A FRAMEWORK FOR FORM-BASED CONCEPTUAL DESIGN IN STRUCTURAL ENGINEERING

by

Michael Gedig

B.A.Sc., The University of British Columbia, 1992
M.A.Sc., The University of British Columbia, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Civil Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)

April 2010

Abstract

Conceptual structural design is a process through which structural forms are created. The forms are shaped by a set of design requirements representing the expected function, and by constraints that reflect physical laws and practical limitations. There is no direct mathematical transformation from requirements to a form; the conceptual design process is nonlinear and iterative. Like all creative processes, it is most effective when ideas can be rapidly synthesized, dissolved, combined and evolved. In structural design, these ideas need to be evaluated in the context of performance, functionality, and cost. Conceptual design, compared to later design stages, is characterized by a high degree of uncertainty and a general lack of knowledge. A key objective in conceptual structural design is therefore to rapidly create, modify and evaluate vague or abstract structural forms.

This work describes a computational framework to support conceptual structural design, emphasizing the importance of form. Techniques from image processing, pattern recognition and linguistics are used to describe, classify, and reason with forms at high levels of abstraction. Most other computer applications in conceptual structural design describe design concepts in terms of words or through simplified spatial relationships. This work highlights the central role that visual information plays in formulating ideas in conceptual design.

The major contributions of this work are an efficient method for synthesizing conceptual designs of discrete structures, and the application of pattern recognition and visual case-based reasoning techniques to conceptual structural design. The framework is directed towards large-scale discrete structures characterized by interconnected linear elements. During synthesis, forms are initially created using topology optimization methods; these forms are processed to extract high level information that supports further structural optimization, including the assessment of stability and relative cost. The high level information is used to describe, classify and store conceptual forms for case-based reasoning. A novel feature of the work is that arbitrary images of shapes may be interpreted as structures by using visual similarity to infer potential boundary conditions, functionality, and behaviour for those shapes.

Abstract

This dissertation gives a complete description of the framework, along with sample applications. A proof-of-concept computer application is also described.

Table of Contents

List of Tables

List of Figures

List of Figures

Acknowledgements

I would like to thank my supervisor during this work, Dr. Siegfried Stiemer, for his invaluable insight and criticism, and overall for his enduring mentorship.

From my supervisory committee, I am especially indebted to David Halliday for his support throughout this endeavour. I would also like to thank Dr. Reza Vaziri, Dr. Farrokh Sassani, and Dr. Ricardo Foschi for their invaluable guidance.

I am very grateful to Empire Dynamic Structures for giving me the opportunity to pursue this work. Many of these ideas were inspired by the unique and challenging projects undertaken at EDSL, and by the work of my very creative and skilled colleagues.

I am grateful for the support and encouragement of Laura and Sven Eriksson.

I would like to thank my mother, Lydia Miller, for showing me the meaning of courage and tenacity.

I am thankful for my son Jakob for helping me keep things in perspective.

Above all, this work simply would not have been possible without my wife, Ann Marie. You have shown me patience and love in infinite measure.

1. Introduction

Conceptual structural design may be described as a process used to create structural forms. In general, design is the process by which a set of requirements are transformed into a description. Conceptual design is concerned with the early phases of the overall design process, where the emphasis is the interpretation of requirements and the exploration of multiple design ideas. In contrast, the later phases of design are concerned with detailed analysis and description, usually of a single concept. In structural design, the goal of the detailed design phase is to produce detailed models, drawings and instructions that can be used to fabricate the structure. In conceptual structural design, the objective is to produce structural forms that are likely to satisfy the requirements once detailed analysis and description have been completed.

The conceptual design phase is an important part of the life cycle of a project. Decisions made earlier in the design process have a proportionally higher impact on overall cost and schedule than later decisions. Conceptual design is a fluid process where design forms may be readily synthesized, dissolved, combined and evolved. As design progresses, concepts become more solidified and entrenched, and more difficult and expensive to change.

As a creative process, conceptual design becomes more effective as a wider a range of different configurations is investigated. The effective and rapid evaluation of design concepts, in terms of performance, functionality and cost, is therefore important in conceptual design. Although there are many tools available to evaluate structural systems in the later design stages, far fewer methods are applicable to conceptual design.

The conceptual design stage, compared to later design stages, is characterized by a higher degree of uncertainty and a greater lack of knowledge. At the conceptual stage, requirements may be flexible, vague, and subject to various interpretations. In complex or unique projects, the requirements are often refined as more information is gained about the potential performance of the system. Most analysis and design applications are designed to be used in the later design stages, where there is less uncertainty. Most applications do not explicitly model numerical uncertainty, and are not designed to handle incomplete or partially-specified input data. For example, analysis programs often require the user to make specific assumptions where data is not known, and do not accept a range of input values for an uncertain parameter, nor do they inform the user of the sensitivity of the result to such parameters. Most computer-aided design (CAD) programs force the user to enter a specific set of dimensions for an object, rather than allow the user to express the fact that a range of dimensions are acceptable. At the conceptual design stage, the ability to express and reason with uncertain or partially specified information is more important than in the latter design stages.

Conceptual structural design is a highly visual and symbolic process, where sketches and diagrams are essential tools to crystallize structural forms from ideas. The form of a structure is the most elemental constituent of a conceptual design solution. It is the form that is captured in the first quick hand sketches, before calculations and models are made. Human designers are very effective at remembering and manipulating concepts in their visual forms. Large databases of images can be quickly scanned to locate a specific image. Viewing an image or diagram initiates the recall of experiences from memory and suggests new areas of design exploration. More abstract images, such as rough sketches, are often more effective than precise drawings in conceptual design because they suggest different ideas to different viewers, opening up a wider expanse of design space.

Most other work to develop computer applications in conceptual structural design describes design concepts in terms of words or through simplified spatial relationships. A common feature of these applications is that designs are classified and indexed using words and text. Although some of these systems employ sophisticated hierarchical classification schemes capable of working at several different levels of abstraction simultaneously, the underlying nature of the classification is textual rather than visual.

A major objective in conceptual structural design is therefore to rapidly create, modify and evaluate vague or abstract structural forms. Although research in conceptual structural design has been ongoing for several decades, there appear to be few tools on the horizon to support the practicing structural designer in the conceptual stages. Currently, the engineer has few structured techniques to augment experience and intuition in this phase.

The main objective of this research is to develop a form-based framework that supports the conceptual design of structures. As a goal, the framework would enable the rapid generation and evaluation of new designs, and facilitate the reuse of past designs. The concept of form is intended to be central to the development of the framework, for both the generation and reuse of design concepts. Ideally, the framework would support the rapid transformation from design requirements to conceptual design form.

The framework developed in this research has been developed to test the following main hypothesis:

Patterns exist that relate structural forms to design requirements.

As a secondary hypothesis, it is proposed that:

A formal language exists to describe structural form.

To guide the development of the framework, a more specific set of objectives has been established. These objectives address current limitations in the conceptual structural design and propose solutions to overcome these limitations. Specific objectives are listed here.

- Case-based reasoning systems are a paradigm for applying past design experience to new design scenarios. A major shortcoming of case-based reasoning systems in structural design is the inability to index and retrieve cases on the basis of visual information. An objective of this research is to apply visual case-based reasoning methods to conceptual structural design.
- The conceptual design of buildings has been widely covered in the literature. Computer applications for conceptual building design are generally limited to buildings with simplified geometric relationships, such as rectilinear grids of beams and columns. An objective of this research is to develop a more general system that is useful for a wide range of structural applications, including scientific instruments, industrial equipment supports, and geometrically complex bridges and buildings.
- Sophisticated graphical user interfaces have been developed in the fields of architecture and industrial design, where natural, pen-based applications simulate sketching, clay modeling and other creative form-finding methods. Unfortunately, such applications do not recognize content such as the meaning of elements and relationships between them. Symbolic sketch recognition computer programs have been developed in architecture and mechanical design, but no such applications exist in structural design. An objective of this research is to establish techniques to develop conceptual structural designs from natural user input such as hand sketches.
- The form of many existing natural and human-made objects is intrinsically connected to an explicit or implicit set of constraints, including loads, support conditions, and functional and geometric constraints. Given two objects with similar form, it seems reasonable to ask whether those objects have other similarities. For example, do the objects have similar support conditions? In the case of human-made objects, do two designed artifacts have similar design requirements? An objective of this research is to develop visual similarity measures for structural design concepts. Used with a database of design cases, such similarity measures would be used to automate the retrieval of designs with similar form and enable other design features to be compared.
- Of the vast number of computer applications for engineering optimization, topology optimization methods are most applicable to conceptual structural design. Topology

optimization seeks the optimal connectivity and layout of members in discrete structures (trusses), and the optimal material distribution in continuum structures (plates or solids). Truss topology optimization is known to be computationally demanding, and is complicated by the fact that small changes, such as removing a single bar, may lead to large differences in overall stiffness. There is a need for efficient methods for finding and optimizing the topology of discrete structures. One of the objectives of this research is to develop an efficient framework for rapidly generating discrete topology at the conceptual design stage.

- Continuum topology optimization methods are efficient, well-established, and commercially available. The major drawback is that the output of such methods is not directly suitable for fabrication, particularly at scales larger than a few meters. For small scale fabrication of arbitrary shapes, material removal or deposition methods are common. At larger scales, structures are more often built up from discrete components. An objective of this research is to utilize the efficiency of continuum topology optimization methods, and extend the range of their applicability to large scale structures.
- Although methods for verifying structural stability are well established, efficient methods for generating stable structures are not. In truss topology optimization, stability is generally ensured using heuristics and generate-and-test methods. A common heuristic is to add sufficient members to ensure that all polygonal cells are triangular. Generate-and-test refers to the generation of a large number of different topological configurations, and filtering out the ones that are unstable. An objective of this work is to develop an efficient method for developing a stable configuration of discrete structural elements.
- Given the importance of visual and graphical information during conceptual design, it is remarkable that few computational tools for conceptual structural design exploit this information. In the fields of image processing and pattern recognition, there are well-established, rigourous techniques for manipulating graphical information. Such techniques have been applied for many years in areas such as medical imaging, remote sensing, and maufacturing, but few of these techniques have been applied to conceptual structural design. An objective of this research is to use such methods to generate conceptual designs and reason with those designs at relatively high levels of abstraction.

This work describes a computational framework that has been developed to support the conceptual design of structures, emphasizing the importance of form. Visual reasoning techniques are central to the framework. Methods from image processing, pattern recognition and linguistics are used to describe, classify, and reason with forms at high levels of abstraction.

In the framework, new conceptual design are generated, or synthesized, using a combination of mathematical optimization, image processing and pattern recognition methods. The reuse of past designs is implemented using visual case-based reasoning methods. During synthesis, forms are initially created using topology optimization methods; these forms are processed to extract high level information that supports further structural optimization, including the assessment of stability and relative cost. The high level information is used to describe, classify and store conceptual forms for case-based reasoning.

The intent of creating the framework is not to replace the designer or to develop a fully automated conceptual design system. Rather, the goal is to provide support during the design process and to augment the designer's intuition. The most important creative force during conceptual design is the human designer. The methods implemented in the framework draw from our understanding of the cognitive processes used by experienced designers. These processes include the retrieval of past experience, and the evaluation and modification of design concepts. The process of inferential reasoning is important in applying past experiences, given a mechanism for representing similarity between concepts.

The framework was not developed for all types of structures. Specifically the framework is designed to handle larger scale steel structures which are assembled from linear elements. Such "skeletal" structures are typically found in a wide range of applications, including bridges, buildings, towers, industrial and scientific equipment. The work was initially inspired by the challenges of designing a support structure for a 30-meter diameter optical telescope [Szeto et al., 2008]. Several existing conceptual design research systems have been developed for conventional buildings. In contrast, this work is more applicable to architectural applications requiring curvilinear and other more complex forms.

The main intent in developing the framework was to form the basis for a computer application to assist the practicing structural engineer during conceptual design. Another potential application for the framework is in education. Conceptual design is a skill that is learned primarily through experience. A system that allows students to draw high level connections between structural design requirements and the resulting forms could be a useful tool to assist in the teaching of conceptual design.

2. Background

This work draws from sources in a range of disciplines, including design theory, structural and mechanical engineering, image processing, pattern recognition, cognitive science and linguistics. This chapter describes the research background and context in which this work was developed.

2.1. Conceptual Design

The act of designing exists because of the human desire to change the world around us. Although design has been practiced for millennia, the study of design is a relatively new field. From antiquity, architects were the *master builders* responsible for design and execution of buildings, bridges, and other structures. The Roman architect Vitruvius left one of the earliest surviving major books on architecture, *The Ten Books on Architecture*, in which he asserted that a structure must exhibit the three qualities of *firmitas*, *utilitas*, and *venustas* – strength, utility and beauty [Vitruvius, 1999].

At the turn of the century, ideas began to germinate about the design of mass-produced goods. What was the meaning of mass-produced ornament and how could useful and beautiful products be created in a fragmented system where parts were made in different factories and assembled into a whole? The Arts and Crafts movement in England and the Bauhaus and De Stijl schools in Europe developed in response to the desire to imbue architecture and machine-produced goods with the care and intent of the master craftsman. In America, architect Louis Sullivan proclaimed "form ever follows function", and this credo had a strong influence on design at least until the 1930s.

During World War II the discipline of *operations research* was born and with it, close scrutiny of the design of products and the means and methods of production became the norm. Mathematical programming methods, such as the Simplex Method [Dantzig, 1963], were developed around this time, heralding the use of generalized optimization methods. Where design changes the world by making artifacts, the goal of optimization is to make those artifacts better.

2. Background

With the advent of high speed electronic computing in the 1950s, computational methods began to have a profound impact on design. Since then, innumerable applications have been developed to aid in the design process. Although these applications may free the designer from tedious calculations and repetitious drawings, in most cases they are not designed to help the designer at the conceptual stage.

Many of the systems developed to provide support for the conceptual stage of design had their origins in the field of artificial intelligence, a term coined in 1956 [Sriram, 2006]. Research in artificial intelligence in the 1960s and 1970s sought to expand the application of computers from repetitive numerical computations into reasoning, knowledge, planning and learning. *Notes on the Synthesis of Form*, a book about the process of design by the mathematician and architect Christopher Alexander [Alexander, 1964], had considerable influence in computer science, in the areas of object-oriented programming and pattern languages. In it, Alexander calls design "the process of inventing things which display new physical order, organization, form, in response to function." Areas of artificial intelligence that have been applied to conceptual design include knowledge representation, logic programming, expert systems, case-based reasoning, qualitative reasoning, grammars, shape annealing, and evolutionary algorithms.

2.1.1. Expert Systems

The first systems to provide computational support for engineering design, other than drafting and analysis, were knowledge-based expert systems. Expert systems aim to simulate the knowledge and analytical skills of human experts by reasoning from a database of stored heuristics ("rules of thumb"), textbook knowledge, standards and engineering experience. The architecture of an expert system involves two principal components: a problem-dependent set of data declarations called the knowledge base or rule base, and a problem independent program called the inference engine. The inference engine uses deductive reasoning to apply a series of inference rules until a desired goal is reached.

Expert systems were the first commercially successful applications of artificial intelligence. The rule-oriented description of human *expertise* and rule-based inference techniques have been brought to the industrial standard level in domains such as medical diagnosis and computer configuration.

Examples of expert systems in engineering design are HI-Rise [Maher et al., 1988], Tall-D [Ravi and Bédard, 1993], DOLMEN [Harty and Danaher, 1994], and an expert system to select structural systems for large span systems [Golabchi, 1997]. Other, more complex research systems that take into account the interdisciplinary nature of building design include DICE [Sriram et al., 1992], IBDE [Fenves et al., 1994], and ACL [Flemming et al., 1996].

Expert systems have been most successful where the domain model is well defined. They are notoriously brittle outside their narrow domain of application. Rule bases are difficult to expand with new data, and validation of existing rules is difficult. The efficiency of expert systems decreases with knowledge base size, and often a large amount of tedious general knowledge needs to be incorporated to provide common sense responses to queries. For these reasons, expert systems are rarely used in the structural engineering office.

Limited success has been achieved with expert systems in conceptual design. This is likely due to the lack of well accepted domain models of design synthesis [Maher and de Silva Garza, 1997]. Expert systems in engineering design have been largely supplanted by case-based reasoning, and an emphasis on experience rather than expertise.

2.1.2. Qualitative Reasoning

In the late 1970s, a group of artificial intelligence researchers sought to develop a knowledge representation capable of describing human commonsense reasoning and explanation about physical systems. De Kleer developed a framework for causal reasoning in which a qualitative description of the behaviour of a system is derived from a qualitative description of its structure [De Kleer, 1977; De Kleer, 1979; Kuipers, 1984]. The framework included a functional description that makes explicit which behaviours are possible for a system.

Research in qualitative reasoning influences conceptual engineering design in several ways. The conceptual design stage is characterized by uncertainty, and incomplete and imprecise information. Qualitative reasoning is a method of analyzing designs using non-numeric computations. Qualitative reasoning has also influenced the theory of design processes by providing a better understanding of how function, behaviour and structure are related in physical systems.

2.1.3. Design Prototypes

In the late 1980s, Gero developed semiformal computational models for the design process based on the concept of design *prototypes*. The models draw from concepts in artificial intelligence and prototype theory [Osherson and Smith, 1981]. It is based on the idea that designers organize their individual experiences into schema or classes consisting of abstract concepts. Gero defined a design prototype [Gero, 1990] as a conceptual schema for representing a class of elements derived from similar design cases, which provides a basis to initiate and continue design.

The design prototype model included a classification scheme for design knowledge, where *function* (F), *behaviour* (B), and *structure* (S) are the three most important classes. The prototype model is sometimes referred to as simply the *FBS framework* [Gero, 1990]. Design knowledge was also classified into relational knowledge between function, behaviour and structure; qualitative knowledge, computational knowledge, contextual knowledge, and other forms of knowledge. Using this computational model, the design process is initiated with a set of required functions and structures. These requirements are then used to initiate a search for candidate prototypes which are indexed using requirements. Once a prototype is selected from the search results, a design instance is created and used as the basis for a new design.

Design may be generally described as a process that transforms functions (F) into design *descriptions* (D). Design descriptions contain the information required to manufacture the artifact, such as drawings and documents. Gero noted that in general no direct transformation between function and description exists. The transformation between function and description occurs indirectly through behaviour and structure, as suggested in Figure 2.1. The transformation from structure to description is today typically carried out by computer-aided drafting (CAD) systems. Function represents the design intent, which is usually transformed into a set of expected behaviours (B_e), which expresses that intent. “The expected behaviour provides the syntax by which the semantics represented by the function can be achieved,” notes Gero. In practice, function is transformed to expected behaviour through the process of *formulation* or *specification*. The actual behaviour (B_s) is derived from the structure through the process of *analysis*.

Figure 2.1 illustrates a number of key concepts in the design process. The comparison between expected and actual behaviour is called *evaluation* in design. The process of design *synthesis* involves transforming function to expected behaviour, then selecting or combining structure to satisfy that behaviour. When structures are synthesized, they produce their own behaviours which may change the expected behaviours and lead to a change in the function, called a *reformulation*. Reformulation also occurs when no structure can be found that produces the expected behaviour. Design is an iterative process that follows a series of synthesis, analysis, evaluation and reformulation loops until a satisfactory solution is found.

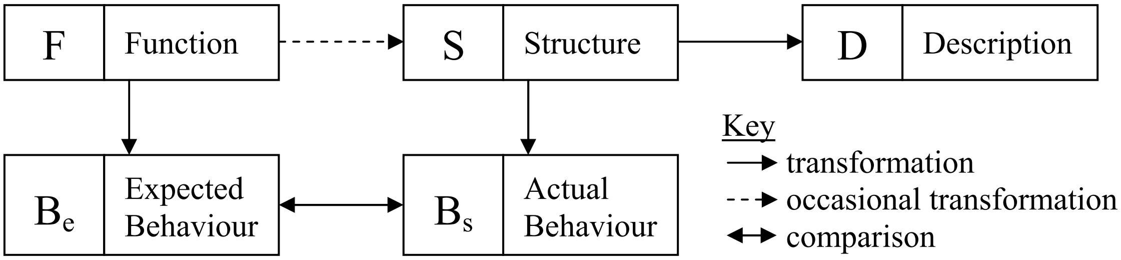


Figure 2.1. Model of design as a process [Gero, 1990]

The FBS model is a commonly-used representation of the overall design process, and contains the elements necessary to represent the conceptual design process. Since its development, Gero has coupled the design prototype representation with a number of different process models to support early architectural design. The prototype model was coupled with an evolutionary process [Gero, 1996], and with the creative design processes of combination, transformation, analogy, emergence and first principles [Gero, 2000].

2.1.4. Case-based Reasoning

Human designers draw heavily from past experiences when synthesizing new designs. Case-based reasoning represents a range of computational methods based on the recall and reuse of specific past experiences, called cases. The methods are analogous to the processes humans use to organize memory, and to recall information to solve problems and generate explanations. Case-based reasoning has been applied to a wide range of fields, including architecture, engineering, medicine and law. Two major considerations in applying case-based reasoning to design are the representation of design cases and process models for recalling and adapting design cases.

To represent design cases to support reasoning, experience must be abstracted into a symbolic form. A range of different approaches and models are described in the literature. In general, an effective design case representation considers the problem, the context, the solution, and the outcome. Design experience may be recorded as stories, sketches, drawings, charts, photographs, or video.

Design case recall involves finding a suitable case for solving the design problem at hand. In case-based reasoning, design cases are organized in a library, or *case base*. Since cases are represented in a symbolic form, the problem must also be formalized. The problem is called the *target case*, while the cases in the case base are called *source cases*. Design case recall consists of the subtasks of *indexing*, *retrieval* and *selection*. Indexing describes how source cases are identified and organized. Retrieval identifies which source cases have features that are relevant to the target case. Selection evaluates the retrieved cases so they can be ranked.

Case adaptation identifies the differences between a selected case and the target case, and changes the selected case to synthesize a solution that matches the target case. Changing the selected case involves evaluation to check the feasibility of the modified case as a design solution.

Designed artifacts influence us in many different ways. From their conception, through their design, manufacture, use and eventual destruction, these artifacts give us different experiences, depending on our viewpoint. Design cases are often complex, and the design case representation must be capable of supporting different perspectives, and different levels of abstraction. For complex cases, it is sometimes necessary to decompose the problem into multiple subcases, or to select and modify multiple source cases. Multiple subcases can be combined iteratively, one case at a time, by extending single case adaptation. Multiple source cases can be adapted iteratively, or simultaneously.

In domains where analytical or heuristic knowledge is available, the notion of design cases can be extended to include generalized design knowledge. For example, causal models can be used in the design of physical devices, and geometric information can be used in the design of buildings.

A key difference between rule-based reasoning (used in expert systems) and case-based reasoning relates to the size of information blocks. Rule-based systems are built from short clauses such as 'if *p* then *q*', called Horn clauses. In contrast, case-based systems reason with larger clusters of information in the form of cases. One of the limitations of expert systems is the difficulty in representing trivial or commonsense knowledge. Cases include such basic knowledge. Design cases encapsulate the tradeoffs, errors, revisions, iterations and lessons that are an intrinsic part of the design of real artifacts. While rule-based reasoning build solutions to a problem from scratch, case-based systems start with an existing solution to a similar problem, and then attempt to modify that solution to solve the problem at hand.

A number of case-based reasoning systems have been developed for architectural and structural engineering applications. A selection of these systems is presented in the following sections. A survey of case-based reasoning applications in design is found in Maher [Maher and de Silva Garza, 1997]. A survey of case-based reasoning systems for building design is found in [Rivard and Fenves, 2000a] and [Watson and Perera, 1997]

STRUPLE (STRUctural PLanning from Experience) is an early attempt to demonstrate the use of case-base reasoning in building design [Zhao and Maher, 1988]. The system proposes high-level structural subsystems from past experiences based on a problem description.

Architectural design is a challenging domain for case-base reasoning applications because of the lack of formal knowledge, which makes it difficult to develop a consistent design case representation. ARCHIE is an interactive system to aid architects in building design. The design case structure represents goals, constraints, outcomes and lessons at the conceptual design level. In response to a query, the system returns multiple cases, and can combine cases to create a high-level qualitative design that meets the intended goals. The system evaluates goals, plans and outcomes using a domain model, which captures causal relationships between case concepts. The work is extended in ARCHIE-2 [Domeshek and Kolodner, 1991; 1992], which augments cases with multimedia.

Research with ARCHIE highlighted several of the practical difficulties in developing architectural case-based reasoning systems. One of the findings was that users engaged in a creative process want to be in control of the process. User interfaces in creative applications should be interactive, providing ideas and alternatives but allowing the user to make design decisions. A strength of ARCHIE was the ability to store and index the work of previous architects, making the range of past design solutions more accessible to the user. One of the limitations of the system is that concepts are primarily expressed in textual, rather than visual form, which is a more natural mode of representation in architecture.

CADSYN (CAse-based reasoning for Design and SYNthesis) uses design cases and generalized knowledge to assist designers in conceptual structural design [Maher and Zhang, 1993]. CADSYN uses a hierarchical decomposition to represent design cases. The generalized knowledge includes decomposition information, structural design constraints, and procedural functions. The decomposition hierarchy can be used to generate a new design or adapt an inconsistent solution using constraint satisfaction methods. CADSYN does not emphasize user interaction and graphics.

CASECAD integrates case-based reasoning and model-based computer-aided design (CAD) techniques to assist structural designers during conceptual design [Maher and Balachandran, 1994]. The design case representation includes design indices, CAD drawings, and graphical illustrations of behaviours of design cases. CASECAD uses a flexible indexing system to provide multiple access paths to cases. Design cases are organized using a hierarchical decomposition and the function-behaviour-structure, or FBS, framework [Gero, 1990]. The FBS models identify a range of allowable values for each variable, providing generalized heuristic knowledge. CASECAD performs case retrieval and selection, but requires the user to perform case adaptation.

CADRE is a case-based reasoning system in which cases are geometric models of the conceptual structural and architectural layouts of a building [Bailey and Smith, 1994]. Cases represent building floor plans at different levels of abstraction, topological and dimensional. The system solves a system of constraints consisting of user-specified constraints and constraints that are generated from the architectural and structural layouts and their relationship to each other. The solution process uses two transformation adaptation methods: dimensional and topology adaptation. Dimensional adaptation involves the solution of a set of linear and nonlinear constraints on parameters used to describe the building. Topology adaptation uses a rule-based system to change the geometry. The system focuses on the representation and adaptation of cases rather than indexing and retrieval.

SEED (Software Environment to support the Early phases of building design) is a multidisciplinary effort aimed at providing computational support for the conceptual design of buildings [Flemming and Woodbury, 1995]. The emphasis of SEED is on supporting early design exploration through the rapid generation and evaluation of alternative concepts. The SEED-Config module within SEED supports the design of three-dimensional building elements in terms of spaces, subsystems and physical elements [Woodbury and Chang, 1995]. The module represents the overall form or massing of the building, the structural system, and the enclosure. The intent is to assist designers in generating designs, not to generate design automatically [Fenves et al., 2000]. SEED-Config includes case-based reasoning functionality to provide designers with initial potential solutions [Rivard and Fenves, 2000a].

2. Background

Cases in SEED are formalized in a generic information model called BENT, the Building Entity and Technology model [Rivard and Fenves, 2000b]. In BENT a building is represented as a set of building entities, which can be systems, subsystems, parts, a feature of a part, a space or joint [Gielingh, 1988]. Each building entity is a generic container that stores the entity's geometry, classifiers, properties, relationships, and design knowledge. Properties are named attribute-value pairs, grouped into three subsets: the functional unit (FU), the design unit (DU), and the evaluation unit (EU). The functional unit stores properties related to the intended purposes, requirements and constraints. The design unit captures properties required for design, such as material and shape characteristics. The evaluation unit represents the behaviour of the entity and records feedback on a design that has been implemented. Properties may be added to the building entity as the design progresses, supporting design evolution and exploration. Two kinds of relationships are modeled: the containment relationship models the hierarchical decomposition of design problems, and domain specific relationships store other essential relationships such as "support" and "connect" in structural engineering. Spatial relationships are not stored in the information model, because these are obtained directly from a geometric modeler. Design information is represented by a technology graph. The graph consists of technology nodes, each of which represents a known design alternative, the constraints that determine its applicability, and the computational procedure needed to assign values to the attributes defining that alternative.

Case retrieval in SEED is an interactive process in which the designer is in control of all steps. The user initiates the case-based reasoning process by selecting a building entity and activating the case retrieval process. The user has three options for preparing a query, one based on the selected entity, a second based on the hierarchical decomposition of the case, and a customized query using SQL/X. SEED uses the faceted classification scheme, where each building entity has a set of associated categories. Categories are hierarchically organized semantic networks, which provide an efficient mechanism for reasoning at a range of different abstraction levels.

Case adaptation in SEED uses the derivational replay method, which reproduces an existing set of reasoning steps in a new situation. This method is particularly well suited to SEED, where technology nodes record all the steps in the design process. Case accumulation and learning in SEED is also facilitated by the information model. Each building entity is essentially a self-contained case, because it includes its problem statement (the functional unit), the design solution (the design unit), the design process (the references to the technology nodes), and the design outcome (the evaluation unit).

SEED supports the reuse of both design information and design procedures. This capability distinguishes it from most other case-based reasoning applications, which reuse one or the other but not both. SEED includes conceptual design capabilities beyond case-based reasoning. These are discussed in a later section.

2.1.5. Building Design

Due to the importance of the building construction sector, much of the research into computational support of conceptual design has been in this area. Early collaboration between architects and engineers remains a challenging problem. Conceptual building design is a complex multidisciplinary activity characterized by rapidly changing and imprecise information, and a relatively high level of abstraction. Researchers have used several different approaches to assist the architect and engineer at the conceptual stage of building design. Early computer applications viewed building design as a planning problem, and drew heavily from research in artificial intelligence. More recent approaches favour model-based representation and reasoning.

Work in artificial intelligence has inspired a range of research applications for conceptual building design. Such applications use techniques such as formal logic and engineering first principles [e.g. Jain et al., 1991; Fuyama et al., 1997; Eisfeld and Scherer, 2003], grammars and shape annealing [Meyer, 1995; Shea and Cagan, 1999], fuzzy logic [Shen et al., 2001], and genetic algorithms [Krishnamoorthy and Rajeev, 2000; Grierson and Khajehpour, 2002; Sisk et al., 2003; Rafiq et al., 2003; de Silva Garza and Maher, 1996; Soibelman and Peña-Mora, 2003].

Given the importance of geometric modeling in the later stages of building design, some research programs have attempted to apply model-based reasoning to conceptual design. One example is SEED, a conceptual design system for buildings that integrates model-based reasoning and case-based reasoning. The SEED project highlights a number of important features in a conceptual building design system. Such a system should support design synthesis, design evolution, design exploration, multiple views, and be extensible [Rivard and Fenves, 2000b]. In addition, the system supports the rapid generation and evaluation of alternative design concepts, and reasoning at multiple levels of abstraction. SEED uses a three level information model: an object-oriented *data model*, an *information model* (BENT) that stores and shares design data, and acts as a case library, and a *conceptual model* [Rosenman and Gero, 1996]. The conceptual model defines the types of objects, relationships and data needed to fully represent the information in a given design domain. Effectively, the conceptual model specifies the semantics of the design domain using the syntax in the information model [Rivard and Fenves, 2000b].

The design process is alternately a top-down and a bottom-up process. Bottom-up design starts with the definition of components and connections, followed by aggregation into subassemblies. In contrast, top-down design starts with an overall definition which is refined into functional subsystems and assemblies, and physical components. A top-down method is often favoured for conceptual design because it supports successive design refinements [Martini and Powell, 1990]. An early approach to top-down design in buildings was the "total system" approach [Lin and Stotesbury, 1988], which starts with a 3D structural mass that observes the architectural space-form requirements, then proceeds to structural subsystems consisting of 2D elements and finally to 1D structural connections. Other top down models include those of [Sauce et al., 1992], [Hauser and

Scherer, 1997], and [Sacks and Warszawski, 1997]. While many researchers view the design process as primarily a top-down process, others see design as a nonlinear process that involves reasoning at alternating levels of detail, and in practice involves backtracking.

Geometric modelers in conceptual building design systems often have very limited modeling capabilities. The geometric modeler implemented in SEED is limited to geometric objects that are rectilinear in shape and positioned orthogonal to the global axes. In many building applications, the hierarchy and semantics of structural elements is well defined. Each structural element in a building is relatively easy to classify as a beam, a column, or a brace. Structural elements in general structural applications are not so readily classified as in buildings. The strength of SEED lies in the support of recurring building types rather than in support of universal structural configurations [Fenves et al., 1995].

Applications to support building design use a range of user interface approaches. Different views are often required to suit the various needs of users, including architects, structural engineers and mechanical engineers. Other differences between applications relate to the balance between user control and automation. For example, StAr (Structure-Architecture) [Mora et al., 2005; 2008] uses an interactive algorithmic process controlled by the engineer. Other researchers favour more automated approaches such as automated reasoning and generative structural design [e.g. Meyer, 1995; Krishnamoorthy and Rajeev, 2000; Hofmeyer et al., 2006].

Early collaboration between architects and engineers provides useful insights into conceptual structural design. While architects are trained in design synthesis, the bulk of engineering education focuses on analysis. Concepts relating to the overall appearance and layout of structures, like repetition, visual rhythm, order, modularity, symmetry, balance, scale, proportion and unity are familiar to the architect [Mora et al., 2003]. The architect's terminology reflects a level of abstraction that is useful in effectively synthesizing a range of conceptual design configurations.

2.1.6. Commercial Software

In the latter phases of building construction, highly detailed building information models are commercially-available and in common use. Current versions of industry-standard CAD applications such as Autodesk Revit provide some support for conceptual design of buildings, and integrate architectural and structural models [Mora et al., 2008]. Neutral file formats transfer information between participants in the construction process. For example, the CIMsteel Integration Standard CIS/2 supports information transfer between detail design, fabrication, and erection of steel structures. Efforts have been made to extend existing information models and CAD applications upstream to support the conceptual design of buildings, however such efforts have not been completely successful. The standard building information models are bottom-up models built element by element that were not designed to support higher level concepts like structural subsystems and assemblies. Effective conceptual design requires a top-down approach, with representations that support higher levels of abstraction.

Support for conceptual design by commercial CAD software programs has improved in recent years but the rapid generation and evaluation of imprecisely specified concepts is still not possible with these programs. The CAD software market is dominated by a relatively small number of major vendors. Although new software versions are released regularly, the basic functionality of CAD software in producing working drawings has remained unchanged for some time. Many of the new features that have been added relate to interoperability between different software applications. All the major CAD programs directly support a range of analysis applications, such as finite element analysis, kinematic simulation, and thermal analysis. Integration between CAD applications and production software is highly advanced, particularly in structural steel design and fabrication. The CIS/2 standard defines a neutral file format that allows steel detailing information to be transferred directly from the CAD package into cost estimating and production tracking applications, as well as computer controlled fabricating equipment. With such an advanced level of support for the latter stages of the design process, developers of CAD programs have begun to focus more attention on the conceptual stages. Intuitive sketch-based user interfaces have been developed or acquired by the major commercial CAD packages (e.g. Autodesk Alias and Google SketchUp). Such interfaces are generally not well integrated into the CAD production environment, which limits their effectiveness as conceptual design tools. Support for design synthesis is the part of conceptual design that is most lacking from commercial applications. Although improved user interfaces make it quicker to transfer ideas to models and to analyze those ideas, they will not generate new concepts. Commercial CAD applications do not understand the language of requirements and specifications, and require the designer to interpret these in the form of well defined design concepts. The framework presented in this research directly interprets design requirements, synthesizes new concepts, and evaluates those concepts even though they are not precisely defined.

2.1.7. Visual Case-Based Reasoning

Many applications in case-based reasoning retrieve cases based on matching of indexed attribute-value pairs. Even in domains where visual and geometric information is essential, such as architecture, case retrieval on textual attributes is frequently used. Although sophisticated methods of text-based case retrieval have been developed in the field of document processing, such retrieval methods have limitations in dealing with primarily graphic data:

> "It is hard to capture geometric relationships by means of attribute and attribute values; it is even harder and in most cases practically impossible to base retrieval on such a description since the names of the involved objects are irrelevant and mostly unknown; what matters are the relationships between corresponding objects in the query and source cases. Therefore one has to adopt specialized retrieval methods." [Gebhardt et al., 1997, p.65]

FABEL is an architectural case-based reasoning system that reasons with building floor plans [Gebhardt et al., 1997]. FABEL uses specialized geometric case retrieval methods to reason with cases that contain diagrammatic or geometric information. These specialized methods include the concept of *gestalts*, the Object Density Map, and graph matching algorithms. A central feature of these methods is the use of multiple views at different levels of abstraction.

A general definition of gestalt is a "symbolic configuration or pattern of elements so unified as a whole that its properties cannot be derived from a simple summation of its parts." In FABEL, gestalts are characteristic geometric arrangements of certain components in a building. It was observed that a limited number of gestalts represents a large number of specific configurations, illustrating that gestalts are a useful method to generalize geometric layouts. The set of gestalts used in FABEL is shown in Figure 2.2.

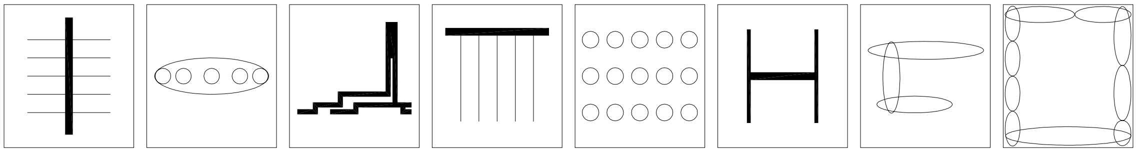


Figure 2.2. Set of gestalts in FABEL [Gebhardt et al., 1997]

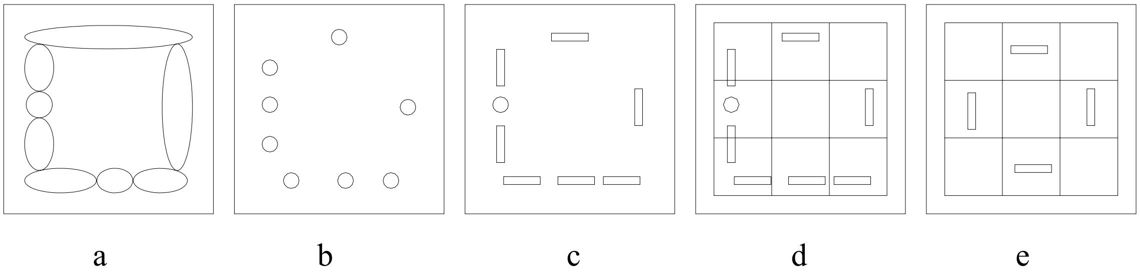


Figure 2.3. Representing gestalts with successive abstraction [Gebhardt et al., 1997]

Figure 2.3 illustrates the process of developing a gestalt from specific configurations of objects using a series of successive abstraction operations. Figure 2.3.a identifies a group of objects, where each ellipse represents a bounding box that contains a building component, such as a room[1]. The configuration shown in Figure 2.3 is commonly referred to as a *quadrangle* in architecture. Figure 2.3.b shows the centers of the objects, and Figure 2.3.c adds alignment information. Bars represent objects with an alignment, and circles indicate objects without alignment. Figure 2.3.d show a sketch which uses a grid to abstract from scale, and from distortion. Finally, Figure 2.3.e indicates an abstraction which neglects the exact position and number of objects. In FABEL, gestalts

[1] Ellipses, as opposed to rectangles, are used to represent bounding boxes because large number of rectangles that share edges are indistinguishable.

are detected by an algorithm that compares stored patterns, like the one in Figure 2.3.e, with patterns generated from the design object under consideration. Matching gestalts are attached to the case as descriptors, or index terms.

FABEL uses another geometric abstraction method, the Object Density Map (ODM) [Coulon and Steffens, 1994], to view specific architectural layouts at different levels of resolution. When images are compared on a pixel-by-pixel basis, large differences may exist even when images convey very similar information. An ODM is constructed by overlaying a low resolution grid over an image. The number of image pixels in each grid square is counted and used to calculate the image density, a number between 0 and 1. The image density on each grid square is represented using a set of overlapping grayscale intervals, as shown in Figure 2.4. The intervals are narrower at lower density to bring better definition to sparse images [Rosenfeld, 1969]. Two images are considered similar if every set of corresponding grid squares share at least one interval of grayscale values. The similarity function is symmetric, but not transitive, however it is invariant with respect to rotation, translation, or sizing. A numerical value for the similarity between two images is calculated with a normalized sum of differences between the corresponding pixels.

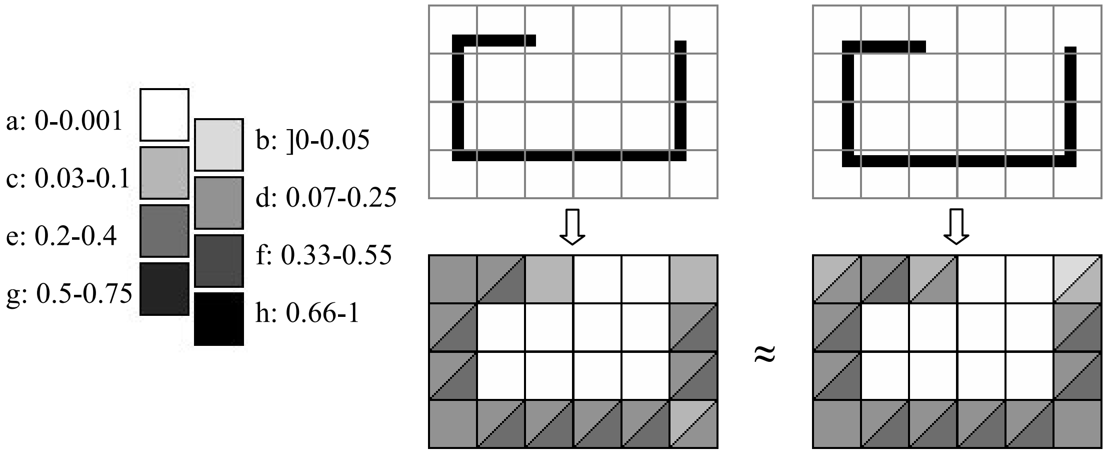


Figure 2.4. Similarity of object density maps [based on Gebhardt et al., 1997]

A third method for determining similarity between two cases in FABEL uses graph matching algorithms. Relational graphs can be used to produce an abstract representation of a configuration of physical objects. [Winston, 1975] used a version of relational graphs to describe structures, such as arches and towers, for the purpose of generalizing and classifying images. TOPO [Börner et al., 1996], a subsystem of FABEL, detects spatial relationships between two objects by comparing the graph representation of those objects. TOPO uses the maximum common subgraph (MCS) to measure the degree of structural similarity between cases. The maximum common subgraph algorithm is NP-complete, so the time for computing similarities between cases may be prohibitive. In alternative approach, constraint satisfaction methods were applied in a visual case-based reasoning system for the retrieval of 2D line drawings [Yaner and Goel, 2003].

Case-based reasoning has been integrated with image processing methods in fields such as medical diagnosis [e.g. Plaza and López de Mántaras, 1990] and molecular biology [e.g. Jurisica and Glasgow, 2004]. A common approach is to use image processing to compute a vector of numerical features for each case image. The feature vectors of two cases are compared to determine similarity. The degree of structural similarity is determined indirectly, in contrast to the direct comparisons used in FABEL.

In the application of case-based reasoning to visual information such as images and diagrams, the methods of pattern recognition are important. These methods are discussed in more detail in a later section.

In architecture [Gross and Do, 1995] describe a method for retrieving images from a database using a freehand sketch as input. This method uses a relatively simple heuristic: given two drawings, it compares the type and number of spatial elements and spatial relations by counting.

2.1.8. User Interfaces

Drawings in their various forms are an essential part of the design process. In the later stages of design, highly specific and detailed fabrication drawings provide precise instructions on how to manufacture an artifact. In the embryonic stages of design, unstructured, informal sketches assist the designer in retrieving images from long-term memory [Simon, 1969]. Sketches support reasoning at a number of different levels of abstraction. They may be relatively unconstrained and ambiguous, encouraging reinterpretation and the emergence of new concepts. The role of sketching in design and its relationship to cognitive processes has been studied by researchers in design and cognitive science [Purcell and Gero, 1998].

Computational support for drawings is well established in the latter stages of design. Support in the early stages has received much less attention. Design is a process of incremental formalization proceeding from abstract to more specific forms. Paper-like or pen-based computer applications in architecture, industrial design and graphic design have been commercially available for some time (e.g. Autodesk Alias and Google SketchUp), but most do not recognize content, such as the meaning of the elements and the relationships between them. The Electronic Cocktail Napkin [Gross, 1996] application is a prototype pen-based interface to allow architects to sketch with various degrees of precision or ambiguity. The system recognizes hand-drawn multi-stroke symbols (glyphs), spatial relationships between symbols, and identifies matching configurations from a library. The EsQUIsE prototype [Leclercq, 1999] interprets architect's sketches, and extracts concepts such as walls, functional spaces and space topology.

2.1.9. Mechanical Design

Qualitative reasoning about the behaviour of mechanical and other physical systems developed as a field of artificial intelligence research [e.g. De Kleer, 1979; Kuipers, 1984]. Causal and heuristic knowledge about such systems were incorporated into early case-based reasoning systems such as KRITIK and KRITIK II [Goel, 1989], and CADET [Navinchandra et al., 1991].

Given a drawing of a new device, human experts are capable of understanding the components of the device (structure), as well as what the device does (function) and how it works (behaviour). Several systems have been developed to attempt to formalize the interpretation of drawings. GeoRep is a diagrammatic reasoning system that takes an arbitrary 2D line drawing as input and produces a description of the physical system in the drawing [Ferguson and Forbus, 2000]. Some applications have also used structure-mapping to use analogy to make inferences at the structural level [Falkenhainer et al., 1990]. With the ARCHYTAS system [Yaner and Goel, 2007], the goal is use visual analogy to infer structural components, connections, causal interactions, processes, and functionality from 2D unlabeled drawings.

2.1.10. Conceptual Analysis Methods

Many computational analysis procedures have been developed for the latter stages of design, in which the design problem has been well specified and the input is complete. In contrast, the conceptual design phase is characterized by incomplete, uncertain and imprecise information. Conceptual design therefore requires a different set of analysis methods than those used in preliminary and final design. The analysis methods must not only handle uncertainty, they must provide solutions quickly. Effective conceptual design synthesis requires rapid analysis procedures that allow the designer to easily explore a broad expanse of design space and to evolve concepts.

Approximate analysis techniques are widely used by experienced designers. Before the advent of computers, approximate techniques were essential. A broad range of computational analysis techniques are used to handle uncertainty and provide for rapid assessment of conceptual designs. Examples of approximate analysis in computer-supported conceptual design of buildings are found in [Ravi and Bédard, 1993] and [Fuyama et al., 1997].

Where the finite element method is used for analysis, structural reanalysis methods may be used for exploring design variations. Both exact and approximate reanalysis methods are available; most of the exact methods are based on the Sherman-Morrison-Woodbury formulae [Sherman and Morrison, 1949; Woodbury, 1950].

A range of methods are used to deal with uncertainty in engineering applications. The most common approach is to use probability theory. Many processes in nature are characterized by random uncertainty, and are well represented by probability distributions. Probability theory is best suited to model the uncertainty related to naturally random behaviour, called *aleatory* uncertainty. During conceptual design, however, much uncertainty is characterized by the simple lack of information, or *epistemic* uncertainty. Approaches such as interval analysis [Moore, 1966] and constraint satisfaction methods, are often more appropriate under conditions of epistemic uncertainty. Evidence theory (or Dempster-Shafer theory) is a generalization of classical probability and possibility theory that can represent probability distributions, membership functions (from fuzzy set theory), and intervals [Shafer, 1976].

2.2. Optimization

Optimization aims to find the best configuration of a system given a set of constraints. What is the best configuration depends on the context and the overall objective in improving the current configuration. In aerospace engineering, a common objective is to minimize the mass. In civil and automotive engineering, the objective is usually to minimize cost. In light of the impact of production of artifacts on our environment, and due to shortages of energy, material, and labour, the optimization problem now generally reduces to one of achieving the maximum of benefit from limited resources.

2.2.1. General Optimization

Classical optimization was largely concerned with finding an optimum function that satisfies a differential equation, the *distributed parameter* problem. An example is seeking the optimal moment of inertia along the length of a beam, subject to the governing beam equation. With the advent of high speed computation and techniques such as finite element analysis, differential equations were largely replaced by algebraic equations, and *discrete parameter* formulations became more common. Discrete parameter optimization refers to systems described by a set of variables that vary continuously within a given range. Another optimization method, *integer* or *combinatorial optimization*, refers to problems where variables may only take on discrete values. *Mathematical programming* refers to the general study of problems in which one seeks to minimize a real function by choosing the values of real or integer variables from an allowed set. Many practical problems involve integer parameters, often due to production limitations. For example, steel mills produce a limited selection of sizes of rolled steel shapes. Since integer problems are computationally more difficult than problems with continuous variables, it is quite common in practical optimization to substitute discrete functions with continuous approximations.

