(p_{origin}, p_{dest}) < K * d_{orig}(p_{origin}, p_{dest})$ $score = \lvert d_{curr}(p_{origin}, p_{dest}) - d_{orig}(p_{origin}, p_{dest})\rvert * W_{endptDist}$

Table 6.2: Road layout component score formations. Summary of the scoring formulae for each component score. The functions and variables used in the formulations are described in section 6.3. All functions are computed based on the current layout unless otherwise specified. We use the subscript *orig* to refer to functions based on the original unscaled version of the route. Note that this table does not reflect any of the special cases for resolving conflicts between false, missing and extended intersection scores.

penalizes layouts that alter the direction of this vector and is computed as follows:

$$score = |\alpha_{curr}(p_{dest} - p_{origin}) - \alpha_{orig}(p_{dest} - p_{origin})| * W_{endptDir} \tag{6.9}$$

The $\alpha_{curr}(p_{dest} - p_{origin})$ and $\alpha_{orig}(p_{dest} - p_{origin})$ functions represent the current and original orientations of the endpoint vector.

We also penalize the layout if the endpoints of the route get too close to one another. If $d_{orig}(p_{origin}, p_{dest})$ is the original distance between the endpoints we would like to ensure that the endpoints are no closer than a factor K times the original endpoint distance. That is if the current distance between the endpoints $d_{curr}(p_{origin}, p_{dest})$ is less than $K * d_{orig}(p_{origin}, p_{dest})$ then we penalize the route layout as follows:

$$score = |d_{curr}(p_{origin}, p_{dest}) - d_{orig}(p_{origin}, p_{dest})| * W_{endptDist} \tag{6.10}$$

Usually K is set less than 1.0 and we have found that $K = \frac{1}{2}$ works well in practice. The distance function in this case is the standard Euclidean distance.

6.3.4 Balancing the Scoring Functions

The constraints embodied in our scoring functions are not equally important. For example, ensuring that all the roads and turning points are visible is far more important than maintaining the original end point direction vector for the route. If there is a choice between growing a short road or maintaining the end point direction vector, we would like our algorithm to grow the short road. Therefore we must balance the effects of the scoring functions with respect to one another so that the more important constraints are given higher priority. We use the weighting constants W to prioritize the scoring functions.

We used an informal usability engineering process to determine the prioritized order for our scoring functions. We chose a set of 5 routes and generated maps containing each of the errors our scoring functions are designed to prevent. To generate these maps we simply turned off each scoring functions one at a time. We then asked a small group of users to rate which maps were most confusing or misleading. Based on these experiments we found the following prioritized ordering for the major classes of constraints:

1. Prevent topological errors: false, missing, misplaced, extended
2. Ensure all roads visible: small
3. Maintain original orientation: orientation
4. Maintain ordering by length: shuffle

5. Maintain overall route shape: endpt direction, endpt distance

Through this usability engineering process we found for example that all three topological errors, false, missing and extended intersections are extremely confusing. In fact, we were somewhat surprised to learn that topological errors are more confusing than shuffles of road length or even extremely short roads. Nevertheless, based on these tests we set preventing topological errors as our highest priority constraints. The remainder of the prioritized list was determined in a similar manner.

The usability engineering process gave us general guidelines for setting the weighting parameters. To determine the exact values for the weighting constants we performed a manual search. We randomly chose weighting constants according to the priorities, and then ran a test suite of 100 maps through the system. For each route we counted the number of topological errors, short roads, shuffles etc. in the final map layout. We then manually adjusted the weighting constants based on these error statistics and re-generated the maps. For example, if the maps contained lots of shuffles we increased the weight of the shuffle score. We kept tweaking the weighting constants in this manner until we generated all 100 maps without a single error.

6.4 Determining Simulated Annealing Parameters

Recall that the simulated annealing search algorithm, as described in section 3.2.2.4, requires setting two parameters, a termination condition or maximum iteration count, and a cooling function Decrease() that decreases the annealing temperature T. Often the cooling function simply sets $T = K * T$ where K is a cooling factor less than 1.0. Both of these parameters, the maximum iteration count and the cooling factor, affect the running time of the algorithm as well as the quality of the search results. If the algorithm is given too few iterations or it is cooled too quickly it may get stuck in local minima. However, if the algorithm is given too many iterations or cooled very slowly its running time becomes much longer.

We use a standard approach for determining the maximum iteration count and cooling factor. We begin by overestimating these factors, running the road layout search with a high iteration count and a very slow cooling rate. Although a single route may take over a day to run with these settings they allow the algorithm to consider an extremely large portion of the search space and escape local minima. Therefore, the resulting road layout should be close to the optimal map layout and this near-optimal road layout becomes our gold-standard. We then manually tune the iteration count and cooling factor to find values which maintain reasonable running times (our goal is to generate most road layouts in about 0.25 seconds) while generating layouts that are close to the near-optimal layout. We perform this manual tuning process on a set of 10 randomly selected routes and took the maximum of the resulting maximum iteration counts and cooling factors to set these parameters for our road layout search. We used a similar approach to set these two parameters for the label layout and context layout searches that are described in the next two chapters.

6.5 Fine-Tuning Road Orientation

Once the search phase of road layout is complete, we snap each shallow angle road in the final layout to the nearest horizontal or vertical axis. Roads that form shallow angles (i.e. < 15 degrees) with the image plane horizontal or vertical axes tend to increase the visual complexity of the map. Furthermore, such roads can be difficult to antialias, especially on personal digital assistant (PDA) displays with limited color support. However, we only reorient a road if doing so does not introduce an inconsistent turn direction or a false, missing or extended intersection. Such reorientation helps regularize the map and produces a cleaner-looking image.

Chapter 7

Label Layout

For the route map to be usable, each road on the map must be labeled with its name. Similarly, the origin and destination of the route should be labeled with their addresses. While the road layout stage of LineDrive is responsible for ensuring that all roads will be long enough to be visible, the label layout stage is responsible for automatically placing the labels on the map so that each one is easy-to-read and clearly associated with a particular object, such as a road or landmark, in the map. As shown in figure 7.1, without such labels a route map is simply an abstract series of lines and is impossible to use for navigation.

Each label is added to the map to communicate a piece of information (i.e. a road's name) through a combination of text and images. We refer to the the graphical elements comprising the label, the arrangement of those elements relative to one another and the constraints on the placement of the elements in the map as a *labeling style*. The final placement of a label as well as its style help communicate which map object (i.e. road, landmark, etc.) it is labeling. We refer to this object as the *label target*.

Automated placement of labels on maps has been well-studied, and several systems have been developed for real-time label placement on standard regional maps [Zor97, ECMS97]. Finding "good" label placements has been shown, in general, to be an NP-complete problem[MS91], and thus the common approach has been to use a randomized search to find near-optimal label placements. Our approach is similar: we

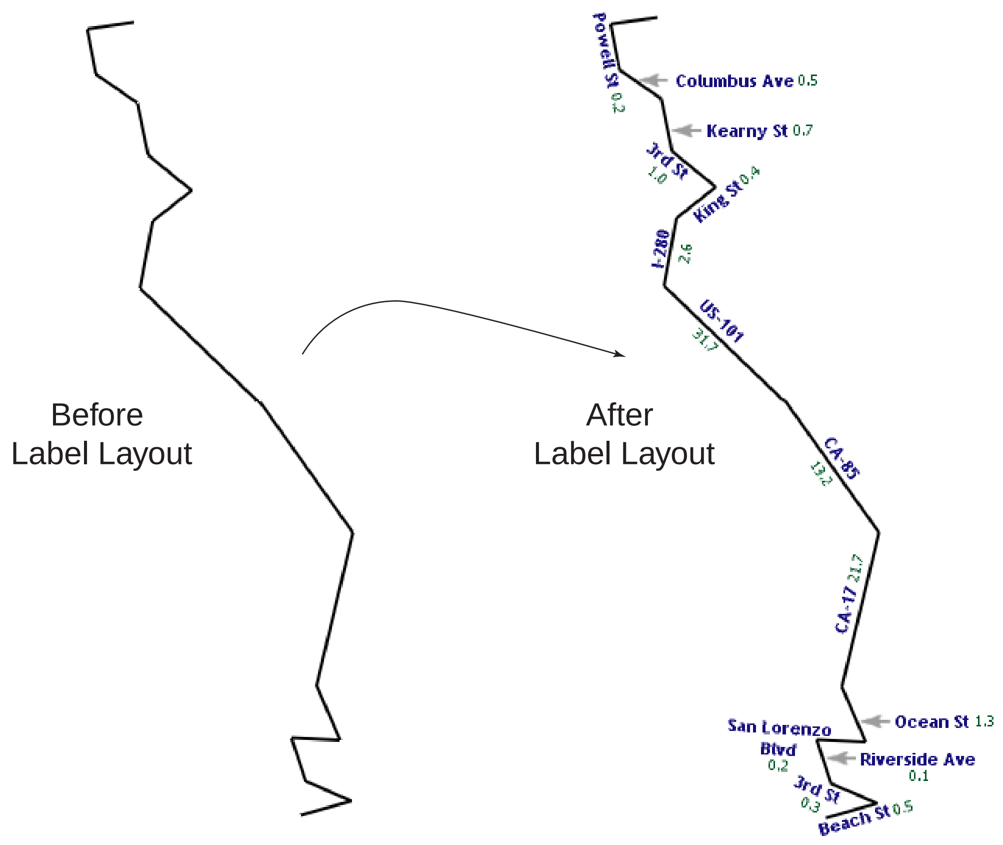


Figure 7.1: Before and after label layout. The label layout stage of LineDrive is responsible for placing all the road labels and context labels in the map. Before label layout the map is an abstract curve. Once the road labels have been placed the image is usable as a route map. The map shown in the after label layout part of the figure contains only road labels. Although we use the label layout engine for placing labels on context objects such as landmarks and cross-streets, the context labels are not placed on the map until the context layout stage of the algorithm described in the next chapter.

first define the search space of possible labelings for a map and then we search the space using simulated annealing to find a near-optimal label placement. However, most existing systems only consider a discrete set of locations and a single style for any label. LineDrive extends the search-based approach to handle a continuous range of locations, defined by bounded regions in the plane, and any number of potential labeling styles for each label. Furthermore, new labeling styles are easily added to our system by simply defining a construction function to perform the relative placement of the necessary images and text, and a scoring function to evaluate placements for that style.

Note that the label layout engine of the LineDrive system is called twice during the route map design process. It is first called immediately after road layout in order to place the labels for each road into the map. It is also called during the context layout stage in order to place the labels for context information such as point landmarks, and cross-streets.

7.1 Labeling Styles

There are many different ways to label a given target object. A typical method for labeling roads is to simply write the name directly above or below the road. This approach uses proximity to associate the label with its target road. Another style is to put the text near the road and then add an arrow pointing to the road to form the association between the name and its target. The labeling style that is best for a particular target usually depends on the geometry of target and the surrounding information on the map. Figure 7.2 shows several styles that might be used to label different objects. As shown in figure 7.3, a *labeling style* is comprised of three components:

- **Graphic Elements:** A set of text and image elements. The primary graphic element is usually a name, and secondary graphic elements can include distance to travel, arrows, highway shields, etc.

- **Arrangement:** The arrangement of the secondary graphic elements relative to the primary element. For example, the arrow-left-of-name labeling style puts the arrow graphic to the left of the primary name graphic.

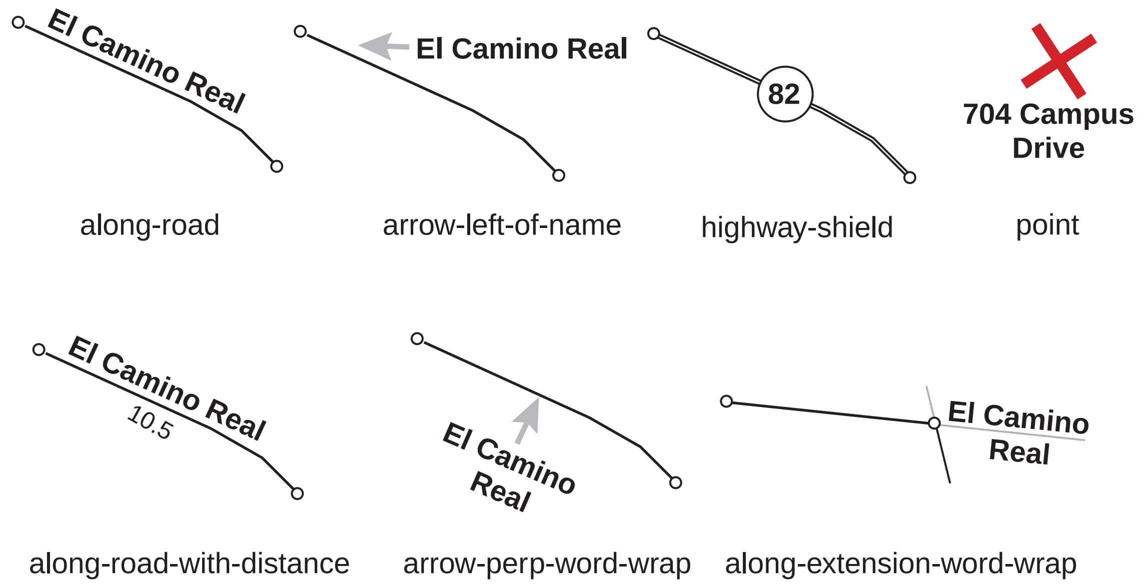


Figure 7.2: Labeling styles. Several possible labeling styles that might be used to label roads or landmarks along the route.

- **Placement Constraints:** Each constraint is a region in the map image defining a set of valid positions and orientations for the center of the primary graphic. The region is defined by either a bounding box or a piecewise-linear curve and a set of orientation vectors.

To place a given label in the map, we must choose both a labeling style and a label location from within one of the placement constraint regions for that style. Therefore, our label layout search space is defined by the set of possible labeling styles and the placement constraints for each style, for every label in the map.

7.2 Initial Layout

In the first step of label layout, we create a list of possible labeling styles for each target object by considering factors such as the size, shape, and type of the target (i.e. highway, residential road, or landmark) and the length of the label name (i.e. if the name is long we might create a word-wrapping style). Each style is also given a rank

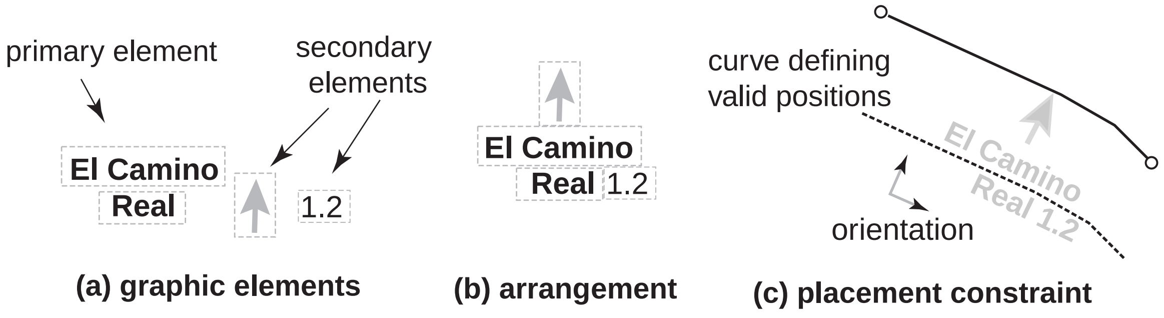


Figure 7.3: Components of a labeling style. The components of a labeling style include (a) a set of graphic elements. (b) an arrangement of those graphic elements relative to a primary graphic, and (c) a constraint specifying the valid positions and orientations for the center of the primary graphic.

based on its desirability. For example, for roads, the *along-road* style is preferable to to the *arrow-left-of-name* style.

We create an initial label layout by placing each label at the most central position within its highest ranked labeling style. We then deterministically fix as many labels as possible. We check if each label in its initial position could ever conflict with the placement of any other label by intersecting each label in its initial position with all potential positions for every other label. The potential positions are determined from the placement constraints defined for each labeling style. If no conflict is possible, then the label is fixed in its initial position and only those labels which are not fixed in this first step are placed during the label layout search. By reducing the number of labels that must be placed during the search we effectively reduce the size of the label layout search space and thereby accelerate the label layout process. Figure 7.4 illustrates this deterministic placement calculation for a given label.

7.3 Perturb

Given the current label layout the perturb function must randomly choose three aspects of the layout to change. First it randomly picks a label to alter. Next it randomly selects a labeling style for that label from among the labeling styles defined

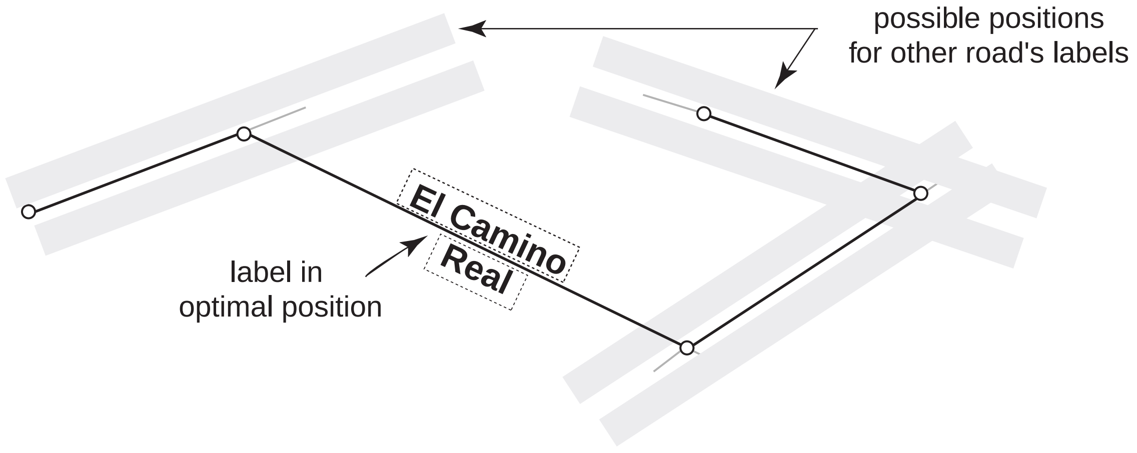
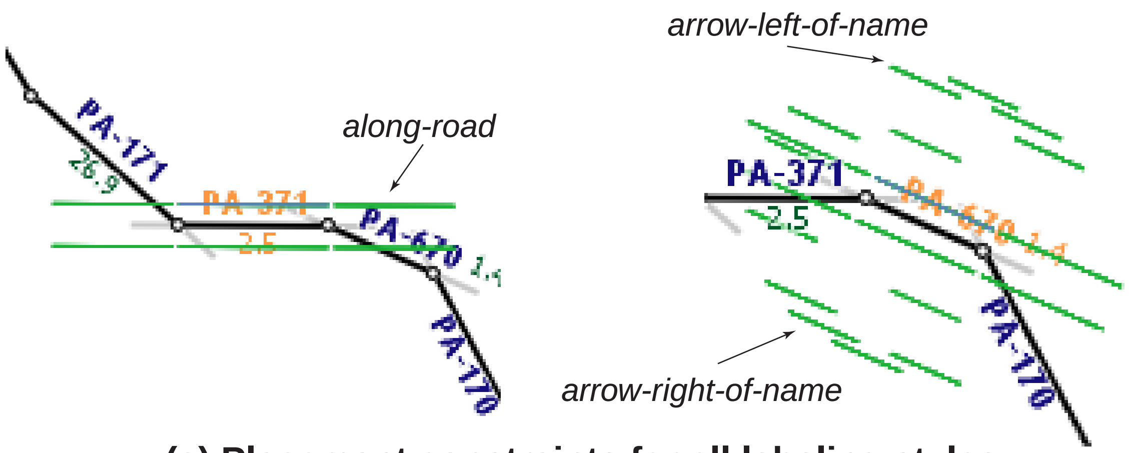


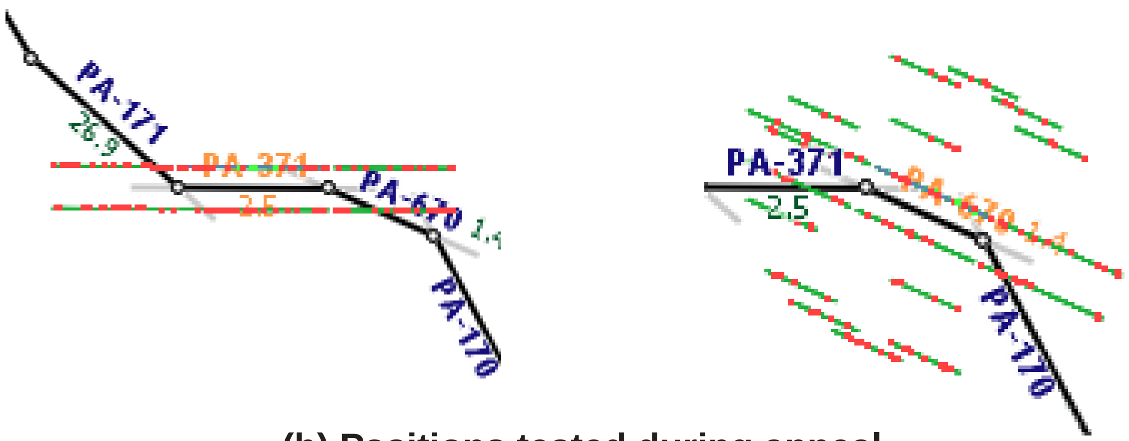
Figure 7.4: Deterministic label layout preprocess. The preprocessing step attempts to deterministically place each label in the most central position within its highest ranked labeling style. A label can only be placed in this initial step if it guaranteed to never interfere with the placement of any other label. Here, El Camino Real's label can be deterministically placed because it does not intersect any possible position for any other label.

fro the label. Finally it randomly chooses a new location for the label from within on the of the style's placement constraints. Before each perturb operation we store the labeling style and placement of each label so that after the perturb it is always possible to revert back the previous label layout if necessary.

