Energy Minimization Methods for Feature Displacement in Map Generalization

Dissertation

zur

Erlangung der naturwissenschaftlichen Doktorwürde
(Dr. sc. nat.)

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät

der

Universität Zürich

von

Matthias Bader

von

Zürich

Begutachtet von

Prof. Dr. Robert Weibel
Prof. Dr. Kurt Brassel
Prof. Dr.-Ing. Monika Sester

Zürich 2001

Die vorliegende Arbeit wurde von der Mathematisch-naturwissenschaftlichen Fakultät der Universität Zürich auf Antrag von Prof. Dr. R. Weibel und Prof. Dr. K. Brassel als Dissertation angenommen.

Abstract

The work reported in this thesis considers the issue of feature displacement in cartographic generalization. Attempts to display cartographic data at scales smaller than the source scale result in spatial conflicts; map symbols clutter or overlap. Several map generalization operators may be applied to resolve these problems, including displacement. The main goal of displacement is to separate conflicting objects, either by deformation of their outline or by shifting them in their entirety. Cartographic displacement is a contextual operation, dealing with several objects at once, and also affecting the neighborhood of the conflicting features. Local reasoning often fails, since conflicts are not solved, but simply pushed back and forth. Existing sequential displacement algorithms fail to incorporate the required broader view necessary for successful generalization.

This thesis proposes new techniques to handle displacement. Instead of working through all proximity conflicts one after the other, as executed by sequential methods, a global approach is adopted. Here, cartographic displacement is interpreted as an optimization problem. The proposed models thereby rely on an interplay of external and internal forces which compete a solution. Internal regularizing forces constrain the displacement of map objects or object structures, while at the same time proximity conflicts give rise to external forces, which try to deform objects and push them away from conflicts. A balance between internal and external forces is iteratively sought – *the system energy is minimized*.

For the modeling of regularizing structures, this research draws upon techniques used in engineering sciences. In the thesis, care is taken to illustrate the background and theory of these techniques for geographers who are perhaps less familiar with this subject. A goal of this study is to adapt these techniques to demands arising in cartography. The success of the new algorithms depends on the way in which cartographic constraints and cartographic knowledge can be translated into the numerical methods.

Three algorithms are proposed. First, the use of *snakes* – a widespread technique in computer vision – is investigated for the cartographic displacement of roads. This technique offers a promising way to constrain road shapes and to reach a good displacement solution iteratively. It is described how snakes can be set up to allow the cushioning of displacement through line networks, which frees roads from the rigid restriction of fixed junction nodes. Further, the importance of junctions is pointed out for the displacement operation. It is shown how formerly computed local modifications of the junction geometry can be incorporated into the iterative process.

A second algorithm for road displacement makes use of *elastic beams*, a concept

originally developed in structural mechanics. Beams prove to be better suited for road displacement than snakes, since they provide a control of road directions and offer an interface over which a cartographic behavior can be carried over on the roads; bending and stretching of road segments can be controlled explicitly.

Finally, the work addresses building displacement in urban blocks. Even though the constraints guiding this class of displacement vary from the ones handled in road displacement, a tool from structural mechanics again provides useful input. The structure of buildings is interpreted as *ductile truss*, wherein characteristic object relations can be controlled and preserved.

Zusammenfassung

Die vorliegende Dissertation beschäftigt sich mit der Objektverdrängung in der kartographischen Generalisierung. Eine Generalisierung wird nötig, wenn eine Massstabsverkleinerung von einer Quell- zu einer Zielkarte zu einem Raumkonflikt der darzustellenden Objekte führt. Um Objekte auch bei kleineren Massstäben noch gut lesbar darzustellen, muss die Kartengraphik angepasst werden. Das Symbol eines Objekts übertrifft bei kleiner werdendem Massstab zusehends dessen wahre Grösse; die Folge sind überlappende Objektsymbole. Versuche, Daten einfach ungeneralisiert in kleinere Massstäbe zu übernehmen, sind daher zum Scheitern verurteilt.

Die Objektverdrängung ist eine mögliche Operation, um Raumkonflikten entgegenzuwirken. Ziel der Objektverdrängung ist es, Konflikte von zu nahe liegenden Objekten durch lokale Deformation der Objektgeometrie oder durch Verschiebung der Objekte als Ganzes zu entschärfen. Die Objektverdrängung erfordert eine simultane Handhabung verschiedener Objekte unter Berücksichtigung ihrer räumlichen Kontexte. Bestehende Algorithmen werden der räumlichen Komplexität dieser Problemstellung nicht gerecht.

Ziel dieser Arbeit ist es, neue und bessere Verfahren für die kartographische Objektverdrängung zu präsentieren. Eine globalere Aufarbeitung von Konfliktzonen wird erreicht, indem die Objektverdrängung als Optimierungsproblem interpretiert wird. Den in dieser Dissertation beschriebenen Methoden liegt der gemeinsame Ansatz zu Grunde, dass über eine Wechselwirkung von inneren und äusseren Kräften eine gut ausbalancierte Verdrängungslösung angestrebt wird. Äussere Kräfte, hervorgerufen durch zu nahe liegende Symbole, versuchen Objekte zu deformieren und zu verschieben; innere Kräfte wirken erhaltend auf die Form, bzw. die räumliche Struktur der Objekte. Eine Balance der Kräfte im Sinne der *Energieminimierung* wird iterativ angestrebt und beschreibt so eine gute Verdrängungslösung.

Für die Modellierung des Problems werden Ansätze aus den Ingenieurwissenschaften beigezogen. Entscheidend für den Erfolg unserer Algorithmen ist die Anpassung dieser Modelle an die kartographischen Bedürfnisse. Es wurde Wert darauf gelegt, diese für den Geographen vielleicht etwas ungewöhnlichen Arbeitsinstrumente detailliert und illustrativ zu erklären.

Drei Methoden werden in dieser Arbeit vorgestellt. In einem ersten Teil werden Energie minimierende Splines, auch *Snakes* genannt, auf ihre Tauglichkeit für die Verdrängung von Strassen hin untersucht und erweitert. Das Konzept der Snakes stammt aus der Bilderkennung (computer vision) und wurde erst kürzlich für die kartographische Generalisierung vorgeschlagen. Snakes haben den Vorzug, Formveränderungen ganzer Linien steuern zu können, und deren iterative Annäherung

an eine gute Verdrängung erweist sich auch als nützlich für die im Folgenden diskutierten Verfahren. Unsere Erweiterungen befreien die Snakes von den restriktiven Fesseln fixierter Strassenknoten. Zudem wird auf die Bedeutung von Kreuzungen für die Verdrängung eingegangen. Es wird gezeigt, wie eine lokal berechnete Anpassung der Kreuzungsgeometrie an verbreiterte Symbole in den iterativen Verdrängungsprozess eingebaut werden kann.

Ein zweiter Ansatz macht sich das Konzept von Biegebalken (*elastic beams*), wie von Bauingenieuren verwendet, zu Nutze. Unser auf einer Biegebalkenstruktur basierender Algorithmus hat gegenüber den Snakes den Vorteil, dass die Orientierung der Strassensegmente präziser beschrieben wird und dass über deren physikalische Eigenschaften, nämlich deren Biegung und Längsdehnung, eine explizite Kontrolle der Strassenverformung erlaubt wird.

Abschliessend wird ein Algorithmus für die Verdrängung von Häusern in städtischen Strassenblocks vorgestellt. Auch wenn sich die kartographsichen Anforderungen und Einschränkungen hier stark von der Strassenverdrängung unterscheiden, lassen sich doch die zuvor erstellten Konzepte wieder nutzen. Eine Interpretation der Häuser-struktur als deformierbares Fachwerk ermöglicht räumliche Besonderheiten zwischen Gebäuden zu modellieren und zu konservieren.

Acknowledgments

Prof. Robert Weibel for the ever-present interest, many stimulating discussions, and the conceded and supported research liberties.

Dr. Mathieu Barrault for surprising me with an endless stream of useful and useless ideas and his motivating enthusiasm for digital cartography.

Prof. Dr. Kurt Brassel and Prof. Dr.-Ing. Monika Sester for acting as reviewers of this dissertation.

Dr. Peter Højholt and Dr. Dirk Burghardt whose research strongly influenced my work.

Dr. Geoff Dutton and Dr. Frank Brazile for accompanying my first steps as PhD student and for their collaboration in the AGENT project.

The staff of the Laboratoire Cogit, at the Institut Géographique National in Paris, for their hospitality and for giving me warm response at their research lab; particularly Dr. Anne Ruas, who enabled the stay, and Bénédicte Bucher and Sébastien Mustière for their support and friendship.

Friends and colleagues in the Department who expressed academic and friendly interest in my work; particularly Daria Martinoni whose support was indispensable not only as contact point for mathematical problems.

Alistair Edwardes for giving this thesis a face-lift in English.

Contents

Chapter 1

Introduction

1.1 Motivation

Cartographic Generalization Since the early 80's, national mapping agencies (NMAs) and commercial providers of cartographic data have gathered together a digital representation of the real world. Considerable effort has been spent on improving data compilation and data storage. Survey and fieldwork are more and more being replaced by remote sensing technology to achieve higher data capture rates. The use of digital databases has superceded printed maps as the mode of storage for geographic information. The availability of all kind of spatial data mirrors the success of this digital revolution. And Geographic Information Systems (GIS), accompanied by faster and cheaper computer hardware, provide a platform to manage this tide of digital information.

The availability of digital databases promises cost-effective update and production of maps. When a mapping agency updates information in its database, usually more than one map is affected by a given change. Many maps may be derived from the largest scale map. The actions associated with such a change of scale are enclosed in the term *map generalization*.

Simply scaling maps results in unreadable, but also incorrect information. Typical consequences of displaying pre-generalized data at a smaller scale than originally intended are that the detail of individual map features may become too small to be legible, while neighboring symbols that should be clearly distinguishable become cluttered or may even overlap. These latter problems of graphic conflict arise in particular when the map symbols are no longer true to the scale of the features they represent. The primary goal of generalization is thus to reduce the complexity of a compiled map product while maintaining the salient elements and characteristics subject to the purpose of the map (Cromley 1992), see Figure 1.1.

Printed topographic maps traditionally impose high requirements concerning accuracy, legibility and aesthetic quality. In the overall process of automated mapmaking, generalization is still a largely unresolved issue. Even though ink and paper disappeared with the digital revolution, generalization is still performed to a large part in time-consuming and cost-intensive manual (interactive) work by trained cartographers. As the variety of demanded maps increases with new applications of spatial data (e.g car navigation screen display, Internet on-demand maps) au-

tomation of generalization becomes more and more essential. Without satisfactory visualizations of spatial datasets, map producers can not meet the requirements of their customers, demanding cheap, up-to-date and thematically focused maps. However, Geographic Information Systems in general also depend on a facility for working through the scales to use their entire potential.

Figure 1.1: Cartographic generalization (Map IGN).

Operators and Algorithms in Generalization Manual cartographic generalization involves an unruly set of techniques, which cartographers learned by tutelage, example and intuition, deliberately, even rigorously, but not necessarily formally. For the automation of generalization, it is essential to gain a better formalized view of the overall process. Therefore, generalization actions have been decomposed into several basic operators. We can distinguish attribute and spatial transformations. The latter consists of – roughly and depending on the classifiers background – selection, simplification, enhancement, exaggeration, typification and displacement.

Over the last twenty years, researchers found and described a variety of algorithms to translate these operators into a piece of code. Of course, the availability of such algorithms alone does not guarantee a tool for automated generalization. This manifold of methods has to be orchestrated – a very demanding task, as one has to simulate in this process the holistic view required in generalization.

For this purpose, the AGENT project (Lamy *et al.* 1999) was launched. The orchestration of generalization methods is carried over to a multi-agent system, a system on which several computational entities, called agents, interact with one another. Agents perceive and act upon the environment in which they are situ-

ated, applying their individual knowledge, skills and other resources to accomplish generalization.

The AGENT project resulted in a successful prototype, integrating both existing algorithms and current knowledge of the generalization process. However, it became clear at an early stage of the project that progress in generalization research had not yet advanced as far as was hoped: the generalization algorithms, implementing individual generalization operators, and so building the fundament of the workbench, did not yet fully meet the requirements and quality known in manual cartography. As algorithms are the core of a generalization module, they must provide stable and predictable results.

Displacement One of the operations not yet transformed successfully into algorithmic form is *displacement*. Displacement is necessary primarily because of the effect of increasing symbol width of map objects relative to reality. At small scales, the map symbols become much wider, when map scale is taken into account, than the size of the objects on the ground. The consequences are a loss of free space and the overlap of object symbols. Displacement attempts to resolve such proximity conflicts between objects by – as the name indicates – displacing objects apart. Cartographic displacement is discussed in more depth in the next chapter.

1.2 Objectives

Cartographic displacement is an important operation in the process of generalization that is still performed interactively by human cartographers. So far, attempts to automate this operation have failed (Section 2.5).

Cartographic displacement cannot be solved by simply analyzing and displacing the objects directly involved in a conflict. Displacement has to be propagated through the map in order to minimize shape distortions and preserve object relationships. Traditional approaches fail to take this wider scope into account.

This work focuses on optimization algorithms to solve cartographic displacement. This type of algorithm was only recently introduced to the field of cartographic generalization. First results reported in the literature were promising (Section 2.5.2), but recommended improvements, because the algorithms were not yet sufficiently adapted to the constraints of cartographic displacement. The objectives of this work are to analyze and improve existing optimization methods for cartographic displacement, by enriching the underlying model, and to propose new techniques to overcome assessed shortcomings (see also Section 2.6).

Chapter 2

Map Generalization and Cartographic Displacement

2.1 Generalizing Spatial Data

2.1.1 Generalization in Conventional Cartography

In conventional cartography, "map generalization is responsible for reducing complexity in a map in a scale reduction process, emphasizing the essential while suppressing the unimportant, maintaining logical and unambiguous relations between map objects, and preserving aesthetic quality. The main objective then is to create maps of high graphical clarity, so that the map image can be easily perceived and the message the map intends to deliver can be readily understood" (Weibel and Dutton 1999, p.126).

Scale reduction of a map leads to competition for space amongst map objects[1] caused by two cumulative effects (see Figure 2.1): at a reduced scale, less space is available on the map to place symbols representing objects, while at the same time, symbol size increases relative to the ground it covers, in order to maintain size relations and legibility. The symbolization of roads, for instance, will no longer be to scale and hence the lines can overlap; distances between buildings fall below the threshold of perception, both for human eyes and output devices; building groups are perceived as clusters, no longer giving an impression of the real world structure.

Map scale is not the only factor that influences generalization. The purpose of the map also has a decisive influence. It defines what is essential, which guides not only the selection of objects and feature classes, but also, with which symbology and what accuracy these objects need to be displayed. This work deals with topographic maps. Topographic maps traditionally impose high requirements regarding accuracy.

[1] We use the term 'object' to denote a set of points, lines or polygons in a database that represent a real world entity, for example a road, or a building. In literature, the terms 'feature' and 'object' are often used synonymously – we use mainly the term 'object'. The term 'object class' denotes a logical classification of objects. We make frequent use of the term 'feature class', which is synonymous to 'object class'.

Figure 2.1: By only scaling a map, objects become too small and the map loses its legibility (top). There is not sufficient space to enlarge objects and adjust symbols (middle). The map needs to be generalized (bottom).

2.1.2 Generalization in Digital Systems

In comparison to generalization in conventual cartography, generalization in digital systems has to be understood in a wider meaning: each transition from one model of the real world to another, which comes with a loss of information, requires generalization. Figure 2.2 shows how transitions take place in three different areas along the database and map production work-flow (Brassel 1990, Müller *et al.* 1995, Weibel 1997).

Object Generalization This process takes place whenever a database is created as a representation of the real world. Since our world presents us with an infinite reservoir of details and resultant data, a representation of all data is impossible. Each database can only hold a selection of the real world data, and usually only a fraction of the captured data; This selection must reflect the intended purpose of the data and will be limited by computer memory.

Model Generalization While the process of object generalization has had to be carried out in much the same way as in the preparation of data for a traditional map, model generalization is new and specific to the digital domain (Weibel and Dutton 1999). The goal of model generalization is a *controlled* reduction of data. The selection of a database-subset is thereby not guided by artificial or intuitive judgments, but by mathematically determined thresholds, which also allow the user to make statements about accuracy after model generalization; ensuring data is still usable for analysis. The reduction of data is desirable in order to save storage and to increase computational efficency.

Cartographic Generalization This is the term commonly used to describe the generalization of spatial data for cartographic visualization. It is the process

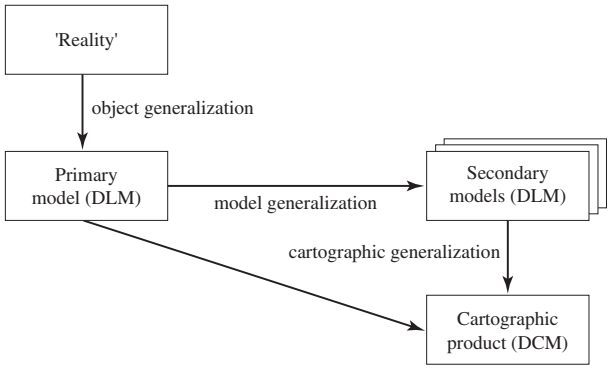


Figure 2.2: Generalization as a sequence of modelling operations (after Grünreich (1985)).

most people typically think of when they hear the term 'generalization' (Weibel and Dutton 1999). By means of spatial and attribute transformations, it is used to counteract the unwanted effects of scale changes in order to produce aesthetic maps.

2.2 Digital Cartographic Generalization

2.2.1 Definition

In keeping with Section 2.1, digital cartographic generalization can be defined as "the process of deriving, from a data source, a symbolically or digitally encoded cartographic data set through the application of spatial and attribute transformations" (McMaster and Shea 1992, p.3). The objectives of this derivation process are to reduce in scope the amount, type, and cartographic portrayal of the data consistent with the chosen map purpose and intended audience. The goal is to maintain clarity of presentation at the target scale.

2.2.2 Constraints

Whilst, cartographic generalization does have an artistic component, it also underlies various other cartographic criteria, known as constraints. A constraint is therefore not understood in the narrow sense of strictly limiting a process; rather it can be thought of as a design specification to which a solution *should adhere* (Weibel and Dutton 1998). The evaluation of constraints allows the quality of a solution to be judged. Ruas (Ruas and Mackaness 1997, Ruas 1999) uses constraints to trade off different solutions and to guide the generalization process, reasoning about the choice and the sequence of algorithms (see Section 2.2.4).

Defining all constraints to which a generalization solution should adhere, inevitably results in an almost endless list. It is impossible to meet all constraints;

many constraints even contradict. In the generalization process, we emphasize those constraints, that are important with respect to the intended map purpose. A schematized metro map, for example, emphasizes the structure of the network, not omitting any route or intersection node, while sacrificing constraints regarding accuracy and shape to ease legibility. In contrast, in topographic maps, constraints regarding the accuracy and shape of roads are important, whereas the road network can be pruned of small roads to make space for other objects.

We do not dwell further on the nature of constraints here; the constraints, which are of importance for our work, are outlined in Section 2.3.4 and 2.3.5. For a detailed discussion of constraints in cartographic displacement see Ruas (1999) and Dutton *et al.* (1998); feature class dependent work exists for building generalization in urban blocks (Regnauld 1998, Ruas 1999) and for line simplification (Weibel 1997).

2.2.3 Operators and Algorithms

Operators To compensate for the visual problems described in Section 2.1, adjustments need to be made. To this end, the overall process of generalization is often decomposed into individual sub-processes, denoted as 'operators' (Lichtner 1979, McMaster and Shea 1992). Operators were identified initially by studies of the human cartographer and later enriched to decompose the task of generalization into more detail for automation. The ideas introduced by algorithm developers led to the additional fragmentation of these operators. Even today, the research community has not agreed upon a common classification of operators – nor on using the terms of the individual operators to mean the same thing (Rieger and Coulson 1993).

Figure 2.3 shows the operator classification proposed for the AGENT project (Bader *et al.* 1999). The scope here is on automated generalization. Other classifications can be found, for example, in Hake and Grünreich (1994) or McMaster and Shea (1992).

The presented classification makes a distinction into 'independent' and 'contextual' operators, two terms well established in generalization literature. Independent operators deal with one object at a time, whereas contextual operators modify groups of objects. In a perfect world, manipulating one object at a time does not mean disregarding its environment; each change of an object's geometry has (whether desired or unintended) an impact on its neighborhood, and therefore, a modification should not be computed without taking the object's context into consideration. Yet, algorithms that modify objects in isolation are so complex and fragile that an extension is barely conceivable; hence objects are generalized detached from their neighborhoods.

Algorithms The relationship between generalization *operators* and *algorithms* is hierarchical. An operator defines the transformation that is to be achieved; a generalization algorithm is then used to implement the particular transformation. This implies that operators are independent of a particular data model, while algorithms are linked to a specific representation, a certain change of scale and a given data structure. One algorithm alone therefore seldom holds for an entire operator.

Over the last decades, researchers from cartography, geography and computer science proposed a variety of algorithms to translate operators into computer programs. The best studied operator is probably the weeding (simplification) of lines. Many algorithms have been proposed, concentrating on simple geometric considerations, and not respecting the character of lines, e.g. Douglas and Peucker (1973), de Berg *et al.* (1995) or Li and Openshaw (1992). However, a generic simplification of lines – whether they define roads or characterize the boundary of polygons – is of little use. The shape of lines needs to be analyzed in detail first, before an algorithm can be launched with adequate parameters. Researchers at the *COGIT Laboratory* at the *Institut Géographique National (IGN)*, the French national mapping agency, have examined in detail the generalization of individual roads, which has resulted in a suite of algorithms that reached the requisite quality for integration in their production system, see e.g. Lecordix *et al.* (1997) and Mustière (1998).

Few algorithms exist for contextual operators. Cartographic displacement moved into the spotlight a few years ago. An evaluation of displacement algorithms is the subject of Section 2.5. Typification is still a stepchild in the generalization community. Promising methods exist by Regnauld (1998) and more recently by Sester and Brenner (2000).

2.2.4 Orchestration

Having a multitude of powerful algorithms at hand by no way means success in automating the generalization process. The main problem is still: How can we orchestrate algorithms to provide cartographically acceptable solutions? The holistic characteristic of generalization must be turned into a strategy, a series of algorithms, chosen in correct sequence and using the right parameters.

The AGENT project (Lamy *et al.* 1999) looked at addressing in this situation. The objective of the project was to model the holistic nature of generalization, where objects compete for space and a valid solution, by means of a multi-agent-systems (MAS). A MAS can be defined as "a loosely-coupled network of problem solvers – the agents – that work together to solve problems that are beyond their individual's capabilities" (Durfee *et al.* 1989). MAS are a field of research in distributed artificial intelligence. For more details on MAS in general, see Moulin and Chaib-Draa (1996) or Shoham (1993), and Baeijs *et al.* (1996) for an early application of MAS to cartographic generalization.

Early experience in the project led to the observation that, even though a variety of algorithms existed for integration, these algorithms did not satisfy cartographic requirements (the exception was the methodology to use with individual roads provided by IGN). When triggered automatically, weaknesses of the algorithms appeared regarding stability, predictability of results and ease of control (parameter settings). Early versions of the AGENT software did not perform to expectations, as the algorithms did not accomplish the requirements. Therefore the project members also continued to devise better algorithms.

The work reported in this thesis is thought of as a contribution towards this fundament of algorithms. Automated generalization will fail, if powerful algorithms are not at hand, despite the fact that concepts already exist to address the next hurdles.

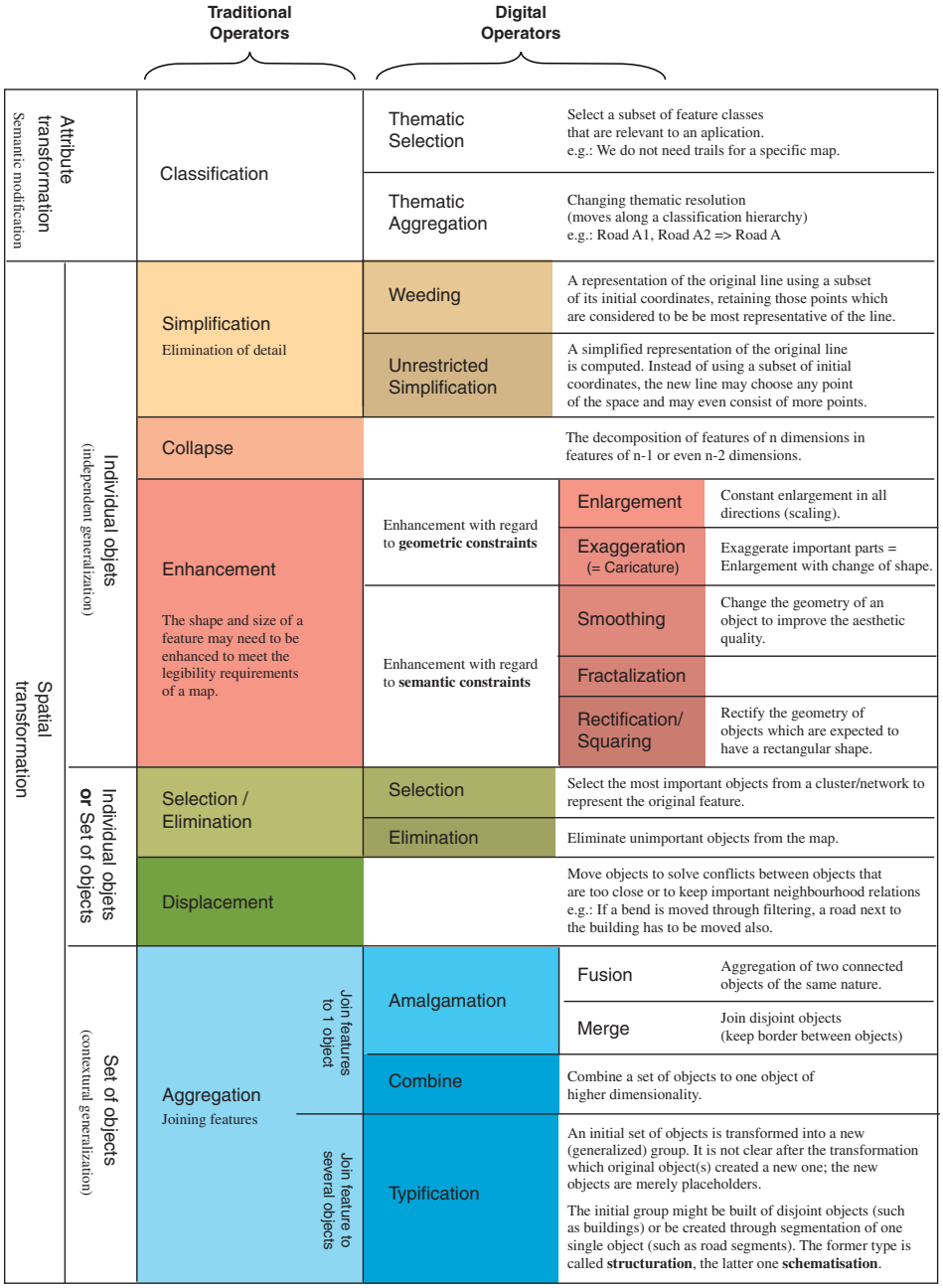


Figure 2.3: Typology of generalization operators.

The focus of this thesis lies on cartographic displacement algorithms. Cartographic displacement – which is defined and properly bounded in the next section – is an important operator in the overall process. Besides its importance, the choice

of this operator as research subject was due to three reasons. Firstly, since the mid 1990s, cartographic displacement research has grown rapidly. Many promising algorithms have been proposed, though often waiting for improvement. Secondly, the field is open for optimization techniques. The knowledge we can gain with these methods is not only useful in generalization, but may also be valuable to solve other problems in the field of geographic information science. Finally, there still does not exist a satisfactory algorithm, particularly for the displacement of linear features, but also of other features which would profit from refinements.

2.3 Cartographic Displacement

2.3.1 Definition

We follow the definition of Weibel and Buttenfield (1988), p.71: "Displacement concerns the resolution of spatial conflicts between map elements in order to maintain spatial relation, provide clarity, or fit other elements on the map". In the following, the shorter term 'displacement' will be used as equivalent to 'cartographic displacement'. In the literature, the terms 'featured displacement' and 'cartographic feature displacement' are also often used. Again, these terms are equivalent to 'cartographic displacement'.

Spatial conflicts are caused either (1) by the decrease of space between objects when moving from one scale to another, or (2) by enlarged symbolization, which is necessary to show objects legible at smaller scales, or, finally, (3) by other generalization algorithms, when they perturb spatial relations, by moving objects without adjusting the environment to counteract such an action.

Cartographic displacement due to causes (1) and (2) is also necessary in traditional cartography. Conflicts of type (3) arise only in the environment of automated generalization, where other algorithms trigger displacement to maintain spatial relations, or worse, when other algorithms introduce new proximity conflicts[2].

2.3.2 Related Terms

Translation and Deformation We distinguish two types of displacement. An object involved in a conflict is separated either by *translation*, meaning that the object is shifted as a whole rigid body, or by *deformation*, where the parts of the object that face the conflict spot are locally deformed to increase the space between objects.

The purpose of deformation is to minimize the area of correction, as each change of geometry increases the danger for new conflicts. The costs of keeping displacements small are paid by the altered shape of objects. Deformation is essential within the road network. Otherwise, a small displacement would perturb the entire map.

Translation, on the other hand, is mainly applied to small objects with clear geometric shapes. For instance, buildings are mainly translated, as a deformation

[2] Ruas (1998) uses the term 'active displacement' to denote a displacement necessary due to proximity conflicts between symbols, and 'reactive displacement' to denote a displacement needed as a result of prior generalization. Using Ruas' classification, (1) demands active displacement, whereas (2) and (3) require reactive displacement.

is not generally possible given the constraints of minimal size and right angles. For buildings, the danger of generating new conflicts is limited to the small neighborhood of a building – though even this displacement may trigger an avalanche of successive corrections.

Propagation This term derives from the functionality of deformation algorithms. Normally, deformation algorithms model a solution in a two-step process. First, a displacement vector – or a set of displacement vectors – are computed which would resolve the proximity conflict at the conflict spot (see Figure 2.4). This shift of single vertices may lead to unacceptable shape distortions. Thus, in a second step, which is called propagation, the displacement vector(s) are cushioned along the line (Figure 2.4,right).

propagation needs to balance two conflicting constraints. On the one hand, displacements must be cushioned to preserve the shape. On the other hand, cushioning reduces positional accuracy and increases the risk of introducing new conflicts due to object interference. The aim is to limit the propagation such that the character of the target line is preserved within a minimal cushioning distance. It is the cartographer who judges whether small shape distortion at the cost of reduced geometric accuracy is more important than higher accuracy at the cost of more severe shape distortion.

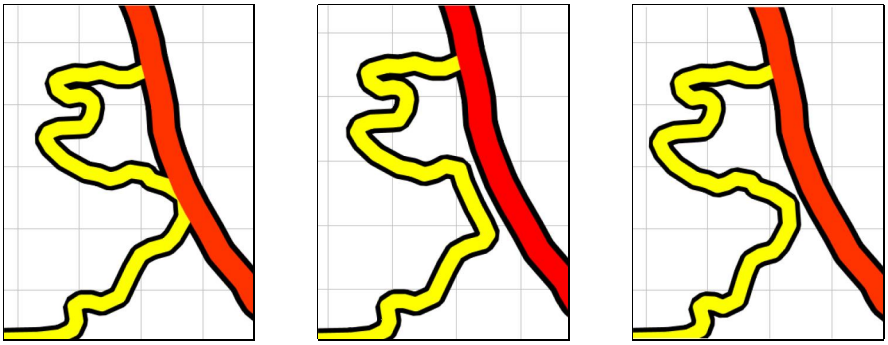


Figure 2.4: Roads in conflict due to symbol overlap (left). Applying only a correction to the interfering vertices gives unacceptable results (middle). The displacement has to be propagated throughout the line (right).

2.3.3 Distinction of Displacement Problems

Displacement is a contextual operation. As such, it must deal with many objects at once. Thereby, interfering objects do not necessarily need to belong to the same feature class; in traditional cartography, good displacements are affected by the many complex interactions among objects of many feature classes (see Figure 2.5). A triangulation point, for example, may trigger the deformation of a road due to its high demands regarding positional accuracy; buildings may follow a previously

shifted building to preserve their initial contextual alignments; a river may be reoriented to indicate a proper underpass under a railroad.

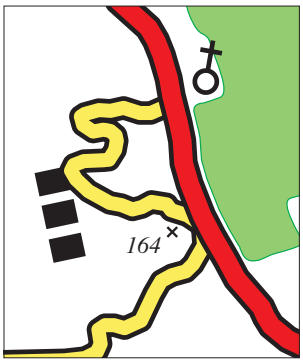


Figure 2.5: Displacement is a contextual operator, which deals with objects of different feature classes at once.

The diversity of objects involved in a conflict makes displacement a difficult task to manage, as each object brings with it its own (feature class dependent) constraints and peculiarities. Splitting into sub-tasks is therefore required – and cartographically also possible. Two kinds of displacement problems can be distinguished, namely the exclusive displacement of linear features and the displacement of points and areas with respect to the fixed structure of lines following on from this. At the heart of this distinction lies the assumption, that we can first solve proximity conflicts in the transport network (and between roads in particular) and then adjust other feature classes to this skeleton. This belief is corroborated by empirical investigations and practical considerations, see Jäger (1991) and Lichtner (1979).

The approach of restricting problems to smaller domains has generally pursued by the research community; mainly two types of displacement problems are tackled. On one side, as a representative for linear feature displacement, *road displacement* is studied. Road displacement is especially demanding, because the symbol width of the roads outweighs their true width significantly, and because road junctions require additional modifications. On the other side, representing the problems of object adaption to a given linear skeleton, *building displacement* in urban blocks is investigated. The goal here is to ensure the maintenance of a given distance between buildings and the bounding road, but also to ensure a minimal distance between the buildings themselves. The geometries of buildings are thereby seldom touched, instead their location is altered ('rigid isolated feature displacement').

The presented distinction of problems is in agreement with the classification of displacement into deformation and translation. The relocation of buildings constrained by a fixed boundary is traditionally solved by translative displacement, while solving proximity conflicts between linear features requires their deformation. The two situations vary not only in the type of correction, but also by the

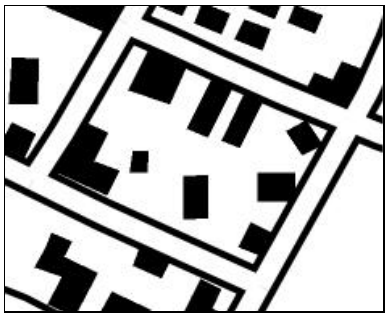
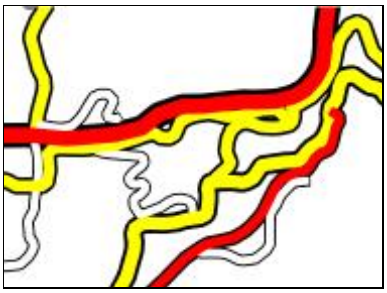



Figure 2.6: The two classical displacement situations tackled in displacement research: the displacement of buildings in urban blocks (left) and the displacement of roads (right)

constraints they emphasize, by which these actions are controlled. Linear feature displacement is subject to high requirements regarding preservation of shape, whereas building displacement is subject to constraints controlling the preservation of patterns and spatial relationships between buildings (see Section 2.3.4 and 2.3.5).

Translation	$\longleftrightarrow$	Building Displ.	$\longleftrightarrow$	Spatial Relations / Patterns
Deformation	$\longleftrightarrow$	Road Displ.	$\longleftrightarrow$	Shape

The question is: How far do these two problems cover the entirety of displacement problems? We did not investigate this question exhaustively. However, the study of sample maps as well as the literature, e.g. Lichtner (1979), Burghardt (2000), suggests that these two situations are representative of at least the overwhelming majority of displacement problems found. Hence, if good solutions for both types of problems were found, one could deal with the majority of conflicts. Displacement problems involving point features could be handled by analogy to building displacement. Small polygons treated like buildings; larger polygons could be represented by their borderline and deformed like linear objects. It is evident that small adaptations may be required, but the core of the algorithms should be portable.

However, we have to be aware, that the reduction of the issue to road and building displacement, and the associated focus to pure deformation and translation, stays an oversimplification, as the combination of such processes is not covered.

Summary Our work focuses on road and building displacement, which are instances of linear feature displacement and rigid isolated body displacement. Both operations are challenging; road displacement extends the problem of linear feature displacement by makeing new demands regarding the legibility of junctions. Building displacement struggles with proximity conflicts in dense and highly constrained urban blocks.

We believe road and building displacement to be representative of the entire class of displacement problems. We abstain from specifying both algorithms potential for use with other feature classes; this abstraction is left to the reader.

2.3.4 Road Displacement

Road Displacement as Generalization Operation among Others Generalization is not just the application of displacement, but combines many generalization operations. The selection of permissible and advisable operations varies from situation to situation and also with scale.

In road generalization, rural and urban situations have to be distinguished, as they encompass different kinds of geographical information. *Urban road networks* consist of a dense set of small roads with simple geometries. A solution is gained by typification of the network; displacement is only a tool of secondary importance. Therefore. we exclude urban road network displacement from our observations.

In contrast, *rural road networks* allow little use of a typification algorithm. The lines hold complex shapes, which require deformation and displacement. Besides displacement, only the elimination of roads leads to the solution of a conflict. However, no roads are eliminated in rural areas down to the scale of $1 : 100'000$. Displacement is therefore often the only possible operation, unless we backtrack and undo corrections performed at the intra-line level, which probably caused the conflict in the first place.

Road Displacement Constraints

- **Enforce Minimal Distance** This constraint expresses the goal of displacement. It ensures that symbols are sufficiently separated to perceive the roads as individual entities.

- **Enforce Legibility of Junctions** At junctions, legibility is not guaranteed by the former constraint, as the minimal distance between objects is no longer a valid criteria for connected lines. The constitution of junctions is often covered by the widest road symbol, hiding the orientation of coinciding roads. This information however is of great interest in topographic maps, as they are often used by motorists.

- **Preserve Shape** This constraint limits the admissible modifications to a geometry. Strictly speaking, it is prohibited to introduce bends in straight areas and to modify lines such that symbol coalescence occurs at bends; more widerly interpreted, it is only admissible to modify a road if its character is thereby not altered. We are aware that this statement is fuzzy and that verification by visual comparison is subjective.

- **Preserve Topology** Preserving the connectivity in a road network is a matter of course in traditional cartography; in digital cartography however, we have to pay attention to ensure that roads stay connected to the junction nodes they originate from and that no unintended intersections are introduced by a careless road displacement.

- **Maintain Positional Accuracy** This constraint is assumed indirectly by the preservation of shape, which leaves little room for geometrically inaccurate solutions. However, it is worth taking positional accuracy into consideration explicitly to avoid useless shifts which possibly jumble spatial relationships.

These constraints provide only the corner pillars to guide the search for a displacement solution. Solutions, satisfying these constraints, will contain no graphical conflicts, and are thus, probably, acceptable in most situations. But such solutions are not perfect, as spatial relationships (i.e. alignments and relative distances) are not yet respected.

Where are the Problems in Road Displacement? To sensitize the reader to the problems arising in road displacement, we state a few questions that have to be answered for a successful solution. This list is inevitably prejudiced by the observed failure of existing algorithms, yet to be discussed in the state of the art, see Section 2.5.

Roughly, we can distinguish three sources of problems. The first category deals with questions arising from trying to find a solution to the immediate conflict (Figure 2.7). Which roads have to be displaced? How is the displacement shared among the lines? In which direction is the displacement to act? By what amount? The main problem here is that the displacement of one line can introduce new conflicts with other lines. A strategy is required that does not simply push the conflict back and forth, but actually resolves them.

A second source of problems are junctions. Junctions require a reasoning that goes beyond proximity assessment; legibility is not ensured by simply pushing close segments apart. A deeper understanding of junctions is necessary in order to modify them in a way that their constitution is not altered (see Section 4.5).

A third source of problems is the propagation of displacement along the roads. To maintain the character of the roads, the rate of cushioning of the displacements is crucial. If cushioning is not possible in the affected lines only, junctions have to be shifted also, which entails a deformation of all coinciding lines to preserve topology. The displacement is thus spread through the network.

Finally, and this is where the holistic character of generalization is important, it needs to be addressed whether it is possible to answer the questions of the identified problem spheres one after the other; the intuitive approach, that problems are first solved at the conflict spot and in junctions, and then after displacement is propagated through the lines, fails. Displacement and propagation are strongly related. It is the shape of a line that determines its 'resistance' to deformation and displacement; the willingness of lines to be displaced and to absorb displacement, varies with the character of the road. A proximit conflict is reliably solved only, if the 'backyard' of the conflict is taken into consideration (Figure 2.7).

2.3.5 Building Displacement

Building Displacement as One Generalization Operation among Others In urban block generalization, typification, elimination and aggregation of buildings are at least as important as displacement. When changing to smaller scales, tiny buildings are enlarged to a minimum threshold, and the minimum distance between

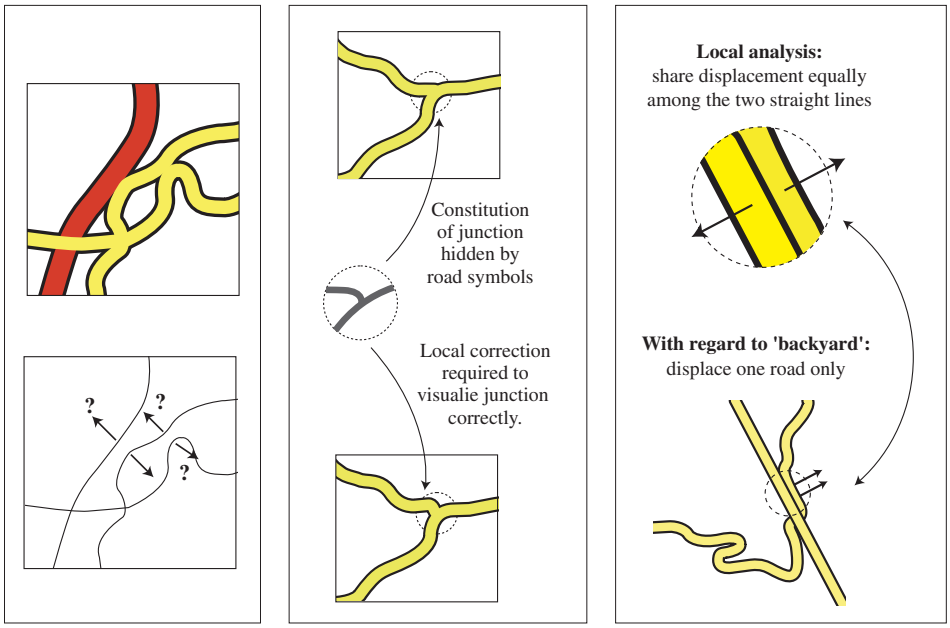


Figure 2.7: Some problems in road displacement: Identifying displacement vectors (left), adapting junctions to the symbolization (middle), and incorporating the roads 'backyard' for decisions (right).

buildings, and also to the roads, increases; it is therefore often not possible to find a solution exclusively by displacement. Displacement is applied mainly for small change of scales and sparsely occupied partitions. The complex interactions between operators are beyond the scope of our work. A detailed analysis of urban block generalization is given in Ruas (1999).

Early in the generalization process, building shapes are changed, either to remove small crenulations in their outlines or as result of aggregation and typification. Hence, we can assume that building geometries are already optimized when we start displacement. This provides the motivation to translate them only – an assumption already stated in Section 2.3.3.

Building Displacement Constraints Based on our assumption of rigid buildings, shape constraints are no longer an issue. Also topology in not really a problem, because the bounding roads of an urban block provide fixed boundaries wherein buildings are trapped. The constraints of importance are then

- **Enforce Minimal Distance** A minimal distance is required not only between the buildings and the surrounding roads, but also between the buildings themselves. In analogy to road displacement, enforcing minimal distances means not only solving conflicts, but also avoiding generating new ones.
- **Preserve Patterns / Spatial Relationships** Buildings, as human arte-

facts, show a regularity in shape and/or distribution. It is necessary to preserve this regularity during the displacement process. We use the term 'pattern' to denote a characteristic arrangement of a set of buildings, whereas 'spatial relationship' is more widely interpreted, indicating any inter-object relation (for instance an alignment of buildings, see Figure 2.8). What is worth being preserved and what not, is difficult to say and requires an antecedent analysis.

- **Preserve Positional Accuracy** A good displacement is as small as possible.

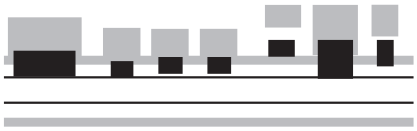


Figure 2.8: Displacement due to increased road symbol preserves alignments (after SGK Arbeitsgruppe 2001).

Where are the Problems in Building Displacement? A main problem in building displacement is the scarcity of space, this makes any modification difficult. Shifting one building may entail new conflicts with other buildings. An intelligent strategy is required to shift buildings without simply pushing the problem elsewhere.

Note that buildings are not only shifted if they are in conflict; a change in position is also required to preserve relative proximity and spatial relationships. To detect, which objects share a relation worth being protected, is by no means trivial and requires careful antecedent analysis.

2.4 Spatial Partitioning

A map is too big to be dealt with at once. Before we can apply an algorithm, it is necessary to delimit its work space. Such an extraction of the map has to cover more than the immediate conflict spot; sufficient space has to be at avaliable to analyze the given situation and to apply the required corrections. This task of decomposing the map space into working units is called *spatial partitioning*.

The delimitation of spatial partitions is a two step procedure. First, we find all the sources of conflict. Without conflicts, there is no need for corrections. For displacement, the conflict-spots originate from symbol overlaps and proximity conflicts. Yet, this very local perspective does not provide an adequate view of the problem to work with, as displacement incorporates the constraints of objects in the surroundings and acts at a more spatially contextual level. Hence, in the second stage, one has to zoom out from the conflict spot until an extraction of the map is found that provides 'enough space' for displacement.

What does 'enough space' mean in this context? The work space we choose must allow us to resolve the conflict spot(s) it contains, given the constraint, that the geometry of the work space border is subject to no changes. This guarantees that all conflicts within an area are solved and that we can deal with one problem at a time, as changing one work space does not alter the connectivity and spatial relation to the rest of the map.

The simpliest case is to allocate partitions dynamically, solving all the problems within one problem area, then paying attention to new conflicts. This works fine when only the displacement operator is applied. However, when objects are also, for instance eliminated or collapsed, these changes influence the appearance of the entire map: e.g. the density of objects is changed unevenly in different parts of the map. This drawback is overcome if the map is partitioned in advance and if the changes in one partition are communicated to neighboring ones (and if the other partitions know how to react to such input). If partitions overlap in this approach, things again become more complex. See Brazile (2000) for an analysis of partitions.

Finding partitions is not easy. It depends on the features one wishes to separate. The best studied case is the displacement of buildings in a given urban block. Given the priorities in displacement (see Section 2.3.3), one assumes that roads are not subject to displacement anymore. Therefore, we can use parcels of the road network to define the work space (Figure 2.9). Brazile (2000) and Ruas (1999) worked with this kind of partitions.

How to delimit partitions for the displacement of roads is still an unsolved problem, see e.g. Fritsch *et al.* (1998). The borders of these partitions are not again defined by roads, but by a virtual border running through the 'open space' of the map. These borders lie perpendicular to the roads and are supported by those vertices of the road network on which it is certain no displacement will be applied (see Figure 2.9). It is cumbersome to determine these vertices, as we do not know in advance which vertices are far enough from the conflict spots that they can be used as part of the skeleton for the fixed partition boundary. For this knowledge, we need to know the displacement values in the conflict spots, which are, of course, the result of the algorithm, and so not known in advance. One can estimate a magnitude of displacement and the rate of propagation based on the character of the involved roads. But this requires an analysis of the situation, which itself requires a delimitation of the work space...

We encourage researchers to tackle the problem of spatial partitioning, as a good delimitation of the workspace is a prerequisite for the successful use of algorithms. In our working examples, we devised the partitioning of maps interactively. The situations displayed in the working examples coincide with the partitions. The terms 'situation' and 'partition' are used interchangeable.

2.5 Displacement Algorithms – Classification and Review

There are many criteria according to which displacement algorithms could be classified. This review of the state of the art is structured chronologically, emphasizing the different views that came up over time. At present, there are two philosophies, varying fundamentally in how a displacement method needs to be designed:

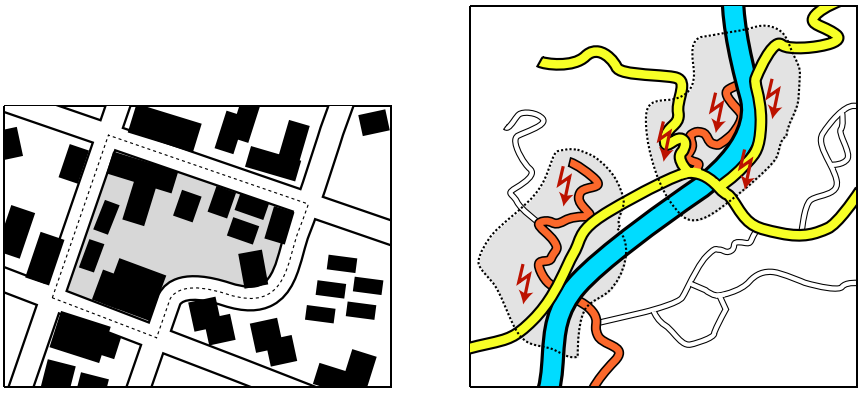


Figure 2.9: Possible partitions built for urban block generalization (left) and for road displacement (right). The conflicts in road displacement are indicated by flashes. Cartographic generalization depends on a broader reasoning than the immediate conflict spots. Partitions can consist of many conflicts, which demand common generalization. Dashed lines delimit possible partitions.

so-called sequential and global methods. This dichotomy is discussed following the state of the art. Raster based approaches are excluded from this review. We assume that raster based methods for vector data are of little use.

2.5.1 Elementary Geometric Approaches (Sequential Methods)

Displacement by Deformation Work on automated displacement of lines has a long tradition, starting in the early 1970s in Germany with work of Töpfer (1974), Gottschalk (1972) and Lichtner (1979). Their basic idea was to apply a displacement vector to each vertex of a line, such that the minimal distance between the lines was ensured. Therefore, they had to tackle the questions *which vertices* would need to be displaced? in *which direction*? and by *what amount* this correction had to be applied? To deduce this knowledge, these early approaches focused only on the conflict spot and their cartographic reasoning was purely distance oriented. Propagation, to avoid unacceptable shape distortions, was performed by a linear decay of the displacement magnitude towards zero with increasing distance from the conflict.

Nickerson (1988) took up this strategy and continued the work. He detected the importance of junctions for the displacement operation. He therefore introduced a classification of conflicts between roads, which allowed him to take the special circumstances around intersections into consideration. Furthermore, he used a triangle filter to smooth the displacement in order to reduce shape distortion. For many years, Nickerson's algorithm has been the best algorithm available for road displacement. It is well suited for the displacement of parallel lines, but fails if major displacements are required or if more than two lines share a conflict. This is because the algorithm's view is restricted to the vicinity of the conflict and it

contains no strategy of how displacement is shared among lines if more than two are in conflict. The algorithm also does not incorporate a consideration of the the shape of lines, except for the directly interfering segments.

Bader and Weibel (1997) took up the same geometric viewpoint for the displacement of polygons. They were studying to what extent the outline of polygons can be treated in the same manner as lines. The focus was on a better analysis of the conflict zone. A constrained Delaunay triangulation was used to gain a better 'feeling' of the proximity conflicts between polygons. Though they modeled the repulsion of the polygons by means of a force field (Figure 2.10), they again concentrated solely on the computation of correction vectors for the vertices making up the polygons, thereby again losing the global properties of repulsion.

Hybrid Displacement Mackaness (1994), whose approach was superceeded later by Tallis (1995) and Roberts (1997), used a radial displacement function to resolve proximity conflicts, initially between point features and later also between other types of objects. These algorithms first detect the source of the conflict and then delimit by means of cluster analysis all the other objects that are involved in the conflict. The vertices are then spread radially outwards from the conflict center, this has the advantage that object distances in the conflict center are enlarged without perturbing topology.

The method has the advantage that all objects – whatever type they are – share the displacement. If the algorithm is applied to point features or representative vertices of an object, the method displaces by translation; if it also incorporates line or polygon vertices, it computes a local deformation of the object.

Roberts (1997) states that for a loose knit of vertices, the algorithm makes the pattern grow without distorting it, and thus Gestalt is preserved. However, if the vertices are part of a larger object, e.g. a line, radial displacement destroys the overall shape, as the deformation is not controlled by the object's constraints. In particular, since the displacement is applied radially, 'ballooning' effects may occur. The method is not qualified for building displacement, as the general enlargement does not lead to a valid solution in a bounded partition.

Displacement by Translation In the mid 1990s interest arose in the displacement of buildings in urban blocks. The shape of buildings is highly constrained and the buildings' size usually does not allow shrinkage (on the contrary, they are usually enlarged beforehand). The objective is thus to translate buildings in order to resolve proximity conflicts, while maintaining inter-object relationships at the same time.

Jones *et al.* (1995) presented an algorithm that makes use of a triangulation to detect and resolve proximity conflicts. The triangulation is formed by means of all object vertices. If during the generalization process, for instance due to the enlargement of small buildings, triangles become inverted, a conflict is reported. The conflict reporting building is then displaced to bring the triangulation into a proper state again. This method has many deficiencies: displacements are not propagated, patterns and alignments are not preserved, and no strategy is implemented into how a group of conflicting buildings can be separated. But the idea to make use of a triangulation has been reapplied and readjusted many times since Jones *et al.*; the advantage of a triangulation is that it keeps track of object relationships and

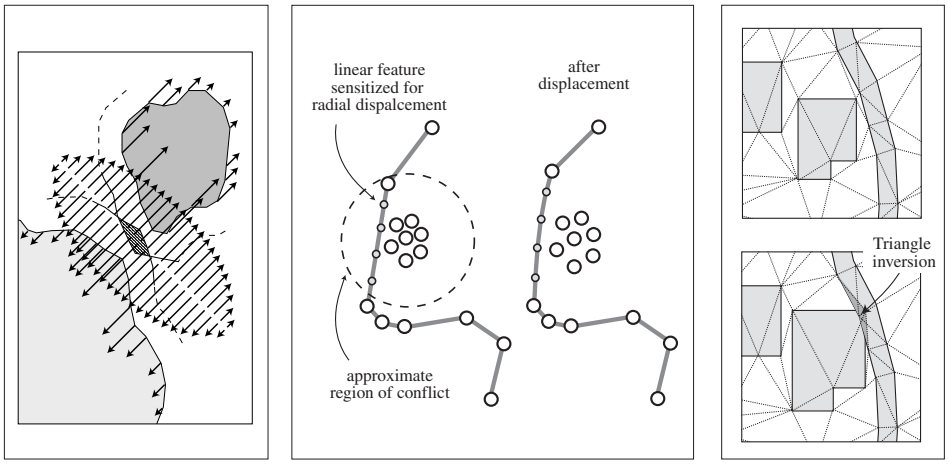


Figure 2.10: (left) Polygon displacement interpreted as object deformation in a force field (after Bader and Weibel 1997). (middle) Solving proximity conflicts by radial displacement (after Mackaness 1994). (right) Proximity conflict caused by building exaggeration detected by triangle inversion (after Jones *et al.* 1995).

stores them for the entire partition, which predefines displacement directions and accelerates computation.

An algorithm that makes use of such a triangulation is that of Ruas (1998). However, her algorithm has few other similarities with that of Jones *et al.* (1995). The difference lies not only in how she constitutes the triangulation – she uses building centroids as triangulation nodes instead of meshing any object vertices – but primarily in her cartographic reasoning guiding the process. Where Jones *et al.* argued purely locally and geometrically, she implements a complex process that carefully weights for each building the influences of its neighbors and which allows the propagation of displacements in order to roughly preserve object relationships. The triangulation is therefore used only as auxiliary data structure, simplifying and accelerating proximity measures. As the space in urban blocks is highly constrained, there are many pitfalls where simple geometric reasoning fails. Such problems have to be anticipated, and special cases must be caught. The result is a ramified procedure, where different impacts are worked out, which make the algorithm stolid and hard to re-implement. We see other drawbacks of the algorithm in the way in which it iterates through the buildings one after the other. The building with maximal conflict is taken and displaced. After that, it is held fixed and no longer involved in the displacement process. This handling makes the solution susceptible to suboptimal displacements, as the information and characteristic of the entire partition do not go into the first computations.