Classical optimization deals with exact, analytical solutions to optimization problems. Although many practical problems are not amenable to closed-form solutions, classical techniques still play an important role in optimization. Most importantly, these techniques give insight into the existence and uniqueness of solutions to optimization problems. For general solutions to most real-world optimization problems, computer-based mathematical programming methods are used.

2. Background

Optimization methods are used to find a set of design parameters, $x = \{x_1, x_2, \ldots, x_n\}$ that define an optimal configuration. A *general problem* in optimization is stated as:

$$\begin{array}{llll} \text{minimize:} & f(x) & & \\ \text{subject to:} & \text{equality constraints} & G_i(x)=0, & i=1,\ldots,m_e \\ & \text{inequality constraints} & G_i(x)\leq 0 & i=m_e+1,\ldots,m \end{array}$$

where x is the vector of length n design parameters, $f(x)$ is the objective function, which returns a scalar value, and the vector function $G(x)$ returns a vector of length m containing the values of the equality and inequality constraints evaluated at x.

The selection of an appropriate procedure depends on the nature of the variables, the objective function and constraints. In a *linear programming* problem, both the objective function and constraints are linear functions of the design variable. A *quadratric programming* problem concerns the minimization or maximization of a quadratic objective function with linear constraints. In a more general *nonlinear programming* (NP) problem, the objective function and constraints are nonlinear functions of the design variables. The solution to the NP problem generally requires an iterative procedure involving the solution of an LP, QP or unconstrained subproblem.

A wide range of solutions exists for *unconstrained* optimization problems. Practical design, however, is generally concerned with *constrained* optimization. The general approach in constrained optimization is to transform the problem into simpler subproblems and to solve the subproblems as part of an iterative algorithm. A class of efficient solutions employs the Kuhn-Tucker (KT) equations, which are necessary conditions for optimality for a constrained optimization problem. For a *convex* programming problem, where $f(x)$ and $G_i(x)$, $i=1,\ldots,m$ are convex functions, the KT equations are both necessary and sufficient conditions for a global solution point.

In practice, most optimization problems cannot be shown to be convex, and have several local minima. Mathematical programming techniques are typically local in nature and return only one of these local minima. In order to find the global minimum, nonconvex problems can be formulated using a series of convex approximations. For example, in *sequential linear programming* (SLP) a linear approximation of the objective function and constraints produces a linear programming problem. Similar methods, such as s*equential quadratic programming* (SQP) and the *method of moving asymptotes* (MMA), have been used in structural optimization.

For problems that are highly nonconvex or nonlinear, classical optimization methods often fail and more robust, *global optimization* techniques are required. While most global methods still cannot guarantee a global optimum and are often computationally inefficient, they may be the only methods available for some problems, and therefore are useful in many complex, practical problems. *Evolutionary computing* (EC) methods fall in the category of global optimization methods. Inspired by the biological processes of evolution and selection, evolutionary computing may be understood as a parallel, stochastic optimization process in which a population of solutions undergoes a process of gradual change [Kicinger et al., 2005]. Evolutionary computing encompasses a range of optimization methods which are classified as one of four main evolutionary algorithms [e.g., Rafiq et al., 2005]: the *genetic algorithm* (GA) [Holland, 1975], the *evolutionary strategy* (ES) [Rechenberg, 1965], *evolutionary programming* (EP) [Fogel et al., 1966] and *genetic programming* (GP) [Koza, 1992]. All four of these algorithms have been applied to engineering design, however the GA and GP are the most common. All EC methods start with the generation of an initial population of solutions. Then, over a sequence of generations, parent solutions are selected and reproduced, and offspring are selected for continued reproduction based on their fitness. After a number of generations, the process is expected to converge to a solution with "maximum fitness". Genetic algorithms generally encode design variables as fixed length binary strings, or *genotypes*. Genetic programming is similar to genetic algorithms, except that parse trees are used instead of fixed length strings.

2.2.2. Multiobjective Optimization

The general problem formulation optimizes a single objective function. Practical design problems require that a number of objectives be addressed simultaneously. *Multiobjective optimization* involves the minimization of a vector of objectives $F(x)$, where the problem stated as:

$$\begin{array}{lll} \text{minimize:} & F(x) & \\ \text{subject to:} & G_i(x) = 0, & i=1,...,m_e \\ & G_i(x) <= 0, & i=m_e+1,...,m \end{array}$$

Often the objectives are competing and their relative importance must be weighed in order to make tradeoffs. The relative importance of the objectives is generally not known in advance, and some analysis is usually done to assess the capabilities of a system before tradeoffs are made.

Since $F(x)$ is a vector of objectives, if any of the objectives are competing, there is no unique solution to the problem. In this case, the concept of *noninferiority* [Zadeh, 1963], or *Pareto optimality* [Da Cunha and Polak, 1967; Censor, 1977], is used to assess multiple objectives. A vector of design variables x^* is said to be noninferior if, for any other vector x either the values of all the objective functions remain the same, or at least one of them degrades compared with its value at x^*. Multiobjective optimization is concerned with the generation and selection of noninferior solution points. There is a wide range of techniques for multiobjective optimization. Two of the most common are the *weighted sum* method, and the *goal attainment* method.

The weighted sum strategy creates a single objective function by constructing a weighted sum of all the objectives. The vector of objectives $F(x)$ is thus converted into a scalar. An advantage of the weighted sum approach is that the single objective function can be handled using the GP formulation. There are several challenges in using this approach, however. The selection of appropriate weighting coefficients is often difficult, since the coefficients do not necessarily correspond directly to the relative importance of objectives. Also, they are not able to clearly express tradeoffs between objectives. Another limitation is that certain points on the noninferior solution boundary may be inaccessible.

The goal attainment method [Gembicki, 1974] avoids some of the difficulties of the weighted sum method, while still using standard optimization procedures. The method specifies a set of design goals related to the objectives, allowing objectives to be over- or under-achieved. A set of weighting parameters allows the user to specify the relative importance of the goals.

In multiobjective design optimization problems, the goal is generally to find a large number of widely differentiated Pareto-optimal solutions. Classical optimization methods like the weighted sum and goal attainment method are limited because most of them produce only one solution on the Pareto front. Also, such methods are sensitive to the shape and continuity of the front. Evolutionary algorithms are well-suited to multiobjective optimization problems because they are population-based, and therefore can generate an entire set of Pareto-optimal solutions in a single run. Also, evolutionary algorithms are more robust than classical methods, and are less sensitive to the shape of the front. There has been a significant amount of research in the area of Multi-Objective Evolutionary Algorithms (MOEA). For an overview of MOEAs, see, for example, [Coello, 2006].

2.2.3. Structural Optimization

Mathematical optimization techniques have been used in a wide range of structural design and analysis applications. Many early applications dealt with exact, analytical solutions for specific problems. Computer-based methods expanded the range of potential optimization problems, and several general structural optimization methods have emerged. These general methods are particularly important in the conceptual design stage, and are emphasized in this overview.

Most early structural optimization methods sought to minimize the weight or volume of the structure considering constraints on member stresses or overall deflection. Methods have now been developed to handle more complex constraints related to buckling, cost, plasticity, and reliability.

2.2.3.1. Truss Optimization

Some of the earliest applications in structural optimization were to truss and truss-like structures. In 1904, Michell developed a theory for the layout of minimum weight structures composed of axially loaded bars [Michell, 1904]. The bars are perpendicular to each other, and follow the lines of maximum tensile and compressive stress (Figure 2.5).

Michell structures are of more interest in theory than in practice. They are designed for only one loading case, and consist of an infinite number of bars of nonstandard length.

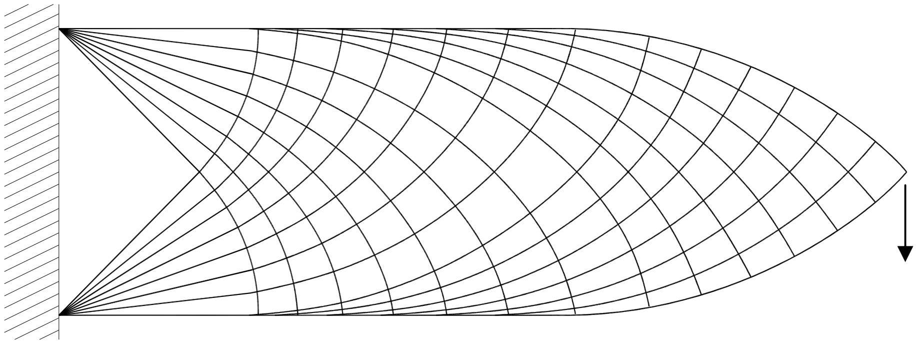


Figure 2.5. Michell structure [from Rozvany, 1997]

The most basic truss optimization method is *sizing optimization*, where the design parameters are the truss member cross-sectional areas. In *geometry optimization*, the design variables are the truss node positions. *Topology optimization* seeks the optimal pattern of connectivity or spatial sequence of members in a structure. The optimization of cross-sectional area, node position, and topology is termed *layout optimization* or *configuration optimization.*

The truss sizing problem was the subject of much analytical work in the 1960s and 1970s. Generally, the goal was to find an assignment of cross-sectional areas that produced the minimum weight structure, subject to stress or displacement constraints. This is a nonconvex problem which may be solved using a constrained optimization algorithm together with finite element solver. For example, the optimality criterion method has been used for the sizing problem [e.g. Taylor and Rossow, 1977; Rozvany, 1989]

Truss topology optimization commonly uses the ground structure [Dorn et al., 1964], consisting of a grid of nodes and potential structural members (Figure 2.6). The task is to find an optimal truss structure that satisfies all load and support conditions, and consists of a subset of the potential members. Topology optimization is inherently a discrete problem, but can be handled as a continuous sizing problem that allows for zero cross-sectional areas. In contrast to the standard sizing problem, there are major differences. The number of design parameters is much larger than the number of degrees of freedom. The use of zero cross-sectional area can lead to a singular stiffness matrix, if the complete ground structure is considered. Since most optimal designs have a singular stiffness matrix, standard structural optimization procedures cannot be used.

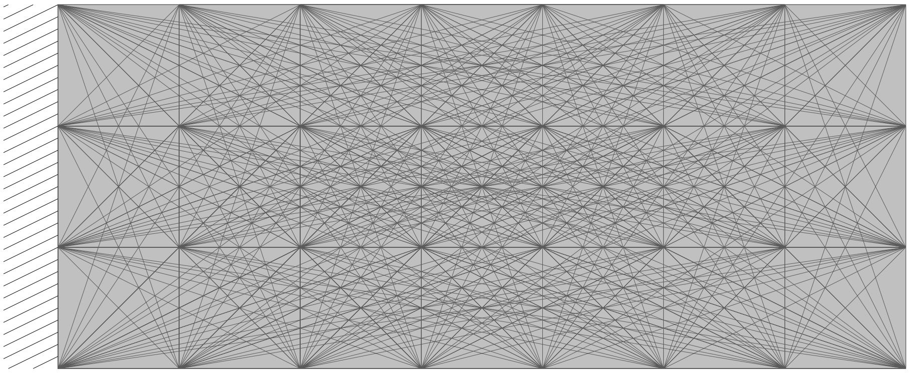


Figure 2.6. Ground structure

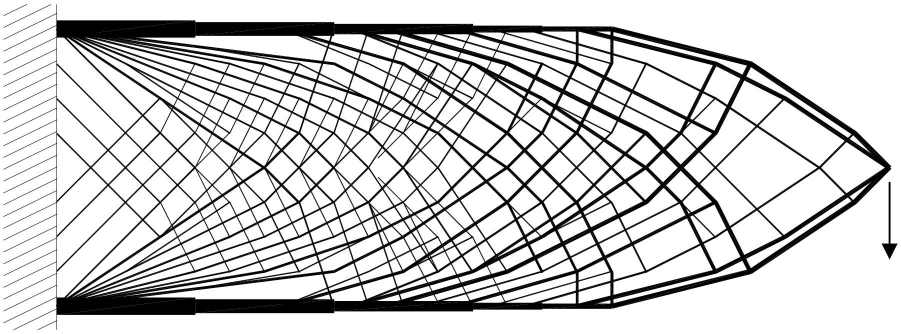


Figure 2.7. Solution using ground structure method [after Rozvany, 1997]

Different computational procedures may be used to solve the topology problem, depending on the formulation of the problem [Bendsøe and Sigmund, 2004]. An example of a solution is given in Figure 2.7. Early researchers [e.g. Dorn et al., 1964] assumed a plastic design constraint, leading to a linear problem that was solved using the Simplex method. The problem of finding the stiffest truss for a given volume, the minimum compliance problem, may be solved using optimality criteria methods. This approach can be interpreted as an implementation of a sequential quadratic programming technique [Svanberg, 1994]. For more general design situations involving stress and displacement constraints, problems are large scale and non-convex. The truss topology problem may also be solved using a discrete optimization approach, using such techniques as simulated annealing [Reddy and Cagan, 1995] and evolutionary methods, which are discussed later in this section.

It was observed early on that the ground structure approach can lead to mechanisms which are in equilibrium under the given loads [Dorn et al., 1964]. In these cases overall stability can be ensured by adding infinitesimal members, or by using multiple loads. Also, truss members may contain interior nodes, which must be removed to ensure stability.

The optimal topology is sensitive to the number of nodes, particularly for coarse grids. For finer grids, the results are less sensitive, however the number of potential members increases quickly with decreasing grid spacing. The ground structure method can be augmented with node position design variables, allowing the use of coarser grids, but this also leads to a highly nonlinear objective function [Topping, 1983], and increased computational difficulty. [Bendsøe and Sigmund, 2004] outline computational procedures for several different formulations of the truss topology optimization problem, noting that the truss topology design problem is a "very challenging mathematical problem".

The truss geometry optimization problem seeks the optimum positions of nodes that satisfy a set of constraints on a structure. Since both displacements and stress are nonlinear functions of node position, the problem is nonlinear. [Svanberg, 1982] used mathematical programming methods and finite element models to solve the geometry problem. As nodes are repositioned during truss geometry optimization, nodes may move close together. [Pedersen, 1992] used a two stage algorithm where the first stage performs geometry optimization, the second stage merges nodes when element lengths are below a given threshold.

Evolutionary computing methods have been studied extensively in structural optimization applications. Geometric and topology optimization of discrete structures pose computational difficulties for gradient-based methods. In layout optimization, even small changes in topology can lead to significant changes in behaviour. For example, removing a single member can change a highly statically indeterminate structure into a mechanism. Evolutionary computing methods such as genetic algorithms are suited to such applications because they do not require gradient information. Another advantage of evolutionary methods is that they naturally handle discrete variables, such as member sizes, which arise frequently in practical engineering applications.

Some of the earliest applications of evolutionary computing to structural engineering were in the sizing and geometric optimization of trusses. [Goldberg and Samtami, 1986] appear to have first suggested the use of a GA for structural optimization, and applied the method to a 10-bar plane truss problem. [Jenkins, 1991] employed a GA for discrete problems involving both sizing and geometric variables. [Rajeev and Krishnamoorthy, 1992] used a GA with a penalty function to perform truss sizing optimization with stress and displacement constraints. [Adeli and Cheng, 1993] used a penalty function approach to handle constraints on cross-sectional area, stress and displacement. [Grierson and Pak, 1993] applied a GA to layout optimization of skeletal building structures, using an approximate analysis method to improve efficiency. [Koumousis and Georgiou, 1994] solved a mixed sizing and layout problem in steel roof truss design with constraints on stress, deflection, and connection types. [Rajeev and Krishnamoorthy, 1997] applied a variable string length genetic algorithm (VGA) to represent topology variations in truss layout optimization. [Louis and Zhao, 1995] used a GA, augmented with engineering heuristics, to do truss layout optimization. The heuristics ensure that during topology generation, members are connected so that the truss is, with high probability, triangularized and stable. Mutation sometimes leads to unstable structures, which are made stable by adding bracing. [Soh and Yang, 2001] used genetic programming for truss layout optimization, in which truss topology is encoded in the form of parse trees. The genetic programming method is shown to be flexible in handling topology variations, and the strategy is claimed to be valid for frames, trusses, plates and shells. Examples are given for a 10-bar planar truss and a 25-bar 3D transmission tower with stress and displacement constraints.

There are several examples in the literature of the use of evolutionary computing in the structural optimization of high-rise building frames. [Arciszewski et al., 1994] include general bracing system parameters in a demonstration of machine learning. [Murawski et al., 2000] used evolutionary algorithms to seek optimal designs for a 3-bay, 26-story building with design variables representing the connectivity of beams, columns and supports. [Kicinger et al., 2003] use an evolutionary method to generate structural configurations for tall buildings, assuming a regular spacing of stories and bays, six different types of diagonals, and two types of beams (rigid and hinged). [Kicinger, 2004] combines cellular automata and a GA to generate and optimize designs, observing emergent behaviour. [Baldock et al., 2005] applied a modified pattern search algorithm to optimize lengths of bracing spirals on a tall building. [Baldock and Shea, 2006] optimize the bracing configuration of a high rise building using genetic programming. In contrast to the work of [Soh and Yang, 2001], this approach evolves programs for generating designs, rather than simply evolving tree representations of designs. The genetic programming approach is demonstrated to generate a relatively wide variety of layouts, which include variation in the size of basic bracing units.

Many of the previous evolutionary computing methods are relatively problem-specific in nature. For example, trusses and high-rise buildings are generally assumed to have a fixed number of panels or bays, and the methods optimize bracing layouts or some basic overall shape parameters. More general discrete optimization techniques, similar to the ground structure approach of [Dorn et al., 1964], have seen less research than the problem-specific methods. [Hajela and Lee, 1995] developed a two-stage genetic algorithm for general topology design based on the ground structure approach. The method generates kinematically stable configurations in the first stage, and optimizes for response constraints (such as stress) in the second stage. Hajela and Lee used a fixed length bit string, which increases in size as the number of possible member connections increases, and as the resolution of the node positions is increased. To lessen the impact of node position encoding, the resolution is successively increased using a multistage approach [Lin and Hajela, 1993]. [Rajan, 1995] proposes an alternative ground structure method that applies a penalty term to unstable structures. [Leung and Nevill, 1994] used a 2D binary array, with the problem domain limited to bars of equal length. [Nakanishi and Nakagiri, 1996; 1997] used a similarly restrictive approach to 2D topology optimization of frames and panel structures.

The use of evolutionary computing with fully-connected ground structures has significant practical limitations. The major limitation is the difficulty in efficiently encoding a wide variety of different topological configurations. The underlying grid limits the topology to the fineness of the chosen grid, and coarse grids severely restrict the topological freedom. The optimum design is highly dependent on the choice of ground structure [Yang and Soh, 2002]. Given that fine grids are usually not possible, the ground structures become problem-dependent and different designers will choose different initial configurations. A number of researchers have proposed more general evolutionary topology optimization methods that do not make use of ground structures [e.g. Shrestha and Ghaboussi, 1998]. [Azid and Kwan, 1999] describe the complexity of developing an efficient encoding scheme that handles a range of topologies and produces structurally meaningful topologies after modification by mutation and crossover operations. Azid and Kwan assert that crossover should produce structures that bear some architectural similarity to their parents, even when the parents have much different complexity. [Azid and Kwan, 1999] describe a GA-inspired method with crossover and mutation algorithms that work directly with the structural configuration rather than an encoded representation. The initial population is constructed using a randomly generated pattern of nodes. Members are added to connect the nodes, guided by heuristics based on bar lengths, joint complexity, and bar colinearity. The approach is extended to 3D truss optimization in [Azid et al., 2002]. [Yang and Soh, 2002] developed a general approach to discrete topology optimization using genetic programming. Compared to the fixed length chromosomes used in most GA applications, genetic programming uses 2D variable length parse strings which have the ability to dynamically evolve and represent a broad range of different topologies.

2.2.3.2. Continuum Optimization

In the analysis of continua, such as plates, shells and solids, optimization is generally classified into sizing optimization, shape optimization and topology optimization. In sizing optimization, the design variables are the thickness of plate and shell elements. The computational methods for continuum sizing optimization are similar to those used for truss sizing. Shape optimization concerns the description of external and internal boundaries, which are parametric functions of the design variables. Topology optimization refers to the distribution or arrangement of material within a structure. Combined shape and topology optimization is called generalized shape optimization.

Much of the research in shape optimization relates to finite element analysis techniques, in particular the remeshing of models as the boundaries change. During optimization, sensitivity derivatives are calculated to guide the search direction. A critical aspect of shape optimization is distinguishing sensitivity to changes in the boundary as from errors related to the accuracy of the mesh. The development of robust automatic mesh generators, while not eliminating sensitivity errors, has improved the performance of shape optimization techniques. Shape optimization is more effective as a tool for refining existing designs rather than synthesizing new designs. The generation of design concepts is one of the strengths of topology optimization.

The goal of topology optimization is to find the optimal distribution of material within a general design domain, such as the shape of the external boundary, the number, size and shape of holes, and the overall connectivity. While shape optimization starts with a predefined parametric description of the boundaries of a structure, topology optimization starts with a more general set of structural requirements. The only inputs to the topology optimization are the applied loads, the possible support conditions, the volume of the structure to be constructed, and restrictions on where material should or should not be located. In shape optimization the design variables represent the domain through the parametric representation of boundaries. In topology optimization, the design domain is fixed, and the shape and size of the structure are defined by a set of distributed functions defined on the domain. These functions represent a parameterization of the stiffness tensor of the continuum.

The early development of topology optimization methods for continuum structures is closely linked with theoretical work in composite materials. In the *homogenization* approach [Bendsøe and Kikuchi, 1988], the design variable is the continuous density of the base material in the composites. In this representation, the composite material consists of an infinite number of infinitely small holes periodically distributed through the base material. The material is parameterized through the material density ρ, where $\rho=0$ corresponds to a void, $\rho=1$ to material, and $0<\rho<1$ to porous composites.

For continuum topology optimization problems involving isotropic material, a simpler approach called the *material distribution method* is available. In this formulation, each point in the design domain is either on structure or in a void. The geometry of the structure is defined by the subset of all the points in the domain that have material. On a domain discretized using finite elements, the structure can be viewed as a black and white image where the pixels of the image are the elements. In contrast, structures produced using the homogenization approach correspond to grayscale images, where the shade represents material density. The material distribution method is formulated as a distributed, discrete-value design problem, or a 0-1 problem.

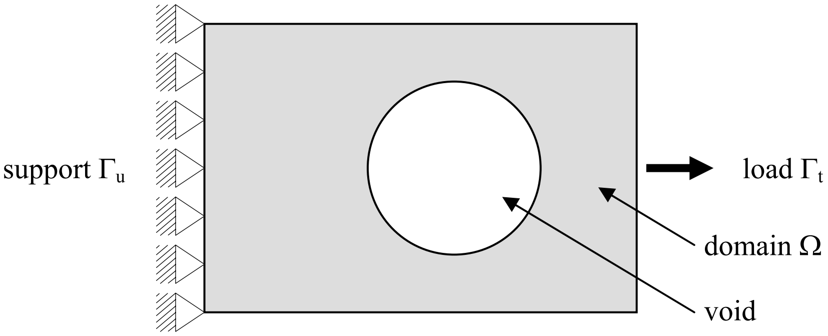


Figure 2.8. Continuum topology optimization problem

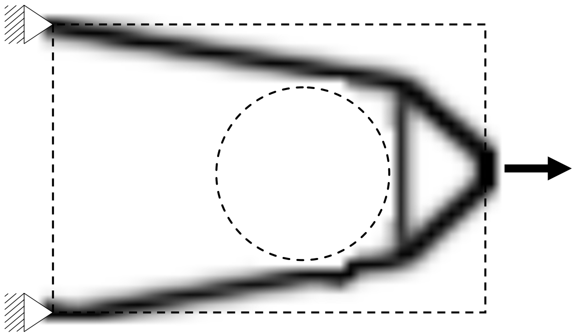


Figure 2.9. Continuum topology optimization output

The topology optimization problem is defined as the problem of finding the optimal choice of stiffness tensor $E_{ijkl}(x)$ which is a variable over the reference domain Ω (Figure 2.8). The goal is to determine the optimal subset Ω^{mat} of points consisting of material. This implies that the set of admissible stiffness tensors consists of those tensors for which

$$E_{ijkl} = 1_{\Omega^{mat}} E^0_{ijkl} \text{ and } 1_{\Omega^{mat}} = \begin{cases} 1 & \text{if } x \in \Omega^{mat} \\ 0 & \text{if } x \notin \Omega^{mat} \end{cases}. \qquad (2.1)$$

A common approach to this problem is to replace the integer variables with continuous variables and use a penalty function to steer the solution to discrete 0-1 values. The design variable then becomes a continuous function that is interpreted as the density of the material. The penalty function is used to avoid regions of intermediate “density”, since the stiffness is small compared to the cost or volume of material. An efficient and commonly-used interpolation method is found in the penalized, proportional stiffness model (SIMP) [Bendsøe, 1989]:

$$\begin{aligned} &E_{ijkl}(x) = \rho(x)^p E^0_{ijkl}, \quad p > 1, \\ &\int_\Omega \rho(x)\,\mathrm{d}\Omega \le V; \quad 0 \le \rho(x) \le 1, \quad x \in \Omega. \end{aligned} \qquad (2.2)$$

For $p > 1$, intermediate densities are treated as unfavourable. To obtain true black and white (binary) designs, $p \geq 3$ is usually required. With the current technology for producing advanced composite materials, intermediate densities are not necessarily uneconomical. For conventional construction techniques for the large scale structures considered in this work, however, binary designs are preferred. Because the interpolation function is continuous, intermediate zones between the existence and non-existence of material are still present, so the final useful geometry is usually obtained by filtering or interpretation using threshold values.

Compared to conventional structural optimization, topology problems have a large number of variables but relatively few constraints, so efficient computational methods are required. The two main classes for solving the material distribution problem are optimality criteria methods [e.g. Olhoff, 1970; Taylor and Rossow, 1977; Rozvany and Zhou, 1991] and mathematical programming techniques, such as CONLIN [Fleury, 1993] and the Method of Moving Asymptotes (MMA) [Svanberg, 1987]. Similar in nature to Sequential Linear Programming (SLP) and Sequential Quadratic Programming (SQP), CONLIN and MMA use separable and convex approximations to solve smooth, nonlinear optimization problems.

A typical result of topology optimization, obtained using optimality criteria methods, is shown in Figure 2.9. The constraints for this problem are to minimize compliance for a given volume fraction (or ratio of optimized volume to available volume). Figure 2.9 shows a characteristic result of the material distribution method; that low volume fractions result in truss-like structures. The material distribution method predicts similar forms to those obtained using classical analytical methods, such as the forms developed by Michell in the study of grid-like continua. For certain conditions, namely the minimum compliance constraint with a single load case, it has been shown that at low volume fractions, the optimal solution for plates in plane stress tends to that of least-weight trusses [e.g. Rozvany et al., 1985].

Two important complications arise in the use of the material distribution method for topology optimization. These are the mesh-dependency of results and the appearance of checkerboard patterns. Mesh-dependency refers to the effect that different finite element discretizations of the domain can produce qualitatively different results. In practical applications it is preferable to have a finer element mesh result in more clearly defined boundaries, rather than a qualitatively different structure. The mesh-dependency problem has been solved, and the methods used fall into three categories. These consist of adding constraints to the optimization, directly reducing the parameter space, or applying filters as part of the optimization procedure. The checkerboard problem refers to alternating solid and void elements arranged in a checkerboard-like pattern, and is related to the finite element discretization. The techniques for reducing mesh-dependency also decrease the likelihood of checkerboard problems. Higher order finite elements, additional constraints, and filters are other ways to eliminate the formation of checkerboard patterns.

Evolutionary computing methods have been applied to continuum optimization problems. Although not based on evolutionary computing principles, the so-called Evolutionary Structural Optimization (ESO) method [Xie and Stevens, 1992] is a heuristic method that involves the sequential removal of lightly-stress elements. Some researchers have shown that ESO may produce highly nonoptimal designs [Zhou and Rozvany, 2001]. A true evolutionary computing approach to continuum optimization was developed by [Sandgren et al., 1990] and [Jensen, 1992]. Their work was extended by [Chapman et al., 1994] to optimized finely-discretized design domains. [Liang et al., 1999] present a method, based on ESO, for developing minimum-weight topologies for continuum structures under stress constraints. [Mijar et al., 1998] and [Liang et al., 2000] use continuum topology optimization to evolve bracing systems for simple 2D multistory frames. [Bentley and Wakefield, 1996] use a GA with a primitive building block representation to form a system of clipped and stretched cuboids. [Griffiths and Miles, 2003] apply a GA to the problem of finding the optimal cross-section of a beam as a shape and discovery problem. More recently, more advanced representation schemes have been developed, including Voronoi-based representations [Periaux and Winter, 1995; Schoenauer, 1996] and representations based on fractal theory [Hamda et al., 2002].

2.2.3.3. Commercial Applications

The major commercial finite element analysis packages, such as ANSYS and MSC.Nastran include continuum topology optimization routines. For example, ANSYS uses a material distribution method based on energy methods [Mlejnek et al., 1993]. Specialized structural optimization programs include Quint OptiShape, Altair OptiStruct, and Vanderplaats Genesis. These software packages each have the capability to do topology, shape and sizing optimization. Some commercial optimization programs have interfaces to major CAD applications; for example, Altair OptiStruct can run topology optimization on Pro/Engineer models using their HyperShape/Pro product.

2.2.4. Postprocessing

The layouts resulting from structural optimization must be processed to put them in a form suitable for manufacturing. At a minimum, the layouts are manually converted into a CAD format for detailing. In combined topology and shape optimization, the rough boundaries produced by topology optimization are represented using smooth parametric descriptions, which are more suitable for manufacturing. In a further level of postprocessing, the topology optimization output may be used as the input for multiobjective optimization to include a range of manufacturing, assembly, cost and other criteria.

Design for Manufacturing (DFM) and Design for Assembly (DFA) methods provide a systematic procedure for analyzing a proposed design from manufacturing and assembly viewpoints [Boothroyd et al., 1994]. The goal of DFM and DFA is to produce simpler and more reliable products which are less expensive to fabricate. More specifically, some of the objectives are to maximize standardization (of materials, concepts, components, tools and fixtures), to simplify manufacturing, to enhance uniformity, to support parallelism (e.g. concurrent engineering and simultaneous manufacturing), and to minimize resource requirements. Modularity is a systematic approach to designing products that share interchangeable components. Modularity shortens the product development cycle by reusing existing components, and by enabling simultaneous work in design and manufacturing.

[Yetis and Saitou, 2002] proposed a systematic method for the decomposition of a complex structure obtained from structural topology optimization using image processing algorithms. Decomposition is framed as a graph partitioning problem, and solved using a genetic algorithm. The objective function considers strength reduction and assembleability criteria related to the number and similarity of welds. [Cetin and Saitou, 2004] extend this method to include modularity criteria, based on the similarity of the components.

[Chickermane and Gea, 1997] present a method for optimal layout topology of multicomponent structural systems connected by joints (or fasteners). The input consists of design domains for each component, an initial distribution of joints, and a target number of joints. Joints are represented using a microstructure-based model. Optimization results in the topology of each component, along with the optimal joint locations.

[Chirehdast et al., 1992; 1994] propose a four-phase design process called the Integrated Structural Optimization System (ISOS). In Phase I, an optimal initial topology is created as a gray-scale image using a homogenization method [Bendsøe and Kikuchi, 1988]. In Phase II, the image is transformed into a simpler parametric model using image processing techniques. In this phase, rules derived from elementary mechanics and engineering intuition are used to delete nodes and add or remove elements. For example, zero force truss elements are deleted, and elements are added to triangulate polygonal cells and ensure stability. Phase III implements truss geometry and sizing optimization using SAPOP [Bremicker et al., 1990; 1991]. In Phase IV the resulting design is either refined or manufactured.

2.3. Image Processing

Image processing is the manipulation and analysis of image data, and is used in many fields, including medicine, astronomy, microscopy, seismology, defense, industrial quality control, and aerial and satellite imagery. Although image processing may be broadly defined, in this context the term is used to refer to operations on digital images where the input and output are both in the form of images. The analysis of images to produce measurements or high level descriptions is discussed in the following section under the topic of *pattern recognition*. Image processing commonly involves 2D images, but is increasingly being used on 3D data sets, such as those generated using magnetic resonance imaging (MRI) in medical imaging. More generally, image processing is signal processing using 2D or 3D signals. Many of the algorithms applicable to image processing, such as filtering, were developed in the signal processing field.

2.3.1. General

Image processing is used to enhance images, using techniques such as brightness adjustment, contrast adjustment and noise filtering. A brightness histogram is a plot of the distribution of pixel brightness in an image. The histogram may be manipulated to adjust brightness and contrast. The histogram is used to select brightness threshold values to convert grayscale images into binary images [Russ, 1999].

2.3.2. Morphology

In biology, *morphology* deals with the form and structure of plants and animals. In the field of imaging, morphology is more specifically used to describe an extensive class of nonlinear image processing and analysis operations [Dougherty and Lotufo, 2003]. Mathematical morphology was developed by Matheron and Serra at the Ecole des Mines in Paris for analyzing the geometric structure of geological data and materials [Serra, 1982]. Mathematical morphology is based on Minkowski set theory and the theory of finite lattices.

An image is a mapping, I, from a set, S_p of pixel coordinates to a set, G, of values such that for every coordinate vector, $\boldsymbol{p}=(r,c)$ in S_p, there is a value $I(p)$ drawn from G. S_p is also called the image plane. A binary image has 2 values, thus $G=\{\nu_{fg}, \nu_{bg}\}$, where ν_{fg} is called the foreground value and ν_{bg} is called the background value. The foreground is the set of locations, $\boldsymbol{p}$, where $I(\boldsymbol{p}) = \nu_{fg}$;

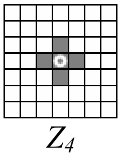
$$FG\{I\} = \left\{ I(p), p = (r,c) \in S_p \middle| I(p) = \nu_{fg} \right\}. \tag{2.3}$$

Similarly, the background is

$$BG\{I\} = \left\{ I(p), p = (r,c) \in S_p \middle| I(p) = \nu_{bg} \right\}. \tag{2.4}$$

A structuring element (SE) is a small image whose foreground identifies neighbours to a given point in the image plane. Figure 2.10 shows an example of two structuring elements. In this figure the origin of the SE is marked with a circle, and the foreground of the element is indicated with gray pixels.

The *translate* of a structuring element Z to a location $\boldsymbol{p}$ in S_p is denoted $Z+\boldsymbol{p}$, and refers to a position of Z where the origin of Z coincides with location $\boldsymbol{p}$ in S_p. The *Z-neighbourhood* of $\boldsymbol{p}$ in I, $N\{I,Z\}$, refers to the set of locations in the image delineated by $Z+\boldsymbol{p}$.



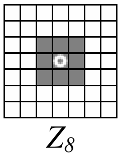


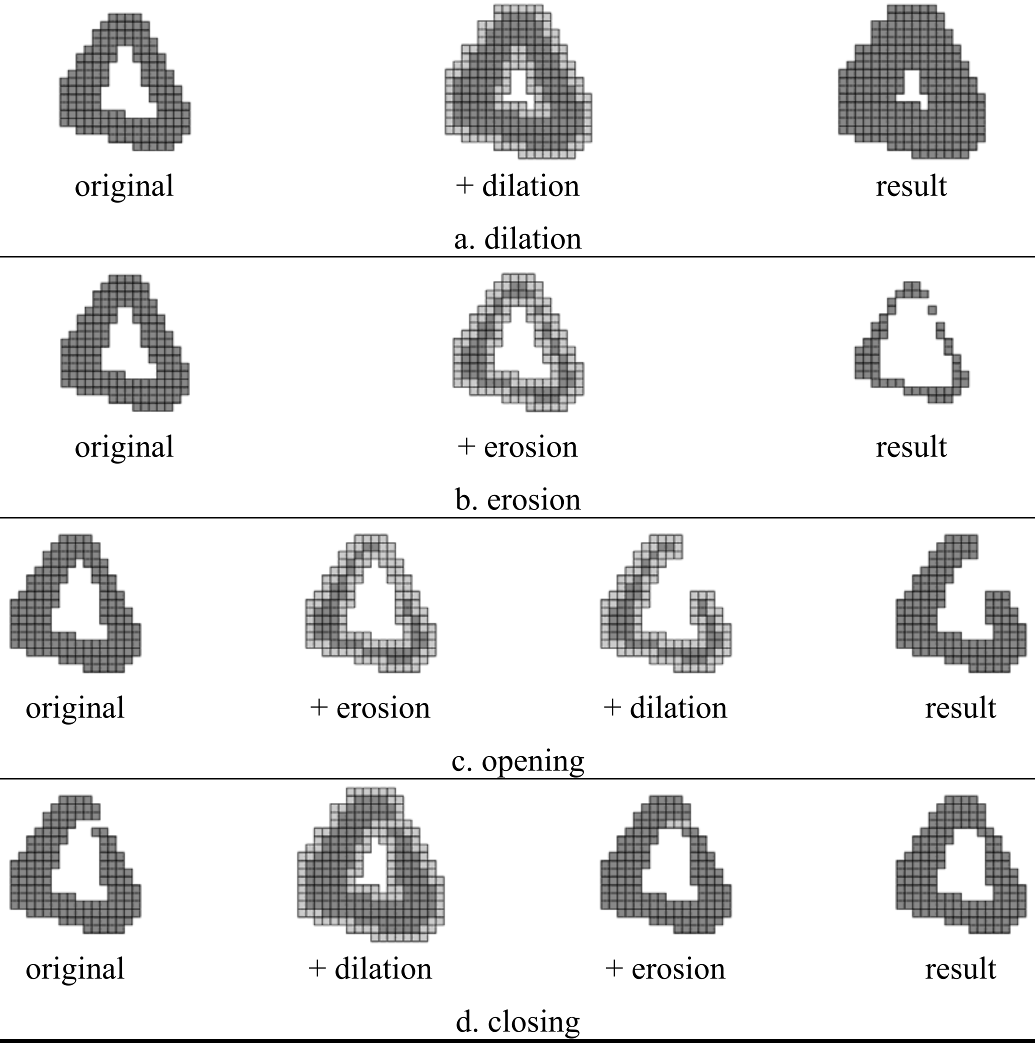
Figure 2.10. Structuring elements

Dilation is an operation that effectively enlarges the foreground by adding pixels to the perimeter. A definition of dilation (Figure 2.11.a) is

$$I \oplus Z = \bigcup_{p \in FG\{Z\}} \left(I + p\right).$$

Erosion is an operation that effectively reduces the size of the foreground by removing pixels at the perimeter. A definition of erosion (Figure 2.11.b) is

$$I \ominus Z = \left\{p \in S_p \middle| Z + p \subset I\right\}.$$



a. dilation

b. erosion

c. opening

d. closing

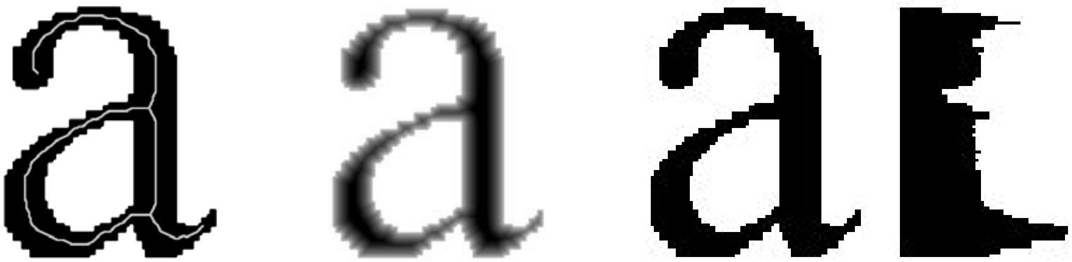
Figure 2.11. Morphological operations (SE=Z_8)

Opening is defined as an erosion followed by a dilation (Figure 2.11.c). *Closing* is defined as a dilation followed by a erosion (Figure 2.11.d). Opening and closing operations are used extensively in image processing to filter noise, smooth edges, fill holes, and to separate or connect multiple shapes.

2.3.3. Thinning

A recurring problem in image processing and analysis is to simplify images while retaining their essential geometric information. *Thinning* is a technique that reduces a pattern to a thin-line representation, and is used to facilitate pattern recognition and provide data compression [Blum, 1967]. Thinning is used in some automatic character recognition systems to simplify characters prior to processing. A commonly used thinning procedure is s*keletonization*, which is based on the concept of maximal disks. Given a point interior to a binary image, there exists a largest disk, the maximal disk, having the point at its center and also lying within the image. The centers of all maximal disks comprise the *skeleton*, or *medial axis* of the image [Blum, 1973]. An example of a skeleton is shown in Figure 2.12.a.

Because of the value of thinning in pattern recognition and data compression, thinning algorithms have been studied extensively [Lam et al., 1992]. Thinning algorithms can be categorized according to whether they use the definition of medial axis given above. As defined, skeletons tend to have numerous spurious branches because of boundary irregularities. Also, the skeletonization process tends to disconnect connected sets. This effect is similar to that shown in Figure 2.11.b, where erosion causes continuous foreground objects to break into separate objects. If the structuring element is large enough or if erosion is repeated, the foreground objects may disappear completely. As an alternative to computing the medial axis, morphological "hit-or-miss" operators can ensure that no foreground pixels are changed where that change would create a disjointed object [Zhang, 1997].



a. skeleton b. distance map c. projection

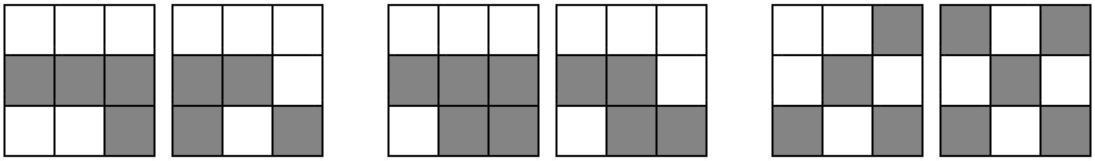
Figure 2.12. Image processing operations

2.3.4. Distance Map

The Euclidean distance map (EDM) produces a grayscale image from a binary image [Danielsson, 1980]. The brightness of a pixel in the grayscale image represents the distance the pixel lies from the nearest boundary. Figure 2.12.b shows the EDM of the image of a character, calculated using the algorithm of Borgefors [Borgefors, 1986]. The darkest points on the EDM plot define a ridge that is equidistant from at least 2 points on the boundary of the feature. This ridge is the medial axis, and can be extracted from the EDM by applying an edge detection method, such as the Laplacian operator.

2.3.5. Hit-or-Miss Operators

Although it is possible to isolate the medial axis from the grayscale distance map, this approach is more complex than using a morphological hit-or-miss algorithm like the one used to generate the skeleton in Figure 2.12.a. The hit-or-miss transform is a general binary morphological operator that can be used to identify particular patterns of foreground and background pixels in an image. The structuring element used in the hit-or-miss transform is similar to those used in erosion and dilation, except that background pixels are used as well as foreground pixels. The hit-or-miss operation is performed by translating the origin of the structuring element to all points in the image, and then comparing the structuring element with the underlying image pixels. Figure 2.13 contains several examples of structuring elements used in hit-or-miss operations. Figure 2.13.a indicates conditions where the central pixel should be retained in a thinning operation. Deleting the central pixel would break up the image, and lead to further disintegration with subsequent passes of a thinning algorithm. Figure 2.13.b identifies conditions under which the central pixel could be deleted without affecting the connectivity of the foreground (assuming 8-connectivity, or that pixels sharing a corner constitutes connectivity). Figure 2.13.c shows how structuring elements may be used to identify intersection points between line-like foreground features.



a. retain during thinning b. remove during thinning c. node identification

Figure 2.13. Hit-or-miss structuring elements

2.3.6. Projection

The *projection* of a 3D object of optical transparency *V*(*x*,*y*,*z*) to an image *f*(*x*,*y*) is defined by

$$f(x,y) = \int V(x,y,z)\,\mathrm{d}\,z \qquad (2.5)$$

where the integration takes place over the volume of *V*, and the object is illuminated along the *z* axis. Radiography (x-ray imaging) is an example of projection. The projection of a planar image f(x,y) corresponds to illuminations of objects along a cross-section, and is given in cartesian coordinates by

$$p(x) = \int f(x, y)\,\mathrm{d}\,y\,. \tag{2.6}$$

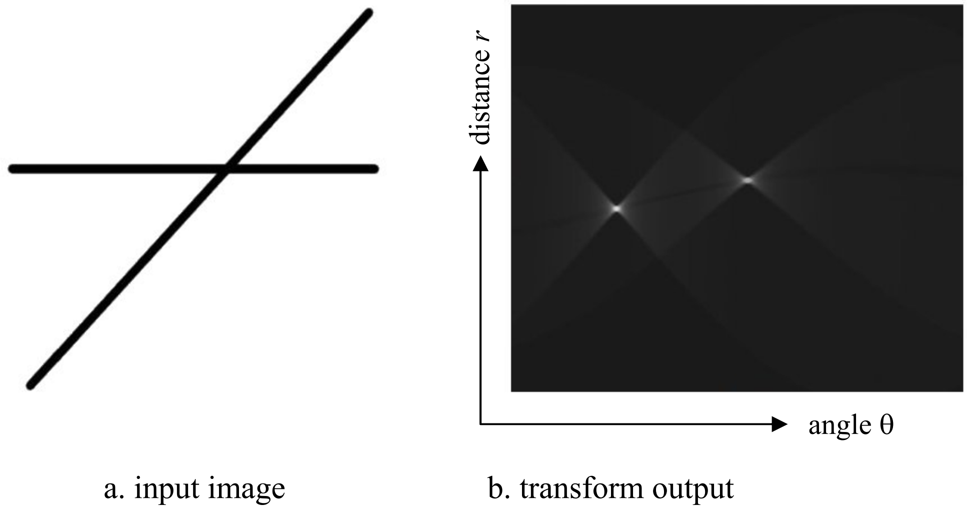
Projections describe the intersection length of a series of section lines, and transform plane regions into plane curves. By intersecting an object in different directions, strong angular features in the object appear as peaks in the projection. An example of cartesian projection is shown in Figure 2.12.c.

2.3.7. Hough Transform

The Hough transform is a method used to detect imperfect instances of objects in images [Hough, 1959; Duda and Hart, 1972]. In its initial form, the Hough transform was used to identify lines, but it has since been extended to detect general parametric shapes such as circles and ellipses. The transform converts an input image into *n*-dimensional accumulator space, where *n* is the number of parameters used to describe a class of object. For line detection, 2 parameters are used to describe a line so the output is a 2D image of the accumulator space (Figure 2.14). A line may be represented by the parameters of slope *m* and intercept *b* as *y*=*mx*+*b*. An alternative form which is computationally simpler is:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right), \tag{2.7}$$

where *r* is the distance between the line and the origin, and θ is the angle of the vector from the origin to the closest point on the line.



a. input image b. transform output

Figure 2.14. Hough transform

Each line in the image is associated with a unique couple (r,θ) in accumulator, or Hough space. At each point (x_o,y_o) in the image plane, an infinite number of lines with varying slope may exist. In accumulator space, the lines have the equation

$$r(\theta) = x_o \cos\theta + y_o \sin\theta\,, \tag{2.8}$$

which defines a sinusoid unique to each point (x_o,y_o). If two points in the image space lie on the same line, the line is defined by the intersection point of the corresponding sinusoids in accumulator space. To identify imperfect lines, the image is processed pixel by pixel. If a pixel belongs to the foreground, that pixel "votes" in accumulator space for the set of potential lines passing through that point. To limit the number of potential lines, the accumulator space is discretized into bins. Dominant lines in image space appear as local maxima in accumulator space, as shown by the bright points in Figure 2.14.