Figure 7.5(a) is a visualization of the placement constraints for two road labels. The *along-road* style is the optimal labeling style because it generates a clear visual association between the label and its road, without introducing extra graphic elements such as an arrow. Therefore, all road labels are given the *along-road* labeling style. Only labels that are longer than their underlying roads are allowed to use other labeling styles including those that contain arrows. Figure 7.5(b) depicts all the label positions tested during the simulated annealing search for these two road labels. Each of these label positions was generated by the perturb function.



(a) Placement constraints for all labeling styles



(b) Positions tested during anneal

Figure 7.5: Label placement constraints and label positions tested. (a) The set of placement constraints across all labeling styles for two labels. All labels are given the *along-road* constraint because it generates the clearest association between the label and its target road. Only when a label is longer than its road as is the case for PA-670, we generate additional labeling styles and their associated placement constraints for the label. For PA-670 the other labeling styles include *arrow-left-of-name* and *arrow-right-of-name*. (b) The perturb function randomly selects a label to perturb and then alters its labeling style and placement. Each red dot in this visualization represents a label position tested during the anneal.

7.4 Score

The label layout scoring function evaluates each label on the following criteria: (1) the proximity of the label to the center of its target, (2) the rank of the chosen label style, and (3) whether the label intersects or overlaps any other object in the map. The score for the complete map labeling is computed as the sum of the scores for each label.

The two main goals of label layout are to place labels on the map so that they are clearly associated with their targets and are easy to read. The proximity score guides the label layout search towards the first goal, by increasing the label layout penalty as a label gets further away from its target. The labeling style rank score and the intersection score guide the search towards the second goal. The labeling style rank score penalizes layouts using complex graphic arrangements, while the intersection score penalizes layouts in which labels intersect other map objects.

For simplicity we describe each of the component score computations, assuming the target for the label is a road. Extending the computation to other target objects such as landmarks is straightforward. We will describe the major differences in scoring road labels versus cross-street and landmark labels as necessary. We conclude this section by describing how we balance these three scoring criteria with respect to one another and then present methods for accelerating the computation of the scoring functions.

7.4.1 Proximity to Target

The proximity of the label to its target is computed as the Euclidean distance between the center of primary graphic of label l_i and the center of its target road r_i, as follows:

$$score(l_i) = \frac{d(center(l_i), center(r_i))}{maxCenterDist(l_i, r_i)} * W_{proximity} \tag{7.1}$$

The normalizing factor $maxCenterDist(l_i, r_i)$ is the maximum possible distance between the center of the label and the center of the road. Since the placement constraints on the center of the label and the center of the road are known a priori, this is factor can be precomputed for each label. The proximity score is minimized as the label gets closer to the center of the road, and it therefore pulls the label towards the center of the road. We believe that it is easier for users to form the association between a label and its target road the closer the label is to the center of its road.

Just as for the road layout scores we include a predefined constant parameter in each of the label scores (in this case is it $W_{proximity}$) so that we can adjust the weight of each component score in relation to the other component scores.

7.4.2 Rank of Labeling Style

As we described earlier, the rank of each labeling style is directly related to the desirability of the labeling style. A simple labeling style such as *along-road* is preferable to a more complex labeling style such as *arrow-left-of-name* because it contains fewer graphic elements and its placement constraints usually allow the label to be placed closer to the road. Since we search for the lowest scoring label layout during the anneal, we set the rank of each labeling style to be inversely proportional to its desirability. Then we can score label l_i based on the rank of its labeling style as follows:

$$score(l_i) = labelStyleRank(l_i) * W_{rank} \tag{7.2}$$

7.4.3 Intersections

A label that overlaps other objects in the map can be difficult if not impossible to read. Therefore, the intersection score is designed to penalize label layouts in which labels intersect other objects in the map. Our basic label intersection score considers label intersections with two main types of map objects; roads and other labels. We consider each of these in turn.

7.4.3.1 Intersecting Roads and Labels

If label l_i intersects road r_j, we include a constant penalty in the score as follows:

$$score(l_i) = W_{roadIntersection} \tag{7.3}$$

This penalty is added for each road the label intersects. Note that the placement constraints of each labeling style are created so that a label cannot intersect its own road.

In most cases when a label intersects another map object the readability of a label is directly proportional to the amount of overlap between the objects. For example the more the overlap between two labels, the more difficult it is to read either of them. Yet, even though intersections between labels and roads can reduce readability, it is almost always possible to read labels despite such intersections. The principally one-dimensional nature of a road versus the two-dimensional nature of a label means that it is unlikely that the road will overlap enough of the label to make it unreadable. Therefore a constant penalty regardless of the amount of overlap between the label and road performs well in practice.

For label-label intersections, on the other hand, we compute an intersection score proportional to the area of overlap between the two labels. This approach guides the label layout towards reducing overlap areas as much as possible even when two labels intersect. We compute the label-label intersection score between labels l_i and l_j as follows:

$$score(l_i, l_j) = \frac{overlapArea(obb(l_i), obb(l_j))}{minArea(obb(l_i), obb(l_j))} * W_{labelIntersection} \tag{7.4}$$

where $obb(l_i)$ and $obb(l_j)$ are the two-dimensional oriented bounding boxes around labels l_i and l_j respectively. Since the maximum area of overlap between these two bounding boxes at most the area of the smallest of the two bounding boxes $minArea(obb(l_i), obb(l_j))$, we can normalize the score by this factor.

7.4.3.2 Accommodating Context Objects

As shown in figure 4.2 LineDrive places road labels before laying out context objects such as point landmarks and cross-streets. We chose to partition the layout problem in this manner because road labels are essential for creating a usable route map. While context information can improve usability it is secondary to roads and their labels.

Based on this reasoning the initial version of LineDrive performed the road label layout completely independently of the context object layout. The road label layouts were scored without considering where the context information might be placed and we left it up to the context layout search to find an appropriate layout for the context information. While this approach produced excellent road label layouts, it generally did not find very good context object layouts. Moreover, in the majority of cases slightly moving a road label would have created a much better space for the context object.

Therefore, we have added another factor to the road label intersection score that attempts push road labels away from the ideal space for context objects. Before performing the road label layout, for each context object, whether it is a point landmark or a cross-street, we pre-compute an oriented bounding box representing its optimal location as described in the next chapter. This optimal location bounding box is essentially a *hint* about areas of the map the road label layout should avoid if possible. If a road label l_i intersects a hint we add a small constant penalty to the overall label layout score as follows.

$$score(l_i) = W_{hint} \tag{7.5}$$

Since intersecting a hint increases the overall layout score, road labels are pushed away from the hint bounding boxes when possible. Hinting the road label layout in this manner allows us to generate much better context layouts.

7.4.3.3 Intersection Score for Context Object Labels

The basic label layout engine is also used to place the labels of context objects, such as names of landmarks and cross-streets. However, since the context objects are placed at the same time as their labels we can intersect the labels with the current bounding box of each context object rather than with the hints as described in the previous section. Just as when intersecting a road or hint we add a constant penalty to the context label layout score for intersecting a cross-street or landmark as follows:

$$score(l_i) = W_{cross-street} \tag{7.6}$$

$$score(l_i) = W_{landmark} \tag{7.7}$$

The penalty for intersecting a cross-street is less for intersecting a landmark, because like the main roads, cross-streets are essentially one-dimensional objects that do not reduce legibility as much as two-dimensional landmarks.

We also slightly modify label-label intersection score for context labels. If a context label intersects another context label we score it just as described in equation 7.4. However, context labels intersecting with road labels can make it extremely difficult to read both labels. Since the context label is less important than the road label we add an extremely large penalty to the context label layout score in such cases.

$$score(l_i) = W_{roadLabelIntersction} \tag{7.8}$$

By setting $W_{roadLabelIntersection}$ to be much larger than the maximum score for the standard overlap based label-label intersection score $W_{labelIntersection}$, we force the context label layout stage to avoid such intersecting layouts.

7.4.4 Balancing the Scoring Functions

Once we have developed the scoring functions we need to balance their effects with respect to one another. Just as for road layout we use the weighting constants W to prioritize the scoring functions. Using the same informal usability engineering process described in section 6.3.4 we found the following prioritized ordering for the road label layout scoring functions:

1. Prevent label-label intersections.
2. Ensure proximity of label to target road.
3. Prevent label-road intersections.
4. Choose desirable labeling style.
5. Prevent label-hint intersections.

The ordering of the context label layout constraints are similar:

1. Prevent label-label intersections.
2. Prevent label-landmark intersections.

3. Ensure proximity of label to target context element.
4. Prevent contextLabel-street/road intersections.
5. Choose desirable labeling style.

In both cases preventing intersections with labels is the top priority because such intersections can make it difficult to read either label. Similarly preventing label-road or label-street intersections is far less important because it is still possible to read labels even if such intersections occur. In general the label layout algorithm must first try to prevent intersections that would reduce readability, then it must place the label near its target and finally it must choose the highest ranking, or most desirable labeling style for each label.

7.4.5 Accelerations

We use two simple optimization techniques to accelerate the score computations for each label layout. The first technique relies on the fact that on each iteration of the simulated annealing search we perturb exactly one label. Therefore, the component scores for most of the labels remain unchanged. In fact, if label l_i is perturbed, then at most only the component score for l_i and the label-label intersection scores for the other labels can change. Based on this observation, we have implemented an incremental score update algorithm. After generating the initial layout we compute all the component scores for every label. Thereafter, we recompute only the subset of layout component scores affected by a perturb or revert operation. Intersection calculations are the most expensive part of the label layout search. This incremental approach can greatly reduces the number of label-road intersections we must perform on each iteration of the label layout search.

Our second acceleration technique is also aimed at reducing the number of intersection calculations we must perform. We initially build a regular two-dimensional spatial subdivision grid in the map image plane and place all the map objects within this grid. Then, to compute the intersections between label l_i and the other map objects we first check if the grid cells containing l_i contain any other objects. The objects in the same cells as l_i are a conservative superset of the objects that actually intersect l_i and therefore those are passed into an exact intersection function which computes whether or not the objects actually intersect $l - i$. This is a standard approach in computer graphics for accelerating intersection computations by eliminating large regions of space that cannot possible contain an intersection [AK89, Sam90].

7.5 Two-Phase Simulated Annealing

In practice we usually specify four to five labeling styles and a large set of placement constraints for each road label. Therefore the space of road label layouts can be extremely large. While the basic simulated annealing algorithm does a good job of sampling this space, we have found that in many cases the labeling style for the labels may be well chosen, but the position of the label within the placement constraint for that style may not be optimal. In particular the layout could have been slightly improved by sliding a label to a slightly better position within its constraint. This problem is largely due to the fact the the high dimensionality of the search space does not allow for adequate sampling within a particular labeling style.

We have found that we can improve the quality of our layouts by performing the anneal in two phases. The first phase is performed as described above. Both the labeling style and the position of a label are perturbed on each iteration. After the first phase is complete we compute the score for each label and for those labels whose score is greater than a pre-defined tolerance we apply a second annealing phase. However, in the second phase we assume that the labeling styles chosen in the first phase were good and therefore we only perturb label positions. This approach reduces the dimensionality of the label layout space for the second phase and thereby allows us to sample a much greater number of positions for each label.

Chapter 8

Context Layout and Decoration

Although a route map is functional once the road labels have been placed, additional context features and decorative enhancements can greatly improve the usability of the map. Yet, context features and decorative enhancements are secondary information, not necessary for communicating the basic structure of the route, and therefore must be added carefully so as not to clutter the map with excessive detail. In this chapter we describe the two final stages of the LineDrive system, context layout and decoration, which are responsible for placing context features and rendering the map with decorative enhancements. Figures 8.1 and 8.4 show how the map evolves as it passes through each of these two stages.

The layout algorithm for placing context features such as cross-streets and point landmarks, is similar to the algorithm used for placing labels. For each context feature we define a set of possible layout positions and an score function which evaluates the fitness of a given position. The main objective of context layout is to find a place for all the context features such that the context features do not interfere with the primary route information which consists of the main roads and their labels. Given the score function we then apply simulated annealing to search for a context feature layout which minimizes the context layout score.

Several aspects of the style in which a route map is rendered can subtly communicate information about the route. For example, placing bullets at each turning point indicates that the navigator must make a decision about which road to follow at that

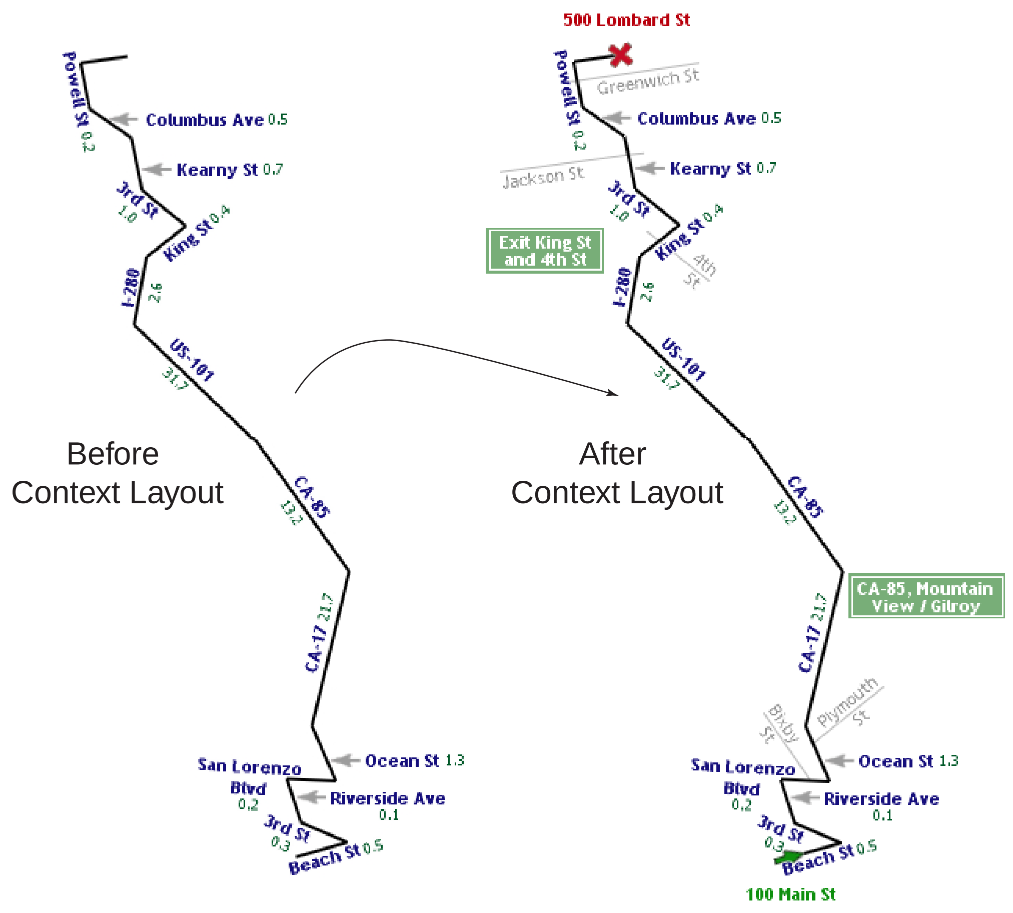


Figure 8.1: Before and after context layout. The context layout stage of LineDrive is responsible for placing cross-streets and point landmarks such as highway entrance and exit signs, in the map. Since the context information is secondary to the main turning point information, the context objects are carefully placed so as not to clutter the map with excessive detail. Only the most important of the context objects are placed and they are rendered in light, desaturated colors to reduce interference with primary map information.

point in the route. Similarly it is possible to indicate the type of a road (i.e. highway, residential or ramp) by using a different line style for each road type. The decoration stage of the LineDrive algorithm adds several such stylistic elements to the map in order to improve map usability.

8.1 Context Layout

LineDrive attempts to place two forms of context within a route map: (1) linear features that intersect the main route, such as cross-streets, and (2) point landmarks along the route such as buildings and signage. Since the roads and their labels constitute the essential information in a route map, the additional context information is only added sparingly around turning points in order to avoid adding clutter to the map. We use the same basic approach for placing both cross-street and local point landmarks. We first present the context layout algorithm in terms of placing cross-streets, and then describe how the layout algorithm must be modified for placing point landmarks.

8.1.1 Preparing the Cross-Street Data

We begin the cross-street layout stage by querying the geographic database for all cross-streets intersecting the main route. The query algorithm searches along each road of the route, for intersecting cross-streets and returns each intersecting street as a separate entity. Thus, if the same cross-street intersects the route more than once, the query algorithm reports multiple versions of the cross-street, one per intersection. To prepare the cross-street data for the search we first simplify the data and then compute an initial cross-street layout.

8.1.1.1 Simplification

For each cross-street the context layout receives a piecewise linear curve of latitude/longitude shape points located in a small neighborhood centered about the intersection point with the route, the name of the cross street, the name of the *main road* the cross-street intersects and an importance value for the cross-street. If the importance value is not pre-specified we place highest importance on the last major cross-street just before each turning point. We have found that these streets are helpful as a warning that the turn is approaching. Since navigators are less likely to be familiar with the region around the destination, we also set the cross-streets near the route destination to be slightly more important than those near the route origin.

Cross-streets can either form T-intersections or X-intersections with their main roads. We initially compute the *original intersection point* between every cross-street and the main route as well as the type of intersection T or X, formed by the cross-street and the main road. We then simplify each cross-street into either one or two segments. If the cross-street forms a T-intersection we form a single segment running from the furthest cross-street shape point to the intersection point. If the cross-street forms an X-intersection we create two segments running from the furthest cross-street shape points on either side of the main road to the intersection point. We also store the orientation angle of each segment with respect to the main road. For some X-intersection cross-streets, the name of the cross-street changes across the intersection. Although the geographic database returns two separate cross-streets in these cases, we unify them into a single a two-segment cross-street and give it two separate names, one for either segment on opposite sides of the main road.

At this point we may have several cross-streets for each main road. Yet, for most roads it is unlikely that more than one cross-street will be helpful for navigation. Each cross-street breaks a main road into intervals and navigators must usually be aware which interval they are currently located in. Extra cross-streets can make it more difficult for navigators to match their real-world location to a map location since they have more intervals to keep track of. With a single cross-street navigators need only know if they are currently located before or after the cross-street. Extra cross-streets tend to clutter the map. Therefore, as the final step of cross-street simplification we reduce the set of cross-streets along each main road to the single cross-street of highest importance, which is usually the cross-street just before the turning point. Note that this reduction must be performed after cross-street unification so that cross-streets forming X-intersections that change names at their intersection point are not turned into single segment T-intersection cross-streets.