2.5.2 Optimization Methods (Global Methods)

Throughout the history of automated displacement research, we can observe the influence of methods developed in physics and/or engineering. The term 'force'

has been used by many researchers, whether they were of a magnetic, elastic or electronic nature. The idea of a displacement potential to model proximity conflicts was introduced by Volkert (1978) and Christ (1979) whose work was succeeded later by Jäger (1991).

The first approach in cartographic displacement that made use of an optimization algorithm was by Bobrich (1996). Bobrich suggested a spring model for the displacement of roads. Different types of springs, varying in stiffnesses, model different constraints on the line (see Figure 2.11). Positional accuracy (S_P), elongation of segments (S_e) and change in angularity (S_t) are all handled. Chains of springs thereby, for the first time in the history of road displacement, allowed the shape of lines to be modelled explicitly. To trigger the displacement, Bobrich used a raster-based displacement potential, borrowing from the work of Jäger (1991). The optimization, performed by a Downhill-Simplex-Algorithm, finds a balance between the regularization forces based upon the springs and the raster forces.

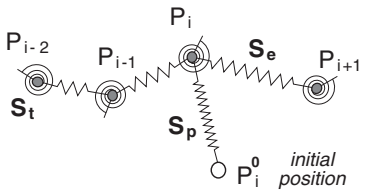


Figure 2.11: Spring model (Bobrich 1996)

Harrie (1999) has built on the idea of constraints more explicitly. He models each constraint as a mathematical expression of the coordinates, linearizes these formulae (if necessary) and adds them to an over-determined system of equations. As constraints contradict each other, one has to find a displacement that minimizes the cost. He uses the method of 'least squares adjustment', a technique that is in wide-spread use in surveying engineering and photogrammetry, to find the coordinates that evenly distribute the residuals amongst the constraints. The underlying assumption is that we can find a good solution for the entire map if we sum up the local corrections enforced by the local constraints. The approach is encouraging. Different types of objects (e.g. points, lines, polygons) can be involved in the process. Each constraint can be weighted by its importance. Orchestrating all these weights, however, might be a problem. More recent generalization algorithms that are based on least squares adjustment exist from Sarjakoski and Kilpeläinen (1999) and Sester (2001).

For buildings, Ware and Jones (1998) discuss the use of a steepest gradient method and simulated annealing. Their evaluation function is simple, limited to proximity measures between objects. The simplicial data structure helps to accelerate the evaluation of a state after each step. Even though displacement is limited to discrete locations in order to manage computability, the method is slow. Therefore, the method has been improved by reducing the number of candidate solutions, see Ware *et al.* (2000) and Lonergan and Jones (2001). An evaluation of

their method is postponed to Chapter 6.

Recently, two approaches were presented that better model the underlying characteristics of the object features involved. Burghardt and Meier (1997), and in more depth Burghardt (2000), use energy minimizing splines for line displacement. He adopts the principle of snakes, a widespread technique in computer vision, for cartographic displacement. The problem with snakes as presented by Burghardt (2000) is that the shape of lines is not explictly modeled, the junction nodes are fixed and that no cartographic behavior is imposed on the way how displacement is cushioned. Despite this, we belief this approach to be powerful. Snakes are discussed in Chapter 4.

Højholt (2000) treats the map space as an elastic body. Buildings, modeled as pebbles in a rubber sheet, are no longer floating in space, but are embedded in their surroundings, preserving proximity relations and topology. He uses a finite element method (FEM) to solve this engineering view of cartography. The method treats only the displacement part. The forces (or other boundary conditions) to release the displacement have to be found by the user beforehand. Højholt's approach is discussed in Chapter 6.

Classification It is too early to devise a good classification of optimization algorithms in cartographic displacement; this field of research is still evolving and changes rapidly.

A possible distinction of approaches is already made in mathematics. Extremal problems are discussed in the theory of nonlinear function analysis, see e.g. Zeidler (1985). One traditionally distinguishes variational problems and optimization in the narrow sense, even though extremal problems in general are based on a unified theory. Snakes and elasticity models belong to the field of variational problems. *We focus on such methods in our work.* Methods using simulated annealing and Downhill-Simplex algorithms, respectively, belong to the category of optimization in the narrow sense.

Optimization methods also require boundary or side conditions that trigger or control a solution. Side conditions, in the form of equations, are typical for the classical calculus of variations. Side conditions in the form of inequalities are typical for optimization (Zeidler 1985).

We can also distinguish algorithms regarding their iteration scheme, not only in the field of optimization techniques but also in the field of elementary geometric algorithms. Traditional algorithms usually find their solution in a single run. For instance, Ruas (1999) first moves the building with the most severe conflict and then holds the position of this building fixed during the subsequent processing. This requires the definition of a *sequence* in which the buildings are relocated. In contrast, single-iteration optimization methods perform the displacement at once. To allow simultaneous computation, it is necessary that the problem can be transformed into a global equation system[3]. This demand is met either if the system holds for all the constraints (least squares) or if the displacement potential, forcing a correction, is known *a priori* and does not vary over time (e.g. elasticity in Højholt's approach for a given displacement potential).

[3]We don't care at this point if the equation system is solved directly, e.g. with a LU-decomposition, or using an iterative technique, such as a successive over relaxation method. What counts is that only one equation system is produced.

A second type of algorithm finds the solution iteratively, striving to improve the solution over time. The solution is found by balancing displacement forces with internal regularization forces. The objects are 'shape-conserving'. As they remember/incorporate how their initial state looked, they strive towards maintaining their initial shape and position if they are not subject to displacement forces.

A third kind of method is also iterative, but there is no physical reasoning about how the next state is found. The objects are allocated randomly, an evaluation function – which qualifies the 'goodness' of the solution – is evaluated. The best solution is taken. Of course, intelligent strategies replace the random allocation of objects, see for instance Lonergan and Jones (2001). Nevertheless, there is no opportunity in such an approach to find the optima explicitly.

2.5.3 Sequential vs. Global Methods

There are substantial differences in the ideas of sequential and global methods. Sequential methods require both an analysis and understanding of a partition (not only of the conflict spot), as well as a specification of the sequence of the corrections to apply. The solution leads to a complex system of cartographic reasoning, see Ruas (1999). The advantage of such an approach lies in the knowledge we gain about the overall process and in the ability to provide for each anticipated situation an individual solution (if necessary). The drawback is the impracticability of a partition-wide analysis and thus the inability to find a good global strategy.

At the heart of global methods is the hope that the weighted sum of local corrections guarantees also a good global solution. For instance, the algorithm by Harrie (1999) produces a solution where the violation of constraints is minimized and shared amongst the objects. The developer of the algorithm is only responsible for the incorporation of the important constraints; the calculation is then left to the computer. How the solution is found is not relevant; the reasoning that led to the solution is not accessible as it is not formulated explicitly.

We take up a discussion started in Ruas (1999), p.177. She compares the advantages and drawbacks of global to sequential approaches for the displacement of buildings in urban blocks. She states that the sum of the local constraints does not necessarily lead to a good global solution. She advocates the use of sequential algorithm and asks for a local analysis of each situation before a correction is carried out. Her view is also influenced by the fact that she studies displacement as but one of a set of generalization operations, including also selection, aggregation, and typification.

In contrast to Ruas position, it is our strong belief that optimization techniques are (currently) better suited for the design of displacement algorithms. The drawback of traditional (sequential) approaches is that *no global ('synoptic') strategy enters the solution, as we are not able to provide and process the required analysis.* Already the local analysis, which never goes beyond the immediate vicinity, ends up in a complex system of measures and trade offs. Sequential methods are based on this local view and start to move buildings iteratively. Propagation alone does not compensate for the lack of a global view, as a building is fixed after it has been relocated. In contrast, using optimization algorithms, buildings interact across the whole urban block. Shifting one building possibly affects all other buildings – a model that much better represents the holistic nature of generalization and

cartographic displacement.

Clearly, the advantages are not distributed as unilaterally as that. There is no guarantee that optimization techniques provide good solutions for all problems. Many models are still incomplete. We also have to distinguish the possibilities of the two approaches for different cartographic situations. For displacement of isolated features, the traditional approach is practical; this is shown impressively by Ruas' results. Buildings are only translated, the shape of buildings does not need to be taken into consideration. For linear features, however, we strongly believe that it is not possible to define a global strategy at the beginning. The first vertex moved prejudices the entire solution. We lack many tools which would assist the understanding required to match such a far-reaching decision as the displacement of the first vertex, an understanding which is a prerequisite for developing successful sequential approaches.

Critical to the success of optimization techniques is obviously the function they optimize. If we optimize against inadequate criteria, we can not expect a good solution. This motivates us to split the group of optimization techniques. On the one hand, there are algorithms that implement a physical model. A 'natural behavior' is imposed on the objects. Højholt (2000), for example, models the map as an elastic rubber sheet, Bobrich (1996) views the road network as a set of springs. Such techniques have the advantage that they exhibit a 'natural behavior'. They are usually more resistant against non-conforming results as the overall shape is constrained by a reasonable model.

On the other hand, we have 'empty optimization' techniques. Such algorithms build the optimization function from scratch and do not build upon known models. Members of this class are the methods of Harrie (1999) and Ware and Jones (1998). It is hard to anticipate the results of such methods. A slight mis-modeling may result in strong failure. In return, they allow to model almost any demand, as they have no underlying model fending against modifications. In our work, we only make use of models that impose a 'natural behavior' on the objects[4].

2.5.4 Conclusions

Road Displacement The displacement of linear features is still an unsolved problem in cartographic generalization. If more than two lines are involved in a conflict, any classical algorithm will be far from providing acceptable results. Complex problems require, in order to settle a successful strategy, an analysis that goes beyond the direct conflict source; such analysis is not supported by any existing sequential algorithm and a failure is therein preprogrammed.

With the coming of optimization techniques, the possibility for road displacement significantly improved. Both, the spring model proposed by Bobrich (1996) and the snakes technique introduced by Burghardt (2000) are promising, but both models are not sufficiently adapted to cartographic requirements. Bobrich's work was not carried on further, and so no results are available that go beyond the simple example given in his work. The use of snakes is limited by fixed intersection nodes,

[4]With the end of this section, we give up our distinction of 'natural' and 'empty' models, as the distinction is spongy and follows no mathematical concept. Obviously, 'natural behavior' is just a constraint that is explicitly modeled and thus it is just a known criteria against which we optimize. Nevertheless, we use the distinction here as it helps to group techniques in the intricate group of optimization methods.

by missing capabilities to introduce local corrections around junctions, and by the directional indifference of snakes. An explicit modeling of lines is only partially possible.

Road displacement is still an unsolved problem.

Building Displacement For building displacement, more successful solutions exist. The algorithm by Ruas (1999) is the most advanced as it employs cartographic reasoning. The drawback is its complexity, which makes it hard to reimplement, and that the cartographic reasoning guiding the algorithm may probably fail in unanticipated situations and dense built-up blocks. Also, alignments are not explicitly modeled and thus it is not possible to protect important spatial relationships. The algorithm was not conceived exclusively for displacement, but unfolds its potential if used in combination with other operators.

We belief that the complex cartographic process defined by Ruas can be more easily formulated by means of an optimization algorithm.

Currently, existing optimization algorithms fail to maintain spatial relationships; they focus exclusive on proximity conflicts (Ware and Jones 1998, Lonergan and Jones 2001, Burghardt 2000). An exception is probably the algorithm presented by Højholt (2000), wherein spatial relationships can not be upset drastically, as they are trapped in an elastic body. Yet, this approach allows no explicit control of alignments.

Algorithms succeed in resolving proximity conflicts in urban blocks, but fail to preserve alignments and patterns sufficiently.

The potential of least square adjustments (Harrie 1999, Sester 2001) for cartographic displacement was not investigated in this thesis.

2.6 Summary of Objectives and Assumptions

Motivation Displacement is an important operation in the process of cartographic generalization. Displacement is still performed interactively by human cartographers. National Mapping Agencies are interested in replacing this time consuming and costly task. The Swiss Topographic Institute, for instance, is currently evaluating how road displacement algorithms could ease their work for producing topographic maps at a scale of 1:250'000 from base maps of at the scale of 1:100'000.

So far, attempts to automate cartographic displacement failed. No algorithm exists for a cartographically acceptable solution of proximity conflicts between linear features. Techniques for the solution of proximity conflicts in urban blocks exist; yet, they fail to preserve important spatial relationships and patterns, such as alignments.

Objectives It is our goal to present algorithms for cartographic displacement that outperform existing solutions. We therefore improve existing algorithms, especially for the displacement of roads. New ideas and algorithms are proposed where we see the chance to overcome shortcomings of existing algorithms.

We will investigate optimization algorithms only. Based on the analysis of the literature as well as results of the AGENT project with sequential techniques, we

belive that optimization methods are better suited for modeling displacement than traditional sequential methods.

The experience we gain using these optimization methods will be of hopefully of value beyond simply the development of the displacement algorithm themselves. Other applications in the domain of geographic information science may benefit also from these techniques, as our modeling of linear and aerial objects is a general issue in this field.

Assumptions Two constraining assumptions simplify our work. First, we assume that maps are already decomposed into partitions. Such partitions reduce the amount of data to a reasonable size and delimit the borders inside which the displacement problem can be solved. Second, we deal with pre-generalized data and concentrate only on the displacement step. The road network is already pruned and the remaining roads have been simplified. Small buildings have been removed from the database and the outline of buildings simplified. We do not involve other operations even though the elimination and typification of buildings in an urban block often seems more likely than displacement.

Chapter 3

Mathematical Background

This chapter outlines the basics of the mathematical techniques used in later chapters. It focuses on the calculus of variations and gives an overview of finite elements methods (FEM).

The reader may ask what is the use of sketching the mathematical background for topics that have been treated exhaustively in applied mathematics and engineering and that are discussed in depth in dozens of books? Researchers in the field of cartographic generalization are mainly cartographers, geographers, engineers, and computer scientists, with considerably variable background knowledge. It is our goal to provide the fundamental ideas of the methods used in later chapters to a reader not familiar with these topics. 20 pages of text – maybe half an hour of reading – turns nobody into an expert on the finite element method. But we hope it is possible to illustrate the ideas to enable everybody to see the advantages, but also the caveats and constraints, of the underlying model.

We abstain from strict definitions and formal proofs. Readers interested in more details and formalisms are referred to the end of the chapter, where further readings are provided.

3.1 Exploring the Idea of Functionals

Definition A mapping $J : \mathcal{V} \rightarrow \mathcal{R}$, which assigns each function f of a linear vector space $\mathcal{V}$ a real number $J(f) \in \mathcal{R}$, is called a *functional* on $\mathcal{V}$.

A function f maps each number x a corresponding number $f(x)$. On the other side, a functional J determines for each function $f(x)$ a corresponding scalar value $J(f(x)) \in \mathcal{R}$. Courant and Hilbert (1993) used the term 'Functionfunction' (translated from the German 'Funktionenfunktionen') to express that the varying thing inside a functional is not a set of unknowns, but a function itself. For more informations on functionals see, for example, Walter (1994), Schwetlick and Kretzschmar (1991) and Henwood and Bonet (1996).

Example 1 A common operation that has the properties of a functional, is the integral. If we define[1]

$$J(f) = \int_G f dx\,, \qquad \text{with } f \in C^1(G) = \mathcal{V} \text{ and } G \subset \mathcal{R}, \text{ connected},$$

then, J defines a functional. We can put any function $f \in \mathcal{V}$ in the functional and obtain a real value. We try $f(x) = x^2$ on $G = [1, 3]$:

$$J(f(x)) \;=\; J(x^2) \;=\; \int_1^3 x^2 dx \;=\; \frac{x^3}{3}\Big|_1^3 \;=\; \frac{26}{3}\,.$$

Example 2 A manifold of natural problems leads to expressions using functionals. We proceed with a famous example that illustrates the importance of functionals. Assume two given points A and B, B shall be lower than A. Depending on the shape of the path that connects A and B, how long does it take a given object of mass $m = 1$ to move from A to B? The object shall be subject to gravitation only and movement along the path is frictionless.

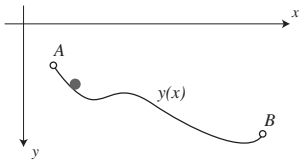


Figure 3.1: An object under gravitation moves along a path free of friction $y(x)$. The path connects two given points A and B, where B lower than A.

We choose the coordinate system as illustrated in Figure 3.1. The y-axis points downwards. The path is described by $y(x)$. The total time T of descent from $A = (x_A, y_A)$ to $B = (x_B, y_B)$ is given by

$$T \;=\; \int_0^l dt \;=\; \int_0^l \frac{ds}{v}\,,$$

where the element of curve-length is ds, l is the length of the path, and v the velocity of the object. From differential geometry we know that $ds = \sqrt{1 + y'^2}dx$. We obtain herewith

$$T \;=\; \int_0^l \frac{ds}{v} \;=\; \int_{x_A}^{x_B} \frac{\sqrt{1 + y'^2}}{v}\, dx\,. \tag{3.1}$$

[1] It is helpful at this point to introduce a standard definition and notation to express the degree of continuity of a function f. If f is continuous, we say that f has C^0 continuity, $f \in C^0$; if, in addition, the first derivative is continuous, we have C^1 continuity; if the second derivative is also continuous, we say $f \in C^2$, and so on.

The theorem of the conservation of energy postulates

$$\frac{1}{2}mv^2 + mgy = const \quad \Longrightarrow \quad v^2 + 2gy = const \quad \Longrightarrow \quad v = \sqrt{2gy}\,. \tag{3.2}$$

Inserting (3.2) in (3.1) leads to the functional

$$T(y) = \int_{x_A}^{x_B} \frac{\sqrt{1+y'^2}}{\sqrt{2gy}}\,dx \tag{3.3}$$

subject to the boundary conditions $y(x_A) = y_A$, $y(x_B) = y_B$. The boundary conditions impose that a chosen path passes through the start- and endpoints A and B.

Assume, for instance, the points $A = (0,0)$, $B = (4,1)$, and use the straight line $y(x) = 1/4x$, which satisfies the boundary conditions (meaning it connects the given endpoints). The particle's time to move along this path is then

$$\begin{aligned} T &= \int_0^4 \frac{\sqrt{1+(1/4)^2}}{\sqrt{2gx/4}}\,dx \overset{g=10}{\underset{\downarrow}{=}} \sqrt{\frac{17}{16\cdot 5}}\int_0^4 \frac{1}{\sqrt{x}}\,dx \\ &= \sqrt{\frac{17}{80}}2\sqrt{x}\Big|_0^4 = \sqrt{\frac{17}{5}}\,. \end{aligned}$$

Generally, we use the notation $F\,(x, y(x), y'(x))$ to denote the function in the integral. So, the functional in this example is written as

$$T(y) = \int_G F(x, y(x), y'(x))dx \qquad \text{with} \quad F(x,y,y') = \frac{(1+y')^{1/2}}{(2gy)^{1/2}}\,.$$

From a researchers point of view, equation (3.3) is not very exciting. More interesting is the question, which path y, that connects the given points A and B, allows the object to reach B in *minimal* time. This problem is known as the *Brachistochrone Problem*. It was first solved by Jakob Bernoulli in 1697. It is the starting point of the calculus of variations.

3.2 Calculus of Variations

3.2.1 Introduction

When discussing the characteristics of functions, the question arises, for which values is a function is maximized or minimized. The necessary condition for a local extremum of a differentiable function $y = y(x)$ is

$$\frac{dy}{dx} = 0\,. \tag{3.4}$$

All the points x meeting this requirement are called stationary. A deeper analysis then provides information on whether such a point is a minima, a maxima or an inflection point.

In analogy, one can ask for the function that extremizes a functional. E.g., how must the path $y(x)$ in Example 2 be constructed so that the particle moves in

minimum time. A necessary condition for stationarity of functionals is expressed by the *Euler Equation*. This equation is a criteria analogous to (3.4), but valid in the world of functionals. Before we derive the Euler Equation, we give a helpful theorem, also referred to as fundamental theorem of the calculus of variations.

3.2.2 The Fundamental Theorem of the Calculus of Variations

The following theorem lies at the heart of many variational expressions. It is intuitively clear and easy to prove.

Theorem 1 The only continuous function $f \in C^0$ for which

$$\int_a^b f(x)\, v(x) = 0\,, \qquad \forall v(x) \in C^0\,,$$

is $f(x) \equiv 0$, for $a < x < b$.

The statement that the integral should be zero *for all* $v(x)$ is very demanding. The theorem can be seen by assuming, the contrary, that $f(x_0) > 0$ for some $x_0 \in [a, b]$. Since $f(x)$ is, by hypothesis, continuous, there exists a neighborhood $\mathcal{U} = (x_0 - \epsilon, x_0 + \epsilon)$ of x_0 wherein $f(x) > 0$. Now pick a function $v(x)$ such that $v(x) > 0$ for $x \subset (x_0 - \epsilon, x_0 + \epsilon)$ and $v(x) = 0$ for all other $x \in [a, b]$. Then $\int f(x)v(x) > 0$, which is a contradiction. A similar conclusion is reached if we assume $f(x) < 0$.

3.2.3 Euler-Lagrange Equation

We now come to a central question in the calculus of variations. Which function $y(x) \in \mathcal{V}$ extremizes a given functional $J(y)$, where $\mathcal{V}$ shall define a space of sufficiently smooth functions. The first observation we make is that not every $y(x)$ in the entire function space $\mathcal{V}$ is involved in our search for an extremum. The boundary conditions prescribe the values of $y(x)$ on the interval boundaries. Incorporating this constraint in our problem, we end up with the following question:

Which function $y(x)$ minimizes the functional

$$J = J(y) = \int_{x_1}^{x_2} F(x, y, y')dx\,,$$

with respect to the boundary conditions

$$y(x_1) = y_1 \qquad \text{and} \qquad y(x_2) = y_2\,.$$

Our steps to answer this question follow Kleiber and Breitkopf (1993) and Fliessbach (1996). Assume, we had somehow found the function $y(x)$ that minimizes the functional J. We introduce an auxiliary function $\eta(x) \in C^1$ defined over the same interval $[x_1, x_2]$, and such that $\eta(x_1) = \eta(x_2) = 0$. If now ϵ is a small parameter, then the function

$$y(x) + \epsilon\eta(x)$$

neighbors on the function $y(x)$, which is minimizing the functional. The deviating function $y(x) + \epsilon\eta(x)$ always passes through the given boundary values, since

$\epsilon(x_1) = \epsilon(x_2) = 0$, see Figure 3.2. Furthermore for any $\eta(x)$ the varied function equals the extremizing function when we set $\epsilon = 0$. The function $\epsilon\eta(x)$, which represents small changes in the function $y(x)$ over the interval $[x_1, x_2]$ is called the variation of $y(x)$. In literature, the notation $\partial y(x) := \epsilon\eta(x)$ is used.

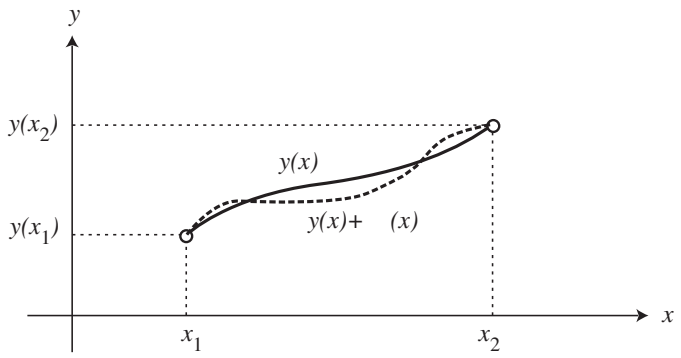


Figure 3.2: A function $\epsilon\eta(x)$ is added as variation to the given function $y(x)$ with fixed boundary values. This deviating function is denoted by $y + \epsilon\eta$.

For $y(x)$ and $\eta(x)$ both fixed, $J(y+\epsilon\eta)$ is a simple function of ϵ, and is denoted with $J(\epsilon)$. This function $J(\epsilon)$ must be minimal for $\epsilon = 0$:

$$\forall\eta(x) : \; J(y + \epsilon\eta) \text{ is minimal for } \epsilon = 0\,. \tag{3.5}$$

If this condition did not hold for any $\eta_0(x)$, we could choose $\epsilon \neq 0$ to make $J(y+\epsilon\eta_0)$ smaller than $J(y)$, contradicting the premise that y minimizes $J(y)$.

Condition (3.5) implies

$$\left(\frac{dJ(y + \epsilon\eta)}{d\epsilon}\right)_{\epsilon=0} = 0\,, \quad \forall\eta(x)\,. \tag{3.6}$$

This is the condition for stationarity, and herewith, in analogous to (3.4), a necessary condition for a local extremum.

To evaluate (3.6), we write

$$J(y + \epsilon\eta) \;=\; \int_{x_1}^{x_2} F(x, y + \epsilon\eta, y' + \epsilon\eta')\, dx\,.$$

Taylor series expansion of $J(y + \epsilon\eta)$ around $F(x, y, y')$ then gives us

$$J(y + \epsilon\eta) = \int_{x_1}^{x_2} \left(F(x, y, y') + \frac{\partial F}{\partial x}(x, y, y') + \frac{\partial F}{\partial y}(x, y, y')\epsilon\eta(x) + \frac{\partial F}{\partial y'}(x, y, y')\epsilon\eta'(x)\right) dx$$

where the terms of order ϵ^2 and higher were omitted. Assuming that $F(x, y, y')$ is differentiable, we get

$$\left(\frac{dJ(y+\epsilon\eta)}{d\epsilon}\right)_{\epsilon=0} = \int_{x_1}^{x_2} \frac{\partial F}{\partial y}(x,y,y')\eta(x) + \frac{\partial F}{\partial y'}(x,y,y')\eta'(x)$$

and inserting this in condition (3.6) results in

$$\begin{aligned} 0 &= \left(\frac{dJ(y+\epsilon\eta)}{d\epsilon}\right)_{\epsilon=0} \\ &= \int_{x_1}^{x_2} \left(\frac{\partial F}{\partial y}(x,y,y')\eta(x) + \frac{\partial F}{\partial y'}(x,y,y')\eta'(x)\right) dx \\ &\overset{\text{Partial Integration}}{=} \overbrace{\frac{\partial F}{\partial y'}(x,y,y')\eta(x)\Big|_{x_1}^{x_2}}^{=0} + \\ &\qquad \int_{x_1}^{x_2} \left(\frac{\partial F}{\partial y}(x,y,y') - \frac{d}{dx}\frac{\partial F}{\partial y'}(x,y,y')\right)\eta(x)\,dx \qquad \forall \eta(x)\,. \end{aligned}$$

This condition must hold for any arbitrary $\eta(x)$. Thus the term in the braces of the integrand must be zero, as seen in the fundamental theorem of the calculus of variations (Theorem 1). Therefore

$$\frac{\partial F(x,y,y')}{\partial y} - \frac{d}{dx}\left(\frac{\partial F(x,y,y')}{\partial y'}\right) = 0\,, \qquad \textbf{Euler Equation}\,. \tag{3.7}$$

This is the Euler equation in the calculus of variation. It is a second order ordinary differential equation for the sought function $y(x)$. If a function $y(x)$ satisfies this equation, the functional $J(y)$ is stationary. Herewith, the Euler equation is a necessary condition for an extremum. We do not discuss sufficient conditions here.

Example 3: Shortest Path Let us consider the minimization problem of finding the curve $u(x)$ of minimal length connecting two points $A = (0, y_0)$ and $B = (1, y_1)$. The functional defining the length takes the form

$$J(y) = \int_0^1 \sqrt{1+y'^2}dx\,,$$

where we used the definition of length introduced in example 2. We minimize this functional over all functions $y(x)$ for which $y(0) = y_0$ and $y(1) = y_1$. Using the terms above we have

$$F(x,y,y') = F(y') = \sqrt{1+y'^2}\,.$$

The Euler equation thus is

$$\frac{\partial F(y')}{\partial y} - \frac{d}{dx}\left(\frac{\partial F(y')}{\partial y'}\right) = -\frac{d}{dx}\left(\frac{y'}{\sqrt{1+y'^2}}\right) = 0 \qquad \text{in } [0,1]\,,$$

together with the boundary conditions $y(0) = y_0$, and $y(1) = y_1$. Integrating, we see that this equation is satisfied if $y' = const$, and we conclude that the minimizing curve is a straight line, as expected.

Example 2 (cont.): Brachistochrone Problem We continue the Brachistochrone problem started in Example 2, Section 3.1. The brachistochrone (Nelson 1998) is the path (curve) along which a particle will slide in the shortest time from one point to another, lower point (not directly beneath the first).

How must this path be constructed such that the particle slides in shortest time, from, say, point A to point B? The time taken by the article is, see (3.3),

$$T = \int_A^B \frac{\sqrt{1+y'^2}}{\sqrt{2gy}} dx = F(y, y') .$$

This term is introduced in the Euler equation. We gain a separable differential equation, which can be solved analytically. As this is rather cumbersome, we abstain from doing so. We refer to Fässler (1995) or Baierl and Heinl (1993) for more details. The solution for a case where B has nearly the same height as A is displayed in Figure 3.3. The curve describes a cycloid, which is the curve defined by the locus of a point on the circumference of a circle as the circle rolls along a straight line. The radius of the circle is defined by the boundary points A and B.

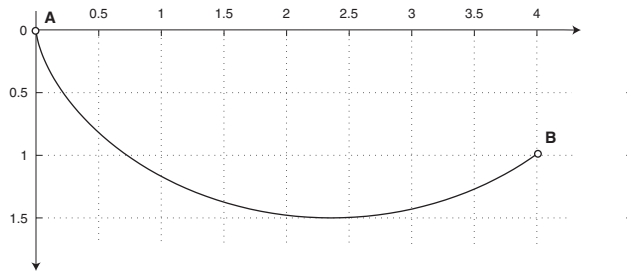


Figure 3.3: Solution of the Brachistochrone Problem: the displayed curve allows an object to move from $A = (0, 0)$ to $B = (4, 1)$ in the shortest time possible. The curve describes a cycloid.

3.2.4 Weak Variational Form

When dealing later with the finite element method, it will be convenient to translate a differential equation into an equivalent but slightly different form. This form is called *the weak variational form* of a differential equation and is derived in this paragraph.

For ease of understanding, we apply the steps directly to a one-dimensional boundary value problem. The ideas are easily generalized to other problems, different boundary conditions and two dimensions. Assume the differential equation

$$\begin{cases} u''(x) = -f(x) , & x \in \Omega = [a, b] , \\ u(a) = 0, \; u(b) = 0 . \end{cases} \tag{3.8}$$

According to the fundamental theorem of the calculus of variation (see Theorem 1), u is a solution of (3.8) if and only if

$$\int_a^b (u''(x) + f(x))\, v(x)\, dx = 0\,, \qquad \text{for } \textit{all } v(x) \in \mathcal{V}\,. \tag{3.9}$$

Since the solution is known at both a and b (where $u(a) = 0$, $u(b) = 0$), it is usual and convenient to choose $v(x)$ to be zero at these points. We rearrange the terms in (3.9) and apply partial integration to the left hand term

$$\begin{aligned} -\int_a^b u''(x)v(x)dx &= \int_a^b f(x)v(x)dx\,, \\ -\underbrace{u'(x)v(x)|_a^b}_{=0} + \int_a^b u'(x)v'(x)dx &= \int_a^b f(x)v(x)dx\,, \end{aligned}$$

which leads to the following condition

$$\int_a^b u'(x)v'(x)\, dx = \int_a^b f(x)v(x)\, dx\,, \qquad \forall v(x) \in \mathcal{V} \;\text{ with } v(a) = v(b) = 0\,. \tag{3.10}$$

Statement (3.10) is the differential equation (3.8) changed into *weak variational form*. The problem, although changed in form, is not any easier to solve (Henwood and Bonet 1996), but proves to be useful for the finite element method.

Some notations will ease reading in later chapters. The terms in (3.10) are often written as

$$\begin{aligned} &B(u,v) = L(v)\,, \\ &\text{where} \quad B(u,v) = \int_a^b u'v'dx \quad \text{and} \quad L(v) = \int_a^b fvdx\,, \end{aligned}$$

where $B(u,v)$ denotes a bilinear form and $L(v)$ a linear functional.

3.2.5 The Coherence of Differential and Variational Formulation

There is a deeper coherence between variational and differential form. This paragraph sketches some relations between these formulations.

The calculus of variations deals with the question, which function extremizes a given functional. For instance, which function $y(x)$ minimizes the integral

$$J(y) = \int_0^1 F(x, y, y')\, dx \tag{3.11}$$

among all functions $y(x)$ in the interval $[0, 1]$ satisfying the boundary conditions $y(0) = y_0$ and $y(1) = y_1$. We refer to $F(x, y, y')$ as the *Lagrangian* function.

We established with the Euler equation (see section 3.2.3) a necessary condition for a function $y(x)$ to be an extremizer of J

$$\frac{\partial F(x,y,y')}{\partial y} - \frac{d}{dx}\left(\frac{\partial F(x,y,y')}{\partial y'}\right) = 0\,. \tag{3.12}$$

So it turns out that the solution of the minimization problem (3.11) also solves a differential equation 3.12 that is related to the Lagrangian. These two formulations are not a mathematical gimmick. The variational formulation expresses a different view of problems in mechanics and physics in general than the differential formulation does.

Euler and Lagrange created the calculus of variation in the 18th century. Their variational view disengages from the thoughts of 'cause and effect' and postulates a *final cause*. Final cause suggests a purposive intelligence guiding all things and is an expression of the simplicity and rationality of the world (Euler and Leibniz).

It was Hamilton who extended the ideas of Euler and Lagrange in the 19th century. *Hamiltons principle* states that the dynamics of certain physical systems may be characterized as stationary points of the Lagrangian function that represents the difference of the kinetic and potential energies (Eriksson *et al.* 1996).

This only strives the complexity of the materia – after all, modern physics may be posited entirely in variational form. For many problems however, the Lagrangian simply denotes the energy stored in a system. The *Principle of Minimum Potential Energy* states that amongst all geometrically possible configurations which a body can take up, the true one, corresponding to internal and external equilibrium of forces, is identified by a minimum value of potential energy. Hence, the minimization of a functional often corresponds to the *minimization of energy*. As some applications presented in later chapters are naturally posed as problem of energy minimization, the formulation in variational form is crucial.

The following lemma gives another insight in the coherence of variational and differential formulations. It gives an explicit formula how a differential equation is translated to variational form.

Theorem 2 In what follows the functional B is assumed to be bilinear, symmetric and positive definite, and L is assumed to be linear.

(1) A function u satisfies its boundary conditions $u(0) = u_0$, $u(1) = u_1$ and

$$B(u, v) = L(v) \qquad \text{for all } v \text{ with } v(0) = v(1) = 0\,.$$

(2) A function u satisfies its boundary conditions $u(0) = u_0$, $u(1) = u_1$ and minimizes the functional

$$V(u) = \frac{1}{2}B(u, u) - L(u)\,.$$

Theorem: (1) $\Longleftrightarrow$ (2)

The interested reader may find the proof of this theorem in Ciarlet (1991). The proof is straightforward and follows the steps used to prove the Euler equation. An example clarifies the power of this abstraction.

Example: Elastic Bar The model of an elastic bar is discussed in structural mechanics (see also Section 3.3.2). The extension $u(x)$ of an elastic bar with modulus of elasticity E under a load w is given by the differential equation

$$\begin{cases} -Eu'' = w\,. \\ u(0) = u_0, \quad u(1) = u_1\,. \end{cases} \tag{3.13}$$

The problem translated to *weak variational form*, see paragraph 3.2.4, is

$$\underbrace{\int_0^1 Eu'v'dx}_{B(u,v)} = \underbrace{\int_0^1 wvdx}_{L(v)} .$$

The underbraces show the terms used in Theorem 2. B is bilinear, symmetric and positive definite; L is linear. According to Theorem 2, solving (3.13) is identical to minimizing

$$\begin{aligned} V(u) &= \frac{1}{2}B(u,u) - L(u) \\ &= \frac{1}{2}\int_0^1 Eu'u'dx - \int_0^1 wudx \end{aligned} \tag{3.14}$$

$V(u)$ gives the *potential energy* stored in the elastic bar. According to the principle of minimal potential energy, the elastic bar will find a form where its potential energy is minimized,again expressing the equivalence stated in Theorem 2. Applying the Euler equations turns the variational formulation back to the differential form (3.13).

3.3 Finite Element Method

3.3.1 Meet the Finite Element Method

What is the Finite Element Method (FEM) There is no agreement on what the FEM is. According to Huebner *et al.* (1995), the finite element method is a numerical analysis technique for obtaining approximate solutions to a wide variety of engineering problems. Other books see the FEM only as special case of trial functions within a broader context (Bronstein and Semedjajew 1996). Common to all the description is that the FEM is embedded in a system for solving differential equations, mostly partial differential equations. The solution of a differential equation is a function that describes the behavior of the sought field variable over the problem domain (and maybe also over time if an evolving system is observed). A possible *scalar* field variable could be temperature or pressure. Its distribution could be studied over an aircraft wing (its problem domain). Other engineering tasks may ask for a *vector* field. E.g. we may ask for the deformation of a bridge under load, where the deformation in each point is given by a vector in 3-dimensional space. The FEM is a tool that provides *numerical* approximations of the sought field variable – though it may happen for particular cases that the FEM provides exact results.

Finite Differences vs. Finite Elements Several approximate numerical analysis methods have evolved over the years. A commonly used method is the finite difference schema, which gives a pointwise approximation to the governing equations (Figure 3.4). The model is formed by writing difference equation for an array of grid points. If we encounter irregular geometries or an unusual specification of boundary conditions, we find that finite difference techniques become hard to use (Huebner *et al.* 1995).

Unlike the finite difference method, which envisions the solution region as an array of grid points, the finite element method envisions the solution region as built up of many small, interconnected elements. The basic premise of the FEM is that a solution region can be analytically modeled or approximated by replacing it with an assemblage of discrete elements. Particular to the FEM is the ability to formulate solutions for individual elements before putting them together to represent the entire problem.

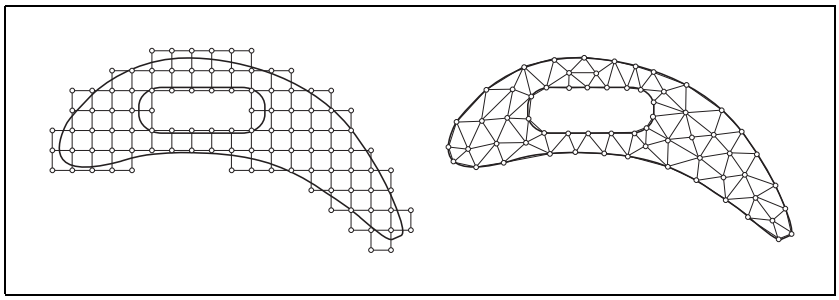


Figure 3.4: Finite difference (left) and finite element (right) discretization of a turbine blade profile (after Betten 1997)

Formulation of Element Properties The fundamental goal is to formulate the properties of individual elements. There are basically three different approaches, see Huebner *et al.* (1995). The first way to obtain element properties stems from structural analysis and is limited to relatively simple problems. We will use this *direct approach* to model elasticity in two dimensions. A second approach relies on the calculus of variations and involves extremizing a functional (*variational approach*). As already sketched in section 3.2.5, for problems in solid mechanics the functional turns out to be the potential energy. The third, and most versatile approach for deriving element properties has its basis in mathematics and is known as the *weighted residuals approach*. It will be used to solve the differential equations arising in dealing with snakes in Chapter 4.

Structure of this Introduction At first glance, the FEM is probably confusing, as new terms come into play. However, the method is easy to use and has the advantage that, once understood for a simple one-dimensional problem, it is easily generalized to more complex issues. The same simple problem will hold throughout this introduction; it is first described in detail in the next section. In section 3.3.3, we sketch Ritz' idea that lies at heart of any application. Once this concept is understood, we give step-by-step cookbook-like instructions for the FEM.

3.3.2 Problem Statement: the Elastic String Example

Consider an elastic string lying at rest on a table. The physical behavior of the string is defined by its length l, weight per unit length w, modulus of elasticity E, and cross-section A, where we assume that E and A are constant within the string.

The string is now turned onto its vertical and fixed to the ceiling. The elastic string is stretched naturally under its weight, due to gravity. We could also think of other tangential loads, which add to gravity. We denote the magnitude of this tangential load by $f(x)$. We can also fix the low hanging end of the string at distance $l + b$ from the ceiling, elongating it additionally, see Figure 3.5.

We ask for the extension $u(x)$, by which a general point x is displaced when turning the string from rest to its vertical. The extension will clearly vary according to the original position of the point. The point fixed at the ceiling will not be extended at all ($u(0) = 0$). Then, starting from the top, expansion arises. The bottom point is displaced by $u(l) = b$.

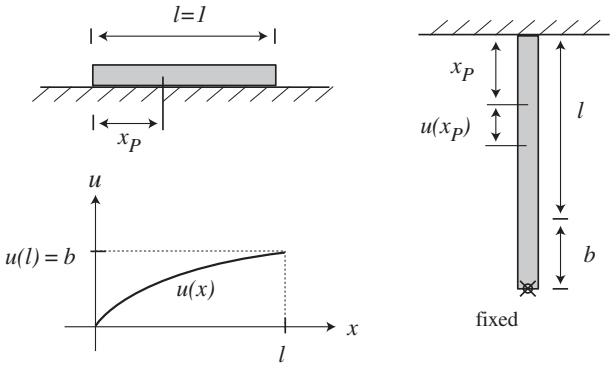


Figure 3.5: Initial (left) and final (right) position of an elastic string. The extension u at a given point x is due to a load $f(x)$ and due to an imposed fixation. The problem is to find the extension $u(x)$ which, among other things, must satisfy the boundary conditions $u(0) = 0$ and $u(l) = b$ (left bottom).

The physical laws which the solution obeys are defined by the differential equation

$$AE\frac{d^2u(x)}{dx^2} = -Af(x) \tag{3.15}$$

with boundary conditions

$$u(0) = 0 \qquad \text{and} \qquad u(l) = b\,.$$

We shall ignore the fact that this problem has an exact solution (by direct integration) and proceed to find an approximate solution using the ideas to establish below.

We first translate the problem in a more convenient form. The energy stored in the string due to an arbitrary extension u is given by, cf. (3.14),

$$J(u(x)) = \int_0^l \left[\frac{AE}{2} \left(\frac{du(x)}{dx} \right)^2 - Af(x)u(x) \right] dx + C_l \,, \tag{3.16}$$

where the boundary conditions of the differential equation require

$$u(0) = 0 \qquad \text{and} \qquad u(l) = b \,.$$

C_l denotes a simple constant that depends on the length and weight of the string. The principle of minimal potential energy states that at equilibrium, the extension u is such that the potential energy is minimal. Thus we seek a solution of the variational problem (3.16) with $J(u) \rightarrow min$. This connection between differential form (3.15) and variational form (3.16) was already stated in section 3.2.5.

In what follows, we set $A = 1$, $E = 1$ and $l = 1$ to facilitate reading.

3.3.3 The Ritz Method: Illustrating the Basic Idea

Assume a problem stated in variational form, consider the problem stated above. For the purpose of minimizing the given functional, we can insert any function that meets the boundary conditions into the variational form, and compute therewith the functionals value. Obviously, there is no reasonable chance of finding the correct (minimizing) function by chance. The pool of test functions is too large. Or talking formally: the function that minimizes a given functional is chosen from an infinite function space.

Ritz now proposes a method to approximate this infinite function space $\mathcal{U}$, wherein we seek for the minimizing function $u(x)$, by a finite-dimensional function space $\mathcal{U}_n$, where $\dim(\mathcal{U}_n) = n + 1$. A set of basis functions

$$u_0(x), u_1(x), \ldots, u_n(x)$$

is chosen in $\mathcal{U}_n$, and by combination of these basis functions the *approximate field variable* $\tilde{u}$ is constructed

$$u(x) \approx \tilde{u}(x) = u_0(x) + \sum_{i=1}^{n} c_i u_i(x) \tag{3.17}$$

where $c_i \in \mathcal{R}$ give the parameters yet to be determined. The basis functions are chosen such that the boundary conditions are satisfied regardless of the choice of the parameters c_i. To guarantee this, u_0 is set to meet the boundary conditions while all other functions u_i, $i = 1, \ldots, n$, are set to vanish on the boundary.

Instead of searching a function u that minimizes the functional $J(u)$ in the infinite function space $\mathcal{U}$, we restrict our search to all functions $\tilde{u}$ in $\mathcal{U}_n$. If $\mathcal{U}_n$ happens to still contain the correct solution, our restriction leads to no loss of quality. Otherwise we can not expect to find the correct solution u anymore, but find only an approximation $\tilde{u}$, where $\tilde{u} \neq u$.

The advantage of this approach becomes clear when we introduce the function $u(x)$ from (3.17) in the functional $J(u)$. As the basis functions u_i are fixed and

known, the only variables are c_i. The functional $J(u(x))$ turns into a function $J(c_1, \ldots, c_n)$, varying only in parameters c_i, a finite set of unknowns.

We now seek the parameters c_i that minimize the function $J(c_1, \ldots, c_n)$, or make it at least stationary. As well established in calculus, all the partial derivatives with respect to the unknowns c_i have to vanish in a minimum

$$\frac{\partial J}{\partial c_i} \stackrel{!}{=} 0\,, \qquad (i = 1, \ldots, n)\,.$$

As the function $\tilde{u}$ consists of a linear combination of the u_i, all the parameters c_i show up squared in (3.16). After differentiation, the unknowns c_i are again linear, forming a linear equation system for the c_i. Having solved for c_i, the approximation $\tilde{u}$ is constructed by inserting these c_i in (3.17).

Example Consider the elastic string differential equation

$$\begin{cases} \dfrac{d^2u(x)}{dx^2} = -f(x), & 0 \le x \le 1 \\ u(0) = 0,\ u(1) = b\,. \end{cases}$$

For this example we further assume that $b = 1$ and $f(x) = kx$, $k \in \mathcal{R}$ a constant. The problem is equivalent to finding the function $u(x)$ that minimizes the functional

$$J(u) = \int_0^1 \left[\frac{1}{2}\left(\frac{du}{dx}\right)^2 - kx\,u(x)\right] dx\,, \qquad \text{where } \ u(0) = 0,\ u(1) = 1\,.$$

We approximate the solution function u by $\tilde{u} \in \mathcal{U}_n$. For example, let us choose $n = 2$ (so $\mathcal{U}_n = \mathcal{U}_2$). A basis of $\mathcal{U}_2$ is constructed in a simple and convenient way by polynomials (see Figure 3.6)

$$u_0(x) = x\,, \qquad u_1(x) = (1 - x)x\,, \qquad u_2(x) = (1 - x)x^2\,.$$

We use a linear combination of this functions to build the trial function

$$\tilde{u}(x) = u_0 + c_1u_1 + u_2v_2\,, \qquad (0 \le x \le 1)\,.$$

Note that the polynomials have been chosen such that the trial function satisfies the boundary conditions for any choice of c_1 and c_2, as $u_0(0) = 0$ and $u_0(1) = 1$ while $u_1(x)$ and $u_2(x)$ vanish on this boundary. This discretization of the solution space turns the functional into a function of the variables c_1 and c_2

$$\begin{aligned} J(\tilde{u}) &= \int_0^1 \left[\frac{1}{2}\left(\frac{d\tilde{u}}{dx}\right)^2 - kx\tilde{u}(x)\right] dx \\ &= \int_0^1 \left[\frac{1}{2}\left(\frac{d(u_0(x) + c_1u_1(x) + c_2u_2(x)}{dx}\right)^2 - kx(u_0(x) + \right. \\ &\qquad \left. c_1u_1(x) + c_2u_2(x)\right] dx \\ &= J(c_1, c_2) \end{aligned}$$

We require c_1, c_2 be chosen to minimize J. Hence, we apply from calculus the conditions for a minima

$$\frac{\partial J}{\partial c_1} = 0 \qquad \text{and} \qquad \frac{\partial J}{\partial c_2} = 0\,.$$

We compute this equation for the first term. For sake of simplicity, the differentiation with respect to c_1 is applied before integration. Therefore

$$\begin{aligned}
\frac{\partial J}{\partial c_1} &= \int_0^1 [u_1'(u_0' + c_1 u_1' + c_2 u_2') - kxu_1]\,dx \\
&= \int_0^1 \left[(1-2x)(1 + c_1(1-2x) + c_2(2x-3x^2)) - kx(1-x)x\right] dx \\
&= \frac{1}{3}c_1 + \frac{1}{6}c_2 - \frac{1}{12}k
\end{aligned}$$

and analog in c_2. This results in the conditions

$$\begin{aligned}
\frac{\partial J}{\partial c_1} &\stackrel{!}{=} 0 = \frac{1}{3}c_1 + \frac{1}{6}c_2 - \frac{1}{12}k \\
\frac{\partial J}{\partial c_2} &\stackrel{!}{=} 0 = \frac{1}{6}c_1 + \frac{2}{15}c_2 - \frac{1}{20}k
\end{aligned}$$

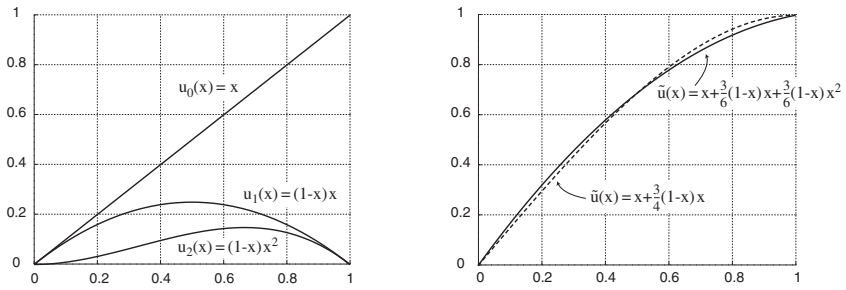


Figure 3.6: The basis functions u_0, u_1, u_2 (left). Approximation $\tilde{u} \in \mathcal{U}_2$ as linear combination of these basis functions (solid line, right). The computed curve happens to be the correct solution too. For comparison, the approximation using one basis function less is also displayed, i.e. $\tilde{u} \in \mathcal{U}_1$ (dashed line, right).

This linear equation system is solved with $c_1 = k/6$ and $c_2 = k/6$. Our approximate solution is thus

$$\tilde{u}(x) = x + \frac{k}{6}(1-x)x + \frac{k}{6}(1-x)x^2\,.$$

Figure 3.6 shows the computed solution. It happens that our approximation $\tilde{u}$ coincides with the exact solution u. This is only possible because the subspace $\mathcal{U}_2$ contained the exact solution. This can not be expected to happen in general. Having this principle idea in mind, we give now a cookbook-like tutorial how to proceed when applying the Finite Element Method.

3.3.4 Workflow of the FEM

Regardless of the application, the solution of a continuum problem by the finite element method always follows an orderly step-by-step process.

1. **Discretize the continuum** Divide the continuum (the solution region) into elements. For 1-dimensional problems, this is simply a division of the axis into segments. In higher dimensions there exists a variety of possible elements. In 2-dimensions, a triangulation is often the most convenient division.

2. **Select interpolation function** Choose an interpolation function to represent the field variable over the element. In the example above, where we did not perform a discretization into elements – the elastic string was handled as one single element – we have chosen a function consisting of polynomials. Simple linear interpolation is often used.

3. **Find the element properties** Determine the matrix equations expressing the properties of the individual elements. It is an advantage of the FEM that we can formulate these equations for each element, before we put them together.

4. **Assemble the element properties to obtain the system equations** All the matrix equations expressing the behavior of the individual elements are combined to express the behavior of the entire system.

5. **Impose the boundary conditions** As the assemblage in step 4 is done by summing up the properties of a 'general element', the boundary conditions are not yet included. At this stage, known nodal values and known nodal forces are imposed.

6. **Solve the system equation** The assembled system is a linear equation system that needs to be solved for the unknown values defining the linear combination of the trial functions.

3.3.5 Step by Step: An Example

We illustrate the FEM again by means of an elastic string. The problem is defined by the differential equation

$$\begin{cases} \dfrac{d^2u(x)}{dx^2} = -f(x)\,. \\ \text{with boundary conditions } \; u(0) = 0,\; u(1) = b\,. \end{cases}$$

The solution u that satisfies this differential equation also minimizes the functional

$$\begin{cases} J(u) = \displaystyle\int_0^1 \left[\frac{1}{2} \left(\frac{du}{dx} \right)^2 - f\,u \right] dx\,. \\ \text{with boundary conditions } \; u(0) = 0,\; u(1) = b\,. \end{cases}$$

In what follows, we use $f(x) \equiv 1$.

Step 1: Discretize the continuum We decompose our string into 3 elements $e_i, \quad (i = 1, 2, 3)$ of equal length $h = l/3$. This defines 4 nodes $x_0, \ldots, x_3$ (see Figure 3.7).

Step 2: Select interpolation function The selection of an interpolation function corresponds to the reduction to a subset of the function space wherein we seek the approximate solution $\tilde{u}$ in the Ritz method.

The FEM is the Ritz method with a special set of basis functions (Bronstein and Semedjajew 1996). For instance, instead of using polynomials which are defined over the entire problem domain (such as u_0, u_1, u_2 in the previous example), we use functions with small support. A function with small support denotes a function that is vanishing everywhere except at a very local region of the problem domain. In our example, we use 'hat-like' functions as basis functions. They are illustrated in Figure 3.7. The approximation function $\tilde{u}(x)$ of the sought function $u(x)$ is then given by

$$\tilde{u}(x) = \sum_{i=0}^{3} c_i u_i \,. \tag{3.18}$$

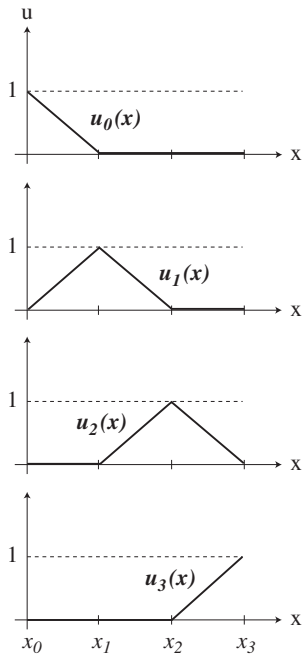


Figure 3.7: 'Hat-like' functions $u_0, \ldots, u_3$ with small support are chosen as basis functions.

An advantage of the FEM is the property that the computations are performed on each element alone before the elements are assembled to a bigger whole. The

approximation $\tilde{u}$ stated in (3.18) does not yet support this element-wise treatment sufficiently. All the functions are incorporated in the analysis of $\tilde{u}$.

Here, the special choice of 'hat-like' basis functions is valuable. As these functions all have small support, we do not need to incorporate them all in order to establish the element equations. The approximate solution $\tilde{u}$ on, for instance, element 1 – denoted $\tilde{u}^{e_1}$ – is set up by u_0 and u_1 only. All the other basis functions vanish over said element and have no influence there.

Focussing now on an arbitrary element e_k with nodes x_{k-1} and x_k. The only functions that influence $\tilde{u}^{e_k}$ are the functions u_{k-1} and u_k which are defined over e_k by

$$u_{k-1}^{e_k}(x) = \frac{x_k - x}{h}\,, \qquad u_k^{e_k}(x) = \frac{x - x_{k-1}}{h}\,, \qquad x \in [x_{k-1}, x_k]\,, \tag{3.19}$$

with $h = x_k - x_{k-1}$. Hence, the run of the curve $\tilde{u}^{e_k}$ over element e_k is given by

$$\tilde{u}^{e_k}(x) = c_{k-1} u_{k-1}^{e_k} + c_k u_k^{e_k}\,. \tag{3.20}$$

$\tilde{u}^{e_k}$, as sum of two linear functions, is again linear. The overall approximation $\tilde{u}$ an assemblage of straight lines and is thus a piecewise linear function, see Figure 3.8. The connection between elements is guaranteed as each coordinate c_i uniquely defines the value of the function $\tilde{u}$ at the element nodes x_i, i.e. $\tilde{u}(x_i) = c_i$.