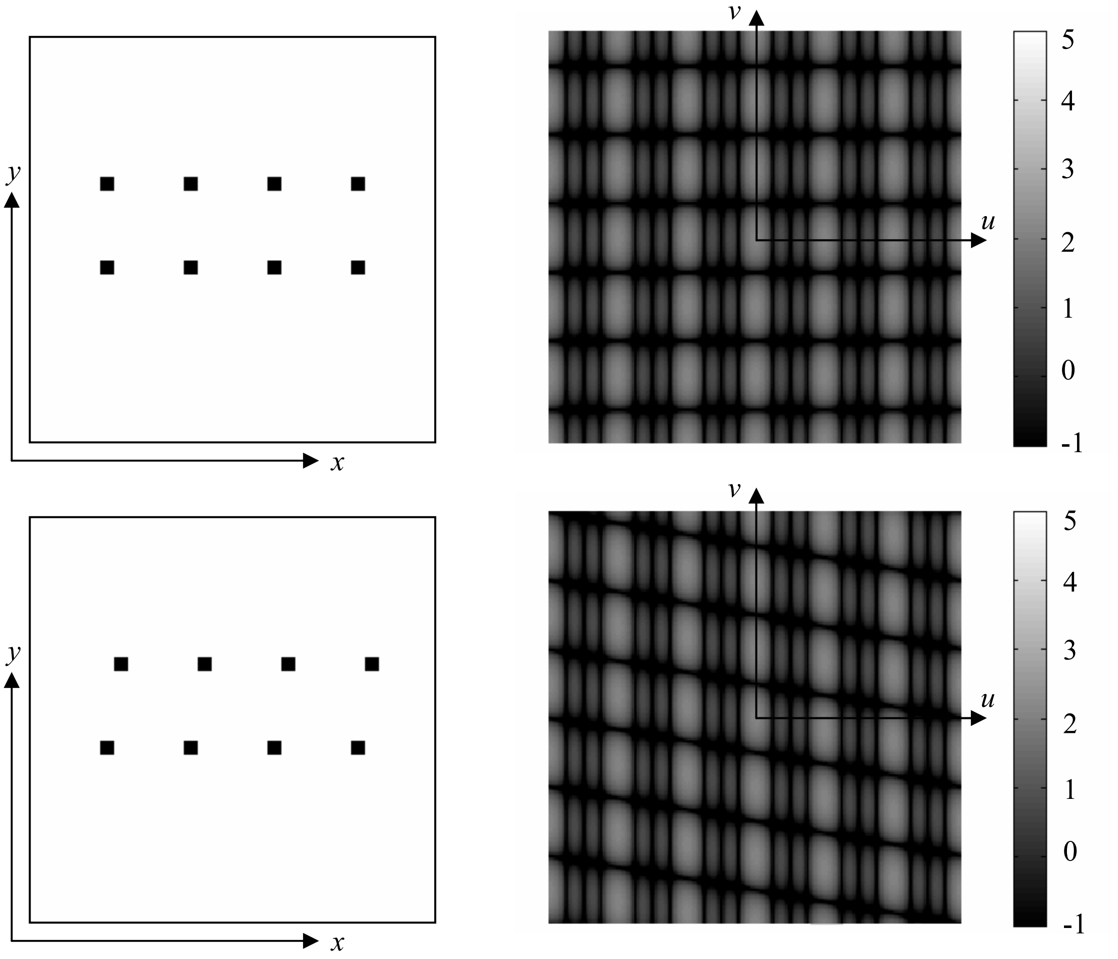


Figure 2.15. Fourier transform

2.3.8. Fourier Transform

The Fourier transform is used to decompose an image into sinusoidal components, providing insight into the overall periodicity and directionality of the image. The Discrete Fourier Transform (DFT) of a discrete 2D spatial domain image *f*(*x*,*y*) of width *M* and height *N* is

$$F(u,v) = \frac{1}{M}\frac{1}{N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) e^{-2\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)} ; \begin{matrix} u = 0,\dots,M-1 \\ v = 0,\dots,N-1 \end{matrix}, \qquad (2.9)$$

where the values *F*(*u*,*v*) are the DFT coefficients of *f*(*x*,*y*). The transform produces a complex number valued image of size *M* x *N*, where *M* and *N* correspond to the number of discrete frequencies in the *x* and *y* directions. The Fourier transform thus represents a transformation from the spatial domain to the frequency domain.

The output from the Fourier transform can be displayed as a pair of images, either as the real and imaginary parts, or as the magnitude and phase components. In image processing, often only the magnitude is displayed, as it contains most of the information on the geometric structure of the spatial domain image.

In Figure 2.15, the Fourier transforms of two spatial images are shown. The output is plotted on a logarithmic scale, with the center of the plot representing zero frequency (DC) components. The Fourier coefficients capture the periodicity of the input image in different directions, and are commonly used in pattern recognition schemes.

2.4. Pattern Recognition

Pattern recognition techniques seek to describe and classify data from a wide range of sources. Often the data are 2D images: aerial and satellite images, radiograph and computed tomography (x-ray and CAT scan) images, scanned books or handwritten letters, microscopic images of amoebas or crystals. One dimensional waveforms such as seismographs and electrocardiogram traces, and 3D images from magnetic resonance imaging are other data sources. Data are described and classified using a set of reference patterns, which are either predefined or derived from similarities within the data. Pattern classes may be established *a priori* by experts, usually with a training set, during supervised learning. Alternatively, classes are based on statistical regularities in large sets of data, using unsupervised learning.

2.4.1. Overview

A complete pattern recognition system has the following components: a *sensor* to collect data, a *feature* extractor to identify numeric or symbolic features in the data, and a *description* or *classification* scheme based on a set of patterns. Image processing techniques are frequently used to remove noise from sensor data or enhance the raw data in other ways before features are extracted. There are two main approaches to pattern recognition. The classical approach follows a "Gestalt" view where the raw data is represented by an array of numbers representing various measurements performed on objects. If the numbers are the coordinates of points in space, then points which are geometrically close to each other represent similar objects. The problem of pattern recognition is to identify the regions in space where points from a single pattern lie. Ideally, the points corresponding to a pattern are tightly clustered and distinct from other patterns. Often, techniques from probability, statistics and decision theory are used, so this method is called *statistical pattern recognition* or *decision theoretic pattern recognition*. A limitation of the statistical method is the difficulty in selecting an effective set of measurements, particularly when patterns are complex or there a large number of pattern classes. *Syntactic pattern recognition* or *structural pattern recognition* is based on the concept that a complex pattern could be described in terms of simpler patterns. Similarly, complex phrases can be broken down into words, and letters of the alphabet can be described using strokes. Many techniques from the study of formal languages are used in syntactic pattern recognition, which is also known as *linguistic pattern recognition*.

Much of pattern recognition is dedicated to finding a compact representation of data. Image data is particularly dense and direct comparison and classification of images is computationally expensive. Pattern recognition uses mathematical techniques to compactly represent image data and support symbolic reasoning with images. Patterns are represented as feature vectors, strings, or relational graphs. In the classical decision theoretic approach, each pattern is represented by an *n*-dimensional feature vector, and patterns are recognized by applying techniques in discriminant analysis and statistical decision theory. For complex patterns, the number of features *n* required for recognition becomes large, and the decision theoretic technique may become ineffective. In this case, complex patterns can be represented by simpler subpatterns, where decision theoretic methods are used on the subpatterns [e.g. Pavlidis, 1977]. In syntactic pattern recognition, the relationship between the subpatterns is encoded using strings or graphs.

2.4.2. Regions

Images may be analyzed using global or local analysis methods. Global methods include the Fourier transform and the colour histogram. The Fourier transform produces a sparse matrix representation of an image, which can be used as elements of a feature vector. Global analysis is limited in the ability to capture local features of an image, which are often important in discriminating between images.

Local analysis methods isolate uniform regions or features in an image which can be used with syntactic pattern recognition methods. A common method in pattern recognition is to identify homogeneous regions of texture and colour, a process called *segmentation*. For example, texture captures the granularity or repetitive patterns of surfaces within an image. In satellite imaging, differences in texture are used to distinguish water from grassland, and woodlands from urban areas [Haralick et al., 1973]. Regions of similar colour or intensity are identified using *thresholding*. Once images are segmented into regions, those regions can be characterized by a number of different methods. Regions can be described by scalar quantities, such as the ratio of the perimeter squared to the area, or by vector quantities, such as centroids. Formulations exist for moments that are invariant with respect to scale, position and rotation [Hu, 1962; Wood, 1996]. Scalars and vector quantities are incorporated into the feature vector. Regions are also described by projections, or by the analytical representation of their boundaries or skeletons.

Cartesian projections have been used to describe the shape of typewritten letters [Pavlidis, 1977] and polar projections through a common point have been used in the recognition of chromosomes [Rutovitz, 1970]. Discrete projections may be used directly as a feature vector for describing the shape of regions. They may be also processed syntactically to extract key features from the region.

2.4.3. Boundaries and Curves

Region boundaries, skeletons, and curves in general are compactly represented using a range of different curve fitting techniques. Although higher order polynomials and 1D Fourier transforms can be used to fit curves, these are global techniques that may miss important local features. A more effective method in pattern recognition is to use a linear piecewise continuous approximation or spline. A number of different algorithms are available for piecewise curve fitting, many of which use the techniques of iterative splitting and merging. In splitting, the input is repeatedly subdivided into smaller segments until the approximation error is below a specified threshold. In merging, segments are iteratively joined until no further segments can be joined without exceeding an error limit.

The Hough transform may be used to extract parametric features from an image, such as lines, circles and ellipses. In the content-based retrieval of line drawings, the angles as well as locations of strong linear features are stored in a feature vector [e.g., Franti et al., 2000].

2.4.4. Structural Descriptions

Syntactic or structural descriptions produce compact representations of shapes without using high level, or semantic information. They deal strictly with local features and the relationships between them. The earliest examples are the syntactic descriptions of Ledley [Ledley, 1964], and those based on the Freeman chain code [Freeman, 1961]. The earliest application of Freeman chain codes was to the encoding of line drawings, as shown in Figure 2.16. Assuming a rectangular grid, eight basic directions are defined. A line superimposed on the grid is represented by a series of moves from one square to another; thus the line in Figure 2.16 is represented by the string of numbers 44445707670.

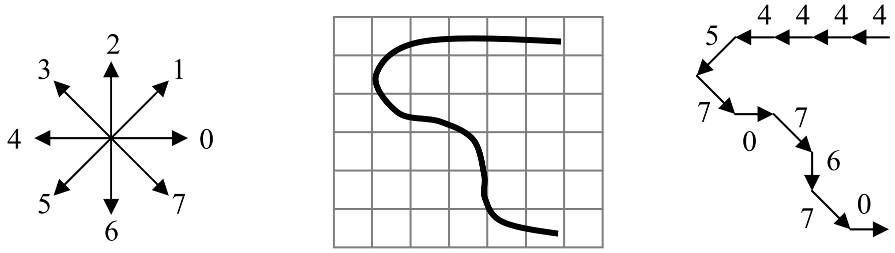
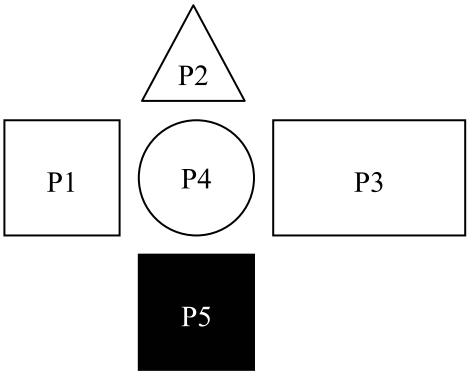


Figure 2.16. Freeman chain code [after Pavlidis, 1977]

In general, the encoding of a shape into a string with symbols from a fixed "alphabet" has the advantages of fast algorithms, compact storage, and well-developed methods such as the theory of formal languages. The limitations of chain code are that they are not rotationally invariant, and become complex when describing global features like the closing of boundaries. In a slightly higher level of encoding than Freeman, Ledley uses a finite set of symbols to represent arcs, where for example, "a" is a convex arc of high curvature and "b" is a straight line [Ledley, 1964].



Dp	=	{*P*, *RP*}
P	=	{*P1*, *P2*, *P3*, *P4*, *P5*}
PR	=	{(*Left*, *Left_P*), (*Above*, *Above_P*)}
Left_P	=	{(*P1*, *P4*), (*P4*, *P3*)}
Above_P	=	{(*P2*, *P4*), (*P4*, *P5*)}
P1	=	{(*shape*, *rectangular*), (*colour*, *white*)}
P2	=	{(*shape*, *triangular*)}
P3	=	{(*shape*, *rectangular*)}
P4	=	{(*shape*, *circular*)}
P5	=	{(*colour*, *black*)}

Figure 2.17. Structural description example

In a generalization of the above methods, a *structural description D* of an object is formulated as a pair $D = (P, R)$, where $P = \{P_1, \dots, P_n\}$ is a set of primitives [Shapiro and

Haralick, 1981]. Each primitive P_i is a binary relation $P_i \subseteq A \times V$, where A is a set of possible attributes and V is a set of possible values. $R = \{PR_1, ... PR_K\}$ is a set of named N-ary relations over P. For each $k = 1, ... , K$, PR_k is a pair (NR_k, R_k) where NR_k is a name for relation R_k, and for some positive integer M_k, $R_k \subseteq P^{Mk}$. Thus, set P represents the parts of an object, and set R represents the relationships between the parts. The elements of any relation R_k may include as components primitives, attributes, values and any symbols necessary to specify the given relationship. An example of a structural description using the above syntax is shown in Figure 2.17.

2.4.5. Relational Graphs

The structural description may be realized as an *attributed relational graph* (ARG) [Eshera and Fu, 1986; Foggia et al., 1999]. The ARG is defined as $G = (V, E, A_V, A_E, \alpha_V, \alpha_E)$ where V and E are respectively the sets of the vertices and the edges of the ARG; A_V and A_E are the sets of the vertex and edge attributes, and α_V and α_E are the functions associating to each vertex or edge the corresponding attributes. The attributes of a node or edge have the form $t(p_1, ..., P_{kt})$, where t is a type chosen over a finite alphabet T, and $(p_1, ..., p_{kt})$ are a tuple of parameters, also from finite sets $P_1^{'}, ..., P_{kt}^{'}$. Both the number of parameters k_t and the sets they belong to depend on the type of attribute, and for some type k_t may be equal to 0 (the attribute has no parameters). The type information discriminates amoung different kinds of nodes or edges, and the parameters characterize the nodes or edges of a given type. Usually the nodes of a graph represent the primitives of the structural descriptions, and the edges represent the relations between the primitives. The graph definition is often abbreviated as G=(V,E). The numbers of vertices and edges have been expressed as |V| and |E| in the literature.

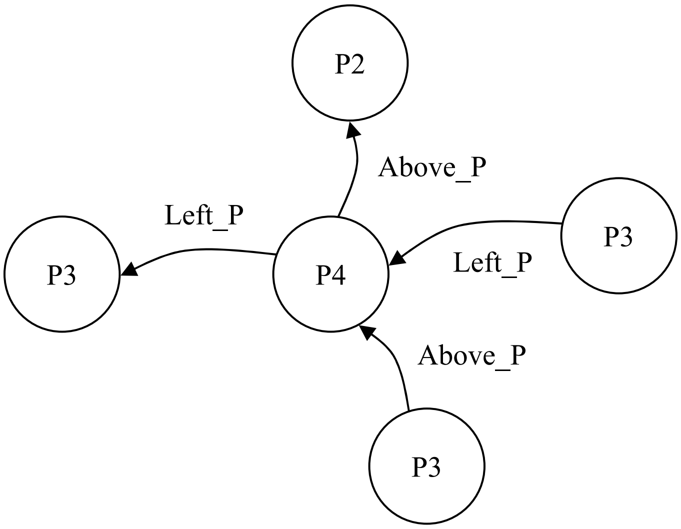


Figure 2.18. Attributed relational graph

Relational attributes such as *Left* and *Above*, shown in Figure 2.18, are frequently used in pattern recognition. Since the attributes have no associated numerical values, only binary values, their expressive power is limited. If additional relations such as *Above_Left* and *Touches* are added, understanding similarities between objects becomes more difficult. Alternatively, spatial relationships between primitives may be represented by a single character hexadecimal direction code in the range [0, ..., f], which allows for a compatibility function that can discern similarity by numerical difference [Blake, 1994]. This definition still does not express the spatial distance between primitives. [Thoresen, 2007] proposes the use of an n-dimensional vector $\boldsymbol{d}$ as edge attribute, where n is the number of spatial dimensions in the image. In a computer vision application where the vertices represent regions, $\boldsymbol{d} = \boldsymbol{c}_2 - \boldsymbol{c}_1$, where $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are the centroids of the regions.

2.4.6. Output

Given an input image, the goal of pattern recognition is generally to describe or classify the image. A *description* in this context usually means a natural language representation in text, or another language that can be easily converted to such a form. A goal in machine vision, for example, is to describe all the relevant information in a scene, including the types of objects in the scene and the spatial relationships between those objects. The goal of *classification* is to assign an image, or parts of it, to a class. Classes may be defined by a formal description, or more commonly, by a set of example images from each class. Classification is closely linked to inference and learning. Where no formal definition of a class exists, it is necessary to infer the definition from the examples. Related to classification, *clustering* is the process of determining similarity between images.

Using pattern recognition techniques, images may be represented in a number of different formats, including feature vectors, character strings, and relational graphs. The ease with which images support description, classification, learning and clustering strongly depends on the representation format.

Classification using feature vectors is a well developed subject in pattern recognition. Statistical pattern classification determines the probability $P(c|x)$ that an object represented by a vector x belongs to class c. This probability may be estimated using Bayes' theorem, given that a set of objects with known classification is available. Similarity between feature vectors is readily assessed using a nearest neighbour algorithm, often just the Euclidean distance between vectors. The description of an image using feature vectors is more problematic than classification or clustering, because they often lack the ability to represent structural relationships.

Character string representations of images support both classification and description, provided the string elements come from a finite set. In this case, the theory of formal languages can be used for pattern recognition. Inference using strings is more difficult, and clustering is often limited by a lack of meaningful similarity measures.

For graphs, the classification, description, learning and clustering problems are generally more difficult than for vectors or strings. Graphs may be classified by first constructing a feature vector from the graph. The vector may contain graph theoretic properties, such as the maximum degree of nodes, connectivity and number of branches, or graph label information. To generate descriptions from graphs, graph languages and grammars have been developed. For classification, methods used for strings may be applied. Similarity measures for graphs have been widely studied; this is the graph matching problem.

2.4.7. Graph Matching

Since the early 1970s when relational graphs were first used to represent 2D scenes in machine vision [Barrow and Popplestone, 1971], there has been a strong interest in developing practical methods to determine the similarity between graphs. An important property of graphs in pattern recognition is that by definition they are invariant with respect to transformation, rotation and scaling[2]. The *graph isomorphism* problem is to determine whether two graphs are the same. Another, more common problem is to determine whether one graph is part of another graph. This is the *subgraph isomorphism* problem, where a subgraph is obtained from a graph by deleting vertices. The subgraph isomorphism problem is stated as follows [Cook, 1971]:

> *Given graphs $G=(V_1,E_1)$, $H=(V_2,E_2)$,*
> *does G contain a subgraph isomorphic to H, i.e. a subset $V \subseteq V_1$ and a subset $E \subseteq E_1$ such that $|V|=|V_2|$, $|E|=|E_2|$, and there exists a one-to-one function $f{:}V2 \rightarrow V$ satisfying $\{u,v\} \in E_2$ if and only if $\{f(u),f(v)\} \in E$?*

The subgraph isomorphism problem is computationally demanding because of its combinatorial nature, and has been proven to be NP-complete [Garey and Johnson, 1979]. Using a brute-force matching algorithm, the computing time increases exponentially with the size of the graphs, restricting graph-based techniques to graphs with a relatively small number of vertices and edges. A procedure that significantly reduces is the size of the search space is the backtracking algorithm of [Ullmann, 1976], still commonly used today. To reduce the complexity, researchers have imposed topological restrictions to planar graphs [Hopcroft and Wong, 1974] or trees, or have employed pruning methods [Corneil and Gotlieb, 1970].

Where solutions to the graph isomorphism and subgraph isomorphism problems do not exist, the *maximum common subgraph* gives a measure of similarity between graphs. A common subgraph of two graphs G_1 and G_2 is a graph G_3 such that there are subgraph isomorphisms from G_3 to G_1, and from G_3 to G_2. A common subgraph G_3 of G_1 and G_2 is called a maximum common subgraph if there is no other common subgraph of G_1 and G_2

[2] Invariance is not always desirable; in character recognition, the characters '6' and '9' are similar under rotational invariance. Note if vertex or edge attributes contain absolute position information, then the graph is not invariant to translation, scale and rotation.

that has more vertices than G_3. Note that the maximum common subgraph is usually not unique.

In practical pattern recognition applications, noise and error are often present. The methods discussed up to this point describe *exact graph matching* techniques. *Inexact graph matching* is concerned with determining the similarity between graphs where exact matches do not exist. Most of the classical methods for error-tolerant graph matching [e.g. Shapiro and Haralick, 1981], are variations of the *A** search procedure, a tree search incorporating heuristic lookahead. The similarity between two graphs is often expressed as a distance measurement [e.g. Eshera and Fu, 1984]; the smaller the distance, the more similar the graphs. The distance between graphs is commonly defined as the minimum cost of a sequence of operations that transforms one graph to another. The *minimum cost subgraph isomorphism* problem may be formulated as:

> *Given graphs $G=(V_1,E_1)$, $H=(V_2,E_2)$ where $|V_1|<=|V_2|$, a vertex cost metric $E_v(v_i,v_j)$ for associating a vertex $v_i \in V_1$ to a vertex $v_j \in V_2$, and an edge cost metric $E_e(e_k,e_l)$ for associating an edge $e_k \in E_1$ to an edge $e_l \in E_2$, what is the minimum cost subgraph isomorphism from graph G to graph H?*

In an alternative formulation, graph edit distances [Bunke, 1998] measures similarities through a series of graph edit operations. Typical operations are the insertion, deletion and substitution of vertices and edges. Bunke shows that graph edit distance computation is equivalent to solving the maximum common subgraph problem.

Morphological graph matching and elastic graph matching are other inexact matching techniques. Elastic graph matching takes into account the potential deformation of objects during recognition; in a two step process objects are first matched to a rigid grid, and the grid is then deformed using operations such as rotation and scaling. This method has been used for identification and tracking of cyclones [Lee and Liu, 1999]. In morphological graph matching, hyperplanes or deformable spline-based models are applied to the skeletons of non-rigid discrete objects [di Ruberto and Dempster, 2001]. Morphological graph matching has been applied to shape recognition from large image libraries [Huet and Hancock, 1999].

Graph matching is an inherently intractable problem, and there are many different approaches in the literature that attempt to reduce the complexity. Approximate methods can reduce the complexity in most cases from exponential to polynomial, however they do not guarantee an optimal solution. Graph matching methods have used heuristics, probability-theory based approaches, fuzzy set theory, genetic algorithms, neural networks, decision trees, clustering techniques and constraint satisfaction methods, amoung others.

Probability theory has been applied by many researchers. A general review on probabilistic graph matching can be found in [Farmer, 1999]. One of the first uses of probability theory in graph matching was in an iterative approach called probabilistic relaxation [Hancock and Kittler, 1990]. Fuzzy set theory has been used to represent distance between objects in images [Bloch, 1999]. [Messmer and Bunke, 1991] applied decision trees to the computation of error-correcting graph isomorphisms. Neural networks were used in a face authentication system with deformable graphs [Duc et al., 1999]. [Fan et al., 1998] employ clustering techniques in the automatic recognition of form documents. [Cross, Wilson and Hancock, 1996] describe a framework for performing relational graph matching using genetic search with Bayesian consistency measures. Graph matching is formulated as a constraint satisfaction problem in [Yaner and Goel, 2003].

2.5. Structural Stability

Structural stability is a term often used to describe the essential characteristic that separates structures from mechanisms. In this boundary region between the traditional disciplines of structural and mechanical engineering, several other terms are used to describe the same characteristic, including *mobility*, *rigidity*, and *kinematic determinacy*. Stability in this context must be distinguished from the more common usage of stability in structural engineering, in the context of the buckling of compressive elements. While the distinction between a structure and a mechanism is often obvious to the human designer, such a distinction is not readily made by the computer. In developing a computational system to assist conceptual structural design, the notion of stability must be studied more closely. This section outlines several different approaches to stability, including Maxwell's rule, linear structural analysis, and mathematical rigidity theory.

From the beginning, a single geometric shape has been central to the study of structural stability: the triangle. In a mechanics textbook from 1868, the following guidance was given to ensure the rigidity of a truss:

> *The word truss is applied in carpentry and iron framing to a triangular frame, and to a polygonal frame to which rigidity is given by staying and bracing, so that its figure shall be incapable of alteration by turning of the bars about their joints. If each joint were [...] like a hinge, incapable of offering any resistance to alteration of the relative angular position of the bars connected by it, it would be necessary, in order to fulfill the condition of rigidity, that every polygonal frame should be divided by the lines of resistance of stays and braces into triangles and other polygons so arranged, that every polygon of four or more sides should be surrounded by triangles on all but two sides and the included angle at farthest. For every unstayed polygon of four sides or more, with flexible joints, it is flexible, unless all the angles except one be fixed by being connected with triangles.*
>
> *[Macquorn Rankine and Millar, 1868]*

Such geometric rules for stability are still widely used, and can be exploited during conceptual design. In addition, several mathematical approaches to stability are available.

2.5.1. Maxwell's Rule

In 1864, James Clerk Maxwell published an algebraic rule specifying a condition for a pin-jointed frame composed of b rigid bars and j joints to be both statically and kinematically determinate[3]. The number of bars needed to stiffen a two-dimensional frame free to translate and rotate on a plane as a rigid body is given by

$$b = 2j - 3 \tag{2.10}$$

The physical reasoning behind the rule is that each added bar links two joints and removes at most one internal degree of freedom. The rule just equates the number of external and internal degrees of freedom. Maxwell's rule for three dimensions can be more generally written as

$$b = 3j - c \tag{2.11}$$

where c is the number of kinematic constraints ($c \geq 6$ in three dimensions) [Maxwell, 1864]. As Maxwell noted, this equation is a necessary, but not a sufficient condition for establishing determinacy. As an example, both structures shown in Figure 2.19 comply with Maxwell's rule, but the one on the left is obviously stable while the right one contains a mechanism. Maxwell also observed there were special assemblies, with fewer than $2j$-3 bars, that tighten up as its mechanisms are mobilized, warning that the stiffness of these assemblies is "of an inferior order" and that "a small disturbing force may produce a displacement infinite in comparison to itself" [Calladine and Pellegrino, 1991].

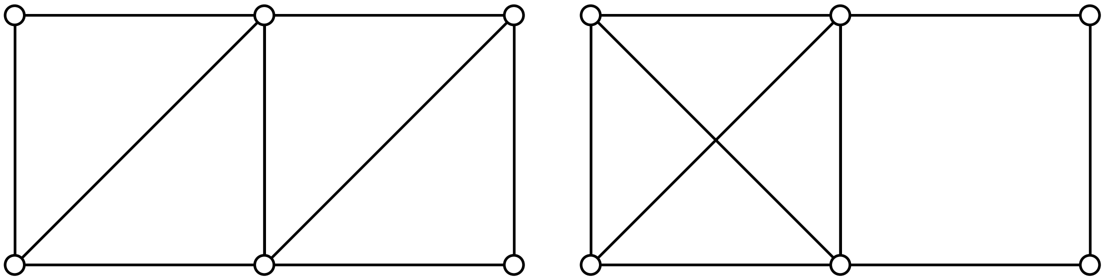


Figure 2.19. Limitation of Maxwell's rule

2.5.2. Gruebler's Equation

In a well-known kinematics approach, Gruebler's equation [Gruebler, 1885] defines the degree of freedom, or mobility M, of a planar assembly as

[3] Although the rule commonly bears Maxwell's name, this algebraic relationship was known at least as early as 1837 (Möbius, A.F., Lehrbuch der Statik, Vol. 2, Leipzig, Göschen).

$$M = 3(L\text{-}1) - 2J \tag{2.12}$$

for an assembly with L rigid links and J joints. If M is positive, the assembly is a mechanism, and if not, the assembly is a structure. Although a degree of freedom equal to or less than zero is necessary for immobility of the structure, this does not guarantee the structure is not mobile. Even more sophisticated formulae for the degree of freedom, such as the one derived by Kutzbach [Norton, 2003], represents only a necessary condition, but not a sufficient condition.

2.5.3. Static and Kinematic Indeterminacy

Structures are referred to as *statically indeterminate* if there is a unique solution to the equilibrium equations for any applied loading. A statically indeterminate structure will admit different states of self-stress, where the structure can be stressed against itself, even in the absence of external loads. In practical terms, if the bars of a statically indeterminate truss are not fabricated to precise lengths, force will be required to fit the bars, resulting in a self-stressed structure.

Structures are termed *kinematically indeterminate* if there is a unique solution to the compatibility equations for any set of internal extensions of the bars. If a structure is kinematically indeterminate, then there will be certain movements of the joints, where, at least to the first-order approximation, there are no changes in bar lengths. Kinematically indeterminate structures are *mechanisms*. Traditionally, structural engineers avoid mechanisms, and redesign structures with mechanisms to avoid instability and large motion. The assumption of first-order joint displacements has a practical significance. Small-displacement, linear finite element analysis is widely used in engineering practice, and this method is limited to kinematically determinate structures. In the special case of prestressed structures, however, mechanisms play an important role. The development of cable nets, fabric roofs and tensegrity structures [Fuller, 1975] in the 1970s led to a renewed interest in the theoretical aspects of prestressed mechanisms [Kuznetzov, 1975; Calladine, 1978; Pellegrino and Calladine, 1986].

2.5.4. Linear Structural Analysis

This section gives an overview of the linear structural analysis of pin-jointed bar frameworks. The analysis concerns a three dimensional assembly with j joints connected by b pin-jointed bars; a total number of c kinematic constraints prevent the joints from moving.

In linear structural analysis, three principles must be satisfied; that internal forces t are in equilibrium with the applied forces $\boldsymbol{f}$, that any internal deformation $\boldsymbol{e}$ is compatible with external displacements $\boldsymbol{d}$, and that internal forces $\boldsymbol{t}$ and elongations $\boldsymbol{e}$ are related by a material law. The ($3j$-c)-dimensional *nodal load vector* $\boldsymbol{f}$ contains the x-, y- and z-

components of the external forces applied at each node of the assembly. The b axial forces are assembled in the *tension vector* $\boldsymbol{t}$, where the axial force in bar i, t_i, is positive if tensile. The x-, y- and z-components of the displacement of each node, excluding kinematically constrained directions, are assembled in the ($3j$-c)-dimensional nodal displacements vector $\boldsymbol{d}$. The *elongation vector* $\boldsymbol{e}$ consists of the b bar elongations, with the convention that the elongation of bar i, e_i, is positive for an increase in length.

For small perturbations about the initial equilibrium configuration of a structure, these relationships can be linearized as three matrix relationships:

$$\boldsymbol{f} = \boldsymbol{At} \tag{2.13}$$

$$\boldsymbol{e} = \boldsymbol{Cd} \tag{2.14}$$

$$\boldsymbol{t} = \boldsymbol{Ge} \tag{2.15}$$

where the ($3j$-c×b) coefficient matrix $\boldsymbol{A}$ is the *equilibrium matrix*, the (b×$3j$-c) coefficient matrix $\boldsymbol{C}$ is the *compatibility matrix*, and $\boldsymbol{G}$ is a diagonal matrix with element stiffnesses EA/L on the diagonal.

The static-kinematic duality of the above equations can be demonstrated using the principle of virtual work. Equating internal and external work,

$$\delta W_{int} = \delta W_{ext}\text{ , or} \tag{2.16}$$

$$\delta \boldsymbol{e}^T \boldsymbol{t} = \delta \boldsymbol{d}^T \boldsymbol{f}\, , \tag{2.17}$$

and substituting equations 2.13 and 2.14 this becomes

$$\delta \boldsymbol{d}^T \boldsymbol{C}^T \boldsymbol{t} = \delta \boldsymbol{d}^T \boldsymbol{At}\, , \tag{2.18}$$

which is valid for any displacement $\delta\boldsymbol{d}$, and results in

$$\boldsymbol{C}^T = \boldsymbol{A}. \tag{2.19}$$

Using the stiffness method, the internal forces are condensed out of the above equations (2.13-2.15) to form a single stiffness relationship which relates external forces to nodal displacements. Substituting equations (2.15), (2.14) and (2.19) into equation (2.13),

$$\boldsymbol{f} = \boldsymbol{AGA}^{\boldsymbol{T}}\boldsymbol{d} = \boldsymbol{K_o d}\, , \tag{2.20}$$

where $\boldsymbol{K_o}$ is the *linear stiffness matrix*. If a system is to be stable, the matrix $\boldsymbol{K_o}$ must be positive definite, or equivalently, of full rank, where $r(\boldsymbol{K_o}) = 3j - c$. It will be shown, however, that determining the rank of the equilibrium matrix is sufficient to establish the stability of the system [Deng and Kwan, 2005].

$$\boldsymbol{K_o} = \boldsymbol{AGA^T} = \boldsymbol{A(G^*G^{*T})A^T} = \boldsymbol{(AG^*)(AG^*)^T} \qquad (2.21)$$

where $\boldsymbol{G^*}$ is also a diagonal matrix. For regular bar frameworks, all elements in $\boldsymbol{G}$ are positive, so the rank of $\boldsymbol{AG^*}$ takes its value from the rank of $\boldsymbol{A}$. Furthermore, since the rank of a matrix $\boldsymbol{X}$ is the same as that of $\boldsymbol{XX^T}$, then

$$r(\boldsymbol{K_o}) = r[\boldsymbol{(AG^*)(AG^*)^T}] = r(\boldsymbol{AG^*}) = r(\boldsymbol{A}) = r. \qquad (2.22)$$

Therefore, whether $\boldsymbol{K_o}$ is positive definite can be judged from the rank of the equilibrium matrix $\boldsymbol{A}_{3j\text{-}c\times b}$.

2.5.5. Equilibrium matrix analysis

A detailed analysis of the equilibrium matrix yields considerable insight into the characteristics of a structure, as shown by [Pellegrino and Calladine, 1986]. The equilibrium matrix $\boldsymbol{A}$ can be considered a linear operator between two vector spaces, the bar space $\Re^b$ and the joint space $\Re^{3j-c}$. The four fundamental subspaces associated with $\boldsymbol{A}$ have the following properties.

- If the Nullspace of $\boldsymbol{A}$ has dimension s=0, then the structure is statically determinate.
- If s>0 then the assembly is statically indeterminate, and s=b-r is the number of independent states of self-stress it admits.
- If the Left-nullspace of $\boldsymbol{A}$ has dimension m=0, then any load $\boldsymbol{f}$ can be equilibrated by the assembly in its initial condition, and the assembly is kinematically determinate.
- If m>0 the assembly is kinematically indeterminate, and m=$3j$-c-r is the number of independent inextensional mechanisms.

Note that these subspaces coincide with the subspaces associated with the compatibility matrix $\boldsymbol{C}$(=$\boldsymbol{A}^T$). Also, note that the dimensions of the four subspaces can easily be computed once the rank r of the equilibrium matrix is known.

The value of r can be determined in several different ways, but using the Singular Value Decomposition (SVD) on the equilibrium matrix also gives orthogonal sets of m inextensional mechanisms and s states of stress. Note that algorithms to compute the SVD are part of commercially-available software such as MATLAB.

$$\boldsymbol{A} = \boldsymbol{U\,\Sigma\,V^T} \qquad (2.23)$$

where $\boldsymbol{U}=\{u_1,u_2,...,u_{3j-c}\}$ consists of a set of left singular vectors, $\boldsymbol{V}=\{v_1,v_2,...,v_b\}$ contains a set of right singular vectors, and a set of singular values is found in the first r non-zero diagonal elements of $\boldsymbol{\Sigma}$.

The singular vectors, all of unit norm, can be grouped into the following submatrices

$$\boldsymbol{U_r}=\{u_1,u_2,...,u_r\} \qquad \boldsymbol{U_m}=\{u_{r+1},...,u_{3j-c}\}$$
$$\boldsymbol{V_r}=\{v_1,v_2,...,v_r\} \qquad \boldsymbol{V_s}=\{v_{r+1},...,v_b\}$$

which have the following interpretations:

- $\boldsymbol{U_r}$ contains modes of extensional deformation (i.e. loads that can be equilibrated by the structure in its current configuration);
- $\boldsymbol{U_m}$ contains modes of inextensional deformation, i.e. mechanisms (i.e. loads that cannot be equilibrated by the structure in its current configuration);
- $\boldsymbol{V_r}$ contains set of kinematically compatible extensions corresponding, through the singular values, to the extensional modes in $\boldsymbol{U_r}$;
- $\boldsymbol{V_s}$ contains sets of kinematically incompatible extensions (i.e., states of self-stress).

The basis for states of self-stress is often referred to as

$$\boldsymbol{SS} = [v_{r+1},...,v_b] \tag{2.24}$$

and the basis for the mechanisms is

$$\boldsymbol{D} = [u_{r+1},...,u_{3j-c}]. \tag{2.25}$$

The mechanisms in $\boldsymbol{D}$ can be either internal mechanisms or rigid-body mechanisms, as a result of inadequate kinematic constraints on the structure. A scheme to separate the internal mechanisms from the rigid-body ones was proposed by [Pellegrino and Calladine, 1986], and it can accommodate up to six rigid-body mechanisms.

2.5.6. Mechanisms

A kinematically indeterminate structure with internal mechanisms is not necessarily unstable. If the structure tightens up as the mechanism is displaced, because of second-order or higher order bar length changes, the mechanism is called *infinitesimal*. If the structure does not tighten up, the mechanism is called *finite*. Infinitesimal mechanisms have been classified according to the order to which the changes of bar length relate to the displacements [e.g. Vassart et al., 2000]. First-order infinitesimal mechanisms, where displacements are related to second-order bar length changes, may be stabilized by a state of self-stress, in which case the structure is termed *prestress stable*.

The stiffness of mechanisms can be assessed using geometrically nonlinear iterative methods, so-called large displacement analysis. Another approach is to use the *product forces* to calculate the geometric loads associated with an inextensional mechanism [Pellegrino and Calladine, 1986; Pellegrino, 1990]. This method has been found to be a special case of the geometrically nonlinear tangent stiffness matrix found in nonlinear FEA [Deng and Kwan, 2005].

The rigidity or stability of mechanisms is also covered by mathematical Rigidity Theory [Connelly and Whiteley, 1992], using different terminology. A *tensegrity framework* $G(\boldsymbol{P})$ is defined by a graph G on $\boldsymbol{P}=[p_1,p_2,...,p_n]$, where each edge is either a cable, strut or bar. Cables cannot increase in length, struts cannot decrease in length, and bars cannot change length. A broadly defined term *rigidity* incorporates all frameworks that do not have finite mechanisms. Since rigidity theory is not directly linked to the physical realization of structures, stability is established independent of material properties and member cross-sectional properties.

The structural synthesis of kinematic mechanisms uses computational methods to enumerate all mechanisms having a specified number of links, degrees of freedom, and types of joints. Graph theoretical techniques are used to detect rigid, or *degenerate*, mechanisms [e.g. Davies, 1968; Lee and Yoon, 1992]. Many of these techniques are limited to mechanisms satisfying Gruebler's equation, to planar mechanisms, or planar graphs of mechanisms.

2.6. Summary

This section summarizes the research background covered in this Chapter, highlighting the gaps in the existing body of research and the need for further work.

Early artificial intelligence methods such as expert systems have not found successful application in conceptual structural design, primarily because of the lack of well accepted domain models for design synthesis. Expert systems for design have largely been supplanted by case-based reasoning, which has the potential to leverage the vast amount of information contained in past experiences. Research in CBR for design applications has yielded effective methods of human-computer interaction, particularly in creative disciplines like architecture, where it is important for the user to retain control of the design process. A major shortcoming of CBR in conceptual structural design is the lack of support for visual information. Existing CBR systems for conceptual design use primarily textual attributes or simplified geometric attributes. Although sophisticated text-based case retrieval methods have been developed, these methods cannot fully represent the visual and graphical information that is an important part of conceptual structural design. One of the goals of this research is to apply visual case-based reasoning techniques to conceptual structural design.

2. Background

The conceptual design of buildings has been extensively covered in the literature, given the obvious economic implications. Computer systems that support conceptual building design generally use geometric reasoners with limited capabilities. The reasoners are limited to the relatively simple spatial relationships found in economical buildings based on a rectilinear grid system with conventional column and beam construction. Although there are many potential applications for such systems, they are generally not suitable for the free-form, curvilinear construction that is common in modern architecture. This research proposes a system that is useful for a wide range of structural applications, including scientific instruments, industrial equipment supports, bridges and buildings.

Sophisticated graphical user interfaces have been developed in fields such as architecture and industrial design, where natural, pen-based applications simulate sketching, clay modeling and other creative form-finding methods. Unfortunately, such applications do not recognize content such as the meaning of elements and relationships between them. Symbolic sketch recognition computer programs have been developed in architecture and mechanical design. For mechanism design, there are diagrammatic reasoning systems that convert unlabeled line drawings into a description of a physical system. The systems infer structural components, connections, causal interactions, processes and functionality from drawings. No similar applications exist in the field of conceptual structural design.

Major commercial CAD software applications now support natural input methods such as sketching. A significant drawback of such applications is that they do not understand the language of requirements, and require the user to interpret these in the form of well-defined design concepts. The framework proposed in this research directly interprets design requirements, synthesizes new concepts, and evaluates these concepts even though they are not precisely defined.

Of the vast number of computer applications for engineering optimization, topology optimization methods are most applicable to conceptual structural design. Continuum topology optimization methods are used for the design of plate, shell and solid structures. Truss and frame topology optimization methods are used for the design of skeletal structures consisting of an assemblage of discrete members. For truss topology optimization, one of the most general methods is the ground structure approach, which starts with a large number of potential design configurations. This method is known to be computationally demanding, particularly for a fully connected ground structure with fine node spacing. In practical applications, the ground structure is therefore relatively coarse, and must be carefully selected based on both domain- and problem-specific considerations. Truss topology optimization is complicated by the fact that even small changes in topology can lead to large differences in stiffness, limiting the use of classical gradient-based optimization techniques. Global optimization methods such as evolutionary computing have been found to be effective for discrete topology optimization. For design problems that are highly nonlinear or nonconvex, global optimization methods may be the only practical solution technique. Methods that use randomly generated ground structures to generate initial populations represent some of the most flexible and efficient evolutionary computing techniques for conceptual structural design. Such methods are particularly useful when the search space is complex or poorly-understood, and they can be used as a tool to explore and gain a better understanding of that space. If the search space contains structure that can be exploited by special-purpose search techniques, the use of evolutionary methods is generally computationally less efficient than those techniques [De Jong, 1990]. There is a need for efficient methods for finding and optimizing the topology of discrete structures. This research proposes an efficient, special-purpose framework for rapidly generating discrete topology at the conceptual design stage.

Continuum topology optimization methods are efficient, well-established, and commercially available. The major drawback is that the output of such methods is not directly suitable for fabrication, particularly at scales larger than a few meters. Research in continuum topology optimization has appeared to shift towards micro- and nano-scale fabrication and material design. Some researchers suggest that the difficulties with fabricating the shapes produced by topology optimization will lessen as manufacturing capabilities are improved, a clear reference to micro- and nanotechnology [e.g. Rozvany, 2009]. Changes in manufacturing techniques for large scale structures do not occur as quickly, and the limitations in applying continuum topology optimization to such structures are expected to persist for some time. The framework presented in this research leverages the efficiency of continuum optimization methods, and extends their range of applicability to large scale structures. Although other research has been done in this area, the work proposed here represents a wider approach that integrates topology generation with visual case-based reasoning and visual inference methods. Also the this work proposes an efficient method for generating stable skeletal structures.

Although methods for verifying structural stability are well established, efficient methods for generating stable structures are not. In truss topology optimization, stability is generally ensured using heuristics and generate-and-test methods. A common heuristic is to add sufficient members to ensure that all polygonal cells are triangular. Generate-and-test refers to the generation of a large number of different topological configurations, and filtering out the ones that are unstable. This work proposes an efficient method for generating stable skeletal structures using classical mathematical optimization methods. The stability optimization method presented here produces information that directly supports the detailed analysis and design of economical connections between members.

Given the importance of visual and graphical information during conceptual design, it is remarkable that few computational tools for conceptual structural design exploit this information. In the fields of image processing and pattern recognition, there are well-established, rigourous techniques for manipulating graphical information. Such techniques have been applied for many years in areas such as medical imaging, remote sensing, and maufacturing, but few of these techniques have been applied to conceptual structural design. The framework presented here makes extensive use of these methods to generate conceptual designs and reason with those designs at relatively high levels of abstraction.

3. Framework

This chapter gives an overview of a proposed framework for the conceptual design of structures. The overview is followed by a detailed discussion on the components making up the framework.

3.1. Overview

The conceptual design of structures is a fluid process. Concepts are repeatedly synthesized, dissolved, combined and evolved. To effectively explore a wide expanse of design space requires rapid evaluation of concepts. Evaluation at this stage of design is more subtle than at later stages. The concepts are still abstract and lacking numerical certainty, and often requirements are not completely formed. Compounding this, there are many different possible views of an abstract concept, reflecting the varied experiences of the participants. Conceptual design is a highly visual and symbolic process, where sketches and diagrams are essential tools to crystallize design from ideas.

The most important creative force during conceptual design is the human designer. To effectively aid the designer, computational tools should recognize the key processes a human designer uses. These processes include the retrieval of past experience, and the evaluation and modification of design concepts. Evaluation techniques are generally qualitative, but quantitative methods are also used. Qualitative evaluation includes the use of logic and heuristics. Conceptual design requires reasoning at high levels of uncertainty and abstraction. Inferential reasoning is important in applying past experiences, provided there is a mechanism for understanding similarity between concepts. Throughout the design process, the designer modifies, combines and reevaluates existing concepts to produce new ones. Once a concept has been selected, the designer acquires additional knowledge as the design is developed to the detail stage, then fabricated, installed and put in service. This knowledge becomes part of the store of experience for future designs.

3. Framework

The framework proposed here has been developed to support the conceptual design of structures by enabling the rapid generation of new designs, and by facilitating the reuse of past designs. Visual reasoning techniques are central to the framework, and are used to evaluate, describe, classify and learn from the forms that embody structural design concepts. A formal computational framework to support design may be described in terms of a *design process model*, a *representation model*, and a set of *problem-solving methods*.

The design process model is centered on the *generation* or *synthesis* of alternative conceptual forms. Conceptual structural design starts with a set of design requirements, from which lower level specifications or expectations are formulated. A conceptual design description is typically developed through a cycle, where a concept is proposed, its behaviour is analyzed, and this behaviour is compared to expected behaviour. In the design process model proposed here, generation relies heavily on *mathematical optimization* methods. Optimization is a constrained, stepwise search for an optimal configuration. In a design problem, the meaning of the term optimal is provided by the requirements and specifications. At each optimization step, analysis is performed to determine a new search direction and step size. Although optimization is a widely-used strategy for problem solving in general, this work uses optimization methods that are specifically suited to conceptual structural design, including topology optimization.

Design creativity, unlike design analysis, is a process that is not readily learned in the classroom. While reliable procedures can be described to do analysis, creativity is almost always learned by example. The most creative designers draw heavily from a range of personal experiences and observations, often applying knowledge from past situations in a new context. *Case-based reasoning* is a computational paradigm for reuse of past experiences. The design process model described here relies on case-based reasoning techniques to classify, store and retrieve past design experiences. Case-based reasoning has been used in several research systems for conceptual structural design. In those systems, the emphasis is has been textual information. In contrast, this work focuses on visual case-based reasoning, which uses images and shapes, in addition to text. Visual case-based reasoning overlaps the related areas of pattern recognition and content-based image retrieval.

Reasoning at multiple levels of abstraction, using incomplete and uncertain information, is a fundamental requirement during conceptual design. A flexible data representation model is needed to support this type of reasoning. Abstraction is an important concept for organization, classification and retrieval of data. Hierarchical data models are often used to represent data abstraction. Object-oriented programming methods support hierarchy and abstraction through techniques such as inheritance. For effective retrieval of cases from a case-based reasoning system, concepts should be related at different levels of abstraction. This is because different users will view the same concept at different levels of abstraction, or with differences in specific terminology. In this work, abstraction is supported at a number of levels. For example, images of specific forms are generalized into symbolic structures, such as relational graphs. Also, images are viewed at multiple levels of resolution to enable higher level, content-based comparisons. Using abstraction, comparisons between concepts can be made at several different levels, supporting learning by inference.

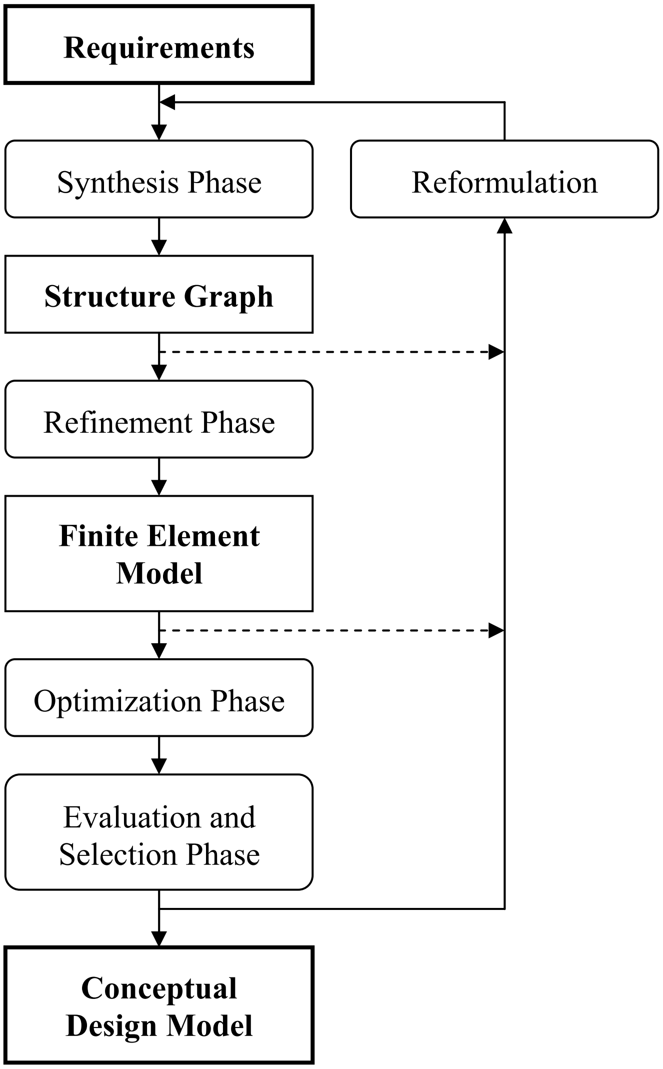


Figure 3.1. Overview of conceptual design generation

All stages of the design process have uncertainty, but none more so than early design. In the final stages of design, much of the uncertainty is random in nature. In contrast, the bulk of the uncertainty during conceptual design is due to a lack of information. As design concepts are incrementally refined, more detailed information is added. A flexible representation model is required to represent both early abstract concepts and later detailed information, and allow for a seamless transition from one to the other.