8.1.1.2 Initial Layout

The ideal location for each cross-street is its original intersection point with the main road. Therefore, we initially place each cross-street at this original intersection point. However, if the cross-street interferes with other map objects it may not be acceptable

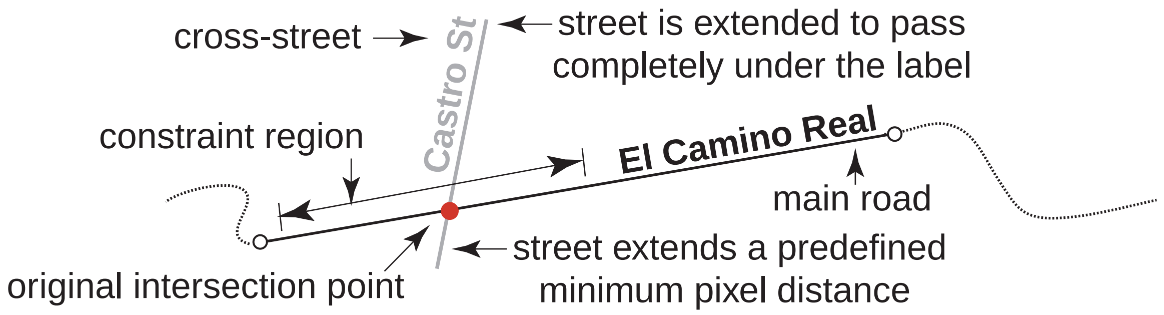

cross-street
street is extended to pass completely under the label
Castro St
constraint region
El Camino Real
main road
original intersection point
street extends a predefined minimum pixel distance

Figure 8.2: Placement constraints for cross-streets. The cross-street layout search considers placing Castro street within the constraint region as close to the original intersection point as possible. Once the cross-street and its label are placed, the cross street is extended to a minimum pixel length on either side of its base road, and if necessary, it is further extended to pass under its label.

to leave the cross-street at this initial position. For this reason, we also create a constraint region around the original intersection point which specifies the acceptable range of positions for the intersection point, as shown in figure 8.2. The constraint region is defined in terms of a parameter t along the main road. If the original intersection point lies in the first half of the main road then the constraint region extends over the parametric range $[0.05, 0.5]$. Similarly, if the original intersection point lies in the second half of the main road the region extends over the range $[0.5, 0.95]$[1]. Note that the constraint regions are designed to include positions that are at least a minimum parametric distance of $0.05 * t$ from route turning points. By maintaining this gap we ensure that cross-streets will not clutter or obscure the important turning points in the final map. Finally, cross-street labels are created just like main road labels and initially placed using the same rules, as described in the previous chapter.

[1]If we allowed more than one cross-street to be placed along a main road we would modify these constraint intervals so that the ordering of cross-streets along the main road could not be altered. Each cross-street would be given a disjoint interval in the order the streets initially intersected the main road.

8.1.2 Perturb

The perturb function for context layout randomly selects a cross-street and then randomly changes either the position of the intersection point between the cross-street and the main road, the position of the cross-street label, or whether the cross-street is hidden. Once the street is perturbed, we set the length of the cross-street to a predefined minimum extension length. Then, if the label has been placed directly above or below the cross-street, we extend the street to pass completely over or under its label.

Unlike the perturb functions for road layout and label layout we allow the context layout perturb function the option of hiding context features. In some cases two or more cross-streets may be located so that no matter where they are placed within their constraint regions they will interfere with one another and all of these cross-streets will end up with poor layout scores. With the option of hiding cross-streets it becomes possible to find a layout in which some cross-streets are hidden in order to make space for the other cross-streets. Hiding context features is acceptable because such features provide secondary information that is not essential for navigating the route.

It is possible that in some cases the cross-street constraint region along the main road will contain a significant curve. In such situations we have two choices for orienting the cross-street as the position of the intersection point is perturbed within the constraint region. We can either maintain the original orientation of the cross-street even as the underlying main road curves, or we can reorient the cross-street so that the angle between the cross-street and the main road is preserved. We have found that even though the latter option may significantly alter the orientation of the cross-street from its true orientation, it is more important to maintain the turning angle at the intersection rather than the overall direction of the cross-street. Since navigators do not travel down a cross-street and only see the intersection with the cross-street maintaining the true direction of the cross-street is less important than maintaining the turning angle at the intersection.

8.1.3 Score

We score each cross-street based on four criteria: (1) the distance between the current position of the cross-street intersection point and the true intersection position, (2) the number of other objects the cross-street intersects, (3) whether or not the cross-street is hidden, and (4) the layout score of the cross-street label. The fourth component score based on the placement of the cross-street label is described in the previous chapter. Here we consider each of the first three component scores in detail. We then describe the prioritized ordering of these scoring criteria.

8.1.3.1 Maintaining the Intersection Point Position

Ideally we would like the cross-streets to be placed as close to their original intersection points as possible. Therefore, if $iPt_{orig}(c_i)$ and $iPt_{curr}(c_i)$ are the original and current intersection points for cross-street c_i we compute a position-based score for c_i as follows:

$$score(c_i) = dRte(iPt_{curr}(c_i), iPt_{orig}(c_i)) * W_{streetPosition} \tag{8.1}$$

The $dRte()$ function computes distance along the main road between the current and original intersection points. Thus, the further the current intersection point is from the original intersection point the higher the score.

Usually the main road runs straight between the two intersection points and $dRte()$ reduces to the standard Euclidean distance. However, if there is a curve in the main road calculating $dRte()$ requires computing the arc length of the main road between the two intersection points. In practice, we simply store the intersection points as parameter values with respect to the main road. Suppose $t_{i,orig}$ and $t_{i,curr}$ are the parameters along the main road r_i of the original and current cross-street intersection points. Since the current length of the main road $l_{curr}(r_i)$, is known after road layout we can compute the distance between the intersection points as follows:

$$dRte(iPt_{curr}(c_i), iPt_{orig}(c_i)) = \|t_{i,orig} - t_{i,curr}\| * l_{curr}(r_i) \tag{8.2}$$

8.1.3.2 Intersections

Just as in label layout we add a constant penalty to the cross-street layout score for each map object, other than its own main road, the cross-street intersects. The basic form of the intersection score is:

$$score(c_i) = W \tag{8.3}$$

where W is dependent on the type of map object the cross-street intersects. We consider intersections with four types of objects, roads, labels and cross-streets and landmarks, each with its own weight factor W_{road}, W_{label}, $W_{cross-street}$, and $W_{landmark}$ respectively.

Our geographic database only provides local information about each cross-street, near its intersection with the main road. In particular we do not have any information about whether or not the cross-street intersects other roads or cross-streets. It could be extremely misleading to show intersection between streets and roads that do not actually intersect. Since we do not have enough information to determine whether a cross-street intersects any other roads or streets, we severely penalize cross-street layouts which contain any such intersections, by setting W_{road} and $W_{cross-street}$ to be very large. While intersections between cross-streets and label or landmark increase clutter and make the map more difficult to read, such intersections, unlike intersections with other roads or street, cannot give navigators false impressions about the topology of the local road network. Therefore, set W_{label} and $W_{landmark}$ to be much smaller than W_{road} and $W_{cross-street}$.

8.1.3.3 Hiding

If a cross-street is hidden we add a penalty proportional to the importance of the cross-street as follows:

$$score(c_i) = importance(c_i) * W_{importance} \tag{8.4}$$

We would like to place as many cross-streets as possible. By penalizing hidden cross-streets in this manner we only hide the streets when hiding is necessary to alleviate intersections between cross-streets. If there is a choice between hiding two cross-streets we would like to hide the least important of the two. Thus, we set the hiding penalty proportional to cross-street importance in order to guide the cross-street layout search to hide the least important streets in favor of hiding more important streets.

8.1.3.4 Balancing the Scoring Functions

For cross-street layout we again created maps containing the errors that these scoring functions are designed to prevent and then asked a small group of users to informally rate which errors were most confusing or misleading. The usability engineering process revealed the following prioritized ordering for the constraints:

1. Prevent intersections with other map elements.
2. Maintain original intersection point with main road.
3. Cross-street label layout score.
4. Prevent hiding cross-street.

8.1.4 Cleanup

Once the search phase of cross-street layout is complete, we clean up the layout. If the label of a cross-street overlaps any other object on the map, we remove the cross-street from the map. Label-object overlap can make the label difficult to read and obscure important route information. Since cross-streets are secondary features, removing them from the map is preferable to allowing such overlap. We do, however, allow the cross-streets to intersect other map objects. This is acceptable because cross-streets are thin, 1D objects, and are drawn underneath the other map objects in a light gray color so that they do not interfere with the legibility of the other objects. The clean up is performed in order from least important to most important cross-street.

Therefore if the labels of two cross-streets overlap, only the least important cross-street is removed from the map. Finally, we clip each cross-street to every other road and cross-street in the route. This final step ensures that we do not introduce any false cross-street intersections in the maps.

8.1.5 Placing Local Point Landmarks

Like cross-streets, point landmarks such as buildings and signage along the route can provide valuable context information and confirmation that the navigator is correctly following the route. While our search-based approach to placing point landmarks is similar to our approach for placing cross-streets, there are also a few fundamental differences. These differences are largely due to the fact that there are many different types of point landmarks, including stop signs, the gas station on the corner, highway entrance and exit signs, and the local McDonald's. In our maps these point landmarks also include the icons at the start and end of the route as well as traffic circles located along the route. In this section we consider how the type of landmark affects three aspects of the landmark layout search; (1) the importance value of the landmark, (2) the constraint region for the landmark and (3) the position-based score for evaluating the placement of a landmark.

8.1.5.1 Point Landmark Importance

Rules for determining the importance of landmarks are not well-understood. A landmark that is important to one person may be insignificant to another person. Yet, because point landmarks are secondary information we would like to place them on the map in order of most important to least important. If, due to lack of space, there is a choice between placing two different landmarks we place the one that is more important.

Cognitive psychologists and cartographers have developed a number of different theories for determining the salience of landmarks [Mac95, Lyn60]. Based on our own experience, the rules for determining landmark importance are usually dependent on the type of landmark. For example, during cross-street layout we apply the rule that the last major cross-street before a turn is more important than the second-to-last cross-street. However, this same rule probably does not apply to buildings along the route. If the second-to-last building is a McDonald's while the last building is a residential apartment building, the McDonald's is probably the more important landmark.

Choosing an importance value for each type of landmark falls outside the scope of the LineDrive system. We believe that a designer must set importance values based on knowledge about the type of landmark and the purpose of the map. However, as a general principle we have found that even broad importance categorizations are useful in designing the landmark layout search-algorithm. We have already described how we broadly set the importance values for cross-streets based on their positions along the route. As a contrasting example, consider the problem of setting importance for highway entrance and exit signs. Based on our experience, we believe that the exit signs between highways and standard residential roads are most important, followed by highway entrance signs. This ranking is based on the fact that drivers usually exit off highways by finding a sign that contains either the number or name of the road they must turn onto. Since there are no stops on the highway it is easy to miss exits and therefore the more information the driver has about an exit the easier it becomes to correctly navigate the route. While highway entrance signs can be useful, drivers are usually traveling slower and can usually see the highway from residential road, making it easier to enter the highway without the entrance sign information.

For cross-streets we use importance in three ways; to initially reduce the number of cross-streets we attempt to place, to penalize hidden cross-streets, and to order the cross-street cleanup at the end of the layout search. Point landmark importance is used similarly. Initially we remove any landmarks that fall below a given importance threshold. During the anneal we penalize hidden landmarks by a score proportional to their importance. During cleanup we remove conflicting landmarks in order from least important to most important.

LineDrive also has the ability to break the landmark layout search into independent phases based on the importance. We initially bin the point landmarks into two or more importance categories and then perform simulated annealing in phases, placing the landmarks in the highest ranking bin first, then the second bin and so on. For example, all the highway exit signs are annealed first and then the highway entrance signs are annealed. The higher ranking landmarks are more likely to be placed since their are fewer potential map objects they might intersect. By breaking the context layout search into independent phases we reduce the dimensionality of the search space.

8.1.5.2 Constraint Region

We have already described placement constraint regions for label layout and cross-street layout. For point landmarks we have found that the size of the constraint regions fall into three categories; landmarks constrained to lie on a single point, landmarks constrained to lie along a main road, and landmarks constrained to lie near a turning point. Some landmarks such as the origin and destination icons as well as traffic circles lie directly on the route and are tightly constrained. In fact, there is exactly one location in our map corresponding to each of these types of landmarks. For example, the origin icon must be placed at the origin of the first main road in the route. However, unlike the landmarks, we allow much greater flexibility in placing the labels for these landmarks. As shown in figure 8.3(a) the landmark labels usually have several placement constraints surrounding the landmark itself.

Like cross-streets, some point landmarks such as buildings located along a road can be placed within a constraint region parallel to the road. For these landmarks we use the same type of placement constraints as we use for cross-streets and road labels generated with the *along-road* labeling style. Just as for cross-streets, we initially compute the closest point on the uniformly scaled main route to the original position of the building and then construct an constraint region interval around this point. If more than one building is to be placed along the same road, we limit the lengths of the constraint regions to ensure that they do not overlap so that the proper ordering of the buildings along the road is maintained. As shown in figure 8.3(b) we can then simply treat the landmark as a road label containing two graphic components; an icon and a text label.

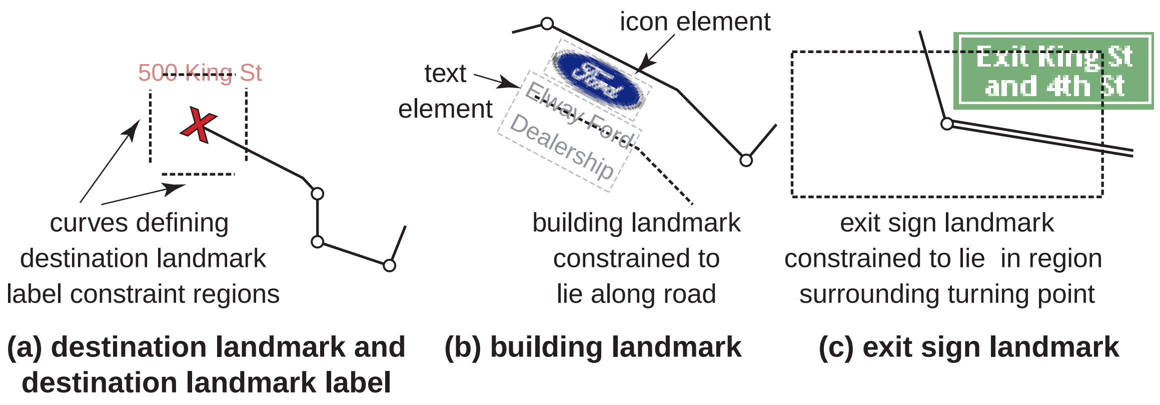

500 King St
curves defining destination landmark label constraint regions
(a) destination landmark and destination landmark label
icon element
text element
Elway Ford Dealership
building landmark constrained to lie along road
(b) building landmark
Exit King St and 4th St
exit sign landmark constrained to lie in region surrounding turning point
(c) exit sign landmark

Figure 8.3: Point landmark placement constraints. (a) Some landmarks such as this destination icon are constrained to lie on exactly one point. The layout algorithm does not have any flexibility in placing the destination icon. However, the label for the destination landmark has more flexible constraints. The center of the destination landmark label must lie on one of the dotted curves. (b) The building landmark is constrained to lie along the road as indicated by the dotted constraint curve. We treat the landmark just like a road label containing an icon element and a text element as its graphic components. (c) The center of the exit sign landmark is constrained to lie within the dotted bounding box centered at the target turning point..

The third category of point landmarks are those which must be placed near a turning point. Such landmarks can be considered as labeling a target turning point. Highway entrance and exit signs, stop signs and stop lights, are a common example of such landmarks. The main constraint on such signs is that they must appear close to the target turning point. As shown in figure 8.3(c), we build a constraint regions for such landmarks as a fairly large bounding box centered about the target turning point.

8.1.5.3 Position-Based Score

Both label layout and cross-street layout searches require the specification of a preferred position within the placement constraint region. For labels this preferred position is the point in the region closest to the center of the target object. For cross-streets the preferred position is the original intersection point. Both of these layout searches then compute a position-based score which penalizes a given label or cross-street position by a factor proportional to distance from the current position to the preferred position. We compute a similar position-based score for point landmarks.

The notion of a preferred position is well-specified for point landmarks that are tightly constrained to a single point as well as point landmarks that lie along a road. For point landmarks labeling a turning point however, we have two position-based requirements. The landmark must be located near its target turning point and it must be located as far away as possible from the other turning points so that the navigator properly associates the landmark at its target. Therefore we compute two position-based scores for such landmarks. First, we penalize a landmark by a score proportional to its distance from its target. Then we compute the distance from the landmark to every other turning point, and if the distance is less than the distance between the landmark and its target turning point we add a constant penalty to the score. We repeat this second test during the cleanup phase of landmark layout and remove any point landmark from the map whose distance to its own target is greater than its distance to another turning point.

8.2 Decoration

As shown in figure 8.4, the decoration stage of LineDrive is responsible for adding four kinds of graphic decorations to the map to enhance its usability; (1) light gray extensions are added to the ends of the roads, (2) a north orientation arrow is added to indicate the map orientation, (3) bullets are added at the turning points to indicate decision points on the route, and (4) the rendering style of the roads is used to indicate the type of the road and can also provide a subtle cue that the map is not drawn to scale. These decorations greatly strengthen the essential map information at the expense of very slightly increasing map clutter. We describe each of these decorations in turn.

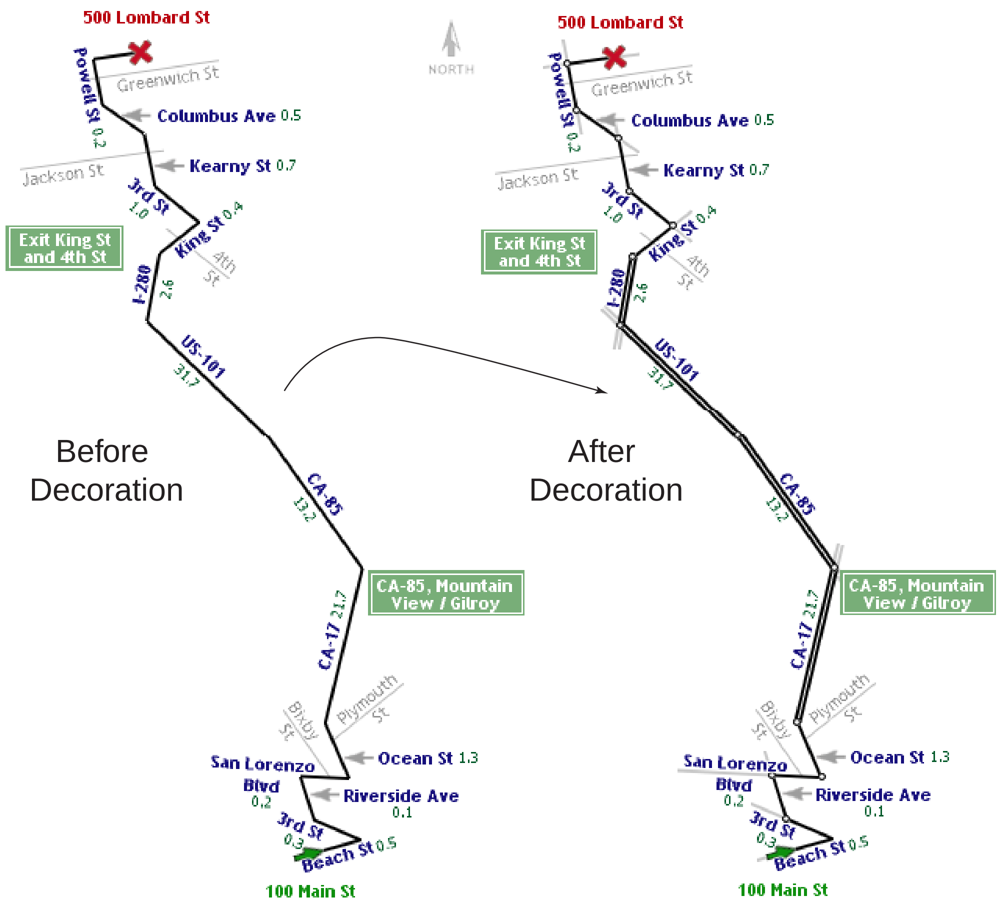


Figure 8.4: Before and after decoration. Four kinds of graphic decorations enhance the usability of the map. Extensions to the ends of roads, rendered in a light gray color, accentuate turning points and help associate road labels with their roads. The orientation arrow shows the overall orientation of the route. Bullets at turning points further accentuate turning points and show where each turn decision is required. The rendering style differentiates roads into three categories. Highways are rendered with double lines, standard residential roads are rendered with single lines and highway on- and off-ramps are rendered with single lines at half-thickness.

8.2.1 Road Extensions

Extensions on the ends of each road accentuate the turning points and help associate the road's label with the road. Before adding extensions, we look up the pair of roads at each turning point in the database to check if they continue beyond the turning point. If a road does extend, we set the length of the extension to a predefined minimum extension length. The extensions are drawn in the same light gray color as cross-streets to de-emphasize them from the main portion of the roads. The change in color helps emphasize the turning points.