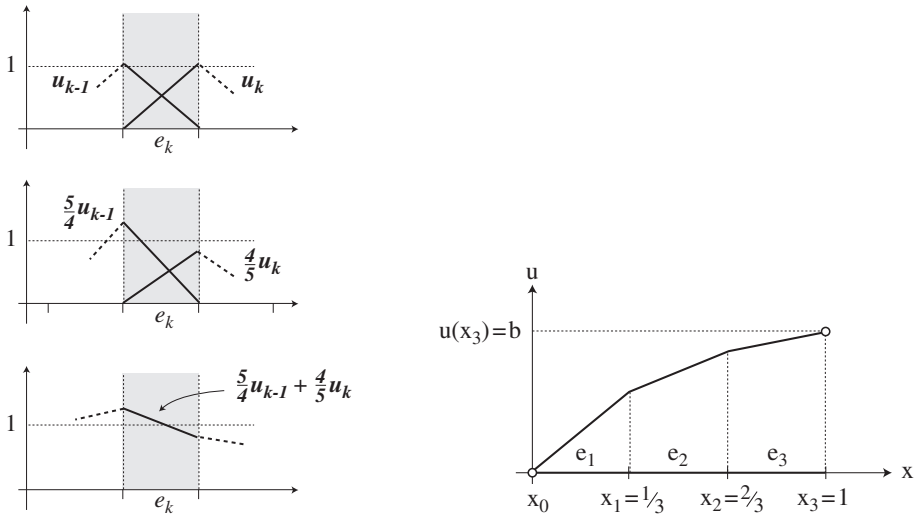


Figure 3.8: The approximated solution $\tilde{u}^e$ over an arbitrary element e is determined by two basis functions only (left). The sum of the two functions results in a linear interpolation over the element. The assembly of several element gives a piecewise linear approximation (right).

In the applications developed in this work, and also in finite element analysis in general, one would not define the functions over the entire problem domain.

Instead, one chooses only the interpolation functions valid over a general element. In the example above we had chosen a linear interpolation function that was defined by two basis functions. Higher order interpolation functions are often required. We make use of them in chapter 4. Clearly, the interpolation functions over an element cannot be chosen arbitrarily; rather, certain continuity requirements must be met to ensure convergence. We refer the reader to the various text-books that list many of the well suited interpolation functions.

As previously pointed out, the FEM allows an element by element computation which makes implementation on a computer fairly easy. To stay as general as possible, we continue with the computations on a *general element*. The characteristics and boundary conditions of the real elements are not respected until the end, when setting up the final equation system.

Step 3: Find the element properties We use the variational formulation to compute the energy stored in an arbitrary element e_k. The energy is given by

$$\begin{aligned} J^{e_k} &= J^{e_k}(u^{e_k}(x)) \\ &= F_{k-1}c_{k-1} - F_k c_k + \int_{x_{k-1}}^{x_k} \left[\frac{1}{2} \left(\frac{du^{e_k}}{dx} \right)^2 - u^{e_k} \right] dx \,. \end{aligned} \tag{3.21}$$

The terms $F_{k-1}c_{k-1}$ and $F_k c_k$ come into play, as we allow for the possibility of external forces at either end.

We continue the computation on a general element. This approach makes sense, as we are in reality allowed to carry out the computation 'element-by-element' and sum them at the end. The justification of this procedure is given by the well known theorem from calculus, which states that an integral may be evaluated by summing the values of the integral over subintervals which make up the original interval.

We substitute now our approximated function $\tilde{u}^{e_k}$, see (3.20), into equation (3.21) and gain

$$\begin{aligned} J^{e_k}(u^{e_k}(x)) &\approx J^{e_k}(\tilde{u}^{e_k}(x)) \\ &= F_{k-1}c_{k-1} - F_k c_k + \cdots \\ &\quad \int_{x_{k-1}}^{x_k} \left[\frac{1}{2} \left(\frac{d}{dx}(c_{k-1}u_{k-1}^{e_k} + c_k u_k^{e_k}) \right)^2 - (c_{k-1}u_{k-1}^{e_k} + c_k u_k^{e_k}) \right] dx \end{aligned}$$

It is important to note that J^{e_k} is now a function of c_{k-1} and c_k, the unknown coordinates of the basis functions. To minimize this function, differentiation with respect to c_i is required. Theoretically, it does not matter whether the integration in (3.22) is carried out before the differentiation or after, but the manipulation is slightly easier if the differentiation is performed first. Therefore

$$\begin{aligned} \frac{\partial J^{e_k}}{\partial c_{k-1}} &= T_{k-1} + \int_{x_{k-1}}^{x_k} \left[\frac{1}{2}\, 2 \left(\frac{c_{k-1}u_{k-1}^{e_k}(x)}{dx} + \frac{c_k u_k^{e_k}(x)}{dx} \right) \frac{u_{k-1}^{e_k}}{dx} - u_{k-1}^{e_k} \right] dx \\ &= T_{k-1} + \int_{x_{k-1}}^{x_k} \left[\frac{1}{h^2}(c_{k-1} - c_k) - \frac{x_k - x}{h} \right] dx \\ &= T_{k-1} + \frac{1}{h}(c_{k-1} - c_k) - \frac{h}{2} \end{aligned}$$

where we used $x_k - x_{k-1} = h$ and $-\frac{d}{dx}u_{k-1}^{e_k}(x) = \frac{d}{dx}u_k^{e_k}(x) = 1/h$. Applying the same step for differentiation with respect to c_k gives

$$\frac{\partial J^{e_k}}{\partial c_k} = -T_k + \frac{1}{h}(-c_{k-1} + c_k) - \frac{h}{2}\,.$$

We write this result in matrix form (*local stiffness matrix*)

$$\begin{bmatrix} \frac{\partial J^{e_k}}{\partial c_{k-1}} \\ \frac{\partial J^{e_k}}{\partial c_k} \end{bmatrix} = \begin{bmatrix} \frac{1}{h} & -\frac{1}{h} \\ -\frac{1}{h} & \frac{1}{h} \end{bmatrix} \begin{bmatrix} c_{k-1} \\ c_k \end{bmatrix} - \begin{bmatrix} \frac{h}{2} - T_{k-1} \\ \frac{h}{2} + T_k \end{bmatrix}$$

Step 4: Assemble the element properties Originally, before any assumption concerning the form of the solution $u(x)$ was made, the search was for a function $u(x)$ to minimize J. Once the finite element trial function $\tilde{u}(x)$ has been substituted for $u(x)$, J depends only on the parameters c_i, so the search for a curve became a search for values of c_i that minimize J

$$\begin{aligned} J(u) &\approx J(\tilde{u}) \\ &= J^{e_1}(\tilde{u}^{e_1}) + J^{e_2}(\tilde{u}^{e_2}) + J^{e_3}(\tilde{u}^{e_3}) \\ &= J^{e_1}(c_0, c_1) + J^{e_2}(c_1, c_2) + J^{e_3}(c_2, c_3) \end{aligned}$$

In analogy to the summation of the element energies, we add the local stiffness matrices. All that is needed is to work out the appropriate positions in the global matrix. This process is known as *assembling*.

$$\text{add} \quad e_1 = e_1(c_0, c_1) \quad \longrightarrow \quad \begin{bmatrix} \frac{\partial J}{\partial c_0} \\ \frac{\partial J}{\partial c_1} \\ \frac{\partial J}{\partial c_2} \\ \frac{\partial J}{\partial c_3} \end{bmatrix} = \begin{bmatrix} \frac{1}{h} & -\frac{1}{h} & . & . \\ -\frac{1}{h} & \frac{1}{h} & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} - \begin{bmatrix} \frac{h}{2} - T_0 \\ \frac{h}{2} + T_1 \\ . \\ . \end{bmatrix}$$

$$\text{add} \quad e_2 = e_2(c_1, c_2) \quad \longrightarrow \quad \begin{bmatrix} \frac{\partial J}{\partial c_0} \\ \frac{\partial J}{\partial c_1} \\ \frac{\partial J}{\partial c_2} \\ \frac{\partial J}{\partial c_3} \end{bmatrix} = \begin{bmatrix} \frac{1}{h} & -\frac{1}{h} & . & . \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & . \\ . & -\frac{1}{h} & \frac{1}{h} & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} - \begin{bmatrix} \frac{h}{2} - T_0 \\ \frac{2h}{2} \\ \frac{h}{2} + T_2 \\ . \end{bmatrix}$$

$$\text{add} \quad e_3 = e_3(c_2, c_3) \quad \longrightarrow \quad \begin{bmatrix} \frac{\partial J}{\partial c_0} \\ \frac{\partial J}{\partial c_1} \\ \frac{\partial J}{\partial c_2} \\ \frac{\partial J}{\partial c_3} \end{bmatrix} = \begin{bmatrix} \frac{1}{h} & -\frac{1}{h} & . & . \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & . \\ . & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \\ . & . & -\frac{1}{h} & \frac{1}{h} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} - \begin{bmatrix} \frac{h}{2} - T_0 \\ \frac{2h}{2} \\ \frac{2h}{2} \\ \frac{h}{2} + T_3 \end{bmatrix}$$

We require each row to be 0, as J is only minimized, when its derivatives with

respect to all unknowns vanish. Using $h = 1/3$ we gain

$$\underbrace{\begin{bmatrix} 3 & -3 & 0 & 0 \\ -3 & 6 & -3 & 0 \\ 0 & -3 & 6 & -3 \\ 0 & 0 & -3 & 3 \end{bmatrix}}_{\boldsymbol{K}} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \frac{1}{6} - T_0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} + T_3 \end{bmatrix}}_{\boldsymbol{f}}$$

where $\boldsymbol{K}$ is usually denoted as *global stiffness matrix* and $\boldsymbol{f}$ as *global force vector*, even if the problem does not deal with mechanics. This stiffness matrix has some interesting properties which are relevant to our work.

Firstly, $det(\boldsymbol{K}) = 0$, indicating that the matrix is singular. So the system equation will not have a solution, unless the force vector also sums to zero, in which case there are infinite many solutions. If the elements of $\boldsymbol{f}$ sum to zero, then the string is in equilibrium under the external forces and its own weight. In this case one displacement can be chosen arbitrarily and the others can then be solved relative to it. In the next step, we show how $\boldsymbol{K}$ becomes regular when boundary conditions are imposed.

Secondly, $\boldsymbol{K}$ is usually symmetric. This is usually true in finite element analysis, and for our problems it is always the case. It expresses the principle of *action = reaction*.

Finally, $\boldsymbol{K}$ has a banded structure, in that the non-zero terms lie in a band symmetrically placed about the leading diagonal. The bandwidth is small for 2-dimensional applications. For triangles the bandwidth depends on the technique by which node numbers are allocated. There has been research how this numbering is best performed to minimize the bandwidth, for example Everstine (1979) and Sloan (1986). Minimizing the bandwidth is important. One can reduce the storage of the elements within the bandwidth and accelerate the solution of the equation system. System equations in engineering usually are huge in size.

Step 5: Impose the boundary conditions The boundary conditions have been disregarded so far. They are now incorporated into the system. We classify boundary conditions into two types (Henwood and Bonet 1996):

1. **Essential Boundary Conditions**, where the value of the problem variable itself is known. These are 'built in' to the solution and consequently are satisfied exactly. In the problem of the elastic string, the extension $u(0) = 0$ defines an essential boundary condition.

2. **Natural Boundary Conditions**, which involve the derivative of the unknown, and are introduced through the functional. In the problem of the elastic string, a pre-described tension $T = AEdu/dx$ would define a natural boundary condition. In general, the finite element solution only approximates to this type of boundary condition.

In our example, we only have the essential boundary conditions $u(0) = 0$ and $u(1) = 1$. We require that also our approximate solution $\tilde{u}(x) = \sum_{i=0}^{3} c_i u_i(x)$

passes through these values. As in a node x_i only the function u_i is not vanishing, we gain $c_0 = 0$ and $c_3 = 1$. These values are introduced in the system equations

$$\begin{bmatrix} 3 & -3 & 0 & 0 \\ -3 & 6 & -3 & 0 \\ 0 & -3 & 6 & -3 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ c_1 \\ c_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{6} - T_0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} + T_3 \end{bmatrix} . \tag{3.22}$$

Introducing the boundary conditions in the system equation is performed easily by hand. See Huebner *et al.* (1995) how this procedure is best translated into a computer program. The incorporation of boundary conditions modifies the global stiffness matrix. It renders the matrix nonsingular.

Step 6: Solve the system equation Rearranging (3.22) gives

$$\left.\begin{array}{rcrcrcl} 6c_1 & - & 3c_2 & & & = & \frac{1}{3} \\ -3c_1 & + & 6c_2 & - & 3 & = & \frac{1}{3} \end{array}\right\} \quad \Rightarrow \quad c_1 = \frac{8}{18}, \quad c_2 = \frac{14}{18} .$$

The calculated c_i are the coordinates of the basic functions $u_i(x)$, which define as linear combination the approximate solution $\tilde{u}(x)$. The result $\tilde{u}(x)$ is displayed in Figure 3.9, together with the real solution $u(x)$. In our example, it happens that the values c_i are exactly the unknown extensions at the places u_i. Clearly, this is not the case in general. If we had applied cubic interpolation functions over each segment the coordinates would not have matched one-to-one to the unknown field variable.

Once we have determined the unknowns c_i, we can solve (3.22) for the forces T_0 and T_3. These forces are necessary to fix the string on the ceiling and at the bottom to a given extension $u(1) = 1$.

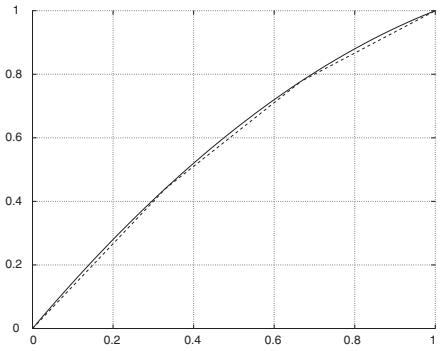


Figure 3.9: The extension $u(x)$ of an elastic string subject to the boundary conditions $u(0) = 0$ and $u(1) = 1$. The numerical solution computed in the example consists of piecewise linear functions (solid). The correct solution is plotted for comparison (dotted).

3.3.6 Galerkin's Method

The method presented in the last section illustrated all steps required in a finite element analysis. We pointed out in section 3.3.1 that there are different ways how the element equations can be computed. The variational approach used before is illustrative and fits for many problems. However, applied scientists and engineers encounter practical problems for which classical variational principles are unknown. A general applicable technique is Galerkin's method of weighted residuals. We establish the principle here. The method is used in Section 4.2.

We shall assume a given differential equation with boundary conditions. We showed in section 3.2.4 how the fundamental lemma of variational calculus is applied to give the related abstract variational boundary-value problem of finding $u(x) \in \mathcal{U}$ such that

$$B(u, v) = L(v)\,, \qquad \forall v \in \mathcal{V}\,, \tag{3.23}$$

where $B(u, v)$ is a continuous bilinear form on $\mathcal{U} \times \mathcal{V}$, $\mathcal{U}$ and $\mathcal{V}$ function spaces (or more precise Hilbert spaces), and L a linear functional on $\mathcal{V}$.

As for Ritz Method, Galerkin's Method starts with a discretization of $\mathcal{U}$ into $\mathcal{U}_n$. An analytic solution $\tilde{u} \in \mathcal{U}_n$ is searched for that hopefully corresponds with the real solution $u \in \mathcal{U}$ sufficiently. However, we still have the drawback, that we can not test the solution candidates $\tilde{u}$ against all functions $v \in \mathcal{V}$, as $\mathcal{V}$ is itself again an infinite-dimensional function space. Therefore Galerkin's method narrows the requirement (3.23) to

$$B(\tilde{u}, \tilde{v}) = L(\tilde{v}) \qquad \forall \tilde{v} \in \mathcal{V}_m\,. \tag{3.24}$$

A finite dimensional subspace $\mathcal{V}_m$ is used in place of $\mathcal{V}$. We assign subscripts n and m to imply the dimension of the subspaces, which generally relates to the mesh size. The functions $\tilde{u}$ are called trial function, the functions $\tilde{v}$ test functions.

Instead of using different function spaces $\mathcal{U}_n$ and $\mathcal{V}_m$, the trial functions are directly applied as test functions, meaning $\mathcal{V}_m = \mathcal{U}_n$ and $\dim(\mathcal{U}_n) = \dim(\mathcal{V}_m) = n + 1$.

Let u_i $(i = 0, \ldots, n)$ be a basis of $\mathcal{U}_n$. Then both the solution $\tilde{u} \in \mathcal{U}_n$ and the test functions $\tilde{v} \in \mathcal{U}_n$ can be represented as a linear combination of these basis functions

$$\tilde{u} = \sum_{i=0}^{n} c_i u_i\,, \qquad \tilde{v} = \sum_{i=0}^{n} b_i u_i\,, \qquad b_i, c_i \in \mathcal{R}\,,$$

The condition expressed in (3.24) becomes

$$B(\sum_{i=0}^{n} c_i u_i, \sum_{i=1}^{n} b_i u_i) \;=\; L(\sum_{i=0}^{n} b_i u_i)\,, \qquad \forall b_i \in \mathcal{R}\,. \tag{3.25}$$

From linear algebra, we now that we do not have to test this condition against each $\tilde{v}$. If the test meets the criteria for all basis functions, it will also fulfill the criteria for any other $\tilde{v} \in \mathcal{U}_n$. Hence, condition (3.25) is fulfilled, if

$$\sum_{i=0}^{n} c_i B(v_i, v_j) = L(v_j)\,, \qquad j = 0, 1, \ldots, n\,,$$

where we used the bilinearity of $B(\cdot,\cdot)$. These equations define the global system equation $\boldsymbol{K}\boldsymbol{c} = \boldsymbol{f}$, where the terms in the matrix $\boldsymbol{K}$ and in the vector $\boldsymbol{f}$ are given by

$$K_{ij} = B(v_i, v_j) \qquad \text{and} \qquad f_j = L(v_j)\,.$$

Underlying Idea Before presenting an example, we sketch the idea behind Galerkin's approach. The solution u of (3.23) holds for any v. So it surely also meets the condition for $\tilde{v} \in \mathcal{V}_n \subset \mathcal{V}$

$$B(u, \tilde{v}) = L(\tilde{v})\,, \qquad \forall \tilde{v} \in \mathcal{V}_n\,. \tag{3.26}$$

Subtracting (3.26) from (3.24), we obtain

$$B(u - \tilde{u}, \tilde{v}) = 0\,, \qquad \forall \tilde{v} \in \mathcal{V}_n\,.$$

We may interpret the result as an orthogonality condition: the error $e = u - \tilde{u}$ of the Galerkin approximation is 'orthogonal' to the subspace $\mathcal{V}_n$ with respect to the bilinear mapping $B(\cdot,\cdot)$. This approach is also known as *Method of Weighted Residuals* (Hungerbühler 1997).

Example Reconsider the elastic string example given by the differential equation

$$\frac{d^2u(x)}{dx^2} = -f(x) \qquad \text{with boundary conditions } \; u(0) = 0,\; u(l) = b\,. \tag{3.27}$$

We introduce the finite dimensional subspace $\mathcal{U}_3$ ($n = 3$). This subspace is spanned by a set of basis-functions $u_0, \ldots, u_3$, for which we use the same linear functions as in the example of Section 3.3.5. The approximate solution $\tilde{u} \in \mathcal{U}_3$ will be constructed as linear combination of these basis functions. Therefore

$$\mathcal{U}_3 = span(u_0, \ldots, u_3) \qquad \text{and} \qquad \tilde{u}(x) = \sum_{i=0}^{3} c_i u_i\,. \tag{3.28}$$

We focus on a general element e_k of length h, connecting the nodes x_{k-1} and x_k, where $x_k - x_{k-1} = h$. As suggested by Galerkin's Method, we apply the basis functions directly as test functions and cast (3.27) in integral form

$$\int_{x_{k-1}}^{x_k} \tilde{u}''(x)\; u_j(x)\; dx = -\int_{x_{k-1}}^{x_k} f(x)\; u_j(x)\; dx\,, \qquad j = 0, \ldots, 3\,. \tag{3.29}$$

We apply integration by parts to the left term and obtain

$$\begin{aligned}\int_{x_{k-1}}^{x_k} \tilde{u}''(x)\, u_j(x)\, dx &= \tilde{u}'(x)\, u_j(x)\,|_{x_{k-1}}^{x_k} - \int_{x_{k-1}}^{x_k} \tilde{u}'(x)\, u_j(x)\, dx\,, \quad j = 0, \ldots, 3 \\ &= \tilde{u}'(x_k) - \tilde{u}'(x_{k-1}) - \int_{x_{k-1}}^{x_k} \tilde{u}'(x)\, u_j(x)\, dx\,, \quad j = 0, \ldots, 3\end{aligned}$$

Inserting this result in (3.29), we have

$$\underbrace{\int_{x_{k-1}}^{x_k} \tilde{u}'(x)\; u_j(x)\; dx}_{B(\tilde{u}, u_j)} = \tilde{u}'(x_k) - \tilde{u}'(x_{k-1}) + \underbrace{\int_{x_{k-1}}^{x_k} f(x)\; u_j(x)\; dx}_{L(u_j)}\,.$$

The first two terms on the right hold the forces applied on the element nodes. They balance out except at the ends. In the following, we concentrate on an inner element and omit these nodal forces. We insert (3.28) and rewrite due to the bilinearity of B

$$\sum_{i=0}^{3} c_i B(u_i, u_j) = L(u_j)\,, \qquad j = 0, \ldots, 3\,.$$

Over element e_k all functions u_i vanish except for u_{k-1} and u_k. Thus, the entries from e_k in the stiffness matrix are formed to

$$\underbrace{\begin{bmatrix} K_{k-1,k-1} & K_{k-1,k} \\ K_{k,k-1} & K_{k,k} \end{bmatrix}^e}_{\boldsymbol{K}^e} \underbrace{\begin{bmatrix} c_{k-1} \\ c_k \end{bmatrix}^e}_{\boldsymbol{c}^e} = \underbrace{\begin{bmatrix} f_{k-1} \\ f_k \end{bmatrix}^e}_{\boldsymbol{f}^e}$$

where

$$\begin{aligned} K_{i,j} &= B(u_i, u_j) = \int_{x_{k-1}}^{x_k} u_i'(x) u_j'(x) dx \qquad \text{and} \\ f_j &= L(u_j) = \int_{x_{k-1}}^{x_k} f(x) u_j(x) dx\,. \end{aligned} \tag{3.30}$$

We use the same parameters as in the previous example, namely $f(x) = 1$ and $h = x_k - x_{k-1} = 1/3$. Further reconsidering the basis functions $u_i(x)$ from (3.19) with derivatives $u_{k-1}'(x) = -1/h$ and $u_k'(x) = 1/h$. The entries in the local matrix for an inner element become

$$\begin{bmatrix} 1/h & -1/h \\ -1/h & 1/h \end{bmatrix}^e \begin{bmatrix} c_{k-1} \\ c_k \end{bmatrix}^e = \begin{bmatrix} h/2 \\ h/2 \end{bmatrix}^e .$$

Assembling the local matrices, imposing the boundary conditions and solving the equation system leads to the same result as computed in section 3.3.5.

3.3.7 Closing Remarks

We illustrated the FEM by means of a trivial example. The complexity will slightly increase in later chapters. Cubic interpolation functions are used in chapter 4. The model of elasticity in the plane (chapter 6) further requires a generalization into two dimensions. The search variable will then no longer be a scalar, but a (displacement) vector. However, all the work steps involved in these more complex analyses have been established in our simple example.

However, we only sought the mathematical foundation of the method. For example, how can we approximate a solution u by a piecewise function $\tilde{u} \in C^0$ when the problem requires differentiation up the second order? Or in general, which interpolation functions are suitable for which problems (an issue known as compatibility requirement)? Can we use any basic functions to span a useful subspace $\mathcal{U}$? We also put aside the problem of convergence. What happens with $h \to 0$? Closely related is the question of stability. And what about the conditioning of the matrices?

We will use the finite element method without answering these questions. The key terms behind the theory are *distributions*, *Hilbert spaces* and *Sobolev spaces*, which are not an issue of our work. We will translate and adapt generalization problems such that we can rely on the methodology provided in various textbooks.

Further Readings The book by Eriksson *et al.* (1996) covers computational differential equations in general. The theory starts at a simple level, the ideas are well motivated. Henwood and Bonet (1996) has written a gentle introduction to the finite element method. It is insomuch gentle, that some more mathematical generalizations would maybe even ease the reading. Huebner *et al.* (1995) gives practical examples and is particularly interesting when dealing with elasticity. As customary, the finite element analysis in Schaum's series (Buchanan 1995) holds many examples. A well structured introduction to the mathematical foundations of the finite element theory is available from Oden and Reddy (1976).

Chapter 4

Road Displacement Using Snakes

The issue of road displacement has been discussed in Section 2.3.4. As explained there, roads are used as an representative example of line feature displacement. The methods shown below for roads also hold great potential to be applied to other line feature classes. The first method we discuss and improve relates to snakes.

A snake is an energy-minimizing spline guided by internal constraint forces and influenced by external enforcement. The snake technique has its roots in the field of computer vision and pattern recognition; Burghardt and Meier (1997) adapted the approach to displacement in cartographic generalization.

We discuss directly the modified approach proposed for generalization. For completeness, the link to classical snakes is given in Appendix A. Section 4.2 provides a detailed description of the numerical implementation. The structure of snakes, composed of internal and external energy, leads to a natural differentiation of propagation and displacement, discussed in Sections 4.3 and 4.4. As the snakes method is established at first rather formally, both sections start with an illustrated example of the method in the field of cartographic displacement.

4.1 Energy Minimizing Splines for Cartographic Displacement

4.1.1 Energy of the Snake

Snakes are a special case of deformable models. They were originally proposed by Kass *et al.* (1987) in the field of computer vision and have been introduced to cartographic generalization by Burghardt and Meier (1997). In the following, the model adapted for the treatment of cartographic lines is formulated.

Let $\mathcal{L}$ denote a line with length l. We define displacement as a mapping

$$s \mapsto \boldsymbol{d}(s) = \big(x(s) - x^0(s), y(s) - y^0(s)\big)^T, \quad 0 \leq s \leq l$$

where x^0, y^0 denote the coordinates of the initial line $\mathcal{L}$ and x, y give the coordinates of the displaced line. In doing so, $d(s)$ is a parametric representation of the

displacement between the initial (original) line and a derived (displaced) line. We define a *deformable model* as a space $\mathcal{A}$ of admissible deformations of the line and a functional $E = E(\boldsymbol{d})$ to represent the energy of a deformation

$$E : \mathcal{A} \mapsto \mathcal{R}\,. \tag{4.1}$$

This functional represents the energy of the model and has the form

$$\begin{aligned} E(\boldsymbol{d}) &:= \int_l E_{tot}\; ds \\ &:= \int_l (E_{int} + E_{ext})\; ds \\ &:= \int_l \frac{1}{2}\left(\alpha(s)|\boldsymbol{d}'(s)|^2 + \beta(s)|\boldsymbol{d}''(s)|^2\right) + E_{ext}\; ds \end{aligned} \tag{4.2}$$

where $\boldsymbol{d}'$ and $\boldsymbol{d}''$ denote derivatives of $\boldsymbol{d}$ with respect to s and where E_{ext} is a potential associated to external influences (see below). The aim of the method is to find among all admissible deformations $\mathcal{A}$ the deformation that minimizes this energy.

Internal Energy In contrast to classical snakes in computer vision, the inner energy defined in (4.2) does not mirror the geometry of a line, but measures the difference between two lines – the initial line $\mathcal{L}$ and its geometry after propagation. Deviations of the first and second order derivatives are used as quality measures. Such a treatment was proposed by Radeva *et al.* (1995) to incorporate structural information of objects in object recognition. We owe the idea of using snakes in cartographic generalization to Burghardt and Meier (1997). The properties of the model are controlled by the parameters $\alpha(s)$ and $\beta(s)$. Their choice determines the elasticity and rigidity of the model. A discussion of these values is the subject of Section 4.3.4.

External Energy The external energy is an arbitrary value that penalizes (or rewards) a current deformation of the line based on external criteria. For the purpose of displacement, we penalize a deformation when two neighboring lines are in conflict due to symbol overlap. The exact definition of external energy is given in section 4.3.

The power of snakes lies to a large part in the interplay between external forces, pushing a snake towards admissible places of interest, and internal regularization forces, attempting to make the line resemble its initial form.

The space of admissible deformations $\mathcal{A}$ in (4.1) is restricted by the boundary conditions $\boldsymbol{d}(0)$, $\boldsymbol{d}'(0)$, $\boldsymbol{d}(l)$, $\boldsymbol{d}'(l)$, which define the displacements and line direction at the snake's head and tail. The aim is to minimize the energy stored in the snake subject to these imposed boundary conditions.

4.1.2 Minimization of a Snake's Energy

In our work, a solution of the active contour model is found using techniques of variational calculus. There are other ways to tackle the problem. Amini *et al.* (1988) and Amini *et al.* (1990) proposed the use of dynamic programming; Williams and

Shah (1992) presented the so-called Greedy algorithm to compute an energy minimum. We are not aware of the potential of dynamic programming. However, the Greedy algorithm is less suited for line displacement than the variational algorithm we will present (see, for instance, the experiments reported in Burghardt 2000).

We recall the ideas debated in Chapter 3. If $\boldsymbol{d}(s)$ is a minimizer of the functional $E(\boldsymbol{d}) = \int F(\boldsymbol{d}', \boldsymbol{d}'')ds$, then, according to the calculus of variations, it must be a solution of the associated Euler differential equations. We proved the Euler equation for a Lagrangian of form $F(s, d(s), d'(s))$. Generally, Euler's theorem extends to cases where F depends on derivatives of d of higher degree (see Berger 1991, Bronstein and Semedjajew 1996). The general Euler equations of a Lagrangian that depends on derivatives up to order n are written

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\frac{\partial F}{\partial y'} + \frac{d^2}{dx^2}\frac{\partial F}{\partial y''} - \ldots + (-1)^n \frac{d^n}{dx^n}\frac{\partial F}{\partial y^{(n)}} = 0\,.$$

By applying this generalized equation to the functional (4.2), we obtain

$$-\frac{\partial}{\partial s}\left(\alpha(s)\frac{\partial \boldsymbol{d}(s)}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta(s)\frac{\partial^2 \boldsymbol{d}(s)}{\partial s^2}\right) + \nabla E_{ext}(x(s), y(s)) = 0\,. \tag{4.3}$$

Writing the components of the vectors explicitly, we have

$$-\frac{\partial}{\partial s}\left(\alpha(s)\frac{\partial d_x(s)}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta(s)\frac{\partial^2 d_x(s)}{\partial s^2}\right) + \frac{\partial}{\partial x}E_{ext}(x(s), y(s)) = 0\,, \tag{4.4}$$

$$-\frac{\partial}{\partial s}\left(\alpha(s)\frac{\partial d_y(s)}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta(s)\frac{\partial^2 d_y(s)}{\partial s^2}\right) + \frac{\partial}{\partial y}E_{ext}(x(s), y(s)) = 0\,. \tag{4.5}$$

This is a pair of independent differential equations of the forth order, except for $\nabla E_{ext}(x(s), y(s))$. It can be solved separately for the x-direction($d_x(s) = x(s) - x^0(s)$) and the y-direction ($d_y(s) = y(s) - y^0(s)$) respectively, since E is constant within each time-step. This separation facilitates numerical treatment substantially, since the problem can be transformed into a set of linear equations. Such treatment is made possible by the choice of the inner energy: with definition (4.2), the differential equations governing the behavior of the snake become linear.

4.2 Numerical Realization

In this section we describe the steps which lead to the numerical solution of the partial differential equations (4.4) and (4.5). Solving these equations in a computer system implies a discretization of the snake's representation. This discretization of snakes in space has been traditionally performed using finite differences. A detailed description of implementation issues regarding finite differences is found in Neuenschwander (1995) and Berger (1991). Our numerical solution relies on finite elements.

Cohen and Cohen (1990) argued that using finite differences, we only have a discrete knowledge of the functions at some points of subdivision and have no information between these points. Therefore the distance between successive points must remain very small in order not to miss too much information, since the external forces are applied to a grid of points. This typically yields large linear systems

to solve. Conversely, still according to Cohen and Cohen, with the FEM, we are really working on continuous curves independently of the size of the grid used. Therefore, the studied curve is known everywhere, independently of the chosen discretization. This yields a lower algorithmic complexity and better numerical stability.

On the contrary, Burghardt (2000) commented that both finite differences and finite elements lead to almost identical solutions. We can confirm this observation, although we used different interpolation functions to Burghardt. In general, snakes in cartographic displacement do not claim to achieve high accuracy. Thus small differences do not matter much, especially since manual generalization also yields no unambiguous solutions.

The final position of a snake is reached when inner and outer energy are in balance. To emphasize this duality of inner and outer energy, we disregarded up until now the evolutionary characteristic of snakes. The movement of snakes implies a discretization in time; this evolution of the solution is described in section 4.2.2.

4.2.1 Solution of the Eulerian Equations with Finite Elements

The numerical solution of Equations (4.4) and (4.5) with finite elements follows the routine established in Section 3.3.4. The equations are solved independently from one another. To make reading easier, we set $\alpha(s) = \alpha$ and $\beta(s) = \beta$ constant. Further, in what follows, we assume that the displacements and their first derivatives vanish at both ends of the line. Thus we consider the problem

$$\begin{cases} \alpha d''(s) + \beta d^{(4)}(s) = F(x(s), y(s)) \\ d(0) = 0,\ d'(0) = 0,\ d(l) = 0,\ d'(l) = 0 \end{cases} \tag{4.6}$$

where the differentiation is with respect to s, and where d denotes d_x or d_y whichever direction is chosen. We build the associated variational problem. As demonstrated in Section 3.2.2, solving the Eulerian equation (4.6) is equivalent to finding the function $d \in H_0^2(\Omega)$ such that

$$\int_\Omega \alpha d''(s)\, v(s)\, ds + \int_\Omega \beta d^{(4)}(s)\, v(s)\, ds = \int_\Omega F(x(s), y(s))\, v(s)\, ds\,, \quad \forall v \in H_0^2(\Omega) \tag{4.7}$$

where Ω denotes the interval $[0, l]$, l the length of the line, and $H_0^2(\Omega)$ the Sobolev space of functions d such that $\int |D^m d|^2 < \infty$ for $m = 0, 1, 2$, where $D^m d$ is the m^{th} order differential of function d.

We recall the basic idea of the variational form (4.7): The terms in (4.6) are equal in some functional space if their scalar product against any vector of the space are equal. The choice of function space depends on the imposed boundary conditions.

Applying partial integration on 4.7, we obtain

$$\int_0^l d'(s)v'(s)ds + \int_0^l d''(s)v''(s)ds = \int_0^l F(x(s), y(s))v(s)ds\,. \tag{4.8}$$

We set

$$\begin{aligned} B(d,v) &:= \int_0^1 d'(s)v'(s)ds + \int_0^l d''(s)v''(s)ds \\ L(v) &:= \int_0^1 F(x(s),y(s))v(s)ds \end{aligned}$$

which corresponds to the notations used in Section 3.3.6. $B(\cdot,\cdot)$ is a bilinear form, $L(\cdot)$ a linear functional. The solution of this variational form is computed by Galerkin's method. As we can not test whether $v \in H_0^2(\Omega)$ is orthogonal to any function d, we approximate the Sobolev space $H_0^2(\Omega)$ by a finite dimensional subspace $\mathcal{V}_h \subset H_0^2(\Omega)$. This leads to the associated discrete problem: find $d_h \in \mathcal{V}_h$ such that

$$B(d_h, v_h) = L(v_h)\,, \qquad \forall v_h \in \mathcal{V}_h\,. \tag{4.9}$$

A solution d_h of this discrete problem is an approximation of the solution d of the continuous variational problem (Cohen and Cohen 1993). Having mastered this transformation of the Eulerian equation to a discrete variational problem, the finite element method provides the methodology for implementation. Hereby we make use of the cook-book like instructions established in Section 3.3.5.

Step 1: Discretization The line, resp. its displacement function, is split into elements. A natural breakdown is given by the segments which make up the line. If the line consists of N vertices, it is decomposed into $N-1$ segments, corresponding to $N-1$ elements. This discretization leads to elements of uneven size.

Step 2: Interpolation Functions We choose Hermite interpolation functions, see Figure 4.1. These cubic polynomials are defined by

$$\begin{aligned} H_{01}(s) &= 1 - 3s^2 + 2s^2\,, & H_{02}(s) &= s^2(3-2s)\,, \\ H_{11}(s) &= (x_k - x_j)s(s-1)^2\,, & H_{12}(s) &= (x_k - x_j)s^2(s-1)\,, \end{aligned} \tag{4.10}$$

$$\text{with } s = \frac{x - x_j}{x_k - x_j}\,.$$

The right-hand subscript, 1 or 2, identifies the node; the left-hand subscript indicates the derivative that takes on the value of unity (Huebner *et al.* 1995).

The function d_h over an element e, connecting x_1 and x_2 is herewith defined as

$$d_h^e = H_{01}^e d(x_1) + H_{11}^e d'(x_1) + H_{02}^e d(x_2) + H_{12}^e d'(x_2)\,.$$

The function $d_h \in \mathcal{V}_h$ is completely defined by the values of the parameters $d_h(x_i)$ and $d_h'(x_i)$ at each of the nodes x_i. The displacement function d_h over the entire line is thus given by the identity

$$d_h = \sum_{i=1}^{N} d(x_i)H_{0i} + \sum_{i=1}^{N} d'(x_i)H_{1i}\,.$$

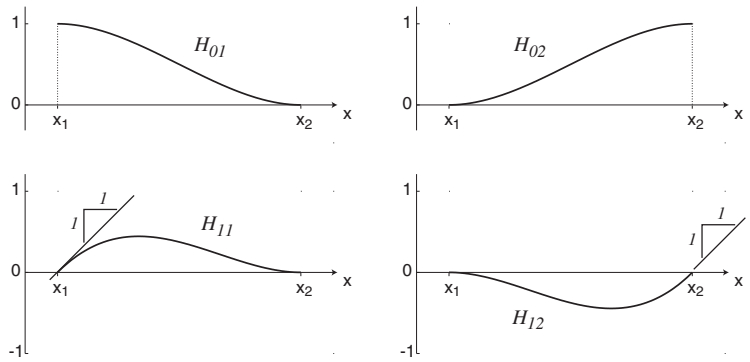


Figure 4.1: First-order (cubic) Hermite polynomials are used as interpolation functions.

As required, such a basis of functions for $\mathcal{V}_h$ has small support. The interpolation model guarantees continuity of both d and d' at the nodes, and hence it is suitable for one-dimensional problems requiring C^1 continuity (Huebner *et al.* 1995). This C^1 continuity is required to ensure compatibility at the element interfaces, as the functions under the integral contain derivatives up to the second order.

Step 3: Element Properties Following Galerkin's method, we use directly the interpolation functions H as test functions v_h in equation (4.9). The entries in the 4×4 local stiffness matrix and the local force vector defined over a general element e of length h are given by

$$K_{ij} = B(H_i, H_j) \qquad \text{and} \qquad f_j = L(H_j)\,, \qquad (1 \leq i, j \leq 4)\,,$$

where $H_1 = H_{01}, H_2 = H_{11}, H_3 = H_{02}, H_4 = H_{12}$. If we use the basis functions defined in (4.10), the symmetric local stiffness matrix is computed to

$$\begin{bmatrix} \frac{6}{5}\frac{\alpha h^2+10\beta}{h^3} & \frac{1}{10}\frac{\alpha h^2+60\beta}{h^2} & -\frac{6}{5}\frac{\alpha h^2+10\beta}{h^3} & \frac{1}{10}\frac{\alpha h^2+60\beta}{h^2} \\ \cdot & \frac{2}{15}\frac{\alpha h^2+30\beta}{h} & -\frac{1}{10}\frac{\alpha h^2+60\beta}{h^2} & -\frac{1}{30}\frac{\alpha h^2-60\beta}{h} \\ \cdot & \cdot & \frac{6}{5}\frac{\alpha h^2+10\beta}{h^3} & -\frac{1}{10}\frac{\alpha h^2+60\beta}{h^2} \\ \cdot & \cdot & \cdot & \frac{2}{15}\frac{\alpha h^2+30\beta}{h} \end{bmatrix}. \qquad (4.11)$$

Step 4: Assemblage The local stiffness matrix needs to be computed for each element, as elements vary in length h (and later also in their shape parameters α and β). They are successively added to a global stiffness matrix $\boldsymbol{K}$. This global matrix is heptadiagonal (has bandwidth $b = 7$). The forces are also computed over each element and summed to the global force vector $\boldsymbol{f}$. We end with the linear equation system

$$\boldsymbol{K}\,\boldsymbol{d} \;=\; \boldsymbol{f}\,. \qquad (4.12)$$

$\boldsymbol{d} = (d(x_1), d'(x_1), d(x_2), d'(x_2), \ldots, d(x_N), d'(x_N))^T$ is the vector of coordinates of d_h. By definition of the Hermitian functions, the values of these coordinates $d(x_i)$ resp. $d'(x_i)$ correspond to the values of the displacement function resp. its derivative. (It is this property of Hermite polynomials that motivated us to denote the coordinates directly by the otherwise misleading terms $d(x_i)$).

As the values in the stiffness matrix (4.11) depend only on the length of a segment, the matrix $\boldsymbol{K}$ is the same for computations in the x- and y-direction. In the current form, $\boldsymbol{K}$ is singular and can not be inverted; Equation (4.12) can not be solved for $\boldsymbol{d}$ as long as the boundary conditions are not integrated. This agrees with the mathematical theory, where a differential equation of type (4.3) must be supplied with 4 boundary values to be well posed.

Step 5: Impose Boundary Conditions Calculation was done under the premise that the displacement function, together with its derivative, is vanishing at the line ends. With regard to the head of the snake, the only functions not vanishing over the first element are H_{10}, H_{20}, H_{11} and H_{21}, where only the first two affect the head-point. Hence the boundary conditions are met if H_{10}, H_{20} vanish, which implies $d(0) = 0$ and $d'(0) = 0$. Analogous treatment is required for the tail of the snake. Incorporating these values makes the stiffness matrix regular. The equation system is then solved for the unknown parameters $d(x_i)$.

4.2.2 Discretization in Time

In Section 4.2.1 we computed the solution of the Eulerian equation

$$-\frac{\partial}{\partial s}\left(\alpha\frac{\partial \boldsymbol{d}}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta\frac{\partial^2 \boldsymbol{d}}{\partial s^2}\right) + \nabla E_{ext} = 0$$

to be

$$\boldsymbol{K}\,\boldsymbol{d} \;=\; \boldsymbol{f}\,.$$

This approach does not yet respect the evolutionary character of snakes. Terzopoulos (1987) extended the original approach by dynamic terms such as kinetic energy and velocity dependant friction. To achieve a more realistic physical behavior of our snakes, we allow energy dissipation. The non-conservative Rayleigh dissipation functional (Neuenschwander 1995)

$$D(\boldsymbol{d}_t) \;=\; \int_l \frac{1}{2}\;|\boldsymbol{d}_t|^2\,ds$$

is added to the conserving snake energy

$$E(\boldsymbol{d}) \;=\; \int_l \frac{1}{2}\left(\alpha\left|\frac{\partial \boldsymbol{d}}{\partial s}\right|^2 + \beta\left|\frac{\partial^2 \boldsymbol{d}}{\partial s^2}\right|^2\right) + E_{ext}\;ds \tag{4.13}$$

used so far[1]. The Euler differential equation governing the motion of the snake are obtained according to Hamilton's principle by minimizing

$$\int_{t_0}^{t_1} E(\boldsymbol{d}) + D(\boldsymbol{d}_t)\, dt\,. \tag{4.14}$$

The Euler differential equations derived by applying the variational principle to Expression (4.14) are (see Neuenschwander 1995)

$$\frac{\partial \boldsymbol{d}}{\partial t} - \frac{\partial}{\partial s}\left(\alpha \frac{\partial \boldsymbol{d}}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta \frac{\partial^2 \boldsymbol{d}}{\partial s^2}\right) + \nabla E_{ext} = 0\,. \tag{4.15}$$

The validity of this approach is seen when we assume that the displacements $\boldsymbol{d}$ stabilize. Then, the left term in (4.15) tends to zero, and the dynamic system reduces to the static one.

For the implementation of (4.15), we use finite differences in time (see Cohen and Cohen 1993). We are only interested in the final result and thus do not need an accurate solution in time[2]. Therefore

$$\frac{\boldsymbol{d}^{(t)} - \boldsymbol{d}^{(t-1)}}{\gamma} + \boldsymbol{K}\boldsymbol{d}^{(t)} = \boldsymbol{f}^{(t-1)} \tag{4.16}$$

where γ denotes the time step. Equation (4.16) is used in the form

$$(\mathbf{1} + \gamma \boldsymbol{K})\boldsymbol{d}^{(t)} = \boldsymbol{d}^{(t-1)} + \gamma \boldsymbol{f}^{(t-1)} \tag{4.17}$$

where $\mathbf{1}$ denotes the identity matrix. At time step t, the forces and displacements of time $t-1$ are used to derive the new displacements. The matrix $\mathbf{1}+\gamma\boldsymbol{K}$ does not depend on t. It is a banded, symmetric and positive definite matrix. In contrast to $\boldsymbol{K}$, the term $\mathbf{1} + \gamma\boldsymbol{K}$ is regular, and hence its inverse may be precomputed using a Cholesky factorization (see for example Oevel 1996).

4.3 Propagation

A snake is an energy minimizing spline. We have noted that the energy consists of an interior energy that controls the shape of the spline, and an external energy that rewards or penalizes a given allocation. This chapter illustrates the behavior of snakes *in the absence* of external energy influences. Factoring out deformation dependent external energies avoids an evolutionary system and thus facilitates the method considerably. In the absence of external forces, the initial state with vanishing inner energy is only left if the snake's head or tail[3] is *fixed to a given displacement*. Such a given displacement is imposed by means of essential boundary

[1] To make the snake move more dynamically, one could add a kinetic energy term to (4.13). We did not investigate if such an extension of the model would result in an improved behavior of the snake for our cartographic purpose. Since displacements in generalization are generally small, no big movements are required. We refer to the accurate work of Neuenschwander (1995) for more information on dynamic terms.

[2] In the following, we make use of *superscripts in brackets* to denote the time parameter in equations and formula.

[3] A snake is symmetric, therefore it has no head and tail by nature. We use these terms only to differentiate the two ends of the line.

conditions and urges the line to perform the required deformation at a given point. The rest of the line adopts a state that minimizes the inner energy. Hence, by the restriction to inner energy, we simulate the cushioning of displacement within a line; a process denoted by *propagation*, see Section 2.3.

The value of studying snakes in the absence of external forces is twofold. On the one hand, propagation itself proves to be a useful operation in the generalization process. On the other hand, the restriction to inner energy helps us study the adjustment of lines under forces, which is an important prerequisite for the modeling of our displacement algorithm. First, we clarify the methodology of snakes for propagation and discuss their benefits in cartography. We then enhance the basic approach such that additional cartographic requirements are met.

4.3.1 Basic Behavior: A Detailed Example

The application of the snake model to propagation is illustrated in Figure 4.2. We assume that a displacement $\boldsymbol{d} = (d_x, d_y)^T$ is required on one line end, while the other one is fixed to its current position. The first point to note is that the method does not act on the line geometry itself, but on the parameterized forms $x(s)$ and $y(s)$, where s denotes the length along the line (i.e. the curvilinear abscissa). A displacement function is computed for each direction; $d_x(s)$ resp. $d_y(s)$ give the displacements for the parametric forms $x(s)$ resp. $y(s)$. It is necessary that these displacement functions guarantee the displacement we impose on the line ends, meaning that they satisfy $d_x(0) = d_x$ and $d_x(l) = 0$, resp. $d_y(0) = d_y$ and $d_y(l) = 0$. Under these constraints, the displacement functions are computed such that they minimize the inner energy given by

$$E(d_x(s)) = \int_0^l \frac{1}{2}\left(\alpha \left|d'_x\right|^2 + \beta \left|d''_x\right|^2\right) ds$$

$$E(d_y(s)) = \int_0^l \frac{1}{2}\left(\alpha \left|d'_y\right|^2 + \beta \left|d''_y\right|^2\right) ds$$

In the following, we assume α and β to be constants.

Now, let's revisit the same example, but with a shift of viewpoint on the numerical (practical) integration. We showed in Section 4.2.1 how the minimization problem leads to a pair of independent linear equations

$$\boldsymbol{K}\,\boldsymbol{d_x} = \boldsymbol{0} \qquad \text{and} \qquad \boldsymbol{K}\,\boldsymbol{d_y} = \boldsymbol{0}\,. \tag{4.18}$$

The matrix $\boldsymbol{K}$ is singular as long as no boundary conditions are imposed.

Natural boundary conditions, which correspond to forces applied to the snake, do not show up here by premise. Therefore the right side of (4.18) is set to zero.

Essential boundary conditions impose user defined displacement vectors at the snake's head and tail. To be well posed, the problem requires 4 boundary conditions, e.g. the specification of the displacement and the displacement derivative at both line ends. While the displacement values are given by the problem, the derivatives are best set to 0. This ensures smooth transitions (C^1-continuity) at points of contacts between the snake and the line it is connected to.

As the Hermite polynomials directly represent the displacement and the displacement derivatives along the line, these boundary conditions are easily translated into the coordinate vector $\boldsymbol{d}$. To respect these altered conditions, $\boldsymbol{K}$ needs to

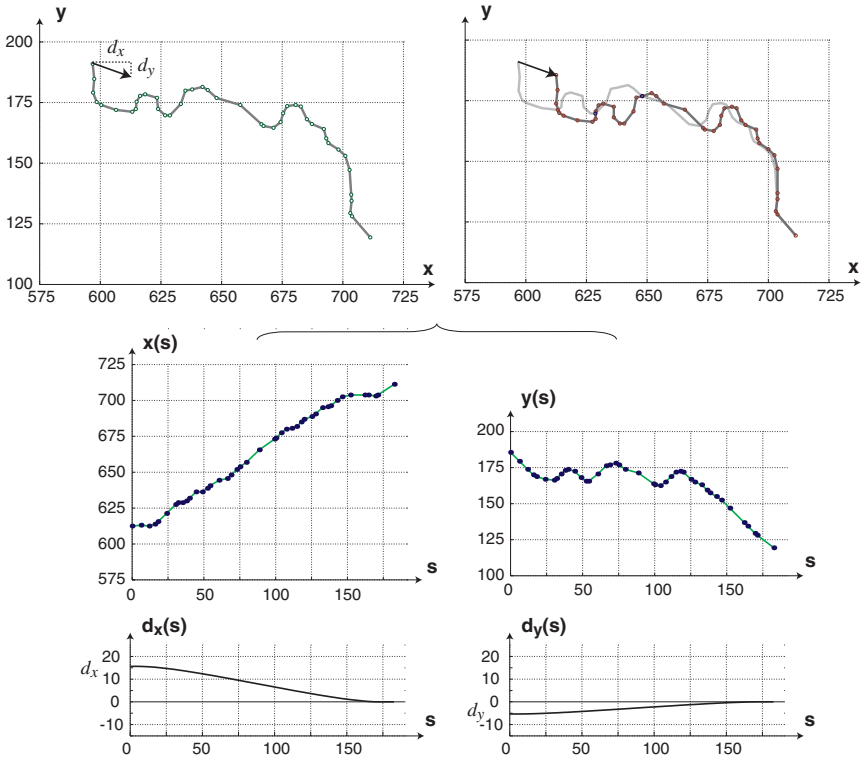


Figure 4.2: The initial line (top left) and its derived parametric forms $x(s)$ and $y(s)$ (middle). The displacement functions (bottom) pass through the imposed displacements at the line ends. Adding the displacements to the parametric forms and combining them gives the geometry of the propagated line (top right).

be adjusted. These modifications make $\boldsymbol{K}$ regular; the equation system (4.18) can be solved for the unknown values in $\boldsymbol{d}$, the coordinates of the basis function that are used to synthesize the final displacement function.

We draw attention to two important points. First, note that the displacements are computed for the x and y-parametric forms *separately*. This considerably facilitates computation, but also brings risks, as 2-dimensional modeling can not necessarily be split up into two independent directions (this point will be discussed again in Section 5.7.1). Second, one has to pay attention to the order of the computations. In a first step, both displacement curves $d_x(s)$ resp. $d_y(s)$ are computed such that their first and second derivatives are minimized with respect to the imposed boundary values. It is only after this first step that the coordinates of the lines are used to define the new shape of the line. But the original coordinates do *not* go into computations when the displacements are built.

4.3.2 Discussion

Computing Time Matrix $\boldsymbol{K}$ in (4.18) is banded with bandwidth $b = 7$ and of size $2n \times 2n$, where n is the number of vertices in the line. Its inverse is found in $O(np^2)$-time. This is the only computation to perform; by factoring out external influences, no iteration is required.

Cartographic Evaluation Figure 4.3 shows examples of propagation using the model described above. The presented method works well, in principle. The snakes approach preserves local and global shapes well. It is very robust. Even for major displacements the results are acceptable. This proves that the underlying energy definition is reasonable and fits cartographic requirements well. Of course, one could think of other energy definitions. Williams and Shah (1992), for instance, studied different techniques of curvature estimation, which would probably marry very well the cartographic requirements; However, the use of the interior energy presented by Kass *et al.* has the advantage over other energy definitions that it is easily solvable by means of variational calculus.

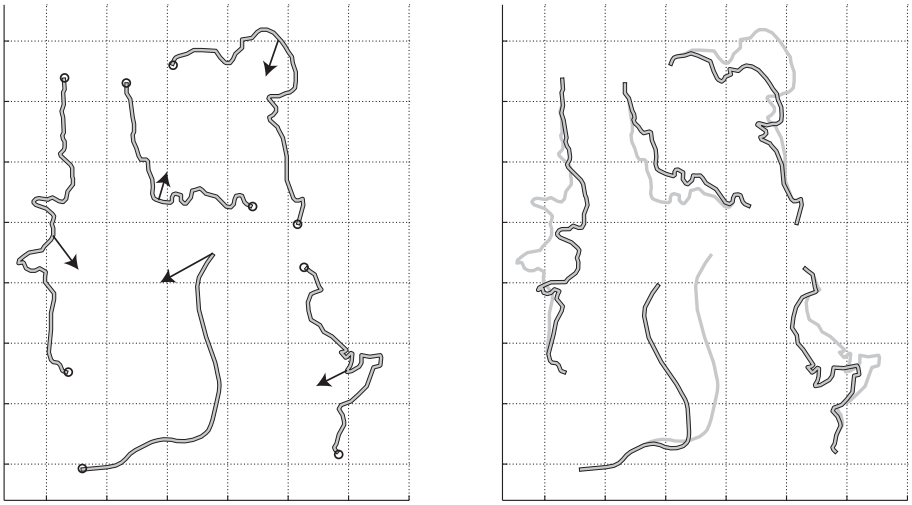


Figure 4.3: Propagation using the standard model as proposed by Burghardt (2000).

Even though the results in the above examples are fine, the basic approach presented so far violates important cartographic requirements:

1. The current result of propagation depends on the position of the snake's tail. While the head is defined by the position of the applied displacement vector, the tail can be placed at any arbitrary position along the line. Wherever the tail is positioned, propagation will finish there. The entire length of the line

is used to cushion the displacement; only then can the derivatives be held as small as possible.

2. The characteristics and the type of a line are ignored, though they are highly significant. Exaggeration is more easily applied on complex (sinuous) lines. Applying a displacement to a straight highway, for instance, is much more of a problem — propagation must be performed carefully, using a greater length to cushion the displacement. All this information does not yet enter into computation.

3. The basic snake technique allows self-intersection, not guaranteeing the absence of symbol overlap.

4. If a displacement vector is applied to a line close to an intersection, cushioning is not possible in the affected line only. A propagation algorithm must have the ability to spread a displacement over junctions as well.

In the following sections, we present solutions to overcome these drawbacks. (1) is guaranteed by changing the underlying energy definition. A shape related setting of the parameters α and β will solve problems (2) and (3). Finally we compose a system to include more than one snake, which results in a solution for (4).

4.3.3 Attraction Term

As pointed out above, cushioning displacement is done using the entire length of a snake: the result depends on the choice of the snake's tail. Finding a good position automatically for this tail is cumbersome. We prefer a method which could take any arbitrary point far enough from the snake's tail and which would cushion the displacement within as little of its length as possible - always subject to the constraint of shape preservation. We would then get rid of a search for admissible snake endpoints and obtain predictable results.

To achieve this goal, we extend the standard snake model with an attraction term

$$E(d) = \int_l \frac{1}{2} \left(\psi \left| \boldsymbol{d}(s) \right|^2 + \alpha \left| \boldsymbol{d}'(s) \right|^2 + \beta \left| \boldsymbol{d}''(s) \right|^2 \right) ds \,.$$

The ψ-term is added to the inner energy and penalizes the line as long as it does not coincide with the original position. As a consequence, the line strives to fall back on its initial position. The higher the value for ψ, the higher positional accuracy is assessed in comparison to the shape parameters α and β.