3. Framework

The design framework uses a number of different problem-solving methods from a range of disciplines, including structural analysis, mathematical programming, image processing and pattern recognition. The finite element analysis method, the workhorse of the structural design office, is used in this work for different purposes, at varying levels of abstraction. Finite element methods are used to evaluate the objective function during structural topology optimization. For certain objectives, such as minimum compliance, topology optimization requires very few details on the specific material and load values, and is an ideal tool for conceptual design. In this work, finite element methods are also used to generate a series of potential stable design configurations, and to evaluate objective functions in multiobjective optimization. This framework also utilizes a range of tools from image processing and pattern recognition to incrementally convert an image of a specific structural form to a more abstract representation of its content.

Using the framework, a design is initiated using one of two methods: by the design generation method, or by case-based reasoning. The design generation uses methods in structural topology optimization to synthesize a structural form from a set of requirements. In case-based reasoning, a structural form is retrieved from a library and is used as the basis for a new design. Forms may be retrieved by supplying a set of requirements, or by specifying a form or shape to match. An overview of the design generation method is given in Figure 3.1. This method is covered in detail in Section 3.2 through 3.4. Alternate generation methods are presented in Section 3.6. Case-based reasoning is discussed starting in Section 3.7.

3.2. Synthesis Phase

The goal of the Synthesis Phase is to transform design requirements into a design description. In this case the design description is embodied by a description of the structural topology. Specifically, in this framework structural topology is limited to planar skeletal structures. Skeletal structures are structures that can be decomposed into a set of linear structural elements whose cross-sectional dimensions are relatively small compared to their length. The term *skeletal* is meant to include both classical trusses with frictionless pin joints, as well as moment frames. The rationale for this terminology is related to the interpretation of topology optimization output, which is discussed in the following sections.

An overview of the Synthesis Phase is shown in Figure 3.2. The components of this phase will be discussed in detail in the subsequent sections. Synthesis begins by translating design requirements into specifications, which are then used to define a topology optimization problem. The result of topology optimization is a structural form. Using image processing and pattern recognition techniques, the form is described at progressively higher levels of abstraction. These higher level form descriptions have several purposes. They support analogical reasoning, which is important in learning and retrieving similar scenarios. Also, form descriptions support the effective use of multiple criteria optimization, to incorporate objectives related to detail design, fabrication, and end use.

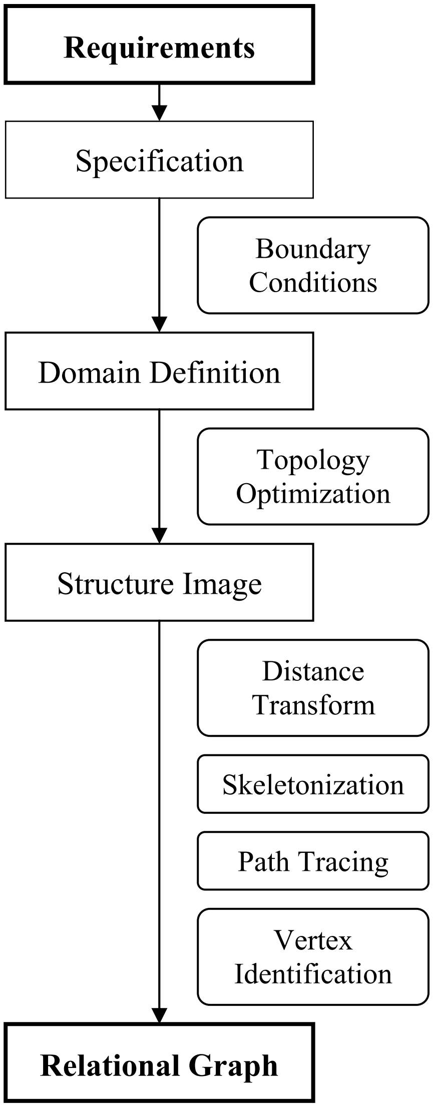


Figure 3.2. Framework overview – Synthesis Phase

3.2.1. Design Formulation

In design, the process of *formulation*, describes how general design requirements are translated into specifications. Formulation has been described as the process of "transforming function to expected behaviours", where "the expected behaviour provides the syntax by which the semantics represented by function can be achieved" [Gero, 1990]. Formulation can be one of the more challenging tasks in early design, particularly for complex designs involving more than one discipline. Design requirements often conflict with each other, and establishing the relative importance of requirements is not straightforward. Requirements may have different levels of "firmness", from "soft" requirements or mere *goals* at one extreme, to essential or non-negotiable "hard" requirements at the other. Techniques of multiobjective optimization have been used to give a mathematical description of varying levels of firmness, but this approach is not always useful. In architectural applications, for example, the overall vision of the designer or particular aesthetic qualities are not easily translated into specifications. In practice, formulation is a process that is repeated several times as more knowledge about the potential performance of a design is gained through analysis. Formulation and reformulation usually requires the direct involvement of designers, end users, and other participants.

The framework described here is designed for situations in which requirements can be described primarily using geometric constraints. All this does not cover all situations, a vast range of conceivable design problems can be formulated in this way. The shape of bridges is constrained geometrically by the potential abutment locations and the clear space to allow for the flow of traffic on and below the bridge. The layout of telescope support structures is limited by the light path volumes defined by the optical configuration. Even in expressive, free-form architectural applications, geometric constraints can be applied by mathematically sculpting a permissible design space from a block of available space.

3.2.2. Representation Model

The notion of the design domain is prominent in the framework. In this work, the design domain is a data model that represents the space available for design, the material, and the boundary conditions. The available space may be shaped using a set of applied geometric constraints. The material is assumed to be isotropic. Boundary conditions are used to specify loading and support conditions

The domain data model is used not just in the Synthesis Phase, but in all other stages including the Refinement, Optimization, and Case-based Reasoning Phases. The design domain is a high-level construct that does not presuppose the topology or details of the structure that is synthesized, making it an effective representation for conceptual design.

An overview of the *Domain* data model is shown in Table 3.1. In two dimensions, the reference domain is constructed from a rectangular bounding box defining a planar area. The domain is associated with an isotropic material, whose properties are stored in the data model. The domain area is discretized; this discretization defines the finite element model used in subsequent topology optimization.

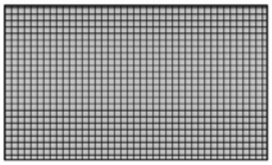
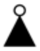
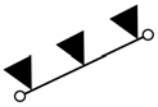
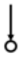
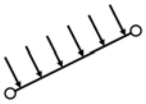
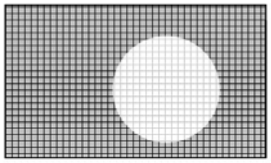
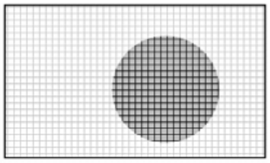
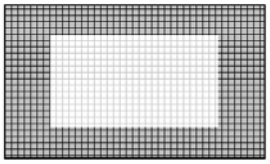
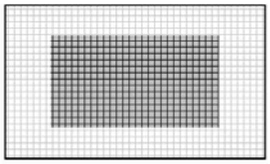
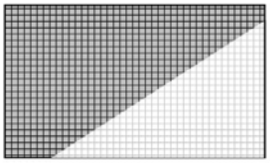
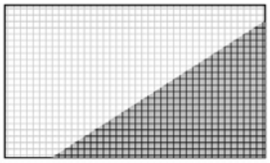
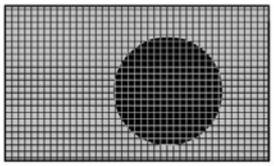
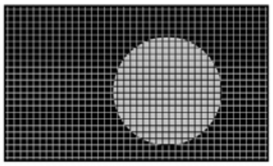
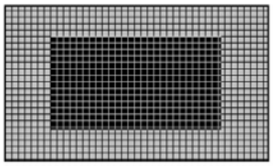
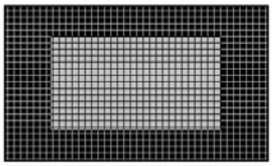
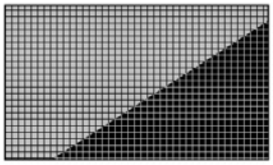
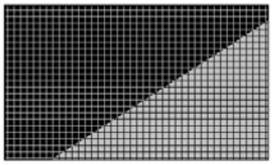
Domain		
Size *Width* *Height* *ElementSize* *Material*		
Bounds *BoundPoints* *BoundLines*		
Forces *ForcePoints* *ForceLines*		
SubtractedAreas *Circle* *Rectangle* *Line*		
◍ *included* ○ *subtracted*		
		
RetainedAreas *Circle* *Rectangle* *Line*		
◍ *included* ● *solid*		
		

Table 3.1. Design domain representation

Boundary conditions are applied to the domain at a series of individual points, or distributed along a line. Point supports and point loads are specified using the cartesian coordinates of a point on the domain. For supports the fixity type (translation in *x*- or *y*-directions) is supplied. For forces, the direction (*x* or *y*) and the magnitude is specified. Geometric constraints on the domain are applied using a set of basic operations, as shown in Table 3.1.

During the design synthesis process, a structural configuration is created using the available material. Initially, the available material is defined by a rectangular bounding area. Voids in the design domain can be specified using the *SubtractedArea* construct. Areas of solid material are included using the *RetainedArea* item. Using circular, rectangular and triangular areas as primitives, the design domain can be shaped with varying levels of complexity.

3.2.3. Topology Optimization

Structural topology optimization is well suited to the conceptual design phase. Topology optimization starts with the essential elements of a structural design problem: the spatial constraints and boundary conditions. Structural optimization techniques may be classified under refinement methods or synthesis methods. While refinement methods start with a configuration and incrementally improve it, synthesis methods create a new configuration using a minimal set of assumptions. Topology optimization methods do not assume a predefined design configuration. The structural configuration is synthesized by using mathematical techniques to optimally distribute material to where it is most efficiently used, and remove where it is not needed. The synthesized shape may be a new and unexpected shape. Topology optimization methods thus support the creative process of conceptual design by creating new configurations and suggesting alternative directions for the designer to pursue.

The framework synthesizes design configurations using continuum topology optimization methods. The domain representation model described in the previous section is directly applicable to such problems. The subject of continuum topology optimization is well developed. Many different problem formulations have been developed, depending on the optimization objectives and the mathematical techniques used. Common objectives include minimizing compliance, maximizing natural frequency, satisfying stress and buckling constraints, or meeting reliability targets. The goal of this work is not to suggest new topology optimization techniques but to leverage this powerful method to for wider use in conceptual design work.

The conceptual design synthesis approach described here emphasizes the minimum compliance objective. Compliance minimization has several advantages over other topology optimization formulations. The input requirements are relatively simple, compared to the stress constraint formulation, for example. The only inputs are the boundary conditions and a volume reduction fraction. The volume fraction sets a target for the ratio of the volume in the structure to the volume of the reference domain. For a single load case, the results are independent of the magnitude of the load. For multiple loads, only the ratio between load vectors is important and precise magnitudes need not be entered. At the conceptual stage, where exact load values are usually not known, topology optimization can still be performed.

A sample design problem will be used to illustrate the application of the framework. The problem is to design a cantilever structure with maximum compliance for a given volume of material. This relatively simple problem will be used throughout this Chapter to illustrate the sequence of processes and flow of data through the different phases of the framework. More detailed design problems are given in Chapter 4. The input to the cantilever structure problem is shown in Figure 3.3. The reference domain has a width 80 units and height 20 units. The domain is discretized into elements of size one unit squared. A downward load is applied at the tip. The material has Poisson's ratio 0.3, and the specified volume ratio is 0.3, indicating the structure should have a weight equal to 30% of the weight of the full design domain.

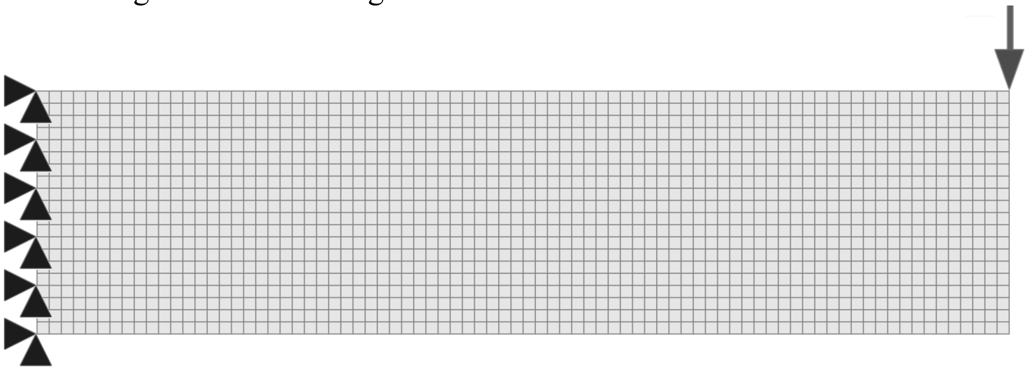


Figure 3.3. Cantilever beam sample problem domain definition

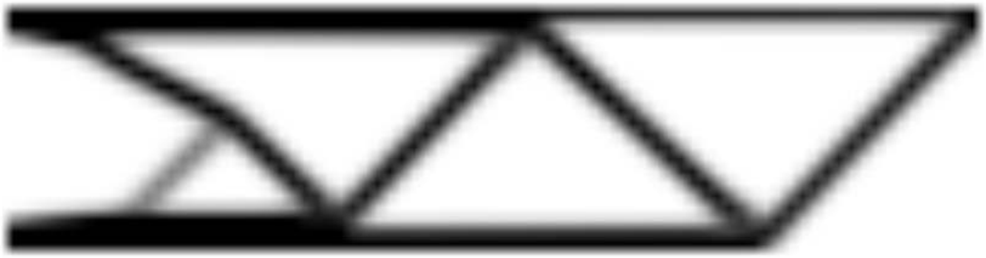


Figure 3.4. Topology optimization output

The output of topology optimization is a raster image, where each pixel corresponds to a finite element. In this work, the material density method is used, and the output for each element is a value between 0 and 1. Using the SIMP interpolation method [Bendsøe and Sigmund, 2004], intermediate densities are penalized, so most of the values fall close to 0 or 1. Since the interpolation function is continuous, however, intermediate values do exist. These intermediate values are converted to 0 or 1 by selecting an appropriate threshold level. The topology optimization output for the sample problem is shown in Figure 3.4, and the results after thresholding are shown in Figure 3.5 and Figure 3.6. Changing the threshold level changes how the topology optimization output is interpreted. Figure 3.6 uses a higher threshold value, which filters out the lighter grayscale values from the topology optimization output shown in Figure 3.4. As a result, Figure 3.6 implies a skeletal structure with at least one less member than Figure 3.5.

Load cases in topology optimization generally have a strong influence on the results. Where a single load case is used, the resulting structures may only be stable under the given set of loads. Single load cases often produce skeletal structures that are not fully triangulated. Topology optimization under multiple load cases is usually required to produce realistic structures. Multiple load cases can be applied simultaneously or combined using a method such as the minimization of a weighted average of the compliances. Different topologies result, depending on the method, so consideration should be given to whether loading is in fact simultaneous.

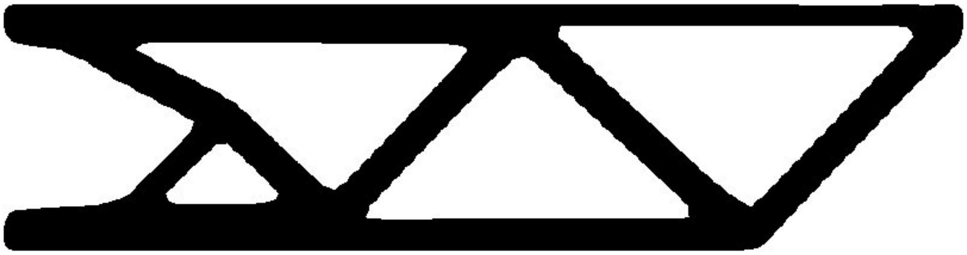


Figure 3.5. Topology optimization output in binary form (threshold 0.2)

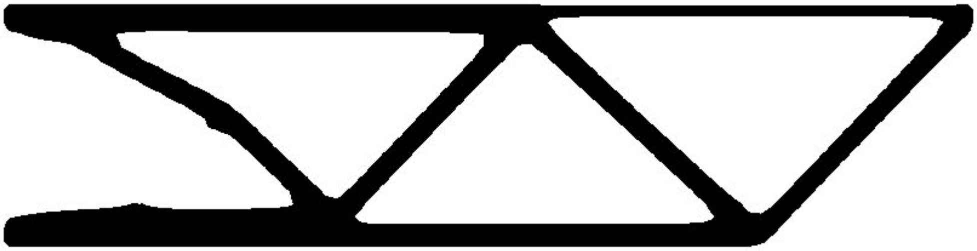


Figure 3.6. Topology optimization output in binary form (threshold 0.5)

Topology optimization results are sensitive to the density of the finite element mesh. As mesh size is decreased, the topology changes as more detailed features are formed. In practical macro-scale applications, a lower bound on the feature size is usually sought. Also, it is preferable to use mesh refinement to produce smoother boundaries on existing topology rather than create completely new topology. For these reasons, many topology optimization codes include mesh-independence filters. This method takes a parameter called the filter radius, which sets the minimum length scale. The example shown here uses the method introduced by [Sigmund, 1997], with a filter radius equal to 1.5 times the element size.

Multiple design configurations may be generated in the framework by using a range of threshold values to interpret the topology optimization output. Multiple designs may also be generated by varying other parameters of the topology optimization, such as the volume fraction, the mesh independency filter radius, and the convergence tolerance. Variations in the loading configuration also can be used to produce a range of different topologies for subsequent processing.

The structure represented in Figure 3.5 could simply be converted to a CAD format by fitting curves to the boundaries. A CAD representation does not capture the essential characteristics of the structure and cannot be used for further reasoning. For example, a simple interpretation of the boundaries does not readily support decisions on the selection and optimization of fabrication and assembly processes. The proposed framework processes the topology image to produce a parametric model that supports practical design decisions.

Topology optimization supports the fabrication of structures at virtually any scale. The method has been used in the design of micro-electro-mechanical systems (MEMS), where microscale devices are fabricated using techniques such as etching, deposition and lithography. Topology optimization has also been used extensively in the automobile and aircraft industries. A structure like the one represented in Figure 3.5, for scales ranging from a few centimeters to several meters, is often fabricated by removing material from a slab or plate by machining or cutting. For example, laser, plasma, and water-jet cutting equipment can produce such shapes relatively economically, if the material thickness is within limits. On the other hand, if there is a high proportion of a void to solids, this method produces considerable waste. Structures can also be made by building up material using solid freeform fabrication processes such as selective laser sintering, electron beam melting or welding, however these processes are often prohibitively expensive and unsuited to structures more than about one meter in size.

For many fabrication processes, particularly at scales larger than a few meters, the most economical approach is to assemble a structure from components. For example, economical truss-like forms are frequently produced by joining a number of straight or curved segments together. The segments are made by cutting lengths of rolled or extruded shapes. Decomposition is the process used to interpret a monolithic structure like the one shown in Figure 3.5 as an assembly of smaller components. This method obviously becomes increasingly cost effective as the similarity between components increases. To decompose the topology image and construct a higher level parametric model, the framework employs image processing techniques.

3.2.4. Image Processing

The field of image processing offers a range of techniques to improve, simplify and compress images, and to facilitate further processing such as pattern recognition. Topology optimization output is easily converted to a binary image using a thresholding process.

The following sections describe how the topology image is transformed into a simplified representation that preserves the essential information contained in the image.

3.2.4.1. Thinning

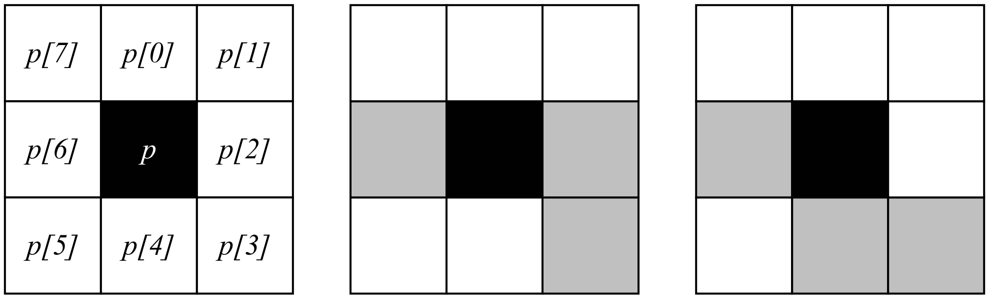
Continuum topology optimization under low volume fractions produces distinct truss-like structural forms. Such forms are characterized by linear elements with cross-section dimensions that are small relative to the length of the element. Thinning is an image processing technique aimed at producing a simplified representation of line-like or curve-like features.

The framework implements a morphological hit-or-miss thinning algorithm based on [Zhang, 1997] to produce a skeleton representation of the structural form. This is an efficient algorithm that generates one-pixel wide skeletons directly. An overview of the algorithm is presented here.

An binary image is a mapping, *I*, from a set, S_p of pixel coordinates to a set *B*={0,1} of values such that for every coordinate vector, ***p****=(r,c)* in S_p, there is a value *I(p)* drawn from *B*. The foreground set (or black pixel set) *FG* is the set of locations, ***p***, where *I(**p**)* = 1. The complement set will be termed the background set (or white pixel set).

$$FG\{I\} = \{I(p), p = (r,c) \in S_p | I(p) = 1\}. \qquad (3.1)$$

The *neighbourhood* of *p*, *N*(*p*), is the set of all neighbours of *p* in a 3x3 *window*, shown in Figure 3.7.a.



a. b. *WN(p)*=76 c. *WN(p)*=88

Figure 3.7. Thinning algorithm definitions

For a 3x3 window, there are $2^8 = 256$ possible combinations of neighbouring pixel values $\{0,1\}$. The *weight number* of p is defined as

$$WN(p) = \sum_{k=0}^{7} p[k] \cdot 2^k \ . \tag{3.2}$$

The *neighbour number* of p is the number of nonzero neighbours of the current pixel p:

$$NN(p) = \sum_{k=0}^{7} p[k] \tag{3.3}$$

The thinning algorithm is an iterative process, where pixels are incrementally removed from the boundary of the shape represented by the foreground set. Each foreground pixel of the image is examined, and compared along with its neighbours to a set of 3x3 *thinning templates.* The thinning templates define the pixel patterns in the neighbourhood of *p* which allow the removal of *p* from the foreground. For example, the pattern of neighbouring pixels in Figure 3.7.c would match such a template, while the pattern in Figure 3.7.b would not. Removal of the central pixel in Figure 3.7.b would disconnect pixels of the foreground set, leading to eventual removal of the entire foreground. Removing the central pixel in Figure 3.7.c refines the foreground shape towards an 8-connected, single pixel width skeleton. A *globally removable set* (GRS) of points is constructed, which contains the weight numbers corresponding to the thinning templates. For example, the GRS contains the weight number *WN(p)*=88, but not *WN(p)*=76.

The procedure described up to this point simply recapitulates existing thinning algorithms. A key difference between existing thinning methods and the one used in this framework lies in the treatment of pixels with a single neighbouring foreground pixel, the case where *NN(p)*=1. In the framework, foreground pixels with one neighbour are deleted *unless a boundary condition is applied near that pixel*. The boundary conditions are stored as part of the domain representation model, which is available to the thinning algorithm. The rationale for removing singly connected branches is that such branches, unless connected to a point where a boundary condition is applied, carry no load and have no structural function. The thinning algorithm thus returns a skeleton that requires no further processing to clean up small branches that are a feature of many thinning algorithms, except near points where boundary conditions are applied. These points can be handled in subsequent processing using methods that operate locally in a region near support points or points where forces are applied.

The output of the thinning algorithm, superimposed on a grayscale version of the binary topology image, is shown in Figure 3.8.

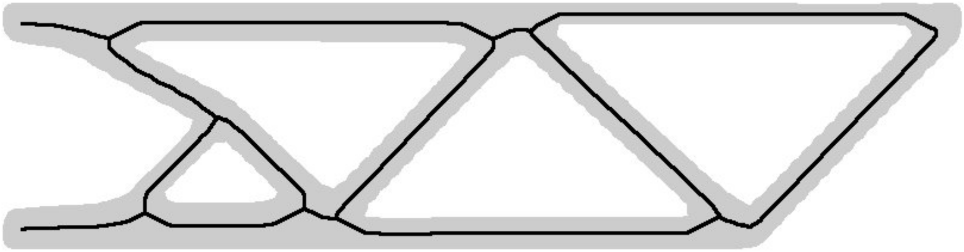


Figure 3.8. Skeleton

3.2.4.2. Vertex Identification

Starting with a single pixel wide skeleton, the process of identifying connections between branches, or *vertices*, is similar to the thinning process. A morphological hit-or-miss procedure is used, along with 3x3 templates used to identify patterns such as those shown in Figure 3.9. The pixel coordinates of the node point, the central pixel in the 3x3 neighbourhood, is stored for further processing. Figure 3.10 shows the skeleton from Figure 3.8 with vertex points superimposed.

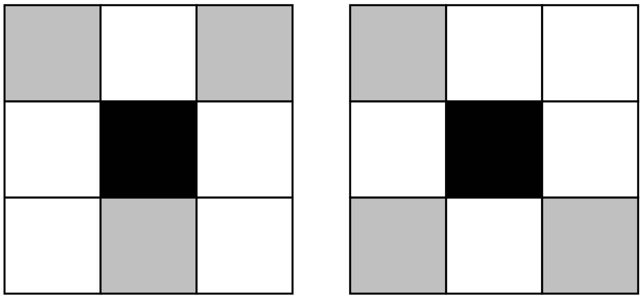


Figure 3.9. Vertex identification

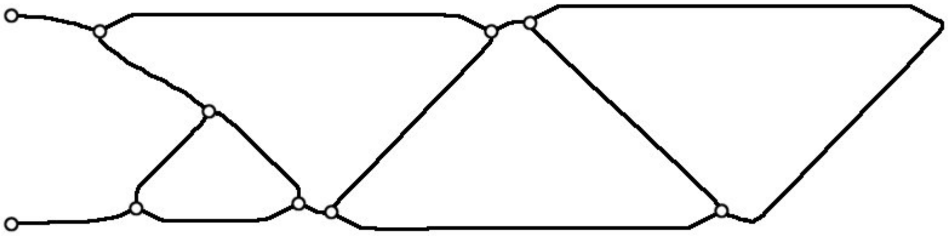
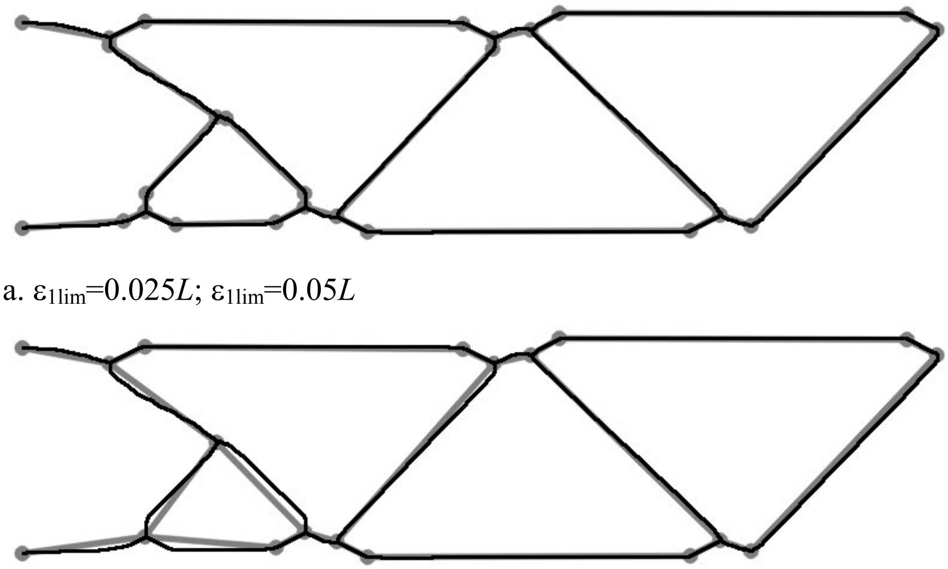


Figure 3.10. Skeleton with vertices identified

3.2.4.3. Path Tracing

Once the vertex points defining intersection points and endpoints of paths are identified, the paths are processed. Starting at a vertex, each path leading from that vertex is traced, adding the coordinate vector $\boldsymbol{p_i}=(x_i,y_i)$ of every pixel on the path to an ordered set $PA=\{p_1,p_2,...,p_n\}$. Each path is terminated when a pixel previously identified as a vertex has been reached. Pixels are marked as the paths are traced to prevent paths from being retraced.



a. $\varepsilon_{1\text{lim}}=0.025L$; $\varepsilon_{1\text{lim}}=0.05L$

b. $\varepsilon_{1\text{lim}}=0.025L$; $\varepsilon_{1\text{lim}}=0.25L$

Figure 3.11. Vector approximation to skeleton

The representation of paths as the set of all discrete points on the path is memory intensive and cumbersome. A more compact path representation that preserves the critical information in the skeleton is sought. Each path is processed using a *split and merge algorithm*, which approximates the path using piecewise continuous line segments. Two parameters control the accuracy of the fitting, as shown in Figure 3.11. The first stage in the split and merge algorithm is a recursive splitting process. For each point in the path, the distance ε_1 between the point and a line joining the path endpoints is computed. If the maximum distance $\varepsilon_{1\text{max}}$ exceeds a threshold $\varepsilon_{1\text{lim}}$, a vertex is added at the midpoint of the path, creating two subpaths. The subpaths are processed in a recursive manner, calculating the maximum deviation from a line segment, and dividing the subpaths if necessary. The splitting stage is complete when all the fitted line segments deviate from the path by less than $\varepsilon_{1\text{lim}}$. The next stage of the fitting algorithm, merging, attempts to reduce the number of line segments by merging nearly colinear adjacent segments. Pairs of adjacent segments are analyzed, and the distance ε_2 between the common vertex and a straight line segment is calculated. If the distance ε_2 is less than a threshold $\varepsilon_{2\text{lim}}$, the common vertex is deleted. The process continues until $\varepsilon_2 \leq \varepsilon_{2\text{lim}}$ for all pairs of adjacent segments. Figure 3.11 shows the results of the algorithm using the skeleton in the sample problem. The fitting parameters $\varepsilon_{1\text{lim}}$ and $\varepsilon_{2\text{lim}}$ are specified in terms of the image dimension *L*, which is the image width or height, whichever is larger. In total, there are three parameters which control the curve fitting: $\varepsilon_{1\text{lim}}$, $\varepsilon_{2\text{lim}}$, and a path curvature limit $(\varepsilon_1/d)_{\text{lim}}$. The path curvature is calculated as the ratio of ε_1 to the segment length *d*, and is designed to prevent the elimination of short, highly curved paths. For both examples in Figure 3.11, $(\varepsilon_1/d)_{\text{lim}}$ is set to 0.4.

As shown in Figure 3.11, the paths making up the skeleton can be closely approximated with a relatively small number of additional vertices. The skeleton in vector format is a compact representation that preserves the critical content of the original skeleton.

3.2.4.4. Distance Map

The Euclidean distance map (EDM) is a grayscale image in which the pixel intensity reflects the distance the pixel lies from the nearest boundary. Figure 3.12 shows the EDM of the binary image in Figure 3.5, calculated using the algorithm of [Borgefors, 1986].

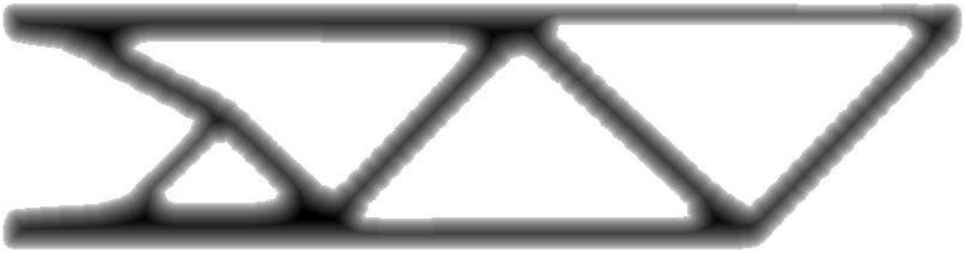


Figure 3.12. Distance transform of binary image

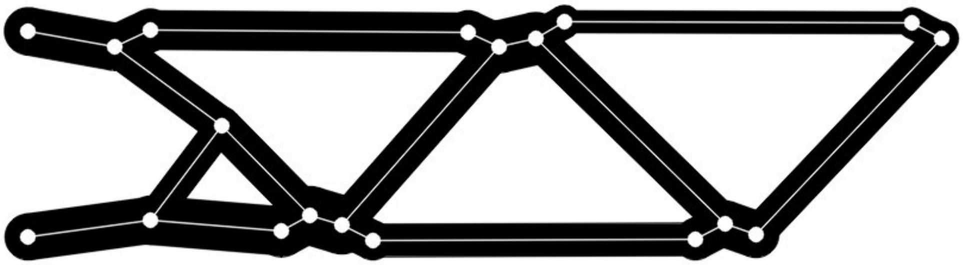


Figure 3.13. Vector skeleton with element thickness superimposed

In the framework, the EDM is used to augment the vector skeleton with width information. The process for extracting segment width information from the EDM is described here. The path information in pixel coordinate format (*PA*) is overlaid on the EDM, and the segment width at each pixel along the path is retrieved. The average path width is calculated and stored as a property of the line segments that make up the vector skeleton. Figure 3.13 shows the sample design problem with line segment width information superimposed on the vector skeleton.

3.2.5. Relational Graph

Given a skeleton in vector form, the skeleton is easily converted into a relational graph *G=(V,E)*. The set of graph vertices *V* corresponds directly to the vertices of the skeleton, and the set of edges *E* consists of the segments connecting pairs of vertices. At this stage, the elements have a single attribute, the segment width. The attributes of the vertices are the coordinate vector $\boldsymbol{p_i}=(x_i,y_i)$, and the vertex *type*, which reflects boundary conditions applied at the vertex. There are three types: *fixity*, representing displacement boundary conditions; *force*, indicating applied forces; and *default*, where no boundary conditions are applied. A graph of the vector skeleton developed for the sample problem is shown in Figure 3.14.

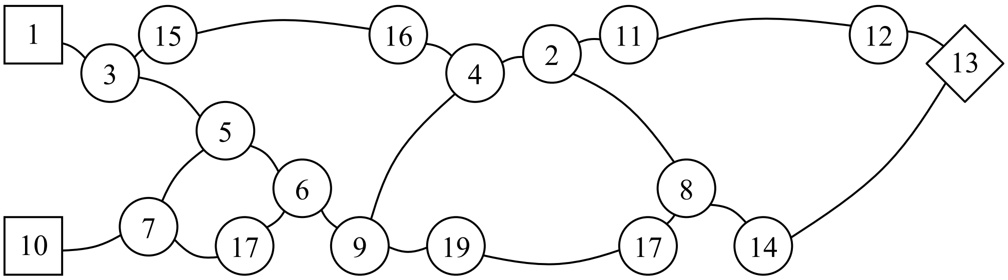


Figure 3.14. Relational graph of vector skeleton

In Figure 3.14, nodes with fixity are drawn as squares, and the node with force applied is shown as a diamond, while default nodes are circles. The vertices are assigned types using information from the domain definition. The entities *BoundPoints*, *BoundLines*, *ForcePoints* and *ForceLines* contain the coordinates and values of boundary conditions applied at points and along lines. Through the topology optimization and skeletonization processes, vertices do not necessarily occur at the same coordinates as the points where boundary conditions are defined. Point boundary conditions are associated with vertices by locating the closest vertex to the point. Line boundary conditions are matched to vertices by first finding the closest vertex, calculating the distance from that vertex to the line, then finding all vertices within that distance, plus or minus a specified tolerance.

3.2.6. Discussion

The relational graph is a relatively compact representation that retains the essential geometry and connectivity of the topology image. The binary image of Figure 3.5 has a size of 840x240 pixels, requiring about 24 kB of storage (840 x 240 x 1 bit/pixel / 1024).

The vector skeleton, consisting of 19 vertices and 22 segments, requires only 1.11 kB (assuming 16-bit integers for vertex coordinates, and 8-bit integers for vertex numbers). The storage requirements for the graph are therefore only about 5% of the requirements for the binary image. Although storage and processing requirements are less for the more compact graph format, the main advantage is that the topology image is now represented at a higher level of abstraction, one more suited to pattern matching and other reasoning techniques.

The vector skeleton shown in Figure 3.13 still appears to be relatively complicated compared to the topology optimization output. There are a number of shorter segments and kinks that are remnants of the thinning process. These features appear to have no structural benefit, and further investigation is required to see if the structure can be simplified further. This is the goal of the Refinement Phase, the next phase in the framework.

The framework follows an essentially top-down process, where a structural form is first synthesized by removing parts of the design domain. The form is then decomposed or deconstructed through subsequent phases of the framework. It will be illustrated that the decomposition has two purposes: most importantly, it supports a formal representation, but it also provides insight into how the form will eventually be produced. The formal representation is composed of primitives and the relationships between them. At a high level this supports reasoning using linguistic-based methods, such as syntactic pattern recognition. The decomposition into primitives mirrors the fabrication process, where pieces are assembled to make the whole structure.

3.3. Refinement Phase

In the Refinement Phase, the objective is to improve the structural skeleton generated in the previous phase. Techniques of mathematical optimization will be used extensively in the search for a new layout. There are two major guiding principles during the Refinement Phase: *simplicity* and *stability*. These are high level concepts that are often implicit in structural design.

Cost minimization is an objective of almost all structural design work. The cost associated with a design is strongly driven by its complexity, through all phases of design. More complex structures take longer to design, detail, fabricate, handle and install, leading to greater labour costs than simpler structures. The more complex the structure, the greater the probability of errors and rework.

Stability must be considered in the design of all structures. Stability is of special concern when generating structural configurations using topology optimization methods. Topology optimization is known to produce configurations, than when interpreted as a truss, are stable only under the given set of loads. When using topology optimization, an early and systematic consideration of overall stability is warranted.

In the Refinement Phase, the structural skeleton from the previous phase is simplified using a geometric optimization procedure, and minimum stability requirements are assessed using a separate optimization method. Both procedures are implemented using nonlinear mathematical programming techniques, supported by finite element analysis.

3.3.1. Finite Element Modeling

The relational graph constructed in the previous phase is readily transformed into a finite element model. The vertices correspond directly to nodes, and edges correspond to elements. Boundary conditions were matched to vertices in the previous stage, and these can in turn be applied to the finite element model.

The purpose of using finite element modeling in this context is quantitative conceptual design evaluation. The results of finite element analysis, together with other quantitative and qualitative information, are later used by the designer to compare multiple concepts (see Section 3.5). The detail level of the modeling, as well as the analysis procedures, reflect the level of detail available during the conceptual design stage. Although finite element analysis is a tool that is often used at the most detailed levels of design, it is also useful in earlier stages of design. Finite element methods are not essential to the refinement stage, however, and approximate analysis methods could be used as well.

Initially, beam elements are used for all elements of the structure, since pin-ended truss elements may lead to an unstable structure. During stability analysis (Section 3.3.3), various combinations of beam and truss elements are studied. During the refinement phase, optimization is performed using comparative compliance analysis. This means that absolute values of deflection, force and stress are not evaluated. The element cross-sections during refinement are assumed to have unit width, and depth equal to the segment width obtained from the Euclidean distance map.

3.3.2. Layout Optimization

An overview of the refinement phase is given in Figure 3.15. In the Layout Optimization Phase, a high resolution finite element model (FEM) is simplified into a low resolution model. The high resolution model corresponds to the vector skeleton resulting from the Synthesis Phase. During Layout Optimization, truss geometry optimization is alternated with topology modification. In geometry optimization, the design variables are node positions. Once geometric optimization reaches a solution, the resulting configuration of nodes and elements is examined. Closely spaced nodes are merged, and nodes that join nearly colinear elements are deleted, thus changing the connectivity, or topology of the structure.

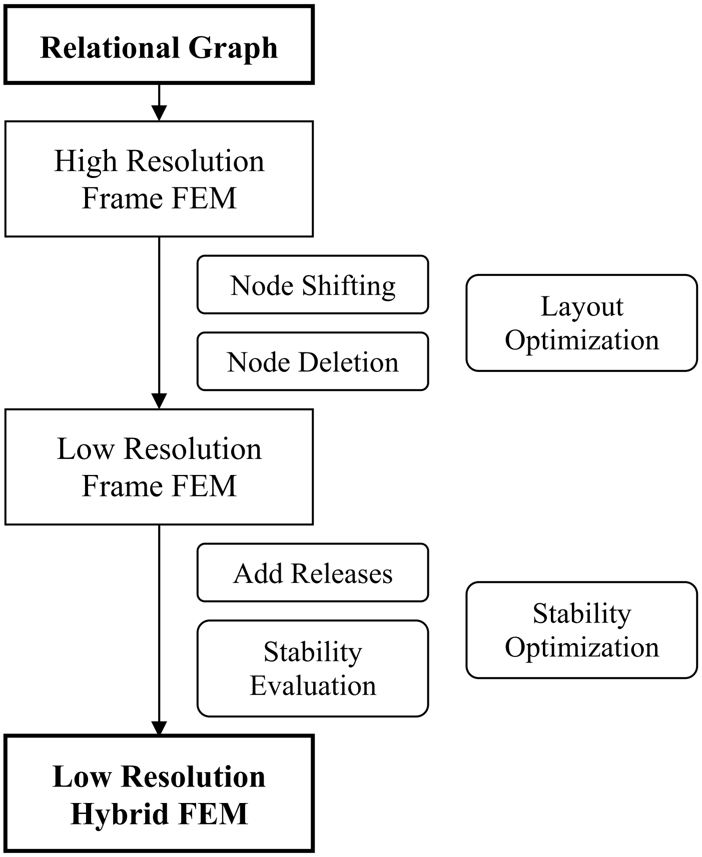


Figure 3.15. Framework overview – Refinement Phase

The geometric optimization problem is formulated for a plane frame (c=2 dimensions) consisting of m elements and N node points, with d=3 displacement coordinates at each node. Support conditions are given in the form of s_s fixed nodal displacement directions. External nodal forces are applied in s_f nodal displacement directions. Analysis is conducted in a global reduced set of displacement coordinates, with $n_1 = N \cdot d - s_s$ degrees of freedom. The design variables are the subset of nodal coordinates with no applied boundary conditions,

$$x = \{x_1, x_2, ..., x_{n_2}\}, \qquad n_2 = N \cdot d - s_s - s_f. \qquad (3.4)$$

Constraints are applied to limit node positions to within the reference domain. In the simplest case in two dimensions, with no subtracted areas, the nodal coordinates are limited to within a rectangle with width *Domain.Width* and height *Domain.Height*. Formulating these constraints as inequalities,

$$\begin{aligned}
&-x_i \le 0, && i = 1,2,...,n_2; \\
&x_i \le Domain.Width, && i = 1,2,...N - s_s - s_f; \\
&x_i \le Domain.Height, && i = N - s_s - s_f + 1,...,n_2.
\end{aligned} \tag{3.5}$$

More complex geometric constraints may be specified by removing areas from the domain with the *SubtractedAreas* field.

The objective of the optimization is to minimize the compliance of the structure. Compliance minimization is formulated as

$$\min \quad W(x) = f^T u, \tag{3.6}$$

where *f* is external static load vector applied at the node points, and *u* is the nodal displacement vector. Nodal displacements are calculated using the stiffness method. The stiffness matrix is built in global reduced coordinates using the standard assumptions of linear elastic material, linearized strain, and small displacements:

$$K(x) = \sum_{i=1}^{m} k(A_i, I_i, x) \tag{3.7}$$

where $k(E,A_i,I_i,x)$ are the local stiffness matrices for each element. Material is defined by Young's modulus *E*. Each element has cross-sectional area A_i and moment of inertia I_i. Nodal displacements are calculated by solving the equation

$$K(x)u = f \tag{3.8}$$

A common formulation for geometric optimization is to minimize compliance subject to a constraint on the volume *V*, where

$$V = \sum_{i=1}^{m} A_i L_i(x), \tag{3.9}$$

for elements with length $L_i(x)$.

In the framework, an alternative problem formulation is used where compliance and volume are both included in the objective function:

$$\min \frac{W(x)}{W_o} \frac{V(x)}{V_o} \tag{3.10}$$

where W_o and V_o are the compliance and volume of the initial structural configuration. This formulation allows the volume of the structure to increase, provided the increase is offset by a comparable decrease in compliance.

The problem as formulated is a constrained nonlinear optimization problem. In the framework, the Sequential Quadratic Programming (SQP) method is used to solve the problem. The principal idea behind SQP is the formulation of a Quadratic Programming (QP) subproblem based on a quadratic approximation of the Lagrangian function. An overview of SQP is given by [Gill et al., 1981; Powell, 1983; and Fletcher, 1987]. The framework uses the MATLAB implementation of SQP. There are three main stages to the implementation. First, at each major iteration a positive definite quasi-Newton approximation of the Hessian is calculated using the method of Broydon, Fletcher, Goldfarb and Shanno (BFGS). The QP problem is then solved using an active set strategy (also known as a projection method), following the approach of [Gill et al., 1984; 1991]. Finally, a line search is conducted in which a step length is determined that produces a sufficient decrease in a merit function. The merit function in this implementation is due to [Han, 1977] and [Powell, 1978].

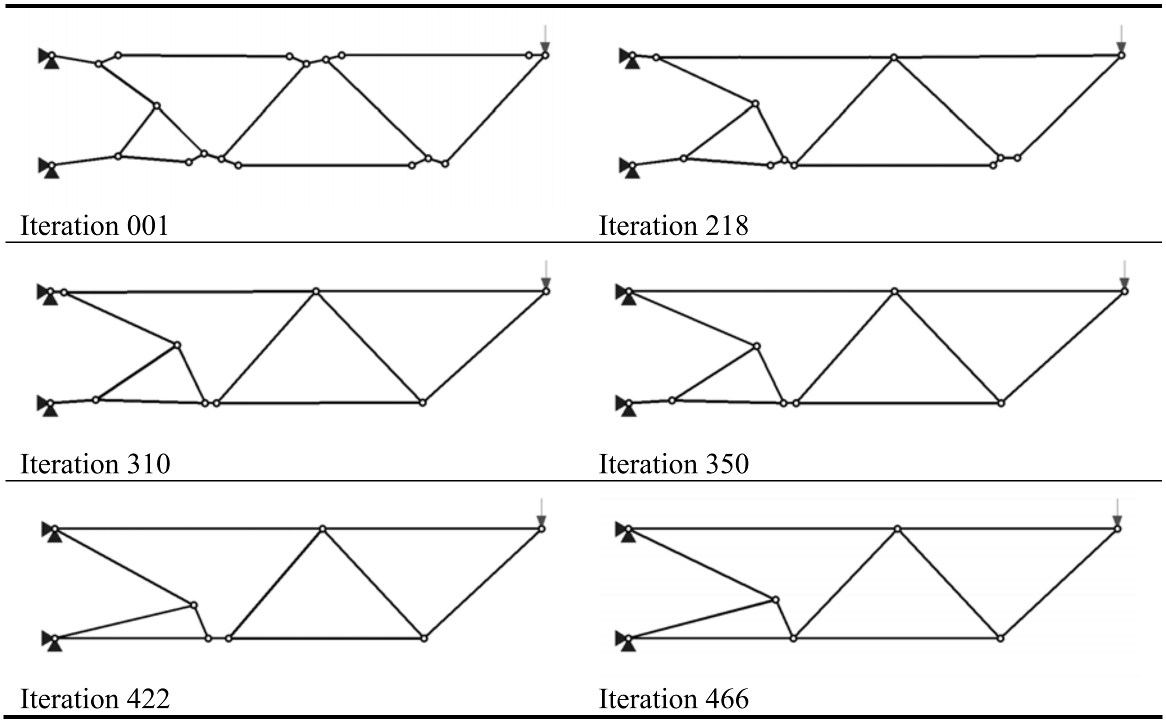
An overview of the Layout Optimization process is given in Algorithm 1. At each iteration step of the optimization, the minimum distance between any pair of nodes, d_{min}, is calculated. If d_{min} is less than a specified tolerance d_{lim}, then the optimization loop is terminated. If no nodes are closer than d_{lim}, optimization continues until first order optimality conditions are satisfied.