The topology at each intersection between a pair of roads can be either a L-intersection, X-intersection or T-intersection. By including the road extensions we depict the topology of each turning point on the main route. This topological information can provide another cue for navigators to verify that they have reached the proper turning point as they follow a route.

If during label layout, the center of the road's label was placed directly above or below an extension, we grow the extension so that it passes completely over or under the label. Growing the extension in this manner helps form the proper association between the label and its target road.

Finally, we clip the extension to all other roads and cross-streets. Very short extensions can look confusing, so if the clipping causes the extension to become smaller than the minimum extension length we completely remove that extension from the map.

8.2.2 North Orientation Arrow

The north orientation arrow shows the overall route orientation with respect to global north and can make it easier for navigators to geographically place the route. To place the orientation arrow, we search the map image for an empty region large enough to hold the arrow. We accelerate the search by first building a fixed resolution occupancy grid over the map image and then only searching in empty cells of this grid. The search is ordered to first look for space in the four corners of the image and then search through the remaining image.

8.2.3 Bullets at Turning Points

Bullets, in the form of small circles, are drawn at each turning point to show exactly where each turn decision must be made. The bullets also help differentiate between roads that are headed in the same general direction. For example, the bullet between highways CA-85 and US-101 in figure 8.4 shows that the navigator must actively switch from one road to the other.

Another approach to accentuating turning points is to alternate the color of the roads at each turning point. For example, the road entering the turning point could be drawn in green while the road exiting the turning point could be drawn in black. However, compared to using small bullets at the turning points, this approach requires using an additional color-based encoding in the map. In our experiments we have found that the color change becomes a striking feature of the map, drawing attention away from the other map elements. The bullets are not as visually prominent and therefore a more appropriate choice for enhancing the decision points..

8.2.4 Rendering Style

The rendering style for each road is set according to the type of the road. Our database differentiates between three types of roads: limited access highways, highway ramps, and standard residential roads. Thus, we set the rendering style for each road based on its type. Limited access highways are drawn as double lines, while ramps are drawn at half the thickness of the standard roads.

We have also experimented with rendering LineDrive maps using a stroke-based, pen-and-ink style. Following the approach described by Markosian et al. [MKT$^+$97], we consider each road and cross-street as a piecewise linear curve and break the curves into sets of small equal length segments. We then perturb the endpoints of each segment by a small random amount and finally interpolate the perturbed points with a B-spline.

As shown in figure 8.5, the variations in the lines makes the map look more like a sketch than a precise computer-generated image. Strothotte et al. [SPR$^+$94] have shown that rendering style can influence how people interpret architectural drawings,

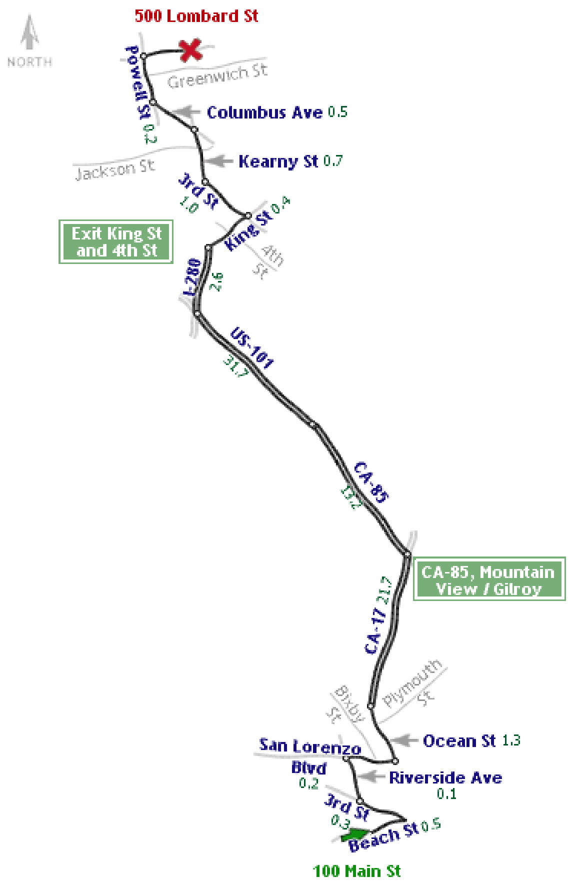


Figure 8.5: Map rendered in sketchy style. We use a stroke-based approach to add small variations to the roads and cross-streets. This gives the map the appearance of a sketch and can cue the navigator that the map is not drawn to scale.

and we believe a similar principle applies to route maps. The sketchy rendering style is a subtle cue that the map is not drawn to scale.

Chapter 9

Designing for Display Constraints

As computers have become ubiquitous, computer display devices[1] have taken a wide variety of forms. Display sizes range from wall-sized screens, covering tens of square feet down to wrist-watch screens covering under one square inch. Spatial resolution and color resolution vary from photographic quality color prints down to 20 text character monochrome displays found in pagers. When designing images to be displayed by a computer it is essential to consider the size and resolution constraints imposed by the display device.

The form-factor of the route map must be easy to carry and manipulate if it is to be used while traversing the route. Any display larger than a single sheet of paper is inconvenient and reduces usability. The clean, clutter-free composition of LineDrive maps is specifically designed to emphasize the essential route information within images that are at most about a third of a printed page in size. In fact, LineDrive maps are also well-suited to smaller displays as found on cell phones and personal digital assistants (PDAs). However, each display device imposes slightly different constraints on the LineDrive maps. In this chapter we describe how we select the image size and modify the graphic design of LineDrive maps to meet the constraints imposed by two categories of display devices; (1) web-sized displays designed to be viewed in a browser on standard computer monitors and printed on letter-size paper,

[1]We include printers and the associated printouts as display devices in addition to standard CRT based displays.

and (2) small screen displays such as those found on cell phones and PDAs.

We conclude the chapter with techniques for placing additional information such as text direction and detail maps, near the LineDrive maps for both categories of display devices. The additional information both reinforces the information depicted in the LineDrive maps, and presents the route in a slightly different manner. The navigator can quickly switch between the different depictions to form a better understanding of the route.

9.1 Image Size Selection

Recall that LineDrive designs route maps to fit within a pre-specified image size. As shown in our block diagram in figure 4.2, we have developed an image-size oracle to automatically select an image size for the map by computing the amount of space the map needs to most effectively communicate the essential route information. The image should provide enough space so that all the roads and turning points are clearly visible. However, if the image is too large, the map will contain large regions of empty space. Moreover, the exact dimensions of the route are not known until after the road layout stage of the algorithm. The challenge therefore, is to balance the image space requirements, without knowing the exact size of the route. Our approach is to choose the image size based on an estimate of the aspect ratio of the route.

The image size oracle must also consider the resolution and interface constraints imposed by the display device. Web pages designed for printing are usually about 650 pixels across by 1000 pixels long. As shown in figure 9.2 we typically place the LineDrive map in the top third of the page, oriented either horizontally or vertically. The orientation is dependent on the estimated aspect ratio of the route.

The resolution constraints on small screen displays are much tighter than for web pages. Cell phones using the Wireless Application Protocol (WAP) are limited to black and white images of $127X127$ pixels, while the most common version of the Palm PDAs can show images of $160X160$ with 16 levels of gray color resolution. Although these devices offer limited interaction capabilities they usually provide good support for vertical scrolling. We provide two methods for delivering LineDrive maps

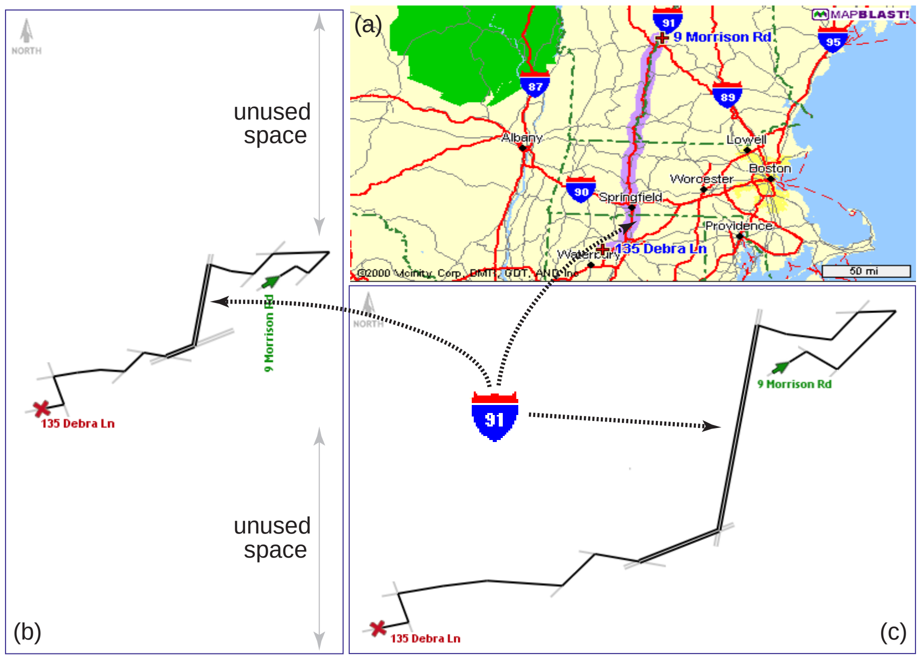
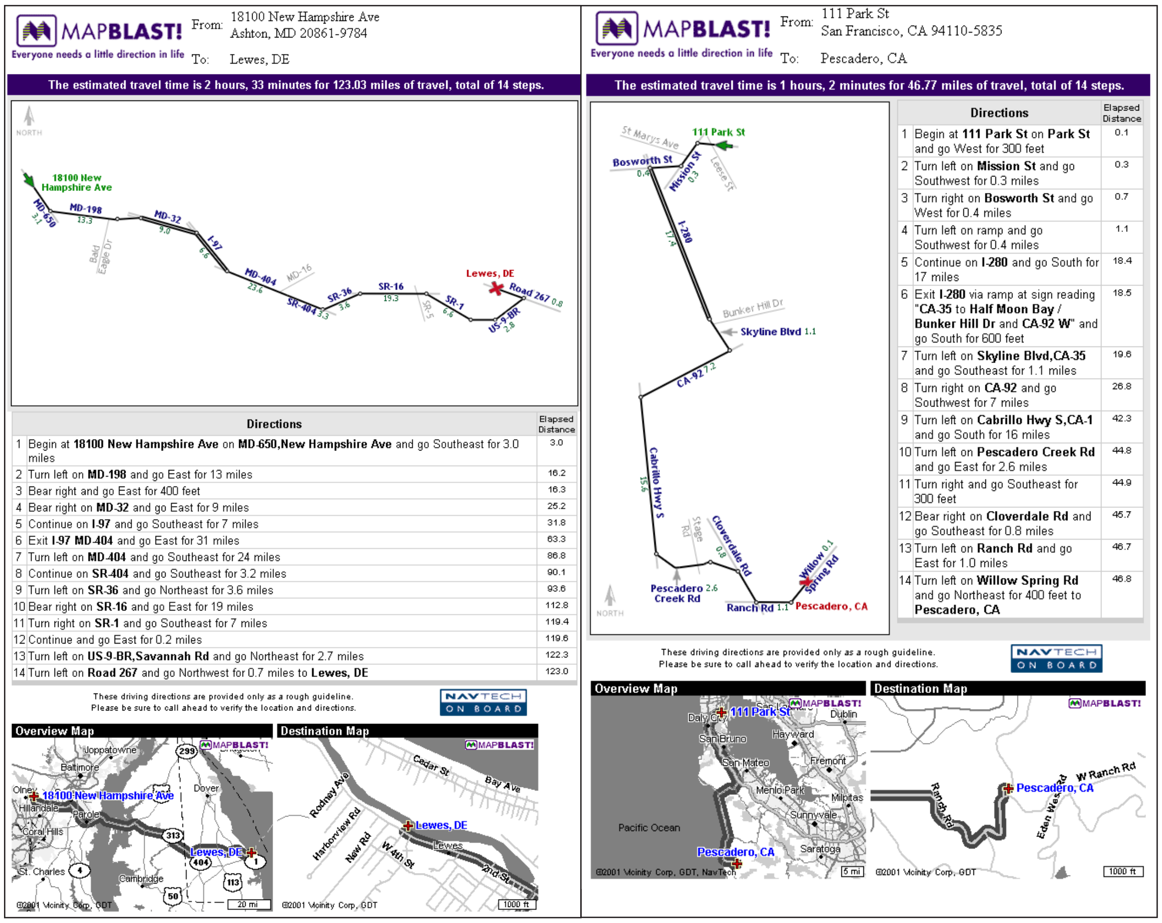


Figure 9.1: Estimating map aspect ratio. (a) The route contains one long north-south road (I-91) and many short east-west roads. (b) If the image size is selected based on the original north-south aspect ratio, the image is given more vertical space than horizontal space. The top and bottom of the image go unused because after growing all the short roads the aspect ratio of the map becomes much wider. (c) Computing the aspect ratio after growing the roads yields a horizontal image size and a more effective use of the space.

on such displays; we can split routes into multiple maps as shown in figure 9.3 and we can rotate the route so that navigator only needs to scroll vertically to see the entire route, as shown in figure 9.4.

9.1.1 Estimating Aspect Ratio

The aspect ratio of the image viewport can have a large affect on the layout of a route map. Consider a route map created for a predominantly north-south route that is designed to fit a wide aspect ratio viewport. All of the north-south roads would end up squashed while large regions of the image to the left and right of route would remain unused.



(a) Wide Aspect-Ratio

From: 18100 New Hampshire Ave
Ashton, MD 20861-9784
To: Lewes, DE

The estimated travel time is 2 hours, 33 minutes for 123.03 miles of travel, total of 14 steps.

	Directions	Elapsed Distance
1	Begin at **18100 New Hampshire Ave** on **MD-650,New Hampshire Ave** and go Southeast for 3.0 miles	3.0
2	Turn left on **MD-198** and go East for 13 miles	16.2
3	Bear right and go East for 400 feet	16.3
4	Bear right on **MD-32** and go East for 9 miles	25.2
5	Continue on **I-97** and go Southeast for 7 miles	31.8
6	Exit **I-97 MD-404** and go East for 31 miles	63.3
7	Turn left on **MD-404** and go Southeast for 24 miles	86.8
8	Continue on **SR-404** and go Southeast for 3.2 miles	90.1
9	Turn left on **SR-36** and go Northeast for 3.6 miles	93.6
10	Bear right on **SR-16** and go East for 19 miles	112.8
11	Turn right on **SR-1** and go Southeast for 7 miles	119.4
12	Continue and go East for 0.2 miles	119.6
13	Turn left on **US-9-BR,Savannah Rd** and go Northeast for 2.7 miles	122.3
14	Turn left on **Road 267** and go Northwest for 0.7 miles to **Lewes, DE**	123.0

These driving directions are provided only as a rough guideline.
Please be sure to call ahead to verify the location and directions.

©2001 Vicinity Corp, GDT

(b) Tall Aspect-Ratio

From: 111 Park St
San Francisco, CA 94110-5835
To: Pescadero, CA

The estimated travel time is 1 hours, 2 minutes for 46.77 miles of travel, total of 14 steps.

	Directions	Elapsed Distance
1	Begin at **111 Park St** on **Park St** and go West for 300 feet	0.1
2	Turn left on **Mission St** and go Southwest for 0.3 miles	0.3
3	Turn right on **Bosworth St** and go West for 0.4 miles	0.7
4	Turn left on ramp and go Southwest for 0.4 miles	1.1
5	Continue on **I-280** and go South for 17 miles	18.4
6	Exit **I-280** via ramp at sign reading "**CA-35** to **Half Moon Bay / Bunker Hill Dr** and **CA-92 W**" and go South for 600 feet	18.5
7	Turn left on **Skyline Blvd,CA-35** and go Southeast for 1.1 miles	19.6
8	Turn right on **CA-92** and go Southwest for 7 miles	26.8
9	Turn left on **Cabrillo Hwy S,CA-1** and go South for 16 miles	42.3
10	Turn left on **Pescadero Creek Rd** and go East for 2.6 miles	44.8
11	Turn right and go Southeast for 300 feet	44.9
12	Bear right on **Cloverdale Rd** and go Southeast for 0.8 miles	45.7
13	Turn left on **Ranch Rd** and go East for 1.0 miles	46.7
14	Turn left on **Willow Spring Rd** and go Northeast for 400 feet to **Pescadero, CA**	46.8

These driving directions are provided only as a rough guideline.
Please be sure to call ahead to verify the location and directions.

©2001 Vicinity Corp, GDT, NavTech

Figure 9.2: LineDrive maps designed for the Web. The size of the LineDrive map image is dependent on the estimated aspect ratio of the route. (a) A wide aspect ratio route is given a wide image that covers the top third of the web page. (b) A tall aspect ratio route is given a tall image that covers the top left part of the web page. Text directions as well as overview and detail destination maps provide additional route information.

A better approach is to choose the viewport size based on the aspect ratio of the route. However, simply using the aspect ratio of the original uniformly scaled route does not always produce the desired result. Suppose, as in figure 9.1, the original route contains many east-west roads near its origin and destination, with one extremely long north-south road in between. Although the original aspect ratio for the route is north-south, after growing the short roads in our road layout, the aspect ratio of the route changes substantially. To estimate the aspect ratio of our final map before performing road layout, we initially fit all the roads at their original lengths to a large square viewport. We then grow all the short roads to their minimum pixel length and finally compute the aspect ratio of this new map, thus generating a more realistic estimate.

The estimated route aspect ratio divides our routes into two categories, those that run predominantly east-west and are therefore wider than they are tall and those that run predominantly north-south and are therefore taller than they are wide. For wide aspect ratio routes we use a fixed image size of $650X350$ pixels, to cover about a third of the page. The horizontal length of the maps is limited to 650 pixels so that the LineDrive maps will print on letter-size paper without any clipping. An example of a wide aspect ratio map designed for the web is shown in figure 9.2(a).

For tall aspect ratio routes we use a base image size of $350X500$ again covering roughly a third of the page. However, letter-size paper does not limit the vertical resolution of our maps and we can therefore extend the vertical image size of tall routes as needed up to a limit of 800 pixels. Thus, we start with a base vertical image resolution of 500 pixels for maps with 10 or less steps and add 20 pixels for each step thereafter. An example of a tall aspect ratio map designed for the web is shown in figure 9.2(b).

9.1.2 Splitting Route Into Multiple Maps

Since LineDrive scales each road in the route individually, space required to generate an overlap-free map is generally related to the number of steps in the route. In the United States most routes, including cross-country routes, contain less than 30 steps.

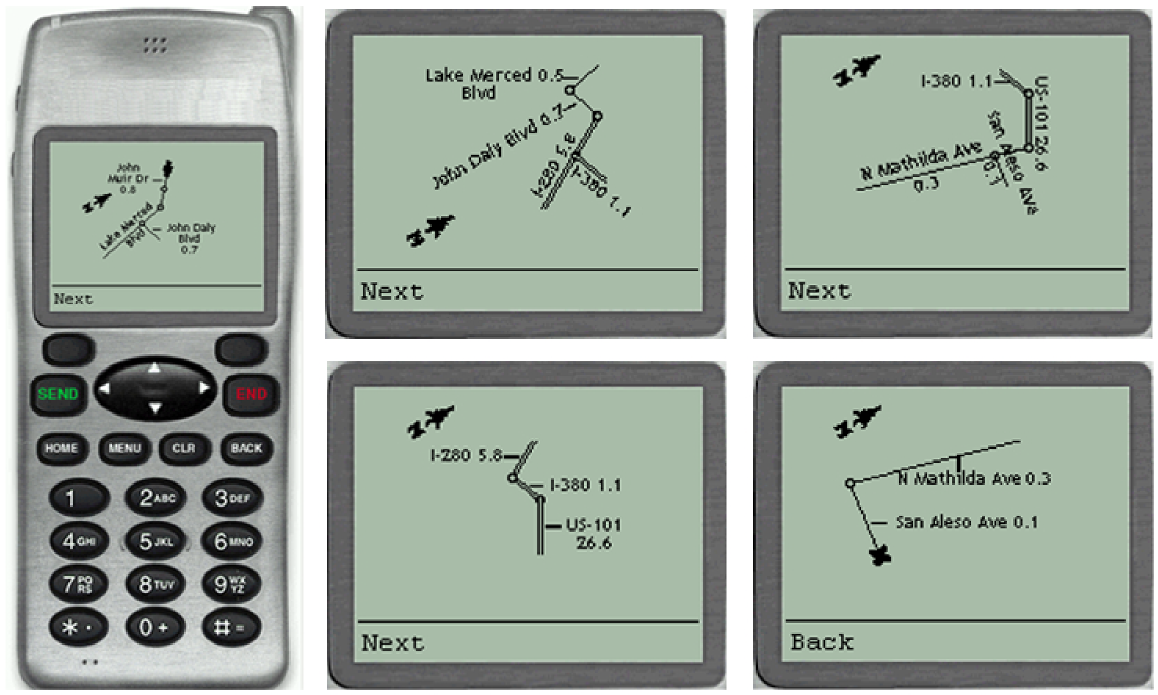


Figure 9.3: LineDrive map on a WAP cell phone. The small-screen display requires forces the route to be split across multiple images. One bit color resolution forces the graphic design of the maps to be simplified as well.