Incorporating this additional term in the equation system is straightforward. Applying the Euler equation, we obtain

$$\psi \boldsymbol{d}(s) - \alpha \boldsymbol{d}''(s) + \beta \boldsymbol{d}''''(s) = 0 \,.$$

Using this equation as source of the finite element method leads to a slight modification of the stiffness matrix $\boldsymbol{K}$: ψ has to be added to the diagonal of $\boldsymbol{K}$. As a nice side-effect of this modification, $\boldsymbol{K}$ becomes regular before any boundary conditions are introduced.

In Figure 4.4 the results of applying the improved propagation are shown for different values for ψ. We can now choose any point on the line as the tail of

the snake and get the same result. The line falls back on its initial position as soon as possible. Only a subset of the entire snake length is used to cushion the displacement. We can always use the line's endpoint as a snake limiting extent and do not have to worry about the extent that is needed for cushioning. Of course, this is only true if the line is long enough to incorporate a specified displacement. If the line is too short, resp. a displacement offset is required close to the line's endpoint, the method mismatches the displacement. The displacement changes the character of the line. We return to this point in Section 4.3.5.

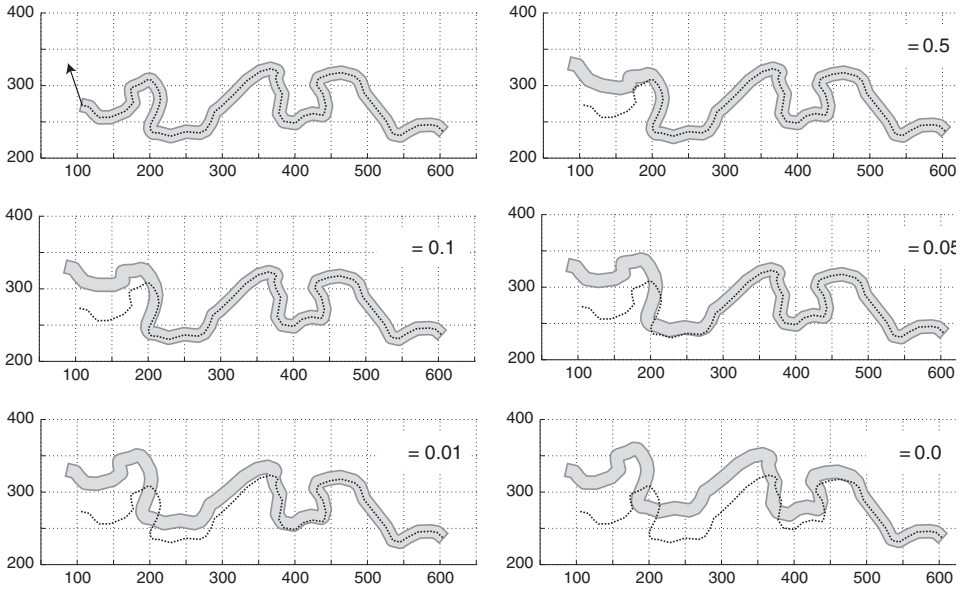


Figure 4.4: Displacement applied to the left of the line with fixed right end node and different values for ψ. As a reference, the thin black line displays the original position. Increasing ψ makes the displaced line return to its original position faster.

4.3.4 Varying Shape Parameters: Adaptive α and β

The shape parameters α and β define the flexibility of the displacement functions and hence in which way an imposed displacement is cushioned in a line. So far, the shape parameters have been treated as constants. The cushioning of displacement obeyed the same rules regardless of the underlying line. No cartographic peculiarity along the line, nor even the simple geometry, was taken into consideration.

We study in this section how the cushioning of displacement is influenced by varying the shape parameters. Our hope is that we can map peculiarities of the cartographic feature onto the shape parameters and thus incorporate cartographic reasoning in our model. For such an improvement we would be willing to pay the

cost of a pre-processing step, where a line is analyzed, and the shape parameters are adjusted based on this analysis.

There are different ways in which the variation of the shape parameters can be exploited. Firstly, we can increase/decrease both α and β together along the line. Secondly, the ratio between the parameters can be changed. Finally, what will become important when dealing with networks of lines, we can vary the values between different lines (e.g. depending on the road class).

1. **Varying both parameters together** Assume a line $\mathcal{L}$ consists of two parts $\mathcal{L}_1$ and $\mathcal{L}_2$ with shape parameters α_1, β_1 resp. α_2, β_2, where $\alpha_1\beta_1 \gg \alpha_2\beta_2$. A displacement imposed on the line is no longer cushioned equally along the line. In the first part, varying the displacement function is accompanied by a high increase of the snake's inner energy. In order to minimize the inner energy, the deformation is mainly shifted into the second part. The displacements $\boldsymbol{d} = (d_x, d_y)^T$ on the first part are fairly constant, as the energy is kept low with $\boldsymbol{d}' \approx \boldsymbol{d}'' \approx 0$.

   This behavior is used in our system to protect feature peculiarities from being deformed. Applying high values on such important parts of features results in constant displacement functions – a translation of the feature. Figure 4.5 illustrates how a sharp bend is protected from coalescing using this approach.

   Note that a constant displacement in both directions prevents rotation. It is not therefore suggested that straight lines be protected by this treatment, since straight line have to be prevented from being bent only, but not from rotation; rotation is often helpful in this case to ease cushioning and should not be suppressed.

2. **Varying the ratio of shape parameters** A second way to influence the behavior of the cushioning lies in varying the ratio between α and β, $\frac{\alpha_1}{\beta_1} > \frac{\alpha_2}{\beta_2}$, where $\alpha_1\beta_1 = \alpha_2\beta_2$. A higher relative value of β results in a smoother course of the curve. The slope of the curve far from the snake's head and tail increases also. However, these slight differences are barely noticed by a map reader. He/She sees the displacements only after being added to the original shape, where they are still barely perceivable. The results are not very sensitive to any relative exaggeration of α and β.

   The slight difference in flexion behavior is of importance only when we try to protect straight zones from being bent. Straight lines are mapped onto straight lines, only if a displacement function with vanishing second derivative ($\boldsymbol{d}' \equiv const$) is added to the initial coordinates. Therefore, we raise β and lower α on straight zones. The displacement function on these zones are then straight lines with raised slopes. This shows the anticipated benefits: The straight zones are not bent but pass over into straight lines again. The higher slope corresponds to a higher acceptance of elongation or shrinkage. As this happens on both displacement functions separately, this attendance to change length is sometimes expressed as a slight rotation of the straight line – a perfectly intended behavior.

   However, in practice, the results are less promising, as the described modulation is weak. Varying the ratio of shape parameters does not show a great effect, even though one might expect this from the physical interpretation of

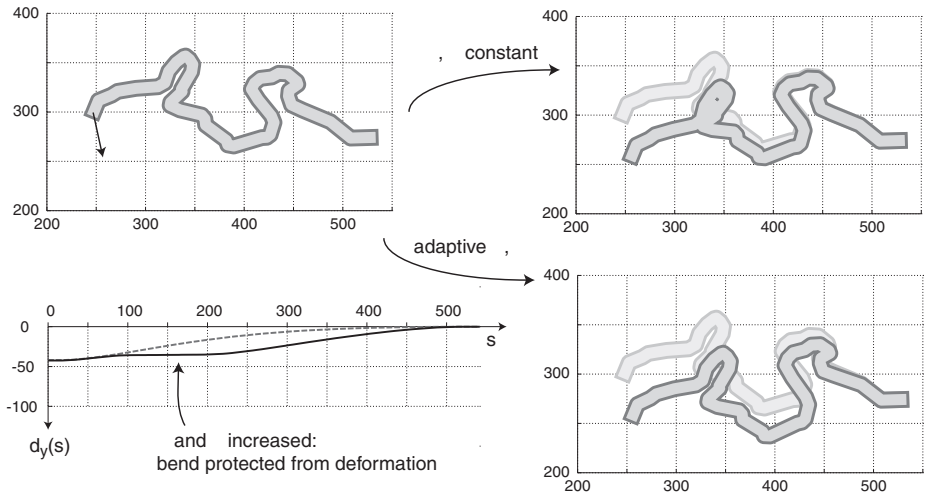


Figure 4.5: The cushioning of displacement can lead to an unacceptable deformation of feature peculiarities (top right). Increasing the shape parameters α and β over the narrow bend domain forces the displacement functions to be constant there (bottom left) and induces a translation of the bend (bottom right)

the model. Neither is a segment really willing to enlarge notably, nor can we guarantee that straight lines pass over into straight lines again, if the displacement is applied directly to the middle of the straight zone. This lack of being able to guide the elongation and bending of a snake explicitly by α and β is serious. It is also a limiting factor in later examples. Given more time to continue our work, we would focus again on this insufficiency, probably trying to overcome it with other interpolation functions in the finite element method.

In order to avoid running into problems with straight zones, we can still increase both α and β on these domains. This prevents bending, but also reduces the possibility of absorbing displacement in the segment's direction. We return to this point.

3. **Varying the parameters between lines** So far, our reasoning has focused on individual lines only. As the approach will be extended below to a *set of lines*, we embrace inter-line variations of shape parameters. Different lines expose different cartographic constraints. Exaggeration is more easily applied to complex (irregular, sinuous) lines. Applying a displacement to a straight highway, for instance, is energy-draining – propagation must be performed carefully, using a greater length to cushion the displacement.

   The technique to meet this demand is analogous to 1. Based on road attributes (e.g. highways are stiffer that agricultural roads) and sinuosity measures, we assign different values for α and β, not only to segments, but to

the whole line. High α, β-values make the snake more resistant against deformation. Note that an increase of the shape parameters has the same effect as if we decreased the attraction term ψ and held α and β constant. Still, we recommend adjusting the shape parameters rather than ψ. Firstly, from an ideological point of view, we characterize the lines, so we should change the shape parameters rather than the ψ term used to quantify positional accuracy. Secondly, when we extend propagation to an entire network, the propagation will be shared based on this deformation resistance. Finally, the behavior of the line against deformations becomes crucial when we reintegrate external forces.

These observations lead to the conclusions that the shape parameters can be used efficiently to protect cartographically remarkable shapes and to characterize the flexibility of lines. Conversely, there is not much potential in varying the ratio between the shape parameters α and β. As we compute only displacements, but not a new line as a whole, the influence of the shape parameters is less significant than might be hoped from physical interpretation and/or experience in computer vision.

Shrinkage and Enlargement The elongation (or shrinkage) of a line is shared by all segments. Generally, this is a good property of snakes. However, in the example illustrated in Figure 4.6, we would be better to let the straight line absorb more of the displacement. This is to prefer since a change in length of long parts is hardly noticed by the map reader. It keeps the correction local and thus reduces the danger of triggering new conflicts. In order to translate this cartographic reasoning into the method, we reduce the shape parameters drastically along the straighter part. This triggers (simulates) shrinkage/elongation. However, such an approach is only applicable if the straighter part has the same direction as the displacement vector. Otherwise, the straight line would be bent drastically. This approach relies on *a priori knowledge* of the displacement direction. This knowledge is available if a user applies a displacement direction by hand. But in the case of snakes, when proximity conflicts trigger the displacement, we can no longer make use of this technique, as the displacement directions are not known beforehand.

Missing Control of Line Orientation We conclude this section by an important observation regarding the missing 'cartographic awareness' of snakes. Consider Figure 4.6. Assume we had a second displacement vector $\boldsymbol{d}_\perp$, acting on the same vertex of the line as $\boldsymbol{d}$ does, but standing perpendicularly on $\boldsymbol{d}$. Which of the two displacements, $\boldsymbol{d}$ or $\boldsymbol{d}_\perp$, would be more easily cushioned in the line? From a cartographic point of view, the displacement vector $\boldsymbol{d}$ pointing in the direction of the general trend of the line is more easily absorbed than $\boldsymbol{d}_\perp$. A snake however, which is generally not aware of the line geometry, has no preferred displacement direction. The elongation of lines is therefore as expensive (in terms of energy) as a displacement perpendicular to the line trend.

4.3.5 Handling Junctions

Applying a displacement to a line close to a junction poses problems. Moving junctions is an expensive action, as their displacement entails corrections to other

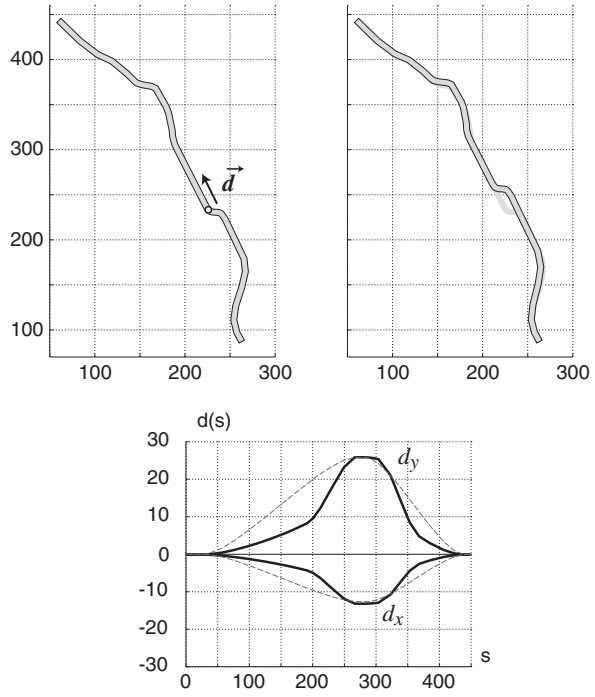


Figure 4.6: Only the decrease of the shape parameters along the straight line allows special adsorption of displacement magnitude, see bold displacement function (below). The dashed line shows the displacement function as it would be computed without adaption.

lines. Nevertheless, if the shape of a line does not allow us to cushion the given displacement, as the characteristics would be distorted in an inadmissible way, end nodes (junctions) can and should be displaced (see Figure 4.7). We present two approaches how the impediment of fixed junctions can be relaxed. The following discussion stems from Barrault *et al.* (2000).

Iterative Propagation We first focus on the line on which a displacement vector is imposed. We use the snakes approach with attraction term for positional accuracy, as described in Section 4.3.3. No boundary conditions on the end nodes are imposed. If the displacement is small enough, resp. if the line is flexible enough to cushion the entire displacement, the end nodes are not touched. The task is completed. However, if the line is too rigid, or if the displacement is applied towards the end of the line, a shift in one or even both end nodes is reported. Connectivity between the line and the rest of the network is lost. To reinforce connectivity, we run the snake algorithm for all other lines connected to this intersection with the displacement computed there for the initial line. In a snowball effect, the propagation is spread throughout the network until the displacement is cushioned

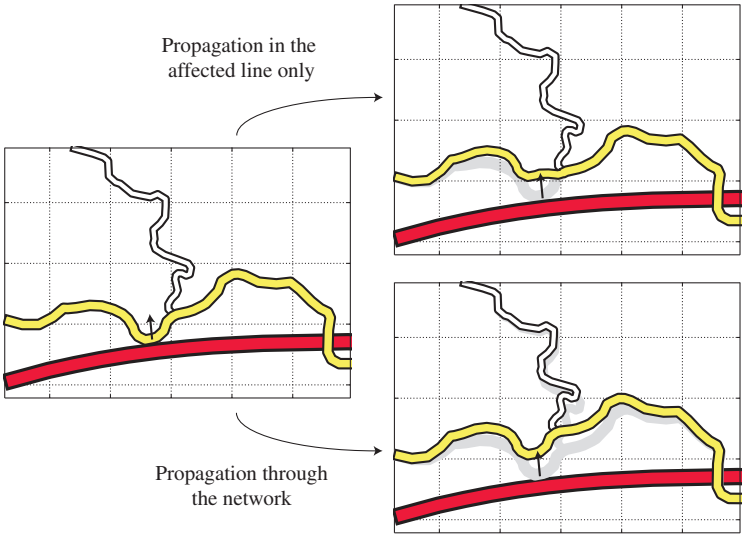


Figure 4.7: A displacement applied close to an intersection needs to be propagated through the neighboring roads.

completely.

Problems can become more difficult for networks with cycles. Different lines may require different displacements at an intersection. The method must provide a strategy to determine a displacement that is acceptable for all lines involved.

Global Approach: Extending Snakes for Road Networks The snake approach that we present here extends the snake algorithm to a set of snakes. Each line is identified as a single snake. For each snake the corresponding linear equation system is built as described in Section 4.2.1. In a next step, we assemble all the single snake matrices in a matrix that mirrors the behavior of the entire road network. When snakes join nodes – which happens at all intersections – the values of the matrix cells are summed. Each intersection node only appears once in the network matrix. All the lines ending in an intersection hold the same coordinate. Connectivity is inevitably ensured. On this network equation system we can then impose boundary conditions similar to those for single snakes.

Note that the network matrix is no longer banded. This property of the single snake matrices was helpful. It allowed us to keep the storage requirements low (only the entries in the diagonal band needed to be saved) and to accelerate the inversion of the matrix drastically. Without this banded structure, we apply a classical Cholesky factorization. The result is computed in $O(n^3)$ – an operation that is noticeable even if only few lines are involved. A partitioning of the road network is required beforehand. As stated in Section 2.6, we assume such a partition is given (for the sake of simplicity and computer power).

A drawback of the global approach is the property of *'heavy nodes'* at junctions with high node cardinality, see Figure 4.8. When many lines meet at an intersection, it becomes very expensive to move this junction. Passing on an amount of displacement from one line to such an intersection means that each line connected to the intersection subsequently contributes to the overall energy. It is worth cushioning the displacement drastically, to pass on just as little displacement as possible. Then, only one road reports energy, which compensates for the higher deformation. For junctions with only three incident roads, this 'heavy node' effect is probably desirable. Without deforming lines drastically, the junction has a slightly higher gravitation to its original position. However, if five or even more roads meet, the junction remains too static/stolid. This same unintended effect arises when several junctions appear in a row.

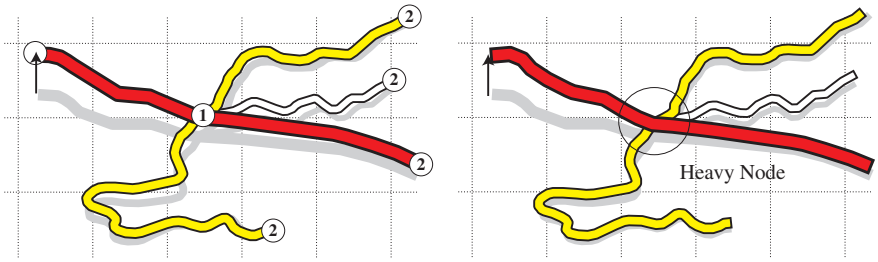


Figure 4.8: Propagation through a network of lines using the iterative (left) and the global approach(right).

Comparison The propagation of displacement through a road network is an important prerequisite for successful displacement. This issue has often been neglected. The iterative and global propagation algorithms presented in this section overcome this drawback.

The iterative approach is best suited to small and recurrent displacements. It is fast and provides good results. There is not much data management required (except for problems with cycles). The iterative approach is integrated and successfully used in the AGENT prototype (Lamy *et al.* 1999).

The global approach is methologically elegant and reliable for any dataset. It is also possible to apply more than one displacement vector at once. For the minimization of the network energy, the network is handled as an entity. External influences can act on this entity and deform its shape. The approach is therefore well suited for the reintegration of external energy. Conversely, the method is accompanied by a greater overhead in data management and is much slower than the iterative solution. The results in dense areas suffer from the problem of 'heavy nodes'. Propagation is applied too locally so to forego moving intersections. Nodes are too 'heavy' to be shifted. This can lead to cartographically sub-optimal results.

Conclusion Loosening snakes from the constraint of fixed junction nodes is important, not only for propagation. Two approaches have been presented to overcome this limitation. If dealing with propagation only, which occurs when the displacement vectors are known as a result of a previous process/method, we recommend the use of an iterative application of snakes to spread the displacement through the network. This guarantees a fast and flexible cushioning. If propagation is computed together with displacement, which is described next (Section 4.4), the global approach should be chosen. The benefit of this approach is that the internal energy of the entire network is computed. The main disadvantage is the 'heavy node' problem. Displacement is cushioned too strongly in those roads where the displacement is triggered. Also, the global approach has the drawback of slowing down the process since the banded structure of the stiffness matrix is lost.

4.4 Displacement

4.4.1 Basic Behavior: An Example

We discuss the behavior of snakes for the solution of proximity conflicts by means of the example illustrated in Figure 4.9. We assume that only the line squeezed between the main roads is subject to geometric deformation. This restriction is made for didactic simplicity and is removed later.

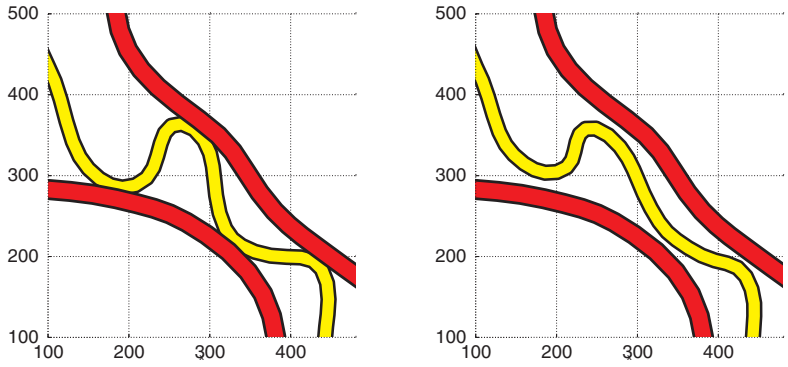


Figure 4.9: Initial situation with proximity conflicts (left) and situation after the application of the snakes algorithm (right).

We first define the displacement potential $E(x, y)$. The set of obstacles $\mathcal{O}$ is defined by the objects that might be within proximity of the snake. In our case, this set consists of two lines; we could add any other feature – points, lines or areas – to this set. For an arbitrary point $P = (x, y)$ we denote by $d(P, \mathcal{O})$ the Euclidian distance from said point to its closest obstacle. We define the displacement

potential by

$$E(P) = \begin{cases} 1 - \dfrac{d(P,\mathcal{O})}{r_o}, & \text{if } d(P,\mathcal{O}) < r_0 \\ 0, & \text{if } d(P,\mathcal{O}) \geq r_0 \end{cases}$$

where r_0 denotes the required minimal distance between the snake and an obstacle. The potential, together with the snake's initial position, is illustrated in Figure 4.10. This simple model of energy potential is sufficient for the current example; enhancements are presented in Section 4.4.3.

In the following, we also make use of forces to model external influences. The relation between forces and energy potential is defined in an unambiguous way by $\boldsymbol{F} = \nabla E$ (apart from the places where the displacement potential is without C^1 continuity). The gradients of E point towards the direction of steepest descent. A solution can be seen either as realizing the equilibrium of the forces or as reaching the minimum of energy.

As pointed out for propagation already, the deformations of the line are not computed on the real geometry. Displacement functions, separate in the x and y-direction, define the modification of shape. By analogy, the forces do not really act on the line, but on the two separated displacement functions (see Figure 4.11).

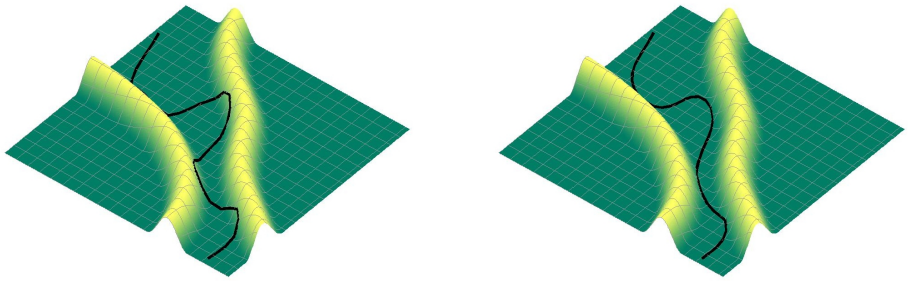


Figure 4.10: Displacement potential with superimposed snake at the start (left) and at the end (right) of the snake algorithm run.

We start with a description of the application flow. The head and tail of the snake is fixed to zero displacement. We use the direction and magnitude of the forces to find a better position for the snake. As snakes theory makes no assumption about the distribution of energy –the forces may vary drastically from one place to another – we make only a small step. The deformation of the displacement functions leads to a new position for the snakes. The shape of the line is now slightly distorted – the inner energy increases. On the other hand, the external energy has decreased because the line is located along locations with lower energy. The snake algorithm continues until the forces imposing the regularity of the shape and the forces pushing the snake towards locations with lower displacement potential are in balance. A further distortion of the shape would then cause a stronger increase

in energy than can be compensated by the decline of energy at the new position – and vice versa. The final state of the snake is displayed in Figure 4.10.

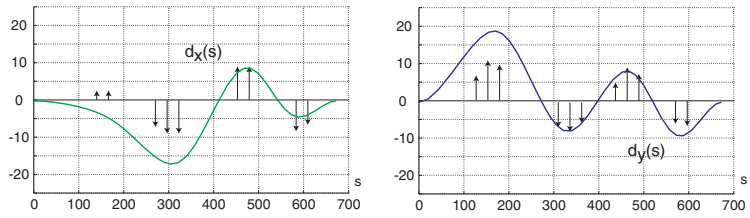


Figure 4.11: Displacement functions at the termination of the snake algorithm in x- (left) and y-direction (right). The arrows indicate the forces that acted on the snake at the beginning.

4.4.2 Initialization

Terminology We distinguish three types of nodes: (1) an *intersection node* is a node where three or more lines intersect; (2) a *partition node* denotes the outmost node on a line before it leaves the partition; (3) an *endnode* denotes the last node of a cul-de-sac.

Topology The topology in the dataset is given such that each connection between two nodes defines an individual road. If roads share important relations – e.g. a set of roads may link up to a long straight stroke that runs through the map as a highway – they have to be derived from the road attributes and the underlying geometry of the lines.

Shape Parameters For each line, an individual snake matrix is composed. We consider two criteria for the assignment of the shape parameters α and β. First, the shape parameters commensurate with the road attribute; e.g. highways hold higher α and β values than main roads and so forth. Second, we identify bends which are susceptible to symbol overlap under slight deformations already. Such bends are stiffened by increasing both α and β considerably.

Network Matrix To allow displacements of intersections also, a network matrix is built. Therefore, the individual snake matrices are assembled as described in Section 4.3.5. To accelerate the algorithm and to save memory, for each sub-network that is topologically independent of the others, an individual network matrix is allocated.

Boundary Conditions Depending on the type of node, different boundary conditions are applied. (1) Intersection nodes are free to move; they require no restraint. (2) Partition nodes are fixed and their derivative of displacement is also

set to zero; this guarantees smooth transitions between partitions. (3) Endnodes are unconstrained regarding their position. However, we have to preserve their direction (meaning no change in displacement derivative), as tails could rotate notably to compensate for forces applied to the snake's body.

4.4.3 External Energy

The external energy describes proximity conflicts between roads. In the field of cartographic displacement this energy is *line- and time-dependent*. Line-dependency comes into play because the displacement potential for a line is constituted by proximity considerations to all other lines. The displacement potential of one line is thus different from the potential defined for other lines. Time-dependency is an issue as all lines may move and thus proximity relations may change over time. Both issues are in contrast to the displacement potential used in applications for pattern recognition and also in contrast to the example illustrated in Section 4.4.1.

For reasons of simplicity, we abstain from describing the displacement potentials, and define the *forces* that act on the lines only. Such a force definition is sufficient as it is this form in which conflicts are taken into consideration to setup the snake system. We hereby restrict the computation to forces acting on vertices. This restriction has the benefit that it accelerates the computation of distances considerably. But we may miss out forces along long segments. This may require (in rare cases) a closer sampling of lines. In all our examples, however, such a resampling was not necessary.

What influences and determines the force vector acting on a vertex? Cartographic considerations motivate us to increase the force, the closer a line lies to a vertex. The direction of the force is defined by the direction of the shortest connection between the vertex and the conflicting line. If a vertex is in conflict with more than one line, forces may cancel each other out. It is therefore reasonable to sum the forces acting from different lines.

A simple approach that meets these criteria is described. Let us assume that the partition consists of n lines, denoted by $\mathcal{L}_1, \ldots, \mathcal{L}_n$ (see Figure 4.12). We focus a vertex P on an arbitrary line $\mathcal{L}_j$, $j \in (1, \ldots, n)$. The force $\boldsymbol{f}_P$ acting on this vertex P is defined by

$$\boldsymbol{f}_P^{(t)} = \sum_{\substack{i=1,\ldots,n \\ i \neq j}} \frac{\boldsymbol{v}_i}{|\boldsymbol{v}_i|} \left(r_{ij} - \min(|\boldsymbol{v}_i|, r_{ij})\right)$$

where r_{ji} : required min. distance between $\mathcal{L}_i$ and $\mathcal{L}_j$,
$\boldsymbol{v}_i$: vector from P to the closest point on $\mathcal{L}_i$.

Here, $\frac{\boldsymbol{v}_i}{|\boldsymbol{v}_i|}$ defines the direction of the force, while the latter term determines its magnitude. Forces only increase if the distance between the vertex and a neighboring line drops below the tolerance distance r_{ij}. This distance r_{ij} respects the symbol width of the roads $\mathcal{L}_i$ and $\mathcal{L}_j$ and incorporates a minimal separation distance ('hardcore distance') defined by the user. The upper index t in $\boldsymbol{f}_P^{(t)}$ has been added to express the time dependent characteristic of the forces.

This model demands modifications to respect the special requirements arise when dealing with *junctions*. Lines sharing a node inevitably fall short of the required distance. Yet, such an allocation does not have to expose/represent a

conflict. It is not easily determined in the immediate vicinity of a junction what a conflict is and what it is not. To avoid this problem, no force is applied to a vertex, if the distance along the graph of the road network from this vertex to the conflicting point on another line is less than $2r_{ij}$. This restriction leads to the following model

$$\boldsymbol{f}_P^{(t)} = \sum_{\substack{i=1,\dots,n \\ i\neq j}} \begin{cases} \frac{\boldsymbol{v}_i}{|\boldsymbol{v}_i|}\left(r_{ij} - \min(|\boldsymbol{v}_i|, r_{ij})\right), & \text{if } (P \rightsquigarrow P+\boldsymbol{v}_i > 2r_{ij}) \\ \boldsymbol{0}, & \text{otherwise} \end{cases}$$

where $P \rightsquigarrow P+\boldsymbol{v}_i$ denotes the distance *through the network* from P to $P+\boldsymbol{v}_i$ ($P+\boldsymbol{v}_i$ is obviously the point on $\mathcal{L}_i$ closest to P).

This model is rather rudimentary. Cartographic reasoning is poor. The force direction is solely determined by the shortest distance between a vertex and the conflicting lines. Later results will show that the force direction computed in this way does not always exhibit the best displacement direction. Whether it is worth establishing a more sophisticated theory on how to apply forces is not studied here.

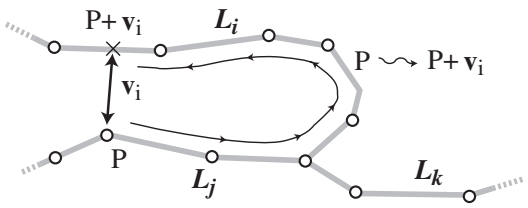


Figure 4.12: Terms used for the definition of external forces.

4.4.4 Energy Minimization Procedure

The optimization process is described in matrix terms by

$$(\mathbf{1} + \gamma \boldsymbol{K})\boldsymbol{d}^{(t)} = \boldsymbol{d}^{(t-1)} + \gamma\mu \boldsymbol{f}^{(t-1)}\,. \tag{4.19}$$

A time step consists of the computation of the forces and the subsequent application of (4.19) with these forces and the current line deformations. This results in new displacement functions from which the new positions of the lines are derived. The next time step is then computed based on this new geometry.

The process is guided by the parameters γ and μ. The weaker γ, the less displacement can be expected in one time-step. The displacement is weak and has only local impact. γ is obviously related to the positional accuracy term. μ is used to scale the forces such that they are in a reasonable ratio to the inner shape preserving strains (see below).

The evolution of energy is displayed in Figure 4.13.

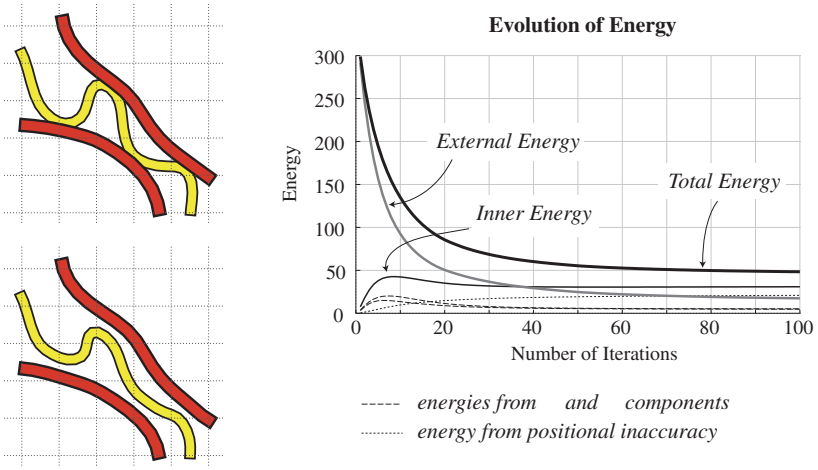


Figure 4.13: Partition before (top left) and after (bottom left) the application of the snake algorithm and the evolution of energy (right).

The example draws attention to two important issues. First, it shows how the energy of the system slowly progresses towards a (local) minimum. This slow convergence raises the question of when the iteration should be terminated. We did not investigate this problem in our prototype; the number of iterations is simply defined by the user.

A second problem associated with snakes in cartographic generalization is apparent from the visualization of the energy evolution shown in Figure 4.13. Snakes find their final position when an equilibrium of inner and outer energy is reached, a behavior that is only partially beneficial. The external energy should not be in balance with the inner one, but vanish instead, since the presence of external energy indicates that lines are still too close. Hence, the required minimal distance between lines is never met and the separation distance depends on the magnitude of the conflict.

The problem is diminished by making the external forces dominate the inner ones. This scaling of the forces is accomplished by increasing μ in (4.19). However, with such strengthened forces, we run the risk that lines will be strongly deformed and displaced. Therefore, we decrease γ; the effects in each time step are thus toned down.

Even with dominate external forces, we will not meet the required distance entirely, as the forces get weak when we are close to the desired distance. Simply specifying a higher minimal distance r is cartographically questionable. The distance r, which is then bigger than required, is met in partitions with low conflict potentials, but is still not guaranteed in places with high conflict potential. A better solution is to apply the snakes algorithm twice. The geometry of the lines after the first run is used as reference for the second run. This corresponds to resetting the inner energy back to zero, as shown in Figure 4.14.

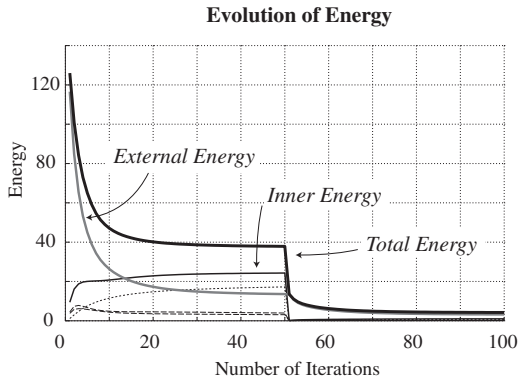


Figure 4.14: The algorithm was stopped after 50 iterations and the temporary result used to trigger the algorithm again.

4.4.5 Results

Figure 4.15 and 4.16 illustrate two results of our snake algorithm. The method shows the expected behavior. The conflicts are solved. The displacement is not just divided equally amongst the involved lines but respects their specific characteristics that make them more or less stiff against deformation and displacement. This is a remarkable feature of the optimization method: even though the modeling is focused only on individual lines (attributes and important bends), the results show the characteristics of a global 'reasoning'.

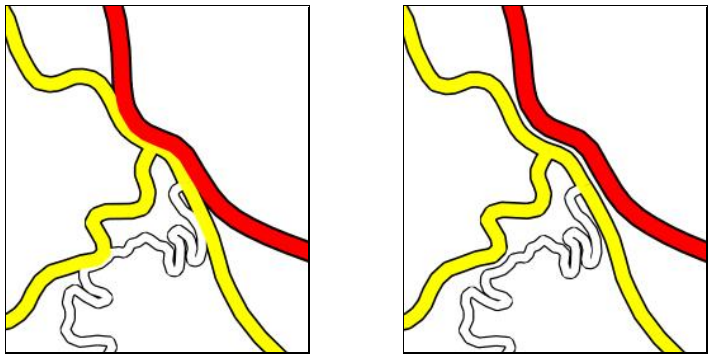


Figure 4.15: Partition before (left) and after (right) the application of our snake algorithm. Proximity conflicts are resolved and parallelism is preserved.

The example shown in Figure 4.16 was chosen as it points out the three major drawbacks of the method. First, we see that the required space between lines is not met everywhere. Where there is only a little space available for corrections or where several lines interfere, the lines lie closer than specified. Even though external forces are higher there, they do not suffice to trigger any further displacement. This problem was already addressed in Section 4.4.4.

A second drawback is the resistance of snakes against local enlargements. By opening the red bend shown in Figure 4.16, the proximity conflict with the stuck yellow line could be relieved. However, this would require the expansion of the segments to the left. Snakes resist such an enlargement in an undesired way (see the discussion in Section 4.3.4).

The example also points at the limitation of the chosen displacement forces. The top bends of the white line shown in Figure 4.16 only report a conflict to the north and hence are pushed southwards. The human eye detects that the neighborhood to the east would provide more space to avoid the conflicts. But, as the immediate conflict predicts the correction direction, this is not perceived by the snakes algorithm. To overcome this drawback, the scope, with which forces are determined, would have to be widened.

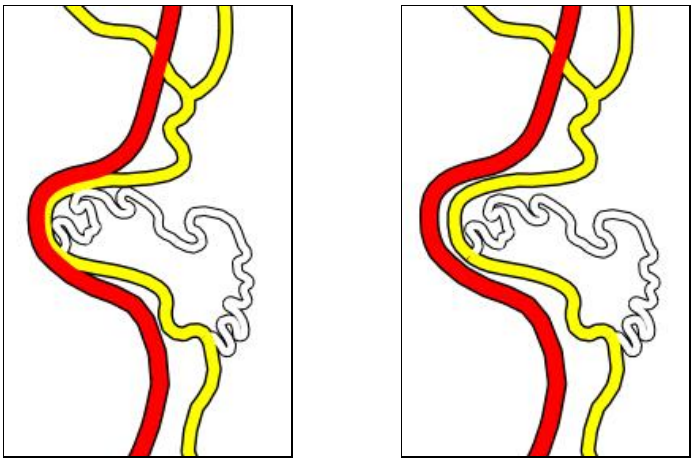


Figure 4.16: Problems associated with our snake algorithm.

4.5 Adaptation of the Algorithm to Intersections

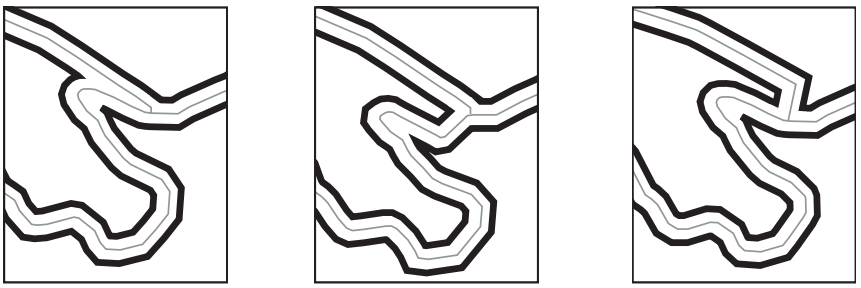
4.5.1 The Need for Improvement

The pleasing cartographic results illustrated in Section 4.4 are only possible by the restriction to a specific subset of displacement problems, namely the problems that require no corrections in the vicinity of junctions. Our definition of the displacement potential in Section 4.4.3 omits conflicts in the neighborhood of junctions.

Forces do not appear there, the geometry of junctions is kept untouched.

However, many displacement conflicts are found at junctions. The precise geometry of a junction is often concealed by the widened symbolization. Connectivity and orientation of exits are masked – important information for the map reader is lost (Figure 4.17). In this section, we show how the snake technique can be embedded in a more comprehensive method which also provides good results for a map partition exposing conflicts around junctions.

The first idea that comes to mind is: Why not also compute a displacement potential for the lines close to intersections? Can we bend the junctions into shape by external forces? We reject this approach, however. Bringing a junction into line with forces is futile. Symbolization problems around junctions require an *a priori* analysis; the understanding of the constitution of a junction is a prerequisite for successful generalization. A correction requires major deformations, limited to the immediate neighborhood. The displacements are often not equally shared between the lines that build up the junction. Also a rotation of incident segments is possible and often helpful. An algorithm must provide these corrections; simple proximity reasoning will fail.



Initial situation: Conflict between the stroke and a third line

Bad Correction: The correction breaks the stroke

Good Correction: The conflict is solved without breaking the stroke

Figure 4.17: Symbolization problems in junctions require an analysis of the strokes.

The analysis and correction of a junction is a demanding task. Thomson and Richardson (1999) and Thomson and Brooks (2000) pointed out the importance of perception. They proposed the 'Good Continuation principle', which sees the basic requirement of group relations between roads in the identification of line continuations. What is easily identified by the human eye needs careful analysis if using algorithms. Strokes[4] that pass through an intersection need to be detected as entity. Thomson and Richardson (1999) accounted that the good continuation usually succeeds in correctly pairing up the incident components of a road at a

[4]In this thesis, the term 'strokes ' is used as proposed by Thomson and Richardson (1999). A stroke is a chain of network arcs. The term itself is prompted by the idea of a curvilinear segment that can be drawn in one smooth movement and without a dramatic change in style.

junction. Having such semantic information at disposal, an algorithm still needs to find the proper geometric corrections to overcome the symbolization conflicts. As such corrections then rely only on local reasoning, we believe that such an algorithm can be implemented. In the following, we assume that such an algorithm for the enhancement of junctions is available.

We then face the problem of how the modifications of a junction can be transferred into the database. Each road is fixed in the network. The elongation or shrinkage of a segment is not possible, as we would lose connectivity at junctions. As an alternative, we could place the junction at its initial position and then tie the roads back to the junction. This approach is not optimal. As the final position of the junction is unknown, the adjustment on the incident lines to the altered situation is possibly incorrect. The new situation prejudices a bad reference geometry for the snake algorithm. A better choice of the junction position might have made the deformation unnecessary in the first place.

We could still hope that we could translate the modifications, proposed by the junction enhancement algorithm, into forces by which the required deformation can be provoked. This approach would have the advantage that connectivity between roads is never lost. Besides the pragmatic objection that there is not much sense in including forces if the shifts that lead to a correction are already known beforehand, this transformation also fails. The enhancements of junctions often requires a serious elongation of single segments. Using the snake energy definition, the resistance of a segment against such drastic deformation is high. To stretch a segment locally, huge forces have to be applied to both vertices of the segment. These forces dominate both the inner regularization forces and external proximity forces. The shape of a junction is broken, the position of the junction is no longer under control and third-part proximity influences are ignored.

4.5.2 Adaptation of the Algorithm

We propose a two-step strategy. In a first step, the junctions are analyzed and geometric improvements are computed. Presently, only a rudimentary algorithm implements this requirement in our prototype. A complete theory of intersection handling is beyond the scope of this work. The correction of a junction starts with the identification of the involved strokes. Strokes are identified based on the attributes of junction incident roads and on the continuity relations between lines, whereby continuity is detected by regression-analysis. With this analysis in hand, we compute corrections for the intersections. For the sake of simplicity, it is assumed that the 'main stroke' requires no correction. Only the incident components are adjusted. Our prototype supports the elongation and insertion of segments only. These operations allow the movement of important segments such that changes in line orientation come to lie outside the symbolization footprint.

The geometric corrections are applied directly to each line end. As we allow the elongation and shrinkage of segments, connectivity between lines is lost by this action (see Figure 4.18).

In the second step, connectivity is re-established and the best junction position is determined. This task requires only a slight modification of the snake algorithm. We briefly discuss here how the snake system must be set up to guarantee connectivity. Appendix B describes the procedure in detail.

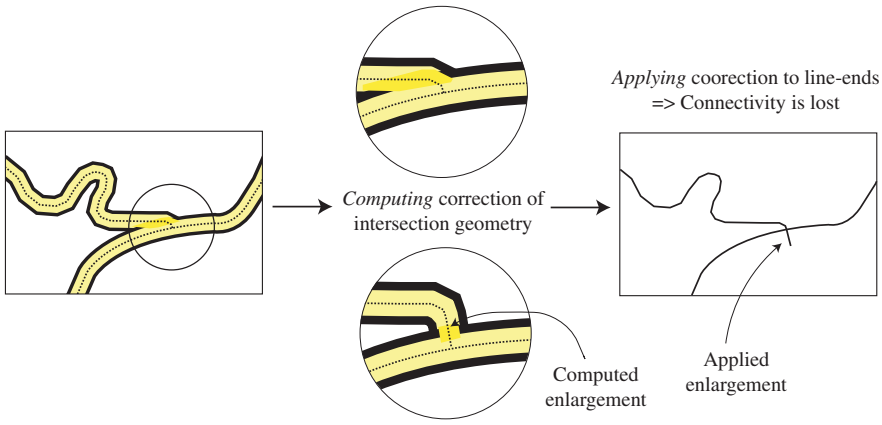


Figure 4.18: The correction of junctions is applied to the line-ends.

We choose the end-vertex of an arbitrary line as a reference point $\boldsymbol{P}_r = (x_r, y_r)^T$. The distance to other end-vertices is measured. This yields to proximity relations of type $\boldsymbol{P}_j = \boldsymbol{P}_r - \boldsymbol{v}_j$, where $\boldsymbol{v}_j = (\Delta x, \Delta y)^T$ denotes the vector from $\boldsymbol{P}_j$ to $\boldsymbol{P}_r$ (see Figure 4.19). These vectors define how far the end-vertices need to be shifted to restore connectivity to the reference point. However, as the initial position of the reference point $\boldsymbol{P}_r$ does not coincide with the final junction position (it is subject to displacement also) relative proximity relations need to be established. We require

$$\boldsymbol{d}_j = \boldsymbol{d}_r - \boldsymbol{v}_j \tag{4.20}$$

where $\boldsymbol{d}_j, \boldsymbol{d}_r$ indicate the displacements of the nodes $\boldsymbol{P}_j, \boldsymbol{P}_r$ yet to be determined. Obviously, if condition (4.20) is met, connectivity is guaranteed.

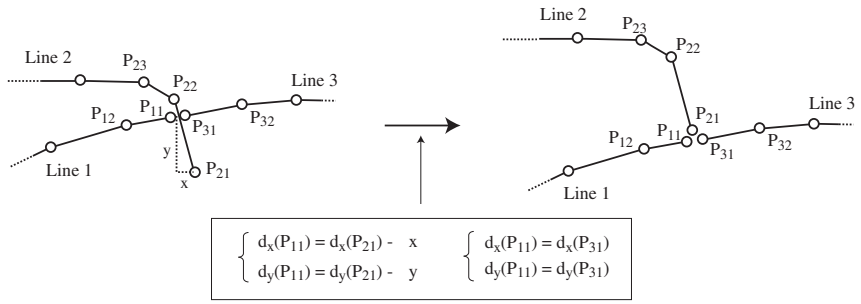


Figure 4.19: Imposing displacement relations on the equation system guarantees connectivity.

Condition (4.20) is directly integrated in the snake matrix and thus connectivity is guaranteed. In the matrix, the entries of those points which share a relation to the reference point are merged. This enforces the same displacements on these points. The vector $\boldsymbol{v}_j$, by which a node must additionally be displaced to ensure connectivity is translated beforehand into the force vector. Note that this force is not needed to stretch the elements (the elongation has been applied in the first stage of our algorithm). It just compensates for the energy that is integrated to provide connectivity. Besides this small modification of the snake equation setup, the process remains similar to that established in 4.4.

4.5.3 Results

Figures 4.20 and 4.21 show results of the algorithm. As the required displacements are substantial, the resistance of the lines to any further movement increases considerably. Therefore, we applied the algorithm twice. The second run used the output of the first run as input. This releases the high inner strains after the first modifications.

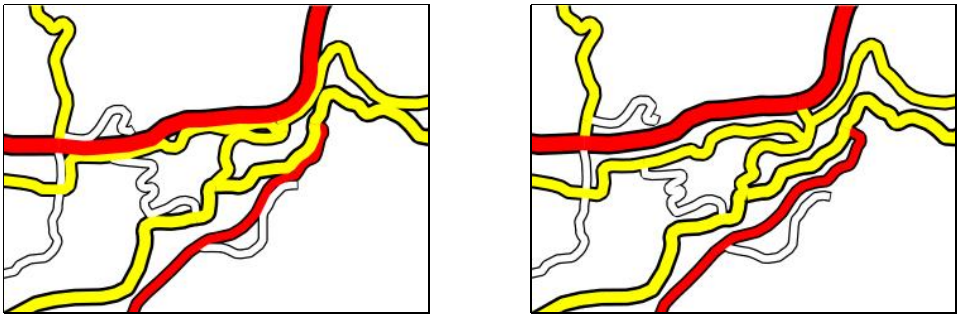


Figure 4.20: Road displacement with snakes, enhanced for road networks with intersections.

The partition displayed in Figure 4.20 is described by 380 vertices. This requires the allocation of a 760×760 matrix. For computer memory and CPU reasons (Cholesky factorization) we recommend to implementations to take advantage of the sparsity of the matrix. Not even 1% of the matrix elements differ from 0. However, as we did not yet make use of this property in our prototype implementation and as a result computation time is notable. Also, the distance measures to define the external energy consume sizeable resources, esp. as this computation is required in each iteration. Careful preprocessing is indispensable. The algorithm is written in Java. On a Sun Ultra 30, it required 1.4 minutes CPU-time to compute the solution of Figure 4.20. Obviously, the implementation still leaves room for many accelerations.

Our prototype does not handle under- and over-passes. This behavior could be easily integrated if the data structure and topology builder supported this property. Because this feature was missing, too many nodes where introduced in the gateway

of the red highway displayed in Figure 4.21 with the consequence that our junction enhancement algorithm failed. Thus we imposed two segment enlargements by hand. However, note that no remedies had to be applied on the snake process *per se*.

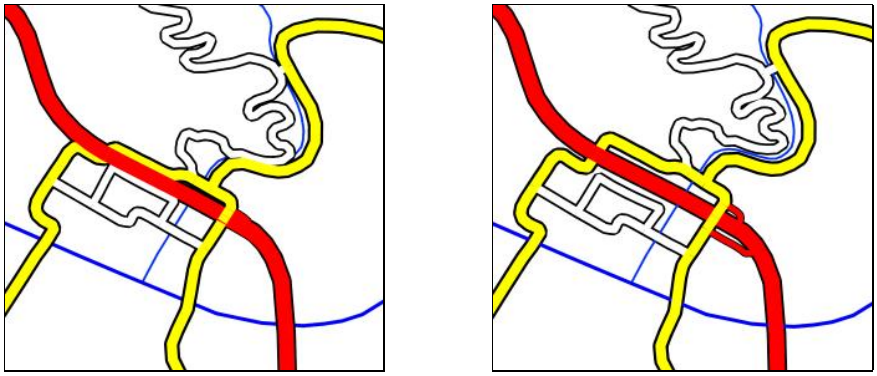


Figure 4.21: Road displacement with snakes. The river in blue did not enter into the computation. It was added later.

4.6 Conclusion

In this chapter, the use of snakes has been investigated for cartographic displacement. First studied was how the behavior of snakes can be influenced through the parameters α and β. The adaptation of these parameters proved to be useful for the modeling different road types and for the protection of important object peculiarities. However, in general, facilities introduce cartographic modeling beyond these shape parameters has shown to be limited. A major drawback is the lack of control over the line orientation: With snakes, two dimensional arguing is not really possible.

Next, it was shown how a set of snakes can be combined to model a road network. This improvement is crucial, since displacement is difficult to manage with the constraint of fixed junction nodes. In practice, many proximity conflicts occur in the vicinity of junctions. Often, a local, and strong correction of the junction geometry is also required to adapt the junctions to the wider symbology. To detect junction related conflicts, a local analysis of the road strokes is applied. Based on this analysis, a correction is computed. It was shown in this chapter how these modifications can be introduced into the set of snakes, such that connectivity is guaranteed and such that the overall energy is minimized concurrently.

The results of our method exceed the quality achievable using existing methods. The method is able to deal with several conflicting lines at once and computes displacements, where line distortions are barely perceived.

A weakness of the method is the inaccuracy with which the specified minimal distance between roads is met. If several lines are in conflict, and if little space for correction is available due to the rigidity of the involved roads, the required minimal distance is not entirely met. In such situations, the external energy does not compensate sufficiently for the increasing internal energy. A further deficiency of the current implementation is probably the way in which forces are computed. The best displacement direction is not always detected. Finally, the snake model does not allow sufficient cartographic modeling. With our implementation, the character of roads can not be modeled accurately through the shape parameters α and β.

Chapter 5

Elastic Beams for Road Displacement

5.1 Motivation

Background In the last chapter we saw the power of the snakes technique for the displacement of roads. Different components of this approach proved to be reliable and well suited for cartography. Cushioning displacement by means of a function that minimizes the second derivative of displacement satisfactorily preserved the character of a line and enabled a smooth transition between lines. The iterative application of forces to push roads away from conflict spots was effective, especially for the solution of conflicts between several lines, where it is hard to side step existing and emerging conflicts. Through the division of displacement amongst lines, we also had good experiences in assessing the possibilities of stiffening segments and/or roads.

However, the snake method can not meet all cartographic requirements. By parameterizing a road geometry with respect to the curvilinear abscissa s, and then computing displacements for both parametric forms $x(s)$ and $y(s)$ separately, one has to work with a distorted image, which allows no real two-dimensional spatial reasoning. Also, snakes compute displacements from scratch, meaning that the original line shape does not enter the computation. We can influence this computation by varying the shape parameters α and β, but the effect of this only affords limited control, especially as the shape parameters only influence displacement functions that are solved independently.

Underlying Idea In this chapter, we present a method that overcomes the aforementioned drawbacks. For this purpose, we retain the successful components of the snakes technique but change the underlying representation of lines in order to gain a real two-dimensional model of the problem.

At its core, this alternative method treats the road network as if it was a structure of *elastic beams*. By a beam, we mean a long and slender elastic bar

(we use the terms elastic member, elastic rod, or elastic string synonymously)[1]. If roads are in proximity conflict, we apply, in analogy to snakes, forces on the roads, resp. to the corresponding beam structure (Figure 5.1).

The behavior of beams under load, described in the mechanics of materials, consists of two characteristics: On the one hand, a force applied in the direction of a beam's longitudinal axis exerts a compression or stretching; on the other hand, under transversal force, the beam is bent (see Figure 5.1). This description of *stretching* and *bending* provides a good steering mechanism for cartographic lines. It nicely reflects the true two-dimensional character of the problem.

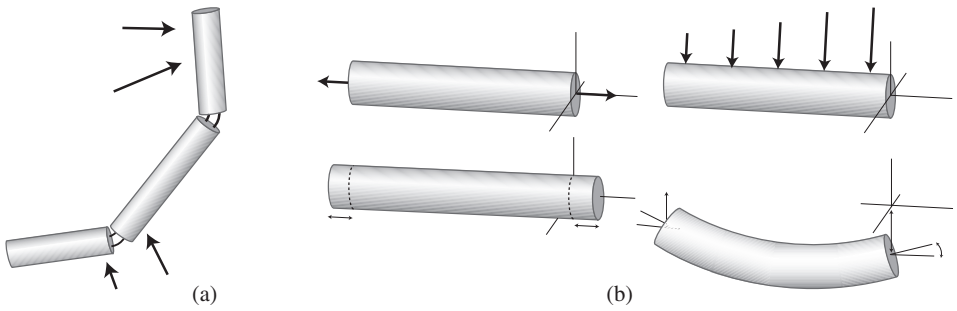


Figure 5.1: (a) A road, resp. a road network is understood as beam structure. Proximity conflicts between roads release forces on the structure. (b) An elastic beam under a force oriented longitudinally along its principal axis is stretched (or compressed), while under transversal force it is bent.

The search for a conflict-free state of the map is again handled iteratively. Therefore, we retain the ideas established and approved for snakes already. The numerical implementation of the approach is accomplished again by a finite element method. The new algorithm closely resembles the snakes algorithm.