Following geometric optimization, nodes and elements are examined to identify two possible conditions: closely spaced nodes, and nearly colinear elements. Nodes that connect exactly two elements are inspected, and the angular distance between the element segments is computed. If the elements are colinear, within a tolerance θ_{lim}, the node and the shorter element are deleted. The structure is again examined for colinear elements, and the node deletion process continues until no pairs of elements are within the colinearity tolerance. Next, the distances between all pairs of nodes is calculated. Nodes closer than a specified distance d_{lim} are merged together, deleting one of the nodes along with the element between the nodes. The process is continued until no pair of nodes in the structure is closer together than $d_{lim.}$

The Layout Optimization process is illustrated using the sample problem in Figure 3.16. As the optimization progresses, nodes tend towards the domain boundaries, and towards each other. The limiting distance for node merging is specified as $0.9w_{min}$, where w_{min} is the minimum width of any element. The colinearity tolerance is assumed to be 0.2 radians (about 11.5 degrees). The resulting structure is shown in the bottom right hand corner of Figure 3.16. For these parameters, the number of nodes is reduced from 19 to 7, and the number of elements from 22 to 10, compared to the input structure.

Algorithm 1 Layout Optimization

```
assign nodal coordinates to design variable vector $x$
calculate the initial volume $V_o=\Sigma A_i L_i(x)$
solve $K(x)u=f$ for $u$
calculate the initial compliance $W_o=f^T u$
$status$ = 0
repeat
    assign updated nodal coordinates to design variable vector $x$
    set constraints on $x_i$ to lie within domain (0:$width$,0:$height$)
    sub min $W(x)V(x)/(W_oV_o)$ using SQP; computing at each step:
        $V=\Sigma A_i L_i(x)$
        $K(x)u=f$ ; solve for $u$
        $W=f^T u$
        $W(x)V(x)/(W_oV_o)$
        minimum separation $d_{min}$ between any 2 nodes
        if $d_{min} < d_{lim}$ then terminate optimization
    end sub
    repeat
        $ndelete$=0
        for all nodes connecting 2 elements do
            if angle between elements < $\theta_{lim}$ then
                delete node
                $ndelete$ = $ndelete$ + 1
                $status$ = 1
            end
        end for
    while $ndelete$ > 0
    repeat
        $ndelete$=0
        for all nodes closer than $d_{lim}$ do
            merge nodes
            $ndelete$ = $ndelete$ + 1
            $status$ = 1
        end for
    while $ndelete$ > 0
while $status$ > 0
```



Iteration 001

Iteration 218

Iteration 310

Iteration 350

Iteration 422

Iteration 466

Figure 3.16. Geometric optimization - pin support condition (*d_{lim}*=0.9*w_{min}*, θ_{lim}=0.2)

The optimization history is shown in Figure 3.17 and 3.18. In Figure 3.17, the cantilever tip displacement history is plotted against the left hand scale, and the structure volume is plotted on the right. The displacement decreases by about 18% and the volume increases by around 7%. The objective function decreases from a value of 1.0 to 0.88, a decrease of 12%. The objective function value is shown to decrease steadily, except at points where nodes or elements are deleted.

A second example of geometric optimization is shown in Figure 3.19. This case is identical to the previous one, except that rotational constraints have been added to the existing support nodes. The resulting structure is similar, except that two diagonals that were previously connected to the support node now connect to a node some distance from the support. The objective function value decreases by 14% over the optimization, so the structure is lighter than the pin support case, as expected.

The use of alternating topology modification and geometric optimization is essentially a heuristic optimization method, and does not guarantee a globally optimal solution. Nonetheless, a good argument can still be made that the method is suitable in this application, where it is applied after continuum topology optimization, and in the conceptual design stage. Continuum topology optimization has performed some level of global optimization, and the frame layout optimization can be viewed as a refinement to this solution.

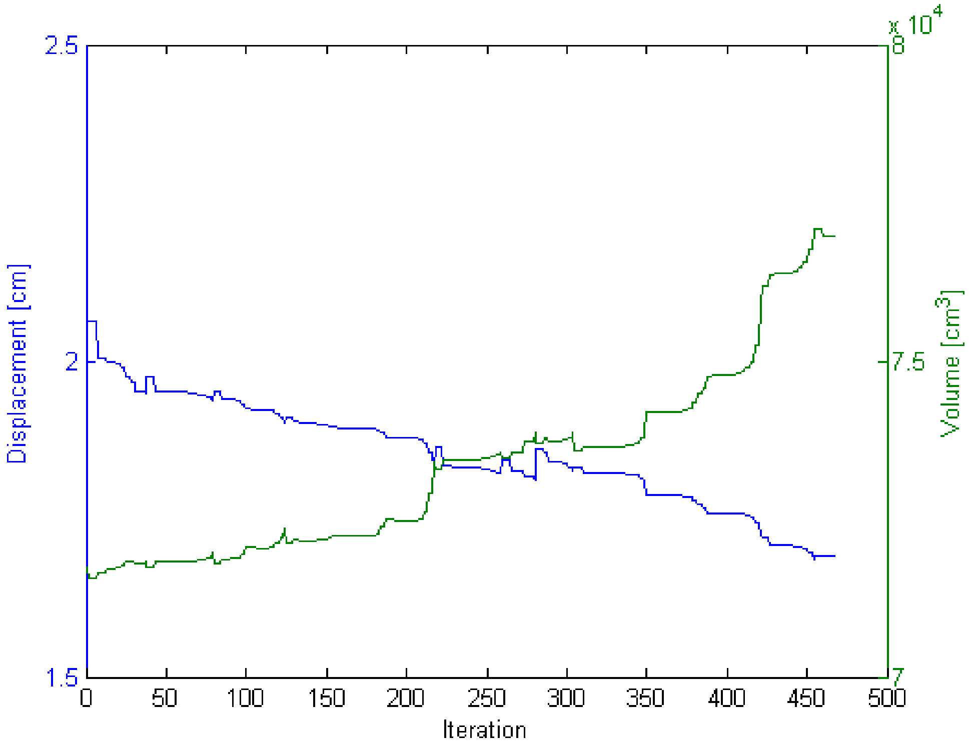


Figure 3.17. Geometric optimization – displacement and volume history

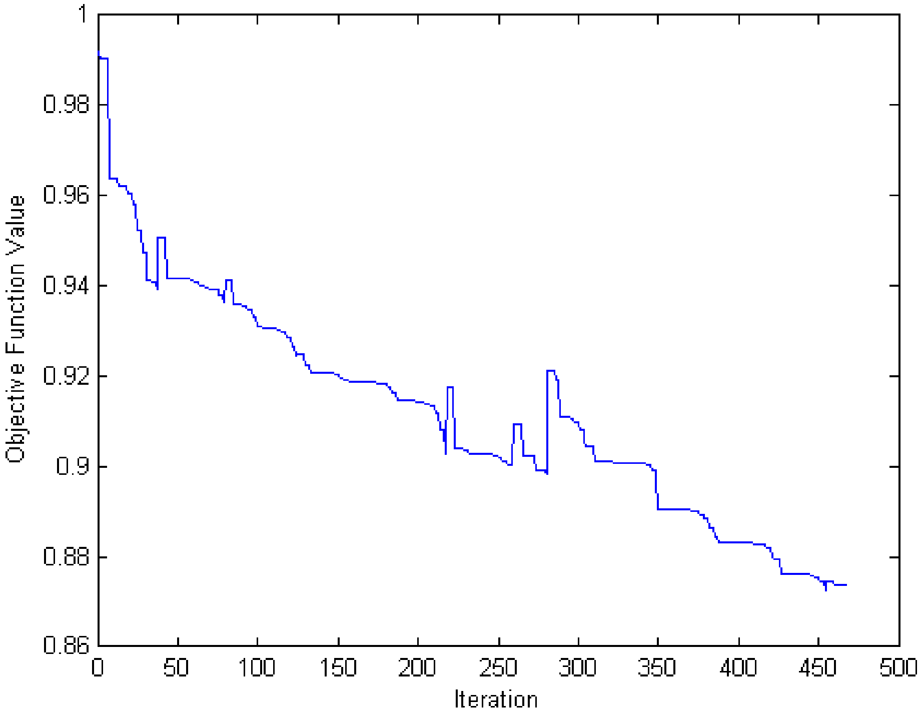


Figure 3.18. Geometric optimization – objective function history

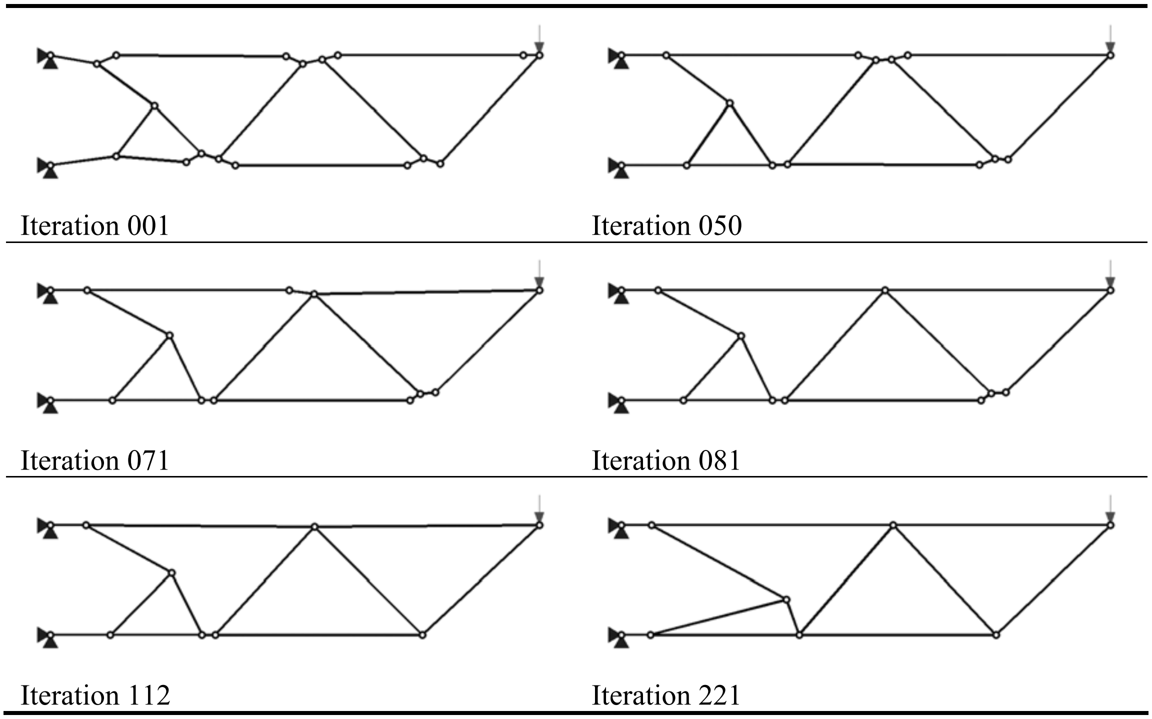


Figure 3.19. Cantilever beam geometric optimization (fixed support condition)

The governing principle followed in Layout Optimization is that of simplification, or:

> *If two structural configurations have similar performance, the configuration with fewer elements is preferred.*

As indicated previously, simplification affects many aspects of the overall cost of a design. If performance can be maintained while achieved greater simplicity, this is viewed as the improvement of a design concept.

For the two examples used to illustrate layout optimization, considerable simplification was achieved. Such results cannot be expected in all cases with a single set of algorithm parameters. For example, the colinearity tolerance may need to be adjusted to avoid straightening arch segment by deleting intermediate nodes. Optimal structures such as Michell trusses often contain curved elements.

3.3.3. Stability Optimization

Interpreting the results of topology optimization as a truss often results in an unstable structure, particularly where a single load case has been analyzed. Using the classical definition, a truss consists of axially-loaded members joined with frictionless pins. In contrast, a frame is a structure where the members transmit bending moments as well as axial loads. In the literature, structures with a combination of bending members and axial members are sometimes called hybrid structures. This will be the convention used in this framework.

In other applications where continuum topology optimization is processed as skeletal structures [e.g. Chirehdast et al., 1992], stability of truss structures is ensured by adding members using heuristics. In contrast, the method in this framework is to formulate the stability problem as an optimization problem. The input to the problem is a frame structure. The objective of the optimization is to find the maximum number of moment releases that can be introduced in the frame while still ensuring stability. The result of the optimization is thus a hybrid structure with both moment and pin connections.

There are several reasons for formulating the stability problem as an optimization problem. First, the topology optimization results may be closely approximated, while still ensuring stability. One of the strengths of topology optimization is the ability to restrict the domain to an arbitrary form, and to generate a structural solution that complies with a relatively complex set of geometric constraints. For example, if a large void space is required in a domain, it is not possible to simply add a brace that crosses the void to ensure stability. As geometric constraints are added to a domain, it is more likely that the optimal solution will use beam elements as opposed to truss elements. A second reason for using a stability optimization method is that frame structures are generally stiffer than truss structures, for a given amount of material. Finally, most practical structures are hybrid or frame structures, as opposed to classical trusses. Real structures use connections with various levels of rotational stiffness. The overall cost of fabricating and installing the structure is largely determined by the connections, and the cost of connections is closely related to the level of rotational stiffness provided. In general, but not always, connections with low rotational stiffness are more cost effective than connections with high stiffness. A trivial exception is given by a truss with straight chords of uniform cross-section. In this case a classical truss implies pin connections between chord member segments, where it is obviously more economical to use a continuous, single member. The degree of stiffness provided by connections also has implications for erection of the structure. Connections with low stiffness may require more bracing and falsework to provide stability during erection, resulting in higher overall costs than if stiffer connections were used.

Stability optimization generally produces a structure that is just stable. In other words, if one release were to be added the structure would lose stability. In subsequent design phases, some of the added joint releases could be eliminated, without affecting overall stability. The stability optimization thus provides the designer with the minimum stiffness requirements for the joints, giving them the freedom to select from a range of different connection rigidities and types, depending on cost and fabrication criteria.

To illustrate the challenges in stability optimization, the sample cantilever problem is studied using combinatorial optimization. In this study, all elements of the structure are modeled using beam finite elements with end moment releases. Four different instances of the element are available: fixed at both ends (fix-fix), fixed at one end and released at the other (fix-pin and pin-fix), and released at both ends, like a truss element (pin-pin). Using as input the result of geometric optimization in Figure 3.19, the input structure has 12 elements. Evaluating all different combinations of the four element types would require 4^{12}, or $1.7\text{x}10^{7}$ evaluations, a huge number for such a simple structure. To reduce the complexity of the problem, using the two element types with identical conditions at each end (fix-fix and pin-pin) results in $2^{12} = 4096$ combinations.

Stability is assessed by constructing the stiffness matrix K and computing the Singular Value Decomposition (SVD) of K:

$$\boldsymbol{K} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^{\boldsymbol{T}} \tag{3.11}$$

where $\boldsymbol{U}=\{u_1,u_2,...,u_{3j-c}\}$ consists of a set of left singular vectors, $\boldsymbol{V}=\{v_1,v_2,...,v_b\}$ contains a set of right singular vectors, and a set of singular values is found in the first r non-zero diagonal elements of $\boldsymbol{\Sigma}$. The *condition number* is calculated as the ratio of the largest diagonal element of $\boldsymbol{\Sigma}$ to the smallest. Large condition numbers indicate a nearly singular stiffness matrix, implying that the structure contains an internal or rigid body mechanism. Note that stability can also be determined by just using the equilibrium matrix, instead of the full stiffness matrix (as indicated in Section 2.5.4).

The results of the combinatorial stability evaluation of the sample cantilever problem are shown in Figure 3.20. Stiffness matrices were constructed for each of the 4096 different combinations of fix-fix and pin-pin members. The condition number for each configuration was calculated, and values over $1\text{x}10^{16}$ were labeled unstable. Nine stable configurations resulted, and these are all shown in Figure 3.20. Pin-ended members are indicated by arcs at both ends of the member. Note that each node is connected to at least one fixed-ended element, in order to ensure rotational equilibrium of the node. The combinatorial approach provides some insight into the stability problem, however it is inefficient. A solution to the inefficiency is to formulate the problem using continuous design variables rather than discrete variables. This alternative approach is described in the following paragraphs.

The stability optimization problem is next formulated with continuous design variables representing the member end release conditions. For a plane frame consisting of m elements, the design variables are parameters representing the degree of release at the two member ends,

$$r = \{r_1, r_2, ..., r_{2m}\}, \quad r_i \in \Re. \tag{3.12}$$

The release values are constrained such that $0 \le r_i \le 1$, where r_i=0 represents a pin-end condition, and r_i=1 indicates a fixed end.

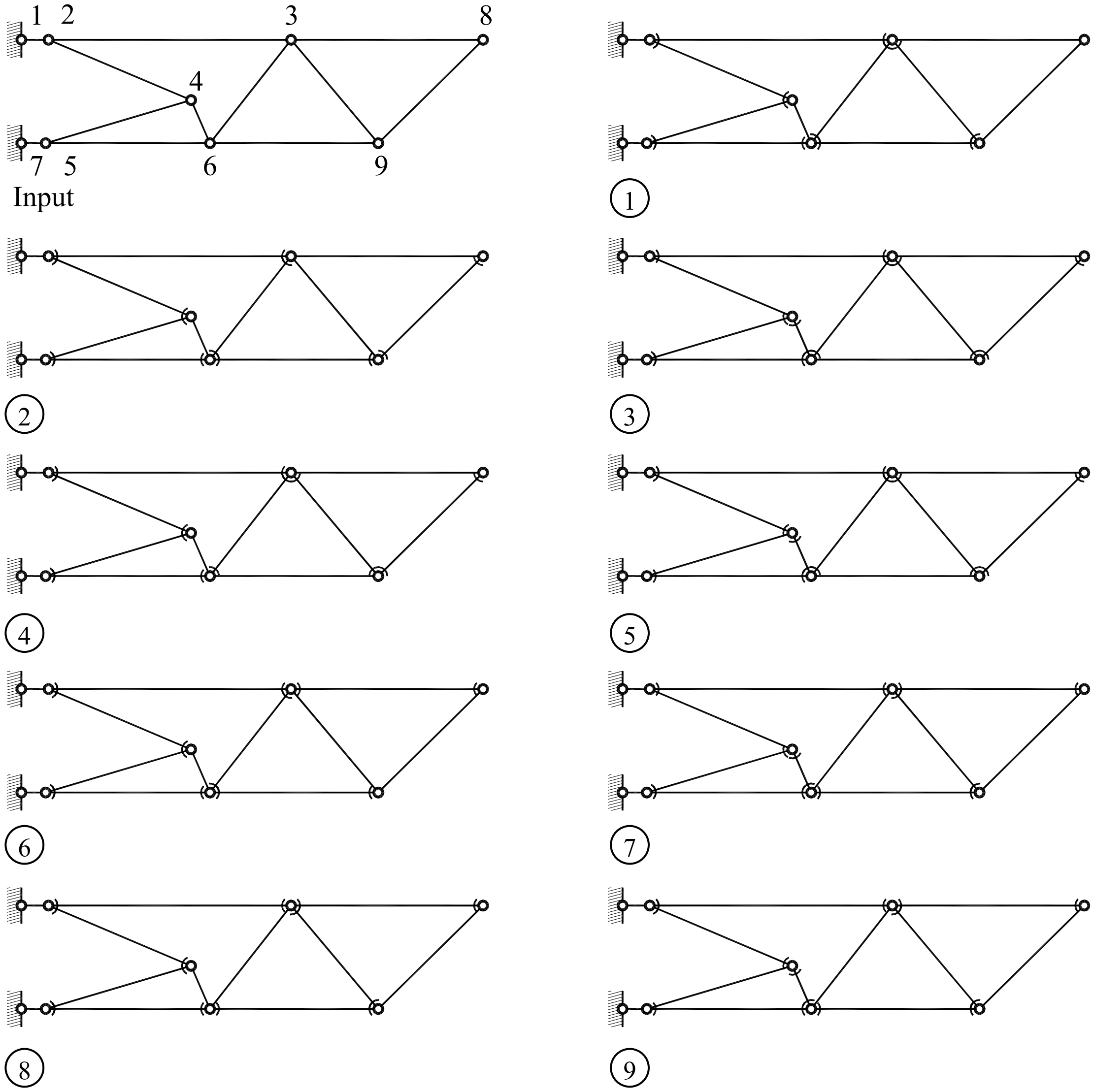


Figure 3.20. Cantilever beam combinatorial stability study

The objective of the optimization is to maximize the number of end releases in the structure or, equivalently, to find

$$\min f(r) = \sum_{i=1}^{2m} r_i \,. \tag{3.13}$$

The stiffness matrix is built in global reduced coordinates using the standard assumptions of linear elastic material, linearized strain, and small displacements:

$$K(x) = \sum_{i=1}^{m} k(A_i, I_i, x) \tag{3.14}$$

where *k(E,A*$_i$*,I*$_i$*,x)* are the local stiffness matrices for each element. The element stiffness matrix takes the form:

$$k(E,A,I,r) = \begin{bmatrix} \frac{EI}{L} r_i(3+r_j) & 2\frac{EI}{L} r_i r_j & 0 \\ 2\frac{EI}{L} r_i r_j & \frac{EI}{L} r_j(3+r_i) & 0 \\ 0 & 0 & \frac{EA}{L} \end{bmatrix} \tag{3.15}$$

where r_i represents the release at end *i* of the member, and r_j the release at the opposite end. For example, $r_i = r_j = 1$ leads to

$$k = \begin{bmatrix} 4EI/L & 2EI/L & 0 \\ 2EI/L & 4EI/L & 0 \\ 0 & 0 & EA/L \end{bmatrix}, \tag{3.16}$$

the familiar stiffness matrix for a beam fixed at both ends. Setting $r_i = r_j = 0$ gives

$$k = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & EA/L \end{bmatrix}, \tag{3.17}$$

the stiffness matrix for a beam with moments released at both ends.

The optimization problem is solved using Sequential Quadratic Programming. At each iteration, the stiffness matrix (3.14) is constructed and evaluated for stability. If the condition number of the stability matrix indicates the matrix is nonsingular, then nodal displacements are calculated by solving

$$K(x)u = f\,. \tag{3.18}$$

If the matrix is singular displacements are set to an arbitrarily large value. The nodal displacements are constrained to lie within a specified tolerance

$$|u| \le |\overline{u}| \cdot (1+\delta) \tag{3.19}$$

where $\bar{u}$ is the set of displacements for the initial frame structure, with all fixed ended beams, and is δ the tolerance.

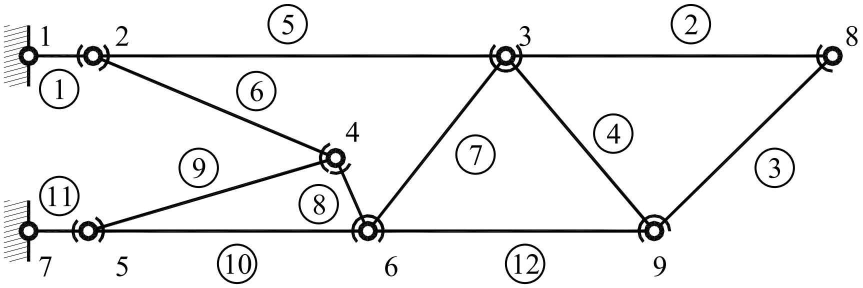


Figure 3.21. Cantilever beam SQP stability optimization

Element	r_i	r_j
1	**0.24**	4.46E-10
2	4.46E-10	4.46E-10
3	6.29E-10	4.46E-10
4	4.46E-10	4.46E-10
5	4.46E-10	4.46E-10
6	4.46E-10	4.46E-10
7	4.46E-10	4.46E-10
8	4.46E-10	4.46E-10
9	4.46E-10	4.46E-10
10	4.46E-10	4.46E-10
11	4.46E-10	**0.11**
12	4.46E-10	1.53E-08

Table 3.2. Cantilever beam stability optimization results

The results of the optimization are given in Figure 3.21 and Table 3.2. Elements 1 and 11 transfer moments at the support. Otherwise, all other elements have small numbers for their end release coefficients, suggesting pin-ended members can be used in most locations. The values of the small release coefficients are controlled by the termination tolerances for the SQP solver, and can be made smaller with a larger number of iterations. Note that a tolerance value of δ=0.05 was used for this analysis, indicating that the nodal displacements in the optimal structure are no more than 5% greater than the corresponding displacements for a frame with full end restraint.

The stability optimization routine may lead to a number of similar solutions. To limit the number of solutions, a modified objective function incorporating weighted end release coefficients is a simple extension. For example, larger member cross-sections could carry a heavier weighting factor to capture the higher costs associated with larger members. A characteristic of stability optimization is that solutions may exist where all joints have a small stiffness, just large enough to ensure a nonsingular stiffness matrix. In many cases it is preferred to have either no joint release or full release; in these cases a penalty function may be formulated to guide the release coefficients to a 0 or 1 value, similar to the SIMP method of topology optimization. Finally, it should be noted that stability optimization can produce configurations that are relatively flexible for load cases other than the one considered. Additional small magnitude loads can be used to produce structures that are both numerically stable and relatively rigid.

In a conceptual design system that also interprets the output of continuum topology optimization [Chirehdast et al., 1992], a two step process is used to ensure stability. First, a mobility detection algorithm [Chirehdast and Papalambros, 1991] is used to determine whether the structure is a mechanism. The mobility detection algorithm is based only on the geometry of the structure. If mobility is detected, two rules are used to automatically generate immobile structures: 1) nodes connected to only one other node are connected to a node where a boundary condition is applied, and 2) polygonal cells in the structure are converted to sets of triangles. Such rules are difficult to apply when there are voids in the reference domain. Also, these rules are more likely to lead to nonoptimal structures, in terms of both performance and cost, than the stability optimization method used here.

3.3.4. Relational Graph

After Layout Optimization and Stability Optimization are performed, the corresponding relational graph is updated. Vertex positions and edge connectivity is modified to reflect the results of Layout Optimization, and member end release information is added to the list of edge attributes. The updated relational graph for the sample cantilever problem is shown in Figure 3.22.

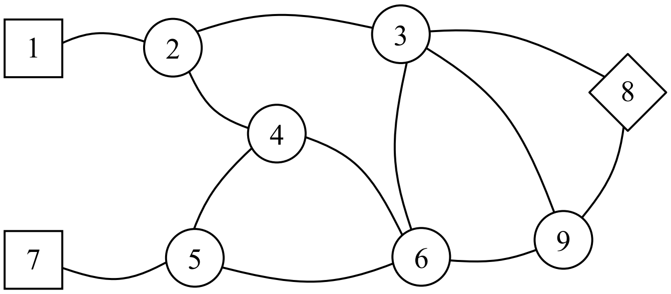


Figure 3.22. Relational graph of cantilever beam output

3.3.5. Discussion

The first two phases of the framework have used a minimal number of specific input parameters. The essential input consists of the domain area and boundary conditions, along with a set of relative load values. Layout Optimization is based on comparative analysis, seeking to improve a structure generated by topology optimization. With little input other than the geometry, Stability Optimization finds a stable configuration and determines the minimum requirements for joint stiffness to ensure stability. The output of the current phase, the Refinement Phase, includes a low resolution finite element model and a corresponding relational graph. These two representations support a range of options for further optimization and more detailed analysis.

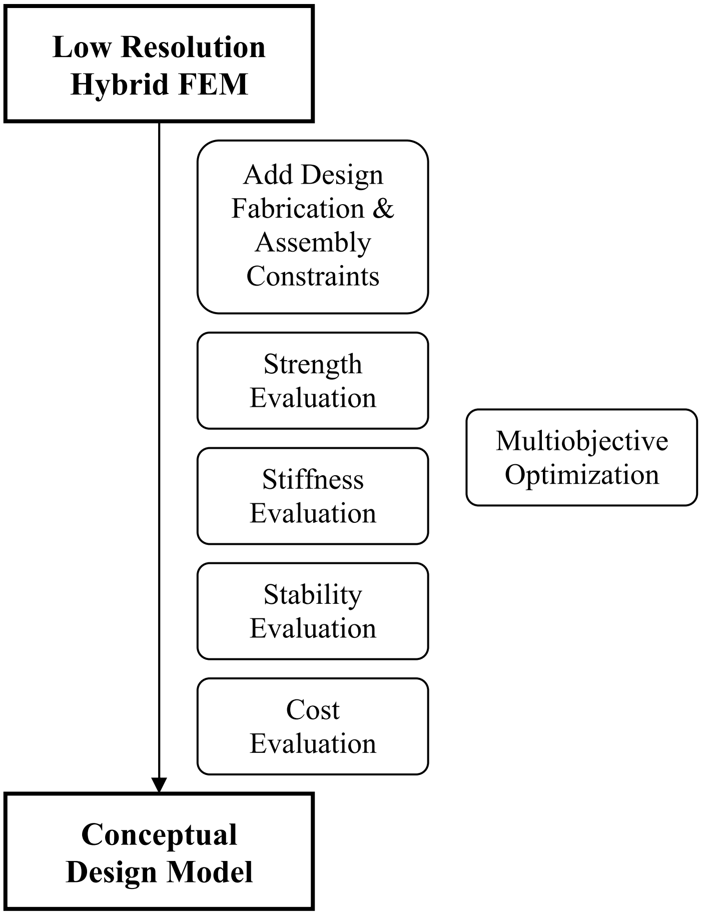

Low Resolution Hybrid FEM
Add Design Fabrication & Assembly Constraints
Strength Evaluation
Stiffness Evaluation
Stability Evaluation
Cost Evaluation
Multiobjective Optimization
Conceptual Design Model

Figure 3.23. Framework overview – Optimization Phase

3.4. Optimization Phase

The goal of the Optimization Phase is to incorporate the information required to implement the design in a specific domain. An overview of the Optimization Phase is given in Figure 3.23. While general goals have guided the design concept up to this point, more specific information relating to design, fabrication, shipping, assembly and use are required for effective conceptual design. The specific information may be formulated as the objectives and constraints of a multiobjective optimization problem. There are many ways to formulate a specific multiobjective optimization problem. This description of the

Optimization Phase uses a practical example to show how multiobjective optimization may be used in the context of the proposed framework.

The example used to illustrate multiobjective optimization is the tubular cantilever frame shown in Figure 3.24. Tubular structures such as this provide a practical example of competing, multiple objectives in structural design.

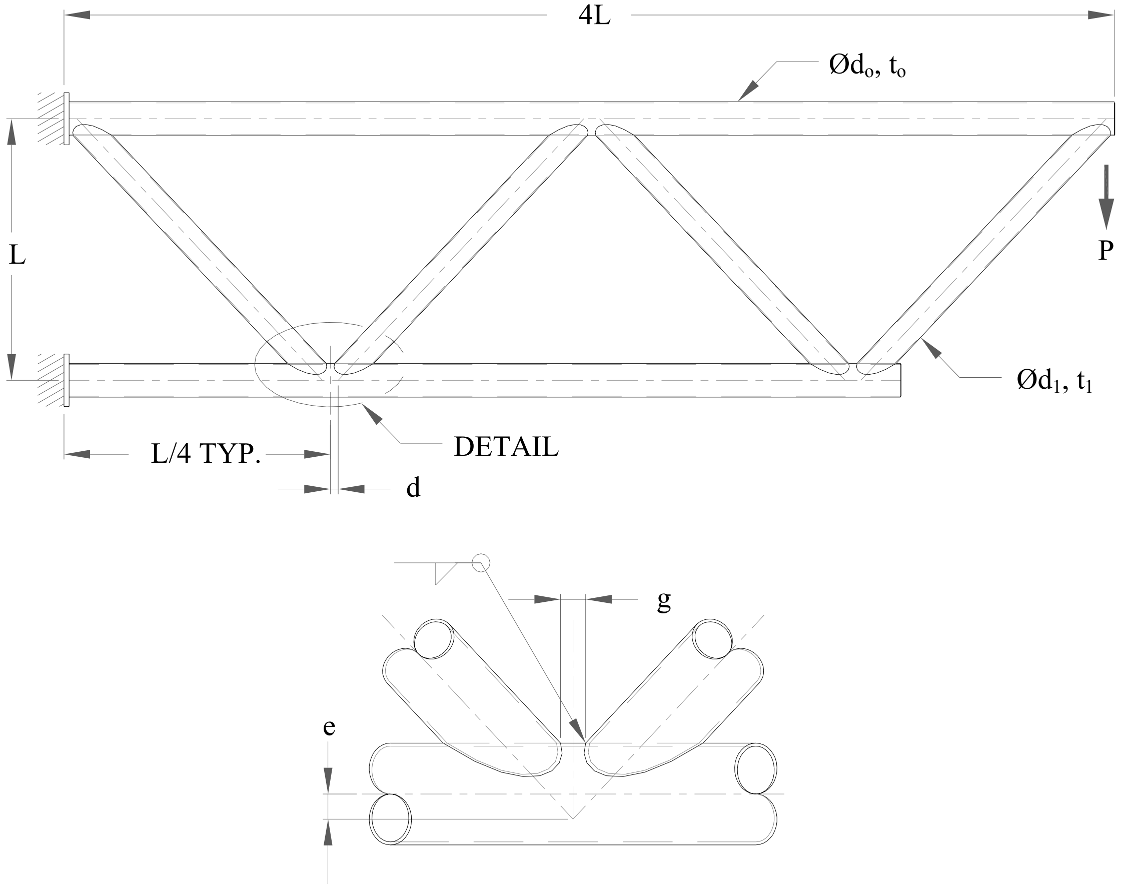


Figure 3.24. Tubular truss design problem

3.4.1. Objectives

A common objective of structural design is to minimize cost. Other objectives, depending on the domain, include maximizing performance, aesthetic values, service life, or maintenance access.

Fabrication costs are primarily a function of the labour hours required. Apparent savings by virtue of minimum material used may result in a much higher overall cost. At current (2009) shop labour rates and tubular material prices, the cost of one hour of fabrication is roughly equivalent to 40 kg of material. Some research has shown that the tonnages for different designs performing the same function vary by about 20-30%, while fabrication costs can vary by a factor of three or four [Firkins and Hemphill, 1990]. Even though such estimates are closely related to the construction type and shop setup, they emphasize the importance of considering cost during conceptual design.

The labour costs to deliver a design are not limited to costs on the shop floor. Labour costs arise in all stages of construction, including design, detailing, sourcing, handling, fitting, welding, painting, shipping and installation. The features of a design the most strongly drive labour cost are the piece count, the degree of similarity between pieces, and the accuracy requirements. The piece count generally has a direct affect on the cost, since total cost is estimated by multiplying the number of pieces by the unit cost. Similarity between pieces decreases design and detailing time, and improves efficiency and reduces fabrication errors. Geometric accuracy requirements dictate the fabrication method. More restrictive tolerances increase fabrication cost, because of additional adjustment and correction labour, additional jigs and fixtures, and higher value fabrication methods such as precision machining.

Increased simplification and a reduced piece count was one of the goals of the Refinement Phase. The Optimization Phase quantifies the number and type of elements and joints, along with the associated fabrication costs. Stability Optimization generated information on minimum joint stiffness requirements, which is useful in selecting the type of joint and the method of fabrication.

Piece similarity may be quantified in a number of different ways. One method is to generate a histogram of piece characteristics, such as their length and cross-section type. A more effective method, proposed as part of this framework, is to use a procedure similar to the Hough transform. An example is shown in Figure 3.25. The input to the procedure is a structural description, including the node coordinates and member topology. A normalized length $L=\overline{L}/S$ $(0\leq L\leq 1)$ and orientation angle θ $(0\leq\theta<\pi)$ are calculated for each member, where $\overline{L}$ is the member length and S is the largest overall dimension of the structure. An accumulator space is created in two dimensions, corresponding to normalized length and orientation angle. The length and angle ranges are discretized into bins. For simplicity, an equivalent number of bins may be chosen for the length and the orientation angle. Each member is processed, and the pair (L,θ) is used to cast a vote in accumulator space. The accumulator space is represented by an image, where the pixel intensity reflects the number of votes cast. A large number of elements with similar length and angle produce a bright spot on the accumulator image. In Figure 3.25, two different accumulator images are shown for the structure shown at the top. The image on the left uses an angular resolution of 5 degrees; the resolution in the right hand image is 10 degrees. Members 2 and 6 are similar in terms of both length and angle. In the left hand accumulator image, every member is shown as separate pixel, all of equal intensity, because no two members are close enough in angle or length to be assigned to the same bin. When the resolution is 10 degrees, members 2 and 6 are assigned to the same pixel. Similarly, member 1 and 10 share the same pixel, which appears white. The remaining pixels are gray because they each correspond to only one member. The brightest pixels thus identify the largest groups of similar members. As the resolution of the accumulator space is increased, more members appear similar. If the geometry of the structure can be modified to make the similar members identical, then the cost can be decreased. A larger accumulator resolution thus corresponds to a lower cost structure. The problem of making compatible geometric adjustments to the structure is addressed in Section 3.4.3.

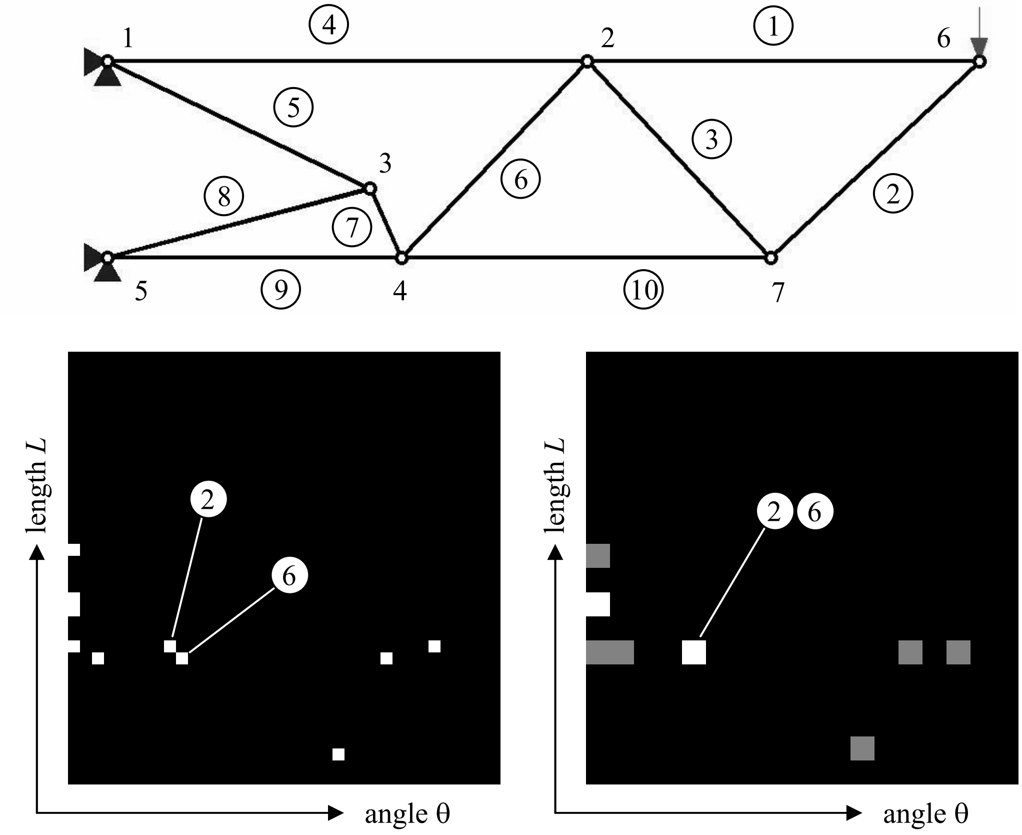


Figure 3.25. Hough style plot of element length and angle

3.4.2. Constraints

The tubular frame in Figure 3.24 is to be designed to satisfy the strength limit state, for a static load *P*. In the strength design of tubular structures, the member design is tightly coupled to the connection design. The minimum cost design of the members does not necessarily correspond to the minimum overall cost.

For minimum weight design of tubes in compression, large diameter, thin-walled sections are preferred to maximize the radius of gyration. On the other hand, thin-walled chords produce lower connection strength because of the higher probability of localized buckling and yielding of the tube wall. The wall may be reinforced with a doubler plate or other reinforcement, however such reinforcement is expensive due to the additional labour and material required. Also note that surface preparation and coating costs are higher for larger diameters.

Connection strength is a function of the layout of the member centerlines and node positions. The connection strength is generally lower for a gap connection (g >0 in Figure 3.24) than for conditions where the web members overlap. From a cost viewpoint however, gap connections are usually less expensive. Gap connections offer greater flexibility during fit-up, and welding access is easier. In overlapping connections, the overlapped member must often be welded before the overlapping member is fitted. In most shops, it is more cost effective to do final welding after all fit-up and tack welding is complete.

The strength design of tubular connections generally uses a set of empirically or theoretically derived formulae [e.g., Packer and Henderson, 2003]. Empirically derived formulae often have limited ranges of applicability because they are based on a limited number of experimental test configurations. These ranges of applicability may be applied as constraints in a multiobjective optimization. Constraints are also derived from practical limitations of the design, fabrication or installation phases.

Examples of constraints in the design and fabrication of tubular structures like the one in Figure 3.24 include [Packer and Henderson, 2003]

- Angles of less than 30 degrees between the web and chord create significant difficulties due to poor welding access
- Web to chord diameter ratio limit: $0.2 < \frac{d_1}{d_o} \leq 1.0$
- Nodal eccentricity limit: $-0.55 \leq \frac{e}{d_o} \leq 0.25$
- Overlap and gap: overlap > 25%; $g > t_1 + t_2$

A common constraint in structural design is that materials are available only in discrete sizes. Tubular shapes are produced in a limited number of diameter and wall thickness combinations. Of the set of shapes that are regularly produced, only a subset of these may be readily available due to variations in demand and production. In multiobjective optimization, discrete sizes are often represented using a continuous curve. Figure 3.26 shows a continuous curve representing the weight of common pipe sizes used in tubular construction.

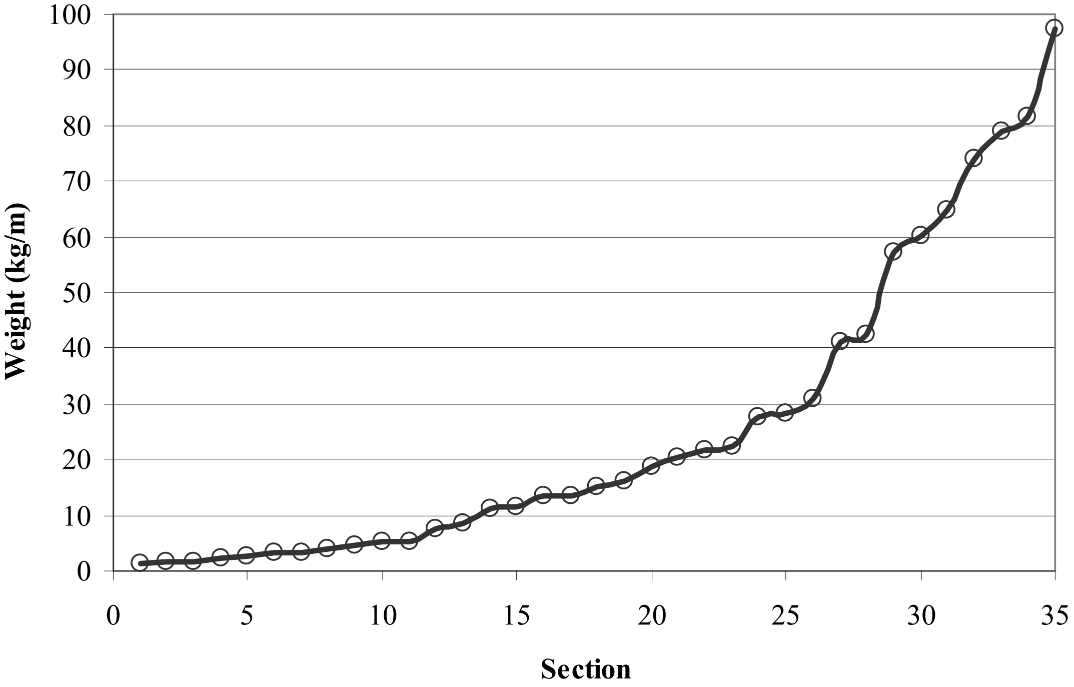


Figure 3.26. Discrete tubular sections

3.4.3. Multiobjective Optimization

For a typical multiobjective optimization problem in structural design, the design variables include node positions, member topology, member cross-sections, joint types, and the number of joints. The node positions and member connectivity have a considerable influence on the overall design. The node and member geometry limits the types of joints, and influences the overall piece count and similarity of components.

Within the framework, geometric domain constraints originating in the Synthesis Phase may be applied during multiobjective optimization. Another feature of the framework is that constraints may be inferred from similar applications using case-based reasoning, and applied to the design case. This is discussed further in Section 3.6.

A common challenge in multiobjective optimization is formulating and assessing the importance of the objective functions. Often the objectives are competing and their relative importance must be weighed in order to make tradeoffs. The relative importance of the objectives is generally not known in advance, and some analysis is usually done to assess the capabilities of a system before tradeoffs are made.

The results of multiobjective optimization are used to update the relational graph structure developed during the Synthesis and Refinement Phases. The vertex attributes are updated with node positions, and new attributes are added for joint design information. The edge attributes are updated with member cross-section information. The relational graph is used in case-based reasoning, as discussed in Section 3.6.

3.4.4. Discussion

While the form of the structure can be represented in a generic way, domain specific information is more difficult to standardize. Ideally, domain specific design procedures should be stored in a library from which they can be retrieved and reused. One method is to separate design process knowledge and data, and to store process knowledge in a hierarchically organized structure, such as in SEED [Fenves et al., 2000]. Once the knowledge is stored, protocols must be developed to maintain and update the knowledge. The authors of SEED propose a repository or brokerage for design process knowledge. Recent developments in semantic networks for mathematical problem solving may be applicable in this context [Wolfram, 2009].

In the context of this framework, multiobjective optimization is intended as an evaluation tool to assist the user in differentiating between multiple designs on the basis of fabrication cost, installation cost, and other practical criteria. Multiobjective optimization is a well established branch of optimization, and range of different tools are available. This framework does not propose any new multiobjective optimization methods. Rather, the intent here is to show how the framework generates a structural model in a form that is directly suited to domain-specific multiobjective optimization.

3.5. Evaluation and Selection Phase

During the Synthesis Phase, multiple design configurations may be generated. Multiple designs result from using different topology optimization parameters (volume fraction, filter radius, or convergence criterion), different thresholds in interpreting the topology optimization output, or through variations in the configuration of boundary conditions. The framework specifies an Evaluation and Selection Phase, during which human users evaluate and compare the designs that were generated, and select a subset of the designs for further, more detailed, design work. In Figure 3.1, the Evaluation and Selection Phase occurs after the Optimization Phase. Given the importance during conceptual design of direct control by the human user, the framework also supports early selection and reformulation of the problem, as indicated by the dashed lines in Figure 3.1.

Evaluation may be either qualitative or quantitative in nature. Quantitative measures are readily developed for structural performance and cost. For example, finite element analysis provides performance data such as stress and deflection. Cost estimates provide numerical values which may also be used to compare competing designs.

Qualitative evaluation is required when numerical values cannot easily be assigned to a design, such as when a structure is evaluated according to aesthetic criteria. Experienced designers use intuition to evaluate designs, and during conceptual design they often rank concepts based on incomplete information, before detailed numerical analysis results are available. Even if a numerical value can be assigned, most practical design situations involve multiple participants, with different sets of experience, values and tastes. For example, participants in building design include architects, structural engineers, mechanical engineers, building envelope specialists, and the owner, amoung others. In telescope design, participants include structural engineers, mechanical engineers, control system engineers, scientists, operations specialists, maintenance personnel, and fabrication specialists. Achieving agreement between multiple participants is a complicated task to automate, and is often most efficiently achieved through direct communication between participants. Evaluation is the process of comparing a design to the set of requirements, specifications, and overall expectations of the participants. If none of the candidate designs meet the required criteria, then the criteria should be reviewed and if necessary, the design problem should be reformulated. Otherwise, a subset of the candidate designs is selected for more detailed design work.

3.6. Alternate Generation Methods

The preceding description of the framework uses topology optimization methods to generate structural forms. Forms may be input to the framework using alternative methods, such as from photographs or hand sketches, as shown in Figure 3.27. Although these are not necessarily sources for “optimal” structures, their information content can be leveraged to assist the conceptual design process.

Arbitrary shapes and forms can be interpreted as structures. Images of shapes and forms generally contain no explicit information about how such a structure would be loaded or supported, or what its function would be. This information has to be supplied by the user. Using the proposed framework, high contrast or binary images can be opened, converted to a skeleton, a graph and a finite element model. At this stage, boundary conditions can be added to the model to begin optimizing the shape under structural design criteria. Converting an arbitrary shape to a graph opens up further possibilities, however, when one considers the capabilities of pattern recognition and visual case-based reasoning. The graph can be used to retrieve similar images, and with those images an associated case containing information about boundary conditions, function, behaviour, and other information. Examples of sketch input and image input are given in Figures 3.28 and Figure 3.29.