In European countries such as the United Kingdom, however, the road networks are older with fewer highways and therefore routes containing over 75 steps are common. Even using the optimization techniques in LineDrive, it is not always possible to produce an overlap-free layout within the given image space for such routes. Therefore, when designing maps for web-sized displays, if the route contains more than 30 steps LineDrive splits the route into roughly equal sized segments of roads. For example, if a route contains 45 steps, LineDrive will split the route into two segments, the first containing 23 steps, and the second containing 22 steps. A web-sized display is then created for each segment of the route. This approach ensures that each segment of the route is given enough space to produce a clear and legible map.

With small-screen devices such as cell phones and PDAs only a few steps of the route will fit in a single image. Splitting the route into multiple segments offers a simple solution to the problem. As shown in figure 9.3 for WAP based cell phones with extremely limited resolution we place at most two turning points in each image. Each map is designed to the full screen resolution of $127X127$ pixels.

The main drawback of this approach is that the entire route is never encapsulated in a single map and is instead split over two or more images. Just as with stripmaps the navigator must flip between multiple maps and to form a mental representation for the route. However, this solution is preferable to generating a crowded, unreadable map containing all the steps.

9.1.3 Verticalization

Another approach to displaying routes on small screen devices is to create a larger map image that can be scrolled. Since many cell phones and PDAs provide good controls for scrolling vertically but not horizontally the best interface would be one which allows the entire map to be seen by scrolling the image only vertically. In such situations, the image size is constrained only in the horizontal direction. Luckily, most routes have some predominant orientation. We find the predominant orientation by fitting a tight, oriented bounding box [GLM96] to the route after growing all the short roads just as we did for the aspect ratio computation. We can then fit the map to our horizontally constrained image by rotating the entire route so that the largest extent of the map is aligned with the vertical axis of the viewport. This approach provides extra space in the direction the route needs it most. As shown in figure 9.4, the orientation arrow helps indicate that the map has been rotated.

A common cartographic convention is that the north orientation arrow should align as closely as possible with the vertical axis of the viewport. Thus, we choose the rotation angle, either clockwise or counter-clockwise, which ensures that north arrow points in the upward semi-circle of directions. The rotation angle is bounded between ± 90.0 degrees and although the north arrow may not be aligned with the vertical axis of the viewport after the rotation it usually has a strong component in the vertical direction. In the worst case the north arrow will point to the left or to the right after the rotation. It will never point downwards.

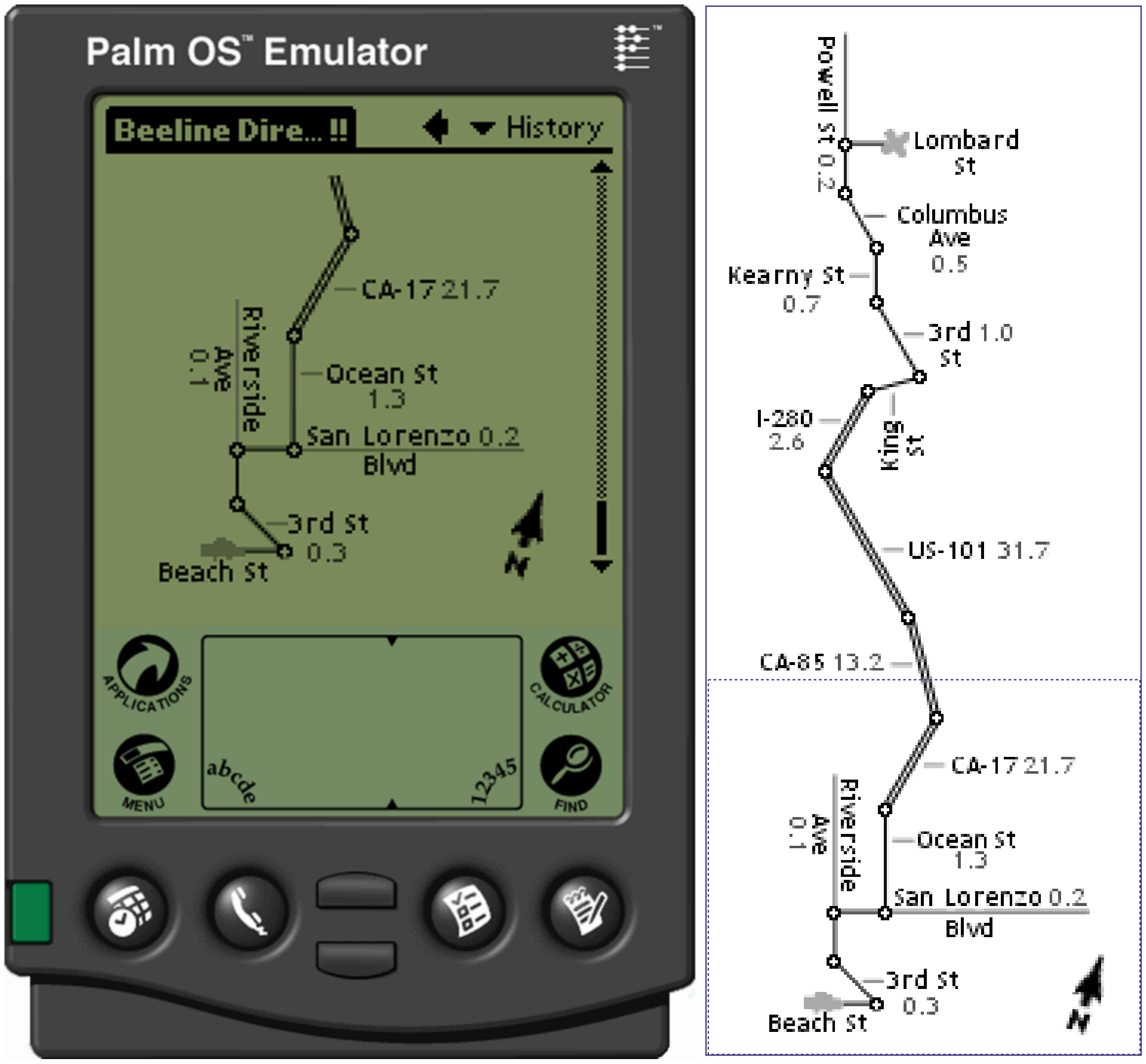


Figure 9.4: LineDrive map on a Palm PDA. The route is rotated so that it fits the horizontally constrained image size of the PDA. The vertical dimension is unconstrained and users can scroll up to see the remainder of the route.

Once the map has been verticalized, we can compute a vertical resolution for the image based on the number of steps in the route. For PDAs we have empirically found that providing a vertical resolution of 200 pixels for maps with less than 10 steps, and adding 10 pixels for each step thereafter, works well.

9.2 Simplifying Graphic Design for Small-Screens

To produce clear, readable, routes maps for small-screen displays, which generally lack both spatial and color resolution, we simplify the graphic design of the maps in several ways. Antialiasing does not work well with limited color resolution. On the common Palm PDA displays containing four bits of color depth, antialiased lines and text that are not oriented either horizontally or vertically appear somewhat blurry. With a single bit of color depth as found on WAP based cell phones, it is impossible to perform any antialiasing and text oriented at an angle can be very difficult to read. Since legible road labels are essential for following the route we modify the label layout stage of LineDrive to favor label placements that are oriented vertically or horizontally. We simply add a penalty to the score of a label if it is oriented at an angle.

Under the tight spatial resolution constraints, secondary route information clutters the map more than it helps navigation. Moreover the reduced color resolution makes is difficult to to de-emphasize secondary information. Cross-streets and road extensions that are colored light gray in the web-sized maps are almost the same color as the main roads in the small-screen maps. For these reasons we eliminate all the context information from the maps designed for small screens, by skipping the context layout stage of the system. In addition, in the decoration stage we only add extensions to roads where they are required to help associate road labels with their target roads.

9.3 Designing the Complete Page

While LineDrive maps contain all the information required to traverse a given route, additional information such as text directions and detail maps can make navigation easier. These additional information sources reinforce and extend the information depicted in the LineDrive map. However, the additional information is most useful when it is placed near the LineDrive map so that the navigator can easily take in all the information with a single glance. The page designer, shown in the block diagram figure 4.2, is responsible for collecting the additional route information and then composing all the route information including the LineDrive map to form a complete description of the route.

For both web-sized and small-screen displays we complement the LineDrive map with the text directions. LineDrive maps present the route from an allocentric third-person, birds-eye perspective, while the computer-generated text directions often present more of a first-person, route perspective. Taylor and Tversky [TT92a, TT92b, TT96] have shown that human-generated verbal directions usually mix these two perspectives. The LineDrive web page presents both perspectives at once so that navigators can choose the one they prefer or use a combination of both. We have often found it useful to be able to quickly look back and forth between the text directions and the LineDrive map in order to verify the action to take at each turning point (i.e. turn left or turn right).

As shown in figure 9.2 for web-sized displays the text directions are placed either below or to the right of the LineDrive map, depending on the aspect ratio of the map. For small-screen PDAs the text directions are placed immediately below the LineDrive map. With cell-phones, the text directions follow the set of LineDrive map images.

One difficulty with using only LineDrive maps combined with text directions is that they provide little detail outside of the main route. If the navigator accidentally strays from the route, it can be difficult to find a way back onto it. This can be especially problematic near the destination of the route where the navigator is less likely to be familiar with the area and may need to stray from the route in order to find parking. Similarly, LineDrive maps can be difficult to place within a larger geographic context because they provide very little geographic context beyond the outside of the channel immediately surrounding the route. We address these problems for the web-sized displays by providing two standard computer-generated maps below the text direction. An overview map shows the entire route at constant scale so the large scale geographic features near the route are likely to be visible and provide some orientation. A detail destination map provides extra information about the road network surrounding the destination. These maps are sized vertically to try to ensure that the complete page including LineDrive map and text fit within a single printed sheet of paper. We do not provide the overview and destination maps for small-screen displays because the standard computer-generated maps contain very little information when constrained to the color resolution of the these displays.

Chapter 10

Results

The intuition behind LineDrive is that, for local trips within a semi-familiar region, hand-drawn route maps are much easier to use while traversing the route than standard road maps drawn to-scale. As we developed the LineDrive system, we periodically tested the validity of our intuition by taking user surveys. The surveys were designed to tell us how people typically use Web-based route mapping services and whether or not our prototype LineDrive map designs would be more useful than standard computer-generated route maps. The user feedback led to many refinements in the graphic design of the LineDrive maps, as can be seen by comparing maps produced by our early prototype system (see figure 4.1) to maps produced by our current system (see figure 4.3). Through this iterative approach we not only validated our initial intuition, but we gathered many insights into how we could improve the usability of the LineDrive route maps as well. While the insights themselves have been described in the earlier chapters of this dissertation, we we begin this chapter by presenting the results from these early surveys and connecting the survey results with the insights we learned.

We completed this iterative, user-centered design for LineDrive in October 2000, and publicly released a beta version of the system through `www.mapblast.com` at that time. Examples of several route maps generated using the current version are presented in figures 10.1- 10.3 as well as figures 9.2- 9.4 of the previous chapter. We tested the performance of the LineDrive system in two ways: (1) by computing

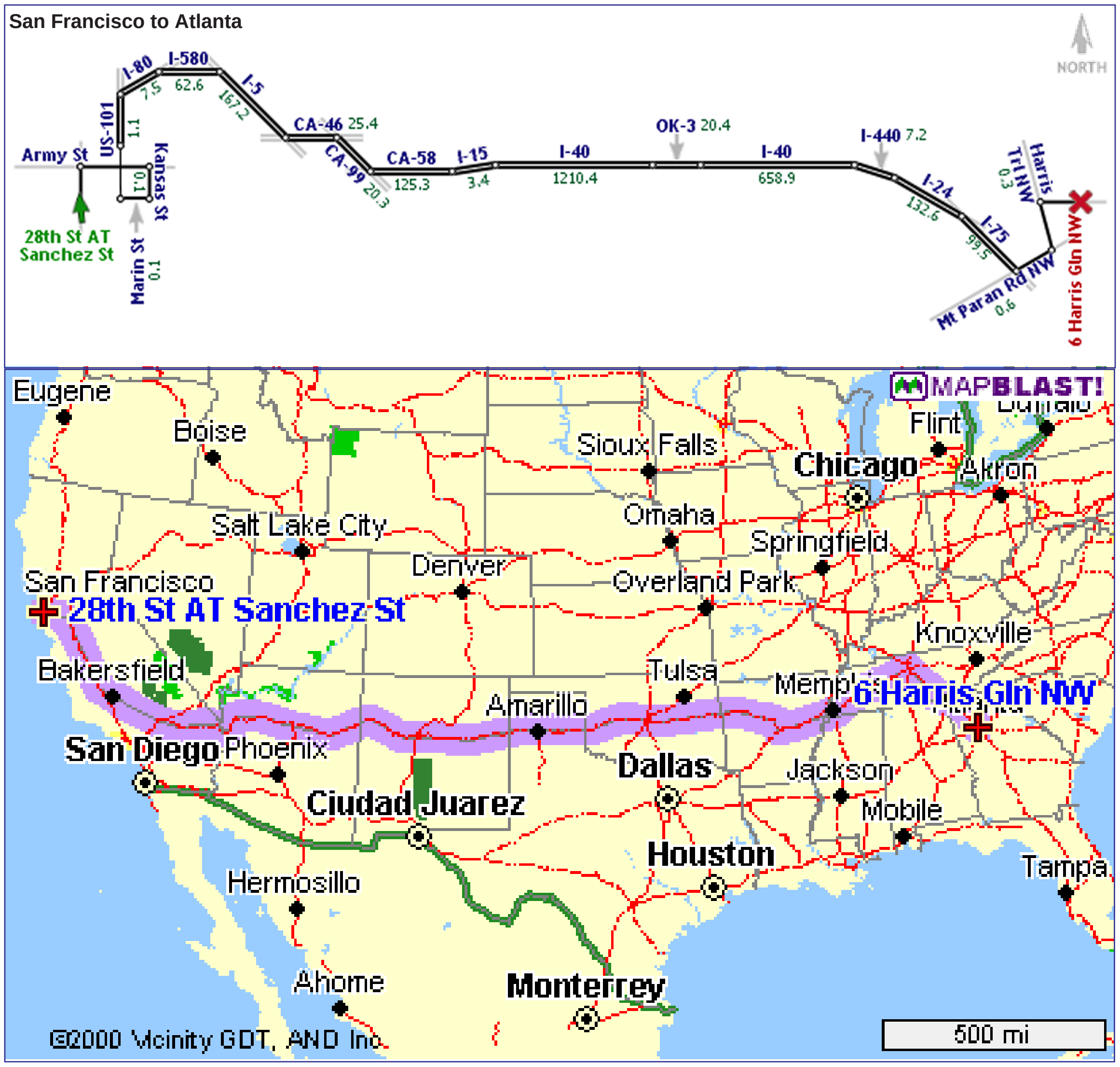


Figure 10.1: LineDrive map from San Francisco to Atlanta. Unlike in the standard overview map shown below, the non-uniform scaling in the LineDrive map allows all roads to be visible in this cross-country route. Since the ramp between Marin St. and US-101 intersects Army Street (actually passes above Army) it is not dropped from the map and proper intersection topology is maintained.

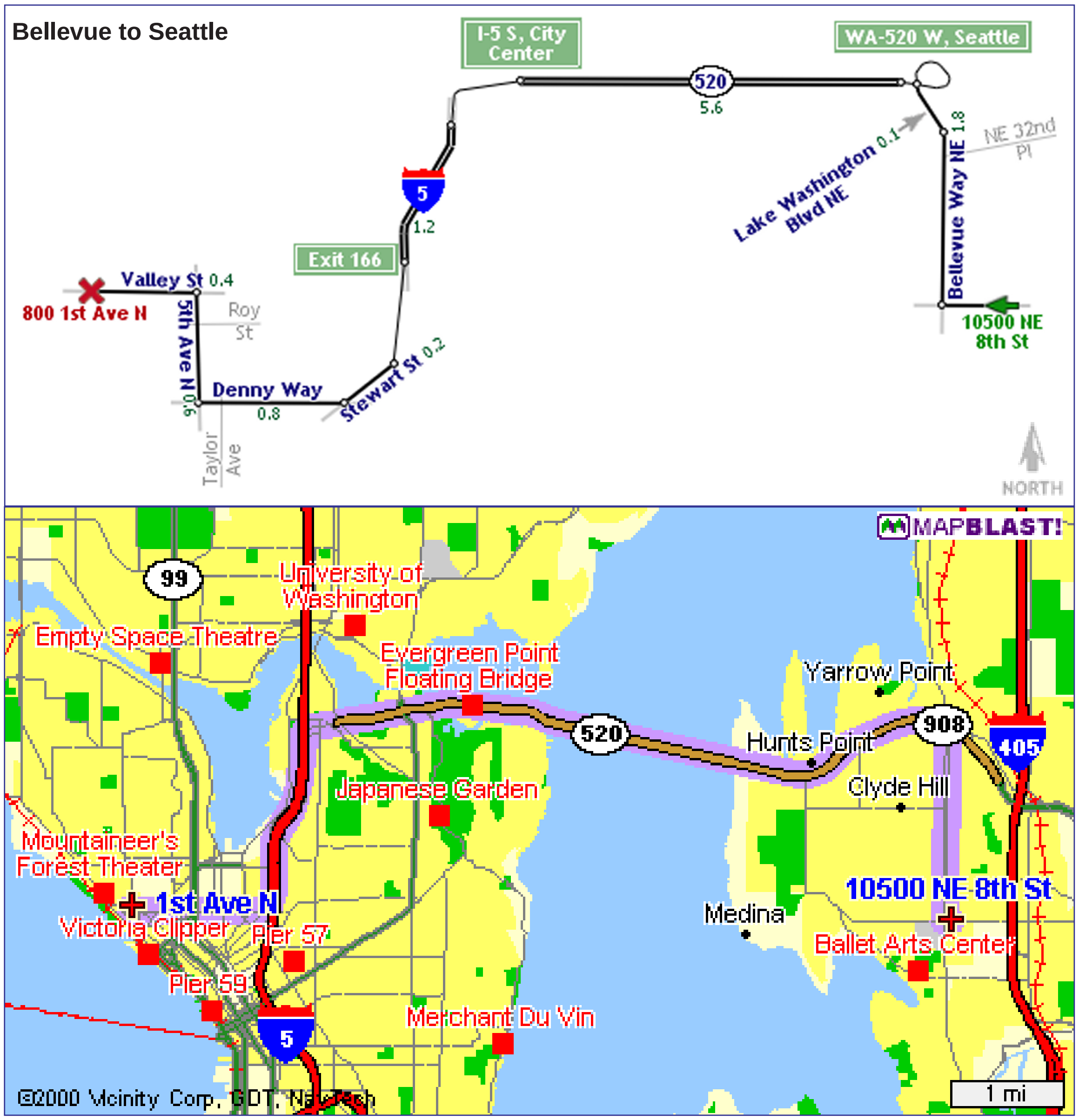


Figure 10.2: LineDrive map from Bellevue to Seattle. All ramps are maintained in this relatively short route from Bellevue to Seattle. Road shape is retained at both ends of I-5 in order to maintain a consistent turn angle with the adjacent ramps. The exit signs provide important context information for entering and exiting the highways. The highways are labeled using the *highway-shield* labeling style which helps differentiate the interstate, state and local highways from residential roads.