This chapter is roughly organized into 3 parts. First, we give a short introduction to the basics of the theory of elasticity (Section 5.2). Hence, we are able to derive the stiffness matrices required for the finite element method (Section 5.3). In the second part, which includes Sections 5.4 to 5.6, we test the method and improve it for our cartographic purpose. Finally, in Sections 5.7.1 and 5.8, we discuss the benefits and drawbacks of the method in comparison to the snake technique.

[1] An engineer associates a 'beam' with a load applied perpendicularly to the beam's longitudinal axis only. Contrarily, we use the term of an elastic bar subject to transversal *and* longitudinal forces.

5.2 Principles of Elasticity for Beams

Here we summarize some of the concepts and basic equations of linear elasticity theory. The text focuses on the description of elastic phenomena occurring in elastic bars, and states the energy related equations required to later deduce stiffness matrices for a finite element method. The reader who desires more background information on a particular equation is encouraged to consult one of the many comprehensive texts on elasticity theory, for example, Hibbeler (1999), Atanackovic and Guran (1999), and Bedford and Liechti (2000).

5.2.1 Stress and Strain

Equilibrium For simplicity, we consider deformable bodies that are in equilibrium. Equilibrium of a body requires both a balance of forces $\boldsymbol{F}$, to prevent the body from translation or acceleration along a curved path, and a balance of moments $\boldsymbol{M}$, to prevent the body from rotating. These conditions are expressed by the two vector equations

$$\sum \boldsymbol{F} = \boldsymbol{0}\,, \qquad \sum \boldsymbol{M}_{any-point} = \boldsymbol{0}\,.$$

Consider now a sample B of some solid material, under the effect of the forces shown in Figure 5.2, in equilibrium. Let us pass an imaginary plane $\mathcal{P}$ through the sample dividing it into parts B' and B'', and isolate part B'. It is clear from Figure 5.2 that part B' can not be in equilibrium under the action of forces $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ alone. Part B'' must exert forces at the plane $\mathcal{P}$ that keep part B' in equilibrium.

What are those forces? They are literally the forces that hold the material together (Bedford and Liechti 2000). At the atomic level, these forces are ionic, metallic and van der Waals forces that act between the individual atoms. By exerting external forces on the sample, we alter the distances between atoms, changing the forces exerted by the atomar bonds.

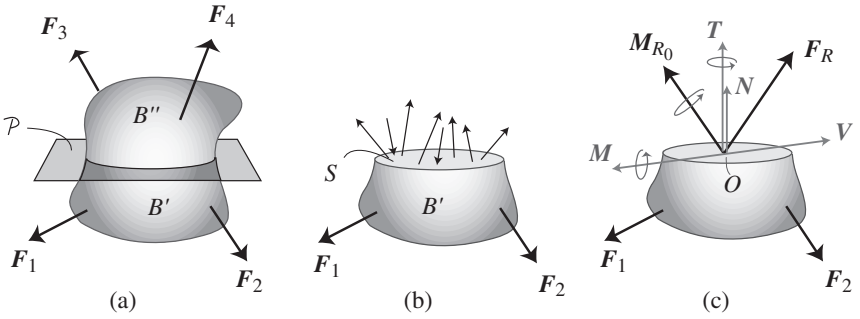


Figure 5.2: (a) A body is held in equilibrium by four external forces. The plane $\mathcal{P}$ divides it into parts B_1 and B_2 for subsequent analysis. (b) The free body diagram of the isolated part B'. (c) The resultant force and moment, $\boldsymbol{F}_R$ and $\boldsymbol{M}_{R_0}$ at a specific point O on the sectioned area.

Resultant Forces Although the exact distribution of internal loadings may be unknown, we can use the equations of equilibrium to relate the external forces on the body to the distribution's resultant force $\boldsymbol{F}_R$ and moment $\boldsymbol{M}_R$ at any specific point O on the sectioned area. Four different types of resultant loadings are distinguished (Figure 5.2). The *Normal force*, $\boldsymbol{N}$, acts perpendicular to the area. The *shear force*, $\boldsymbol{V}$, lies in the plane of the area and is developed when the external loads cause the two segments of the body to slide over one another. A *torsional moment*, $\boldsymbol{T}$, is developed when external loads twist the body. The *bending moment*, $\boldsymbol{M}$, is caused by the external loads that tend to bend the body about an axis lying within the plane of the area. On the elastic beam, for the application in mind, only coplanar loadings are applied and thus no torsion is developed.

Stress It is important to not only know the resultant forces, but also to know the actual *distribution* acting over the sectioned area. To describe this system of forces, we fix an arbitrary point P on the surface S of B'. Let $\boldsymbol{n}$ denote the unit normal on S at P. Let us consider an element of S with the area ΔA, containing P. The element of area ΔA is assumed to shrink in size towards zero but in such a way that it always contains P and that its normal is $\boldsymbol{n}$. Then, we define a quantity $\boldsymbol{p}_n$ by the relation

$$\boldsymbol{p}_n = \lim_{\Delta A \to 0} \frac{\Delta \boldsymbol{F}}{\Delta A}$$

with $\Delta \boldsymbol{F}$ representing the resultant of all forces acting on the surface element of area ΔA that are the result of the action of part B'' on part B' (Atanackovic and Guran 1999). The limit is assumed to exist and is called *stress*. We distinguish normal stress and shear stress (Figure 5.3). Normal stress σ_z is defined as the force per unit area acting normal to the surface of the section. It expresses the intensity of pushing ('compressive stress') or pulling ('tensile stress') on the section. Shear stresses τ_{zy}, τ_{zx} denote the intensity of force at a tangent to the area ΔA. If the body is further sectioned by orthogonal planes, we cut out a cubic volume element of material that represents the state of stress acting around a chosen point in the body.

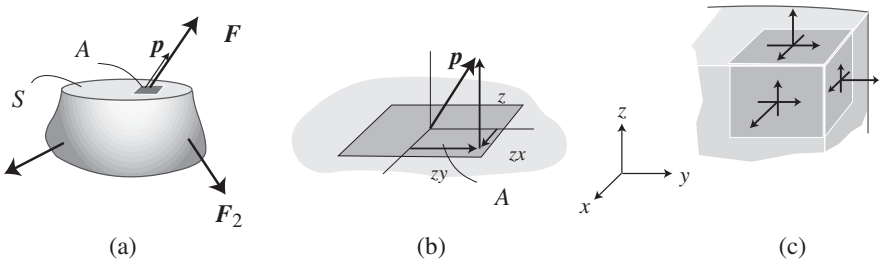


Figure 5.3: (a) A small area ΔA on the sectioned area S and its associated force $\Delta \boldsymbol{F}$. (b) The components of $\boldsymbol{p}$, normal and tangential to ΔA are the normal and shear stresses. (c) General state of stress.

Strain The application of a force provokes a deformation of the body. Strain is a measure of this deformation. The deformation is generally not uniform throughout the body. We can observe two different types of strains. The elongation or contraction of a line segment per unit length is referred to as normal strain ϵ. The change in angle that occurs between two line segments that were originally perpendicular to one another is referred to as shear strain γ. In an informal way (see Figure 5.4) ϵ and γ may be expressed as

$$\epsilon = \lim_{B \to A \text{ along } n} \frac{\Delta s' - \Delta s}{\Delta s} \qquad \text{and} \qquad \gamma_{nt} = \frac{\pi}{2} - \lim_{\substack{B \to A \text{ along } n \\ C \to A \text{ along } t}} \varphi' \,.$$

If the displacement field in the deformed body is represented by the two components u and v parallel to the coordinate axes x and y, respectively, the strain at a point in the body may be mathematically expressed as

$$\epsilon_x = \frac{\partial u}{\partial x} \,, \qquad \epsilon_y = \frac{\partial v}{\partial x} \,, \qquad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \,. \tag{5.1}$$

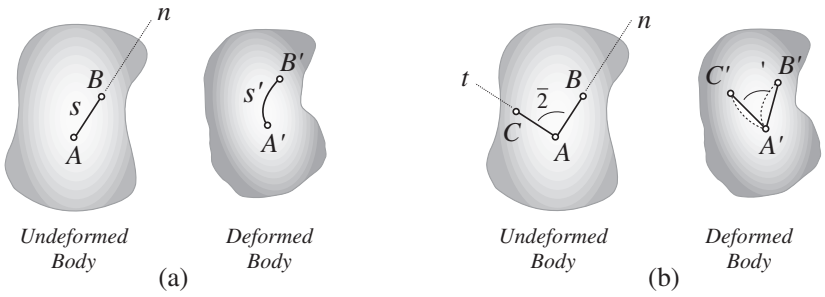


Figure 5.4: (a) Finite line within a sample before and after deformation (normal strain). (b) Perpendicular lines before deformation and in deformed state (shear strain).

5.2.2 Mechanical Properties of Materials: Hooke's Law

The behavior of a material under stress is expressed by the stress-strain diagram. This diagram shows the initiated strain as a reaction to a stress applied on the material. The stress-strain diagram for steel is displayed in Figure 5.5. It can be seen there that the curve is a straight line throughout most of the elastic region, so that stress is proportional to strain. The material is said to be linearly elastic. This fact is known as *Hooke's law* and may be expressed as

$$\sigma = E\epsilon \,. \tag{5.2}$$

Here E represents the constant of proportionality, which is called the *modulus of elasticity* or *Young's modulus*. Beyond the proportional limit (see Figure 5.5), the material still responds elastically, however the curve tends to bend and flatten out. Above the elastic limit a further increase in stress yields a permanent deformation. If the load is removed, a specimen would no longer return to its original shape.

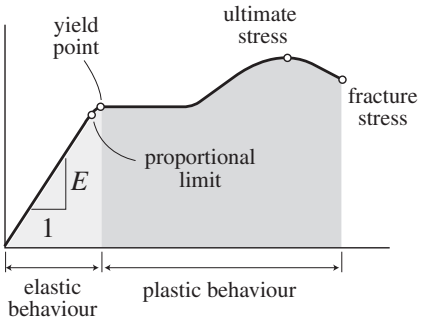


Figure 5.5: Stress-strain diagram for a tension test of ductile material (steel), not to scale, after Hibbeler (1999).

5.2.3 External Work and Strain Energy

As a material is deformed by an external loading, it stores energy internally throughout its volume. Since this energy is related to the strains in the material, it is referred to as *strain energy*. This energy, which is always positive, is caused by the action of either normal or shear stress. For both axial loading and bending moment, it is sufficient to handle normal stress only.

Work of a Force The work U performed by a force F to provoke a displacement x that is in the *same direction* as the force, is usually defined by

$$U = \int_0^x F\,dx\,.$$

In the field of elasticity, the force F is not constant. Its magnitude is gradually increased from zero to some limiting value P, which provokes the final displacement Δ. If the material behaves in a linearly elastic manner, then the force will be directly proportional to the displacement, that is, $F = \frac{P}{\Delta}x$. Thus we get

$$U = \int_0^\Delta \frac{P}{\Delta}\,x\,dx = \frac{1}{2}P\Delta\,. \tag{5.3}$$

Normal Stress If the volume element shown in Figure 5.6 is subject to a normal stress σ_z, then the force created on the top and bottom faces is computed by

$$\sigma_z = \frac{dF_z}{dA} \qquad \Longrightarrow \qquad dF_z \;=\; \sigma_z \; dA \;=\; \sigma_z \; dx\, dy\,.$$

As pointed out above, the force is applied gradually, increasing from zero to dF_z. The work done by dF_z to exert a displacement $d\Delta_z$ is therefore (see (5.3)),

$$dU = \frac{1}{2} dF_z \, d\Delta_z\,.$$

We know from 5.1), that this displacement $d\Delta_z$ is related to strain by $d\Delta_z = \epsilon_z dz$. Since the volume of the element is $dV = dx\, dy\, dz$, we have

$$dU = \frac{1}{2} \sigma_z \, \epsilon_z \, dV\,.$$

In general then, if the body is subject only to uniaxial normal stress σ, acting in a specified direction, the strain energy in the body is

$$U = \int_V \frac{\sigma\epsilon}{2}\, dV\,. \tag{5.4}$$

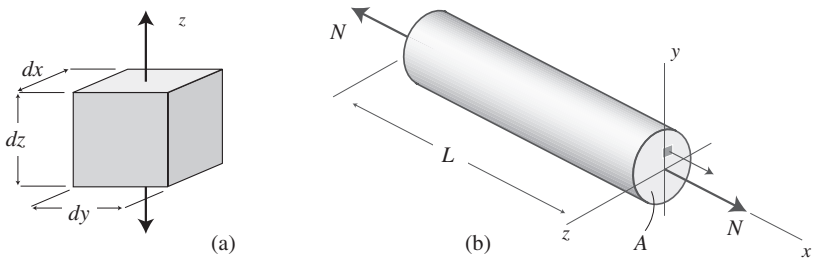


Figure 5.6: (a) A volume element subject to a normal stress σ_z. (b) A prismatic bar of constant cross-sectional area A, length L, under constant load N.

5.2.4 Elastic Strain Energy for Axial Load

Consider a bar of constant cross sectional area A and length L which is subject to a constant axial load F coincident with the bar's centroidal axis (Figure 5.6). Stress and strain inside the bar vary only in the x-direction. Inserting this observation in (5.4), we derive the energy stored in the bar to be

$$U \;=\; \int_V \frac{\sigma(x)\epsilon(x)}{2} dV \;=\; \int_A \int_L \frac{\sigma(x)\epsilon(x)}{2}\, dx\, dy\, dz \;=\; \int_L \frac{A}{2} \sigma(x)\epsilon(x)\, dx\,.$$

Hooke's law states that $\sigma(x) = E\epsilon(x)$, where $\epsilon(x) = \frac{du}{dx}$, with $u(x)$ the displacement along the x-axis. Therefore

$$U = \int_L \frac{A}{2}\sigma(x)\epsilon(x)\,dx = \int_L \frac{AE}{2}\epsilon(x)^2 dx = \int_L \frac{AE}{2}\left(\frac{du}{dx}\right)^2 dx\,. \tag{5.5}$$

5.2.5 Bending

We consider a model for a horizontal clamped beam made of an elastic material subject to a vertical load $w(x)$ (see Figure 5.7). The governing differential equation in terms of the vertical deflection of the beam $u(x)$ is given by

$$EI\frac{d^4u(x)}{dx^4} = F(x)\,.$$

The modulus of elasticity E and the moment of inertia I are assumed to be constant over the beam. The equation $d^4v/dx^4 = w$ results from combining an equilibrium equation of the form $M''(x) = F(x)$, where $M(x)$ represents the bending moment, and a constitutive equation of the form $M(x) = EIv''(x)$ (Eriksson *et al.* 1996). Each derivative of the deflection $v(x)$ has a particular name

$$\begin{aligned}
&\frac{dv(x)}{dx} = \Phi(x) && \text{(slope or rotation)}\,,\\
&EI\frac{dv^2(x)}{d^2x} = M(x) && \text{(bending moment)}\,,\\
&EI\frac{dv^3(x)}{d^3x} = \frac{dM(x)}{dx} = V(x) && \text{(transverse shear)}\,,\\
&EI\frac{dv^4(x)}{d^4x} = \frac{dV(x)}{dx} = w(x) && \text{(load)}\,.
\end{aligned}$$

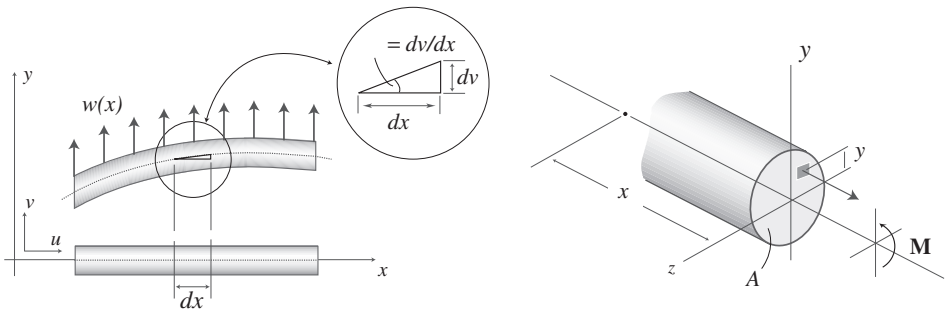


Figure 5.7: (a) A transversely loaded beam with deflection $v(x)$ and slope $\theta(x)$. (b) Axis-symmetric beam.

These equations are only of secondary importance for us, as we require a variational formulation for the finite element method. Thus we also derive the energy stored in a beam due to bending. Since a bending moment applied to a straight member develops normal stress in the member, we can again use (5.4) to determine the strain energy. We assume the beam has constant cross sectional area A and length L. The flexure formula (Hibbeler 1999) relates the normal stress acting on an arbitrary element at a distance y from the neutral axis to the bending moment by

$$\sigma(x) = \frac{M(x)y}{I}\,. \tag{5.6}$$

Using Hooke's law and substituting (5.6) in (5.4), we have

$$U = \int_V \frac{\sigma^2}{2E} dV = \int_V \frac{1}{2E}\left(\frac{M(x)y}{I}\right)^2 dA\,dx = \int_L \frac{M(x)^2}{2EI^2} dx \int_A y^2\,dA\,.$$

As the area integral represents the moment of inertia of the beam about the neutral axis, we have the strain energy as

$$U = \int_0^L \frac{M(x)^2}{2EI}\,dx\,.$$

To evaluate the strain energy, we must express the internal moment as a function of its position x along the beam. Hence with $M(x) = EI\,v''(x)$ we have the energy functional

$$U(v(x)) = \int_0^L \frac{EI}{2}\left(v''(x)\right)^2 dx\,. \tag{5.7}$$

5.3 Beam Finite Elements

We make use of a finite element method for the implementation of our elastic beam approach. The element stiffness matrix of a straight elastic bar under axial and transversal load is derived using equations (5.5) and (5.7). To build this matrix, the influences of bending and compression are first kept separate, meaning that stiffness matrices are formed for both reactions. The resulting matrices are then merged to express the combination of bending and compression. With this result, we can proceed as described earlier (see Chapters 3 and 4), except for the assembly of elements, where the orientation of the individual beams needs to be respected additionally.

5.3.1 Beam Stiffness Matrix

Element Matrix with respect to Bending Shape functions for beam deflection must reflect the behavior of the possible boundary conditions on deflection v_1, v_2 and rotation θ_1, θ_2 at each point (see Figure 5.7). Hermite's interpolation formula, already used in Section 4.2.1, allows the deflection and its first derivative to be satisfied at each point. With this approach, the deflection at any location along a general beam element may be written as

$$v(x) = H_{01}v_1 + H_{11}\theta_1 + H_{02}v_2 + H_{12}\theta_2 \tag{5.8}$$

with the Hermite polynomials

$$\begin{aligned} H_{01}(x) &= 1 - 3\frac{x^2}{L^2} + 2\frac{x^3}{L^3}\,, & H_{02}(x) &= 3\frac{x^2}{L^2} - 2\frac{x^3}{L^3}\,, \\ H_{11}(x) &= x\left(1 - 2\frac{x}{L} + \frac{x^2}{L^2}\right)\,, & H_{12}(x) &= x\left(\frac{x^2}{L^2} - \frac{x}{L}\right)\,. \end{aligned} \tag{5.9}$$

With $\boldsymbol{H} = (H_{01} \quad H_{11} \quad H_{02} \quad H_{12})^T$ and $\boldsymbol{v} = (v_1 \quad \theta_1 \quad v_2 \quad \theta_2)^T$, we can write (5.8) also as

$$v(x) = \boldsymbol{H}\,\boldsymbol{v}\,. \tag{5.10}$$

As seen in Section 5.2.5 the energy, stored in an element, depends on the deflection $v(x)$ perpendicular to the longitudinal axis. We have

$$U(v(x)) = \int_0^L \frac{EI}{2}\left(v''(x)\right)^2 dx - \int_0^L w\,v(x)\,dx\,. \tag{5.11}$$

The first term represents the strain energy derived in (5.7); the second term is the potential of additional external loadings. Inserting the second derivative of (5.10) in (5.11) results in

$$U(v) = \frac{1}{2}\int_0^L \boldsymbol{v}^T\boldsymbol{H}''^T EI\,\boldsymbol{H}''\boldsymbol{v}\,dx - \int_0^L \boldsymbol{v}^T\boldsymbol{H}^T w\,dx\,.$$

With respect to the principle of minimum potential energy, minimize $U(v)$ with respect to $\boldsymbol{v}$ and set the result equal to zero to obtain the matrix equation that defines the local beam finite element

$$0 = \int_0^L \boldsymbol{H}''^T EI\,\boldsymbol{H}''\boldsymbol{v}\,dx - \int_0^L \boldsymbol{H}^T w\,dx\,.$$

Substituting the second derivatives of the shape functions (5.9) gives in the matrix equation

$$\int_0^L E\,I \begin{pmatrix} -\frac{6}{L^2} + \frac{12x}{L^3} \\ -\frac{4}{L} + \frac{6x}{L^2} \\ \frac{6}{L^2} - \frac{12x}{L^3} \\ -\frac{2}{L} + \frac{6x}{L^2} \end{pmatrix} \begin{pmatrix} -\frac{6}{L^2} + \frac{12x}{L^3} \\ -\frac{4}{L} + \frac{6x}{L^2} \\ \frac{6}{L^2} - \frac{12x}{L^3} \\ -\frac{2}{L} + \frac{6x}{L^2} \end{pmatrix}^T \begin{pmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{pmatrix} dx = \int_0^L \begin{pmatrix} 1 - 3\frac{x^2}{L^2} + 2\frac{x^3}{L^3} \\ x - 2\frac{x^2}{L} + \frac{x^3}{L^2} \\ 3\frac{x^2}{L^2} - 2\frac{x^3}{L^3} \\ \frac{x^3}{L^2} - \frac{x^2}{L} \end{pmatrix} w\,dx$$

Performing the matrix manipulations and integrating gives the stiffness matrix of an elastic beam under bending

$$\frac{EI}{L^3}\begin{pmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{pmatrix} \begin{pmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} wL/2 \\ wL^2/2 \\ wL/2 \\ -wL^2/2 \end{pmatrix}\,. \tag{5.12}$$

Element Matrix with respect to Axial Load The energy stored in an elastic bar under axial load is given by (5.5). It is sufficient to model compression by linear shape functions. With

$$G_1 = \frac{l-x}{l} \qquad \text{and} \qquad G_2 = \frac{x}{l}$$

we can write the displacement $u(x)$ along the main axis of the beam as

$$u(x) = u_1 G_1(x) + u_2 G_2(x) \qquad \longrightarrow \qquad u(x) = \boldsymbol{G}\,\boldsymbol{u}\,.$$

The computation of the local stiffness matrix then follows the same procedure as used above. We obtain the matrix

$$\frac{AE}{L}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} N_1 \\ N_2 \end{pmatrix} \tag{5.13}$$

where N_1 and N_2 denote axial forces acting at the start and end node of the elastic bar.

Combining Bending and Axial Load The axial and transverse deformations are un-coupled for beams when the local axis of the beam coincides with the global axis. We gain the complete local stiffness matrix by combining (5.13) and (5.12). Therefore

$$\frac{E}{l}\begin{pmatrix} A & 0 & 0 & -A & 0 & 0 \\ 0 & 12I/l^2 & 6I/l & 0 & -12I/l^2 & 6I/l \\ 0 & 6I/l & 4I & 0 & -6I/l & 2I \\ -A & 0 & 0 & A & 0 & 0 \\ 0 & -12I/l^2 & -6I/l & 0 & 12I/l^2 & -6I/l \\ 0 & 6I/l & 2I & 0 & -6I/l & 4I \end{pmatrix}\begin{pmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} N_1 \\ R_1 \\ M_1 \\ N_2 \\ R_2 \\ M_2 \end{pmatrix} \tag{5.14}$$

where N_i denote the axial forces, and R_i and M_i the shear and moment end actions dependent upon the external beam loading.

5.3.2 Local vs. Global Coordinate System

To derive the local stiffness matrix of a general element, we placed the (local) coordinate system such that the coordinate axis coincided with the transversal and axial displacements. A beam structure however consists of elements that vary in orientation; currently each beam stiffness matrix is formulated with respect to its own local coordinate system. Obviously, we can not assemble stiffness matrices expressed relative to different coordinate systems. We have to reformulate all stiffness matrices according to a global reference system.

Consider a beam element oriented in the local $\xi - \eta$ coordinate system (Figure 5.8), now placed in the global reference $x - y$ coordinate system. Generally, the task of reformulating an element stiffness matrix in a global coordinate system is not simple, as it requires a re-evaluation of the energy integrals whose value may change by rotation. For our beam application, however, it is valid to make use of the following simplified reasoning (Betten 1997). We define a transformation

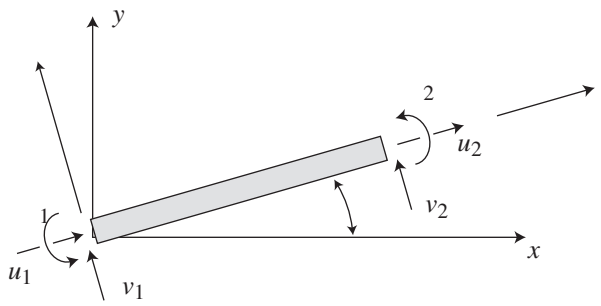


Figure 5.8: Displacement with respect to a local $\xi-\nu$ and global $x-y$ coordinate system.

matrix $\boldsymbol{T}$ that maps global coordinates to local coordinates by

$$\boldsymbol{T} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 & 0 & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & 0 & 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.15}$$

where α denotes the angle that rotates the reference $x-y$ coordinate system onto the local $\xi-\eta$ system. We see that

$$\underbrace{\begin{pmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{pmatrix}}_{\boldsymbol{d}^L} = \boldsymbol{T} \underbrace{\begin{pmatrix} x_1 \\ y_1 \\ \theta_1 \\ x_2 \\ y_2 \\ \theta_2 \end{pmatrix}}_{\boldsymbol{d}^G} \qquad \text{and} \qquad \underbrace{\begin{pmatrix} N_1 \\ V_1 \\ M_1 \\ N_2 \\ V_2 \\ M_2 \end{pmatrix}}_{\boldsymbol{f}^L} = \boldsymbol{T} \underbrace{\begin{pmatrix} F_{1x} \\ F_{1y} \\ M_1 \\ F_{2x} \\ F_{2y} \\ M_2 \end{pmatrix}}_{\boldsymbol{f}^G}$$

where we use the upper index to denote whether the values are expressed with respect to the local (L) or global (G) coordinate system. The relations

$$\boldsymbol{f}^G = \boldsymbol{T}^{-1}\boldsymbol{f}^L\,, \qquad \boldsymbol{f}^L = \boldsymbol{K}^L\boldsymbol{d}^L\,, \qquad \boldsymbol{d}^L = \boldsymbol{T}\boldsymbol{d}^G$$

lead to

$$\boldsymbol{f}^G = \boldsymbol{T}^{-1}\boldsymbol{K}^L\boldsymbol{T}\boldsymbol{d}^G$$

and since $\boldsymbol{f}^G = \boldsymbol{K}^G\boldsymbol{d}^G$, we have

$$\boldsymbol{K}^G = \boldsymbol{T}\boldsymbol{K}^L\boldsymbol{T}^{-1}\,. \tag{5.16}$$

Noting that $\boldsymbol{T}^{-1} = \boldsymbol{T}^T$, we can insert (5.16) in (5.14), which results in the stiffness matrix expressed in global coordinates

$$\boldsymbol{K}^G = \frac{E}{L} \begin{pmatrix} Ac^2 + \frac{12I}{L^2}s^2 & \left(A - \frac{12I}{L^2}\right)cs & -\frac{6I}{L}s & -\left(Ac^2 + \frac{12I}{L^2}s^2\right) & -\left(A - \frac{12I}{L^2}\right)cs & -\frac{6I}{L}s \\ & As^2 + \frac{12I}{L^2}c^2 & \frac{6I}{L}c & -\left(A - \frac{12I}{L^2}\right)cs & -\left(As^2 + \frac{12I}{L^2}c^2\right) & \frac{6I}{L}c \\ & & 4I & \frac{6I}{L}s & -\frac{6I}{L}c & 2I \\ & & & Ac^2 + \frac{12I}{L^2}s^2 & \left(A - \frac{12I}{L^2}\right)c^2 & -\frac{6I}{L}c \\ & \text{symmetric} & & & As^2 + \frac{12I}{L^2}c^2 & -\frac{6I}{L}c \\ & & & & & 4I \end{pmatrix} \tag{5.17}$$

with $c = \cos\alpha$ and $s = \sin\alpha$.

5.4 Interpreting a Road Network as Beam Structure – Algorithm Setup

The motivation for establishing a method for modeling a road network as a beam structure was sketched out at the beginning of this chapter. We interpret a given network as a beam structure, and we are interested in the deformation of the structure resp. road network under applied external loads, which move lines hopefully away from conflict spots.

In contrast to applications in engineering, our model can be extended liberally, as neither a single beam, nor the assembly of beams in a structure, nor the applied loads on the structure, have to behave in a physically reasonable manner: Cartographic lines do not have a physical behavior, but a cartographic one. We use these liberties to carry over cartographic behavior to the beam structure.

The implementation of our displacement algorithm generally follows the implementation of elastic truss deformations by means of a finite element method. Above all, the stiffness matrix (5.17) of a general element and the assembly of such elements to represent a linked and networked structure are retained.

The following points clarify how the algorithm is setup. As the method almost only differs from the snake algorithm in the underlying energy definition, many ideas are taken from Chapter 4.

- **Assembly** The decomposition of a road into elements is given by the straight line segments defined by two successive vertices. After having computed the orientation of a segment, its local stiffness matrix in the global coordinate system is expressed according to (5.17). We discuss in Section 5.5 how the parameters in (5.17) are set in order to produce cartographically reasonable results. The local stiffness matrices are assembled as sketched out in Section 4.3.5. The procedure allows again the assembly of a connected road network. Through this construction, we end up with the global equation system $\boldsymbol{K}\,\boldsymbol{d} = \boldsymbol{f}$. $\boldsymbol{K}$ is the global stiffness matrix, given by the above assembly procedure; $\boldsymbol{d}$ is the displacement vector of the unknown displacements and rotations of the beam ends, and $\boldsymbol{f}$ is the force vector of equivalent computed shear and moment end actions (dependent upon the external beam loading).

- **General Boundary Conditions: Propagation vs. Displacement** A beam structure may be destabilized either by forces or by displacements imposed on the structure. This leads to the same partitioning of *propagation* and *displacement* actions observed already for snakes.

- **Nodal Boundary Conditions** Whether the method is used for propagation or displacement, the network nodes require specific protection imposed by means of boundary conditions on the global equation system. As long as no displacement is incorporated into the stiffness matrix, $\boldsymbol{K}$ stays singular and can not be inverted. It would also not make sense, as the beams structure would simply 'run away' under the forces. The reasoning behind the setup of boundary conditions is similar to the one established for snakes, cf. Section 4.4.2.

  The model properties allow us to specify for a given vertex v_i its displacement x_i, y_i and change in orientation θ_i. For *partition nodes*, both the displacement and rotation are set to zero guaranteeing a smooth transition at partition borders. *Endnodes* are allowed to move, but rotation of the incident beam is prevented ($\theta_i = 0$). A *junction node* is also free to move and we set $\theta_i = 0$, so that the geometry in the junction is preserved.

- **Corrections around Junctions** Again in analogy to the previous chapter, corrections around junctions are separately handled. These corrections – computed in a previous step – are compiled into the global equation system. Though the explicit control of bending and of longitudinal elongation in the beam model (see Section 5.7.1) could possibly provide a tool for efficient corrections of junctions, we believe that an *a priori* understanding and correction is better suited, see Section 4.5.

- **Forces** In order to ensure a minimal distance between roads, we exert deformations on the trusses by means of forces. The procedure for determining these forces is retained from Section 4.4.3. As assessed for the application of snakes already, we notice that the direction of forces given by this model is sometimes too restricted. Only the direction to the closest line is computed, which is not necessarily related to the most reasonable correction direction.

- **Iterative Process** In contrast to most engineering tasks, we do not know the forces we want to apply on the structure beforehand. The forces are varied until deformations are provoked that meet the separation distance. Similar requirements emerged for snakes too. Thus, we keep the evolutional model of Section 4.2.2. The equation system

$$(\mathbf{1} + \gamma \boldsymbol{K})\, \boldsymbol{d}^{(t)} = \boldsymbol{d}^{(t-1)} + \gamma \boldsymbol{f}^{(t-1)} \tag{5.18}$$

  is iteratively solved. The beam structure position computed at time t is used to derive the forces acting at time $t+1$. A discussion of the implication of γ is found in Section 4.4.4.

5.5 Controlling Cartographic Propagation by the Design of Beams

In the following section we discuss the fitness of the beam model to cartography. Cushioning displacement is computed for beams varying in physical character. This gives a good understanding of how displacement is absorbed within such structures. Based on the experience gained, we then address the question of how physical properties can be adjusted locally to improve the results. The finite element method assumes these properties are constant over each element; yet the values may differ among elements.

The behavior of a beam element under load is determined by its physical properties, namely, see (5.17), its modulus of elasticity E, its cross-sectional area A, and its moment of inertia I. The cross-section A controls the stretching and compression of an element under axial load. The bigger this area, the weaker is the impact of a constant force on the element. The bending behavior is determined by the moment of inertia I of the cross section. A higher moment makes the element more resistant to bending. By varying A and I, the susceptibility of the structure to bending and compression is balanced[2]. As a constant term over each bar, the modulus of elasticity E does not influence this balance. Though, it can be used to steer the elasticity between different elements.

5.5.1 Element-wise Stored Energy

The understanding of beam deformation is eased by analyzing the energy that is stored in the elements due to the exerted deformation. We derive in this paragraph the formula that express the energy stored in a single deformed beam element. For this computation, we use the element's endnode displacements and endnode rotations $(x_1, y_1, \theta_1, x_2, y_2, \theta_2)^T = \boldsymbol{d}^G$, which anyway are computed by the finite element method.

These displacements, expressed with respect to the global $x-y$ coordinate system, are first rotated into a local coordinate system, that is oriented along the main beam axis. In accordance to the observation of Section 5.3, the local displacements are given by $\boldsymbol{d}^L = \boldsymbol{T}\boldsymbol{d}^G$, where $\boldsymbol{T}$ denotes the rotation matrix set up in (5.15). We denote these local displacements and rotations by $\boldsymbol{d}^L = (u_1, v_1, \theta_1, u2, v_2, \theta_2)$.

The bending energy E_B and compression energy E_C, stored in the element, are

$$E_B(v(x)) = \int_0^L \frac{EI}{2}\left(\frac{d^2v}{dx}\right)^2 dx \qquad \text{(Bending Energy)} \tag{5.19}$$

$$E_C(u(x)) = \int_0^L \frac{EA}{2}\left(\frac{du}{dx}\right)^2 dx \qquad \text{(Compression Energy)} \tag{5.20}$$

where the functions $v(x)$ and $u(x)$ can be constructed from the displacements $\boldsymbol{d}^L$. $v(x)$ is the linear combination

$$v(x) = H_{01}v_1 + H_{11}\theta_1 + H_{02}v_2 + H_{12}\theta_2$$

[2]We use the term 'compression' as a wild card for both types of axial deformation, namely 'compression and stretching'.

of Hermite polynomials

$$\begin{aligned} H_{01}(x) &= 1 - 3\frac{x^2}{L^2} + 2\frac{x^3}{L^3}\,, & H_{02}(x) &= 3\frac{x^2}{L^2} - 2\frac{x^3}{L^3}\,, \\ H_{11}(x) &= x\left(1 - 2\frac{x}{L} + \frac{x^2}{L^2}\right)\,, & H_{12}(x) &= x\left(\frac{x^2}{L^2} - \frac{x}{L}\right)\,, \end{aligned} \tag{5.21}$$

whereas $u(x)$ is the linear combination

$$u(x) = u_1 N_1(x) + u_2 N_2(x)$$

of the two linear shape functions

$$N_1 = \frac{l - x}{l} \qquad \text{and} \qquad N_2 = \frac{x}{l}\,.$$

Inserting these approximations $v(x)$ resp. $u(x)$ in (5.19) resp. (5.20), we have

$$\begin{aligned} E_B &= 2EI\frac{1}{l^3}\left(3v_1^2 - 6v_1v_2 + \theta_1^2 l^2 + 3v_2^2 + \theta_2 l^2 + \theta_1\theta_2 l^2 + \right. \\ &\qquad \left. 3v_1\theta_1 l + 3v_1\theta_2 l - 3v_2\theta_1 l - 3v_2\theta_2 l\right)\,, \\ E_C &= \frac{EA}{2}\frac{(u_1 - u_2)^2}{l}\,. \end{aligned}$$

E_B and E_C express the energy stored in a deformed elastic bar by means of the nodal displacements u_1, v_1, u_2, v_2 and changes in slope θ_1, θ_2.

5.5.2 Analysis of Pure Bending and Pure Compression

Pure Bending We can only observe pure bending if any longitudinal force is disabled. This requirement is met in Figure 5.9, where a displacement is requested perpendicular to a single line. This straight line has been regularly sampled by 25 vertices.

The course of the propagated curve is such that the second derivative of displacement is minimized. This behavior was expected on the basis of the bending energy definition (5.19) and is, as far as the experiences with snakes show, well suited to simulate cartographic propagation.

The energy stored in the beam elements decreases from the ends towards the center. In the center, almost no bending energy is reported; elements rotate only. This is, according to the bending equations, and validated by the example, an operation that causes no cost[3].

Figure 5.10 shows the same application of displacement for a line with irregular and sparser point sampling. The vertices come to lie on the same position of the curve. The energy distribution also mirrors the same pattern. This example shows that the result is only weakly susceptible to the point sampling. An observation that is generally valid.

[3]By means of the 4 Hermite polynomials used in (5.21) it is not possible to construct displacements that express exclusive rotation; a rest energy is always reported. However, this energy is almost negligible, and thus the model is not perturbed.

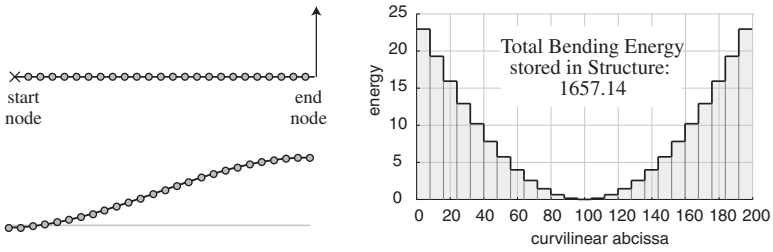


Figure 5.9: Propagation perpendicular to a straight line (left) and the energy stored in the elements (right).

A slight increase of the overall energy is reported. This is due to the decreased dimension of the function space in which the best displacement curve is sought. As the dimension of the function space is proportional to the number of elements, the spanned space decreases with sparser sampling and thus the displacement curve over each elements is modeled in a more energy intensive way.

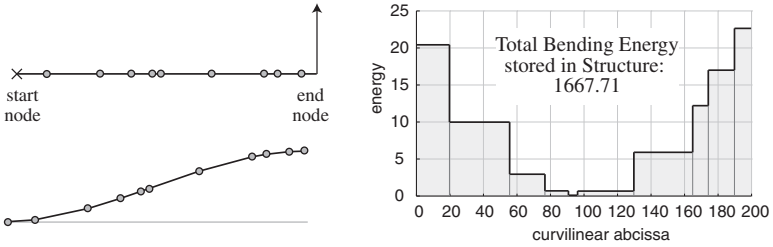


Figure 5.10: Cushioning of the same displacement as in Figure 5.9 in a line with sparser and irregular point sampling (left), and the distribution of energy stored in the elements (right).

To avoid any misunderstanding, we point out that hinges between elements are fixed and do not allow bending. As each segment is visualized by a straight line, and not as a bent curve, we perceive a bending in the hinges. However, this bending was not computed for this hinge, but computed over the interior of the elements.

Pure Compression (Pure Stretching) Pure compression resp. stretching is observed if a longitudinal displacement is applied, see Figure 5.11. Each segment is enlarged proportionally to its length; the energy is equally shared among the segments.

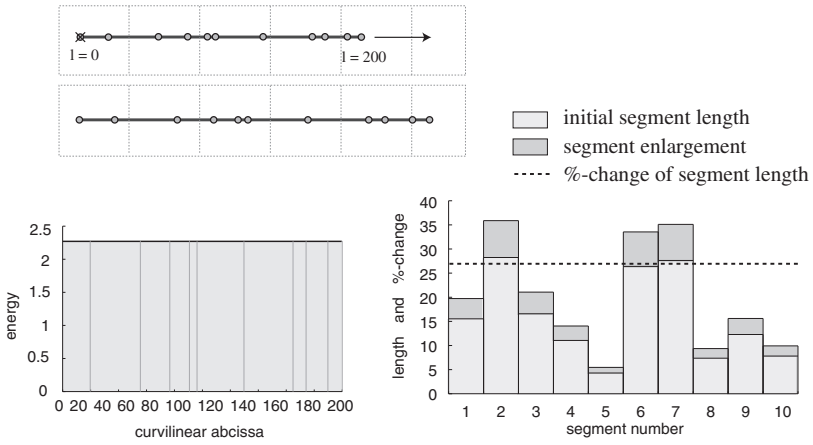


Figure 5.11: Pure Stretching: The energy stored in the elements after deformation is equally distributed among the line segments (left). Therefore, road segments are equally stretched (right).

5.5.3 Combining Bending and Compression/Stretching

In cartographic applications, we never observe pure bending or pure compression. Even if a required shift is perpendicular to a segment – and thus only a bending force is applied on that segment – there are other segments oriented differently, on which the force exerts a longitudinal component triggering compression. The contribution of bending and compression to a solution is controlled via the parameters A and I.

$\boxed{A \gg I}$ By setting $A \gg I$, the willingness of a beam to shrink is almost switched off, see Figure 5.12. A displacement which is oriented perpendicular to the main trend of the road can be applied without releasing high deformation energies. If we compare this cushioning of displacement with a result computed by snakes, the two lines almost coincide.

For a displacement vector pointing along the main axis of the line, a cartographically inappropriate cushioning is computed. The high deformation energy expresses the unwillingness of the structure to be elongated with $A \gg I$. Elongation is achieved by orienting the segments horizontally. The horizontal stretching is accompanied by an attenuation of the bends. A loss of the line's character is observed. The opposite, but still improper, behavior is realized if shrinkage is requested. The bends overshoot as the segments resist a compression. Obviously, if bending is suppressed ($A \gg I$), material is neither 'annihilated' nor 'generated', and thus we run the danger that bends become enlarged or shrunk.

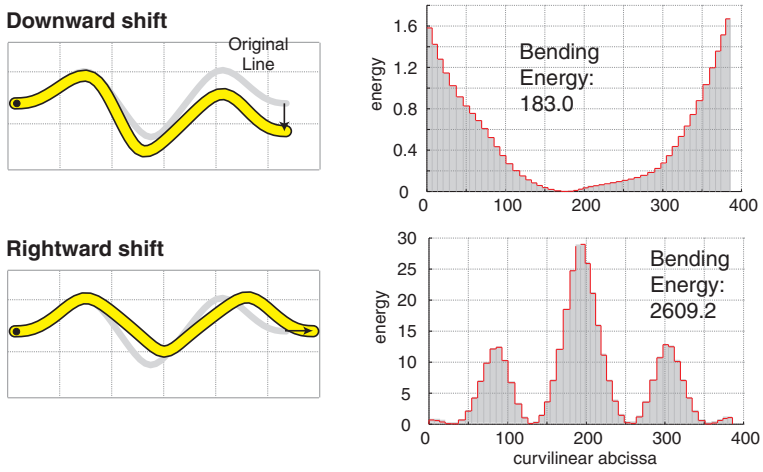


Figure 5.12: Displacement vectors varying in direction applied to a sinuous road. The element stiffness matrices are computed with $A \gg I$; compression and stretching is suppressed.

Snakes would not have any problems with such displacement demands. As the displacement functions are computed independently in the x- resp. y-direction, a horizontal displacement vector does not ask for a shift in the vertical direction. Thus, all y-coordinates stay untouched – a behavior that is completely reasonable here.

$A \ll I$ By setting $A \ll I$, the balance between bending and compression is reversed; segments are stretched and compressed to cushion a given displacement, while bending hardly contributes to the solution.

In Figure 5.13, we observe that the bends become slightly inflated by a request of horizontal displacement. This behavior is explained by the general elongation of almost every segment. Also elements which are not directed exactly along the displacement vector feel some longitudinal force component that causes them to grow. In the mean time, the direction of the segments is hardly altered, as bending is repressed. The result is the observed slight increase of the bend's amplitude. This behavior is cartographically not bad, as it keeps the peak of the bends untouched, meaning that the bends are not opened. Though, it emphasizes the bend as a whole and asks for more space, which is questionable, as space is a scarce resource in cartographic displacement.

$A \approx I$ To take advantage of both element properties, bending and compression, we set $A \approx I$[4]. It proved to be favorable to decrease the resistance against compression slightly more than against bending. However, this observation was

[4]Equations and inequations between A and I used in this section do not give the relations between the two numbers really, but represent their impact on the equation. As both parameters undergo different multiplication, there is no use in making a comparison of their real numbers.

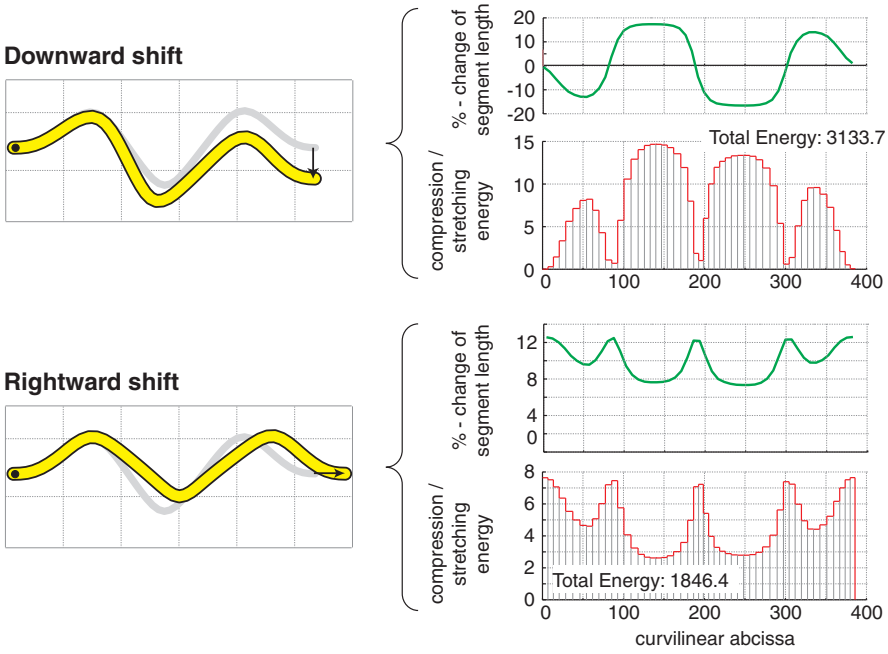


Figure 5.13: Displacement vectors varying in direction applied to a sinuous road. The element stiffness matrices are computed with $I \gg A$; bending is suppressed.

not validated by a representative test – whether it is a general cartographic rule that compression and stretching of lines are less easily perceived than bending and rotation is probably worth further study.

Combining bending and compression for the cushioning of displacement is shown in Figure 5.14. In general, the result is good. Yet, for a displacement required along the main trend of the line, the result is cartographically somewhat inferior to the one gained with snakes. Even though the amplitudes of the bends are preserved, they perform a slight vertical shift as a whole (almost invisible here), released by the constellation at the line ends.

The statement that snakes work slightly better than the proposed beam structure has to be put in perspective. Firstly, there is only a very subtle difference. Secondly, propagation only shows a poorer quality if the displacement vector points in the same direction as the general trend of the line. Cartographic displacements, however, usually point perpendicular to this direction, as the goal is to separate objects. Thirdly, the stated shortcoming is perceivable only if the line shows such

To evaluate the effect of these parameters, we have the choice of either finding the values by trial and error, or of computing these values such that they trigger similar energies on a single element for both types of deformation. Once found, the values can be reused for all applications at the same scale.

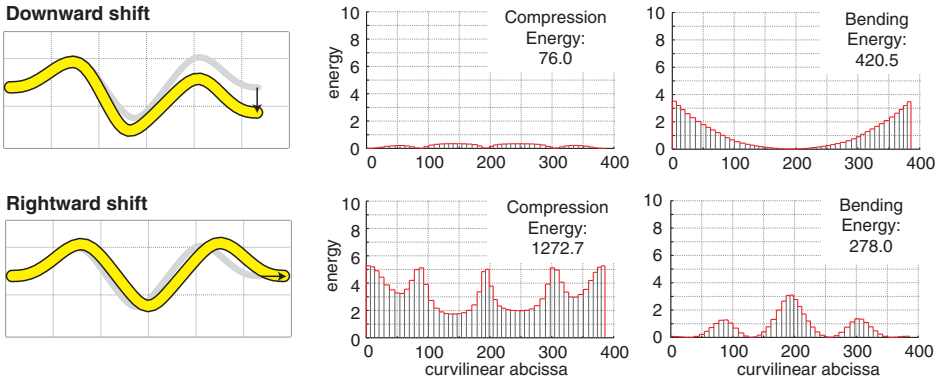


Figure 5.14: Displacement vectors varying in direction applied to a sinuous road. The element stiffness matrices are computed with $I \approx A$; bending and compression/stretching is supported.

a regular geometry as the one used in this paragraph. Lines, as understood by roads in maps, show a more complex and sinuous pattern. The cushioning of displacement(s) for real world roads is illustrated in Figure 5.15. The quality of displacement propagation is good and is barely distinguishable from the results achieved with snakes.

Figure 5.16 illustrates the cushioning of displacement through a road network. Even though not significant here, the model still suffers from the 'heavy node problem', reported already for snakes (Section 4.3.5). In engineering, heavy loads are distributed on pillars to share the force and thus to reduce the deformations. In the same manner are roads behind a junction protected from stronger deformations. The more lines leading into a junction, the more the cushioning is shifted to the part in front of the junction.

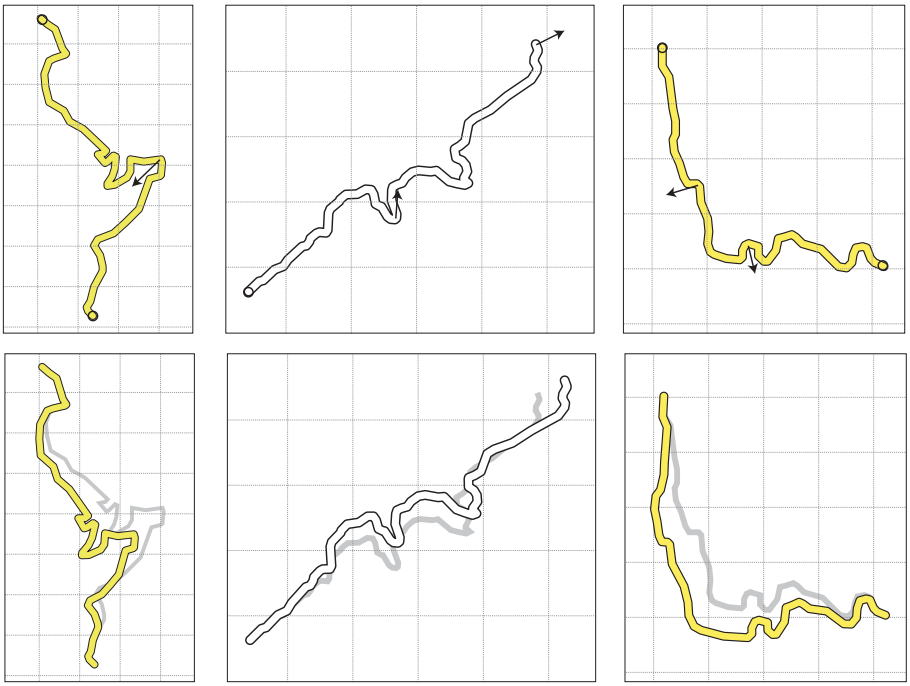


Figure 5.15: Cushioning displacement within roads that are interpreted as chains of elastic bars. More than one displacement vector is applied to the middle and the right-hand road.

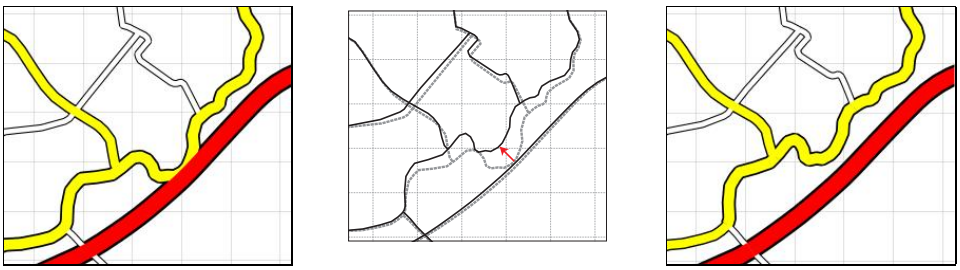


Figure 5.16: Situation before propagation (left). A displacement vector, imposed on the beam structure, is cushioned (middle). Situation after propagation (right).

5.5.4 Cartographic Modeling

The finite element method assumes the elastic properties to be constant over each element. However, the values may vary among elements. We discuss the use of flexible parameter allocation here. An assignment of coefficients that depends on the element characteristics was also used for snakes (see Section 4.3.4). With the snake parameters α and β, this control was limited, since they were imposed on two independently computed functions giving the $x-$ and $y-$displacements. In comparison, the possibility to guide the displacement process by means of the coefficients I, A, and E is more powerful.

In the following, we discuss the option for cartographic modeling by controlling these coefficients. Thereby, the assignment is based on the same cartographic reasoning established in Section 4.3.4. However, the implications are stronger.

Straight Road Sectors

Where possible, it is forbidden to introduce a dent or bend into a straight road sector. This does not mean that such a sector has to remain untouched; a longer straight sector is allowed to be elongated or shrunk without any risk of the overall character of the line being changed remarkably.

$\longrightarrow$

I : increase drastically (avoid bending)
A : slight decrease (improve compression)
E : -

Protecting Cartographic Peculiarities

Narrow bends, bearing the risk of coalescence after deformation, have to be stiffened. They are then resistant to bending and compression. Yet, in contrast to snakes, a rotation of the entire problem area is still possible (see discussion below).

$\longrightarrow$

I : increase drastically (avoid bending)
A : increase drastically (avoid compression)
E : -

Road Attributes

We apply hardly any deformations to highways, whereas side-roads are shifted and deformed more easily. This treatment gave good results for snakes, even though there is no real cartographic reasoning behind it. This argument was established for snakes, where we intended to prevent a shift of straight roads, since it takes then a lot to cushion a displacement and thus bears the risk of provoking new conflicts. However, this case is already handled in our beam structure by stiffening straight sectors, which prevents local bending and thus makes strong forces necessary to trigger a displacement.

$\longrightarrow$

I : -
A : -
E : according to road attribute

We provide an example to illustrate the effect of this setup. However, its main advantages become primarily noticeable when we deal with displacement rather than propagation. In displacement mode, diverse forces are applied to the struc-

ture. Only these make the different resistances of the roads against deformation visible.

Protecting Narrow Bends The protection of a narrow bend that risks collapsing under the impact of displacement is shown in Figure 5.17. With our parameter setup, neither bending nor compression energy is stored in the protected elements. Nevertheless, from the analysis of the segment orientations, we see that the stiff sector was rotated slightly in order to keep the deformations in the other segments low.

Rotation is often a powerful operation that helps keeping overall shape distortion small. In other situations, it is also utilized by the structure to absorb displacement. While working with snakes, a rotation of stiff sectors was not possible. Due to the independence of computations in x and y direction, we had to raise both α and β, which made rotation an energy intensive operation for snakes.