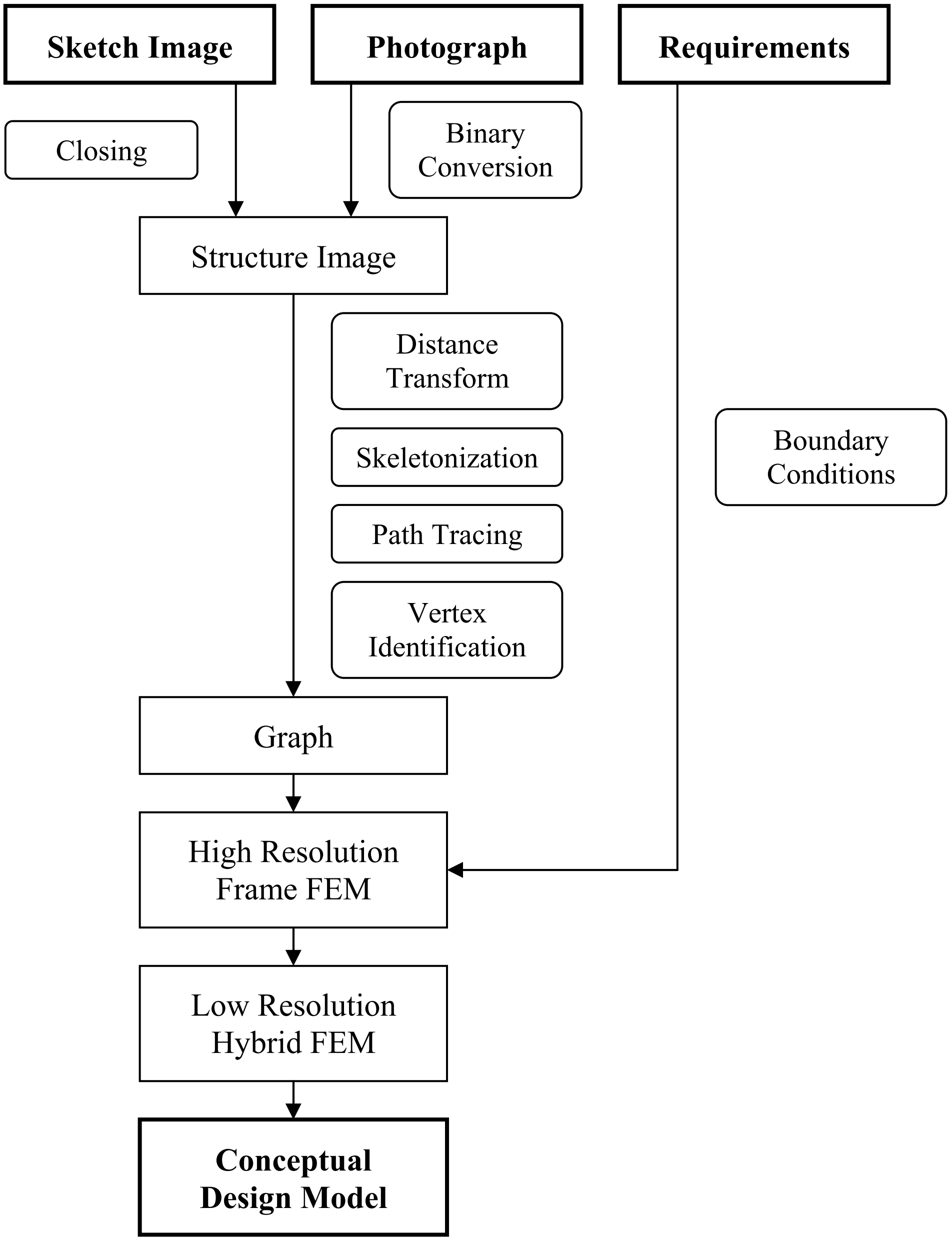


Figure 3.27. Alternate input paths

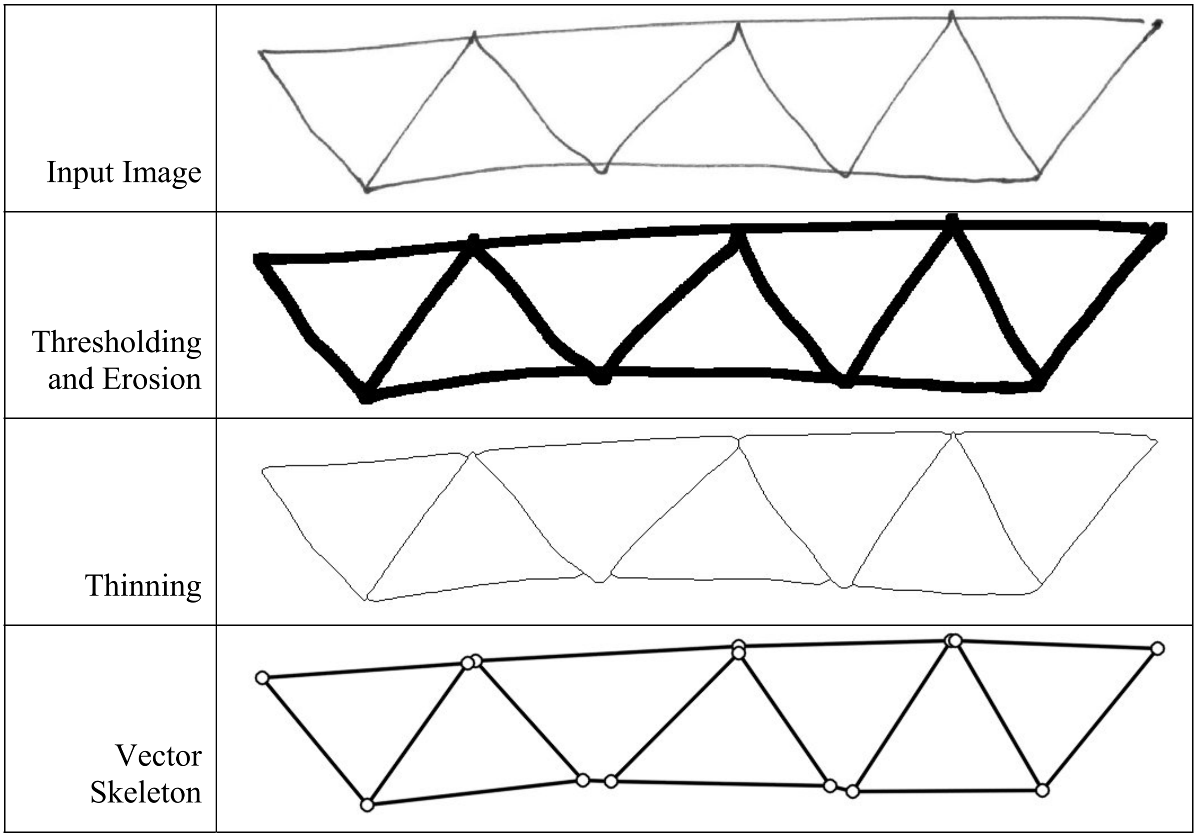
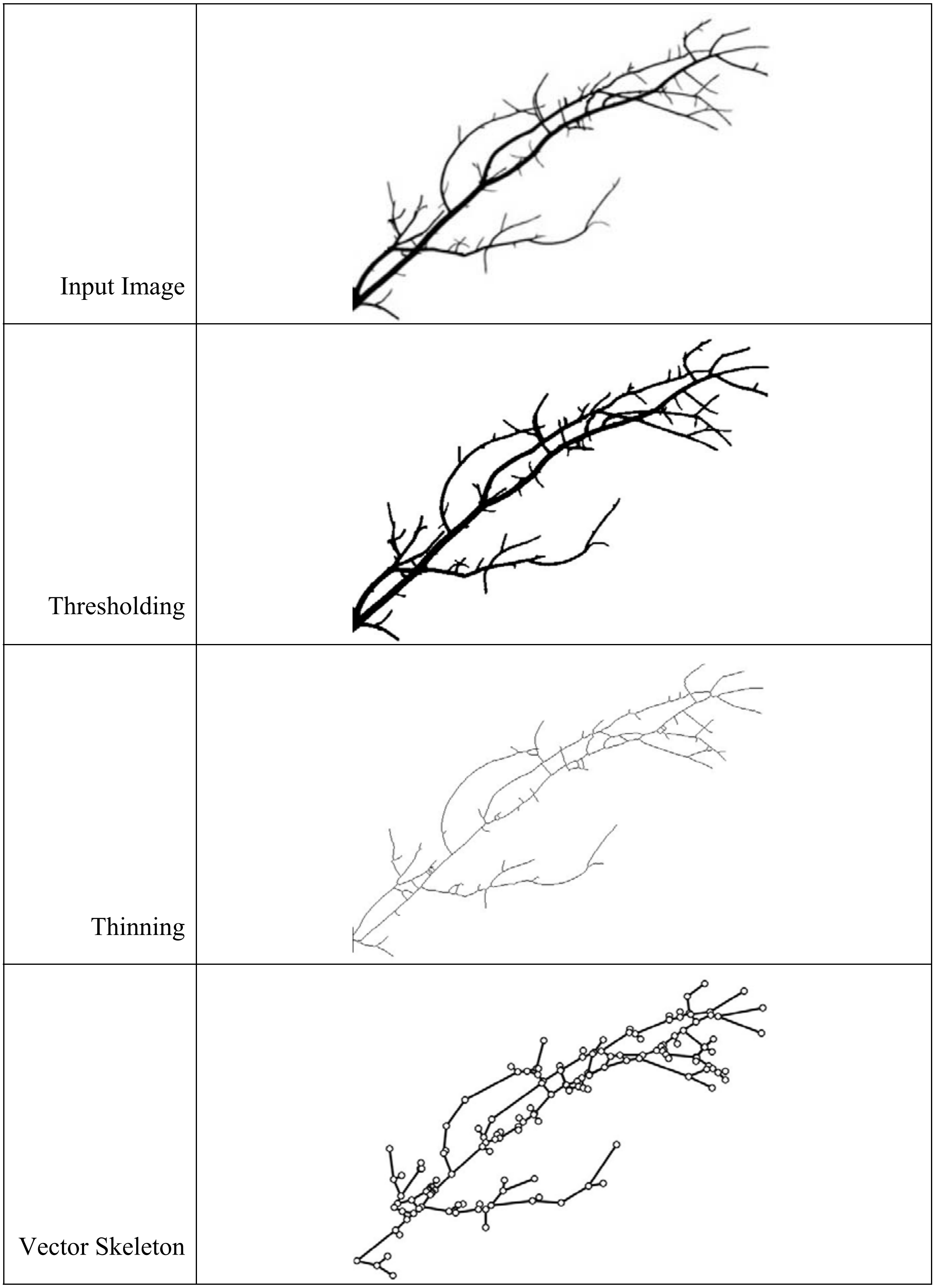


Figure 3.28. Sketch input example



Input Image

Thresholding

Thinning

Vector Skeleton

Figure 3.29. Image input example

3.7. Case-based Reasoning

The framework uses case-based reasoning techniques to classify, store and retrieve past design experiences. In contrast to many other research systems for conceptual structural design that rely primarily on textual information, this work emphasizes visual information about structures. Techniques from visual case-based reasoning are used, a field which overlaps the related areas of pattern recognition and content-based image retrieval.

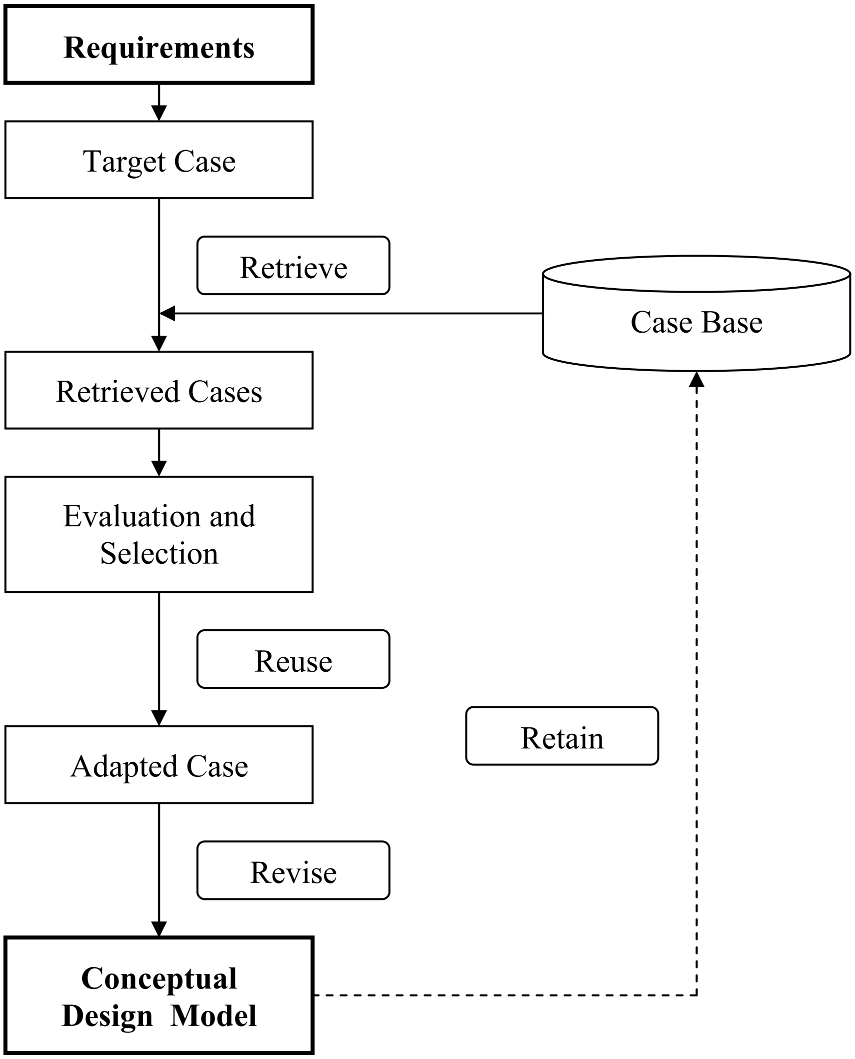


Figure 3.30. Overview of case-based reasoning

An overview of the case-based reasoning process is shown in Figure 3.30. In this case-based reasoning application, a case encapsulates information about the structural form. To initiate the case-based reasoning process, a target case is prepared and submitted to the system. The target case is compared to a library of source cases, and similar cases are returned from the library with a ranking. The user chooses a single case which most closely matches the target case, and this case is adapted for a new design situation. A case-based reasoning system requires a definition of what constitutes a case, a means of determining similarity between cases, mechanisms for retrieving and adapting existing cases, and a method for archiving new cases.

3.7.1. Case Representation

A central aspect of this framework is the representation of structural form within the case definition. The form is encapsulated using the relational graph format, which includes information from the Synthesis, Refinement, and Optimization Phases. The relational graph models the structural topology through its vertices and edges, along with additional information stored as attributes. In addition to topology, the design domain may also be represented as a relational graph for use in case-based reasoning, where graph vertices represent domain geometry and boundary conditions, and edges represent the spatial relationships between them. Both the structure and domain are also stored in image format to support image-based pattern recognition techniques. Similar to other case-based reasoning systems, cases also include a textual representation of high level requirements, performance evaluation, and feedback from fabrication, installation, use, maintenance and decommissioning. An overview of the elements of the case representation is given in Table 3.3.

Case Description
Structure graph
Structure image
Domain graph
Domain image
Requirements
Evaluation

Table 3.3. Case representation

A structural description $D = (P, R)$ consists of a set of primitives P and the set of relationships R between them [Shapiro and Haralick, 1981]. As described in Section 2.4.4, a structural description may be realized as an attributed relational graph (ARG). An ARG is defined as $G = (V, E, A_V, A_E, \alpha_V, \alpha_E)$, where V and E are respectively the sets of the vertices and the edges of the graph; A_V and A_E are the sets of the vertex and edge attributes, and α_V and α_E are the functions associating to each vertex or edge the corresponding attributes [Eshera and Fu, 1986]. Usually the nodes of a graph represent the primitives of the structural descriptions, and the edges represent the relations between the primitives.

The structural topology is represented as ARG where the vertices represent structural nodes and edges represent the topology of the connecting elements. The nodes are the primitives of the structural description, identifying the locations of key geometric features such as member intersections and discontinuities, and points where boundary conditions are applied. The vertex attributes include the node position, joint type, parameters describing the joint, nodal loads, and boundary conditions (Table 3.4).

The design domain is represented using a structural description in which the primitives are the geometric features composing the domain, as well as boundary conditions. For example, the geometric features include the design extents (a rectangle) along with the primitive shapes, such as circles and rectangles, that are subtracted from the extents to create voids. The relationships between the primitives represent spatial relationships, such as *Left*, *Right*, *Above*, *Below*, and *Contains*.

Structure Graph	
Vertex Attributes	
	Node position
	Joint type
	Joint parameters
	Nodal loads
	Boundary conditions
Edge Attributes	
	Cross-section type
	Section parameters
	End releases

Domain Graph	
Vertex Attributes	
	Domain extents
	Subtracted Areas
	Retained Areas
	Boundary conditions
Edge Attributes	
	Spatial relationships

Table 3.4. Relational graph representation in cases

3.7.2. Case Input

Case-based reasoning within the framework operates in three modes, depending on the type of input. In Synthesis Mode, a new case is directly constructed from the graphs, images and performance information developed during the Synthesis, Refinement, and Optimization Phases. In Image Mode, an image representing the target structure is processed to produce a graph, which is used with the image to retrieve suitable source cases from the case base. In Text Mode, keywords are used to guide the search for matching source cases, using the methods of document retrieval and conventional case-based reasoning. The three modes may be used in combination to improve search efficiency.

3.7.2.1. Synthesis Mode

In Synthesis Mode, the target case contains information on the shape of the structure, the boundary conditions, the connections, and expected performance. Source cases stored in the case base include such information, along with a range of other valuable information. Source cases include fully developed sets of requirements and specifications, performance measures from a range of different analyses, and evaluation data. For example, a source case could contain feedback from the fabrication shop, such as how a particular welded joint design contributed to geometric errors due to weld shrinkage. The source case may also contain design specifications or analysis results which are applicable to the target case.

3.7.2.2. Image Mode

In Image Mode, the target case contains only information on the shape of the structure, with no knowledge of boundary conditions and functionality. The goal in case-based reasoning is to infer the missing information from similarly-shaped structures. In a variation of Image Mode, a sketch-based interface could simply be used to aid in browsing the case base, and to retrieve similar structures.

3.7.2.3. Text Mode

In Text Mode, words or phrases at various levels of abstraction are used to retrieve applicable cases. The input text may refer to requirements, behaviour, or evaluation results.

3.7.3. Classification and Indexing

The organization of cases to enable efficient and accurate retrieval is a important issue in case-based reasoning. Classification is the process of arranging objects into classes, and indexing is the assignment of meaningful labels to objects. Effective indexing schemes need to work at several different levels of abstraction and support multiple views. The indexes must be abstract enough to cover a wide enough range of instances, but concrete enough to be recognizable. Indexes may need to account for the fact that different terms may refer to the same concept, depending on the user. Classification and indexing schemes are often hierarchically organized, to capture a range of abstraction levels and enable efficient retrieval. Closely related to classification and indexing is the notion of clustering. Inductive clustering methods generally look for similarities over a series of instances and form categories based on those similarities. When clusters are subdivided, hierarchies are formed.

The degree of similarity between objects is determined by the similarity between features of the objects. Therefore, in order to cluster, classify and index objects, a set of features describing the object is required. In case-based reasoning, features are commonly represented using attribute-value pairs. In design case representation, features may describe all aspects of the design, including function, structure and behaviour. In the relational graph representation developed in this framework, the vertex and edge attributes may be used for conventional classification and indexing. For example, the type and configuration of joints could be used to classify the structure as a truss, a braced frame, or a moment frame.

Techniques from image processing and pattern recognition are used to extract features from images. Table 3.5 lists a number of such features.

Feature	**Description**
domain aspect ratio	ratio of domain width to height
node count	total number of nodes
total path length	ratio of total member length to domain perimeter
average path length	ratio of average member length to domain perimeter
enclosed area	ratio of area enclosed by shape to domain area
bounding box	ratio of bounding box area to domain area
moments	normalized centroid location
Fourier descriptors	frequency content

Table 3.5. Image features

Images may be described in a compact form using feature vectors or histograms. Projections are used to characterize predominating directions of angular elements in images, as shown in Figure 3.31. Projections are generated at discrete angles, and the highest peaks identified (shown as the circled peaks in Figure 3.31). At a minimum, the directions with the highest peaks may be recorded in a feature vector. Another alternative is to decrease the resolution of the projection, and store the projection for each discrete angle as a histogram. In Figure 3.31, the skeleton is captured in the projection. Projections may also be developed to capture the distribution of vertices. Histograms are also used to plot the distribution lengths or angles of lines in the image.

Simple methods exist for measuring similarity between graphs based on global features. For example, similarity can be determined based on the number of vertices or edges, or on the degree of connectivity. More sophisticated graph matching techniques are discussed in Section 3.6.4.

3.7.4. Retrieval and Selection

During case retrieval, the target case is compared to source cases in the case base and the most similar cases are returned. The retrieved cases are then ranked to assist in the case selection process. Design case retrieval is usually concerned with finding partial rather than exact matches. The degree of matching depends on the level of abstraction at which the comparison is made; objects may be in the same category at a high level of abstraction, but not at a more specific level.

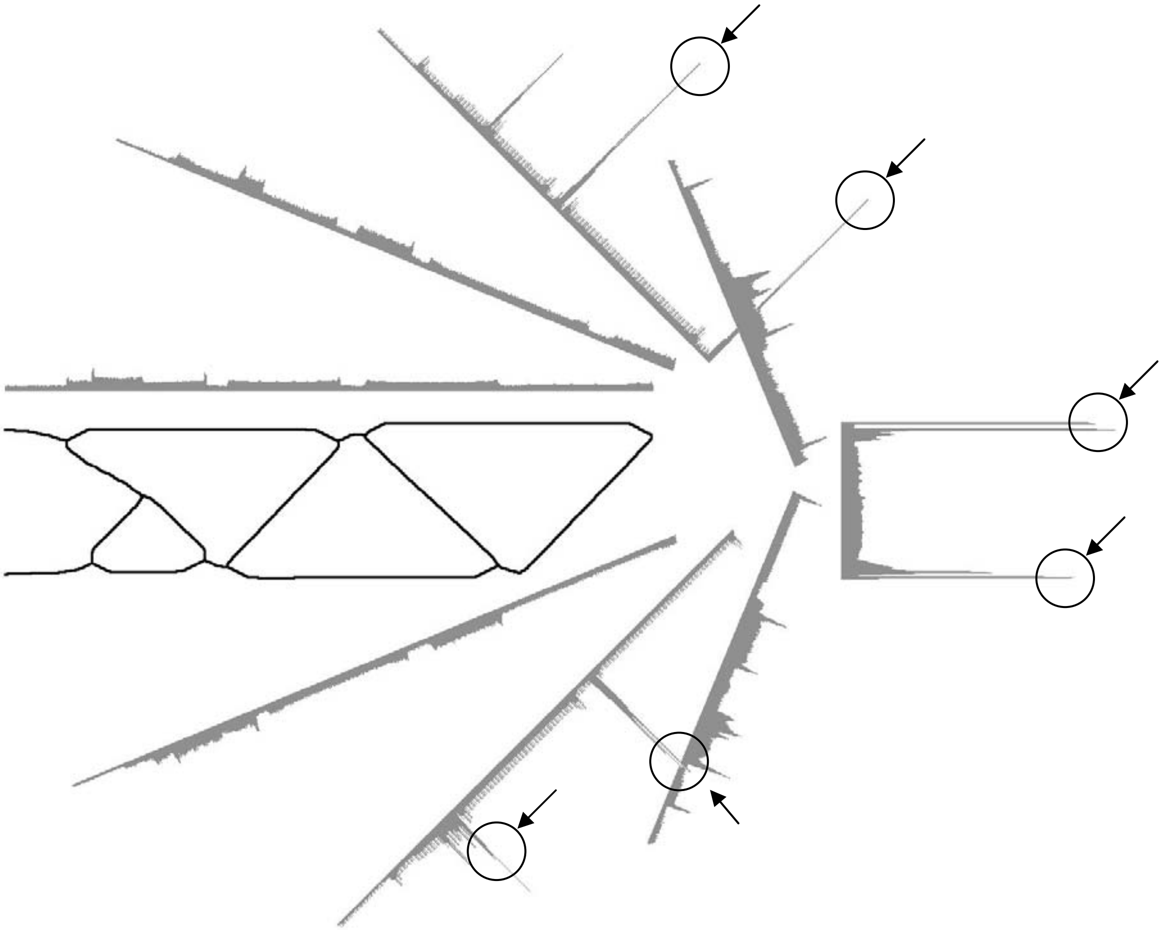


Figure 3.31. Projections of cantilever skeleton

In a simple serial search, the target case is compared sequentially to the source cases. A more efficient method is to structure the case base hierarchically using clustering methods, and to limit the search to the portion of the case base that is most likely to return a matching case.

Different techniques are used for matching, depending on the case content. This framework uses three methods, depending on whether a feature vector, image or graph is to be matched. These methods return a numerical measure of similarity, which is used to rank the retrieved cases. For certain types of case content, such as design functionality, numerical similarity measures are difficult to establish. Heuristic methods may be employed in those situations.

3.7.4.1. General Matching

A common numerical matching method used in case-based reasoning is the nearest-neighbour technique, which calculates a weighted sum of the differences between feature values. Different weighting factors are applied to each feature, depending on the importance of the feature in the match. The aggregate match score *S* comparing input and retrieved cases with *n* features is

$$S = \frac{\sum_{i=1}^{n} w_i \times sim(f_i^I, f_i^R)}{\sum_{i=1}^{n} w_i}, \tag{3.20}$$

where w_i is the weighting factor for feature *i*, *sim* is the similarity function for features, and f_i^I are the values for feature f_i in the input and retrieved cases, respectively.

3.7.4.2. Image Matching

A potential measure of similarity between images is to directly calculate the sum of differences between corresponding pixel intensity values. A difficulty with this method is that images with very similar content can have large differences when compared pixel by pixel.

The Object Density Map (ODM) [Coulon and Steffens, 1994], is an effective method for determining the similarity between two images. A low resolution grid is overlaid on the image, and the density within each grid square is calculated. For each square, the density is mapped to a pair of discrete density values drawn from a set of seven overlapping intervals. To compare two images, the images are first aligned using their centroids as a common reference point. The ODM is generated for each image, and the corresponding grid squares are compared. If the two grid squares have at least one density value in common, then the squares are assumed to have equal density. Otherwise, the minimum difference between density values is calculated. A similarity measure for the two images is taken as a normalized sum of differences over all the grid squares, resulting in a number between 0 and 1. Images with identical ODMs have a difference of 0, and completely dissimilar images have a difference of 1. Two examples of image comparison using ODMs are shown in Figure 3.32 and 3.33. In each of these figures, the input image is shown at the top with the ODM grid superimposed. The ODM is shown below each input image. The similarity measure shown in the figure caption is calculated between the left and right ODM.

In the framework, the technique is used to compare both structure images and domain images for similarity.

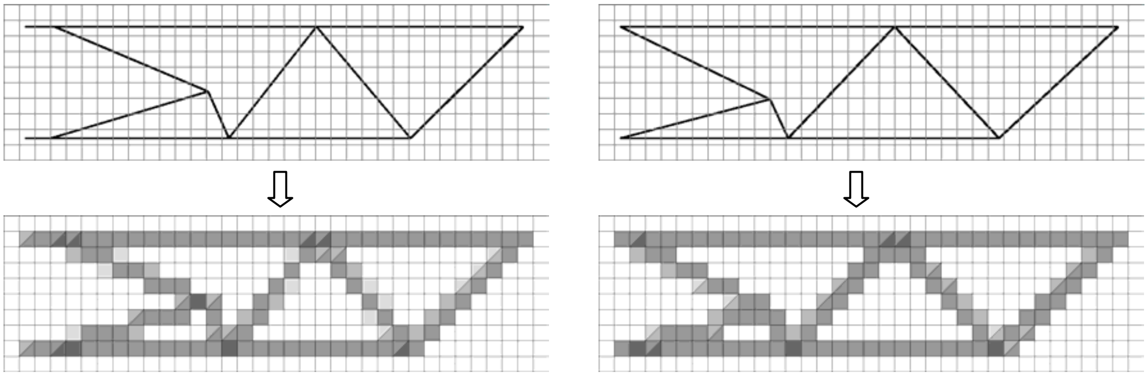


Figure 3.32. Similarity using ODM (difference measure = 0.034)

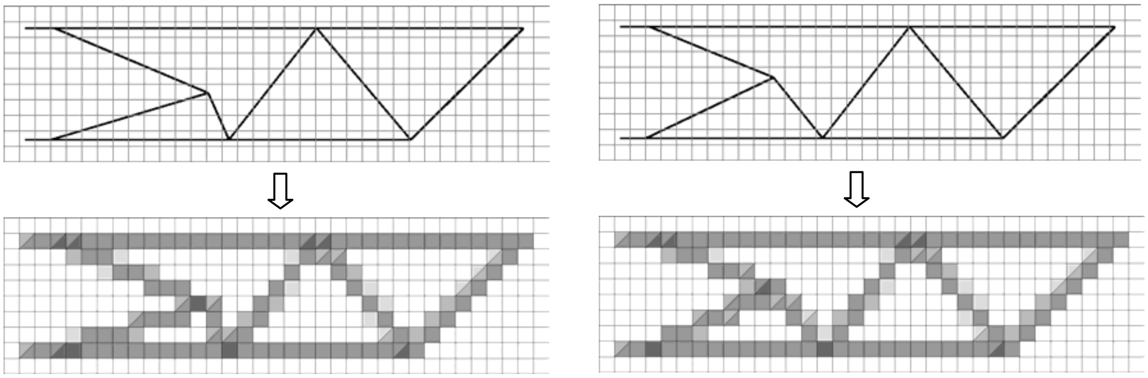


Figure 3.33. Similarity using ODM (difference measure = 0.009)

3.7.4.3. Graph Matching

Graph matching has been studied extensively and many different approaches are found in the literature. The characteristics of the graph representation used in this framework limit the selection of graph matching methods somewhat. Since exact matches between structural graphs are unlikely, an inexact method is required. Another feature of this application is that large training set for developing class definitions are not available, as in character recognition, for example. The emphasis in this framework is on using graph matching for measuring similarity between patterns, rather than on classification.

In the framework, the solution to the minimum cost subgraph isomorphism problem, as formulated by [Thoresen, 2007], is proposed for graph matching. Unlike other graph matching algorithms, this approximate method takes advantage of the fact that nodal coordinates are stored as graph attributes. For such spatially coherent graphs, the method uses the vector between nodes as the edge attributes. The algorithm exploits the connectivity of the graph to produce sufficiently accurate solutions in lower time-complexity time than other published subgraph isomorphism algorithms.

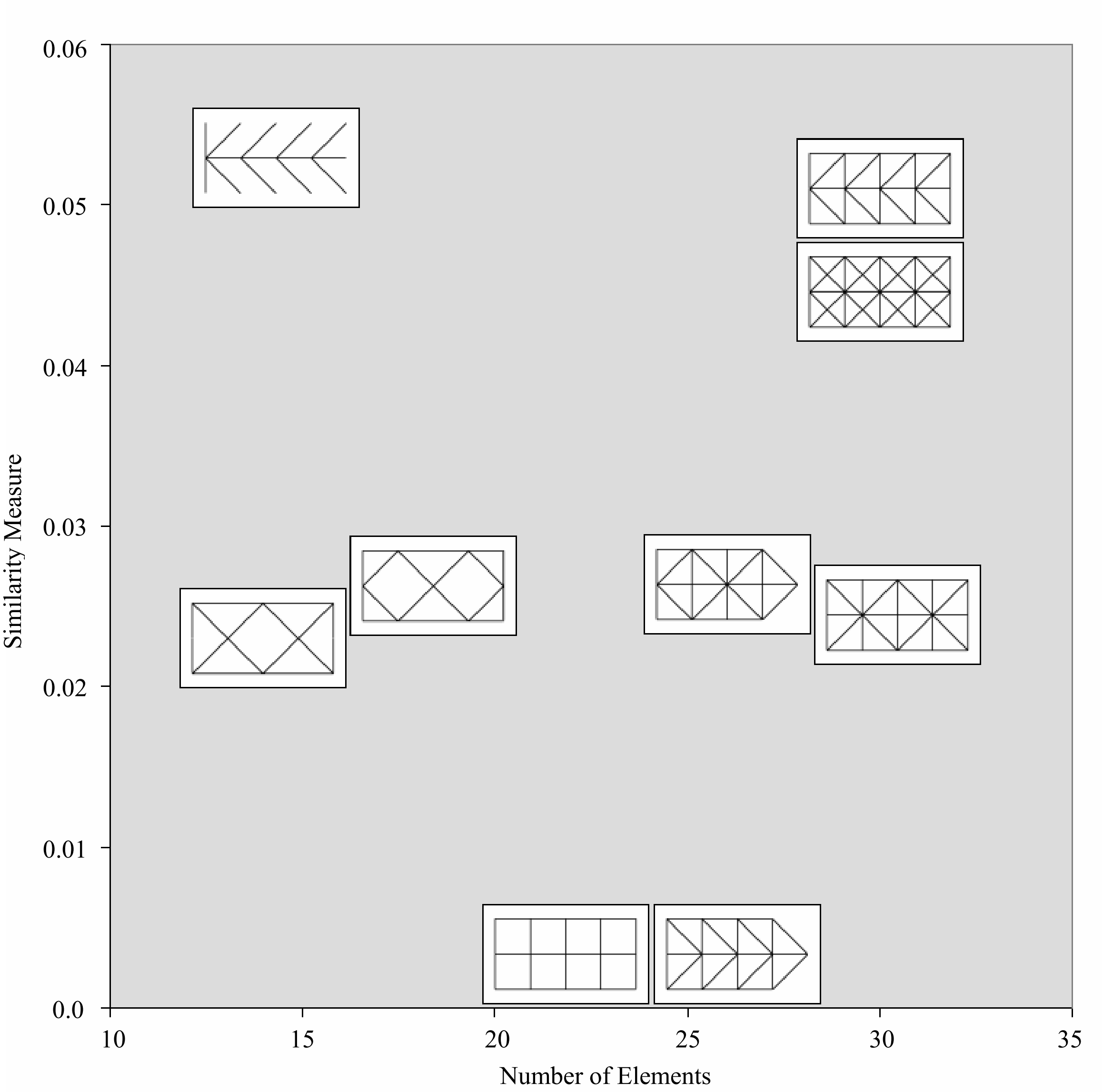


Figure 3.34. Graph matching sample result

The results of graph matching may be displayed as a grid of thumbnail images, similar to Figure 3.34, where the proximity of images indicates the degree of similarity. In Figure 3.34, the numerical similarity measure is plotted on the vertical axis, and the number of elements in the structure are plotted on the horizontal axis. Since similarity measures are calculated, a numerical ranking may be provided to the user to assist in the selection of a relevant design case.

3.7.5. Adaptation

Structural design is an iterative process where a design concept is incrementally changed to satisfy a set of design constraints. For designs that have been successfully executed and stored as cases, it is assumed that all constraints have been satisfied. Past designs therefore encapsulate a substantial amount of work. One of the goals of case-based reasoning is to take advantage of that work in creating new designs, rather than create designs from scratch. Adaptation is the process in case-based reasoning where past designs are modified for use in a new application.

In the framework, the focus of case-based reasoning is to use similarity between cases to infer missing information in the current case from past design cases. Requirements, constraints or feedback from stored design cases can be incorporated early in the conceptual design phase to accelerate the design process and allow more alternatives to be explored. When adaptation is existing cases is necessary, the method of derivational replay is proposed.

Derivational replay is an adaptation method where the process used to derive a previous solution is reused in a new situation [Kolodner, 1993]. Given that design procedures are stored in the framework, and given that design parameters are stored with cases, all the necessary information exists to reuse procedures in new designs.

3.7.6. Discussion

The central issue in case-based reasoning in the context of this framework is the measurement of similarity between structural forms. The image matching techniques presented here, the Object Density Map and projections, work at multiple levels of abstraction. Setting the ODM grid resolution to a coarser value increases the level of abstraction, and increases the possibility of a match. Similarly, projections summarize characteristics at discrete angular increments, and the resolution of the increment can be used to control matching. Drawing similarities between objects at different levels of resolution is an important feature in inductive learning. Inductive learning is the process that allows properties of source cases to be assigned to the target case, leveraging the experience and knowledge of the case base.

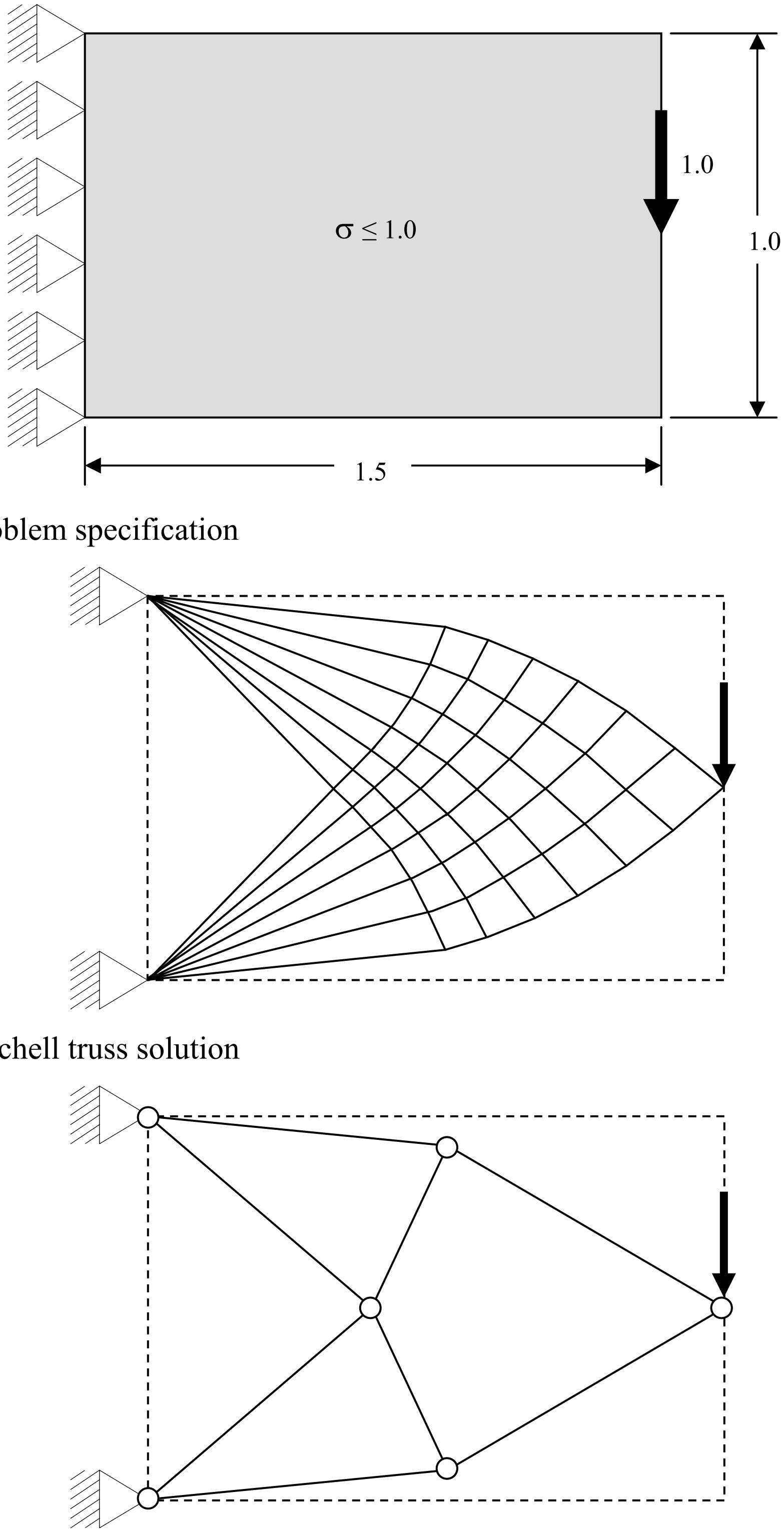
4. Case Studies

This chapter describes a number of case studies that illustrate the application of the framework to conceptual design problems. Several of the case studies compare the performance of the framework against benchmark problems. Other studies show the potential application to real-world conceptual design tasks.

4.1. Cantilever

In a well-known example from structural optimization, the minimum weight is sought for a cantilever truss supporting a vertical load, as shown in Figure 4.1. All members have unit allowable stress in tension and compression, and unit density. Buckling and displacement constraints are not considered. The "exact" solution to the problem is generally attributed to Michell, developed using the theory of "truss-like" continua, rather than discrete members [Michell, 1904]. Since structures developed using this theory consist of an infinite number of bars of nonstandard length, they cannot be fabricated using practical methods, but they do serve as interesting benchmark cases. For this problem, the solution has been shown to be 4.5 (Figure 4.1.b) [e.g., Lewinski et al, 1994]. Discrete truss topology optimization has also been used to solve this example. Kirsch obtained an 8-bar truss topology with weight 4.59 using mathematical programming techniques (Figure 4.1.c) [Kirsch, 1993]. [Azid et al., 2002] found a slightly improved solution using an evolutionary approach, with weight 4.57 (Figure 4.2).

The problem is solved using the framework presented here by first formulating the specifications. The problem domain is assumed to be a grid of square elements 30 wide by 20 high (Figure 4.3.a). A line boundary condition, with fixity in *x*- and *y*-directions, is applied to the left edge of the domain. A point load is applied at midheight on the right edge. Topology optimization is run with Poisson's ratio ν=0.3 and volume fraction 0.15. The resulting image is shown in Figure 4.3.b. The image suggests a truss-like structure with similar topology to that obtained by Kirsch. A significant difference is that short beam elements are indicated adjacent to the two support points.



a) Problem specification

b) Michell truss solution

c) Solution of [Kirsch, 1993] (weight = 4.59)

Figure 4.1. Cantilever beam example

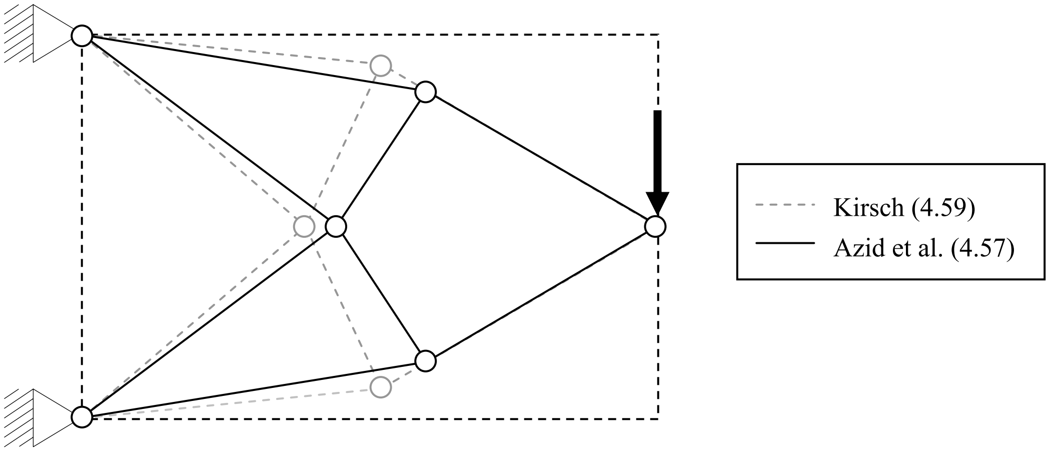
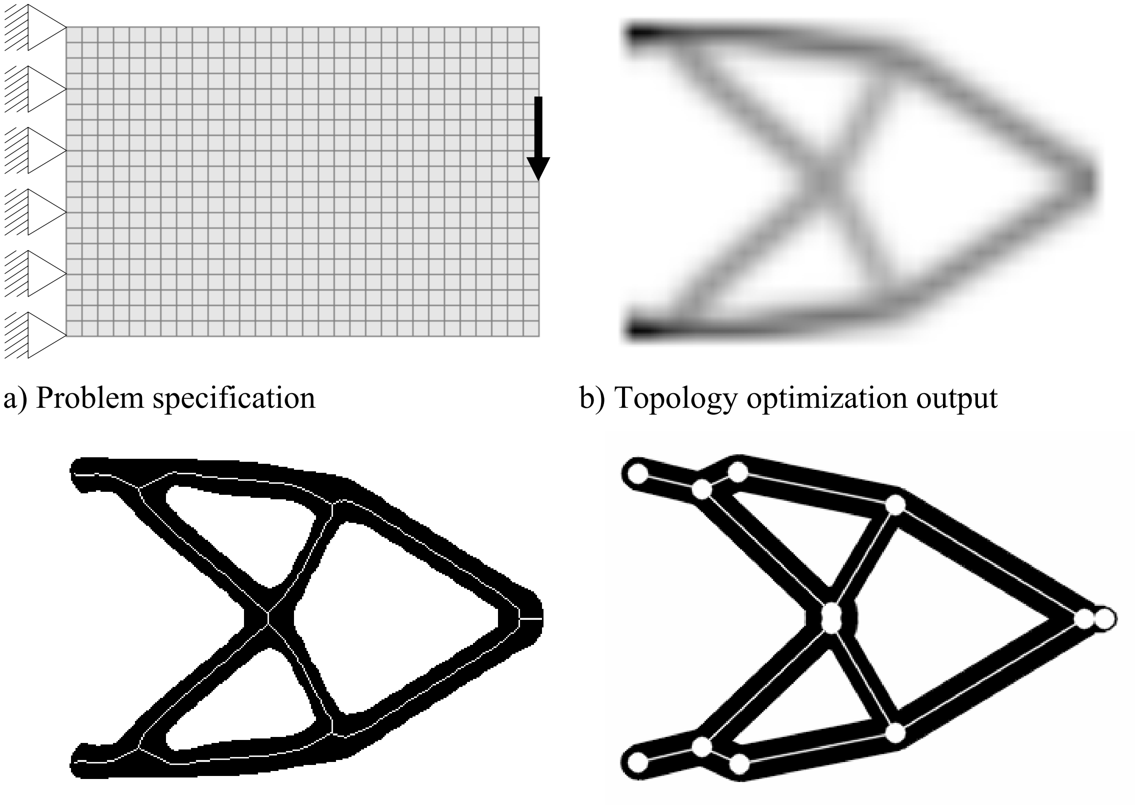


Figure 4.2. Cantilever beam solution by [Azid et al., 2002]



a) Problem specification

b) Topology optimization output

c) Skeleton

d) Element model

Figure 4.3. Cantilever beam model generation

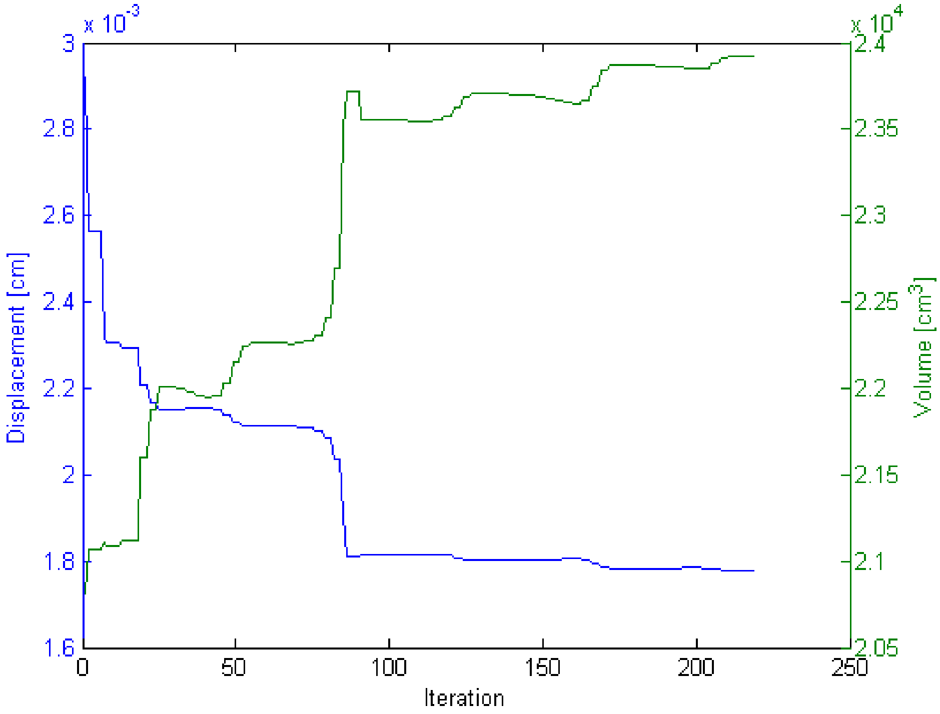


Figure 4.4. Cantilever geometric optimization – displacement and volume history

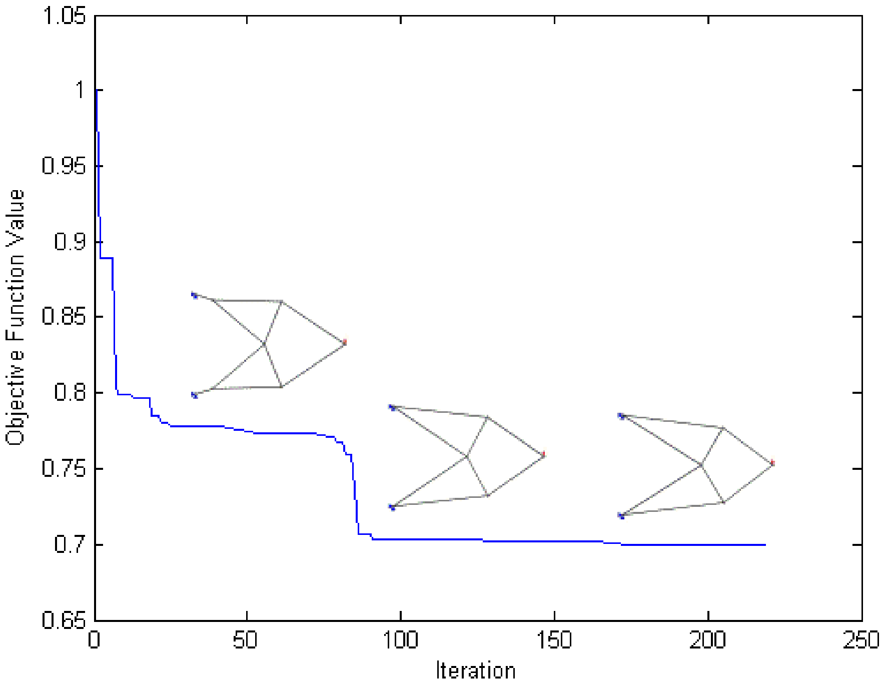


Figure 4.5. Cantilever geometric optimization – objective function history

The topology optimization image is converted to binary formal and a skeleton is generated as shown in Figure 4.3.c. The skeleton is traced to produce the element model in Figure 4.3.d.