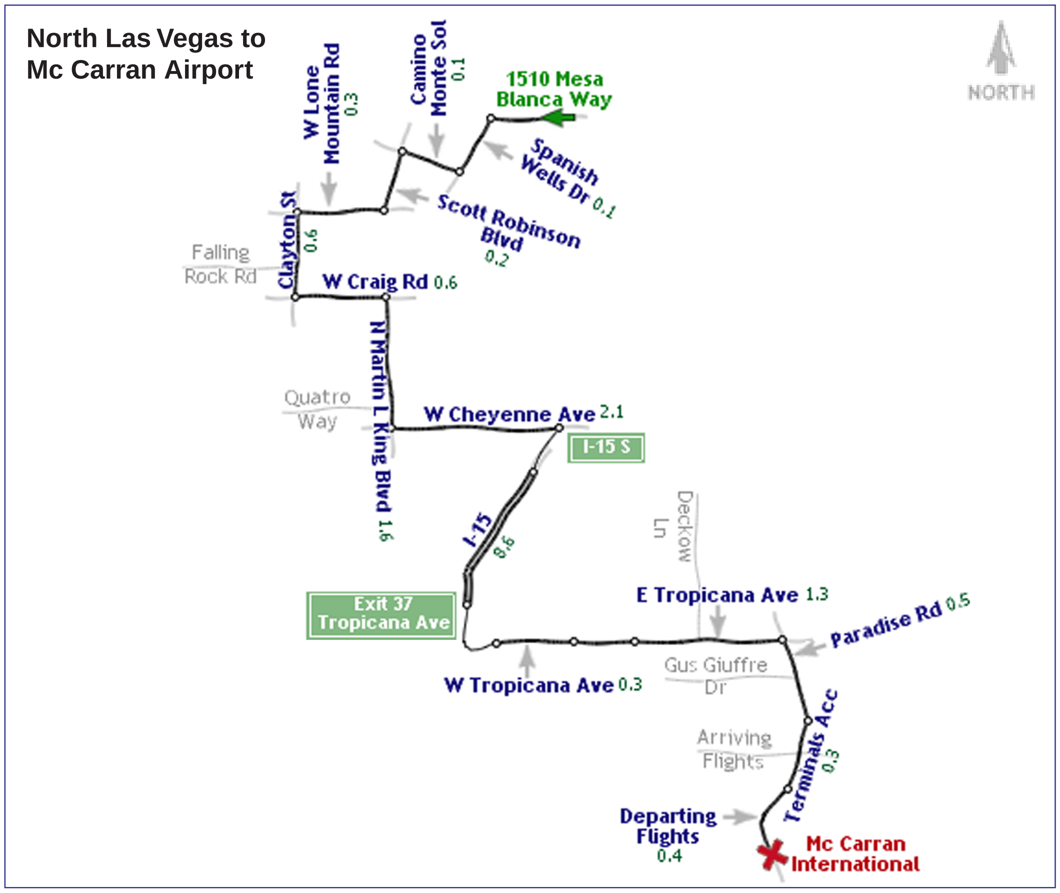


Figure 10.3: LineDrive map from North Las Vegas to McCarran Airport. Cross-streets provide context and aid navigation in this route. The sketchy rendering style in this map is a subtle cue that the map is not drawn to scale.

detailed statistics on a test suite of 7727 routes and (2) by collecting extensive user feedback on the beta version of LineDrive. We describe both of these performance tests in the last two sections of this chapter.

10.1 Early User Surveys

We conducted two user surveys during the early design of LineDrive. The first survey, taken in April 2000, was developed to provide insight into how people use standard

Web-based driving direction services such as MapQuest, MapBlast, Expedia etc. As shown in table 10.1, we asked Stanford business school students to voluntarily fill out a web-based questionnaire about how they typically use such online mapping services. We received 122 responses. Although the respondents were self-selected and as business school students perhaps biased in favor of technology, we believe that the responses do provide insight into how a large segment of users use online driving directions.

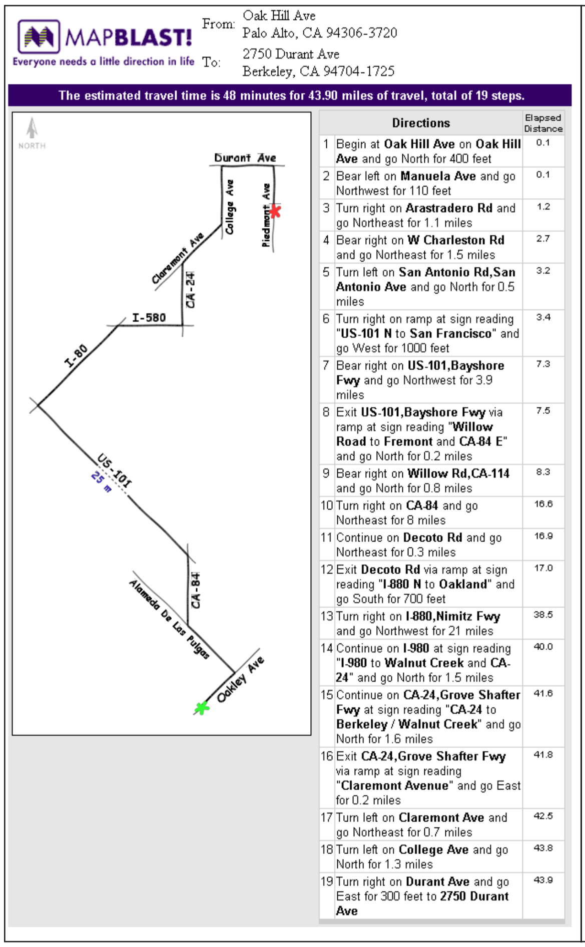
This survey validated several of our hypotheses regarding how people use online driving directions. Responses to the first question show that people who use online driving directions tend to use them fairly often. About 50% of our respondents use them either all the time or pretty often. Almost all people who generate driving directions print them out to take with them on the trip. The response to the second question shows that driving directions are most commonly used while traversing the route. As the responses to our third question show, over three quarters of all trips for which people generate driving directions are within the navigator's own greater metropolitan area. This question validated our intuition that people generally use online driving directions for local trips.

The first survey also shows that many users are dissatisfied with the standard computer-generated route maps. Just over 70% of respondents said that they rely either exclusively or primarily on the text directions, ignoring the maps. The last survey question shows several reasons for the reliance on the text directions. According to 50.1% of respondents, the overview maps are difficult to use, while 64.8% said that the step-by-step focus maps are difficult to use. One person commented that "The [standard] maps would be useful if they were easier to read. The current maps that come with directions never seem to be at the right scale," while another wrote, "I find the [standard] maps to be microscopic and hard to read."

The text directions usually provide a good description of each turning point on the route. Therefore, one of the main functions of the overview and step-by-step maps is to provide additional context for the route, not found in the text direction. However, 42.6% of respondents said that recovering from a wrong turn is difficult with standard online directions. The form-factor of standard online driving directions is

First User Survey		(122 responses)
How often do you use online driving direction?		
15	12.3%	All the time.
44	36.1%	Pretty often.
53	43.4%	Occasionally.
10	8.2%	Rarely.
How often do you print the directions to take with you on your journey?		
95	77.9%	Always.
21	17.2%	Most of the time.
6	4.9%	Half the time.
0	0.0%	Occasionally.
0	0.0%	Never.
About what percent of the time do you use driving directions within your own greater metropolitan area (versus out-of-town)?		
	76.3%	Percentage of use in-town. (Average across respondents)
	24.7%	Percentage of use out-of-town. (Average across respondents)
Which of the following best describes your use of maps and text once you depart for your destination?		
19	15.6%	Use text only.
67	54.9%	Primarily use text.
18	14.8%	Use text and maps equally.
15	12.3%	Primarily use maps.
3	2.4%	Use maps only.
Driving directions usually include one *overview* map and a series of smaller *focus* maps that highlight specific turns. Once you depart for your destination, which of the following best describes your use of the maps in online directions?		
19	15.6%	I don't use the maps. I look at text directions exclusively.
60	49.2%	I use the overview map only.
19	15.6%	I mostly use the overview but sometimes look at focus maps.
12	9.8%	I use the overview map and focus maps equally.
12	9.8%	I mostly use the focus maps but sometimes look at the overview.
0	0.0%	I use the focus maps only.
Would you say that online driving directions suffer from any of these problems?		
61	50.0%	Print-outs are too long and cumbersome.
52	42.6%	Difficult to recover from a wrong turn.
62	50.1%	Overview map difficult to use or not helpful.
79	64.8%	Focus maps are difficult to use or not helpful.
48	39.3%	Directions are not reliable.

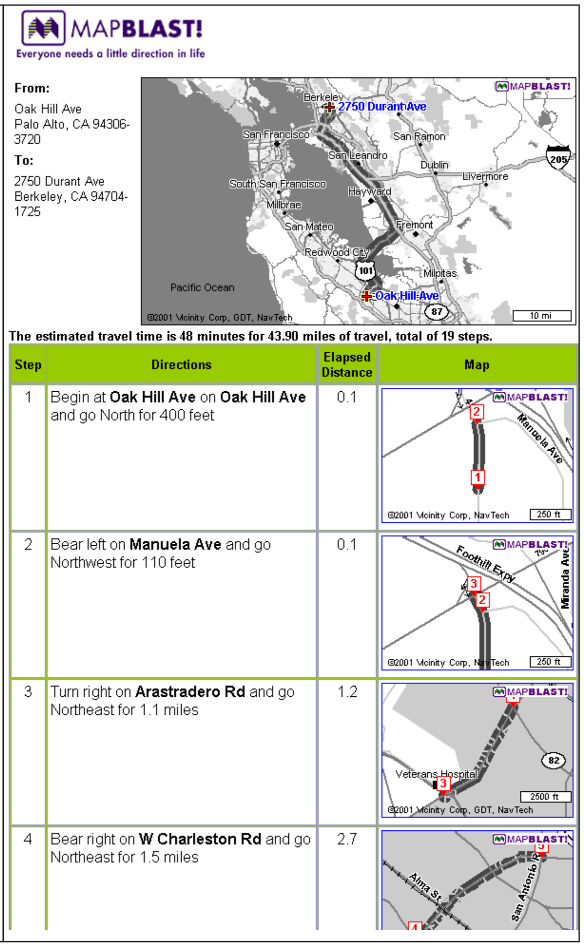
Table 10.1: First user survey. We received responses from 122 Stanford Business School students about their experiences with standard online driving directions.



From: Oak Hill Ave
Palo Alto, CA 94306-3720
To: 2750 Durant Ave
Berkeley, CA 94704-1725

The estimated travel time is 48 minutes for 43.90 miles of travel, total of 19 steps.

	Directions	Elapsed Distance
1	Begin at **Oak Hill Ave** on **Oak Hill Ave** and go North for 400 feet	0.1
2	Bear left on **Manuela Ave** and go Northwest for 110 feet	0.1
3	Turn right on **Arastradero Rd** and go Northeast for 1.1 miles	1.2
4	Bear right on **W Charleston Rd** and go Northeast for 1.5 miles	2.7
5	Turn left on **San Antonio Rd,San Antonio Ave** and go North for 0.5 miles	3.2
6	Turn right on ramp at sign reading "**US-101 N** to **San Francisco**" and go West for 1000 feet	3.4
7	Bear right on **US-101,Bayshore Fwy** and go Northwest for 3.9 miles	7.3
8	Exit **US-101,Bayshore Fwy** via ramp at sign reading "**Willow Road** to **Fremont** and **CA-84 E**" and go North for 0.2 miles	7.5
9	Bear right on **Willow Rd,CA-114** and go North for 0.8 miles	8.3
10	Turn right on **CA-84** and go Northeast for 8 miles	16.6
11	Continue on **Decoto Rd** and go Northeast for 0.3 miles	16.9
12	Exit **Decoto Rd** via ramp at sign reading "**I-880 N** to **Oakland**" and go South for 700 feet	17.0
13	Turn right on **I-880,Nimitz Fwy** and go Northwest for 21 miles	38.5
14	Continue on **I-980** at sign reading "**I-980** to **Walnut Creek** and **CA-24**" and go North for 1.5 miles	40.0
15	Continue on **CA-24,Grove Shafter Fwy** at sign reading "**CA-24** to **Berkeley / Walnut Creek**" and go North for 1.6 miles	41.6
16	Exit **CA-24,Grove Shafter Fwy** via ramp at sign reading "**Claremont Avenue**" and go East for 0.2 miles	41.8
17	Turn left on **Claremont Ave** and go Northeast for 0.7 miles	42.5
18	Turn left on **College Ave** and go North for 1.3 miles	43.8
19	Turn right on **Durant Ave** and go East for 300 feet to **2750 Durant Ave**	43.9



From:
Oak Hill Ave
Palo Alto, CA 94306-3720
To:
2750 Durant Ave
Berkeley, CA 94704-1725

The estimated travel time is 48 minutes for 43.90 miles of travel, total of 19 steps.

Step	Directions	Elapsed Distance	Map
1	Begin at **Oak Hill Ave** on **Oak Hill Ave** and go North for 400 feet	0.1	
2	Bear left on **Manuela Ave** and go Northwest for 110 feet	0.1	
3	Turn right on **Arastradero Rd** and go Northeast for 1.1 miles	1.2	
4	Bear right on **W Charleston Rd** and go Northeast for 1.5 miles	2.7	

(a) Prototype LineDrive Webpage (b) Standard MapBlast! Webpage

Figure 10.4: Prototype LineDrive vs. standard MapBlast! webpages. For the second survey we asked people to choose which webpage they preferred. (a) The prototype LineDrive webpage provides a map and text directions, (b) while the standard MapBlast! map provides an overview map and a step-by-step map with each line of the text directions. We only show the first printed page of the the MapBlast! webpage. The entire webpage extended over 4 printed pages. Note that although the origin-destination pair is the same for both webpages, the route is slightly different.

Second User Survey		(90 responses)
Which set of driving directions do you prefer?		
11	12.2%	Webpage containing standard computer-generated maps.
79	87.8%	Webpage containing prototype LineDrive map.

Table 10.2: Second user survey. We received responses from 90 Stanford Business School students about the set of directions they preferred.

also a problem for many users. In our survey, exactly 50% of respondents said that the printouts were usually too long and cumbersome.

The second survey, taken in July 2000, was developed to determine whether or not navigators would prefer a LineDrive map containing length, angle, and shape distortions to standard computer-generated route maps. We again presented the survey to Stanford Business school students and purposely chose the route origin to be at Stanford and the route destination to be in Berkeley so that the respondents would be likely to be familiar with the area around the route. Our survey first asked respondents to look at two different webpages. As shown in figure 10.4, the first was the standard webpage generated for the route and included step-by-step maps for each turn. The second webpage replaced the standard overview map with the prototype LineDrive map (we generated the page using photoshop) and did not show step-by-step maps. We then asked respondents to choose which map they preferred and why. As shown in table 10.2, 88% of respondents thought the LineDrive prototype map was preferable to the standard overview map.

The main comment echoed by many respondents was that the LineDrive maps are "Much clearer and easier to follow. Less clutter and more streamlined." This second survey helped validate our hypothesis that the length, angle and shape generalizations found in hand-drawn maps and imitated in LineDrive help generate simpler, cleaner maps that are easier to use than standard maps drawn to-scale. In their comments, respondents also provided several suggestions for improving the graphic design of the maps which we subsequently added into the system. A few people mentioned the idea of placing bullets at turning points to help emphasize decision points along the route. As mentioned in chapter 4, the prototype version of LineDrive considered each road as a straight line heading in one of the eight cardinal directions (N, S, E, W, NE, NW, SE, SW). Several people thought that adding more road shape, particularly for highway exit and entrance ramps would improve map usability. Finally, a few people mentioned that adding major cross-streets, particularly before turning points would be very helpful. Interestingly, many of the respondents mentioned that they really liked the clutter-free look of the maps and said that we should make sure not to add anything that would increase clutter.

One limitation of these surveys was that they did not allow the respondents to generate LineDrive maps for their own routes. While LineDrive might be able to generate one good map between Stanford and Berkeley it was unclear whether or not the system could scale to generate an effective map for any route. However, the results of the these two surveys gave us confidence that LineDrive maps could be far more useful than standard computer generated maps. Based on these survey results we convinced Vicinity Corporation, an online mapping company to give us access to their route mapping services and developed the current version of LineDrive.

10.2 System Performance

To test the performance of the current LineDrive system we first collected a test suite comprised of 7727 routes queried over one day at `www.mapblast.com`. The median route distance for the test suite is 52.5 miles and the median number of turning points is 13. We ran each route through the system twice, first generating a webpage size image at a fixed resolution of 600 x 400 and then generating a PDA size image with a fixed horizontal resolution of 160 and a variable vertical resolution. The running time is largely dependent on the number of objects (i.e. roads, labels, etc.) that must be placed in the map. The median run time for a single map on an 800 MHz Pentium III was 0.7 seconds for the first run and 0.8 seconds for the second run. Although the vast majority of maps are clustered around these median times, a few outliers containing over 100 roads took about 13 seconds to generate for the webpage size.

A small percentage of the LineDrive maps generated from the test suite of routes contained layout problems such as topological errors or label-label overlap. In many

Performance Statistics				(7727 routes)
	Web		PDA	
Median Time		0.7s		0.8s
Short Roads ($<$ 10 pixels)	415	5.4%	430	5.6%
False Intersections	25	0.3%	23	0.3%
Missing Intersections	15	0.2%	14	0.2%
Label-Label Overlaps	37	0.5%	289	3.7%
Label-Road Intersections	901	11.7%	2096	27.1%

Table 10.3: Performance statistics. Our test suite contains 7727 routes with a median of 13 turning points per route and a median distance of 52.5 miles. Every row except for median time indicates the number of maps containing at least one instance of the problem. For example, the short roads row presents the number of maps containing at least one road less than 10 pixels long.

cases, these problems were unavoidable because it is not always possible to make all roads large enough to be visible and simultaneously maintain the topology of the route. In a few cases, the problems could have been avoided but the randomized search did not converge to a near-optimal layout. The frequency of various layout problems for the 7727 route test suite are summarized in table 10.3.

The most significant problems that can arise in road layout are (1) that some roads may not be made large enough to be visible and clearly labeled or (2) that false or missing intersections may be introduced during the layout. Short roads, defined as less than 10 pixels in length, occurred in 5.3% of the webpage maps and 5.6% of the PDA maps. In most cases, the short roads could not be made longer either because there were a large number of roads all heading in the same direction or because lengthening the roads would have introduced a false or missing intersection. Although the PDA is horizontally constrained, the increase in the number of maps containing short roads is small because verticalization of these maps provides space for the short roads to grow. False and missing intersections occurred much less frequently than short roads and in all cases, avoiding the false or missing intersection would have required shrinking one or more roads to be extremely small. These percentages do reflect the priorities we gave each of the road layout constraints. Recall that as described in section 6.3.4,

User Feedback		(2242 responses)
Would you use LineDrive maps in the future?		
1246	55.6%	Yes, I would use them instead of standard driving directions.
976	43.5%	Yes, I would use them along with standard driving directions.
20	0.9%	No thanks, I'll stick with standard driving directions.
How would you rate this feature?		
1787	79.7%	It's a blast.
253	11.3%	Just fine.
202	9.0%	Needs some work ...

Table 10.4: User feedback. The beta version of LineDrive has been accessed over 150,000 times and we have received 2242 responses to the system.

preventing topological errors is our highest priority constraint, while growing short roads is given the next highest priority.

The main problems that can occur in label layout are (1) that a label will be placed overlapping another label, or (2) that a label may be placed overlapping a road or landmark. Less than 0.5% of webpage sized maps contained overlapping labels, while 3.7% of PDA sized maps contained label-label overlap. This increase is due to the fact that long labels are especially difficult to place without overlap on the horizontally constrained PDA. Although label-road overlap occurs in a significantly larger number of maps, such intersections are much less detrimental to the overall usability of the map than label-label overlap.

10.3 User Response

The beta version of LineDrive was available to the public from October, 2000 until March, 2001 and served over 150,000 maps in this period. Over 2200 users voluntarily filled out a feedback form describing their impressions of the LineDrive maps. Again, while the group of respondents was self-selected, it is unclear whether any resulting bias would be positive or negative. Despite the potential bias, we believe that the feedback provides valuable insight into users' reactions to the maps. As shown in table 10.4, the general response to the LineDrive maps was overwhelmingly positive. Less than one percent of respondents said they would rather use the standard computer-generated maps than the LineDrive maps.

Nearly half of the respondents said they would like to use LineDrive maps in conjunction with standard maps. As noted in the previous chapter, with the version of LineDrive designed for the Web, we provide two standard road maps drawn to-scale; a detail map of the destination and an overview map showing large-scale geographic context of the entire route. We believe that these maps in conjunction with the LineDrive map and text directions cover the needs of most users.