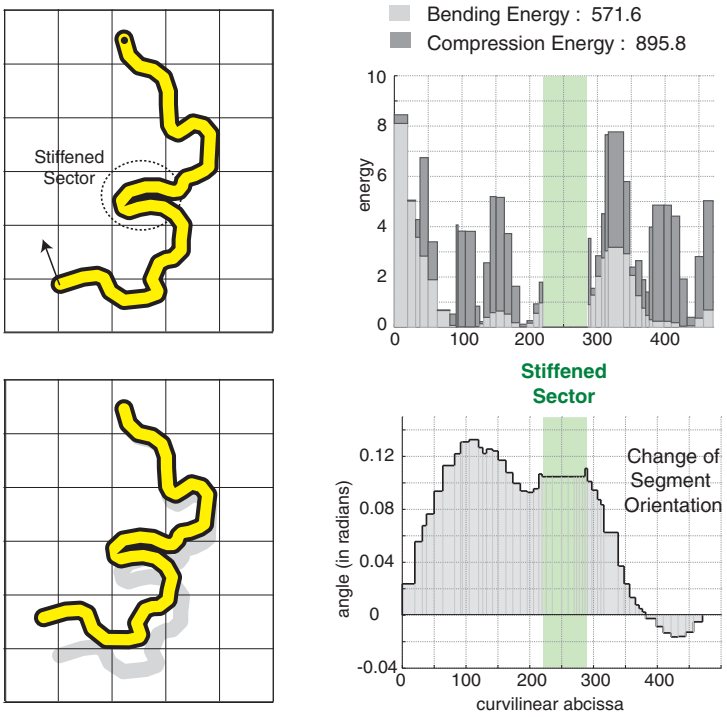


Figure 5.17: A narrow bend has been stiffened (relative increase of A and I) and thus is neither compressed nor bent. Rotation however is possible, as no energy is stored in elements that are only rotated (right).

5.6 Incorporating Positional and Orientational Accuracy

In analogy to Section 4.3.3, we notice that a beam structure uses its entire extent to cushion a given displacement. For the snakes approach, we added an energy term that penalized the line as long as it did not coincide with its initial position.

We follow the same approach here but abstain from deducing the formula explicitely. Following the observations made in Section 4.3.3, we notice that a higher positional accuracy is achieved by adding a scalar term to the main diagonal of the stiffness matrix $\boldsymbol{K}$. We can differentiate between terms that impose positional accuracy, and terms that control the amount of rotation.

Consider the governing equation system $\boldsymbol{K}\,\boldsymbol{d} = \boldsymbol{f}$ resulting from the finite element method. Three successive entries in the displacement vector $\boldsymbol{d}$ are of type u_i, v_i, θ_i, where u_i, v_i express a nodal displacement, and θ_i the change of slope of the elastic beam at this node. If we add a positive value to those diagonal elements of $\boldsymbol{K}$ which shares an index with v_i and u_i, the costs rise to release a displacement for these entries in $\boldsymbol{d}$; higher positional accuracy is imposed. In the same way, a rotation is punished if a scalar is added to those diagonal elements of $\boldsymbol{K}$, which are later multiplied with a slope parameter θ_i of $\boldsymbol{d}$.

The effect of imposing higher positional accuracy on beam structures is shown in Figure 5.18. As expected, the deformation is enforced in the neighborhood of the effective displacement, rather than equally shared along the line. Both compression energy and bending energy decrease with increasing distance from the displaced vertex.

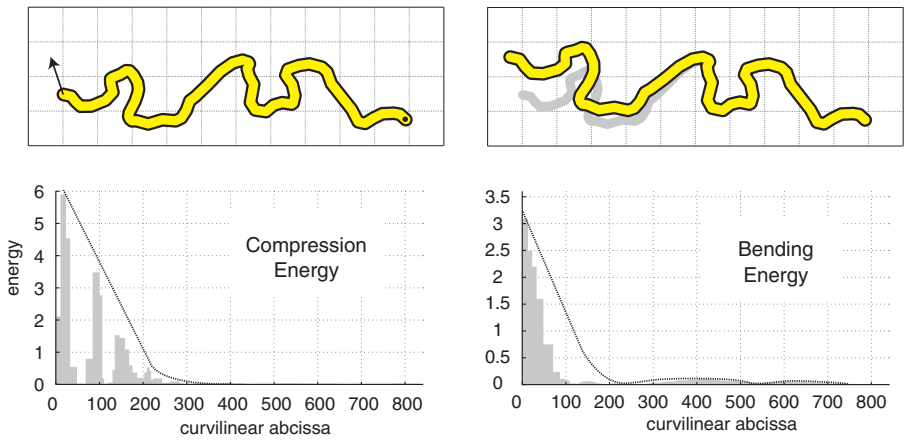


Figure 5.18: Line propagation with additional positional accuracy constraint. The deformation is applied predominantly in the neighborhood of the applied shift, which is reflected by the energy storage distribution of the beam elements. (Those elements, for which no compression energy is reported, are directed perpendicularly to the displacement direction.)

With regard to the preservation of orientation, Figure 5.19 shows that rotation can be prevented effectively – even though there is no good reason to do so in this particular example. We believe that a control of rotation could be beneficial to preserve alignments between roads and their neighborhood, or to control the orientation of segments incident at a junction.

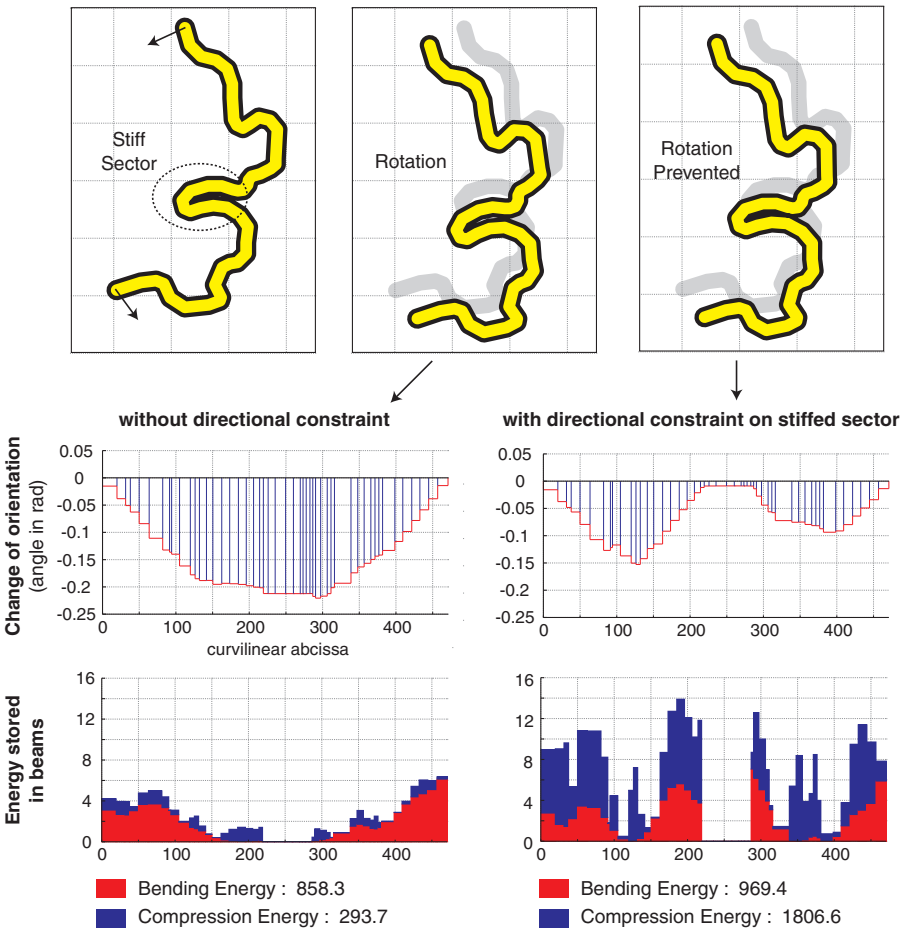


Figure 5.19: The rotation one would expect for the middle bend (see Figure 5.17) is prevented (top right). This is also expressed by the suppressed change of segment direction. However, to preserve the orientation of the bend, neighboring segments had to be bent and shrunk more strongly (bottom right).

5.7 Displacement by Forces

5.7.1 Superiority through Directional Sensitivity

In this section we outline a major benefit of the beam structure, namely that it is, in contrast to snakes, sensitive to the direction in which it is best displaced.

Cartographic Behavior Consider Figure 5.20. The road has been detected as fairly straight. In accordance to the strategy established in 5.5.4, the parameter I, which guides the balance between bending and compression, is increased slightly to avoid strong bending. We impose two displacement vectors on the marked vertex, varying in direction, but not in magnitude. In the propagated roads, the energy, stored in the beams due to the exerted deformation, varies significantly. More energy is stored in the road on which a displacement perpendicular to the principal trend line is applied.

We can also express this observation by means of forces. While a displacement in one direction can be triggered by a small force, it may require a big effort to achieve the same size of displacement in another direction. This is cartographically reasonable. Roads have an 'awareness' of their best displacement direction and therefore exhibit a 'cartographic reaction' to forces.

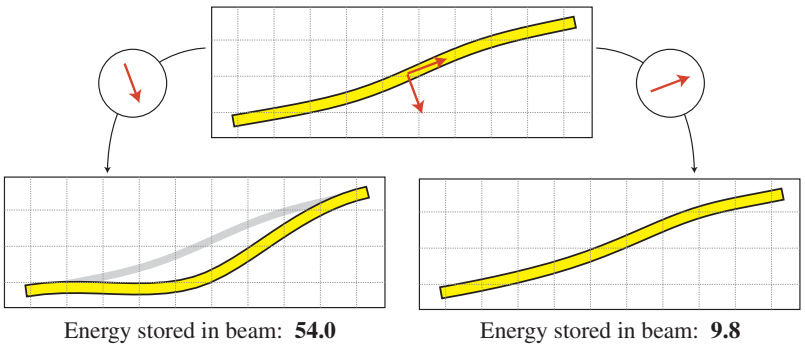


Figure 5.20: On the same line, two displacements, varying in direction but not in size, are requested. The analysis of the results shows that the energy stored in the two beams after displacement varies significantly.

This directional sensitivity is shown in Figure 5.21. Both displayed situations are built upon two roads having the same geometry in the conflict spot (gray shaded area). Therefore, the conflict potential is identical. However, although the forces and the local geometry are similar, the displacements released by the forces vary considerably. The roads are separated in different directions (indicated by gray arrows).

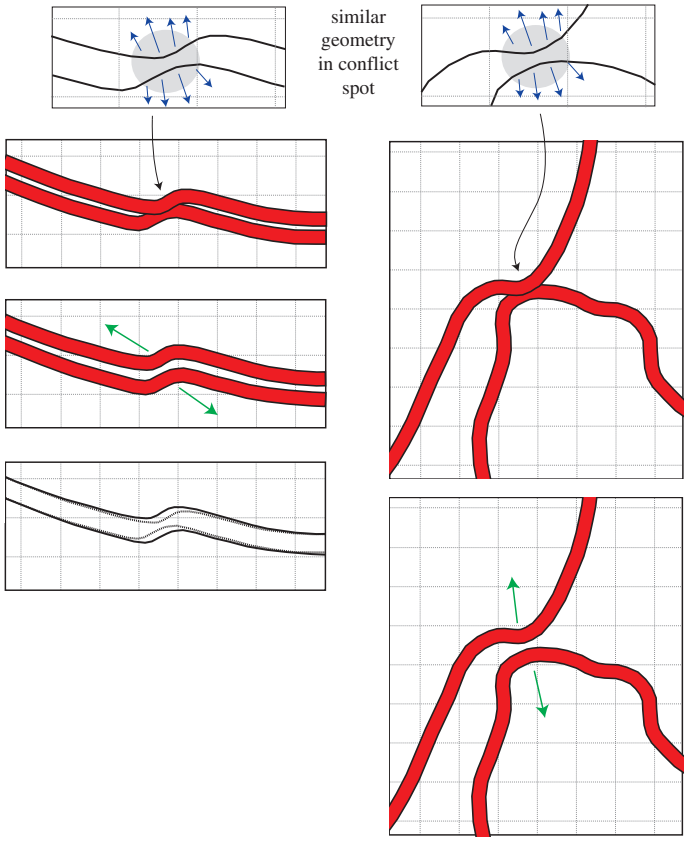


Figure 5.21: Two road partitions, each with two roads in proximity conflict. The geometry of the roads in the conflict spot (gray shaded area) is identical. Thus the same forces, illustrated as black arrows in the top line, are applied to the roads. Yet, the conflict is resolved by separating the lines in different directions (gray arrows).

Directional sensitivity is valuable if a compression of roads is preferred over their bending. In general, such a behavior is desired for lines that are only slightly bent, since a displacement perpendicular to the main run of the line is then cushioned only with difficulty. Hence, privileging compression over bending is useful for most major roads. Also map partitions with an orthogonal, or at least regular pattern of roads, benefit from a preference of compression over bending.

The 'Stolidness' of Snakes Directional sensitivity is not a behavior supported by snakes. This drawback is rooted in the lack of a capability to model the orientation of road segments, which is a consequence of the fact that we work with two independent parametric forms of a line. Working independently with one parameterized line at a time, the concept of 'compression' and 'bending' can not be modeled. The direction of segments can not be determined by considering only one

parameterized form.

Consider, for instance, a straight segment $s = \overline{P_1P_2}$ bounded by the points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. We hold P_1 fixed and ask for the displacements $\boldsymbol{d} = (d_x, d_y)$ of P_2 which extend s, without altering its slope $\phi = \frac{y_2 - y_1}{x_2 - x_1}$. Obviously, this is only possible if $\frac{d_y}{d_x} = \phi$. But since, with the snakes technique, both d_x and d_y are computed independently of each other, we have no control over their ratio, and thus have no influence on the slope of the lines, except if one stiffens them such that no deformation is possible at all.

Consider Figure 5.22. This admittedly artificial situation manifests a perpendicular pattern of straight roads. Using snakes, we raise β to avoid that roads becoming sinuous. But the perpendicular pattern of the situation is lost all the same (see Figure 5.22). Contrarily, the algorithm based on the beam technique resolves the conflict in a cartographically reasonable manner. By having raised the parameter I to impede bending, the problem is solved exclusively by compression and expansion.

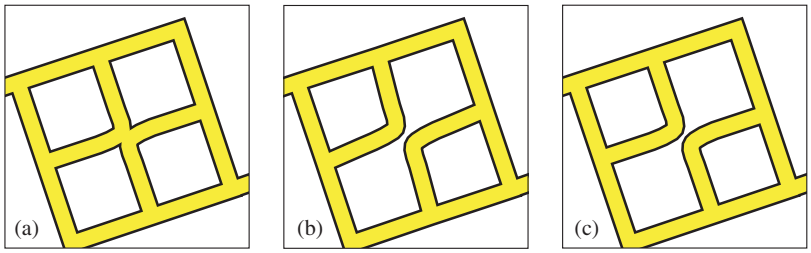


Figure 5.22: Roads in conflict (a). The solution of this conflict by an algorithm based on snakes (b), and an algorithm based on beams (c).

The directional indifference of snakes is further illustrated in Figure 5.23. The example has been used already to show the benefits of the beam technique. With snakes, the roads are displaced in the direction of the effective forces. Snakes do not filter out forces, as they are unaware of directions; stiffening roads does not change this indifference. The result of this behavior is a cartographically undesirable opening of the conflict spot.

5.7.2 Examples and Evaluation

Input Data The solution of proximity conflicts by means of our beam-based algorithm is shown in Figures 5.24 to 5.26.

The following data has gone into computation. First, regarding the setup of the stiffness matrices, we respected the road attributes (roads varying in symbology hold different attributes) to make major roads more resistant to bending (increasing I for more important roads). The parameter A, controlling compression, was not changed. No other line characteristics were used, not even a stiffening of narrow

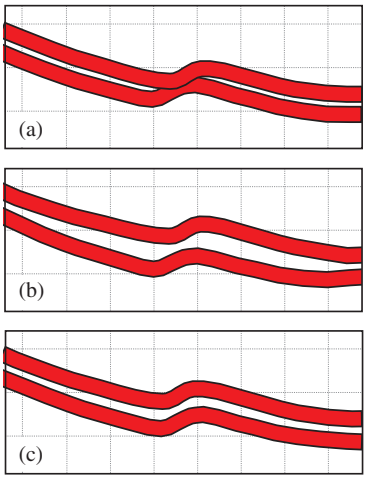


Figure 5.23: Roads in conflict (a). The solution of this conflict by an algorithm based on snakes (b), and an algorithm based on beams (c).

bends, which consequently led to a small symbol coalescence, see Figure 5.26. Before factorizing the stiffness matrix, we added a small term to the matrix to introduce a constraint on the roads' positional accuracy. The default values A, E, and I remained unchanged for all examples. This treatment was possible since the displayed situations are expressed at the same scale. Test data at different scales would make an adaptation of the parameters necessary, since our parameters are scale-dependent.

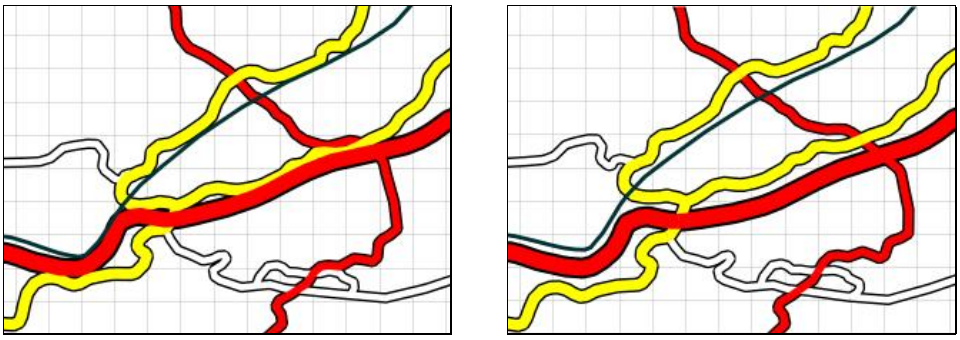


Figure 5.24: Situation before displacement (left) and after displacement (right).

Second, regarding the handling of junctions, we computed local corrections beforehand (as in the snakes case). This correction consisted in the enlargement of those segments that were entirely hidden by the symbolization of other roads ending at the same junction. The pre-computed enlargements were integrated in the equation system as described in Section 4.5.

Third, with regard to the iterative process, we used the displacement triggering forces from the snakes algorithm. Again, the number of iterations is set by the user. All examples are calculated with 200 passes, a number that is often not required, but which guarantees an advanced convergence. The curve of energy decrease shows the same pattern as discussed for the snakes approach. The velocity of displacement propagation, defined over γ, was held constant for all examples. Only the parameter μ by which each force is multiplied, varied from one situation to another. The forces used in example 5.24 did not suffice to deform the roads sufficiently situation 5.25.

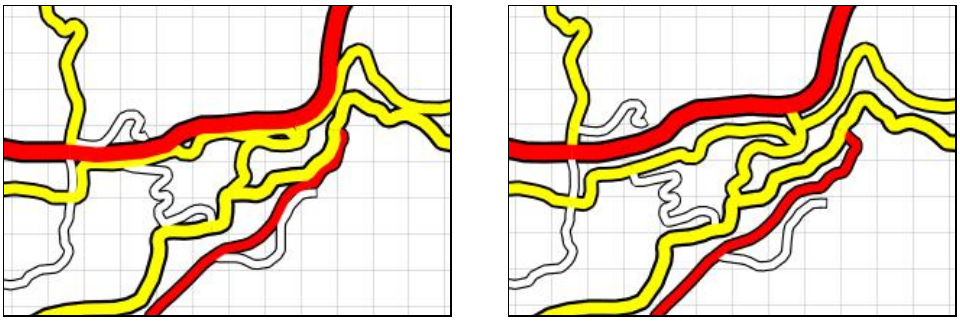


Figure 5.25: Situation before displacement (left) and after displacement (right).

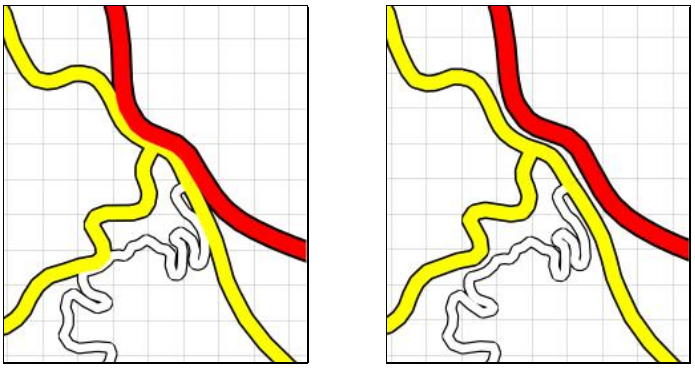


Figure 5.26: Situation before displacement (left) and after displacement (right).

It was not investigated how difficult it would be to automate the parameter setup. But we believe that this task could be accomplished with existing tools. The interested reader is referred to Mustière (2001) for latest progress on this topic.

Quality of Results In general, we observe that the results computed by the beam-based algorithm are very similar to those computed with snakes (cf. Figure 4.20). The similarity of the results is due to a large part to the same method used for the improvement of junctions. It shows again that the solution of proximity conflicts around junctions solves many nearby conflicts at the same time[5].

The effort to find a good balance between the parameters (for beam modeling it is I and A) is higher than in the snake case. The shape parameters α and β of snakes are more strongly related and have less impact; the algorithm is therefore less susceptible to a suboptimal balance of α and β.

The good quality of the results is somewhat surprising as we did not yet make use of the ability to guide the bending- and compression-susceptibility at the segment level. In Figure 5.26, the alignment of the red and yellow line was not preserved accurately. A local decrease of A would provide relief here.

Computing Resources The partition displayed in Figure 5.25 is described by 380 vertices. With beams, the deformation in a vertex is described by 3 parameters, hence a 1140×1140 matrix had to be allocated ($\sim$ 1.3 million entries). In comparison to snakes, the size of data doubled. After the expensive Cholesky factorization of the matrix, the computational effort equals the one of snakes. The force determination, by means of proximity measures between lines, uses the same algorithmically expensive procedure. On a Sun Ultra 30, it required 2.4 minutes CPU-time to compute the solution of Figure 5.25. Obviously, the beam algorithm is slower. Yet, as the number of iterations is a 'soft/spongy' parameter, a comparison is made difficult. For amendments of the method with regard to CPU- and memory-resources, we refer to Section 4.5.3.

5.7.3 Intra-line Corrections

Background In mountainous terrain, roads are often forced to describe hair-pin bends to overcome obstacles. A road covers only a small horizontal distance while it climbs a hill. As a consequence, the symbolization of the road geometry often leads to coalescence (see Figure 5.27). In these cases, conflict occurs not between road symbols of different lines (*inter-line conflict*), but also between the symbology of a single road (*intra-line conflict*).

Intra-line problems are traditionally solved by typification and/or enhancement. Typification, where a series of bends is represented by fewer bends, yet maintaining a similar pattern, is demanding, as it requires the analysis and understanding of a conflict. A well suited algorithm was described by Lecordix *et al.* (1997). Typification is therefore not discussed here.

The solution of an intra-line conflict by means of opening bends, or expanding entire series of bends, is denoted as enhancement. The enhancement of bends comes with the deformation of the road concerned. As a cartographically reasonable

[5]Note again that our prototype does not handle under- and over-passes.

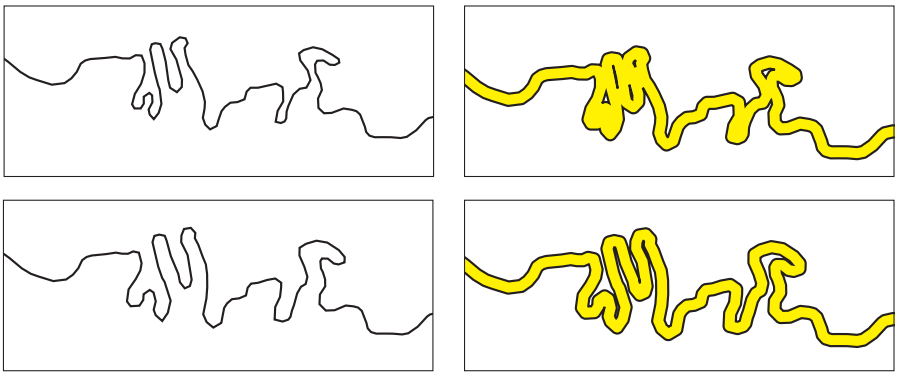


Figure 5.27: Intra-line symbol conflicts (top row) can be solved by widening bends, here computed by our beam-based algorithm (bottom row).

deformation of road geometry is supported by our beam-based algorithm, only a slight upgrading of the algorithm is required to solve such conflicts also. Up until now, the forces which exerted a deformation on a road were released by *other* roads. For the solution of intra-line conflicts, obviously, the forces have to be triggered by the *same* road. Wherever the road is in conflict with itself, a force is released that pushes the two conflicting parts apart. Other changes are not required.

Description of Forces For obvious reasons, a simple distance-oriented view of the conflicts to determine forces fails (each neighboring segment undergoes the minimal distance). Instead, one has to find a criterion to check whether two segments belong to the same side of a bend or not. Our algorithm therefore takes the orientation of segments into consideration.

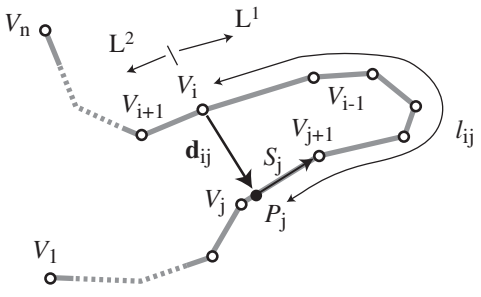


Figure 5.28: Computing intra-line forces.

Consider a directed line $\mathcal{L}$ with n vertices (see Figure 5.28). We determine the force $\boldsymbol{f}_i$ acting on a vertex V_i, $(2 < i < n-2)$. The line $\mathcal{L}$ is split at V_i into two parts $\mathcal{L}^1$ and $\mathcal{L}^2$; both parts exert a force $\boldsymbol{f}_i^1$ resp. $\boldsymbol{f}_i^2$ on the vertex. The sum of these forces $\boldsymbol{f}_i^1 + \boldsymbol{f}_i^2 = \boldsymbol{f}_i$ is the resultant force to apply on V_i. In what follows, we define how the calculation of $\boldsymbol{f}_i^1$; $\boldsymbol{f}_i^2$ is computed in a similar way.

We determine for each segment $S_j = \overline{V_j V_{j+1}}$ $(0 \leq j \leq i-1)$ whether it is in conflict with V_i or not. S_j is in conflict with V_i, if (1) a point $P \in S_j$ is too close to V_i (where the tolerated distance d_{min} is given by the symbol width of the road); and if (2) the oriented segments S_j and $S_i = \overline{V_{i-1} V_{i+1}}$ vary by more than $\frac{\pi}{2}$ in direction.

If a conflict is detected, we compute its severity R_j for V_i. Therefore, we denote by $P_j \in S_j$ the point on S_j closest to V_j. The severity of the conflict is then determined by a relation between the length of the vector $\boldsymbol{d}_{ij}$, connecting V_i and P_j, and the length of the path l_{ij} to travel from V_i to P_j along $\mathcal{L}_1$, where the relation is given by

$$R_j = 1 - \frac{|\boldsymbol{d}_{ij}|}{l_{ij}} \,.$$

This severity R_j is computed for each segment S_j that is in conflict with V_i. The segment S_k with maximal conflict R_k then determines $\boldsymbol{f}_i^1$ by

$$\boldsymbol{f}_i^1 = (d_{min} - |\boldsymbol{d}_{ik}|)\, \boldsymbol{d}_{ik} \,.$$

The computed force $\boldsymbol{f}_i$ is finally weighted by the length of the segment $V_{i-1} V_{i+1}$ to reduce the effect of irregular point sampling.

Evaluation and Outlook Figures 5.27 and 5.29 give examples of the algorithm that was adapted to solve intra-line conflicts and illustrates its capability to deal even with highly complex problems. The opening of the bends is not computed very precisely; existing algorithms may be superior (see Mustière 1998, Lecordix *et al.* 1997), even though the difference at the final publication scale is probably barely noticeable.

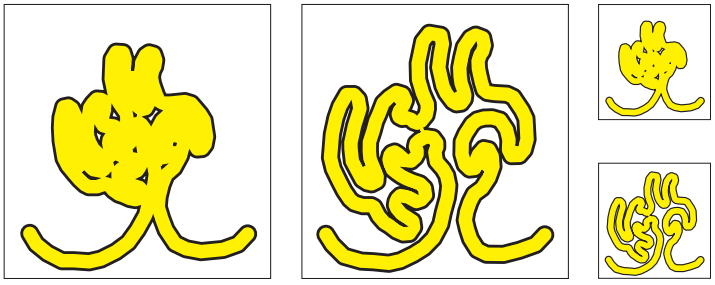


Figure 5.29: The solution of a flower-like, artificial situation by means of our beam-based algorithm.

The advantage of our approach is not its precision, but its robustness on one side (illustrated in Figure 5.29) and its extensibility to the other. By combining intra-line and inter-line inputs to define deformation triggering forces, the enhancement of bends is placed into a contextual environment. We have two operational areas for such an algorithm in mind. First, the incorporation of inter-line relations may trigger an enhancement of bends only in those directions where space for enlargements is available. Current enhancement algorithms do not respect their neighborhood ('independent generalization') and thus fail to respect a constrained area. Second, by adding intra-line inputs to the displacement process, we avoid generating new intra-line conflicts (prior to this, we had to stiffen all narrow bends, even if there was no immediate danger for coalescence).

Whether the interplay of intra- and inter-line forces leads to new algorithmic problems, is not studied further here. However, we are confident that such a combined approach should work reliably.

5.8 Beams and Snakes: Evaluation and Distinction

In conclusion of this chapter, we give in tabular form an overview of the strength and weaknesses of the presented snake and beam methods. Since both methods have similar components, a first evaluation is given for both snakes and beams jointly. Then, this evaluation is followed by a comparison of the two methods. Finally, we summarize the importance of the methods for displacement and give an outlook on possible further improvements.

Strengths of Snakes and Beams for Cartographic Displacement

Shape Both methods deform lines under stress in a cartographically reasonable manner, which means that the character of the roads is preserved. Even a road under the influence of contradicting forces, acting at different positions along the line, is processed without the shape being distorted in an inappropriate way.

Multi-line conflicts The major benefit of both methods is that they handle conflicts between several roads. Proximity conflicts in dense areas are not just pushed back and forth, but indeed solved. The displacement is thereby shared reasonably between the roads.

Network / Topology In contrast to existing methods, displacement is not constrained by immobile junction nodes but allows their displacement too; displacement is propagated through the road network. This is a prerequisite for successful displacement as intersections themselves are often in proximity conflict.

Junctions Junctions are a critical issue in road displacement. The ends of roads, leading into a junction, are often covered by the symbol of the neighboring roads. The arrangement of the junction is then no longer visible. Yet, this is information is of great interest. A local correction is required. Such

corrections can not be computed by simple proximity considerations, but require an analysis of the 'strokes'. Both methods allow the integration of such locally pre-processed corrections which, in combination with the enlargement of small roads, solve many proximity problems in maps.

Modification of Model for Cartographic Enhancement The idea behind both methods stems from other research fields. This allows the use of a fundus of existing knowledge and literature. Both models have then been enhanced and incorporate now a term to weight positional accuracy. The snake and beam-based algorithms both provide the possibility to integrate cartographic reasoning into the computation process. This integration of cartographic knowledge is achieved through an adaptive setting of the shape, resp. material parameters that determine the deformation of the lines under forces.

***Weaknesses* of Snakes and Beams for Cartographic Displacement**

Heavy-Node Problem Common to both methods is the 'heavy-node' problem. Forces are shared among all members coinciding at a junction. The roads behind junctions are therefore protected from major deformations. This comes with an oversized cushioning of deformation in the road in front of the junction. This behavior is often beneficial, but for partitions built up from many junctions with many incident roads, displacement is not propagated sufficiently.

Parameters The methods both require several parameters to express the behavior of roads under forces, to scale the impact of the forces and to guide the iterative solution of the problem. While most parameters hold for all problems at a given scale, the input parameter that weighs the importance of forces had to be adjusted from partition to partition.

Positional Accuracy Our implementation of positional accuracy penalizes each vertex not coincident with its original position. Thus, it does not make a difference whether the vertex is shifted along the line or perpendicularly to this direction. However, a shift along the road should not be penalized; it does not deteriorate positional accuracy.

Energy Minima / Equilibrium of Forces Depending on the complexity of a situation, an equilibrium of forces (resp. a minimization of energy) is found at different levels. The higher the remaining external forces (resp. the higher the remaining displacement potential), the farther away the displacement is from the required threshold. Neither method guarantees that the minimal distance is reached. To overcome this weakness, we could re-apply the method or, probably, a change in the underlying force definition would bring relief.

Forces Our definition of forces does not meet the requirements of all situations. A force should include a component of that direction, in which space for corrections is available. Presently, we only take the nearest point on another line into consideration, a view that is too restricted.

Computing Resources The computational cost is large, both regarding computer memory and processing time. Both methods need to allocate and invert large matrices. In the iterative displacement process, then, the proximity between roads needs to be re-evaluated in each step.

Independency of Data Sampling Neither method depends, *theoretically*, on the sampling of the roads. Denser sampling would improve computational accuracy, but this generally is not an issue in cartographic displacement. *In practice*, the definition of forces is still vertex-based, and thus requires that vertices are placed on all lines participating in a conflict. Also the appearance of a straight sector depends on its sampling. We could get rid of this problem by filtering roads beforehand by means of a Douglas-Peucker algorithm (using 0 threshold). This would not change the flexibility of the line.

While the first three drawbacks in above list are inherent in the method, the remaining weaknesses could be overcome by an extension of the algorithm and a better implementation.

***Comparison* of Snakes and Beams**

	Snakes	**Beam-based**
Two-dimensional Modeling	No real two-dimensional modeling: displacements are computed for the $x-$ and $y-$direction separately.	Full two-dimensional modeling allows a distinction between compression/stretching and bending of elements. Therefore, roads also have a preferred displacement direction, which is highly valuable from a cartographic point of view.
General Cartographic Behavior	Snakes seldom fail to provide cartographically useful results. Though, the result is often not perfect, as the absorption of displacement in straight sectors is made difficult.	Good solutions are only produced if the impact of compression and bending are in balance. In particular, the suppression of compression/stretching may produce cartographically sub-optimal solutions, as material is neither removed nor generated.
Parameter Setup	The impact of the shape parameters α and β is not so strong. Thus, the balance between these values is less crucial for the output.	The line deformations are controlled by the parameters I and A. It is important to achieve a balance between compression and bending. Without element-wise setting, this balance is found without problems. A dynamic allocation of element specific coefficients may pose difficulties. Yet, the parameters have a clear interpretation, which helps their allocation.

	Snakes	**Beam-based**
Control of Rotation	The division of the computation in $x-$ and $y-$displacements allows no control of the orientation of shapes.	The rotation of elastic bars is controllable where required.
Possibilities for Cartographic Control	As the leverage of the shape parameters to the result is weak, the possibilities to control the process by an element-wise allocation of these parameters is less promising. Additionally, the missing two-dimensional treatment of the problem (see above) limits the guidance of the displacement process strongly.	The description of bending and compression would probably allow a detailed control of the displacement process. Further research is required here to make use of its full potential. Currently, only the protection of narrow bends and the allowance of absorption with constrained bending for straighter lines is used.
Computational Costs	High.	Even higher, as six, instead of three, variables are used to describe the displacement over each element.

Summary and Outlook We presented two methods for road displacement in topographic maps. The snake approach originates in computer vision and was introduced to the field of cartographic generalization by Burghardt (2000). The method has been enhanced to allow displacement through entire networks and to incorporate corrections around junctions, and it was studied how a flexible setup of shape parameters makes the method more valuable for cartography. The method is robust and produces good cartographic results. However, it fails to model the problem accurately, as it splits the computation into two directions, which allows no detailed treatment of the two-dimensional character of road partitions.

To overcome these drawbacks, we have built a method that is based on bending and compression/stretching of elastic beams. Our search for a solution in the field of elasticity theory was influenced by Højholt (2000), who was using other ideas from elasticity theory for the displacement of buildings (see Chapter 6). The iterative process of the new algorithm was retained from the snakes technique. This allowed us to combine the advantages of snakes with the advantage of a model that has an explicit control of bending and compression, which makes truly two-dimensional cartographic reasoning possible.

Both methods provide good results. Especially the simultaneous solution of different conflicts, and the cartographically reasonable sharing of displacement among the lines involved, is impressive. It proves the adequacy of optimization methods for cartographic displacement.

The results of both methods are, if used without extensions, quite similar. The snake approach is slightly easier to use, especially with regard to the parameter setting. In contrast, the beam-based algorithm provides a higher flexibility and

is probably more intuitive. We believe that the model, in combination with an analysis of the roads, could be fine-tuned to give even better results.

Through our local analysis of lines, and the associated flexible allocation of shape parameters, we see the next possibilities for improvement. Also, the incorporation of intra-line conflicts could make the algorithm more powerful. We see also room for amendments to preserve the spatial relationships between roads. Currently, only proximity conflicts are resolved. To preserve farther-reaching relationships, one could introduce additional, 'virtual', connections that control the distances between significant vertices.

From an algorithmic point of view, we see two possible improvements of the method. First, the inversion of the stiffness matrix by a standard Cholesky factorization is slow and could be easily accelerated by making use of the matrix' sparsity. Second, we believe that the force computation in each step should be improved, not only to determine a cartographically better direction, but also to be more efficient.

Chapter 6

Building Displacement

This chapter describes building displacement by means of an energy minimizing technique. We first revisit the issues of building displacement, the factors that make it necessary, and the constraints involved in this process. A method by Højholt (2000) is outlined and evaluated. His work influenced our algorithms for building displacement as well as for road displacement. From the analysis of Højholt's algorithm, together with the methodology discussed and approved in former chapters, we synthesize a new algorithm for building displacement, which combines the advantages of different existing methods.

6.1 Revisiting the Issues and Peculiarities of Building Displacement

6.1.1 Necessity for Building Displacement

The reasons for the displacement of buildings away from their true position may be (1) the large increase in the width of all transportation network symbols, (2) the displacement of transportation lines and water courses, (3) the space required because of symbol enlargement of buildings, which are no longer perceivable at their real size at smaller scale, and (4) the increased minimal space required between objects.

The space available for buildings is predetermined by the choice of the symbology of the transportation network. The faces formed by the transportation network create containers (*partitions*) inside which the buildings are trapped and wherein a reasonable displacement must be achieved. Space is limited – and does not necessarily allow a displacement that satisfies all cartographic requirements. A successful generalization of urban blocks is therfore possible only in combination with simplification, typification, elimination, and merging. We are dealing exclusively with displacement here, and assume that the shape of buildings is already enhanced. Therefore, we do not want to change the geometry of buildings, but shift them as a whole (i.e. no deformation of buildings).

6.1.2 Peculiarities of Building Displacement

In contrast to road displacement, neither shape nor topology preservation is an issue in building displacement. The former is excluded by assumption ("no deformation of buildings"), the latter is ensured by the partitions, providing solid boundaries to the problem.

Important in the process of building displacement is the adherence to minimal distances between buildings as well as between buildings and the roads, subject to the constraint of preserving spatial relationships between buildings[1]. Other constraints, such as that the generalization should not lead to a disproportional increased densification of a loosely covered space (SGK Arbeitsgruppe 2001), are not handled here, as displacement algorithms do not have, by definition, the ability to counteract a density problem; Rather, it is the displacement invoking module that judges whether a presented solution is acceptable regarding the modification of density, and invokes other operations (e.g. elimination, typification) where needed.

6.1.3 Optimization Algorithms

If the goal of building displacement is merely the adherence to minimal distances, optimization techniques are predestined for their implementation[2]. By measuring object distances and comparing them to a given threshold, an easy evaluation function is at hand that measures the quality of a candidate solution. An optimization of the displacements with respect to this evaluation function then hopefully leads to a good solution.

Although simple in concept, such an optimization is not easily computed in practice. In map generalization, the number of possible building states is infinite. Even with a discretization of space, this number is still far beyond what we are able and willing to process. Intelligent strategies are required to constrain the number of candidate solutions.

The work of Ware and Jones (1998) and Lonergan and Jones (2001) provides interesting thoughts on how to perform this search for best displacements. Three iterative methods have been tested by those authors. Conflict reduction by steepest gradient descent (Ware and Jones 1998) was least successful. The authors preferred simulated annealing, as it has the ability to escape local minima, and thus provides normally good results. Yet, using simulated annealing, the computational effort increases substantially, as a huge number of candidate solutions needs to be computed. The method was therefore reorganized by Lonergan and Jones (2001). To avoid recalculating too many candidate solutions, 'forces' between buildings are computed, which trigger displacements in direction of the, hopefully, best solution only. This approach reduced the number of moves in comparison to simulated annealing, but obviously complicates the algorithm as more cartographic reasoning enters into the solution. Another algorithm that calculates the quality of a displacement solution merely on basis of minimal distances is the algorithm proposed by Burghardt (2000). Burghardt's iterative conflict solution makes use of the

[1] Preserving relative distances is also an issue for road displacement. Important relationships for roads, however, are rare (except for urban areas, where our algorithms is not intended to be used), and usually consist of parallelism between adjacent lines. Parallelism of adjacent lines is a spatial relation predominantly conserved by our algorithms.

[2] By considering minimal distances only, a method does not attempt to produce the best possible map of a situation, but merely an acceptable one.

Greedy algorithm (Williams and Shah 1992). The 8-neighborhood of each building describes possible new states whereby that point is chosen where the symbol overlap decreases maximally.

Evaluation The iterative optimization algorithms sketched out succeed in finding valid displacements in densely covered areas. We assume that an algorithm based on simulated annealing should also succeeds for a complex situation. The question is whether it is worth paying the high computational cost required. From a cartographic point of view, it is questionable if a displacement algorithm has to find displacements up to any real degree of complexity. Where map space is very scarce, typification and elimination not displacement are the principle operations. Where map space is less scarce, less complex algorithms will also succeed. But in such less densely covered situations, not only minimal distances are an issue, but also the preservation of spatial relations. This constraint, however, is not yet incorporated in the described methods. Results are not of a sufficient quality.

6.2 Elastic Deformation of Space

6.2.1 Background and Principles

A new approach towards building displacement was recently presented by Højholt (1998) (see also Højholt 2000). Højholt's intuitive point of departure is the idea that a map can be modeled as an *elastic continuum wherein different objects compete for space*. This competition is modeled by forces acting within and along the elastic body.

Højholt's method is straight in line with the theory and the applications discussed in former chapters. It also builds the starting point of a new algorithm to be discussed in Section 6.3. Therefore we discuss some implementation issues of this approach.

Computing Elastic Deformation by means of a FEM We consider the map (or at least finite parts of it) to consist of an isotropic elastic material, which is subject to forces in the plane of the map (*plane stress*). The physical description of elastic bodies, together with a methodology to solve this problem, is found in the field of structural mechanics. At the heart of the theory is again Hooke's law, which relates forces to deformations. A numerical implementation of an equilibrium between such forces and deformations is provided by the FEM[3]. Appendix C gives the necessary background to constitute the governing equations.

A valid choice of element for a two dimensional FEM application is the *triangle*; a triangulation of the map space provides these elements (Figure 6.1). The deformation of an element under forces, see equation (C.14), is guided by the modulus of elasticity E and by the Poisson's ratio v, which describes the willingness of the

[3]The title of Højholt's (2000) article 'Solving Space Conflicts in Map Generalization: Using a Finite Element Method' is irritating. The main focus of his work is the adaptation of the model of elasticity – which is studied in solid mechanics – to cartography. The FEM is just the method to solve the basic idea. The FEM can be used for many other applications in cartographic generalization, as shown in this thesis.

material to change its volume. These parameters are assumed to be constant over each element, but may vary between elements.

When assembling the equations defined over each triangle, again an equation system of type $\boldsymbol{K}\,\boldsymbol{d} = \boldsymbol{f}$ is gained, where $\boldsymbol{K}$ denotes as usual the stiffness matrix, $\boldsymbol{d} = (d_{1x}, d_{1y}, d_{2x}, \ldots, d_{Ny})^T$ gives the displacements of the triangle nodes, and $\boldsymbol{f}$ corresponds to the forces applied on these vertices. The problem allows in advance the integration of known displacements (essential boundary conditions). Forces cause the displacements (natural boundary conditions).

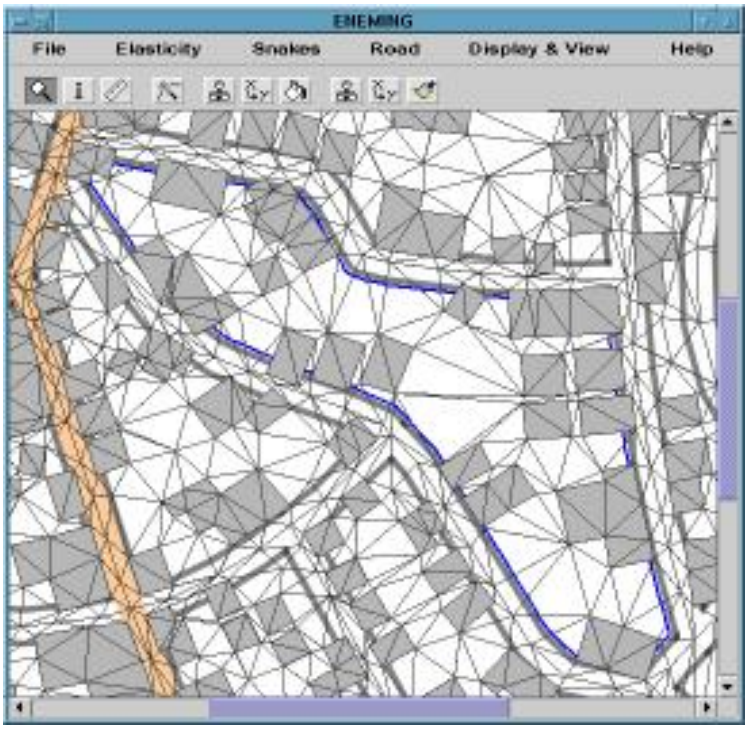


Figure 6.1: Constrained Delaunay Triangulation with few interactively added vertices. The triangulation is computed for each partition separately. (The application name *ENEMING* stands for 'ENErgy Minimization IN Generalization)

Adaptation to Cartography To trigger cartographic displacement and meet its constraints, Højholt discusses the following setup:

- **Stiffening Triangles** Displacement in urban partitions requires that buildings are shifted, and not deformed. By varying stiffnesses of triangular elements, the willingness of these triangles to be deformed, in comparison to their surroundings, is altered. Increasing the modulus of elasticity E of triangular elements that make up part of a building stiffens them with

respect to the surroundings. They are shifted like pebbles in an elastic rubber, not undergoing any deformation.

- **Triangulation Constraints** Stiffening buildings as discussed above is only possible if triangles cover the object's interior. A triangulation has to be formed such that all line segments of the map object coincide with a side of a triangle. Such a meshing is given by a constrained or conforming Delaunay triangulation of all object vertices (Bern and Eppstein 1992, de Floriani and Puppo 1992). But also arbitrary nodes can be added to the triangulation, making the triangles more regular (Figure 6.1).
- **Prohibition of Angle Deformations** For buildings, it is important that their angles remain unchanged. This behavior is indirectly guaranteed already, since a high modulus of elasticity E avoids any deformation. However, the condition can also be directly introduced into the matrix equation, independently of the triangle stiffnesses. This condition has not been used in our implementation of the algorithm.
- **Elements with Expansion** When changing to a smaller scale map, an enlargement of buildings is often desirable to preserve legibility. Such an enlargement was modeled by Højholt by forces acting on building corner points, expanding the buildings.
- **Triggering Displacement** To trigger displacement, forces act along the building partition boundary. These compress the partition such that buildings are sufficiently displaced from the roads.

Implementation Issues From a conceptual point of view, it is obvious that forces best trigger the deformation (displacement) of the elastic body. In practice, there are many ways in which these triggering forces can be implemented.

Højholt (2000) keeps the partition boundaries which coincide with the road centerlines fixed (the displacements of these vertices are explicitly set to zero by essential boundary conditions). The displacement is triggered by exerting forces on the triangles that cover the road area. These triangles are stretched to reach the new width of the road symbol.

We have chosen a different approach. Our implementation looses the elastic partition from the road centerlines; the elastic partition is compressed by exerting forces on all boundary nodes (it is compressed like squeezing an elastic ball in your hand). The partition is held in balance by adding a term for positional accuracy and/or by prescribing displacements for selected vertices. Our implementation approach has been chosen because the data we use does not contain any vertices along the roadside, and therefore no set of triangles that covers the road is available. Further, we believe that the dissociation of the triangles from the road makes the displacement more flexible. On the other hand, our approach is less robust and requires an adaptive setting of forces. However, if we were to continue research on this method, we would stick to Højholt's original setup.

Our implementation further differs from Højholt's in that it does not support an enlargement of buildings and that it uses a manually enriched triangulation. An enriched triangulation makes the triangles more regular and hence less susceptible to inaccuracies and triangle inversions (see also remarks below).

6.2.2 Evaluation

A displacement solution computed by our implementation of Højholt's algorithm is displayed in Figures 6.2 - 6.4. The computation has been performed for one urban block after another.

Figure 6.2: Partitions are compressed to make sufficient space available for road symbols. The exact footprint of the road symbol is not met everywhere at the corners. The partition to the right is insufficiently deformed. Our implementation of Højholt's algorithm has serious problems with narrow partitions.

Strength The method is very intuitive and provides good results if the space for corrections is not too scarce. The guidance of the algorithm poses no problems. *Viewing the map space as cohesive unity has the advantage that it preserves neighborhood relations between buildings.* Alignments, for instance, are widely preserved. However, these spatial relations are not explicitly modeled. They are preserved merely due to the proximity of the buildings. A rotation of buildings is also possible with the elastic model in use.

A strength of the method is the global character of the deformation. Even though the direction of forces, which push on the partition boundaries, are determined locally, the forces are all assembled and evaluated *simultaneously.* This gives a robust and well-fitting model of propagation, which reflects well the holistic character of cartographic generalization. Neither sequential approaches, nor the opti-

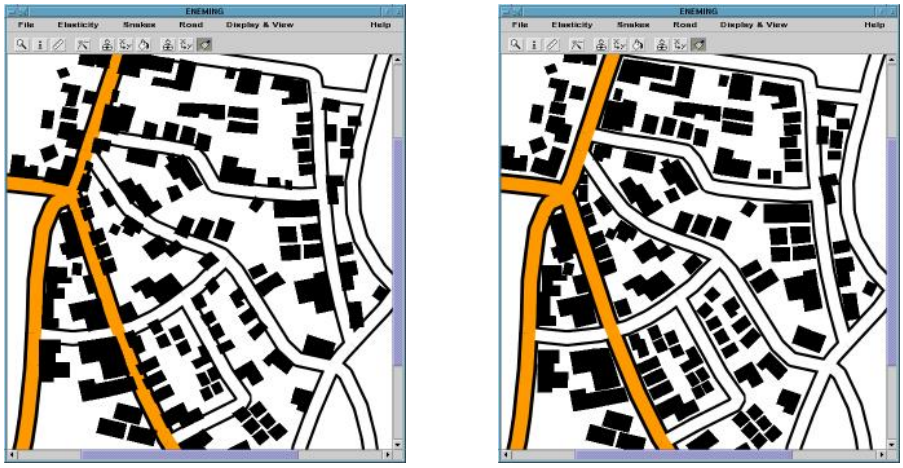


Figure 6.3: Situation before (left) and after (right) displacement displayed with correct symbolization.

mization methods outlined in Section 6.1.3 deal with all influences on the partition at once. Sequential methods traditionally separate displacement and propagation; a truly global view is not achieved. Other optimization algorithms, such as the ones proposed by Ware and Jones (1998) or Burghardt (2000) do not implement propagation at all. Only if new conflicts arise due to an initial displacement, are other buildings shifted. This type of propagation destroys object relationships and is the result of many iterations, by which the process is slowed down remarkably.

The method also provides, with little effort, an analysis of a solution. Strains, emerging in triangles due to deformations, can be analyzed and used to judge the resistance of the partition against the deformation applied (Figure 6.5).

The algorithm is further not restricted to building displacement. *Any type of feature class may participate in the displacement process.* Line vertices, for instance, can also be included into the triangulation. However, a control of the line shape is not possible in the elastic body (unless the model is extended). Note also, that, if we leave the application of building displacement in urban blocks, finding well suited partitions becomes a difficult issue. An *a posteriori* analysis of the strains may help here to evaluate whether the partition chosen was sufficiently large.

Weaknesses Viewing the map as an elastic body is a serious restriction in dense partitions. Building movements are limited; close buildings can seldom be displaced in contradicting directions. The main problem here is the danger of *triangle inversions*. Triangles swap their orientations, as a result of nodes crossing over to the opposite side of the triangles. Højholt also observed this danger. He used an under-relaxation procedure to counteract this problem. Since the method background assumes small displacements only, larger displacements are computed in a 'stepwise' manner. However, we did not only meet problems with inverted triangles

Figure 6.4: To illustrate movements, the initial positions of the buildings are shown in light gray under the final position.

due to computational problems. In Figure 6.3, we see that buildings in partition corners are not displaced sufficiently. Further displacements were not possible however, as this would have led to triangle inversion. Triangle inversion also occurs naturally when buildings are shifted in opposite directions (an observation also made by Jones *et al.* (1995) in the context of his triangulation based algorithm). Also, we met problems in partition corners, where adjacent roads run together acute acute angles. Here, large final displacements are required to move out of the area covered by the road symbols. This makes serious compressions necessary and often ends up in triangle inversions. It would be preferable if some material could be dissolved, rather than only being compressed and shifted. However, this action is not supported by the method. Only a major refinement of the triangulation would provide relief for these problems.

A drawback of the method is also that it does not guarantee minimal distances between buildings (Figure 6.3). In densely built-up partitions, the small space between buildings is additionally compressed, and so buildings almost touch each other. If we were to add building repulsing forces, we would have to change to an iterative process, since the required strengths of inter-building forces are not known beforehand and would need to be adjusted with respect to the displacement triggering boundary forces.

A further difficulty is the determination of forces such that a given displacement is ensured. Exerting the same forces along the partition boundary produces a

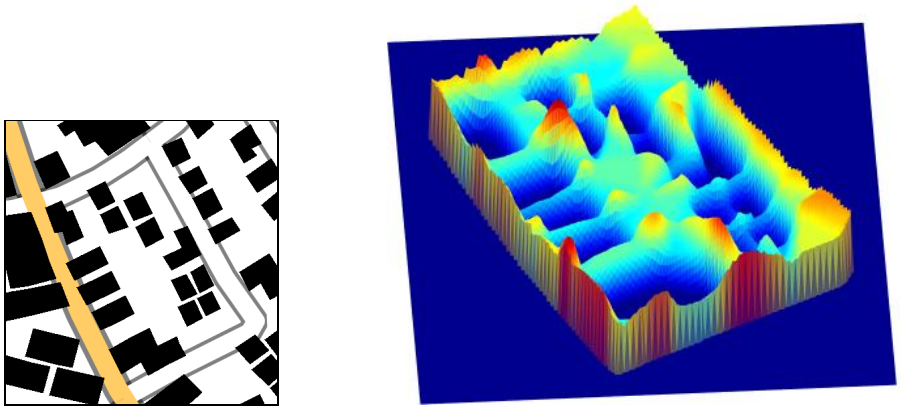


Figure 6.5: Strains provoked by a small compression of the partition illustrated on the left. The strains have been adjusted along the outline such that the same displacement is triggered along the partition boundary. Triangles inside the partition with high strains (red) indicate places where conflicts have to be expected. For comparison, see Figure 6.3, where the partition is illustrated with final displacements.

'jagged' outline. High forces are required where space is scarce; small forces suffice where no buildings with high stiffnesses block the deformation. This drawback is the price to be paid for the simultaneous evaluation of forces: As locally determined forces may be 'overruled' by other forces, local corrections are not performed as intended – displacements respect the impact on the entire partition, even if this behavior is not always desired. To avoid jagged outlines, we adjusted forces iteratively, not carrying out the deformation until a reasonable result was precomputed.

Conclusions The method proposed by Højholt (2000) is reliable and produces good results for loosely build-up areas. It is very intuitive and puts the idea of displacement triggered by forces into practice. The simultaneous evaluation of all body forces models displacement propagation reliably. It inherently preserves the object relationships, even though it is only the distance between buildings that determines their degree of 'togetherness'.

On the other hand, Højholt's method is not especially suited for building displacement in densely covered urban blocks. Solid triangles sometimes limit the movements more than necessary. Problems arise in partition corners and along road bends. A major drawback of the method is further that proximity conflicts between buildings are not included in the computation.

6.3 A Ductile Truss for the Conservation of Building Relationships

6.3.1 Underlying Idea

Motivation The analysis of existing optimization algorithms showed the weaknesses of such methods to model spatial relationships between buildings. We require an algorithm that controls and protects important building relations. A first step towards an inclusion of such relations is the algorithm by Højholt. Yet, the use of a triangulation fails to meet our requirements. First, the triangulation is not a suitable structure to support larger displacements as it is limited by possible triangle inversion. Second, the buildings 'togetherness' is not explicitly modeled. In Højholt's approach, it is only the proximity that determines if buildings share a spatial relation. This is an oversimplification. We can not avoid an antecedent pattern analysis of the building partition with which we quantify the degree of 'togetherness' of buildings.