In the Refinement Phase, geometric optimization is used to shift node positions in a way the minimizes the product of compliance and volume. Pairs of nodes are merged if they move close enough to each other, and are deleted when they produce a pair of nearly colinear elements. As shown in Figure 4.5, after about 86 iterations of geometric optimization, the 8-member truss topology found by Kirsch emerges. To achieve the 8-member topology, a significant increase in volume is required, about 7%. This increase in volume is accompanied by a 17% decrease in deflection, representing a significant increase in performance. As the performance increases, the geometry of the structure becomes simpler, with the elimination of 2 nodes and 2 elements. The reduction of the node and element count results in immediate cost savings in design, fabrication and installation.

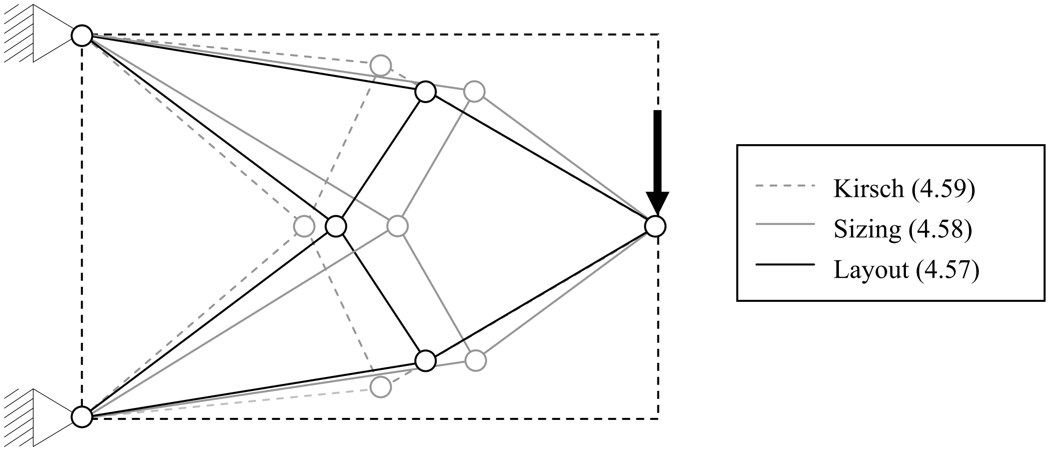


Figure 4.6. Cantilever beam solution

During the refinement phase, no additional information on member cross-sections or stress limits was introduced. With the limited data used to define the problem domain and boundary conditions, a layout similar to that obtained using discrete truss optimization methods was found. To compare the performance of the truss generated using this framework to the benchmark cases, further processing is required. Using the topology and geometry produced during the Refinement Phase as input, member sizing optimization was run. In sizing optimization, the design variables are the member cross-sections, and a constraint was applied to limit tensile and compressive stress to 1.0. Member sizing optimization results in a structure with mass 4.58, as shown in Figure 4.6. To improve on this result, sizing optimization and geometric optimization can be run simultaneously. This procedure produces a layout similar to that obtained by Azid, with the same mass of 4.57, as indicated in Figure 4.6 (identified as the 'Layout' configuration).

The cantilever example demonstrates that the framework can be used to generate a structure with performance similar to that achieved using both classical mathematical programming methods and more recent evolutionary methods. The framework has advantages over both these approaches. Classical optimization methods are generally problem specific and require more effort to implement than the general approach used in the framework. Evolutionary methods generally require much more computational effort than deterministic procedures such as the one implemented in the framework. Speed and simplicity are important properties of computational tools for conceptual design, allowing the user to quickly reformulate the design parameters and evaluate the impact of changes on the resulting structure.

4.2. Bridge

In an example problem in bridge design, a minimum-weight truss-like structure is required for a single, simply-supported span. This problem has been used by researchers to evaluate discrete truss topology optimization with evolutionary algorithms. For example, [Shrestha and Ghaboussi, 1998] used a genetic algorithm and [Yang and Soh, 2002] employed genetic programming. Both of these approaches were notable in that they did not use a conventional ground structure consisting of a predefined grid of node points. Initial populations are generated through randomly generated patterns of nodes. The lack of a ground structure facilitates a general, domain-independent approach to topology optimization that is suited to conceptual design work. Similar to the approach used in the framework presented here, a general design domain representing the extents of the structure is used as input.

The bridge structure is required to span 70 meters, as shown in Figure 4.7. The depth of the structure is limited to 10 meters. Four vertical, evenly spaced point loads are applied on the span. The truss members are to be selected from a set of 30 standard AISC wide flange steel sections [AISC, 1989], from W14x22 through W14x426. The material properties are those of steel (E = 201 GPa, f_y = 248.8 MPa, ρ = 7851.03 kg/m^3). The following AISC design specifications [AISC, 1989] are applied:

- the allowable tensile stress is $0.6f_y$
- the allowable member slenderness is 300 for tension members and 200 for compression members
- member length is to be between 5 m and 35 m
- the allowable joint displacement is limited to 1/1000 of span, or 70 mm
- the allowable compressive stress σ_i^b of member i is determined from buckling considerations as follows:
    - if $\lambda_i > C$ (elastic buckling), $\sigma_i^b = \dfrac{12\pi^2 E}{23\lambda_i^2}$

- if $\lambda_i < C$ (plastic buckling), $\sigma_i^b = \dfrac{\left(1-\dfrac{\lambda_i^2}{2C^2}\right)f_y}{\dfrac{5}{3}+\dfrac{3\lambda_i}{8C}-\dfrac{\lambda_i^3}{8C^3}}$

  where $\lambda_i = L_i/r_i$, $C = \pi\sqrt{2E/f_y}$, and L_i and r_i are the length and radius of gyration of member *i,* respectively.

The optimal topology derived by [Yang and Soh, 2002], is shown in Figure 4.8. Detailed member sizes and node positions are given in Table 4.1.

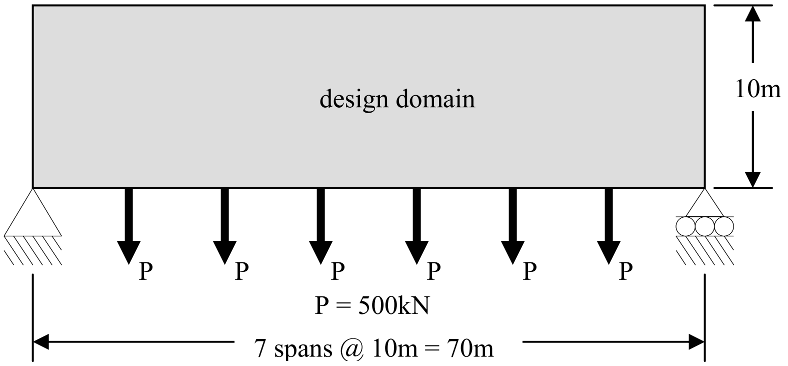


Figure 4.7. Bridge problem specification

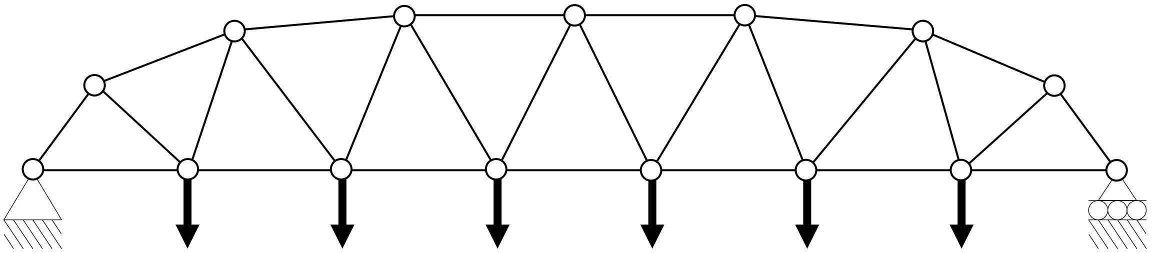
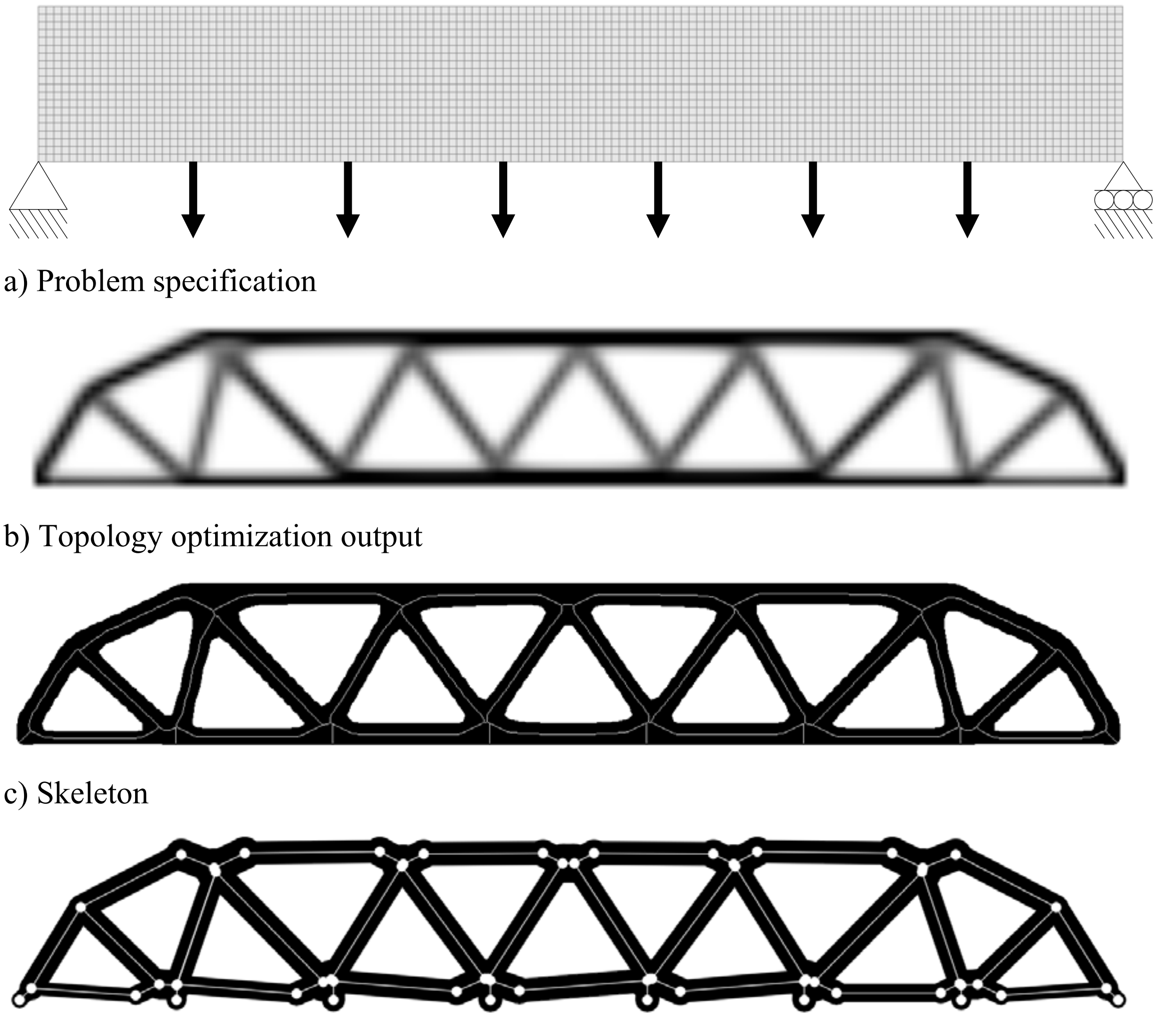


Figure 4.8. Bridge design result [Yang and Soh, 2002] (45 404 kg)

To solve the problem using the framework, the domain is first defined. A domain consisting of 140 x 20 square elements is used, as shown in Figure 4.9.a. The appropriate point boundary conditions are applied at each end and 4 point loads are applied along the bottom edge of the domain. Topology optimization is run for a volume fraction of 0.3; the resulting image is shown in Figure 4.9.b. The skeleton derived from the topology optimization output (Figure 4.9.c) and the element model (Figure 4.9.d) are also given.



a) Problem specification

b) Topology optimization output

c) Skeleton

d) Element model

Figure 4.9. Bridge model generation

Geometric optimization is applied to simplify the topology of the element model. After a mere 73 iterations, the optimal topology is achieved (Figure 4.10 and Figure 4.11). The topology matches the result obtained by [Yang and Soh, 2002] (Figure 4.12).

Sizing optimization is conducted to select member sizes for the truss in accordance with the problem constraints. Sizing optimization uses continuous design variables representing the section number, which ranges from 1 to 30. Both the area and radius of gyration of the cross-sections are expressed as continuous functions of the section number. The area is represented by a cubic polynomial (Figure 4.13). The radius of gyration is approximated using piecewise continuous cubic polynomials (Figure 4.14).

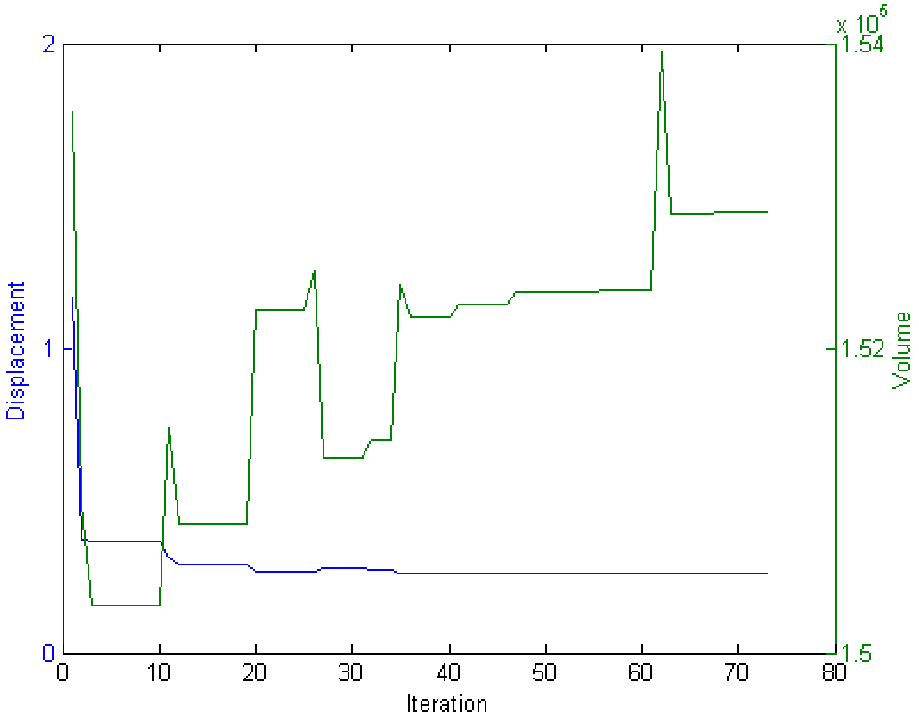


Figure 4.10. Bridge geometric optimization – displacement and volume history

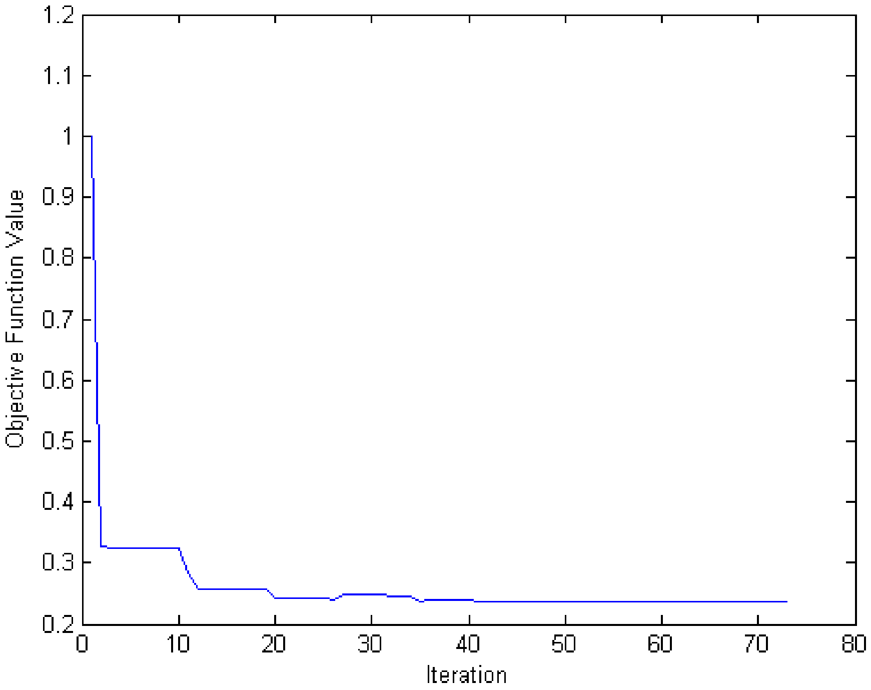


Figure 4.11. Bridge geometric optimization – objective function history

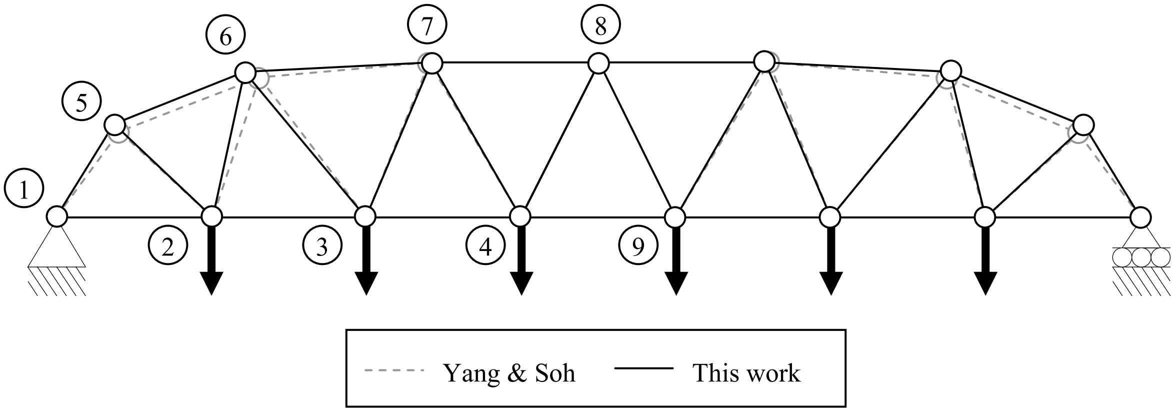


Figure 4.12. Bridge design result

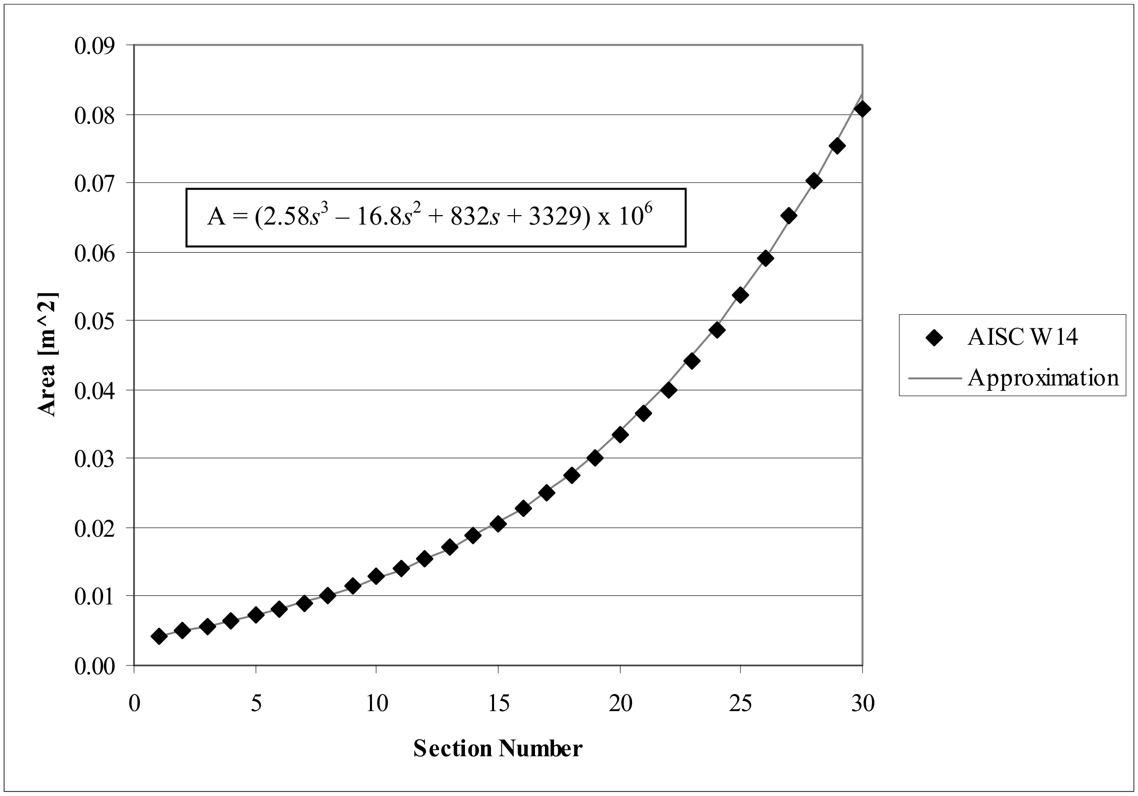


Figure 4.13. Bridge member cross-section area

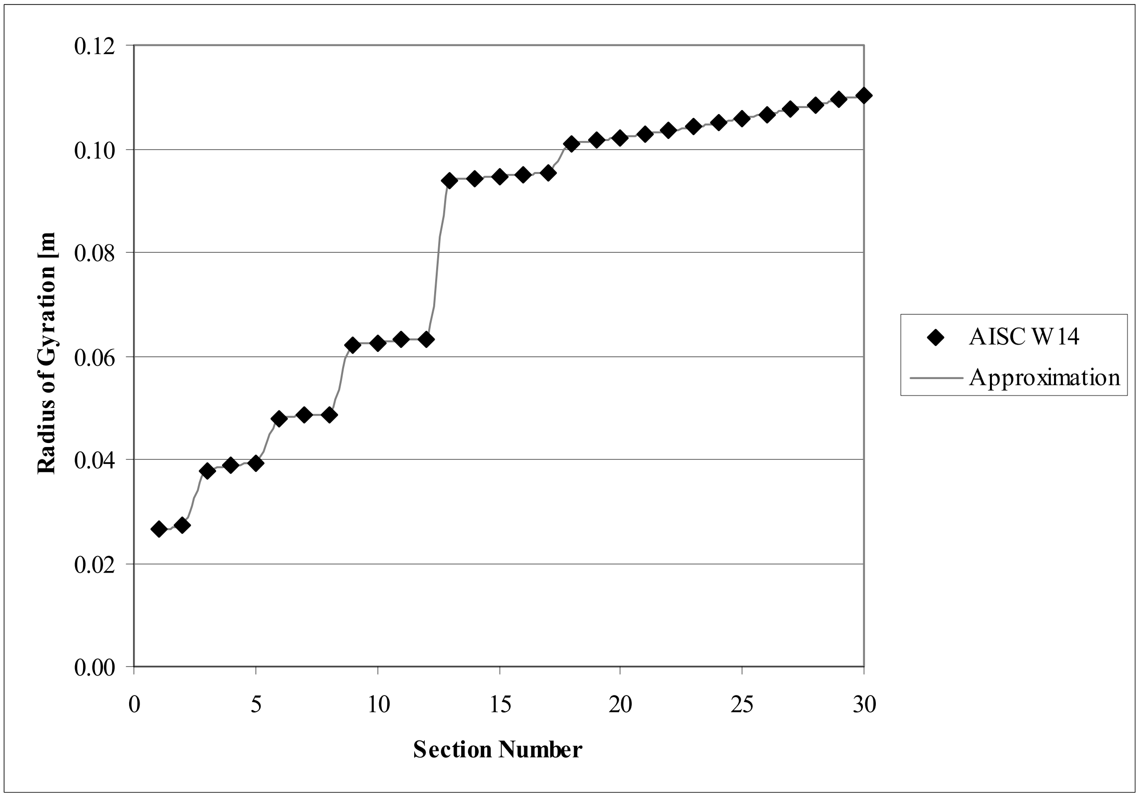


Figure 4.14. Bridge member cross-section radius of gyration

Table 4.1 summarizes the results of the bridge design example, and compares the results to those obtained by other researchers. The results for the genetic algorithm of [Shrestha and Ghaboussi, 1998] and the genetic programming approach of [Yang and Soh, 2002] are markedly different in terms of both optimum weight and computational efficiency. The genetic algorithm used in this comparison produces a weight 30% higher using over five times the number of total iteration steps of the genetic programming method.

Running the example with the framework described in this work, the optimal weight is within 0.6% of the result obtained using genetic programming. A notable difference between the solutions is that only 395 iteration steps are required using the framework, where each iteration step corresponds to a single finite element run. For the continuum topology optimization stage, 35 iterations are completed. A further 73 iterations are required for geometric optimization, and 287 iterations are used for the final member sizing optimization. The number of finite element runs is several orders of magnitude less than the number required for the genetic programming method for this example.

Result		Shrestha & Ghaboussi [1998]	Yang & Soh [2002]	This work
Member sections	1-2	NA	W14×68	W14×53
	2-3	NA	W14×109	W14×132
	3-4	NA	W14×132	W14×211
	4-9	NA	W14×233	W14×193
	1-5	NA	W14×132	W14×90
	5-6	NA	W14×132	W14×120
	6-7	NA	W14×176	W14×193
	7-8	NA	W14×193	W14×257
	2-5	NA	W14×61	W14×68
	2-6	NA	W14×74	W14×61
	3-6	NA	W14×82	W14×68
	3-7	NA	W14×82	W14×68
	4-7	NA	W14×90	W14×43
	4-8	NA	W14×61	W14×30
Node coordinates (m)	x_5	NA	4.0	3.71
	y_5	NA	5.5	5.92
	x_6	NA	13.0	12.2
	y_6	NA	9.0	9.45
	x_7	NA	24.0	24.3
	y_7	NA	10.0	10.0
	x_8	NA	35.0	35.0
	y_8	NA	10.0	10.0
Weight (kg)		60 329	45 404	45 677
Total iteration steps		975 400	166 000	395

Note: NA indicates that results are not available

Table 4.1. Bridge design result

A common limitation of the genetic algorithm in truss topology optimization is the fixed length chromosome. Efforts to find efficient methods to represent a wide range of different topologies with varying complexity using a fixed length string have not been completely successful. The approach of [Shrestha and Ghaboussi, 1998] uses relatively long chromosomes, and requires a large number of iterations to converge. One of the strengths of genetic algorithms is their simplicity, but it is difficult to implement simple crossover and mutation methods that do not produce unstable offspring with disconnected members. [Yang and Soh, 2002] encode the structural configuration using two-dimensional, varied length parse trees, which can more flexibly represent a diverse range of topologies than fixed length chromosomes.

Evolutionary computing techniques are a very effective approach to global optimization in problems that are highly nonlinear or nonconvex. The previous two examples show that, for some problems, it is possible to generate optimal topologies more efficiently using an approximate, deterministic optimization procedure. The examples show the approximate results are more than sufficient for conceptual design purposes. The ability to generate solutions efficiently and quickly is seen as a major advantage in conceptual design, where a computational tool can rapidly provide feedback to a human designer as they explore a range of possible solutions.

4.3. Bicycle

The bicycle frame problem shown in Figure 4.15 has been used by researchers [Rasmussen and Olhoff, 1992; Chirehdast et al., 1994] as a test case for topology optimization. The objective is to optimize the rigidity of a bicycle frame with no predefined topology. The input to the problem is a rectangular design domain, a load case, and support conditions. Loads are applied to the top of the seat tube (A), the head tube (B), and the bottom bracket (C).

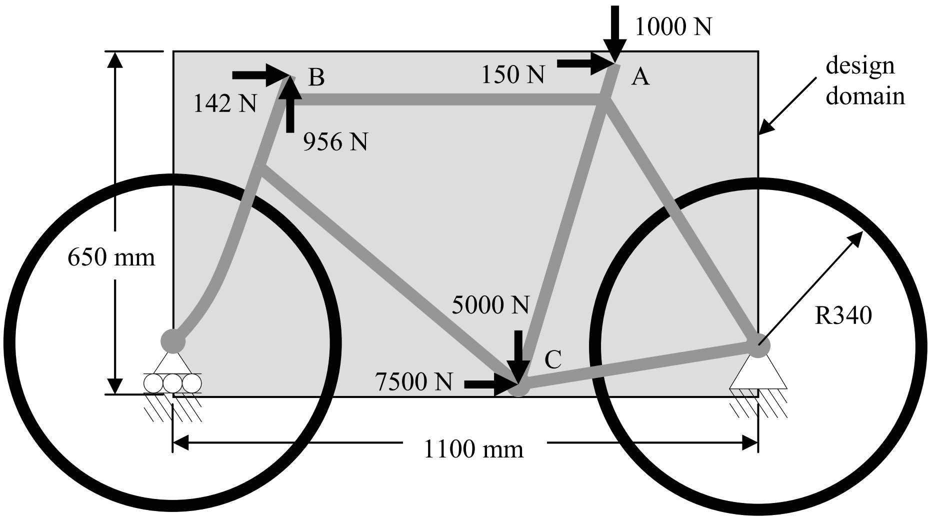
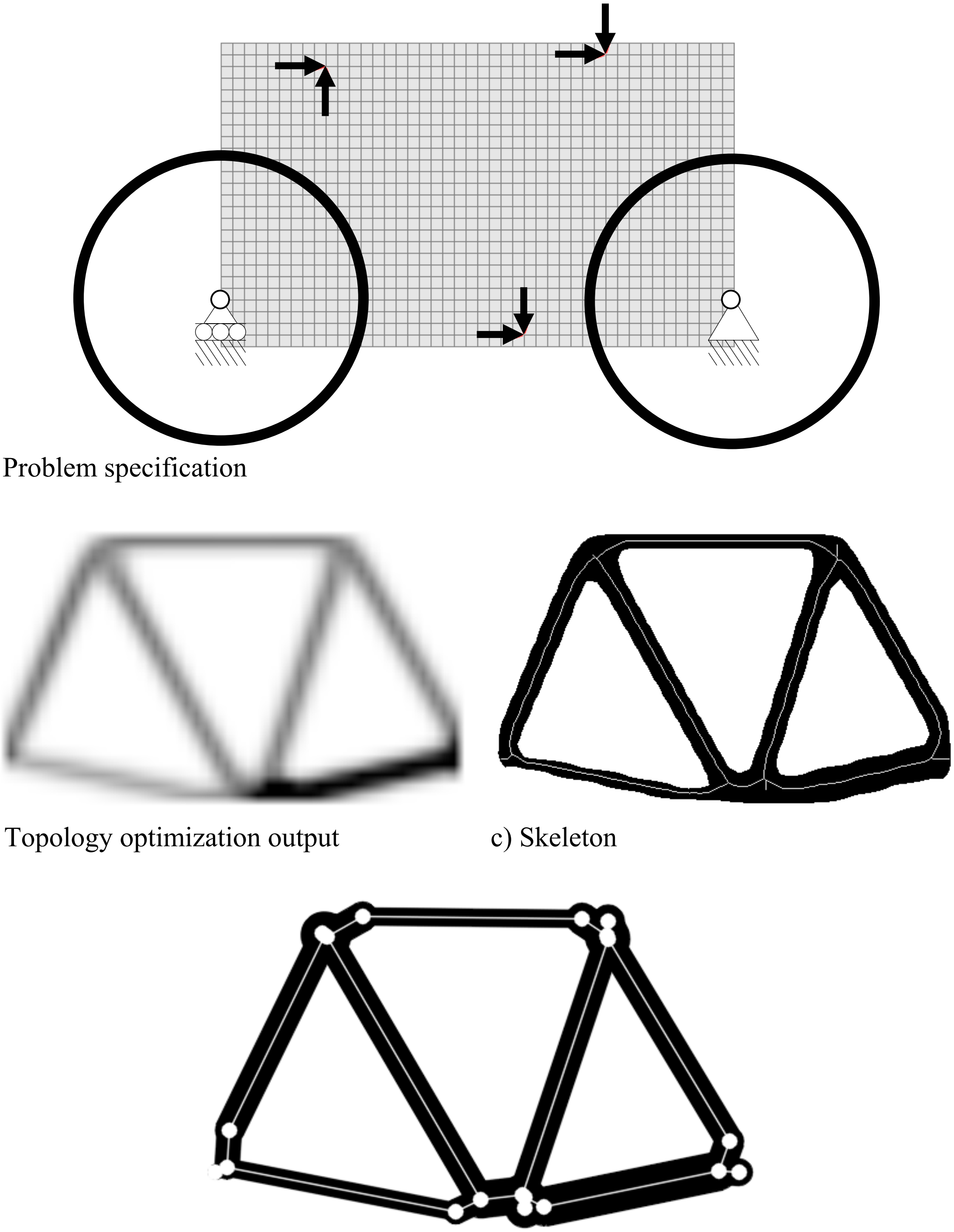


Figure 4.15. Bicycle frame problem specification

To solve the problem, a domain consisting of 44 x 26 square elements is formed, with each element 25 mm in size. Six point loads are applied and three nodal degrees of freedom are constrained as shown in Figure 4.16.a. The frame is pin supported at the rear wheel, and a horizontal roller support is used at the front wheel. Topology optimization is run for a volume fraction of 0.15 (Figure 4.16.b).



a) Problem specification

b) Topology optimization output

c) Skeleton

d) Element model

Figure 4.16. Bicycle frame model generation

The topology optimization output suggests the familiar diamond-shaped bicycle frame, with the exception that the front fork is connected to the bottom bracket. The frame in this case has no steering functionality, as in the two examples given in the literature.

In the Refinement Phase, the number of nodes is reduced from 20 to 5 and the maximum displacement decreases by about 27%. The resulting frame is shown in Figure 4.18, a simple truss structure with the minimum number of possible nodes. Also shown in Figure 4.18 is the topology found by [Chirehdast et al., 1994].

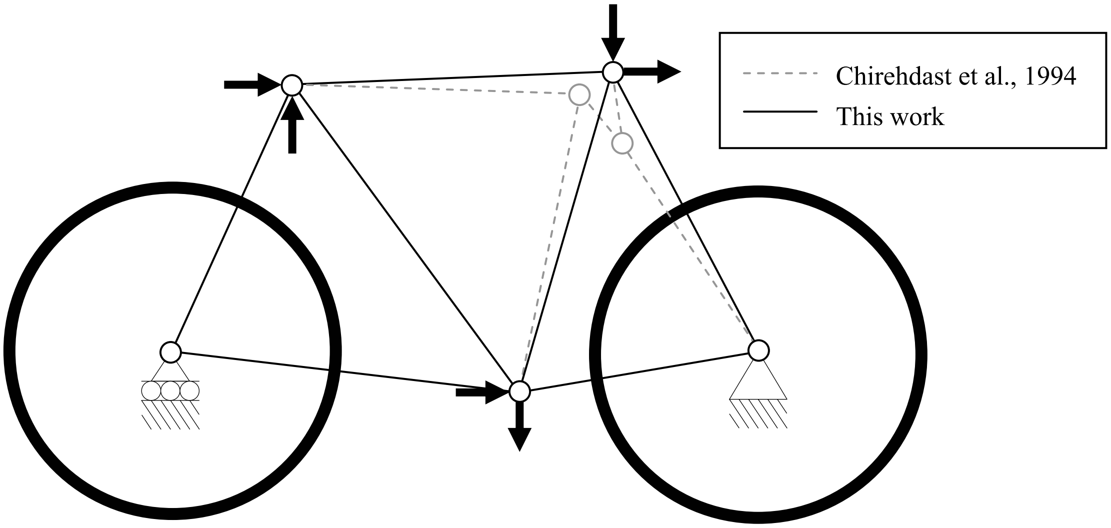


Figure 4.17. Bicycle frame design result

In an approach similar to that used in the framework described here, [Chirehdast et al., 1994] extract truss or frame structures from continuum topology optimization output using image processing techniques. There are several major differences between the system of Chirehdast, called ISOS, and the one presented here. First, the approach used in this framework is to trace the skeleton paths to an arbitrary level of accuracy, adding as many nodes as necessary to follow the path. The resulting topology is then simplified using mathematical optimization. In contrast, ISOS uses heuristics derived from elementary mechanics and engineering intuition to interpret a truss or frame structure. In this framework, nodes may be added at intermediate points between path junctions. This allows the framework to model arches and other curvilinear elements. In ISOS, nodes are located only at path junctions, so curvilinear elements cannot be modeled. Another difference between this framework and ISOS is in the treatment of stability. ISOS interprets a skeletal structure as either a truss or frame structure. If the truss interpretation is unstable, then heuristic methods are used to add members to ensure stability. In contrast, this framework generates stable hybrid structures with joints of variable flexibility, from rigid beam connections to frictionless pin connections. The configuration of pins and moment connections is established using a mathematical optimization procedure. Finally, this framework introduces a number of methods that are not part of

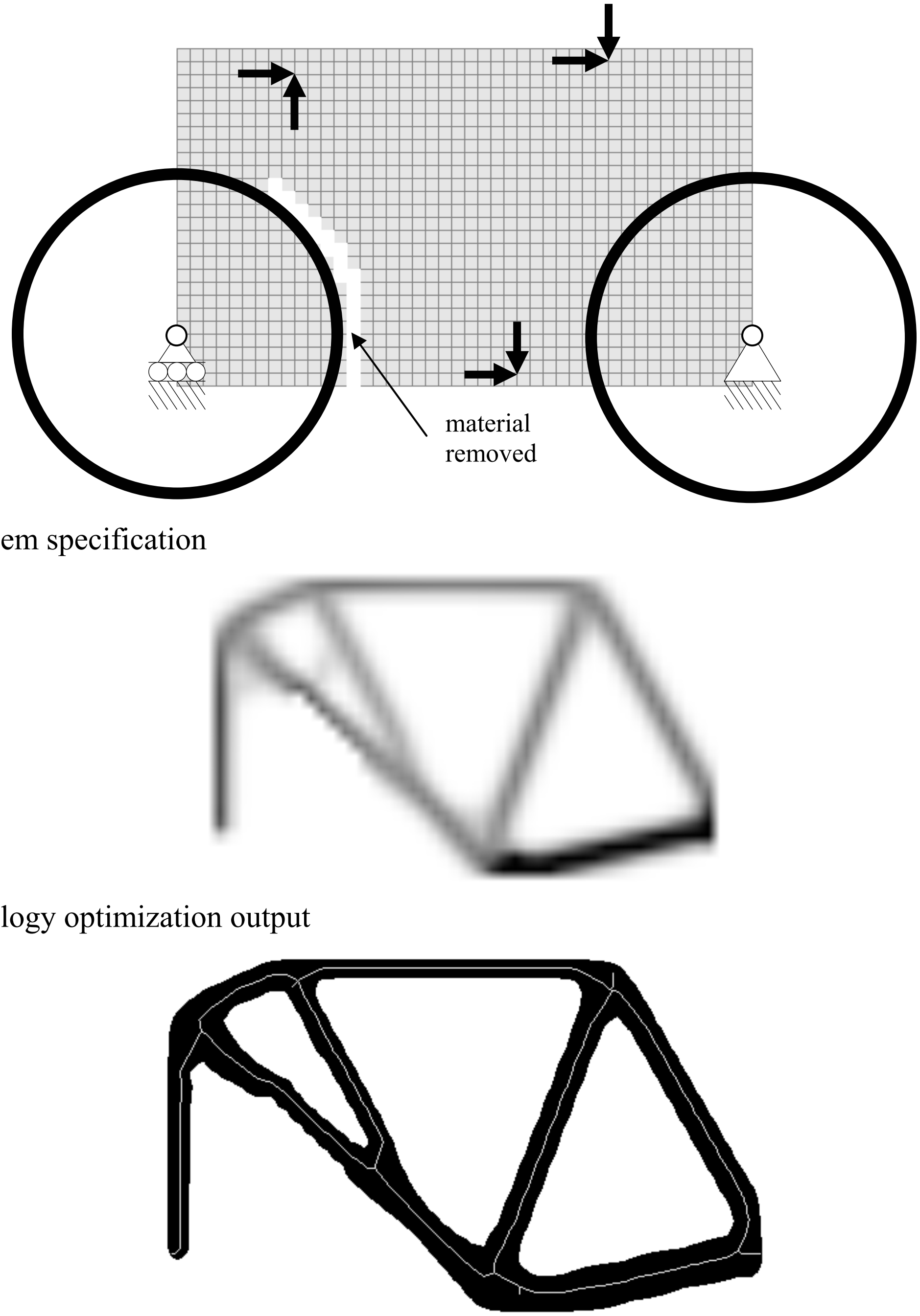
ISOS, including visual case-based reasoning, structural similarity measures, and boundary condition inference.

To further the design of the bicycle frame, member cross-sections are selected using a sizing optimization procedure. The frame is assumed to consist of beam elements, with one design variable per element. The members have tubular cross section with wall thickness equal to one tenth of the outer radius. The design variable is the outer radius of the tube. Initially, each member is a tube with radius 10 mm, and cross-sectional area 59.69 mm^2. The material is assumed to be the same for all members, with maximum permitted stress 50 MPa in tension and compression, and density 7800 kg/m^3. Using a Sequential Quadratic Programming (SQP) method, this framework produces a configuration with weight 2.34 kg (Figure 4.18, in black). In [Chirehdast et al., 1994], ISOS is shown to generate a comparable solution that weighs 4.29 kg. In that paper, the results of simultaneous sizing and geometric optimization are also presented. This optimization procedure results in a structure weighing 2.25 kg (Figure 4.18, in gray).

Some discussion on the difference between the two solutions presented in Figure 4.17 is warranted. The framework formulates the topology optimization problem as a minimization of a weighted average of the compliance corresponding to each load component. This approach to multiple load topology optimization is more likely to result in stable trusses than optimization as a single load case [e.g., Bendsøe and Sigmund, 2004]. This method also tends to produce structures in which the stiffness is less sensitive to the direction of loading. In Figure 4.17, the Chirehdast design is highly dependent on the orientation of the load vector at the top of the seat post. Multiple load optimization is well suited to practical conceptual design, since real-world structures are typically designed using load cases that are characterized by a significant degree of uncertainty. Along with the benefit of reduced sensitivity to load direction, the bicycle frame configuration produced in this work has fewer nodes and elements than the Chirehdast design. Design and fabrication costs of skeletal structures are highly dependent on the number of joints. In terms of cost, the advantage of a 4% weight saving (2.25 kg vs. 2.34 kg) is insignificant compared to a 25% increase in the number of joints (5 vs. 4).

Conceptual structural design is an iterative process where designers evolve and reformulate requirements as they gain knowledge about the design space. The following examples demonstrate how a user might interact with the framework to advance a conceptual design towards a practical final design.

In the bicycle frame shown in Figure 4.17, the axle of the front wheel is connected to the bottom bracket, preventing the bicycle from steering. To remove this connection, material is removed from the design domain, as shown in Figure 4.18.a. This change results in a vertical fork (Figure 4.18.b and 4.18.c). This configuration enables steering, but it implies a vertical steering axis, and very little *trail* (the distance from the front wheel contact point to the steering axis). Although this arrangement was used on some of the earliest bicycles, it is now well known that both stability and handling are much improved by using an inclined steering axis. An inclined axis produces greater trail, which allows for a self-centering caster effect.

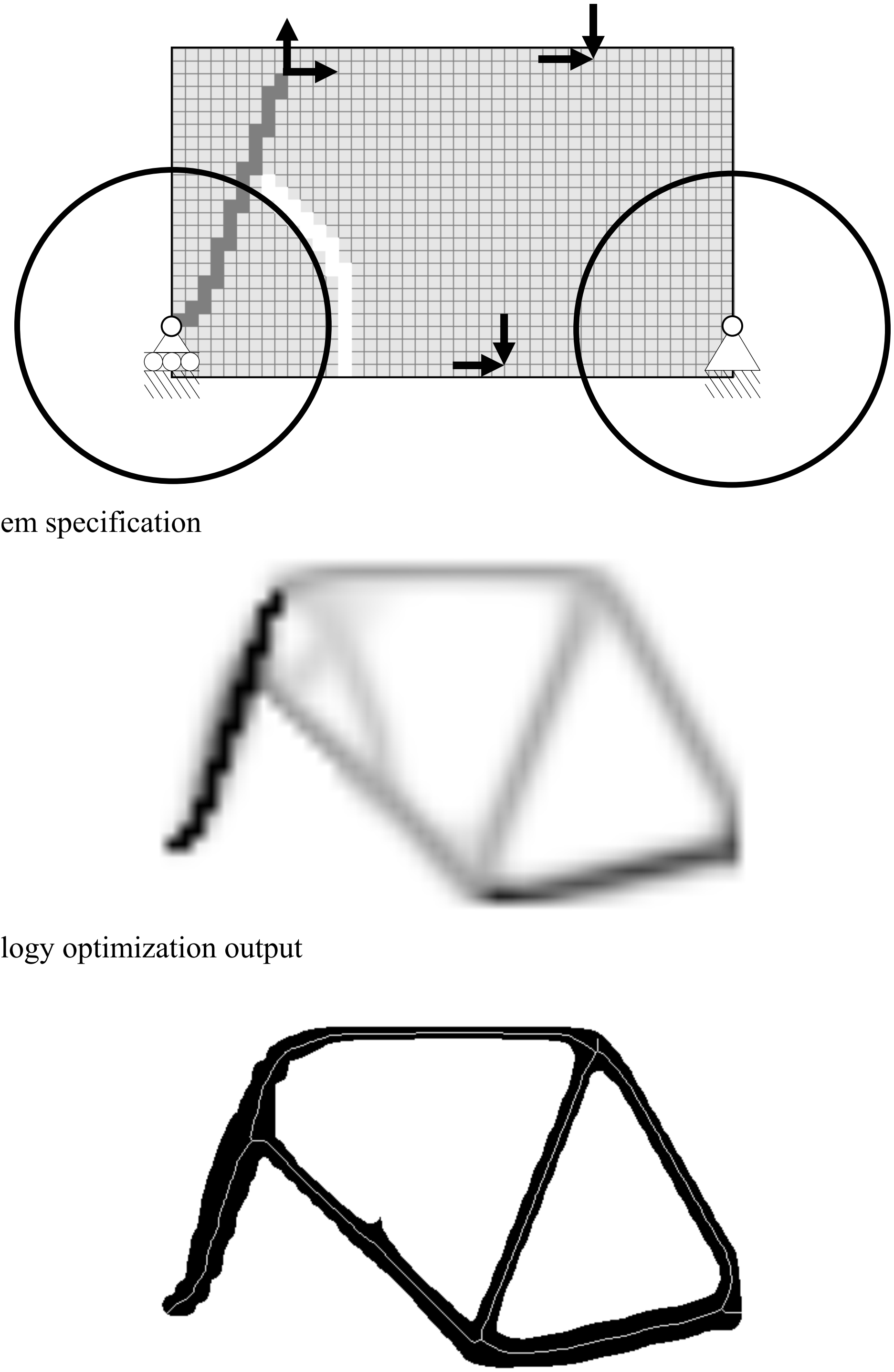


a) Problem specification

b) Topology optimization output

c) Skeleton

Figure 4.18. Bicycle frame respecification with steering



a) Problem specification

b) Topology optimization output

b) Skeleton

Figure 4.19. Bicycle frame respecification with fork

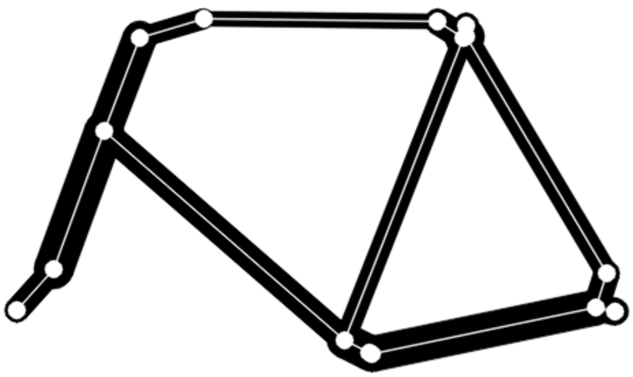


Figure 4.20. Bicycle frame and fork element model

To incorporate handling as a design requirement, an inclined steering axis and fork is incorporated in the design domain definition, as shown in Figure 4.19.a. The darker gray elements indicate that during topology optimization, these elements are constrained to have the full material density. The resulting topology of the frame changes (Figure 4.19.b and Figure 4.19.c) to provide increased stiffness in response to bending moments caused by wheel reactions on the fork. Tracing the path of the skeleton in Figure 4.19.c, results in the element model shown in Figure 4.20. Geometric optimization produces the design in Figure 4.21.