Long distance trips often require more context than LineDrive maps provide. While the cross-country map in figure 10.1(a) is a good stress-test showing that LineDrive can produce readable maps for routes containing many steps at vastly different scales, it is probably not the ideal map during such a long trip. Most navigators taking this trip would require a road atlas showing detailed local context along the way. LineDrive maps are designed for relatively short trips (i.e. under 100 miles) within a familiar region. Our experience is that most car-based trips fall within this range and the majority of people who use web-based mapping services generate directions to locations within their own greater metropolitan area.

One difficulty with using LineDrive maps alone is that they provide little detail outside of the main route. If the navigator accidentally strays from the route, it can be difficult to find a way back onto it. This can be especially problematic near the destination of the route where the navigator is less likely to be familiar with the area and may need to stray from the route in order to find parking. We address these problems on the website by providing a standard computer-generated map of the region near the destination of the route along with the LineDrive map.

About 9% of the respondents said the LineDrive system needs some work. However, most concerns were not with the LineDrive map, but instead with the particular route chosen by the route finding service. The beta version of LineDrive did not support cross-streets and local landmarks and the most common feature requests applicable to the maps were for the addition of cross-streets and exit signs. Based on the results of the beta test, LineDrive became the default map style for driving directions at `www.mapblast.com` in March 2001. This version supports cross-streets.

Chapter 11

Conclusions and Future Work

In this dissertation we have shown how to build a fully automated system for designing route maps that are as effective as hand-drawn route maps. To develop LineDrive we used a general two step approach that required first identifying the cognitive design principles used in the best hand-designed examples of route maps and then encoding those principles algorithmically. We review our contributions in section 11.1 and 11.2. In section 11.3 we describe several directions for future work.

11.1 Cognitive Design Principles for Route Maps

To formalize the cognitive design principles used in hand-drawn route maps we examined a variety of hand-drawn examples as well as prior research on the cognitive psychology research of wayfinding. From this analysis we identified three main cognitive design principles for hand-drawn route maps:

Cognitive Design Principle 1: People interpret the route in terms of paths and turns. Both verbal route directions and hand-drawn route maps are structured as a series of turns from one road to the next. The emphasis is on communicating the roads entering and exiting each turn and the turn direction (left or right) between them. The names of the roads connect the map to the physical world and the turn direction specifies the action to take at the turn. This turning point information is essential for following the route.

Cognitive Design Principle 2: People use landmarks for context and error recovery. Additional context information can facilitate navigation. Landmarks such as cross-streets, signage and buildings along the route provide consistency checks which navigators can use to verify that they are correctly following the route. Similarly, distances along each segment of the route can help navigators judge their progress. Larger scale area landmarks like nearby bodies of water and global properties of the route such as its overall shape can help navigators orient the route to the surrounding geography. However, context elements are not essential for following the route and are usually included in a hand-drawn route map only when they do not interfere with the primary turning point information.

Cognitive Design Principle 3: People mentally distort geometry. As long as the turns (the names of the exiting and entering roads and the turn direction between them) are present in the map, navigators usually do not need to know the exact road length, turning angle or road shape to follow the map. Hand-drawn maps often emphasize the turning points by distorting these geometric properties of the route. In fact, cognitive psychologists have shown that people's mental representations of routes also contain many such simplifying distortions. Moreover, the environment in many cases prevents the simplifications from causing errors; the exact degree of turn does not have to be indicated in the map because the shape of the physical intersection of the roads determines the turn.

11.2 LineDrive: A Fully Automated Route Map Design System

The key insight underlying our algorithmic design is that carefully instantiating these three cognitive design principles can dramatically improve the usability of a route map. We have incorporated all three into an automated route map design system called LineDrive and as a result LineDrive maps retain the ease of use found in hand-drawn route maps. In particular all the turning points are visible, the maps are completely clutter-free and they present all the information required to traverse the route within a compact and convenient form-factor.

LineDrive is set up as a search-based layout optimization algorithm. Given a set of map elements, roads, their labels, context elements like cross-streets and so on, LineDrive lays them out by choosing visual or retinal attributes like position, orientation and size for each element. LineDrive expresses each of our cognitive design principles as a numerical constraint functions and then uses search-based optimization techniques to automatically find a route map layout that best adheres to these principles. The difficult aspect of characterizing the route map layout problem as an optimization problem is developing efficient constraint functions that capture all the features of our cognitive design principles. This dissertation details our approach to algorithmically expressing these cognitive design principles.

The LineDrive system generates most route maps in about a second. It is currently the default map rendering engine for driving directions at www.mapblast.com. The system is serving about 250,000 maps per day. About 2200 users have answered a questionnaire; over 99 percent reported that they prefer LineDrive maps to standard computer-generated route maps. Based on these results we believe that LineDrive maps are far more effective than standard computer-generated route maps.

11.3 Future Work

There are several directions for future research. In this section we begin by presenting some approaches for extending LineDrive. We then describe how the techniques developed for LineDrive might be applied to automatically designing other types of maps. We conclude with a discussion of how our two-step approach for developing automated visualization design systems might apply to visualization domains other than cartographic maps.

11.3.1 Extensions to LineDrive

There are a number of directions in which the LineDrive system could be extended. Here we consider a few of these directions.

Optimizing visualizations for small and large displays: While we described some methods for designing LineDrive maps to the constraints of PDA's and cell phones in chapter 9, it would be useful to understand how to effectively present the maps on all kinds of display devices from cell phones to high-resolution, large-size display walls. Fully adapting LineDrive maps to small-screen devices will require accommodating their limited spatial and color resolution. We already reorient the maps for devices that provide vertical scrolling, and we split the route into smaller segments for extremely small-screen devices like cell phones. We could further examine how to adapt LineDrive maps to limited color resolution. For example, antialiasing text is difficult under these color resolution limitations, especially when text is drawn at an angle. To accommodate this limitation we would have to change the optimization constraints to prefer placing road labels in horizontal orientations to other orientations. Similarly we would like to adapt LineDrive maps to work with high-resolution devices like Stanford's Interactive Mural [GSW01] or the IBM High-Resolution Bertha display (3840 by 1560 with 204 dots per inch of 24-bit color)[IBM01]. With such displays we might adapt LineDrive maps to show more of the surrounding context for the route.

Adapting maps to the capabilities of the navigator: The LineDrive system currently does not have any model of the navigator and cannot adapt to the abilities of the navigator. It may be possible to extend LineDrive to account for users' abilities, such as their familiarity with the neighborhood of the route. A visualization designed for a user who is unfamiliar would contain more information to keep the user on course. It would also avoid abbreviations. For many routes the user will be more familiar with the region near the origin of the route than the destination of the route. In such cases we might give the user the option of removing the roads leading from the origin to the nearest highway in order to reduce clutter. Similarly we would like to investigate how spatial and verbal abilities of the viewer might affect the design of the maps. We currently orient the LineDrive maps so that North always is up. Certain experts like pilots prefer maps oriented so that the forward direction is up. These maps can reduce error because they do not require mental rotation to infer direction of turn. A north-up map might be better for planning and a forward-up map better for execution.

Dynamic route maps: LineDrive maps are currently designed as static visualizations, which navigators can print out to take with them on a trip. With handheld computers or in-car navigation systems containing Global Position Service (GPS) devices tracking the navigator, it would be possible to generate dynamically updating versions of the maps as the underlying data and goals of the route change. For example, as traffic patterns change, the map might update itself to show the fastest possible route. Dynamic route maps would be particularly useful if the navigator becomes lost. The map would update as necessary to help the navigator recover from the error. It would show either the best route back to the original route or a new route from the current location to the destination.

With constant location tracking through GPS the form of the route map visualization might change as well. A simple extension would be to place an animated icon in the map to show the navigator's current position in the route. A more complex interactive visualization could provide more zoomed-in detail around the current location of the navigator and gradually reduce detail in parts of the route that are far away from the navigator, to create a dynamic focus-plus-context view of the route. As we noted in chapter 3, most of the prior work in producing such views for maps, introduces severe distortions along the borders between the focus view and the context view. The distortions can make it difficult to understand how the two views fit together. It may be possible to develop a visualization that smoothly varies the level-of-detail from a detailed view at the current location of navigator a less detailed view as distance increases. Moreover such an interactive map should smoothly update the region as the navigator moves along the route.

Route maps for pedestrians and other modes of transport: LineDrive maps are currently designed to show driving directions. We would like to extend LineDrive to support other types of routes including pedestrian, hiking and biking routes. While these types of routes are similar to driving directions in many respects, they do change the requirements of the route visualization in several ways. For example, it is usually important to understand how the terrain, in particular hills, might affect the route. Such route maps should show the grade of each segment on the route. Similarly a driver is usually less interested in knowing the exact shape of each road, but a hiker or bicycler will often want to know how the paths along the route curve and switchback. For drivers the names of the roads serve as the major landmark and are the most important information marking a turning point. For pedestrian routes and hiking trails, the paths along the route may often be unnamed and therefore physical landmarks like buildings are much more important for marking turns.

11.3.2 Point Location Maps and Routes Through Buildings

While LineDrive was developed for designing long-distance route maps, we believe that many of the generalization techniques implemented in LineDrive may be applied to other types of cartographic maps. Here we consider two types of maps that are related to route maps: point location maps and short-distance 3D routes through buildings.

Point location maps: Point location maps, such as those in brochures, business cards and yellow pages show all the major routes to a point location such as an office building or restaurant. As shown in figure 11.1, hand-designed point location maps contain the types of scale distortions we see in LineDrive maps. Specifically, all the relevant roads, regardless of their length, are visible. But, point location maps are designed to show a 2D region around the point, while LineDrive maps are designed to show a single route, which is essentially a 1D curve. Therefore, the constraints designed for LineDrive maps may not transfer directly to point location maps. Point location maps will probably require stronger constraints to ensure that the 2D region around that point does not appear excessively stretched or distorted.

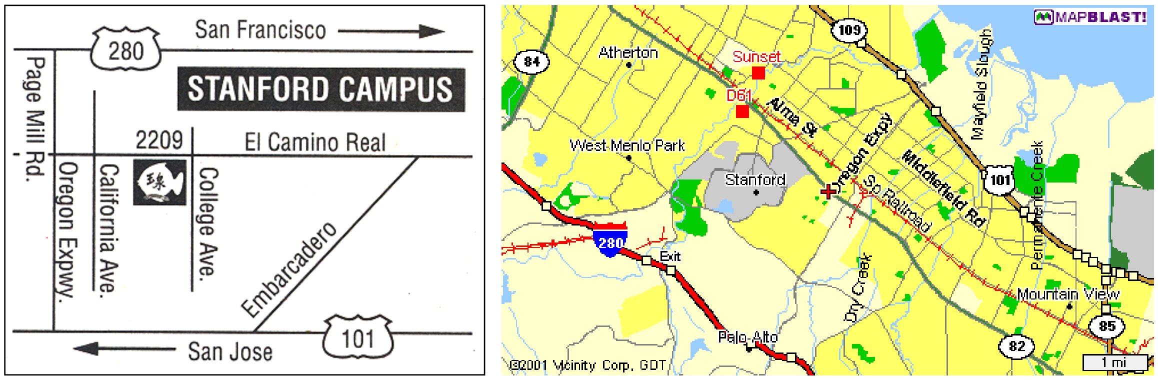


Figure 11.1: Point location map. (left) A hand-designed point location map showing routes from the major highways to the Jade Tree Restaurant at 2209 El Camino Real, Stanford California. (right) A computer-generated map for the same point location. The scale distortions in the point location map make it possible to see the highways 280 and 101 as well as the local roads like El Camino Real, California Ave. and College Ave, in the vicinity of the restaurant. The constant scale factor and clutter make it impossible to see these roads in the standard computer-generated map.

Point location maps can also be extended to include tourist maps that show all the attractions, restaurants or shops within a relatively small region, as well as the routes from the major highways to this region. Such tourist maps are often stylized and distorted to emphasize the major points of interest in the map. In some cases, tourist maps mix perspectives; they show the pattern of streets from above and the landmarks from a street point of view, so they are easy to recognize. In other cases, these maps are drawn with 3D perspective and/or 3D relief. A 3D view makes it possible to present a frontal view of the landmarks along the route and by cutting away rooftops it is possible to present the activities that are going on within a building.

Routes through buildings: We would also like to consider methods for visualizing routes within 3D environments such as buildings or subway stations. A typical route in such an environment might show emergency, fire escape evacuation routes, wheelchair accessible routes, or how to get to a particular office or platform. Currently such routes are shown by presenting each floor of the environment in plan view and

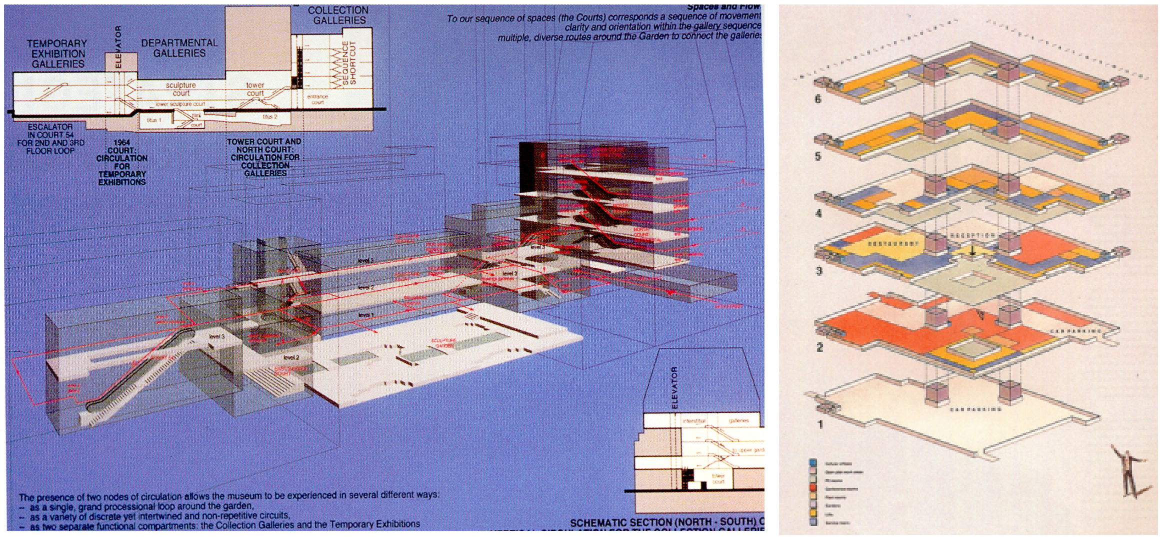

TEMPORARY EXHIBITION GALLERIES
ELEVATOR
DEPARTMENTAL GALLERIES
COLLECTION GALLERIES
SEQUENCE SHORTCUT
Spaces and Flow
To our sequence of spaces (the Courts) corresponds a sequence of movement
clarity and orientation within the gallery sequence
multiple, diverse routes around the Garden to connect the galleries
sculpture court
tower court
lower sculpture court
entrance court
titus 1
titus 2
court
ESCALATOR IN COURT 54 FOR 2ND AND 3RD FLOOR LOOP
1964 COURT: CIRCULATION FOR TEMPORARY EXHIBITIONS
TOWER COURT AND NORTH COURT: CIRCULATION FOR COLLECTION GALLERIES
level 3
level 2
level 1
interstitial
galleries
tower court
The presence of two nodes of circulation allows the museum to be experienced in several different ways:
-- as a single, grand processional loop around the garden,
-- as a variety of discrete yet intertwined and non-repetitive circuits,
-- as two separate functional compartments: the Collection Galleries and the Temporary Exhibitions
SCHEMATIC SECTION (NORTH - SOUTH)
6
5
4
3
2
1
RESTAURANT
RECEPTION
CAR PARKING

Figure 11.2: Methods for visualizing 3D routes.(left) The walls and floors are rendered semi-transparent so that the route shown in red remains visible. (right) The building is exploded vertically so that each floor is visible. A route drawn in this exploded view would be visible as it moved from floor to floor.

drawing the route separately in each view. The navigator must mentally connect the route at transfer points between the floors such as stairways or elevators. These visualizations force the viewer to mentally reconstruct the 3D route from a series of 2D slices.

The main challenge of visualizing routes within the complete 3D environment is dealing with the occlusions created by the walls and floors. Navigators need to see where walls and floors create physical barriers to the route, but they also need to be able to see the route itself. One approach, as shown in figure 11.2(left), may be to use translucency to show where the walls and floors exist but also allow the route to remain visible. Another approach as shown in figure 11.2(right) might be to use cutaways and exploded views to separate the building at each floor to allow each floor to be seen individually. Note that since 3D environments consist of localized spaces the types of scale distortions we use in LineDrive route maps are less useful and could be very confusing. However, we would like to investigate the types of distortions that are useful in such 3D environments.

11.3.3 Applications Beyond Cartographic Maps

The two-step approach we developed for LineDrive is general and can be applied to creating automated visualization design systems for a variety of different types of visualizations, including, 1) visualizations of instructions, directions and processes which show how a certain task is performed, and 2) visualizations of numerical or abstract data, such as graphs and charts. A number of research efforts have been directed toward automating the latter [Mac86, RM91, RLS$^+$96], but far fewer to the former [SF91].

Examples of the first category, visual instructions, are a standard part of our daily lives. Maps, training manuals, textbooks, architectural plans, scientific papers, and street signs all use visual diagrams to communicate instructions. We believe that it is possible to apply our two-step approach to build fully automated visualization design systems for creating visual instructions for a variety of domains. Here we consider a few domains outside of cartographic maps:

Mechanical assembly instructions: Assembly of a complex structure or machine is a common task. We define a mechanical assembly as a collection of 3D parts that are placed in specific spatial relationships with respect to one another. Assembly instructions are designed to show how the 3D parts must be arranged to form the complete object. Assembly is conceived as a hierarchy of actions on objects or object parts. At the coarse, higher level, actions are segmented by the objects or parts effected. At the finer level, actions are segmented by a sequence of refined actions on the same part. Congruent with this mental representation of assembly, diagrams of assembly show successive placements of parts or objects. Illustrators have developed a number of techniques such as cutaways, exploded view and insets to depict such instructions. While assembly instructions usually describe how to put together 3D objects like furniture, buildings, or machines, we can expand this domain in two ways. We can include other functions for the visualization including showing how the objects work, explaining their structure or repair and troubleshooting. We can also include other types of 3D objects such as biological specimens and geological samples.

Logistics plans and timelines: Large projects require some form of logistics planning to arrange when each subtask will be performed and who will work on each subtask. Such planning is especially important so that subtasks that are dependent on one another can be performed in the proper order. A logistics plan is conceived of as a dependency graph between the subtasks. Congruent with this mental representation of a logistics plan is a time series or timeline showing the timeframe within which each task will be completed. For example, a building construction project requires arranging when the various contractors such as electricians, plumbers, roofers, etc. should work on the project. In this case a visualization showing the various parts of the building being added to a blueprint view of the building, as they are finished can help the planner find dependencies among the contractors' subtasks. In planning the movement of forces in a battle a visualization showing the positions of the forces on a map would help the planner better understand the flow of the forces over time.

Architectural plans: In the early design stages a building is conceived at a number of different levels including room placement, furniture layout, movement through the building, 3D structure, lighting, etc. Congruent with these conceptions are design sketches such as bubble diagrams showing placements of rooms, plan views showing furniture layout and movement patterns, and axonometric and perspective views showing 3D structure and lighting. As the design progresses these drawings become more refined until they are fully formalized as blueprints for the building. While architectural plans usually deal with physical buildings we can expand this domain to include landscape architecture as well as neighborhood and city planning.

The cognitive design principles for instructional visualizations primarily depend on the domain to be conveyed and the goals of the task. For the research presented in this dissertation we focused our work on the domain of route maps. By considering how our approach might extend to other, vastly different domains we believe that it will be possible to uncover general techniques to automate the visualization design process.

Appendix A

Topological Error-Free Initial Road Layout

To create an initial road layout that is free from topological errors we must first determine where such errors may arise. Given a route and a viewport, we begin by uniformly scaling all the roads to fit within the viewport. We then search the route for intersections between non-consecutive roads on the route. These intersections are "true" intersections on the route and should be maintained in the final road layout. Suppose roads r_i and r_{i+m} intersect. We mark all the roads within the intersection interval formed by these two roads with an interval index $interval_a$. We can ensure that the true intersection is maintained in our initial layout by growing all the roads within the intersection interval by the same factor.