Regularizing Structure A good displacement algorithm must therefore provide a technique by which it manages pre-computed building relations. The starting point of our algorithm is the idea that we superimpose an elastic truss over the buildings, where buildings that share a spatial relation worth consideration are connected by elastic rods. These rods vary in stiffness. Different degrees of 'building togetherness' are modeled by different stiffnesses. Stiff rods tie those buildings, which share a strong relation, together. Buildings sharing less important relations are either not, or only weakly connected by rods. The varying stiffnesses cause a deformation of the ductile truss in places with fragile links, where displacements do not destroy important spatial relations.

A structure built of linear elastic rods has the advantage over the two dimensional elastic model proposed by Højholt that it allows the explicit modeling of each building relation, and that it allows a less frictional displacement where required.

External Forces The proposed ductile structure is thought to preserve spatial relationships. Obviously, in spite of this preservation structure, a perturbing input is required to shift buildings if they are in proximity conflict to other objects. This leads us again to a model, in which external, problem abating and internal, preserving forces compete for a good displacement solution – a concept used throughout this thesis already.

6.3.2 Algorithm Sketch

In accordance to the detected similarity between building and road displacement, we adopt the mathematical formula and the algorithmic core of the algorithm presented in Chapters 4 and 5. In the elastic truss, each building is represented as one node. Beam elements figure as connecting links. Proximity conflicts between buildings and between buildings and the road are translated into forces that act on the structure nodes (Figure 6.6).

The following schema gives the main steps in the application flow.

1. Build ductile structure	
`1a. build structure geometry`	Compute enriched weighted Minimal Spanning Tree (MST) on the building centroids. The weight of a graph edge is given by the minimal distance between the associated building pair.
`1b. stiffen structure`	Replace the weights of the graph edges by new weights that reflect the 'togetherness' of the associated building pair. Use a measure of collinearity and minimal distance to define this value.
`1c. build stiffness matrix K`	Use graph as template for the elastic truss covering the buildings. We use the snake energy definition in our definition; a beam truss could be used likewise.
2. "Exaggerate" buildings[a]	Replace original building shapes with exaggerated shapes.
3. Iterative Solution $(t = 1, \ldots, n)$	
`3a. compute building forces`	Place buffers around buildings; buffer intersections indicate conflicts and are used to determine repulsing forces.
`3b. compute road forces`	
`3c. merge forces`	Merge all forces acting on one building to a generalized force which acts on the associated vertex of the truss.
`3d. compute displacements`	Use $(\mathbf{1} + \gamma \boldsymbol{K})\boldsymbol{d}^{(t+1)} = \boldsymbol{d}^{(t)} + \gamma \boldsymbol{f}^{(t)}$ to determine the current displacement $\boldsymbol{d}^{(t+1)}$.
`3e. shift buildings`	

[a]This step is not part of the displacement algorithm in the narrow sense. It only simulates more complex generalization, where at this stage of the generalization process geometric modifications at the object level may occur.

The workflow is very similar to the algorithms presented so far. Above all, the evolutionary approach (4.17) from Section 4.2.2 is kept unchanged. Only the procedure for the setup of the stiffness matrix, which holds the buildings together, and the procedure that determines the external forces need to be reformulated. The implementation of these modules is described in the two following Sections 6.3.3 and 6.3.4.

6.3.3 Controlling Displacement over a Ductile Truss

In accordance to the ideas established above, a structure is required which links spatially related buildings and where the stiffnesses of these links (resp. the resistance of the links under forces) mirrors the degree of 'togetherness' between the associated buildings. For this purpose, we build as a first step the topology of this structure, and then assign, in a second step, the stiffnesses of the links to the structure. These two steps are discussed in the following paragraphs.

Figure 6.6: A ductile structure (dark line) holds buildings together and regulates displacements. Proximity conflicts with roads and other buildings trigger forces on the buildings, which are summed over each building and which act on the nodes of the ductile structure (grey arrows).

1. Enriched MST (1a) Assume a partition of n buildings $\mathcal{B}_1, \ldots, \mathcal{B}_n$. The superimposed ductile structure is considered as a weighted graph $\mathcal{G}$, connecting the different buildings in the partition. A graph $\mathcal{G}$ is a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ a set of edges connecting these vertices. A weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ is a graph together with a real valued function $w : \mathcal{E} \rightarrow \mathcal{R}$ that assigns each edge $e \in \mathcal{E}$ a weight $w(e)$.

The set of vertices $\mathcal{V} = \{V_1, \ldots, V_n\}$ in $\mathcal{G}$ consists of the n buildings' centers of gravity. The choice of such vertices representing the buildings is not crucial for the application, because it is not the length of the vertex connecting edges that determines the displacement under forces, but the stiffness later assigned to these edges[4].

The weight $w(e_{ij})$ of an edge e_{ij} of $\mathcal{G}$, connecting V_i and V_j, is given by the minimal distance $d(\mathcal{B}_i, \mathcal{B}_j)$ between the buildings $\mathcal{B}_i$ and $\mathcal{B}_j$ associated to the given vertices, i.e. $w(e_{ij}) = d(\mathcal{B}_i, \mathcal{B}_j)$. Note that it is not valid to infer the the proximity between buildings via the distance $d(V_i, V_j)$. We have to take the actual geometry of $\mathcal{B}_i$ and $\mathcal{B}_j$ into consideration.

At the beginning, $\mathcal{G}$ is complete. $\mathcal{G}$ is then first pruned all but the most important edges. To avoid an end result where buildings 'float randomly' across the partition, we use the edges of the *weighted minimal spanning tree* (weighted MST) as starting point for our truss-like structure. A spanning tree is a subgraph of $\mathcal{G}$ that connects all vertices of $\mathcal{G}$. The MST in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges, i.e. $\sum_{i=1}^{n} w(e_{ij}) = \min$.

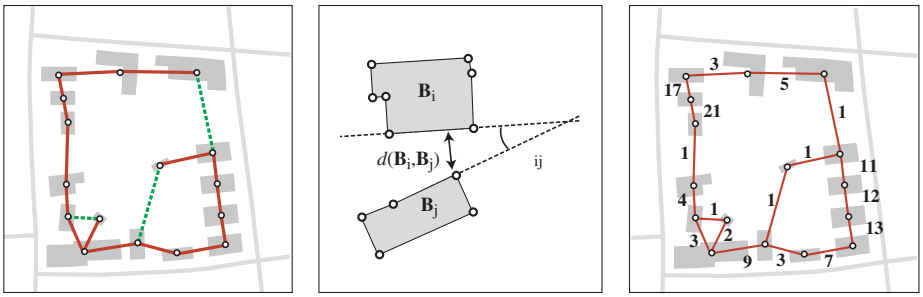
In a second work step, we enrich the weighted MST with important edges. Buildings that are close (again with regard to the proximity of the outlines) are

[4]In particular, as we work with snakes, the direction of the edges is not crucial, as snakes reveal an orientational indifference.

tied by new edges into the graph (if they are not already in there). To avoid adding too many edges, which would handicap deformations and displacement later, only edges which really model a new relationship are introduced. What is a direct relationship, and what relation is already satisfactorily modeled indirectly by existing edges in $\mathcal{G}$, is subjective. For our application, the following criteria has proven to be useful. We insert an edge e_{kl} in $\mathcal{G}$ if

$$d(\mathcal{B}_k, \mathcal{B}_l) \leq d_{max} \qquad \text{and} \qquad d(\mathcal{B}_k, \mathcal{B}_l) \leq \frac{4}{7} w(\mathcal{B}_k \xrightarrow{\mathcal{G}} \mathcal{B}_l)$$

where d_{max} is a maximal search distance (we used $d_{max} = 10 d_{min}$, where d_{min} denotes the minimal distance between buildings, to be defined by the user) and $w(B_k \xrightarrow{\mathcal{G}} B_l)$ denotes the minimal cost to travel from V_k and V_l through $\mathcal{G}$. This path always exists, since $\mathcal{G}$ contains at least the elements of the weighted MST. Figure 6.7a gives an example of an enriched MST as computed by the presented method.



(a) (b) (c)

Figure 6.7: (a) Enriched MST on the buildings' centers of gravity. The edges shown in solid black compose the MST; the edges in dashed grey have been added later. (b) Detecting collinearity between building shapes. (c) The enriched MST with weights expressing the degree of 'togetherness' of the buildings.

Stiffening Structure Links (1b) The stiffness of a link between two buildings has to express how important their spatial relation is. Detecting spatial relationships and patterns in groups is a demanding task. The interested reader can find profound information on this research item in Regnauld (1998). The idea of using a MST-structure as an organizational structure for groups of buildings stems from his work.

We conclude from the *distance* between a pair of buildings and from their *collinearity* their 'togetherness'. This model, in its present state, misses many inputs that influence the importance of a spatial relation (i.e. alignments). Nevertheless, it is a reliable first estimate which suffices for our prototype. If needed, our procedure can be replaced by any other method; what is required at the end is simply a single number that expresses the degree of 'togetherness'.

Consider two buildings $\mathcal{B}_i$ and $\mathcal{B}_j$. As measure of distance, we use the minimal distance $d(\mathcal{B}_i, \mathcal{B}_j)$ between the buildings. To estimate the collinearity of $\mathcal{B}_i$ and $\mathcal{B}_j$, we calculate first the slopes m_i, m_j of those segments for which the minimal distance was reported (Figure 6.7b), and then compute the minimal angle $\varphi_{ij} = \angle(m_i, m_j)$ enclosed by these orientations ($\varphi_{ij} \in [0, \frac{\pi}{2}]$). If the minimal distance was measured on a vertex of the building outline, the slope of both adjacent segments is computed and the segment is taken for which φ_{ij} is minimal. Obviously, φ_{ij} is a measure for the collinearity of the buildings.

The closer the buildings $\mathcal{B}_i$, $\mathcal{B}_j$, and the stronger their collinearity, the higher the stiffness of the edge in $\mathcal{G}$. We scale both the degree of collinearity and the degree of proximity to scores ranging from 1 to 5, and then multiply these values to express the stiffness of the edge $w(e_{ij})$. Figure 6.7c shows the weights (stiffnesses) computed for an enriched MST. The numbers correspond to the weights of the edges, which correspond to the stiffnesses of the elastic rods.

The presented assignment fits for many cases. However, it fails if buildings are not properly squared (collinearity is not detected). Further, it sometimes protects relations that are not really worth being protected. If we were to continue our investigations on the detection of spatial relations, we would explore the use of Voronoi diagrams (see for example Hangouët and Djadri 1997).

Building the Stiffness Matrix (1c) The enriched MST, together with the computed weights define the elastic truss, for which the displacements under external forces are then computed. The stiffnesses of the elastic rods are given by the weights w of the graph. We could now formulate each elastic beam with the formula established in Chapter 5 and assemble a stiffness matrix on the basis of this equation (5.17).

However, we did not integrate these formula, but made use of the energy functional used for snakes instead (see Chapter 4). It doesn't really matter which model we use. We require only a functional that measures the degree of deformation of the structure links. Neither with beams nor with snakes are we interested in the actual geometry of the link deformation: We connect buildings with straight links anyway. Therefore, we do not need to worry about the shape parameters α and β of snakes. We set $\alpha = 1$, $\beta = 0$.

Example Without having discussed forces yet – this is done in the next section – an example is given that demonstrates how the superimposed ductile truss influences displacement. The example is deliberately kept simple as the forces should not yet be an issued here. An exaggeration of small buildings is not applied.

Consider the building partitions illustrated in Figure 6.8; its ductile structure has been illustrated in Figure 6.7c. Due to the supporting structure, the displaced solutions shows two beneficial characteristics.

First, if we take a closer look at the buildings to the left (marked by an A), we see that their alignments and their relative proximity to the road is preserved. Note that a force was exerted on the northern-most building only. The other two buildings moved only due to the superimposed structure.

Second, when considering the group marked by a B, we see that building patterns are preserved. The regular formation of these buildings resulted in rods of equal stiffnesses. These similar stiffnesses caused a uniform deformation of the

Figure 6.8: Buildings before displacement (left). Buildings after displacement (middle). Comparison of building position before and after displacement (right). (The illustrations are screen-shots with superimposed building outlines)

connecting rods, which goes in hand with a uniform spacing of the buildings; the relative distances of the group are preserved.

6.3.4 Forces on Buildings

General Remark on Building Displacement and the Computation of Forces Identifying the direction and magnitude in which buildings are best displaced is not trivial. The displacements are not only affected by the immediate conflict, but also by the neighboring buildings and roads. Approaches in which displacements are computed in a single run have to implement an analysis and displacement strategy that goes beyond the measure of the immediate conflict. It is an advantage of iterative methods that they do not have to specify the conflict resolving translations at once. They have the chance to feel their way iteratively, *adjusting* displacements in each step.

One way of adjusting displacements in each step is to test in which direction the proximity conflict is decreasing most rapidly. With this type of 'trial and error' approach, the probability of building 'dead-locks' is reduced and no cartographic strategy is required to determine the displacements. However, this benefit is paid by more expensive time computations (see for instance Ware and Jones 1998, Burghardt 2000).

In our method, in each iteration we can specify a force that pushes buildings that are too close to either the road or other buildings away from the conflict. The calculation of forces acting on interfering building seems to be trivial on first sight: Compute the points on the building outlines that are closest and connect these points by a vector. This vector defines direction and size of the force pushing these buildings apart. However, such a simple approach does not allow the use of the entire potential of iterative methods. Moving buildings along the vector described by the minimal distance always uses the same two points to determine the direction of displacement; the force direction does not change over time (unless the building relationship is shaken by other influences). The force contains no component that provides a possible way out if the principal displacement direction is misaligned.

We choose an approach for the computation of forces that goes beyond the analysis of the closest points only. By working with *buffers*, we advance deeper into the neighboring context of the buildings and hence gain a better estimate of possible displacement directions. Displacement forces vary more often; the risk of blocked buildings is reduced.

The following two paragraphs describe how forces are computed (for both cases: building ↔ buildings, building ↔ road). How the forces, released from different objects, are merged to one resultant force is discussed afterwards.

Forces: Building vs. Building In a first step, we replace the original buildings by their exaggerated shape. Such an enlargement is often required to keep buildings clearly visible when changing to smaller scale. We applied a general enlargement by adding a buffer of equal width to the original building geometry. The exaggeration process could also be outsourced and combined with a simplification of the outline. It doesn't matter if buildings overlap after this process.

In a second step, a buffer[5] is constructed around the exaggerated building outline. Let us denote the minimal distance between buildings again by d_{min}. Then, the buffer width is chosen to be $\frac{1}{2}d_{min}$. By construction, overlapping buffers mark conflicts (Figure 6.9).

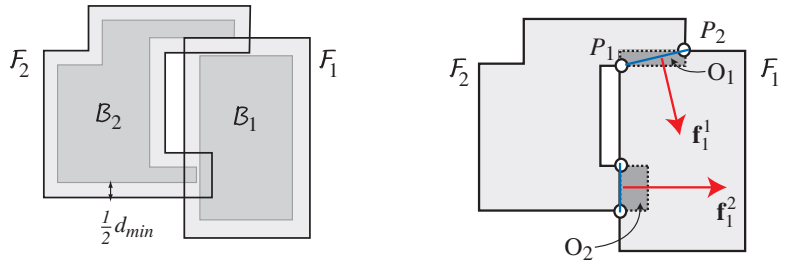


Figure 6.9: A buffer of width $\frac{1}{2}d_{min}$ is drawn around the exaggerated buildings (left). Buffer overlaps indicate conflicts and are used to determine repulsive forces (right).

Consider two buildings $\mathcal{B}_1$ and $\mathcal{B}_2$ with buffers $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively (see Figure 6.9). Intersecting $\mathcal{F}_1$ with $\mathcal{F}_2$ produces a set of overlap areas $\mathcal{O}_1, \ldots, \mathcal{O}_n$ (where mostly $n = 1$, except for complex building shapes). We now focus on an arbitrary overlap area O_i and compute its associated force on $\mathcal{B}_1$ (analogous thoughts lead to the force acting on $\mathcal{B}_2$). There are exactly two points P_1, P_2 on O_i which meet the condition $(P \in \mathcal{F}_1) \cap (P \in \mathcal{F}_2)$. These two points are connected and a vector $\boldsymbol{p}_i$ perpendicular to this connection is computed, i.e. $\boldsymbol{p}_i \perp \overline{P_1P_2}$, where

[5] To simplify and accelerate computations, we used a rectangular buffer around buildings, as illustrated in Figure 6.9. We are aware that this is geometrically wrong, as rectangular buffers may overlap in corners even if buildings are actually separated by more than the minimal threshold d_{min}. Yet, consequences of this inaccuracy are small.

$\boldsymbol{p}_i$ points in the direction of $\mathcal{B}_2$, and $|\boldsymbol{p}_i| = 1$. The force $\boldsymbol{f}_1^i$ induced by O_i and acting on B_1 is then given by

$$\boldsymbol{f}_1^i = \frac{A(\mathcal{O}_i)}{A(\mathcal{B}_1)} \boldsymbol{p}_i \,,$$

where the function $A(\cdot)$ returns the area of an object. Hence, the first term scales the strength of the force. The resultant force $\boldsymbol{f}_1$ acting on $\mathcal{B}_1$ is then given as the sum of the forces computed for each overlap area $\mathcal{O}_i$, namely

$$\boldsymbol{f}_1 = \sum_{j=1}^{n} \boldsymbol{f}_1^j \,.$$

Forces: Road vs. Building The computation of forces which are induced by proximity conflicts between roads and buildings could follow the same principles as established above. We have chosen another approach, which is perhaps not better, but which is slightly easier to code, as intersections of rounded buffers, which come into play when dealing with road symbols, can be avoided.

Consider the outline $\mathcal{R}$ of the road symbol inside the partition, and a building $\mathcal{B}$, whose shape consists of straight segments connecting subsequent building vertices $v_1, v_2, \ldots, v_m$. For each vertex V_i, the vector $\boldsymbol{p}_i$, which connects V_i to the closest point on $\mathcal{R}$, is computed. Further, we denote by $\tilde{d}(\mathcal{B}, \mathcal{R}) = \min(|\boldsymbol{p}_i|, \ldots, |\boldsymbol{p}_m|)$ the shortest distance between any building vertices and the road. If $\tilde{d}(\mathcal{B}, \mathcal{R}) \geq d_{min}$, the building is not in conflict with the road and therefore the process is aborted.

To build the force $\boldsymbol{f}$, acting from the road $\mathcal{R}$ on the building $\mathcal{B}$, we weight the influences $\boldsymbol{p}_i$ by their importance to the conflict, sum these up to a temporary force vector $\boldsymbol{f}_{tmp}$, and scale this force by the overall severity of the conflict:

$$\begin{aligned} \boldsymbol{f}_{tmp} &= \sum_{i=1}^{n} \max(d_{min} - |\boldsymbol{p}_i|, 0) \, \frac{\boldsymbol{p}_i}{|\boldsymbol{p}_i|} \,, \\ \boldsymbol{f} &= (d_{min} - d(\mathcal{B}, \mathcal{R})) \, \frac{\boldsymbol{f}_{tmp}}{|\boldsymbol{f}_{tmp}|} \,. \end{aligned}$$

In comparison to a buffer-based method, this computation has the drawback that it depends on the sampling of the building. However, minor deflections of the force directions from the best directions are not a big problem for the application, as the algorithm has the property that temporarily wrongly directed forces do not have a crucial impact on the final solution.

Assembling forces We do not care where the forces act on the buildings. The computed inter-object forces are just summed for each building to a resultant force which acts on the representative vertex of the superimposed ductile structure. This model is valid if a rotation of buildings is not considered. In Section 6.4, rotation is also enabled. It is crucial, then, to identify where the forces take effect on the buildings.

Example The advantage of a buffer intersection related method over a method that considers minimal distance for the determination of forces is illustrated in

Figure 6.10. The buildings are first exaggerated. If only a consideration of the minimal distance for the computation of forces was made, the buildings would be blocked. The force would point in the same direction during the entire iterative process. In contrast, when dealing with buffers, the force directions are better suited, as they are 'fanciful' with regard to possible corrections. The directions of the forces vary over time.

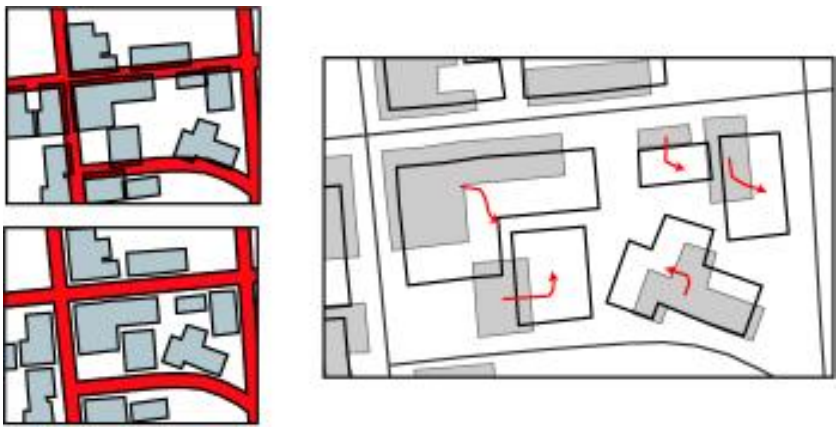


Figure 6.10: Building displacement with forces derived from buffer intersections. The arrows (right) indicate the paths along which the buildings walk over time.

The reader may object that the buffer strategy is probably rich in creativity but may not necessarily displace the buildings in the best direction, when this creativity is not required. If there is enough space available to displace buildings along the path of minimal distance, the buffer overlap method adds a component that is merely incidentally and probably wrong. However, we oppose that this component disappears over time and that, due to the ductile structure, a displacement is revoked if it is not really necessary.

6.3.5 Examples and Evaluation

Figures 6.11 - 6.13 give results of the presented method. Buildings have been exaggerated only slightly[6] Major enlargements would have made the task impractical as our algorithm is limited to displacement. The displacements were then computed for one partition after the other.

The algorithm produces reasonable results. Building patterns are sufficiently preserved. Buildings are also heavily displaced if required. The algorithm succeeds in moving buildings out of corners that are later overlapped by the road symbology.

[6]The building enlargement by adding a buffer of constant width to the initial geometry is only a provisional solution used for the sake of simplicity in this thesis. It produces the undesired effect of decreasing the ratio of building areas. A more adequate enlargement algorithm would try to maintain this ratio.

However, such behavior is not generally guaranteed. On the other hand, in some situations, buildings are displaced more than necessary. This is a poor cartographic behavior, as buildings that before the process where aligned to the roadside should also be aligned to the (widened) road afterwards (see below for a short discussion of this behavior).

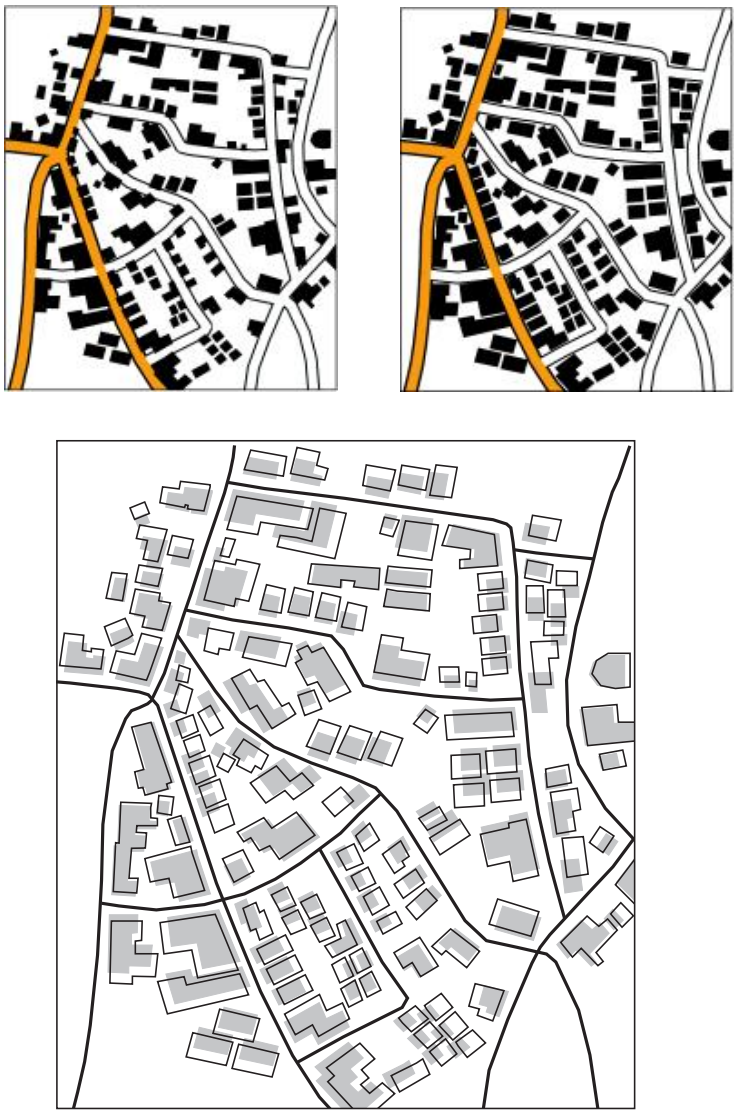


Figure 6.11: Initial situation (top left) and situation with exaggerated and displaced buildings (top right). Comparing the initial and the final state (bottom).

Figure 6.12: Initial situation (top left) and situation with exaggerated and displaced buildings (top right). Comparing the initial and the final state (bottom).

Figure 6.13: Initial situation (top left) and situation with exaggerated and displaced buildings (top right). Comparing the initial and the final state (bottom).

Parameters The following parameters are used for the algorithm. From the map specifications, the minimal distance between buildings and the minimal distance from a building to the road is required. The user has to define how important spatial relationships are for the application ('scaling the stiffness of the elastic structure links'). The algorithm then requires two parameters to set the strength of forces in a reasonable ratio to the displacements in one iteration (these parameters μ and γ have been discussed in Section 4.4.4). The computation is terminated after 100 iterations. *The same parameters have been used for all partitions.*

Strength of Forces The problem requires a hierarchic composition of forces. First, a domination of external forces over regularizing internal forces enables sufficient displacements in most situations. The regularizing forces come only into play if there is enough space to preserve spatial relations. Second, the domination of road forces over building forces makes the buildings meet the minimal distance to the road, even if there is not enough space is left at the end to provide suffi-

cient space among buildings afterwards. We believe this strategy to be useful when viewed within the overall generalization process. It offers other algorithms an estimate of the severity of space shortage and it points out places where buildings will interfere without other generalization operations.

It has been pointed out above that buildings are sometimes displaced more than needed, while at the same time other buildings did not reach the required minimal object spacing. This handicap stems from the method's objective to balance external and internal forces, which also provoked similar deficiencies for road displacement. For instance, a building, that is aligned with other buildings, but sandwiched between two roads, already has to heavily deform the ductile truss to leave the conflicting area only partially solved. At some point, then, the attenuated forces are no longer sufficient to trigger an even stronger deformation of the truss; the building is in balance without having reached the demanded minimal distance. On the other hand, if a building is only loosely connected to the truss, very small forces still provoke a displacement.

Computing Resources Our implementation does not support any spatial indexing. For each building, its proximity to any other building and its distance to any road segment is computed, instead of considering the surrounding building only. The computation of forces is therefore slow; a triangulation would drastically accelerate the computation.

The time required to compute, administer and evaluate the ductile truss is negligible in comparison to the required distance measurement calculations. The size of the stiffness matrix, which implements the given ductile truss, is proportional to the number of buildings in the partition, and hence generally small. The introduction of such a structure therefore does not slow the process down remarkably.

Our algorithm has the advantage over simulated annealing and over the Greedy algorithm that proximity measures are used not only to determine the quality of a candidate solution, but also to determine forces which point towards better building positions. This reduces the number of candidate solutions. We assume that our algorithm, if optimized, is therefore much faster.

Options for Enhancements We see the following options to enhance the algorithm:

- In comparison to simulated annealing and to the Greedy algorithm, our algorithm is more susceptible to buildings that block each other from further displacement. The buffer-related force determination can not prevent deadlocks. Our algorithm could be enhanced in such a way that it alternates the way in which forces are computed between successive iterations. For instance, we could use in every second step, forces that relate to the minimal distance only (instead of buffers). This approach would make the forces more versatile, helping to avoid blocked buildings.

- In the current implementation, the only items triggering displacement are forces. As the buildings are captured in an urban block, they can not escape into other areas and hence no essential boundary conditions are required to constrain the solution[7]. However, even for building displacement, it may be

[7]For road displacement, we had to fix the partition boundary nodes to ensure smooth transi-

beneficial to use essential boundary conditions. We recall that by essential boundary conditions the location of any structure vertex can be predefined in a compulsory way. If the displacement of a building is known in advance, it can be integrated directly into the equation system. Consider, for example, a building close to a junction, which is bounded by coinciding roads that form an acute angle. The building is caged in a location that is later covered by the widened road symbol. There is only one direction out of this cage. This displacement with which the symbolized area is left can be determined in advance. Such a displacement could be integrated in the ductile structure by means of essential boundary conditions. It helps to guarantee the minimal distance requirements, and accelerates computation as fewer iterations are necessary.

However, even relative displacements could be integrated beforehand. If two buildings which share a strong spatial relation are too close, their relative displacement could be specified and integrated into the solution.

- Our algorithm runs into problems with buildings located away from roads and other buildings ('free buildings'). Such a free building does not feel any external force. It is displaced in the same way as the building to which it is connected in the ductile structure, even if the connecting beam is weak and thus no displacement would be required.

  To overcome this drawback, we could enrich the ductile structure with road vertices. This would keep isolated buildings grounded. Such an enrichment of the ductile structure with a road vertex could also be beneficial for a building that is located along a remarkable place of the road (e.g. close to a bend). If the building was connected by an elastic beam to the road, its willingness to leave the exposed position could be decreased.

- Instead of computing displacements for one partition after the other, we could also build the ductile structure over several partitions and compute all the displacements at once. This treatment would have the advantage that spatial relations over partition boundaries are preserved (e.g. alignments of buildings extending across partition boundaries).

- The displacements go in hand with a deformation of the elastic beams building the ductile structure. This deformation is stored as internal energy in the beams (or in the snakes). An analysis of such stored energy could provide valuable information to the invoking module to gain information about the complexity and feasibility of solutions.

  Also, an analysis during the iterative solution could be interesting. If a given energy threshold in a elastic beam is exceeded, the impracticability of displacement for the given situation is recognized ('the beam is broken'). As a reaction, we could eliminate the rejecting forces between the associated buildings and connect them by a weaker rod. As a result, the buildings would overlap, indicating a pair of buildings to deal with later (e.g. candidates for merging), while at the same time leaving more space for other displacements.

tions between partitions.

- As our implementation makes use of snakes, it is not possible to constrain the direction in which buildings are displaced. With elastic beams, we could probably limit some displacement to the longitudinal direction of the truss links (suppress bending in beams), which would better allow alignments to be preserved. However, if one made use of this facility, one would have to consider the setup of the truss in more depth (the geometry of the truss then becomes more relevant), and one would have to improve the analysis of building alignments.
- The method can also be extended to allow rotation. This option is discussed in the next section.

6.4 Rotation

Rotating buildings is not a behavior of high priority. However, it makes a building conglomeration more flexible, which allows one to find solutions of proximity conflicts in partitions with higher building densities. Our rotation respecting method builds on the application presented so far. The method is used exactly as was previously described. Only two new steps are added.

1. Identification of Rotatable Buildings In the setup stage, we compute possible candidate buildings for rotation. Many buildings should not be rotated, as they are aligned to a road. By measuring the minimal distance and the alignment(collinearity) to the nearest road for each building, it is decided whether buildings may be rotated later or not (see Figure 6.14).

Figure 6.14: Buildings shown in dark gray are candidates for later rotation. Buildings shown in light gray are not allowed to rotate, as they are aligned to a close road segment.

2. Computing Rotations During the iterative solution, in each time step, a rotation module operates on the buildings. The principle of this module is the following: Assume a spring that controls torsion. Under stress, it rotates; the more the spring is already twisted, the higher the force required to hold the spring in its state. If no forces any longer act on the spring, it is rotated backward in the direction of its neutral state. We view each building that is a candidate for rotation as such a spring. Still, the buildings are rigid, buildings can only be rotated in their entirety.

The prerequisite for rotation is a moment of force. Such a moment is only available if we consider buildings as area objects, where forces can be located on the outline. To locate forces acting between two buildings, we compute, as before, building buffer overlaps. The force corresponding to a buffer overlap is shared among the two points limiting the buffer (denoted as P_1, P_2 in Section 6.3.4). The repulsing force from a road on a building is located at the building vertex where the proximity conflict was detected.

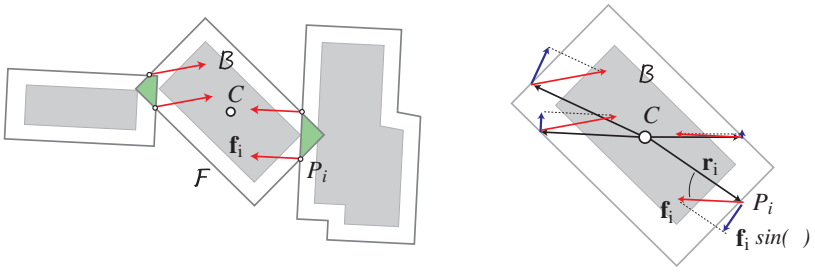


Figure 6.15: Repulsion forces are computed as before, but their working point is assigned to the constituting vertices of the buffers (left). The impact of these forces is studied with respect to a rotation of the building around its center of gravity (right).

After having computed the forces $\boldsymbol{f}_i$ acting on the points P_i of the building buffer $\mathcal{F}$, the total moment of forces $\boldsymbol{M}$ of the building $\mathcal{B}$ with respect to its centroid C is computed (Figure 6.15). We use

$$\boldsymbol{M} = \sum_i \boldsymbol{r}_i \times \boldsymbol{f}_i \,,$$

where $\boldsymbol{r}_i = CP_i$. To respect partially the moment of inertia of the building, the moment of forces $\boldsymbol{M}$ is scaled by $(\sum_i |r_i|^2)^{-1}$. The willingness w of a building to rotate is therefore

$$w_{rot} = \frac{\sum_i |\boldsymbol{f}_i||\boldsymbol{r}_i| \sin\varphi}{\sum_i |\boldsymbol{r}_i|^2} \,. \tag{6.1}$$

Instead of implementing a real spring, we simply scale the willingness of the building

to rotate with respect to its current rotation r_t.

$$w_{rot} := \frac{\pi/2 - r_t}{\pi/2} w_{rot}$$

This has the effect that the higher the current rotation is, the less willing the building is to rotate any further. Finally, we compare w_{rot} to a threshold w_{tol}. If $w_{rot} > w_{tol}$, the building is additionally rotated by w_{rot}, i.e. $r_{t+1} = r_t + w_{rot}$; if $w_{rot} < w_{tol}$, the buildings is rotated backwards, i.e. $r_{t+1} = 0.95\, r_t$.

Results of the enriched algorithm are shown in Figure 6.16. The method requires two additional parameters. First, we have to scale the forces such that the computed willingness w_{rot} in (6.1) has a reasonable magnitude. Second, the threshold w_{tol} is required. Again, for all partitions, the same parameters have been used.

Figure 6.16: Building displacement combined with rotation.

Although the results are good, the current implementation of rotation has several weaknesses and leaves room for improvement. First, it depends on the sampling of the building geometry. Second, the center of gravity is used as a pivot point, which is not necessarily a good choice. Third, the presented implementation of the spring idea is quite a strong simplification.

6.5 Conclusion

The method presented in this chapter combines the benefits of different existing algorithms. At the core, it takes on the idea from Højholt (2000) that displacement can be handled as an elastic deformation. Forces, also used by Lonergan and Jones (2001), act on and along the partition and trigger the displacement.

As Højholt's algorithm runs into problems with larger deformations, the two-dimensional modeling has been replaced by linear elastic rods. This allows both more flexible movements of the buildings and explicit control of building relations. The weakness of Højholt's approach that proximity conflicts between buildings are not taken into consideration is eliminated by passing over to an iterative solution, where also inter-building forces are respected. For it, the iterative snake model has been overtaken.

When judging the quality of the presented method, the position and the importance of building displacement in the process of urban block generalization must be taken into consideration. In this process, displacement is simply one operation among other – perhaps even more important – operations. In dense built-up partitions, it is particularly typification, but also building elimination and building merging, which are applied to counteract spatial conflicts. Displacement is an operation primarily applied in less dense situations, or applied after an initial reduction of building density.

Having this interplay of operations in mind, a displacement algorithm does not have to displace buildings at any cost. But it has to provide good and reliable displacements when sufficient space is available. This requirement is met by our algorithm. It is superior to existing approaches in so far as it also models, incorporates, and preserves spatial relationships between buildings during this displacement process. By its property, that it does not displace buildings at any cost, the algorithm can also be used at the beginning of the generalization work. If insufficient space is available, it points out locations where other algorithms have to be activated.

The major drawback of the method is the inaccuracy with which it meets (or does not meet) the demanded minimal separation distance. A repeated application of the algorithm could bring relief here.

Chapter 7

Conclusion

7.1 Results

7.1.1 Achievements

In Sections 1.2 and 2.6 the goals for this research were defined as improving *optimization methods* for *cartographic displacement* by analyzing and improving existing approaches and by proposing new techniques to overcome assessed shortcomings.

The work was simplified by focusing on cartographic displacement in isolation from other operations. The important interplay with other operations in the generalization process was not investigated. The development of algorithms was further restricted to applications of road and building displacement. We believe, however, that they are representative of the entire range of displacement problems.

Road Displacement One of the starting points of this thesis was the work carried out by Burghardt and Meier (1997), who brought up the idea to use *snakes* for the displacement of roads. Such energy minimizing splines provide an efficient tool to model the behavior of lines under shape modifying external factors. By adjusting the underlying energy definition, Burghardt showed the fitness of snakes for cartographic displacement.

Yet, the algorithm proposed by Burghardt was, from our point of view, not sufficiently adapted to the requirements and problems occurring in cartographic generalization. To overcome this deficiency, we enhanced Burghardt's algorithm in several ways (Chapter 4). Regarding the underlying energy definition, a term for positional accuracy was added. In the assemblage part, the method has been released from the exceedingly limiting constraint of fixed intersection nodes, thus enabling the treatment of complex node situations and large scale reduction factors. To counteract symbolization problems in junctions, an integration of local corrections was made possible. It is not feasible to recognize and induce these local deformations solely by forces in the optimization process; they have to be made available based on a prior analysis of the junctions. Finally, the setting of snake parameters was discussed and fine-tuned to match them to a cartographically well-suited behavior.

The underlying parametric representation of lines used by snakes, however, remained a major drawback of the approach. It impedes a control of the snake's direction, which makes an explicit control of bending and stretching impossible. The lines exhibit a directional indifference. To overcome this drawback, a new algorithm based on *elastic beams* was developed (Chapter 5). Roads are viewed as chains of elastic beams which are connected by stiff hinges. With beams, a truly two-dimensional control of the deformation was achieved. By the explicit control of bending and compression (resp. extension), the nature of roads can be taken much better into account. To trigger and control the displacements, the energy minimizing process of snakes, in which external repelling forces and internal regularizing beam forces are balanced, was reused. The improvements for road networks and local corrections around junctions can be carried out in the same way as in the snakes approach. Further, it was discussed how intra-line conflicts can be integrated into the process.

Building Displacement The second task of this research was to consider and design an algorithm for the displacement of buildings in urban blocks. In urban blocks, cartographic constraints vary considerably from the ones studied and defined for road displacement (Section 2.3.5). The presented algorithm is a fusion of an approach presented by Højholt (2000), in which he modeled the map space as elastic continuum, and the principles approved for beams and snakes already. The buildings in an urban partition are connected/covered by a *ductile truss*, made up of a modified weighted minimal spanning tree, in which the degree of building 'togetherness' is expressed by beams of varying cohesion (Section 6.3). Proximity conflicts between the road and the buildings, as well as between the buildings themselves, give rise to forces that act on the truss. A displacement solution is found again through the interaction of the proximity released external forces with inner, truss regularizing, forces. Finally, an enhancement of the algorithm to incorporate building rotation was outlined (Section 6.4).

Universal Methodology The thesis has produced a universal methodology and shown the usefulness of energy-minimizing principles in cartographic generalization (see also 'Insights' below). It points out a possible way in which cartographic constraints can be introduced into a solution process. The universal methodology presented integrates spatial analysis and problem solution; Conflict detection and conflict resolution are combined.

7.1.2 Discussion of Results

The results presented for *road displacement* by means of snakes and beams clearly exceed the cartographic quality achieved by algorithms known in the literature. Our algorithms have the ability to solve conflicts between several lines, to spread displacements through entire networks and to clear symbol overlaps around junctions, while preserving line shapes sufficiently. In particular, the incorporation of local corrections computed around junctions into a global displacement solution proved to be extremely powerful. Many displacement conflicts are defused drastically if junctions are cleared prior to the actual displacement computation.

Additionally, with beams, we overcame the snakes' weakness of directional indifference. By providing the possibility to guide segment bending and stretching explicitly, cartographically more demanding situations can be reliably processed. However, to make use of these advanced features, parameter setting has become slightly more difficult. While detailed control is now possible, a careless parameter setup ends in results of a lower cartographic quality than if snakes were used.

Room for improvements still exists at the implementation and conceptual level. We did not pay great attention to the determination of external forces. As a consequence, forces are sometimes misaligned and aggravate a better solution. We believe that further research could eliminate this deficiency (see also suggested improvements below). The problems arising due to the high consumption of computing resources can only partially be eliminated. Both methods (snakes and beams) are currently slow. A better implementation could offer some relief. The 'heavy node problem', on the other hand, is inherent in the method (Section 4.3.5). Displacements are not sufficiently spread through road networks with high junction density. The methods may therefore fail in urban areas. This problem can probably best be alleviated by prior analysis and preprocessing by other generalization operators.

The method presented for *building displacement* is capable of handling and monitoring spatial relations. In comparison to Højholt's approach, it also takes minimal distances between buildings into consideration and allows larger displacements. In comparison to other optimization algorithms, which do not model propagation, it is less flexible in arranging buildings if space is rare. The method has difficulties to move objects out of corners, if the exit is blocked by other buildings. We believe this is not a substantial drawback. Proximity conflicts in urban blocks are not solved merely by displacement; complex situations require primarily typification. Once displacement is integrated with other generalization operators such as typification and aggregation, the above problems disappear.

The introduced algorithms all suffer from a lack of accuracy in how they separate objects to a minimal distance. While some objects (roads or buildings) may still be too close, others may have been displaced more than required. This problem is model inherent: a stable displacement is found if internal and external forces are in balance, no matter how strong the repulsive forces are in that equilibrium. Note however that similar issues occur also in manual cartography.

All algorithms presented in this thesis are implemented as *prototypes*. They succeeded in solving proximity conflicts for the given selected, small datasets. Although the datasets used contained a lot of complex situations, a validation of the concepts provided in the prototypes by means of large datasets is still missing.

Associated with this lack of broader experience is our inability to predict possible difficulties to setup suitable *algorithm parameters*. Each presented algorithm contains parameters to stiffen element shapes, to scale forces, and to control the iterative process. Parameters *per se* are not a problem; it is in the nature of cartographic generalization that even the perfect displacement algorithm has to contain parameters by which the user can define the importance of different impacts to the solution. However, a simple way of setting these values is crucial, in the sense that the impact of a parameter's modification is predicatable. We believe that our algorithms meet this criterion, even though our few test-cases do not allow a final judgment.

7.1.3 Insights

We take up the discussion of Section 2.5.3 where we opposed advantages and drawbacks of traditional sequential methods to global optimization methods.

The work confirmed our assumption of the superiority of optimization methods over sequential methods for *road displacement*. A global method *simultaneously* evaluates all the forces that act on (external forces) and within (internal forces) the observed object. The result of this simultaneous evaluation is that, even though forces are determined with a restricted local scope of reasoning, a deformation with global attributes is computed. If some forces are misaligned to the general trend of correction, they are overruled by the sum of the assembled other forces. Displacement is not computed detached from propagation; both processes are merged and executed at the same time. The solution of a proximity conflict is not solved locally, but the 'backyard' (spatial context) of roads is taken into consideration. At the mean time, shape distortion at the local level (e.g. individual bend, junction) is kept under control by the underlying model of lines. This combination of both local and global control gives the power of the presented algorithms.

It is important to notice that our optimization methods do not make cartographic analysis and reasoning superfluous. On the contrary, only a thorough understanding of the road and network shapes allows algorithm parameters to be setup such that the quality of the results is maximized. The benefit of optimization techniques is that they can manage all the information which is locally specified and put it in a global context. With sequential methods, we do not have the tools and concepts available to let all the available cartographic knowledge flow into a displacement computation.

Judging the fitness of sequential algorithms for *building displacement* shows a slightly different picture. The costs for the simultaneous evaluation of forces in the presented optimization methods, and the associated preservation of global pattern, are paid by less flexible adjustments at the local level. Local forces may be outvoted due to remote effects; an intended correction is not performed. Ruas (1999, p.178) is right when she says that it should be due to a local analysis and violation of constraints that buildings are shifted. By considering the immediate neighborhood of a conflict, the scope is often wide enough to apply cartographically well suited displacements in urban blocks. Hence, sequential methods are still suitable candidates for this kind of process, as long as they are equipped with appropriate reasoning capabilities.

This does not mean that optimization methods do not match the requirements of building displacement. Their eligibility is justified by the fact that they may find also displacements for very constrained partitions and that they allow additional spatial relations to be incorporated, steered and preserved more easily. Also bigger displacements than the ones shown here are easily possible with the evolutionary kind of solution presented in this thesis.

In this thesis, we only made use of a small subset of possible optimization techniques. The choice is well justified by the suitability of the approach of the presented algorithms by which displacement is tackled. The interplay of external and regularizing internal forces reflects in an elegant and intuitive way what is happening in displacement. It combines the ability of local reasoning with its holistic counterpart to comprehensively deal with a cartographic situation. The iterative way in which forces compete for a solution allows them to be re-adjusted

over time. This reduces the cartographic demands when forces are first established. If corrections are temporarily not appropriate, they are compensated for later, as the structures in use strive towards their initial position when they are not subject to external forces.

7.2 Outlook

7.2.1 Suggested Improvements

The concepts and methods shown in this report can be extended in a number of ways to improve their workability, broaden the scope of treatment, and fine tune displacement results. These possibilities are discussed with regard to the conceptual level on which they are active.

Improvements at the Algorithmic Level

- The focus of this thesis in the experimental part was on functional experimentation not efficiency. Consequently, the algorithms pose high demands regarding computer resources. For further tests, an efficient implementation is essential (indispensable). In the setup stage, a re-implementation should make use of the matrix sparsity, both for the allocation of arrays (memory) and for the solution of equations by means of a Cholesky factorization (CPU-time). In the iterative minimization process, it is crucial to compute proximity conflicts efficiently (e.g. using spatial indexing).
- The derivation of repulsive forces was not exhaustively investigated in this thesis. This process could be enhanced at two levels. First, we could improve the cartographic reasoning by which the forces are determined in each step. Second, we could pay attention to the evolution of forces. Forces could then be adjusted dynamically or furnished with a random component to give rise to better displacement directions or to de-block buildings.
- It has been pointed out that an equilibrium of internal and external forces may be found at different levels; displacements are then advanced to different degrees. To increase the quality of the solutions, and to make the parameter setting (and hence the results) more predicatable, techniques must be developed to guarantee the same minimal distances within one and among different partitions. One solution to this problem is the repeated execution of the algorithms, by which the force equilibrium is pushed to a lower level (Section 4.4.3). How far a repeated execution disturbs initial spatial object relations has not been studied here. Alternative solutions involve a dynamic adjustment or an exponential decay of the force strengths.
- To a large extent the road displacement algorithms owe their good quality to the inclusion of previously computed local corrections around junctions. The snake and beam algorithms both provide an interface to include these corrections in the global solution. However, the preceding analysis and displacement suggestion has to be made available by an external procedure. Our prototype deals with simple cases only and does not respect under- and over-passes. Further research is required here. With the use of beams, it is

possible, in principle, to readjust the orientation of roads also incident to junctions.

Extending the Algorithms to other Feature Classes

- The work reported in this thesis was built upon the assumption that road and building displacement are representative for the entire range of displacement problems (or at least the overwhelming majority). This assumes that the developed algorithms are easily adjustable to the constraints of other feature classes. A proof of this claim is still outstanding. The adequacy of the beam and snake algorithm for polygonal subdivisions (such as the polygon mosaics of categorical maps, e.g. landuse in geology) is currently a matter of research in our group.

  Beams could be of general use also in the process of building generalization. They allow the deformation of segments to be constrained to pure elongation/stretching, which preserves the perpendicular angularity of a building well.

- The algorithms designed in this work are based on the same foundation (i.e. energy minimization). In principle, a combination of the two approaches is therefore possible: One could deal with building and road displacement at once. Relaxing the concept of immovable streets in the generalization of urban blocks, for instance, is suggested by the SGK Arbeitsgruppe (2001) as a possible solution in certain cases if other displacements fail.

Embedding the Algorithms

- We envision our algorithms to be embedded in a comprehensive, high-level system, in which algorithms are invoked, supervised and orchestrated (e.g. the AGENT platform). With respect to such a concept, algorithms are only of use if the delimitation of work space (i.e. the partitioning) and the setting of parameters can be processed automatically. Partitioning is still an unresolved issue for road displacement. The latter point is still to be studied for our algorithms; but related work in the AGENT consortium using machine learning tools has generated encouraging results (Mustière 2001).

- All of our methods provide valuable information about the success of a displacement solution: The inner energy stored in the model at process termination mirrors the strength of the computed deformation. This information is of use for the algorithm invoking module to rate the quality of a solution. Furthermore, it provides knowledge about the object relationships and could thus be of help for choosing other generalization operations.

- We have pointed out the limits of the displacement operation several times. The presented algorithms unfold their virtue only if applied to datasets, which have been prepared by other operations to the displacement at issue. This interplay with other generalization operators is not sufficiently investigated at present yet.

7.2.2 Concluding Remarks

In recent years, research for automated cartographic generalization made substantial advances. Especially the results achieved in the AGENT project showed that the understanding and implementation of the generalization process has reached a level where its automatization is no longer an utopia.

The AGENT prototype put together the achievements of a decade of research – and this with a good deal of success. From a scientific point of view, the AGENT project helped to identify deficiencies in the state of current research, and put a system at disposal where concepts and new ideas can be integrated and verified.

To reach the quality demanded by national mapping agencies, many advances are still outstanding. One prerequisite for successful improvements, and for automation in general, is a foundation of reliable algorithms. This foundation is only now partially being built. In particular for the implementation of contextual operations, approved concepts are still missing. It has been the author's intention to contribute to this foundation of algorithms by pinpointing possible new techniques for the automation of cartographic displacement.

Appendix A

The Roots of Snakes: Applications in Computer Vision

Snakes, also called Active Contour Models, were designed for the extraction of contours in raster images; they were first proposed by Kass *et al.* in 1987. A snake is an energy-minimizing spline guided by photometric and geometric constraints. Due to the photometric constraints the snake is pulled by image features; snakes lock onto nearby edges, localizing them accurately. The geometric constraints control the tension and the rigidity of the optimizing snake and force its curve to be smooth (Laptev 1997).

A.1 Basic Snake Behavior

The position of a snake is represented parametrically by

$$\boldsymbol{v}(s) = (x(s), y(s)), \qquad 0 \leq s \leq 1,$$

where s is proportional to arc length, and x and y are the curve's image coordinates. In order to find the optimal position of the snake, its state is represented as sum of energies

$$E_{snake} = E_{int}(\boldsymbol{v}) + E_{image}(\boldsymbol{v}), \tag{A.1}$$

where E_{int} represents the internal energy of the spline due to bending, and E_{image} gives rise to image forces. Optionally, a third energy E_{con} could be added representing external constraint forces. The problem of the optimization of the snake's position is equivalent to the minimization of its energy.

Internal Energy The internal energy makes it possible to introduce geometric constraints on the shape of the snake. It can be defined as

$$E_{int} = \frac{1}{2}\int_0^1 \alpha(s)|\boldsymbol{v}_s(s)|^2 + \beta(s)|\boldsymbol{v}_{ss}(s)|^2 ds\,. \tag{A.2}$$

The spline energy is composed of a first order term controlled by $\alpha(s)$ and a second order term controlled by $\beta(s)$. The first order term makes the snake act like a membrane and the second-order term makes it act like a thin plate. Setting for example $\beta(s)$ to zero at a point allows the snake to become second-order discontinuous and develop a corner.

Image Forces The image energy of the snake can be defined as

$$E_{image} = -\int_0^1 P(\boldsymbol{v}(s))ds\,. \tag{A.3}$$

where $P(\boldsymbol{v})$ is a function with high values corresponding to the features of interest; thereby snakes are attracted to salient features. There are various ways to define these image forces. The simplest useful image functional is the image intensity $I(x, y)$ itself. If we set

$$E_{line} = \int_0^1 \pm I(x, y)ds\,,$$

then depending on the sign, the snake will be attracted either to light lines or dark lines. Finding edges in an image can be done using

$$E_{edge} = \int_0^1 -|\nabla I(x, y)|^2\, ds\,.$$

The snake is attracted to contours with large image gradients.

Minimization of the Snake's Energy In the absence of external forces the substitution of (A.2) and (A.3) into the equation of the snake's total energy (A.1) results in

$$E(\boldsymbol{v}) = -\int_0^1 P(\boldsymbol{v}(s))ds + \frac{1}{2}\int_0^1 \alpha(s)\left|\frac{\partial \boldsymbol{v}(s)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 \boldsymbol{v}(s)}{\partial s^2}\right|^2 ds\,.$$

Assuming $\alpha(s)$ and $\beta(s)$ to be constants α and β, not varying along the snake, minimizing this energy functional gives rise to the following two independent Euler equations

$$-\alpha\frac{\partial^2 x}{\partial s^2} + \beta\frac{\partial^4 x}{\partial s^4} = -\frac{\partial P}{\partial x}\,, \tag{A.4}$$

$$-\alpha\frac{\partial^2 y}{\partial s^2} + \beta\frac{\partial^4 y}{\partial s^4} = -\frac{\partial P}{\partial y}\,. \tag{A.5}$$

See Berger (1991) or Neuenschwander (1995) for snakes with computations with variable $\alpha(s)$, $\beta(s)$.

A.2 Discrete Representation

Unlike our finite element implementation in Chapter 4, most implementations of snakes make use of finite differences in space. By sampling the snake at regular

intervals into a polygonal curve with n vertices, a discretization of the snake's representation is achieved. This representation is used to approximate the derivatives used in (A.4) and (A.5) by means of central differences

$$\begin{aligned}\frac{\partial^2 x_i}{\partial s^2} &= \frac{1}{h^2}\left(x_{i-1} - 2x_i + x_{i+1}\right) ,\\ \frac{\partial^4 x_i}{\partial s^4} &= \frac{1}{h^4}\left(x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} + x_{i+2}\right) .\end{aligned}$$

When assuming equidistant points, for each snake point (x_i, y_i) the following equation must hold:

$$-\alpha\left(x_{i-1} - 2x_i + x_{i+1}\right) + \beta\left(x_{i-2} - 4x_{i-1} + 6x_i - 4x_{i+1} + x_{i+2}\right) = -f_x^i$$

with $f_x^i = \frac{\partial P}{\partial x_i}$. Using this approximation, we can write the Euler equations in matrix form as

$$\begin{aligned}\boldsymbol{A}\boldsymbol{x} &= \boldsymbol{f}_x(\boldsymbol{x}, \boldsymbol{y}) ,\\ \boldsymbol{A}\boldsymbol{y} &= \boldsymbol{f}_y(\boldsymbol{x}, \boldsymbol{y}) .\end{aligned} \tag{A.6}$$

If we deal with a closed snake (which has the advantage that central differences can be evaluated everywhere, since the snake has neither head nor tail), matrix $\boldsymbol{A}$ is of form

$$\begin{bmatrix}
2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \cdots & \cdots & 0 & \beta & -\alpha-4\beta \\
-\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \cdots & \cdots & 0 & \beta \\
\beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \cdots & \cdots & 0 \\
0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \cdots & 0 \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \\
0 & \cdots & \cdots & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta \\
\beta & 0 & \cdots & \cdots & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta \\
-\alpha-4\beta & \beta & 0 & \cdots & \cdots & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta
\end{bmatrix}$$

Since in practice $P(\boldsymbol{v})$ is a discrete function, but also since $\boldsymbol{A}$ is singular and can not be inverted, equations (A.6) can not be solved for $\boldsymbol{x}$ and $\boldsymbol{y}$ directly. Kass *et al.* (1987) use a friction force in order to constrain the displacement of the snake. This is made by setting

$$\begin{aligned}\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{f}_x(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) &= -\gamma(\boldsymbol{x}_t - \boldsymbol{x}_{t-1}) ,\\ \boldsymbol{A}\boldsymbol{y}_t - \boldsymbol{f}_y(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) &= -\gamma(\boldsymbol{y}_t - \boldsymbol{y}_{t-1}) ,\end{aligned}$$

where γ is the iteration step (or a viscosity factor). As a result, the discrete equation of the snake's motion can be written in the form

$$\begin{aligned}\boldsymbol{x}_t &= (\boldsymbol{A} + \gamma\boldsymbol{1})^{-1}\left(\gamma\boldsymbol{x}_{t-1} - \boldsymbol{f}_x(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})\right) ,\\ \boldsymbol{y}_t &= (\boldsymbol{A} + \gamma\boldsymbol{1})^{-1}\left(\gamma\boldsymbol{y}_{t-1} - \boldsymbol{f}_x(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})\right) ,\end{aligned} \tag{A.7}$$

where $\boldsymbol{1}$ is the $n \times n$ identity matrix. The solution is found iteratively; in each time step, the image forces of the last position are used. Since $\boldsymbol{A} + \gamma\boldsymbol{1}$ is constant over time, its inverse can be computed at the beginning (e.g. by a LU-decomposition or Cholesky factorization).

A.3 Example

Figure A.1 gives an example of shape recognition by means of snakes. The application that computed the illustrated result was a side-product of our main application only; no investigations were done regarding parameter setup.

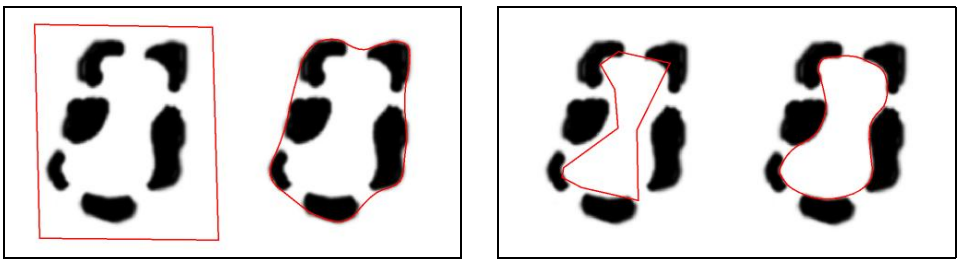


Figure A.1: Subjective contour illusion detected by snakes. The images to the left in the boxes show the initialization of the snakes; the images to the right the resulting snakes in equilibrium.

A prerequisite for the successful application of snakes is their initialization close to the object to be interpreted[1]. Snakes have the tendency to shrink, since the inner energy is minimized (vanishing) when the active contour is reduced to a single point. We took advantage of the characteristics in Figure A.1 (left): since a flat potential causes weak image forces only, the snake shrinks to a size where it meets the edge of the objects. As such, shrinkage is welcome, since it makes the snake move when no image forces exist. On the other hand it causes contours to be rather convex.

γ serves as parameter for the magnitude of oscillation. The higher γ, the slower the evolution and the higher the inertia of the snake. If a large value of γ is chosen, then $\boldsymbol{A} + \gamma\mathbf{1} \approx \gamma\mathbf{1}$, which leads, when inserted in (A.7), to

$$\gamma \boldsymbol{x}_{t-1} - \frac{\partial f}{\partial x}(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}) \approx \gamma \boldsymbol{x}_{t-1} \,,$$

from what follows

$$\boldsymbol{x}_t \approx \boldsymbol{x}_{t-1}$$

which proves the contour to be quasi stationary (the same considerations are valid with respect to the y-direction). With small values for γ the snake has a low inertia, resulting in strong oscillation and fast movements. Note that the matrix $\boldsymbol{A}$ is *a priori* singular and can not be inverted. With low values for γ the conditioning $cond(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\|$ is still bad. Choosing higher values for γ improves conditioning.

[1]Note that the initialization of snakes for cartographic displacement is not an issue. The original state of the lines give a good approximation of the final solution, since we can assume displacements to be small.

Appendix B

Assembling Snakes and Beams for Junction Peculiarities

B.1 Background

Motivation The symbolization of roads at smaller scales may lead to symbol overlaps between adjacent roads. In junctions, the enlarged symbol width may cover the structure and organization of the incident roads (Section 4.5). The organization of junctions, however, is important information and has to be kept legible.

Geometric Correction of Lines With increasing symbol width, the required geometric corrections of junctions, resp. of their incident roads, gets bigger. We are better not to think of these as local 'deformations' anymore, but as 'geometric modifications', since some parts have to be strongly altered. It has been pointed out in Section 4.5 that such local modifications can only be computed on the basis of a preceeding local analysis of the 'strokes'. This analysis results in a modification of the lines incident to the junction. These corrections are applied to the initial line geometry. The lines, originally ending in a common junction node, are *disconnected*.

Goal We show in this Appendix how to set up a system of snakes or beams, *such that the inner energy (shape distortion) of the lines is minimized subject to the constraint of granted connectivity.* The minimization of the overall inner energy implies that we can not make any assumption about the final junction location, since it would predetermine a later solution. *In the following, the computation is illustrated for snakes, and for the x-direction only.* Note, however, that the methodology to achieve this goal is the same for snakes and beams.

B.2 Description of Example

Notation We explain the three-line situation displayed in Figure B.1. The left-hand subscripts of vertices indicate the line numbers, the right-hand subscripts the vertex numbers within such lines. The topology requires that $P_{13} = P_{21} = P_{33}$, which is assumed to be true before the junction is enhanced for large symbols.

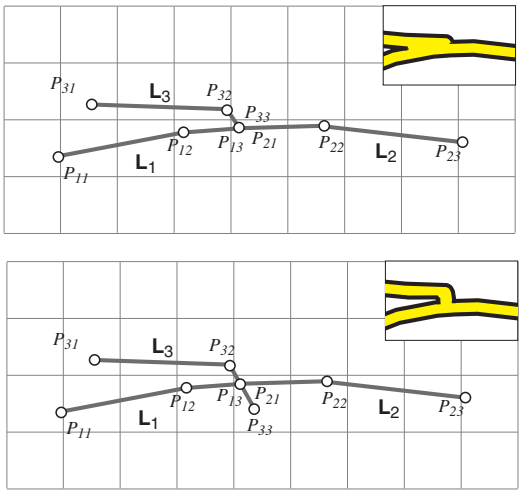


Figure B.1: Junction Example. Initial situation (top); Enlargement of segments $\overline{P_{32}P_{33}}$ destroys topology (bottom), but if reconnected, it clears the junction from symbol overlap.

Adaption of Junctions to Symbology To clear the junction for a wide road symbol, its geometry must be modified. By a local analysis of the junction, we note that a modification of lines $\mathcal{L}_1$ and $\mathcal{L}_2$ is not required. To clear the junction, while preserving the stroke built by $\mathcal{L}_1$ and $\mathcal{L}_2$, line $\mathcal{L}_3$ has to be enlarged at its end. Therefore, P_{33} is displaced by $\boldsymbol{d} = (0.4, 0.4)$[1]. Due to this geometric adaption, the topologic condition $P_{13} = P_{21} = P_{31}$ is now no longer met; the lines are disconnected.

[1] Alternatively, we could also add a new point P_{34} to the line. Such treatment would have the advantage that the direction of the incident segment could also be changed, which is not possible by a simple enlargement of existing segments. The enlargement procedure has been chosen here to keep the number of points low, which reduces paperwork later.

B.3 Equation Setup

The following paragraphs give step-by-step instructions on how local stiffness matrices must be assembled in the finite element method procedure such that lines are guaranteed to be reconnected later. As such it replaces only the 4^{th} step in the finite element procedure established in 3.3.4.

Single Line Setup For each line, its associated stiffness matrix and its associated force vector is computed. The properties for *one general (finite) element* were established in Section 4.2.1, equation (4.11), for snakes, and in Section 5.3, equation (5.17), for beams. Remember that, for snakes, each segment enters the stiffness matrix as 4×4 matrix. The assembly is shown for line $\mathcal{L}_1$, consisting of two segments only, with $\alpha = 5.0$ and $\beta = 0.05$ (all matrix entries are rounded to 1 digit after the decimal point)

$$\overbrace{\begin{bmatrix} 2.5 & 0.6 & -2.5 & 0.6 \\ 0.6 & 12.8 & -0.6 & -0.4 \\ -2.5 & -0.6 & 2.5 & -0.6 \\ 0.6 & -0.4 & -0.6 & 12.8 \end{bmatrix}}^{h(\overline{P_{11}P_{12}})=2.4} \quad \overbrace{\begin{bmatrix} 6.6 & 0.8 & -6.6 & 0.8 \\ 0.8 & 6.6 & -0.8 & -0.1 \\ -6.6 & -0.8 & 6.6 & -0.8 \\ 0.8 & -0.1 & -0.8 & 6.6 \end{bmatrix}}^{h(\overline{P_{13}P_{14}})=1.0}$$

$$\searrow \qquad \swarrow$$

$$\begin{bmatrix} 2.5 & 0.6 & -2.5 & 0.6 & & \\ 0.6 & 12.8 & -0.6 & -0.4 & & \\ -2.5 & -0.6 & 9.1 & 0.2 & -6.6 & 0.8 \\ 0.6 & -0.4 & 0.2 & 19.4 & -0.8 & -0.1 \\ & & -6.6 & -0.8 & 6.6 & 0.8 \\ & & 0.8 & -0.1 & -0.8 & 6.6 \end{bmatrix}$$

Taking the segment length into account, namely $h(\overline{P_{21}P_{22}}) = 1.5$, $h(\overline{P_{22}P_{23}}) = 2.8$ for $\mathcal{L}_2$ and $h(\overline{P_{31}P_{32}}) = 2.7$, $h(\overline{P_{32}P_{33}}) = 0.9$ for $\mathcal{L}_3$, the snake equation systems become

$$\mathcal{L}_1: \quad \begin{bmatrix} 2.5 & 0.6 & -2.5 & 0.6 & & \\ 0.6 & 12.8 & -0.6 & -0.4 & & \\ -2.5 & -0.6 & 9.1 & 0.2 & -6.6 & 0.8 \\ 0.6 & -0.4 & 0.2 & 19.4 & -0.8 & -0.1 \\ & & -6.6 & -0.8 & 6.6 & 0.8 \\ & & 0.8 & -0.1 & -0.8 & 6.6 \end{bmatrix} \begin{bmatrix} d_{11} \\ d'_{11} \\ d_{12} \\ d'_{12} \\ d_{13} \\ d'_{13} \end{bmatrix} = \begin{bmatrix} f_{11} \\ f'_{11} \\ f_{12} \\ f'_{12} \\ f_{13} \\ f'_{13} \end{bmatrix}$$

$$\mathcal{L}_2: \quad \begin{bmatrix} 4.2 & 0.6 & -4.2 & 0.6 & & \\ 0.6 & 8.6 & -0.6 & -0.2 & & \\ -4.2 & -0.6 & 6.3 & -0.1 & -2.2 & 0.5 \\ 0.6 & -0.2 & -0.1 & 23.3 & -0.5 & -0.4 \\ & & -2.2 & -0.5 & 2.2 & -0.5 \\ & & 0.5 & -0.4 & -0.5 & 14.7 \end{bmatrix} \begin{bmatrix} d_{21} \\ d'_{21} \\ d_{22} \\ d'_{22} \\ d_{23} \\ d'_{23} \end{bmatrix} = \begin{bmatrix} f_{21} \\ f'_{21} \\ f_{22} \\ f'_{22} \\ f_{23} \\ f'_{23} \end{bmatrix}$$

$$\mathcal{L}_3: \quad \begin{bmatrix} 2.3 & 0.5 & -2.3 & 0.5 & & \\ 0.5 & 14.2 & -0.5 & -0.4 & & \\ -2.3 & -0.5 & 9.7 & 0.3 & -7.5 & 0.9 \\ 0.5 & -0.4 & 0.3 & 20.5 & -0.9 & -0.1 \\ & & -7.5 & -0.9 & 7.5 & -0.9 \\ & & 0.9 & -0.1 & -0.9 & 6.3 \end{bmatrix} \begin{bmatrix} d_{31} \\ d'_{31} \\ d_{32} \\ d'_{32} \\ d_{33} \\ d'_{33} \end{bmatrix} = \begin{bmatrix} f_{31} \\ f'_{31} \\ f_{32} \\ f'_{32} \\ f_{33} \\ f'_{33} \end{bmatrix}$$

Translate Implicit Relations Topological knowledge is included by expressing the displacements with respect to a virtual junction node J. If all lines, initially ending in J, should be connected later, the vertex displacements must hold

$$d_J \;:=\; d_{13} \;=\; d_{21} \;=\; d_{33} + 0.4\,.$$

We also merge the entries for

$$d'_J \;:=\; d'_{13} \;=\; d'_{21} \;=\; d'_{33}\,.$$

These values will be set later to zero anyway in order to stiffen the road directions in the junction. Including these conditions into the algebraic equations of $\mathcal{L}_1$ and $\mathcal{L}_2$ means renaming terms only

$$\mathcal{L}_1\text{:}\quad \begin{bmatrix} 2.5 & 0.6 & -2.5 & 0.6 & & \\ 0.6 & 12.8 & -0.6 & -0.4 & & \\ -2.5 & -0.6 & 9.1 & 0.2 & -6.6 & 0.8 \\ 0.6 & -0.4 & 0.2 & 19.4 & -0.8 & -0.1 \\ & & -6.6 & -0.8 & 6.6 & 0.8 \\ & & 0.8 & -0.1 & -0.8 & 6.6 \end{bmatrix} \begin{bmatrix} d_{11} \\ d'_{11} \\ d_{12} \\ d'_{12} \\ \boldsymbol{d_J} \\ \boldsymbol{d'_J} \end{bmatrix} = \begin{bmatrix} f_{11} \\ f'_{11} \\ f_{12} \\ f'_{12} \\ f_{13} \\ f'_{13} \end{bmatrix}$$

$$\mathcal{L}_2\text{:}\quad \begin{bmatrix} 4.2 & 0.6 & -4.2 & 0.6 & & \\ 0.6 & 8.6 & -0.6 & -0.2 & & \\ -4.2 & -0.6 & 6.3 & -0.1 & -2.2 & 0.5 \\ 0.6 & -0.2 & -0.1 & 23.3 & -0.5 & -0.4 \\ & & -2.2 & -0.5 & 2.2 & -0.5 \\ & & 0.5 & -0.4 & -0.5 & 14.7 \end{bmatrix} \begin{bmatrix} \boldsymbol{d_J} \\ \boldsymbol{d'_J} \\ d_{22} \\ d'_{22} \\ d_{23} \\ d'_{23} \end{bmatrix} = \begin{bmatrix} f_{21} \\ f'_{21} \\ f_{22} \\ f'_{22} \\ f_{23} \\ f'_{23} \end{bmatrix}$$

where the terms in bold indicate modified entries. For line $\mathcal{L}_3$, the same conditioning requires few matrix manipulations, namely from

$$\begin{bmatrix} 2.3 & 0.5 & -2.3 & 0.5 & & \\ 0.5 & 14.2 & -0.5 & -0.4 & & \\ -2.3 & -0.5 & 9.7 & 0.3 & -7.5 & 0.9 \\ 0.5 & -0.4 & 0.3 & 20.5 & -0.9 & -0.1 \\ & & -7.5 & -0.9 & 7.5 & -0.9 \\ & & 0.9 & -0.1 & -0.9 & 6.3 \end{bmatrix} \begin{bmatrix} d_{31} \\ d'_{31} \\ d_{32} \\ d'_{32} \\ \boldsymbol{d_J - 0.4} \\ \boldsymbol{d'_J} \end{bmatrix} = \begin{bmatrix} f_{31} \\ f'_{31} \\ f_{32} \\ f'_{32} \\ f_{33} \\ f'_{33} \end{bmatrix}$$

we gain

$$\begin{bmatrix} 2.3 & 0.5 & -2.3 & 0.5 & & \\ 0.5 & 14.2 & -0.5 & -0.4 & & \\ -2.3 & -0.5 & 9.7 & 0.3 & -7.5 & 0.9 \\ 0.5 & -0.4 & 0.3 & 20.5 & -0.9 & -0.1 \\ & & -7.5 & -0.9 & 7.5 & -0.9 \\ & & 0.9 & -0.1 & -0.9 & 6.3 \end{bmatrix} \begin{bmatrix} d_{31} \\ d'_{31} \\ d_{32} \\ d'_{32} \\ \boldsymbol{d_J} \\ \boldsymbol{d'_J} \end{bmatrix} = \begin{bmatrix} f_{31} \\ f'_{31} \\ \boldsymbol{f_{32} - 7.5 \cdot 0.4} \\ \boldsymbol{f'_{32} - 0.9 \cdot 0.4} \\ \boldsymbol{f_{33} + 7.5 \cdot 0.4} \\ \boldsymbol{f'_{33} - 0.9 \cdot 0.4} \end{bmatrix}$$

Global Assemblage - Merge Similar Entries The algebraic equations describing the characteristics of the 3 lines are now combined in the global (overall) system equation. For nodes, where two or more lines meet (which here is only the case for a junction) the nodal stiffnesses and nodal loads are added. The common junction entry has been placed at the end.

Expanding first $\mathcal{L}_1$

$$\begin{bmatrix}
2.5 & 0.6 & -2.5 & 0.6 & . & . & . & . & . & . & . & . & . & . \\
0.6 & 12.8 & -0.6 & -0.4 & . & . & . & . & . & . & . & . & . & . \\
-2.5 & -0.6 & 9.1 & 0.2 & . & . & . & . & . & . & . & . & -6.6 & 0.8 \\
0.6 & -0.4 & 0.2 & 19.4 & . & . & . & . & . & . & . & . & -0.8 & -0.1 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & -6.6 & -0.8 & . & . & . & . & . & . & . & . & 6.6 & 0.8 \\
. & . & 0.8 & -0.1 & . & . & . & . & . & . & . & . & -0.8 & 6.6
\end{bmatrix}
\begin{bmatrix}
d_{11} \\ d'_{11} \\ d_{12} \\ d'_{12} \\ d_{22} \\ d'_{22} \\ d_{23} \\ d'_{23} \\ d_{31} \\ d'_{31} \\ d_{32} \\ d'_{32} \\ \boldsymbol{d_J} \\ \boldsymbol{d'_J}
\end{bmatrix}$$

now adding the values of $\mathcal{L}_2$

$$\begin{bmatrix}
2.5 & 0.6 & -2.5 & 0.6 & . & . & . & . & . & . & . & . & . & . \\
0.6 & 12.8 & -0.6 & -0.4 & . & . & . & . & . & . & . & . & . & . \\
-2.5 & -0.6 & 9.1 & 0.2 & . & . & . & . & . & . & . & . & -6.6 & 0.8 \\
0.6 & -0.4 & 0.2 & 19.4 & . & . & . & . & . & . & . & . & -0.8 & -0.1 \\
. & . & . & . & 6.3 & -0.1 & -2.2 & 0.5 & . & . & . & . & -4.2 & -0.6 \\
. & . & . & . & -0.1 & 23.3 & -0.5 & -0.4 & . & . & . & . & 0.6 & -0.2 \\
. & . & . & . & -2.2 & -0.5 & 2.2 & -0.5 & . & . & . & . & . & \\
. & . & . & . & 0.5 & -0.4 & -0.5 & 14.7 & . & . & . & . & . & \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & -6.6 & -0.8 & -4.2 & 0.6 & . & . & . & . & . & . & 10.8 & 1.4 \\
. & . & 0.8 & -0.1 & -0.6 & -0.2 & . & . & . & . & . & . & -0.2 & 15.2
\end{bmatrix}
\begin{bmatrix}
d_{11} \\ d'_{11} \\ d_{12} \\ d'_{12} \\ d_{22} \\ d'_{22} \\ d_{23} \\ d'_{23} \\ d_{31} \\ d'_{31} \\ d_{32} \\ d'_{32} \\ \boldsymbol{d_J} \\ \boldsymbol{d'_J}
\end{bmatrix}$$

and finally the entries of $\mathcal{L}_3$

$$\begin{bmatrix}
2.5 & 0.6 & -2.5 & 0.6 & . & . & . & . & . & . & . & . & . & . \\
0.6 & 12.8 & -0.6 & -0.4 & . & . & . & . & . & . & . & . & . & . \\
-2.5 & -0.6 & 9.1 & 0.2 & . & . & . & . & . & . & . & . & -6.6 & 0.8 \\
0.6 & -0.4 & 0.2 & 19.4 & . & . & . & . & . & . & . & . & -0.8 & -0.1 \\
. & . & . & . & 6.3 & -0.1 & -2.2 & 0.5 & . & . & . & . & -4.2 & -0.6 \\
. & . & . & . & -0.1 & 23.3 & -0.5 & -0.4 & . & . & . & . & 0.6 & -0.2 \\
. & . & . & . & -2.2 & -0.5 & 2.2 & -0.5 & . & . & . & . & . & \\
. & . & . & . & 0.5 & -0.4 & -0.5 & 14.7 & . & . & . & . & . & \\
. & . & . & . & . & . & . & . & 2.3 & 0.5 & -2.3 & 0.5 & . & . \\
. & . & . & . & . & . & . & . & 0.5 & 14.2 & -0.5 & -0.4 & . & . \\
. & . & . & . & . & . & . & . & -2.3 & -0.5 & 9.7 & 0.3 & -7.5 & 0.9 \\
. & . & . & . & . & . & . & . & 0.5 & -0.4 & 0.3 & 20.5 & -0.9 & -0.1 \\
. & . & -6.6 & -0.8 & -4.2 & 0.6 & . & . & . & . & -7.5 & -0.9 & 18.3 & 0.5 \\
. & . & 0.8 & -0.1 & -0.6 & -0.2 & . & . & . & . & 0.9 & -0.1 & -1.1 & 21.5
\end{bmatrix}
\begin{bmatrix}
d_{11} \\ d'_{11} \\ d_{12} \\ d'_{12} \\ d_{22} \\ d'_{22} \\ d_{23} \\ d'_{23} \\ d_{31} \\ d'_{31} \\ d_{32} \\ d'_{32} \\ \boldsymbol{d_J} \\ \boldsymbol{d'_J}
\end{bmatrix}$$

leads to the demanded stiffness matrix. The force vector is assembled the same way

$$\begin{bmatrix}
f_{11} \\ f'_{11} \\ f_{12} \\ f'_{12} \\ f_{22} \\ f'_{22} \\ f_{23} \\ f'_{23} \\ f_{31} \\ f'_{31} \\ f_{32} - 3.0 \\ f'_{32} - 0.36 \\ f_J + 3.0 \\ f'_J - 0.36
\end{bmatrix}$$

with $f_J = f_{13} + f_{21} + f_{33}$ and $f'_J = f'_{13} + f'_{21} + f'_{33}$.

Impose Boundary Conditions To finish the example, boundary conditions are imposed and the equation system is solved. The roads are fixed at their end-nodes

$$\begin{aligned} d_{11} &= 0\,, & d_{23} &= 0\,, & d_{31} &= 0\,, \\ d'_{11} &= 0\,, & d'_{23} &= 0\,, & d'_{31} &= 0\,, \end{aligned}$$

and the geometric structure of the junction is protected

$$d'_J = 0\,.$$

For sake of simplicity, it is assumed that no forces act on the road network

$$f_{11} \;=\; f'11 \;=\; \ldots \;=\; f'_{33} \;=\; f_J \;=\; f'_J \;=\; 0\,.$$

Various approaches exist for how boundary conditions can be integrated into the equation system. One valid manipulation leads to

$$\begin{bmatrix}
1 & 0 & 0 & 0 & . & . & . & . & . & . & . & . & . & . \\
0 & 1 & 0 & 0 & . & . & . & . & . & . & . & . & . & . \\
0 & 0 & 9.1 & 0.2 & . & . & . & . & . & . & . & . & -6.6 & 0 \\
0 & 0 & 0.2 & 19.4 & . & . & . & . & . & . & . & . & -0.8 & 0 \\
. & . & . & . & 6.3 & -0.1 & 0 & 0 & . & . & . & . & -4.2 & 0 \\
. & . & . & . & -0.1 & 23.3 & 0 & 0 & . & . & . & . & 0.6 & 0 \\
. & . & . & . & 0 & 0 & 1 & 0 & . & . & . & . & . & . \\
. & . & . & . & 0 & 0 & 0 & 1 & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & 1 & 0 & 0 & 0 & . & . \\
. & . & . & . & . & . & . & . & 0 & 1 & 0 & 0 & . & . \\
. & . & . & . & . & . & . & . & 0 & 0 & 9.7 & 0.3 & -7.5 & 0 \\
. & . & . & . & . & . & . & . & 0 & 0 & 0.3 & 20.5 & -0.9 & 0 \\
. & . & -6.6 & -0.8 & -4.2 & 0.6 & . & . & . & . & -7.5 & -0.9 & 18.3 & 0 \\
. & . & 0 & 0 & 0 & 0 & . & . & . & . & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
d_{11} \\ d'_{11} \\ d_{12} \\ d'_{12} \\ d_{22} \\ d'_{22} \\ d_{23} \\ d'_{23} \\ d_{31} \\ d'_{31} \\ d_{32} \\ d'_{32} \\ d_J \\ d'_J
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -3.0 \\ -0.36 \\ 3.0 \\ 0
\end{bmatrix}$$

This linear equation system is solved for the unknown displacements. This gives rise to $d_J = 0.14$, from which we conclude $d_{13} = d_{21} = 0.14$ and $d_{33} = -0.26$. Combined with the other displacements, we have

$$\begin{aligned}
\mathcal{L}_1 : d_{11} &= \;\; 0\,, & d_{12} &= \;\; 0.10\,, & d_{13} &= \;\; 0.14\,, \\
\mathcal{L}_2 : d_{21} &= 0.14\,, & d_{22} &= \;\; 0.09\,, & d_{23} &= \;\; 0\,, \\
\mathcal{L}_3 : d_{31} &= \;\; 0\,, & d_{32} &= -0.20\,, & d_{33} &= -0.26\,,
\end{aligned}$$

To compute the displacements along the y-axis, the same stiffness matrix can be reused, since its entries depend on segment lengths only. Different junction relations, and therefore different force entries, lead to different displacements.

Shifting the line vertices by the required displacements, finally, reconnects them in the junction (Figure B.2).

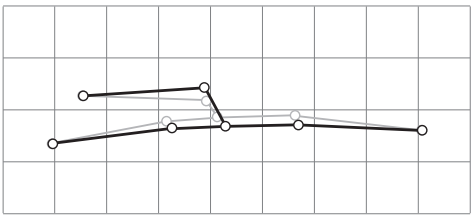


Figure B.2: Topology is restored.

The procedure established in this Appendix is costly and cumbersome. It was used for didactic reasons in the hope of making the underlying ideas more visible. An implementation of the underlying ideas omits many demonstrated steps, since the element equations can be directly added to the global system equation; only a mapping of the line vertices on the entries of the global matrix is required.

Appendix C

Plane Stress in Elasticity by Means of Finite Elements

Principles from Mechanics Consider a body composed of a homogeneous, isotropic, linear elastic material. By homogeneous we mean that any element volume of the body possesses the same specific physical properties as any other element volume of the body; by isotropic we mean that the physical properties are the same in all directions (Huebner *et al.* 1995). The terminology of strain and stress has been introduced in Section 5.2. The components of stress are again denoted by σ_x, σ_y and τ_{xy}. The strain-displacement equations of the theory of elasticity relate the displacement u and v in the x- and y-coordinate directions to the corresponding strains

$$\epsilon_x = \frac{\partial u}{\partial x}\,, \qquad \epsilon_y = \frac{\partial u}{\partial x}\,, \qquad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\,. \tag{C.1}$$

A body that is subject to only forces in the plane of the map is said to be in *plane stress*. In plane stress, Hooke's law, relating strain and stress, is given for an infinitesimally small element as

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \frac{E}{1-\nu} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{C.2}$$

or

$$\boldsymbol{\sigma} = \boldsymbol{C}\boldsymbol{\epsilon}$$

with the engineering constants E (modulus of elasticity) and ν (Poisson's ratio).

Minimum Potential Energy Principle We now shift our attention to integral statements that hold throughout the entire body. The theorem of minimum potential energy states that "the displacement (u, v) which satisfies the differential equations of equilibrium, as well as the conditions at the bounding surface, yield smaller values for the potential energy than any other displacement which satisfies the same conditions at the bounding surface" (Huebner *et al.* 1995, p.570).

If $E(u, v)$ is the potential energy, U_P is the strain energy, and $V_P(u, v)$ is the work done by the applied load during displacement changes, then, according to the

minimum principle, we have at equilibrium

$$\begin{aligned} \partial E(u,v) &= \partial\left[U_P(u,v) - V_P(u,v)\right] \\ &= \partial U_P(u,v) - \partial V_P(u,v) \;=\; 0\,. \end{aligned}$$

The strain energy of a linear elastic body, see also Section 5.2.3, is defined as

$$U_P = \frac{1}{2}\int_\Omega \boldsymbol{\epsilon}^T \boldsymbol{\sigma} dV\,, \tag{C.3}$$

where Ω denotes the volume of the body. With Hooke's law (C.2), we have

$$U_P = \frac{1}{2}\int_\Omega \boldsymbol{\epsilon}^T \boldsymbol{C}\boldsymbol{\epsilon} dV\,.$$

Finite Element Method: Discretization and Interpolation Function The finite element method requires first a discretization of the continuum. Easiest, even though not necessarily required, is a triangulation, which uses the triangle as the basic geometric element. The behavior of a triangle is defined over its displacement at each node, which we denote by $\boldsymbol{d} = (u_1, v_1, u_2, v_2, u_3, v_3)^T$. The displacement function within the element clearly must be a function of x and y, and it must be *uniquely* determined by the six nodal displacements (Huebner *et al.* 1995). With this constraint a linear displacement function is the only choice, that is

$$u(x,y) = \frac{a_1u_1 + a_2u_2 + a_3u_3}{2\Delta} + \frac{b_1u_1 + b_2u_2 + b_3u_3}{2\Delta}x + \frac{c_1u_1 + c_2u_2 + c_3u_3}{2\Delta}y \tag{C.4}$$

$$v(x,y) = \frac{a_1v_1 + a_2v_2 + a_3v_3}{2\Delta} + \frac{b_1v_1 + b_2v_2 + b_3v_3}{2\Delta}x + \frac{c_1v_1 + c_2v_2 + c_3v_3}{2\Delta}y \tag{C.5}$$

with

$$\begin{aligned} a_1 &= x_2y_3 - x_3y_2\,, & a_2 &= x_3y_1 - x_1y_3\,, & a_3 &= x_1y_2 - x_2y_1\,, \\ b_1 &= y_2 - y_3\,, & b_2 &= y_3 - y_1\,, & b_3 &= y_1 - y_2\,, \\ c_1 &= x_3 - x_2\,, & c_2 &= x_1 - x_3\,, & c_3 &= x_2 - x_1\,, \end{aligned}$$

and Δ the area of the triangular element.[1]

[1] Even though looking dangerous, these interpolation functions are gained with few algebraic manipulations from the assumption that the element's value varies linearly over its extent. Then, the functions u and v are of type

$$\begin{aligned} u &= \alpha_1 + \alpha_2 x + \alpha_3 y \\ v &= \alpha_4 + \alpha_5 x + \alpha_6 y \end{aligned} \tag{C.6}$$

It is now possible to find the constants α_1 through α_6 in terms of the coordinates of the node points and the nodal displacements, which require

$$\begin{aligned} &\text{for node 1:} && u_1 = \alpha_1 + \alpha_2 x_1 + \alpha_3 y_1 && v_1 = \alpha_4 + \alpha_5 x_1 + \alpha_6 y_1 \\ &\text{for node 2:} && u_2 = \alpha_1 + \alpha_2 x_2 + \alpha_3 y_2 && v_2 = \alpha_4 + \alpha_5 x_2 + \alpha_6 y_2 \\ &\text{for node 3:} && u_2 = \alpha_1 + \alpha_2 x_3 + \alpha_3 y_3 && v_2 = \alpha_4 + \alpha_5 x_3 + \alpha_6 y_3 \end{aligned} \tag{C.7}$$

If we now solve (C.7) for the unknowns $\alpha_1, \ldots, \alpha_6$, and substitute these values in (C.6) we have the quoted interpolation functions u and v over a triangular element.

Finite Element Method: Element Properties To gain the element properties of a general triangle, an expression for the element strain is derived. By substituting the derivatives of u and v from (C.4) and (C.5) into (C.1), we have

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} b_1 & 0 & b_2 & 0 & b_3 & 0 \\ 0 & c_1 & 0 & c_2 & 0 & c_3 \\ c_1 & b_1 & c_2 & b_2 & c_3 & b_3 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} \tag{C.8}$$

which we express by

$$\boldsymbol{\epsilon} = \boldsymbol{B}\boldsymbol{d}\,.$$

To find the element equilibrium equations, we use the principle of minimum potential energy. According to (C.3), the strain energy is

$$U_P \;=\; \frac{1}{2}\int_\Omega \boldsymbol{\epsilon}^T \boldsymbol{C}\boldsymbol{\epsilon}\, dV \;=\; \frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{C}\boldsymbol{\epsilon}\int_\Omega dV \;=\; \frac{1}{2}\Delta\, \boldsymbol{\epsilon}^T\boldsymbol{C}\boldsymbol{\epsilon} \tag{C.9}$$

where the second equality is valid since the strains are constant within an element, and where the last equality is true when we assume the triangle to have a thickness of 1; Δ denotes the area of the triangle. Substituting (C.8) into (C.9), we may express the element strain energy as

$$U_P = \frac{1}{2}\Delta \boldsymbol{d}^T\boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}\boldsymbol{d}\,. \tag{C.10}$$

The work of the applied nodal forces is

$$V_P = f_{1x}u_1 + f_{1y}v_1 + \ldots + f_{3x}u_3 + f_{3y}v_3$$

or in matrix notation

$$V_P = \boldsymbol{d}^T\boldsymbol{f}\,. \tag{C.11}$$

Combining (C.10) and (C.11), we obtain the element's potential energy

$$E = \frac{1}{2}\Delta \boldsymbol{d}^T\boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}\boldsymbol{d} - \boldsymbol{d}^T\boldsymbol{f}\,. \tag{C.12}$$

For equilibrium (minimum principle), we require

$$\frac{\partial E}{\partial u_1} \;=\; \frac{\partial E}{\partial v_1} \;=\; \ldots \;=\; \frac{\partial E}{\partial v_3} \;=\; 0\,.$$

Since (C.12) is a quadratic function, we see that minimizing the element potential gives

$$\Delta \boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}\boldsymbol{d} - \boldsymbol{f} = \boldsymbol{0}\,. \tag{C.13}$$

By inspection we recognize that (C.13) has the standard form

$$\boldsymbol{K}\boldsymbol{d} = \boldsymbol{f}\,,$$

with

$$\boldsymbol{K} = \Delta \boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}\,.$$

Performing the matrix multiplications leads to the following expression for the element matrix:

$$K = \frac{Et}{4\Delta(1-\nu^2)} \cdot \tag{C.14}$$

$$\begin{bmatrix} b_1^2+\lambda c_1^2 & (\nu+\lambda)b_1c_1 & b_1b_2+\lambda c_1c_2 & \nu b_1c_2+\lambda b_2c_1 & b_1b_3+\lambda c_1c_3 & \nu b_1c_3+\lambda b_3c_1 \\ & c_1^2+\lambda b_1^2 & \nu b_2c_1+\lambda b_1c_2 & c_1c_2+\lambda b_1b_2 & \nu b_3c_1+\lambda b_1c_3 & c_1c_3+\lambda b_1b_3 \\ & & b_2^2+\lambda c_2^2 & (\nu+\lambda)b_2c_2 & b_2b_3+\lambda c_2c_3 & \nu b_2c_3+\lambda b_3c_2 \\ & & & c_2^2+\lambda b_2^2 & \nu b_3c_2+\lambda b_2c_3 & c_2c_3+\lambda b_2b_3 \\ & \text{symmetric} & & & b_3^2+\lambda c_3^2 & (\nu+\lambda)b_3c_3 \\ & & & & & c_3^2+\lambda b_3^2 \end{bmatrix}$$

where $\lambda = (1-\nu)/2$.

Finite Element Method: Assembling the Elements The next step in the finite element analysis of the system is to combine all these equations to form a complete set governing the composite of elements. The system assembly procedure is based on our insistence of compatibility at element nodes. By this we mean that at nodes where elements are connected the values of the displacement are the same for all triangles connected at that node. The assemblage poses no problems here, since the local element matrices are already formulated with respect to the global coordinate system.

Bibliography

Amini, A. A., Tehrani, S., and Weymouth, T. E. (1988). Using Dynamic Programming for Minimizing the Energy of Active Contours in the Presence of Hard Constraints. In: *Proceedings of the Second International Conference on Computer Vision*, pp. 95–99. IEEE Computer Soc. Press, Tampa, Florida. 56

Amini, A. A., Weymouth, T. E., and Jain, R. C. (1990). Using Dynamic Programming for Solving Variational Problems in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12** (9), pp. 855–866. 56

Atanackovic, T. M. and Guran, A. (1999). *Theory of Elasticity for Scientists and Engineers*. Birkhäuser. 91, 92

Bader, M., Barrault, M., Regnauld, N., Mustière, S., Duchêne, C., Ruas, A., Fritsch, E., Lecordix, F., and Barillot, X. (1999). AGENT Workpackage D2 - Selection of Basic Algorithms. Technical report, Department of Geography, University of Zurich. 8

Bader, M. and Weibel, R. (1997). Detecting and Resolving Size and Proximity Conflicts in the Generalization of Polygonal Maps. In: *Proceedings of the 18th ICA/ACI International Cartographic Conference*, pp. 1525–1534. Stockholm. 21, 22

Baeijs, C., Demazeau, Y., and Alvares, L. (1996). SIGMA: Application of Multi-Agent Systems to Cartographic Generalization. In: *7th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, MAAMAW'96*. Springer-Verlag, Eindhoven, the Netherlands. 9

Baierl, R. and Heinl, J. (1993). *Einführung in die Variationsrechnung*. Kompaktinformation Mathematik. Thai-Verlag, Berlin. 35

Barrault, M., Bader, M., and Weibel, R. (2000). Topology Preserving Conflict Removal between Symbolized Roads in Cartographic Generalization: Extending Snakes Methods. In: *Abstract for GIScience2000*. 71

Bedford, A. and Liechti, K. M. (2000). *Mechanics of Materials*. Prentice-Hall, Inc. 91

Berger, M. D. (1991). *Les contours actifs: modélisation, comportement et convergence*. Ph.D. thesis, Institut National Polytechnique de Lorraine, France. 57, 166

Bern, M. and Eppstein, D. (1992). Mesh Generation and Optimal Triangulation. In: D.-Z. Du and F. Hwang (editors), *Computing in Euclidean Geometry*, pp. 23–90. World Scientific. 133

Betten, J. (1997). *Finite Elemente für Ingenieure 1*. Springer, Berlin Heidelberg, Germany. 39, 99

Bobrich, J. (1996). *Ein neuer Ansatz zur kartographischen Verdrängung auf der Grundlage eines mechanischen Federmodells*. Ph.D. thesis, Deutsche Geodätische Kommission, München, Reihe C, H. 455. 23, 26

Brassel, K. (1990). Computergestüetzte Kartographie. Kartographisches Generalisieren. *Kartographische Publikationsreihe Nr.10*. Schweiz. Gesellschaft für Kartographie. 6

Brazile, F. (2000). *Semantic Infrastructure and Methods to Support Quality Evaluation in Cartographic Generalization*. Ph.D. thesis, Department of Geography, University of Zurich, Zurich, Switzerland. 19

Bronstein, I. N. and Semedjajew, K. A. (1996). *Teubner-Taschenbuch der Mathematik*. Teubner. 38, 45, 57

Buchanan, G. R. (1995). *Theory and Problems of Finite Element Analysis*. Schaum's Outline Series, McGraw-Hill Inc. 54

Burghardt, D. (2000). *Automatisierung der kartographischen Verdrängung mittels Energieminimierung*. Ph.D. thesis, Institut für Planetare Geodäsie, Technische Universität Dresden, Germany. 14, 24, 26, 27, 57, 58, 65, 126, 130, 135, 143, 157

Burghardt, D. and Meier, S. (1997). Cartographic Displacement using the Snakes Concept. In: W. Foerstner and L. Pluemer (editors), *Semantic Modeling for the Acquisition of Topografic Information from Images and Maps*. Birkhaeuser-Verlag, Basel. 24, 55, 56, 157

Christ, F. (1979). Ein Programm zur vollautomatischen Verdrängung von Punkt- und Linienobjekten bei der kartographischen Generalisierung. In: *Internationales Jahrbuch für Kartographie XIX*, pp. 41–63. 23

Ciarlet, P. G. (1991). *Introduction to Numerical Linear Algebra and Optimmisation*. Cambridge Texts in Applied Mathematics. Cambridge University Press. 37

Cohen, L. D. and Cohen, I. (1990). A Finite Element Method applied to new Acitve Contour Models and 3D Reconstruction from Cross Sections. In: *Proceedings of the Third International Conference on Computer Vision*, pp. 587–591. IEEE Computer Soc. Press. 57, 58

Cohen, L. D. and Cohen, I. (1993). Finite-Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1131–1147. 59, 62

Courant, R. and Hilbert, D. (1993). *Methoden der Mathematischen Physik*. Springer, 4th edition. 29

Cromley, R. G. (1992). *Digital Cartography*. Prentice-Hall Inc. 1

de Berg, M., van Kreveld, M., and Schirra, S. (1995). A new approach to subdivision simplification. In: *Proceedings of AUTO-CARTO 12*, volume 4, pp. 79–88. 9

de Floriani, L. and Puppo, E. (1992). An On-Line Algorithm for Constrained Delaunay Triangulation. *CVGIP: Graphical Models and Image Processing*, **54** (3), pp. 290–300. 133

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its character. *The Canadian Cartogapher*, **10** (2), pp. 112–123. 9

Durfee, E. H., Lesser, V. R., and Corkill, D. D. (1989). Trends in cooperative distributed problem solving. *IEEE Transaction on Knowledge Data Engineering*, **11** (1), pp. 63–83. 9

Dutton, G., Weibel, R., Peter, B., Bader, M., and Brazile, F. (1998). AGENT Workpackage A2 - Constraint Analysis. Technical report, Department of Geography, University of Zurich. 8

Eriksson, K., Estep, D., Hansbo, P., and Johnson, C. (1996). *Computational Differential Equations*. Cambridge University Press. 37, 54, 96

Everstine, G. C. (1979). A Comparison of Three Resequencing Algorithms for the Reduction of Matrix Profile and Wavefront. *International Journal for Numerical Methods in Engineering*, **14**, pp. 837–853. 49

Fässler, A. (1995). *Variationsrechnung mit Einführung in die Methode der finiten Elemente*. Publikationen der Ingenieurschule Biel, Biel - Bienne, Switzerland. 35

Fliessbach, T. (1996). *Mechanik*. Spektrum Akademischer Verlag, 2nd edition. 32

Fritsch, E., Lecordix, F., Duchêne, C., Barrillot, X., Mustière, S., Hangöuet, J. F., and Ruas, A. (1998). Table ronde de plusieurs jours sur les mécanismes de déplacement d'objets linéaires. Internal Report. Institut Géographique National (IGN), Paris, France. 19

Gottschalk, H.-J. (1972). Ein Modell zur automatischen Durchführung der Verdrängung bei der Generalisierung. *Nachrichten aus dem Karten- und Vermessungswesen*, **1** (58), pp. 21–26. 20

Grünreich, D. (1985). Computer-assisted generalisation. In: *Papers CERCO Cartography Course*. Frankfurt am Main, Institut für angewandte Geodäsie. 7

Hake, G. and Grünreich, D. (1994). *Kartographie*. de Gruyter, Berlin, Germany. 8

Hangouët, J. F. and Djadri, R. (1997). Voronoï Diagrams on Line Segments: Measurements for Contextual Generalization Purposes. In: *Proceedings of COSIT 97*, pp. 207–222. Laurel Highlands, Pittsburgh, Pennsylvania. 142

Harrie, L. E. (1999). The Constraint Method for Solving Spatial Conflicts in Cartographic Generalization. *Cartography and Geographic Information Science*, **26** (1), pp. 55–69. 23, 25, 26, 27

Henwood, D. and Bonet, J. (1996). *Finite Elements - A gentle introduction*. Macmillan Press Ltd. 29, 36, 49, 54

Hibbeler, R. C. (1999). *Mechanics of Materials*. Prenitce-Hall, Upper Saddle River, New Jersey, 4th edition. 91, 94, 97

Højholt, P. (1998). Solving Local and Global Space Conflicts in Map Generalization Using a Finite Element Method Adapted from Structural Mechanics. In: *Proceedings 8th International Symposium on Spatial Data Handling*, pp. 679–689. Vancouver, Canada. 131

Højholt, P. (2000). Solving Space Conflicts in Map Generalization: Using a Finite Element Method. *Cartography and Geographic Information Science*, **27** (1), pp. 65–74. 24, 26, 27, 126, 129, 131, 132, 133, 134, 135, 137, 138, 155, 158, 159

Huebner, K. H., Thornton, E. A., and Byrom, T. G. (1995). *The Finite Element Method for Engineers*. John Wiley and Sons, New York, 3rd edition. 38, 39, 50, 54, 59, 60, 177, 178

Hungerbühler, N. (1997). *Einführung in partielle Differentialgleichungen*. vdf Vorlesungsskripte, ETH Zürich. 52

Jäger, E. (1991). Investigations on Automated Feature Displacement for Small Scale Maps in Raster Format. In: *15th Conference of the ICA*, pp. 245–256. Bournemouth. 13, 23

Jones, C. B., Bundy, G. L., and Ware, J. M. (1995). Map Generalization with a Triangulated Data Structure. *Cartography and Geographic Information Systems*, **22** (4), pp. 317–331. 21, 22, 136

Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: Active Contour Models. In: *Proceedings of the First International Conference on Compute Vision.*, pp. 259–268. IEEE Computer Soc. Press. 55, 65, 165, 167

Kleiber, M. and Breitkopf, P. (1993). *Finite Element Methods in Structural Mechanics*. Series in Mechanical Engineering. Ellis Horwood, Chichester, England. 32

Lamy, S., Ruas, A., Demazeau, Y., Jackson, M., Mackaness, W., and Weibel, R. (1999). The Application of Agents in Automated Map Generalisation. In: *Proceedings of the 19th ICA/ACI Conference*, pp. 1225–1234. Ottawa. 2, 9, 73

Laptev, I. (1997). *Road Extraction Based on Line Extraction and Snakes*. Master's thesis, Department of Numerical Analysis and Computing Science (NADA) at the Royal Institute of Technology (KTH), Stockholm, Sweden. 165

Lecordix, F., Plazanet, C., and Lagrange, J.-P. (1997). A Platform for Research in Generalization: Application to Caricature. *GeoInformatica*, **1** (2), pp. 161–182. 9, 120, 122

Li, Z. and Openshaw, S. (1992). Algorithms for automated line generalization based on a natural principle of objective generalization. *Internation Journal of Geographical Information Systems*, **6** (5), pp. 373–389. 9

Lichtner, W. (1979). Computer-Assisted Processes of Cartographic Generalization in Topographic Maps. *Geo-Processing*, **1** (1), pp. 183–199. 8, 13, 14, 20

Lonergan, M. and Jones, C. B. (2001). An Iterative Displacement Method for Conflict Resolution in Map Generalization. *Algorithmica*. To appear. 23, 25, 27, 130, 155

Mackaness, W. A. (1994). An Algorithm for Conflict Identification and Feature Displacement in Automated Map Generalization. *Cartography and Geographic Information Systems*, **21** (4), pp. 219–232. 21, 22

McMaster, R. B. and Shea, K. S. (1992). *Generalization in Digital Cartography*. Association of American Geographers, Wahington D.C. 7, 8

Moulin, B. and Chaib-Draa, B. (1996). An Overview of Distibuted Artifical Intelligence. In: G. M. P. O'Hare and N. R. Jennings (editors), *Foundations of Distributed Artificial Intelligence*. John Wiley & Sons, Inc. 9

Müller, J. C., Weibel, R., Lagrange, J. P., and Salgé, F. (1995). Generalization: State of the Art and Issues. In: J. C. Müller, J. P. Lagrange, and R. Weibel (editors), *GIS and Generalization, GISDATA 1*, pp. 3–17. Taylor & Francis. 6

Mustière, S. (1998). GALBE: Adaptive Generalisation. In: *Proceedings GIS PlaNet 1*. Lisbon (Portugal). 9, 122

Mustière, S. (2001). *Apprentissage automatique pour la généralisation cartographique*. Ph.D. thesis, Université Pierre et Marie Curie (Paris 6). In press. 120, 162

Nelson, D. (editor) (1998). *The Penguin Dictionary of Mathematics*. Penguin Books. 35

Neuenschwander, W. M. (1995). *Elastic Deformable Contour and Surface Models for 2-D and 3-D Image Segmentation*. Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. 57, 61, 62, 166

Nickerson, B. G. (1988). Automated Cartographic Generalization for Linear Features. *Cartographica*, **25** (3), pp. 15–66. 20

Oden, J. T. and Reddy, J. N. (1976). *An Introduction to the Mathematical Theory of Finite Elements*. John Wiley & Sons. 54

Oevel, W. (1996). *Einführung in die Numerische Mathematik*. Spektrum Akademischer Verlag. 62

Radeva, P., Serrat, J., and Marti, E. (1995). A Snake for Model-Based Segmentation. In: *Proceedings of the Fifth International Conference on Computer Vision*, pp. 816–821. IEEE Computer Soc. Press, Massachusetts Institute of Technology, Cambridge, Massachusetts. 56

Regnauld, N. (1998). *Généralisation du bâti: Structure spatiale de type graphe et répresentation cartographique*. Ph.D. thesis, Université de Provence, Aix-Marseille 1, France. 8, 9, 141

Rieger, M. and Coulson, M. (1993). Consensus or Confusion: Cartographer's Knowledge of Generalisation. *Cartographica*, **30**, pp. 69–80. 8

Roberts, J. (1997). *The Development of an Autonomous Self Evaluating Algorithm for Map Generalization: Automated Feature Displacement*. Master's thesis, Department of Geography, University of Edinburgh. 21

Ruas, A. (1998). A Method for Building Displacement in Automated Map Generalisation. *International Journal of Geographic Information Science*, **12** (8), pp. 789–803. 11, 22

Ruas, A. (1999). *Modèle de généralisation de données géographiques à base de contraintes et d'autonomie*. Ph.D. thesis, Université de Marne la Vallée. 7, 8, 17, 19, 24, 25, 27, 160

Ruas, A. and Mackaness, W. (1997). Strategies for Urban Map Generalization. In: *Proceedings of the 18th ICA/ACI International Cartographic Conference*, pp. 1387–1394. Stockholm (Sweden). 7

Sarjakoski, T. and Kilpeläinen, T. (1999). Holistic Cartographic Generalization by Least Squares Adjustment for Large Data Sets. In: *Proceedings of the 19th ICA/ACI Conference*, pp. 1091–1098. Ottawa. 23

Schwetlick, H. and Kretzschmar, H. (1991). *Numerische Verfahren für Naturwissenschaftler und Ingenieure*. Fachbuchverlag Leipzig, Leipzig, Germany. 29

Sester, M. (2001). Massstabsabhängige Darstellungen in digitalen räumlichen Datenbeständen. Habilitationsschrift, Institut für Photogrammetrie, Universität Stuttgart. 23, 27

Sester, M. and Brenner, C. (2000). Typification based on Kohonen Feature Nets. In: *GIScience 2000. The First International Conference on Geographic Information Science*, pp. 21–22. 9

SGK Arbeitsgruppe (2001). Kartengrafik und Generalisierung. Technical Report. In prep. 18, 130, 162

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, **60**, pp. 51–92. 9

Sloan, S. W. (1986). An Algorithm for Profile and Wavefront Reduction of Sparse Matrices. *International Journal for Numerical Methods in Engineering*, **23** (1), pp. 239–251. 49

Tallis, M. (1995). *The Development of a Graphical User Interface for Interactive Feature Displacement*. Master's thesis, Department of Geography, University of Edinburgh. 21

Terzopoulos, D. (1987). On matching deformable models to images. *Topical Meeting on Machine Vision. Tech. Digest Series*, **12**, pp. 160–167. 61

Thomson, R. C. and Brooks, R. (2000). Efficient Generalisation and Abstraction of Network Data Using Perceptual Grouping. In: *GeoComputation 2000: Proceedings of the 5th International Conference on GeoComputation*. Manchester, UK. 82

Thomson, R. C. and Richardson, D. E. (1999). The "Good Continuation"Principle of Perceptual Organization applied to the Generalization of Road Networks. In: *Proceeding of the 19th ICA/ACI Conference*, pp. 1215–1222. Ottawa. 82

Töpfer, F. (1974). *Kartographische Generalisierung*. Ergänzungsheft Nr. 276 zu Perermanns Geographische Mitteilungen, VEB Hermann Haack, Geographisch Kartographische Anstalt Gotha/Leipzig. 20

Volkert, J. (1978). Algorithmen zur Generalisierung. *Nachrichten aus dem Karten- und Vermessungswesen*, Reihe I, Heft 75, pp. 111–132. 23

Walter, W. (1994). *Einführung in die Theorie der Distributionen*. BI-Wissenschaftsverlag. 29

Ware, J. M. and Jones, C. B. (1998). Conflict reduction in Map Generalization Using Iterative Improvement. *GeoInformatica*, **2** (4), pp. 383–407. 23, 26, 27, 130, 135, 143

Ware, J. M., Jones, C. B., and Lonergan, M. E. (2000). Map Generalization by Iterative Improvement. In: *GIScience 2000. The First International Conference on Geographic Information Science*, pp. 300–301. 23

Weibel, R. (1997). A Topology of Constraints to Line Simplification. In: M. J. Kraak and M. Molenaar (editors), *Advances in GIS Research II*, pp. 533–546. Taylor & Francis. 6, 8

Weibel, R. and Buttenfield, B. P. (1988). Map Design for Cartographic Information Systems. In: *GIS/LIS'88 Proceedings*, volume 1, pp. 350–359. 11

Weibel, R. and Dutton, G. (1998). Constraint-based Automated Map Generalization. In: *Proceedings of the 8th International Symposium on Spatial Data Handling (SDH)*, pp. 214–224. 7

Weibel, R. and Dutton, G. (1999). Generalising spatial data and dealing with multiple representations. In: P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind (editors), *Geographical Information Systems*, volume 1, pp. 125–155. John Wiley & Sons, Inc. 5, 6, 7

Williams, D. J. and Shah, M. (1992). A Fast Algorithm for Active Contours and Curve Estimation. *CVGIP: Image Understanding*, **55** (1), pp. 14–26. 56, 65, 131

Zeidler, E. (1985). *Nonlinear Functional Analysis and its Applications III - Variational Methods and Optimization*. Springer. 24

Curriculum vitae

Matthias Bader
born October 10, 1971, in Zurich
citizen of Zurich, Switzerland

Education

1978 – 1984	Primary school in Bülach.
1984 – 1990	High school in Bülach (Kantonsschule Zürcher Unterland), concluded with "Matura" exam type "B" (classical gymnasium).
1991 – 1997	Studies in geography at the University of Zurich. Minors in computer science, physics, geology and mathematics.
1997	Diploma in geography with a thesis on "Methoden zur Erkennung und Lösung von metrischen Konflikten in der Generalisierung von Polygonmosaiken" ("Methods for the detection and solution of metric conflicts in the generalization of polygonal maps") advised by Prof. R. Weibel and dipl. phys. Martin Heller.
since 1997	Studies at the Swiss Federal Institute of Technology Zurich (ETHZ) in mathematics.
1997 – 2001	Research assistant at the Department of Geography, University of Zurich. Work on the European Esprit-Project AGENT. Dissertation with a thesis on "Energy Minimization Methods for Feature Displacement in Map Generalization" advised by Prof. R. Weibel.