Yet, it is not enough to maintain true intersections. We must also ensure that our initial layout does not contain any false intersections. To determine roads that might generate false intersections, we grow all the short roads to be at least L_{min} pixels long and again mark the roads within each intersection interval. In this case some of these intersections may have been introduced by growing the short roads and therefore represent false intersections. However, we can ensure that our initial layout does not contain false intersections if we return to the uniformly scaled route and grow all the roads within each one of these false intersection intervals by the same factor. Note that since we computed these intersection intervals after growing short roads up to L_{min} pixels, this approach only ensures that false intersections are not created up to a scale factor C that grows the shortest road within the interval up to L_{min} pixels.

All roads within each intersection interval should be grown by the same scale factor to either maintain a true intersection or avoid a false intersection. We can optimize the process of growing roads by combining overlapping intersection intervals into a single interval. That is, if there is some road r_i that is part of $interval_a$ and $interval_b$ we make a new intersection interval $interval_{a,b}$ and place all roads r_j within either $interval_a$ or $interval_b$ into $interval_{a,b}$. In this manner we compute the union of overlapping intersection intervals and partition the route into disjoint intersection intervals.

To create the initial layout we return to the uniformly scaled route. Within each intersection interval we find the shortest road and if it is shorter than L_{min} we scale it up by a factor C so that it is L_{min} pixels long. We also scale all the other roads in the interval by C, thereby ensuring that topological errors are not generated. Any remaining road shorter than L_{min} must not lie within an intersection interval and can therefore be individually scaled up to L_{min} pixels.

After this growth stage all the roads are at least L_{min} pixels long and the route does not contain any topological errors. However the route may no longer fit within the viewport and therefore we finish the initial layout stage by rescaling the entire route by a uniform factor so that it fits inside the viewport.

Bibliography

[AAL$^+$00] David Anderson, Emily Anderson, Neal Lesh, Joe Marks, Brian Mirtich, David Ratajczak, and Kathy Ryall. Human-guided simple search. *Proceedings of AAAI 2000*, pages 209–216, August 2000.

[AK89] James Arvo and David Kirk. A survey of ray tracing acceleration techniques. In Andrew. Glassner, editor, *An Introduction to Ray Tracing.* Academic Press, 1989.

[All99] Gary L. Allen. Spatial abilities, cognitive maps, and wayfinding: Bases for individual differences in spatial cognition and behavior. In Reginald G. Golledge, editor, *Wayfinding Behavior*, pages 46–80. Johns Hopkins University Press, 1999.

[AS00] Maneesh Agrawala and Chris Stolte. A design and implementation for effective computer-generated route maps. In *AAAI Spring Symposium on Smart Graphics*, March 2000.

[AS01] Maneesh Agrawala and Chris Stolte. Rendering effective route maps: Improving usability through generalization. *Proceedings of SIGGRAPH 01*, August 2001.

[BB98] Greg J. Badros and Alan Borning. The cassowary linear arithmetic cosntraint solving algorithm: Interface and implementation. Technical Report UW-98-06-04, University of Washington, 1998.

[BD86] Alan Borning and Robert Duisberg. Constraint-based tools for building user interfaces. *ACM Transactions on Graphics*, 5(4):345–374, 1986.

[Bel95] Scott M. Bell. Cartographic presentation as an aid to spatial knowledge acquisition in unknown environments. Master's thesis, Department of Geography, University of Santa Barbara, 1995.
`http://www.geog.ucsb.edu/ bell/thesis/contents.html`.

[BETT99] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tolls. *Graph Drawing - Algorithms for the visualization of graphs.* Prentice Hall, 1999.

[BHR95] Franz J. Brandenburg, Michael Himsolt, and Christoph Rohrer. An Experimental Comparison of Force-Directed and Randomized Graph Dr awing Algorithms. In *Proceedings of Graph Drawing '95, Lecture Notes in Computer Science 1027*, pages 76–87. Springer-Verlag, 1995.

[BLR00] Thomas Barkowsky, Login J. Latecki, and Kai-Florian Richter. Schematizing maps: Simplification of geographic shape by discrete curve evolution. In C. Habel C. Freska, W. Brauer and K.F. Wender, editors, *Spatial Cognition II*, pages 41–53. Springer-Verlag, 2000.

[BM91] Barbara P. Buttenfield and Robert B. McMaster, editors. *Map Generalization: Making rules for knowledge representation.* Longman Scientific, 1991.

[Bor98] Jorge L. Borges. On the exactitude of science. In *Collected Fictions*, page 325. New York: Penguin, 1998. Translated by Andrew Hurley.

[But86] Barbara P. Buttenfield. Comparing distortion on sketch maps and mds configurations. *Professional Geographer*, 38(3):238–246, 1986.

[Byr82] R. W. Byrne. Geographical knowledge and orientation. In A. Ellis, editor, *Normality and Pathology in Cognitive Functions*, pages 239–264. London: Academic Press, 1982.

[CCF95] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Three-dimensional pliable surfaces: For effective presentation of visual information. In *ACM Symposium on User Interface Software and Technology (UIST 95)*, pages 217–226. ACM Press, 1995.

[Cha83] William G. Chase. Spatial representations of taxi drivers. In D. R. Rogers and J. A. Sloboda, editors, *Acquisition of Symbolic Skills*, pages 391–405. New York: Plenum Press, 1983.

[Cho99] Eric Chown. Error tolerance and generalization in cognitive maps: Performance without precision. In Reginald G. Golledge, editor, *Wayfinding Behavior*, pages 349–370. Johns Hopkins University Press, 1999.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* M.I.T. Press, Cambridge, Massachusetts, U.S.A., 1990.

[dBvKOS97] Mark de Berg, Marc van Krevald, Mark Overmars, and Otfried Schwarzkopf, editors. *Computational Geometry: Algorithms and Applications.* Springer, 1997.

[dBvKS98] Mark de Berg, Marc van Krevald, and Stefan Schirra. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Systems*, 25(4):243–257, 1998.

[Den97] Michel Denis. The description of routes: A cognitive approach to the production of spatial discourse. *Cahiers de Psychologie Cognitive*, 16(4):409–458, 1997.

[DH96] Ron Davidson and David Harel. Drawing Graphics Nicely Using Simulated Annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[DP73] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[DPCB99] Michel Denis, Francesca Pazzaglia, Cesare Cornoldi, and Laura Bertolo. Spatial discourse and navigation: An analysis of route directions in the city of Venice. *Applied Cognitive Psychology*, 13(2):145–174, 1999.

[ECMS97] Shawn Edmondson, Jon Christensen, Joe Marks, and Stuart Shieber. A general cartographic labeling algorithm. *Cartographica*, 33(4):12–23, 1997.

[EL82] R. J. Elliot and M. E. Lesk. Route finding in street maps by computers and people. In *Proceedings of the AAAI-82*, pages 258–261, 1982.

[Far88] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design.* Academic Press Ltd., 1988.

[Fei88] Steven Feiner. A grid-based approach to automating display layout. In *Proceedings of Graphics Interface '88*, pages 192–197, 1988.

[FR91] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software–Practice and Experience*, 21(11):1129–1164, 1991.

[GHMS93] Leonidas J. Guibas, John E. Hershberger, Joseph B. Mitchell, and Jack S. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry and Applications*, 3(4):383–415, 1993.

[GLM96] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Proceedings of SIGGRAPH 96*, pages 171–180, August 1996.

[Gol99] Reginald G. Golledge. Human wayfinding and cognitive maps. In Reginald G. Golledge, editor, *Wayfinding Behavior*, pages 5–45. Johns Hopkins University Press, 1999.

[Gra92] Winfried H. Graf. Constraint-based graphical layout of multimodal presentations. In T. Catarci, M. F. Costabile, and S. Levialdi, editors, *Proceedings Advanced Visual Interfaces AVI 92*, pages 365–385. World Scientific, May 1992.

[GSW01] Francois Guimbretiere, Maureen Stone, and Terry Winograd. Fluid interaction with high-resolution wall-sized displays. In *14th Annual Symposium on User Interface Software Technology*, November 2001.

[GW94] Michael Gleicher and Andrew Witkin. Drawing with constraints. *The Visual Computer*, 11:39–51, 1994.

[HG96] Walter Hower and Winfried H. Graf. A bibliographic survey of constraint-based approaches to CAD, graphics, layout, visualization and related topics. *Knowledge-Based Systems*, 9(7):449–464, 1996.

[Hof95] Micha Hofri, editor. *Analysis of Algorithms.* New York: Oxford University Press, 1995.

[Hol91] Nigel Holmes. *Pictorial Maps.* Watson-Guptill Publishing, 1991.

[Hol93] Nigel Holmes. *The Best in Diagrammatic Graphics.* Quarto Publishing, 1993.

[HS92] John Hershberger and Jack Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. In *5th Intl. Symp. on Spatial Data Handling*, pages 134–143, 1992.

[HWB95] Mikako Harada, Andrew Witkin, and David Baraff. Interactive physically-based manipulation of discrete/continuous models. *Computer Graphics*, 29(Annual Conference Series):199–208, 1995.

[IBM01] IBM introduces world's highest resolution computer monitor, 2001. `http://www.research.ibm.com/resources/news/20010627_display.shtml`.

[Imh75] Eduard Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[Imh82] Eduard Imhof. *Cartographic Relief Presentation.* Berlin: de Gruyter, 1982.

[Int97] Victoria L. Interrante. Illustrating surface shape in volume data via principal direction-driven 3d line integral convolution. *Proceedings of SIGGRAPH 97*, pages 109–116, August 1997.

[JBW95] Christopher B. Jones, Geraint L. Bundy, and J. Mark Ware. Map genrealization with a triangulated data structure. *Cartography and Geographic Information Systems*, 22(4):317–331, 1995.

[JHR98] Ning Jing, Yun-Wu Huang, and Elke A. Rundensteiner. Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *Knowledge and Data Engineering*, 10(3):409–432, 1998.

[Kea98] T. Alan Keahey. The generalized detail-in-context problem. *Proceedings of the IEEE Symposium on Information Visualization*, pages 171–180, October 1998.

[Kea00] T. Alan Keahey. A brief tour of nonlinear magnification, 2000. `http://www.cs.indiana.edu/hyplan/tkeahey/research/nlm/nlmTour.html`.

[KK89] Tomihisa Kamada and Satoru Kawai. An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, 31(1):7–15, April 1989.

[Knu99] Donald E. Knuth. *Digital Typography (CSLI Lecture Notes 78).* Center for the Study of Language and Information, Stanford, California, 1999.

[Len90] Thomas Lengauer, editor. *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley & Sons Ltd., 1990.

[LF01] Simon Lok and Steven Feiner. A survey of automated layout techniques for information presentations. In *1st International Symposium on Smart*

Graphics, March 2001.
`http://www.smartgraphics.org/schedule.html`.

[LHM99] Kristin L. Lovelace, Mary Hegarty, and Daniel R. Montello. Elements of good route dirtections in familiar and unfamiliar environments. In C. Freska and D. M. Mark, editors, *COSIT*, pages 65–82, 1999.

[LPL97] François Lecordix, Corinne Plazanet, and Jean-Philippe Lagrange. PlaGe: A platform for research in generlization. application to caricature. *GeoInformatica International Journal*, 1(2):161–182, 1997.

[LR94] John Lamping and Ramana Rao. Laying out and visualizing large trees using hyperbolic space. In *ACM Symposium on User Interface Software and Technology (UIST 94)*, pages 13–14. ACM Press, 1994.

[Lyn60] Kevin Lynch. *The Image of the City*. Cambridge, Massachusetts: The MIT Press, 1960.

[Mac86] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[Mac92] Alan K. Mackworth. The logic of constraint satisfaction. *Artificial Intelligence*, 58(1–3):3–20, 1992.

[Mac95] Alan M. MacEachren. *How Maps Work*. The Guilford Press, 1995.

[Mas94] Toshiyuki Masui. Evolutionary learning of graph layout constraints from examples. In *ACM Symposium on User Interface Software and Technology*, pages 103–108, 1994.

[MASC85] Joel McCormack, Paul Assente, Ralph Swick, and D. Converse. *X Toolkit Intrinsics – C Language Interface*. Digital Equipment Corporation, Maynard, Massachusetts, 1985.

[MB83] Carl Moreland and David Bannister, editors. *Antique Maps.* Phaidon Press, 1983.

[MDL92] Victor J. Milenkovic, Karen Daniels, and Zhenyu Li. Placement and compaction of nonconvex polygons for clothing manufacture. In *Proceedings 4th Canadian Conference on Computational Geometry*, pages 236–243, 1992.

[MF00] Zbigniew Michalewicz and David B. Fogel, editors. *How to Solve It: Modern Heuristics.* Springer, 2000.

[Mic97] Microsoft. *Microsoft Visual C++ MFC Library Reference.* Microsoft Press, Redmond, Washington, 1997.

[Mic98] Rainer Michel. Tactile maps for blind people. In Thomas Strothotte, editor, *Computational Visualization: Graphics, Abstraction and Interactivity*, pages 339–355. Springer-Verlag, 1998.

[MJ87] Alan M. MacEachren and Gregory B. Johnson. The evolution, application and implications of strip format travel maps. *The Cartographic Journal*, 24(2):147–158, 1987.

[MKT$^+$97] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-time nonphotorealistic rendering. *Proceedings of SIGGRAPH 97*, pages 415–420, August 1997.

[MMD97] Francesco Maffioli, Silvano Martello, and Mauro Dell'Amico, editors. *Annotated Bibliographies in Combinatorial Optimization.* John Wiley & Sons, 1997.

[MMK93] Brad A. Myers, Richard G. McDaniel, and David S. Kosbie. Marquise: Creating complete user interfaces by demonstration. In *INTERCHI '93, Human Factors in Computing Systems*, pages 293–300, Amsterdam, Netherlands, April 1993.

[Mon91] Mark Monmonier. *How to Lie With Maps.* The University of Chicago Press, 1991.

[MS91] Joe Marks and Stuart Shieber. The computational complexity of cartographic label placement. Technical Report ITR-05-91, Center for Research in Computing Techniology, Harvard University, March 1991.

[Ogi89] John Ogilby. *Britannia.* 1675: Old Hall Press, 1989. Facsimile of 1675 edition.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[PAF96] Corinne Plazanet, Jean-George Affholder, and Emmanuel Fritsch. The importance of geometric modelling in linear feature generalisation. *Cartography and Geographic Information Systems*, 22(4):291–305, 1996.

[Pai69] J. Pailhous. Representation de l'espace urbain et cheminements (representation of urban space and traveling). *Le Travail Humain*, 32:239–250, 1969.

[PBR98] Crinne Plazanet, Nara M. Bigolin, and Anne Ruas. Experiments with learning techniques for spatial model enrichment and line generalization. *GeoInformatica International Journal*, 2(4):315–333, 1998.

[Ram72] Urs Ramer. An iterative apprach for polygonal approximation of planar closed curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.

[RLS$^+$96] Steven F. Roth, Peter Lucas, Jeffrey A. Senn, Cristina C. Gomberg, Michael B. Burks, Philip J. Stroffolino, John A. Kolojejchick, and Carolyn Dunmire. Visage: A user interface environment for exploring information. In *Proceedings of Information Visualization*, pages 3–12, October 1996.

[RM91] Steven F. Roth and J. Mattis. Automating the presentation of information. In *Proceedings of the IEEE Conference on Artificial Intelligence*, pages 90–97, February 1991.

[RMS97] Kathy Ryall, Joe Marks, and Stuart Shieber. An interactive constraint-based system for drawing graphs. In *ACM Symposium on User Interface Software and Technology (UIST 97)*, pages 97–104. ACM Press, 1997.

[RR99] Charles Rubin and B. Russel. *Running Microsoft Word 2000*. Microsoft Press, Redmond, Washington, 1999.

[Saa99] Alan Saalfeld. Topologically consistent line simplification with the Douglas-Peucker algorithm. *Cartography and Geographic Information Systems*, 26(1):7–18, 1999.

[Sam90] Hanan Samet. *Applications of Spatial Data Structures*. Addison-Wesley, 1990. Held in Reading, Massachusetts.

[SF91] Dorée Duncan Seligmann and Steven Feiner. Automated generation of intent-based 3D illustrations. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):123–132, July 1991.

[SMFBB93] Michael Sannella, John Maloney, Bjorn Freeman-Benson, and Alan Borning. Multi-way versus one-way constraints in user interfaces: Experience with the deltablue algorithm. *Software–Practice and Experience*, 23(5):529–566, 1993.

[SPR$^+$94] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey. How to render frames and influence people. *Computer Graphics Forum*, 13(3):455–466, 1994.

[SV86] Lynn A. Streeter and Diane Vitello. A profile of drivers' map-reading abilities. *Human Factors*, 28(2):223–239, 1986.

[SVW85] Lynn A. Streeter, Diane Vitello, and Susan A. Wonsiewicz. How to tell people where to go: Comparing navigational aids. *International Journal of Man-Machine Studies*, 22:549–562, 1985.

[Tal83] Leonard Talmy. How language structures space. In Herbert L. Pick Jr. and Linda P. Acredolo, editors, *Spatial Orientation, Theory Research and Application*, pages 225–282. Plenum Press, 1983.

[Tel99] Telcontar. Rich Map Engine Route Finding Library API Reference Manual, 1999. `http://www.telcontar.com/documents/route_api_ref_rev1.2.pdf`.

[Thr72] Norman J. Thrower. *Maps and Man: An Examination of Cartography in Relation to Culture and Civilization.* Englewood Cliffs, New Jersey: Prentice Hall, 1972.

[Thr96] Norman J. Thrower. *Maps in Civilization: Cartography in Culture and Society.* Chicago: University of Chicago Press, 1996.

[TL99] Barbara Tversky and Paul Lee. Pictorial and verbal tools for conveying routes. In C. Freska and D. M. Mark, editors, *COSIT*, pages 51–64, 1999.

[TMB] Barbara Tversky, Julie B. Morrison, and Mireille Betrancourt. Animation: Does it facilitate learning? *Internation Journal of Human Computer Studies.* In Press.

[Tol48] Edward C. Tolman. Cogntive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948.

[TT92a] Holly A. Taylor and Barbara Tversky. Descriptions and depictions of environments. *Memory and Cognition*, 20:483–496, 1992.

[TT92b] Holly A. Taylor and Barbara Tversky. Spatial mental models derived from survey and route descriptions. *Journal of Memory and Language*, 31:261–282, 1992.

[TT96] Holly A. Taylor and Barbara Tversky. Perspective in spatial descriptions. *Journal of Memory and Language*, 35:371–391, 1996.

[Tuf90a] Edward Tufte. *Envisioning Information*. Conneticut: Graphics Press, 1990.

[Tuf90b] Edward Tufte. *The Visual Display of Quantitative Information*. Conneticut: Graphics Press, 1990.

[Tuf97] Edward Tufte. *Visual Explanations*. Conneticut: Graphics Press, 1997.

[Tun93] Daniel Tunkelang. A Layout Algorithm for Undirected Graphs. In *Proceedings of Graph Drawing '93, ALCOM International Workshop PA RIS 1993 on Graph Drawing and Topological Graph Algorithms*, 1993.
`http://reports-archive.adm.cs.cmu.edu/anon/1994/ CMU-CS-94-161.ps.`

[Tve81] Barbara Tversky. Distortions in memory for maps. *Cognitive Psychology*, 13(3):407–433, 1981.

[Tve92] Barbara Tversky. Distortions in cognitive maps. *Geoforum*, 23(2):131–138, 1992.

[VW93] Maheswari Visvalingam and J.D. Whyatt. Line generalisation by the repeated elimination of points. *Cartographic Journal*, 30(1):46–51, 1993.

[WGW90] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):11–21, 1990.

[WM98] Zeshen Wang and Jean-Claude Müller. Line generalization based on analysis of shape characteristics. *Cartography and Geographic Information Systems*, 25(1):3–15, 1998.

[WW94] Louise Weitzman and Kent Wittenburg. Automatic presentation of multimedia documents using relational grammars. In *In Proceedings* *of ACM International Conference on Multimedia MULTIMEDIA 94*, pages 443–452. New York: ACM Press, October 1994.

[ZB96] F. Benjamin Zhan and Barbara P. Buttenfield. Multi-scale representation of a digital line. *Cartography and Geographic Information Systems*, 23(4):206–228, 1996.

[ZM00] Michelle Zhou and Sheng Ma. Towards applying machine learning to design rule acquisition for automated graphics generation. In *In Proceedings 2000 AAAI Spring Symposium Series on Smart Graphics*, March 2000.

[ZM01] Michelle Zhou and Sheng Ma. Representing and retrieving visual presentations for example based graphics generation. In *1st International Symposium on Smart Graphics*, March 2001. `http://www.smartgraphics.org/schedule.html`.

[Zor97] Steven Zoraster. Practical results using simulated annealing for point feature label placement. *Cartography and GIS*, 24(4):228–238, 1997.