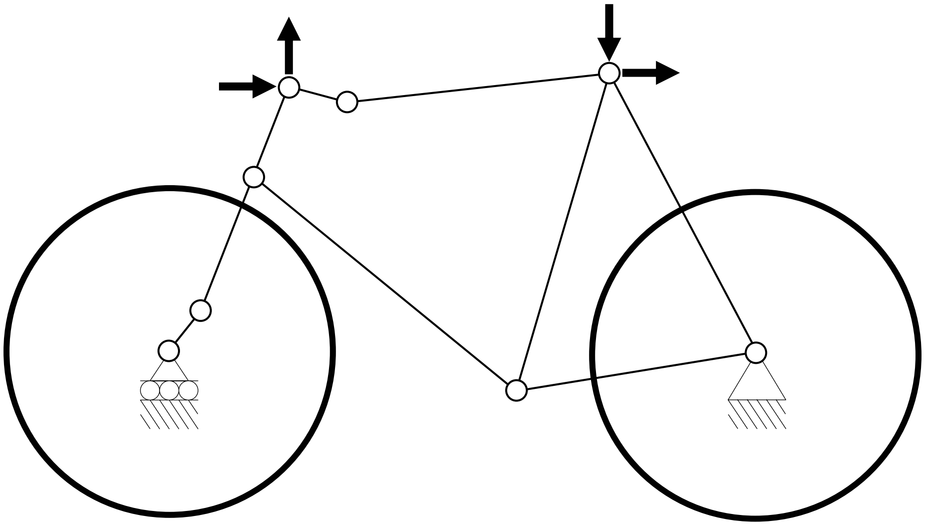


Figure 4.21. Bicycle frame redesign result

Stability optimization is performed on the redesigned bicycle frame in order to understand joint design requirements. The results are shown in Figure 4.22, where circles at the end of each member represent the required joint stiffness. The circles are filled with shades of gray corresponding to stiffness values ranging from zero (pinned) to one (rigid). Such information is an important consideration in designing cost effective joints.

Note that the stability optimization ensures that the stiffness of the frame with flexible joints is within 5% of the stiffness calculated assuming rigid joints.

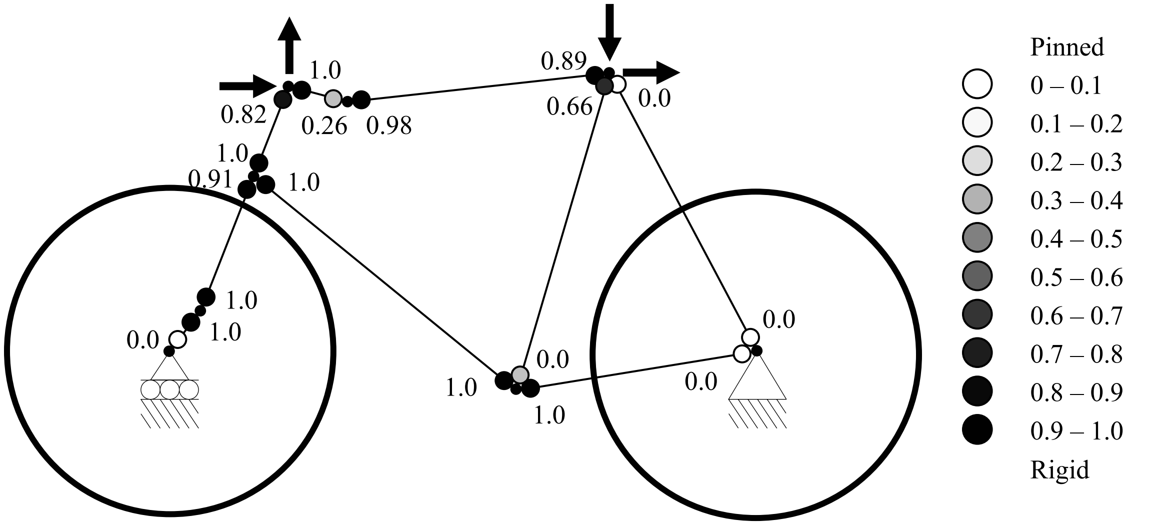
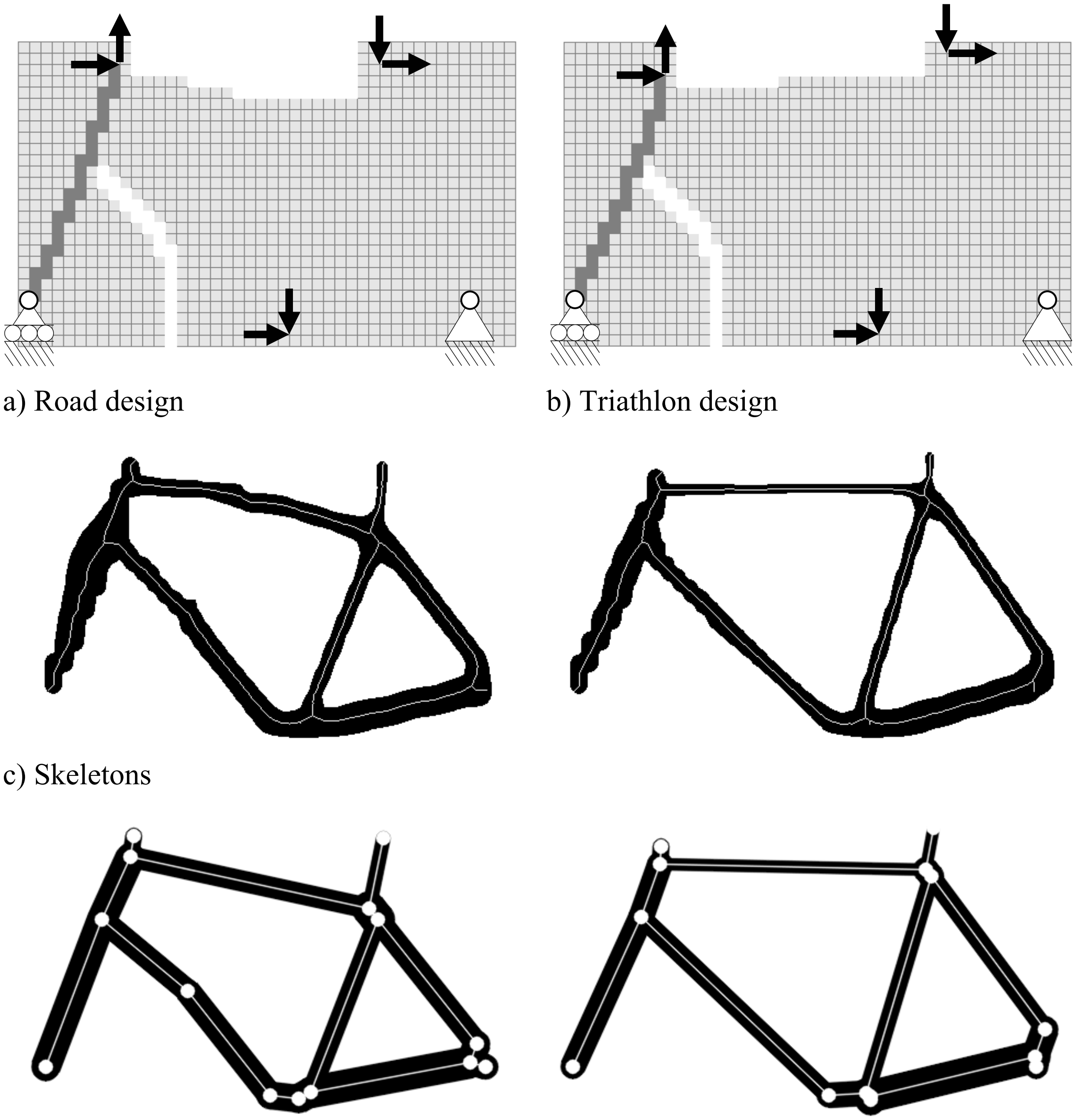


Figure 4.22. Bicycle frame stability analysis result

Visual case-based reasoning may be used during conceptual design to retrieve relevant information from a database of past design cases. The bicycle frame example is further expanded to demonstrate the application of the framework to visual case-based reasoning. An essential requirement in visual case retrieval is the measurement of similarity between forms. In order to form a database of design cases, two additional bike frame designs are generated by refining design requirements.

The shape of racing bicycle frames is driven in part by established regulations. Overcoming aerodynamic drag forces requires a significant portion of rider effort. Riders can substantially reduce effort by closely following in the slipstream of the rider ahead. This practice, called *drafting*, is well established in road racing but is prohibited in some forms of bicycle racing, such as in triathlons. Aerodynamic considerations are therefore of more importance in triathlon racing than road racing, and significantly influence the design of the bicycle. For triathlon racing, improved aerodynamics are achieved by rotating the position of the rider forward compared to road racing. As the position of the rider is rotated forward, the handlebars drop and the head tube becomes shorter. Also as the rider is rotated forward, the rider center of gravity shifts forward, and weight shifts from the rear wheel to the front. The weight redistribution is partially countered by moving the rear wheel forward. The wheelbase of the triathlon design is lengthened to improve handling with the changed weight distribution.

Figure 4.23 gives an example of the design of a road and triathlon bicycle frame, reflecting the differing requirements for these two applications. Figure 4.23.a and 4.23.b show the design domain for the road frame and triathlon frame, respectively. For the triathlon frame, there is a steeper angle between the seat and bottom bracket, the handlebars are lower, and the wheelbase is longer. Areas near the top of the frame are removed from the design domain to accommodate standover clearance requirements. Figure 4.23.c shows the binary topology optimization results and skeletons, and Figure 4.23.d contains the element models for the two frame styles.


a) Road design

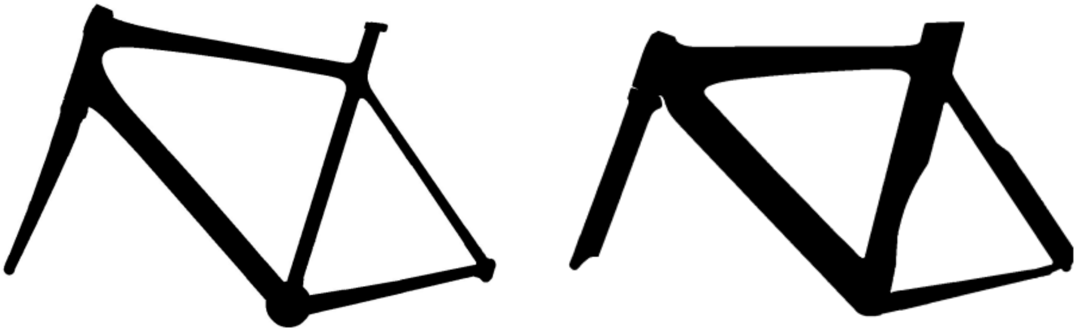
b) Triathlon design

c) Skeletons

d) Element models

Figure 4.23. Bicycle frame generation for two designs

Figure 4.24 shows binary images of two commercially-available frame designs, one for road racing and the other for triathlon. Note the similarity between these designs and the designs generated in Figure 4.23. The triathlon design appears thicker in profile primarily because the tubes have a more aerodynamic profile than the road bike tubes.



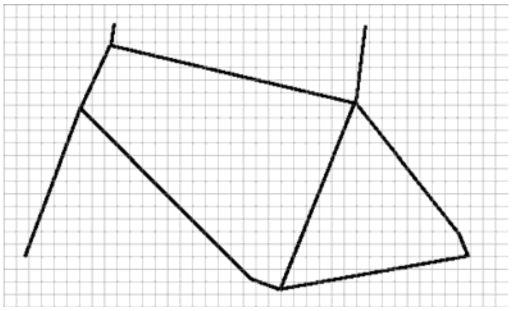
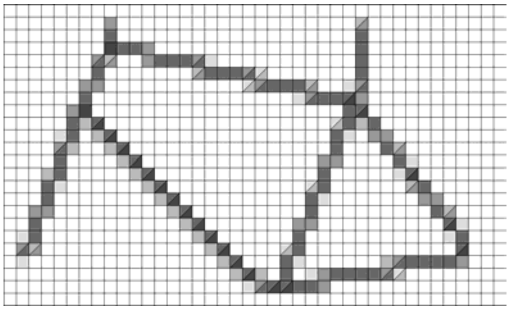
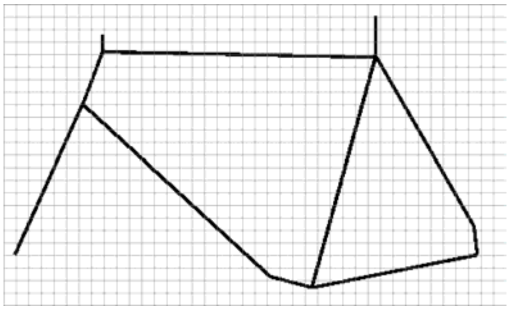
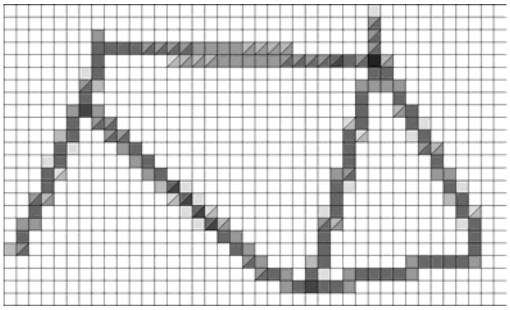
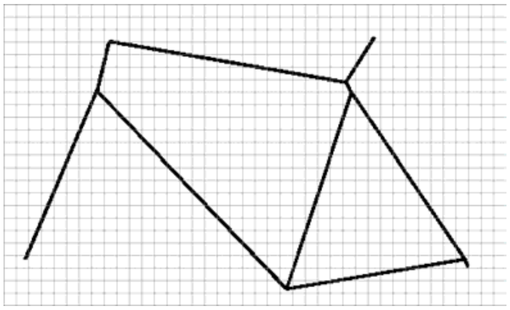
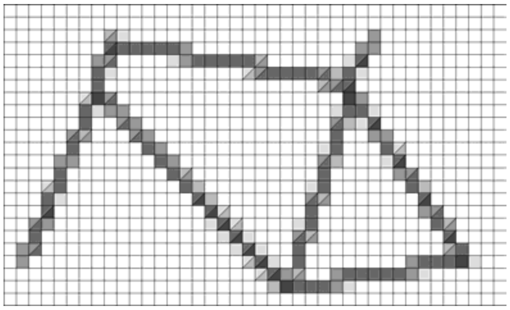
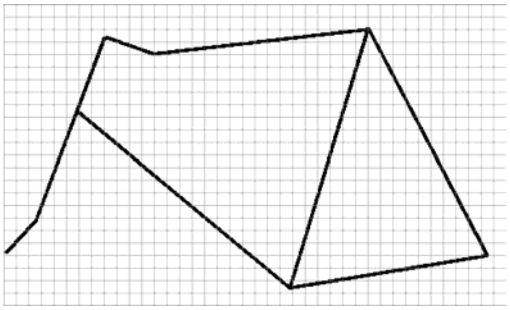
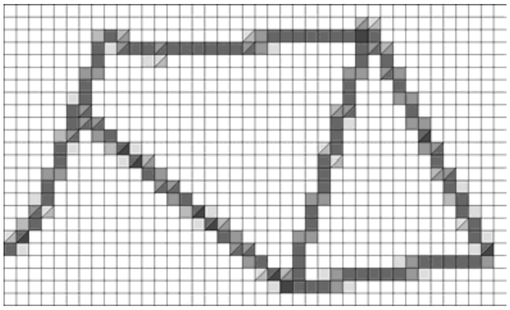
a) Road design b) Triathlon design

Figure 4.24. Bicycle frame commercial designs

To demonstrate the retrieval of similar designs in visual case-based reasoning, a case base of several different bicycle frame designs is assembled, as shown in Table 4.2. The source case corresponds to the road frame designed to the requirements of Figure 4.23.a. The target cases consist of previous examples from this Chapter, along with some test cases to demonstrate more dissimilar structures. The target image for the commercial design in Figure 4.24 is obtained by finding the image skeleton and performing path tracing to extract a simplified unit width representation. This processing is intended to highlight the key topology rather than member sizes.

The source case is sequentially compared to each target case in the case base, and the similarity between source and target cases is measured. The similarity is measured using the Object Density Map (ODM). The image resolution is 480 x 288 pixels in each case, with a grid size of 24 pixels. The ODM returns a number between 0 and 1, where 0 indicates the images are identical, and 1 signifies that one image is the negative of the other. To verify the algorithm, the source image is compared to itself and the similarity is zero, indicating a perfect match (Table 4.2.a.). The similarity measure for each source-target pair is presented in the right hand column of Table 4.2. The lowest similarity measure indicates the closest match between a source and target. The table indicates that the source image of the road frame derived using topology optimization most closely matches the commercial road design frame, as shown in Table 4.2.c. This shows the ODM algorithm is effective at calculating a quantitative measure of similarity between images of the kind used in structural conceptual design.

For visual case-based reasoning with practical cases, an effective similarity algorithm would need to deal with images of multiple size, as well as shifted or rotated images. Fortunately, image processing and pattern recognition research has produced many tools to deal with problems such as this.

Design	ODM Similarity Measure and Map
a. Source: Road design (Figure 4.23.a) 	0.0 
b. Target: Triathlon design (Figure 4.23.b) 	0.0597 
c. Target: Road design (Figure 4.24.a) 	**0.0427** 
d. Target: Bicycle frame (Figure 4.21) 	0.0513 

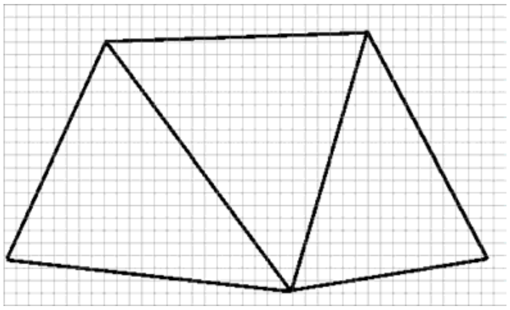
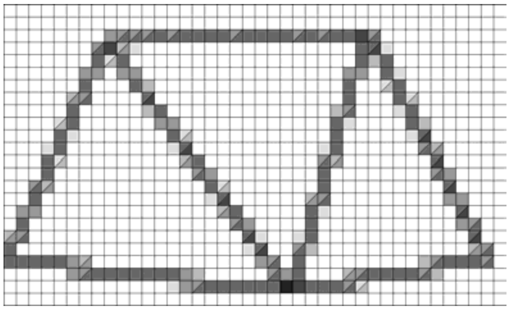
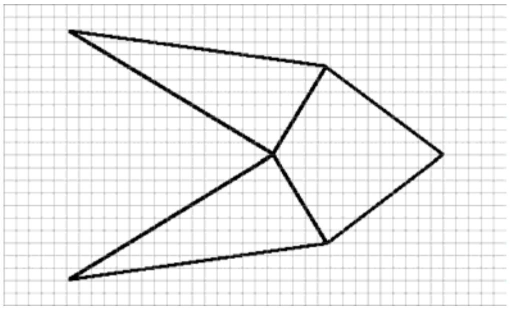
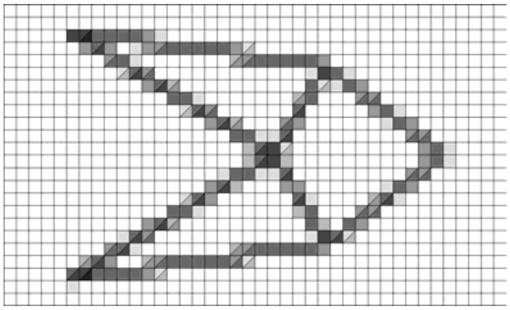
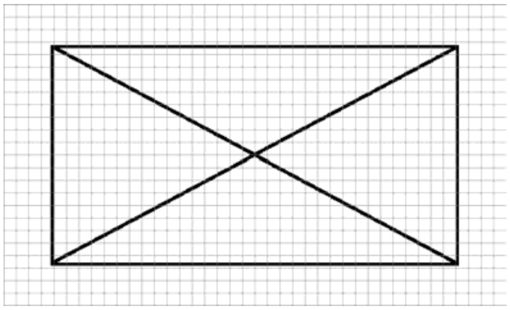
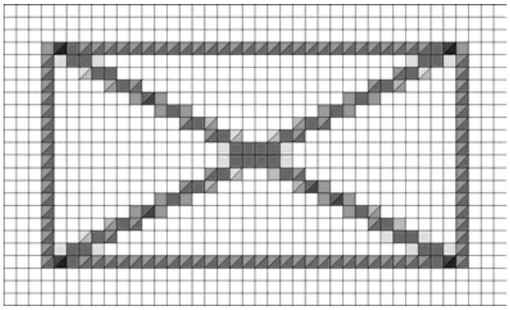
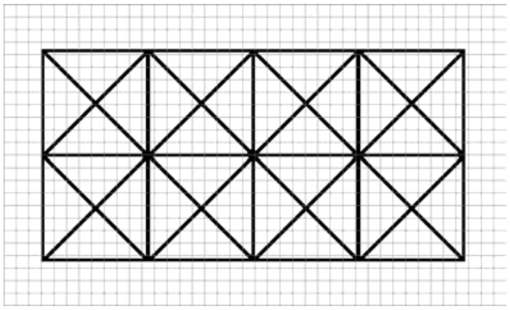
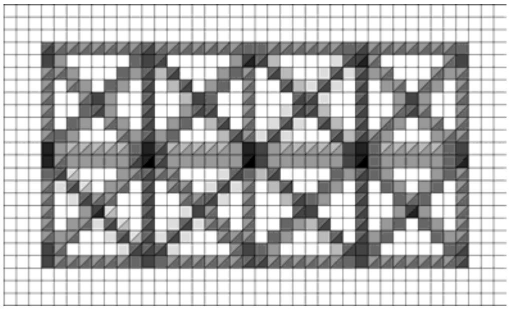
Design	ODM Similarity Measure and Map
e. Target: Bicycle frame (Figure 4.18)	0.0747
	
f. Target: Cantilever truss (Figure 4.6)	0.0735
	
g. Target: One-bay braced frame	0.0796
	
h. Target: Four-bay braced frame	0.171
	

Table 4.2. Bicycle frame case retrieval

In Table 4.2, the conventional diamond frame designs with an articulating fork (Table 4.2.b-d) have similarity values which are fair closely clustered, between 0.04 and 0.06. Interestingly, some relatively different topologies also have clustered values. For example, Table 4.2.e-g have values between 0.74 and 0.80. A much different configuration, the four-bay braced frame in Figure 4.2.h, has a measure of 0.17. These results indicate that for images of skeletal structures, which consist mostly of background pixels, similarity measurements will be generally closer to zero on the similarity scale than to one.

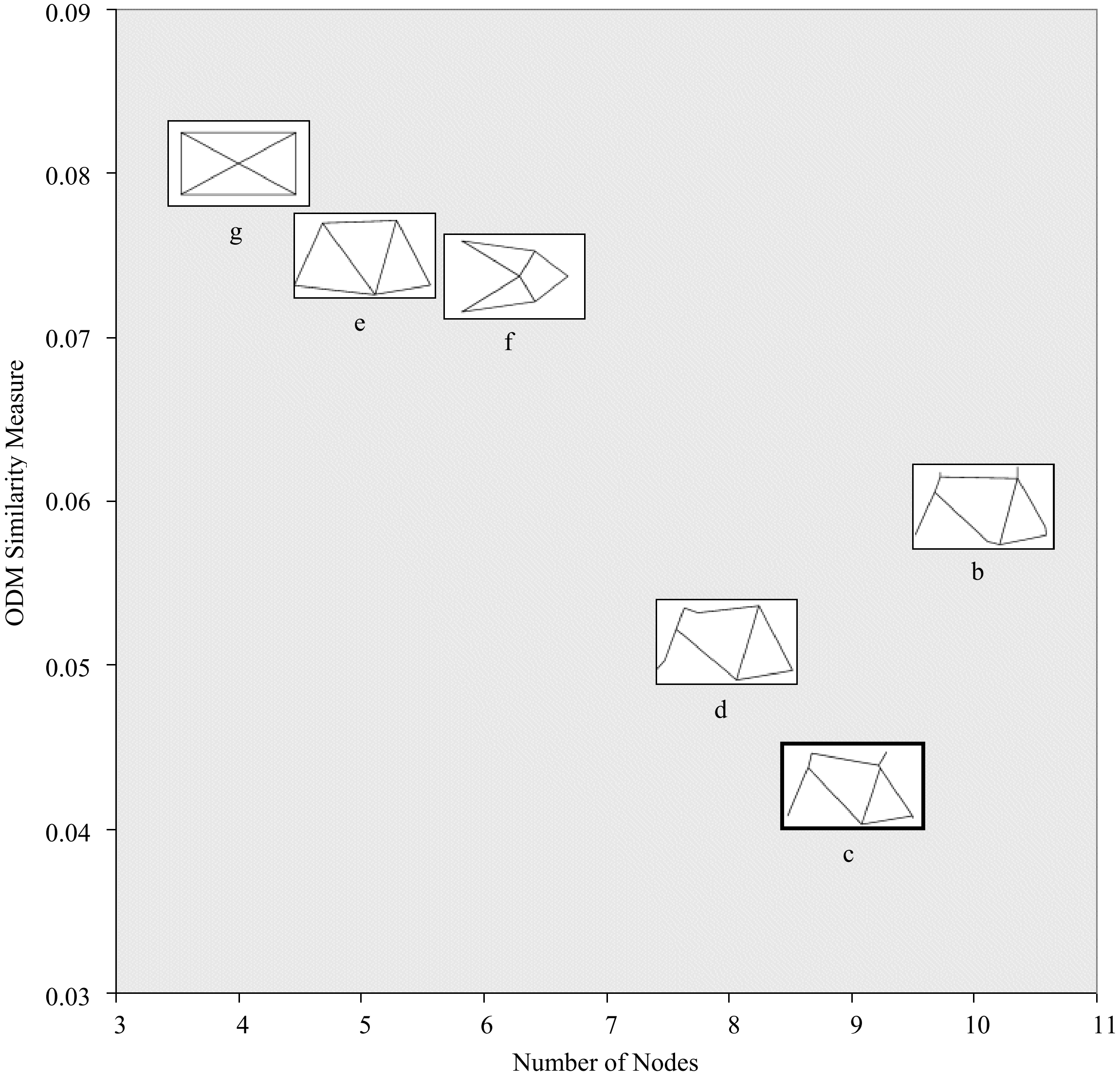


Figure 4.25. Bicycle frame retrieval results

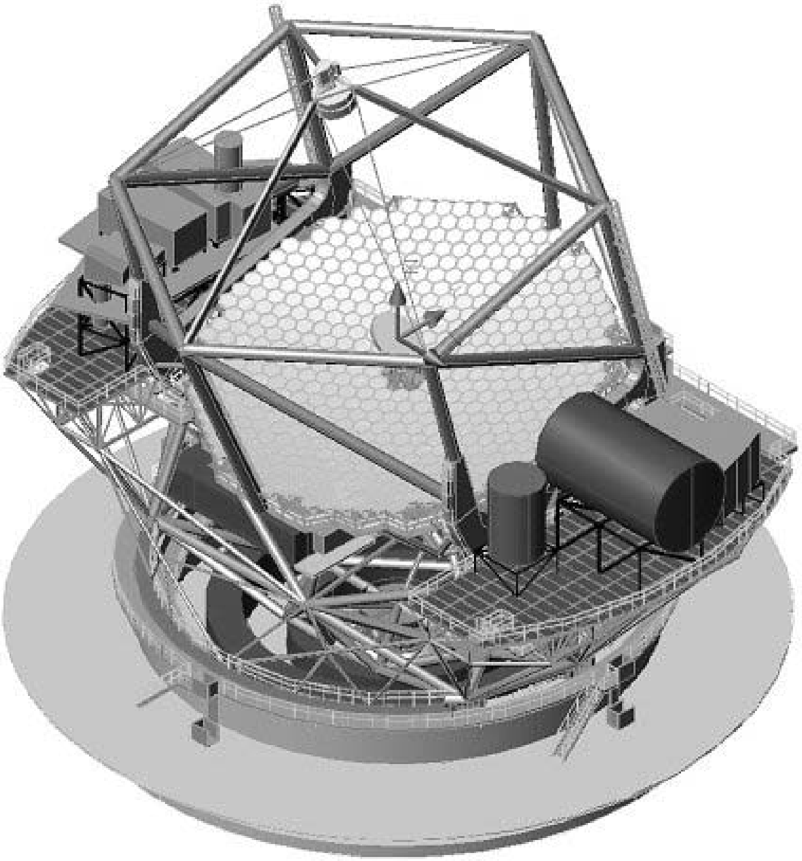
The case retrieval process should be an interactive one which facilitates the discovery of existing cases. Ideally, the process should return several cases which are similar, so that the user can browse and select an appropriate case. To facilitate the presentation of multiple target cases, retrieved images may be displayed in two or three dimensions, with one dimension used for the similarity measure and the other dimensions used for appropriate quantitative comparison values. In Figure 4.25, the similarity measure is plotted against the node count for each target case. The similarity measure is plotted on the ordinate, so the most similar designs appear closer to the bottom of the chart. In an interactive conceptual design system, it would be possible for the user to navigate a chart like Figure 4.25, panning across the case base and focusing in on cases of interest.

Visual case-based retrieval represents a potentially powerful method of using past design cases to assist in conceptual design. For the example given here, a bicycle frame image generated by topology optimization is matched with an image of an existing commercial frame design. Such an existing design could be stored with a large amount of information relating to design, fabrication, maintenance, and user feedback. For example, all the load cases and associated boundary conditions could be stored in the case base. Bicycles are typically designed for a number of load cases which are prohibitively time consuming to compute during conceptual design. Such cases include front wheel impact, rear wheel skidding, hill climbing and starting, high speed bump, and fatigue cases. Information on which load cases and combinations were critical in past work can be used to assist in effective and rapid evaluation of new design concepts. Visual case-based retrieval methods may also be applied to the input domain image, as opposed to the structural image. For example, the design domain image in Figure 4.19.a. could be used to retrieve past design cases with similar design domains, and similar requirements, to the target case.

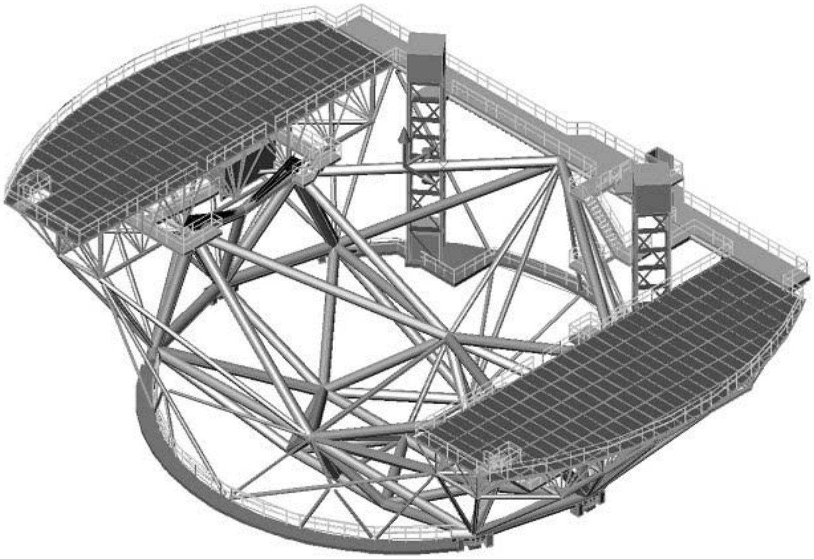
4.4. Telescope

The conceptual design of large telescope structures is a complex task. This case study concerns the design of a ground-based optical telescope with a thirty meter diameter segmented primary mirror. To date, the largest optical telescope has a ten meter primary, so the 30-meter telescope represents a large step forward, with ten times the light collecting area and over ten times the number of actuated primary segments. Given the unprecedented size of such a telescope, it is difficult to adapt specifications from existing telescopes. The telescope design concept must evolve with the requirements as more is learned about the potential performance and range of potential configurations. The telescope support structure payload includes almost 500 primary mirror segments weighing a total of 120 tonnes, an actuated secondary mirror (6 tonnes), an actuated tertiary mirror (10 tonnes), and instruments (170 tonnes).

A general requirement is that the support structure should be as stiff as possible in order to maintain the accurate alignment of the telescope optics. Although some optics components are actuated to compensate for telescope deformation under gravity, thermal and wind loads, increasing the actuator range increases the cost of the actuators and decreases their accuracy.



a) Telescope overall assembly



b) Telescope azimuth assembly

Figure 4.26. Telescope structure

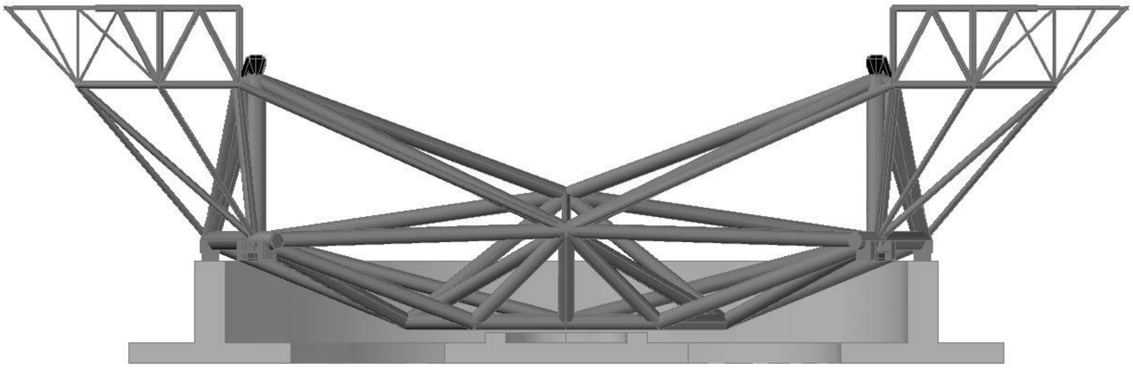


Figure 4.27. TMT telescope azimuth structure front view

In addition, it is generally cost-prohibitive to make all optics degrees of freedom active, so some deformation components are uncorrected and have a direct impact on the performance of the telescope. Even for the active degrees of freedom, a more flexible structure requires more frequent motion of the actuators, resulting in a more complex control system design.

The full telescope assembly is shown in Figure 4.26.a. The telescope has two drive axes: the vertical azimuth axis (±270º) and the horizontal elevation axis (0 to 90º from the horizon). The subassembly that rotates about the elevation axis, the elevation structure, supports the major optics systems. The azimuth structure (Figure 4.26.b. and Figure 4.27) supports the elevation structure and most of the instruments.

For the design of the telescope structure, the two most important objectives are to minimize compliance and mass. These two objectives are obviously in conflict, since adding mass to a structure, if done efficiently, will result in a decrease in compliance. An effective approach to developing a structural concept is to fix the mass and search for a configuration that minimizes compliance. The intent of the search is therefore to find the configuration that extracts the maximum stiffness per unit of mass. Although the concepts of minimizing compliance or maximizing stiffness sound simple, in large telescope structural design they are relatively complicated to implement. A large telescope has thousands of degrees of freedom. Some deflection components have relatively benign consequences on the optical performance of the telescope. Some deflections may be corrected by moving optical components with actuators. In some cases, the telescope may be relatively flexible in a given direction, but no significant loads are applied in that direction.

In telescope design it is important to distinguish between quasi-static and dynamic stiffness. Quasi-static stiffness refers to the stiffness of the structure in response to relatively slowly changing loads, such as gravity or thermal loads. As the telescope rotates in elevation, the direction of the gravity load vector relative to the optical components changes, resulting in changes to the deflected shape. One of the functions of the optical actuators is to change the position of the mirrors to compensate for these deflections. The required actuator range, or stroke, is directly related to quasi-static stiffness.

Dynamic stiffness refers to the stiffness of the structure in response to dynamic loading. In large telescope structures, the primary source of dynamic loading is wind. The response of the telescope to wind load is related to both the structural system and the control system. In general, increasing the first natural frequency of the structural system allows for greater control system bandwidth, and an increased ability to reject disturbances caused by wind loads. In some cases, vibration modes other than the first mode drive the design of the control system. In detailed design, it is therefore important to understand how the various vibration modes of the structure respond to drive system inputs.

Mass minimization is an objective in telescope structural design for several reasons. Reducing the mass of the structure lowers its cost, within limits. As structures become increasingly lighter, at some point fabrication costs begin to increase. For example, lightweighting a material through the added fabrication step of cutting holes increases cost. More important reasons for minimizing telescope structure mass are to reduce thermal and mechanical inertia. Temperature control of telescope structures is extremely important to their performance, and reduced thermal mass allows temperature to be more easily controlled. Larger structural mass and inertia increase the initial cost and operating cost of mechanical systems such as drive motors.

In optimizing the structure for minimum compliance and mass, a large number of design constraints must also be considered. The major constraints relate to geometry, member resistance and acceleration.

Geometric constraints ensure the telescope structure subassemblies achieve the required range of motion without interference. For example, clearances between the elevation and azimuth structures must be maintained through the full 90 degree elevation angle range. The telescope enclosure represents a significant portion of overall observatory costs, and its costs are strongly dependent on the size of the telescope swept volume. To reduce costs, the swept volume should be minimized. An important goal of telescope structure design is to ensure that light paths between various optical elements are as unobstructed as possible, so the paths constrain the placement of structural material. Finally, since wind loads tend to degrade telescope performance, they should be minimized. This objective is often interpreted as a geometric constraint on the size and shape of member cross-sections, since both size and shape influence aerodynamic drag forces.

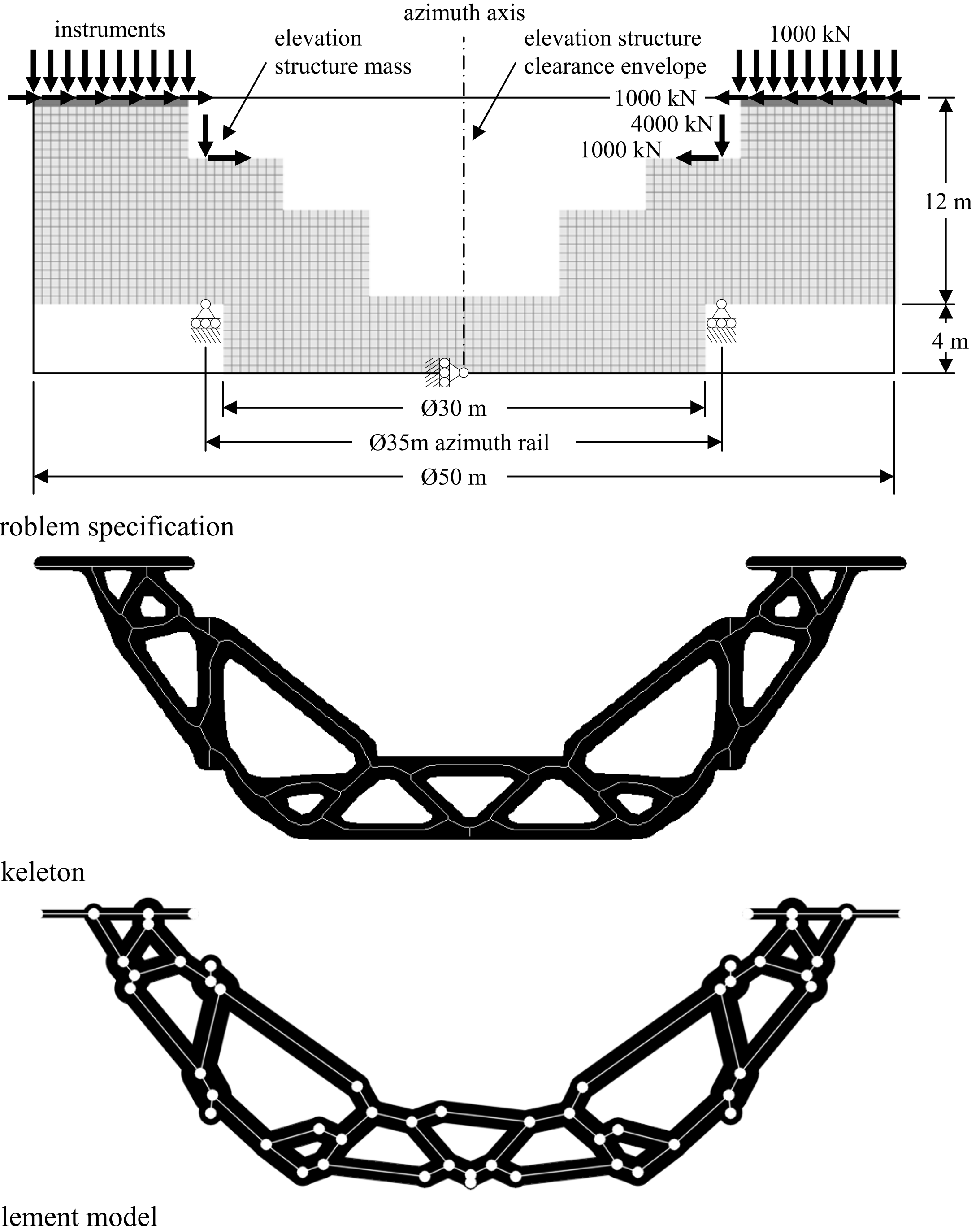
Stress constraints are applied to ensure that the telescope structure remains elastic under “operating basis earthquakes”, to reduce the amount of downtime required to bring the telescope back into operation after smaller earthquakes. For larger earthquakes, some damage is expected, however the design intent is usually to limit the damage to more easily replaceable elements. Acceleration constraints are used to limit the transmission of damaging seismic acceleration components to sensitive on-board optical systems. Deflection constraints are often required at critical structural-mechanical interfaces. For example, hydrostatic bearing systems operate on a thin film of oil which is compromised by structural deflections. The deflections should be constrained to maintain the required oil gap and ensure the system performs as expected.

The specific telescope case study described in this section concerns the conceptual design of the azimuth structure for the thirty-meter telescope. Obviously, it is a challenge to address all elements of the complex set of requirements described above at the conceptual design stage. As a starting point, the framework described in this work can be used to generate a number of topologies that represent optimal, minimum compliance structures for a range of different loading conditions.

For initial conceptual design, a two dimensional model of a vertical section through the azimuth structure is studied. Three different load cases are presented in Figure 4.28 through figure 4.30. Figure 4.28.a defines the geometric constraints and boundary conditions. The design domain is shaped by three geometric constraints; these include subtracted areas for elevation structure clearance and azimuth track clearance, and retained areas at the top level of the azimuth structure which contains instrument interfaces. The three load cases in Figure 4.28 through Figure 4.30 represent different combinations of dead load and simulated dynamic loads. All load cases are analyzed statically. Simulated dynamic loads are applied to increase stiffness in a given direction, resulting in increased natural frequencies. Static load cases may be used in this way to influence frequencies for mode shapes that are expected to create difficulties for the control system. Although this approach is approximate, it is suited to the conceptual design stage, where the designer seeks an understanding of how topology affects stiffness in various directions.

Figure 4.31 presents the results of node position optimization using topology generated in Figure 4.28 through Figure 4.30. The resulting designs are seen to have a relatively strong dependence on the load case. By applying different load cases and combinations, the designer gains insight into potential design configurations that are candidates for more detailed analysis and design.

A stability analysis was conducted for Case 3. Connection stiffness requirements are shown in Figure 4.32. The grayscale value in the filled circles represents the release values at each member endpoint.

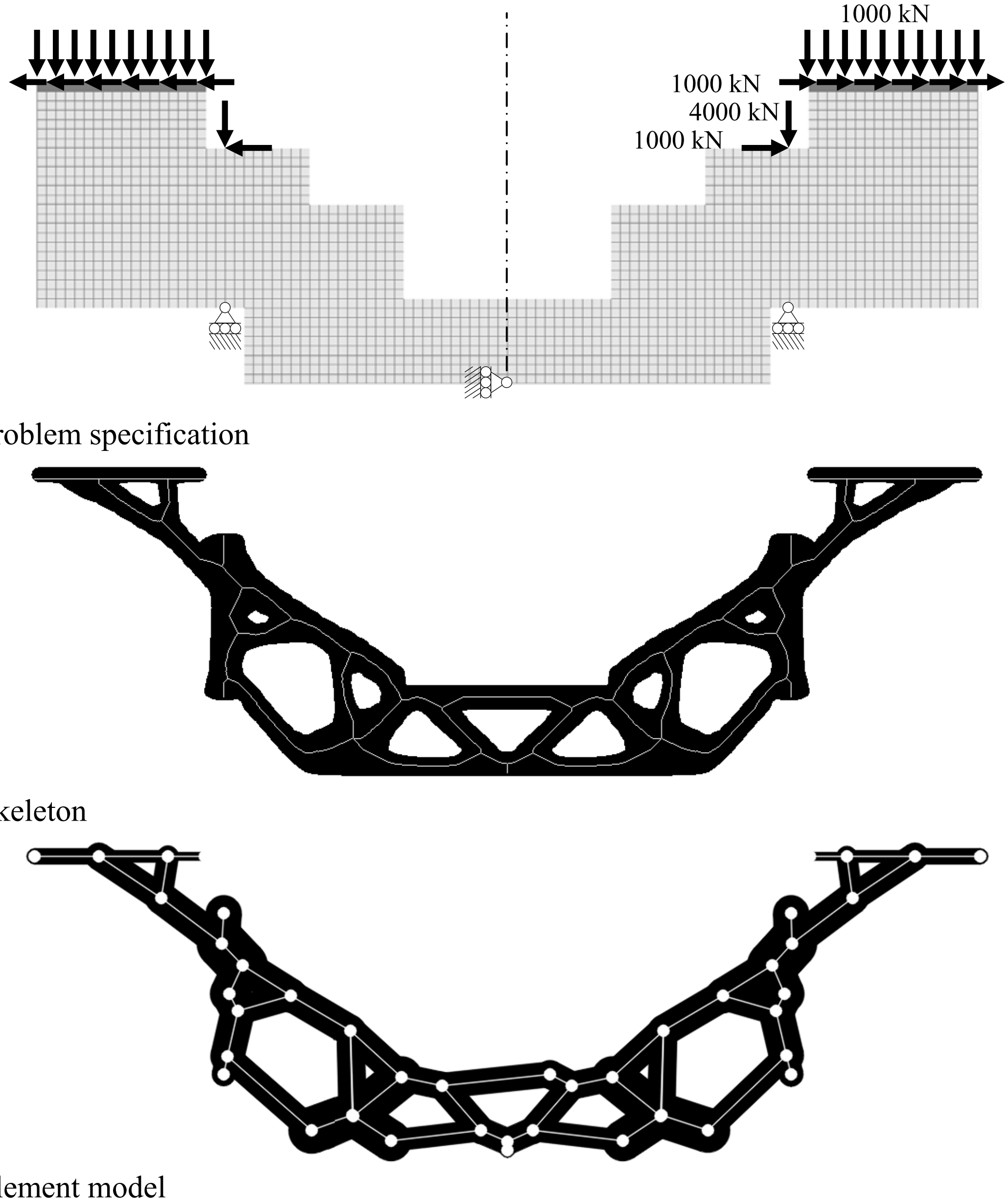


a) Problem specification

b) Skeleton

c) Element model

Figure 4.28. Telescope load case 1

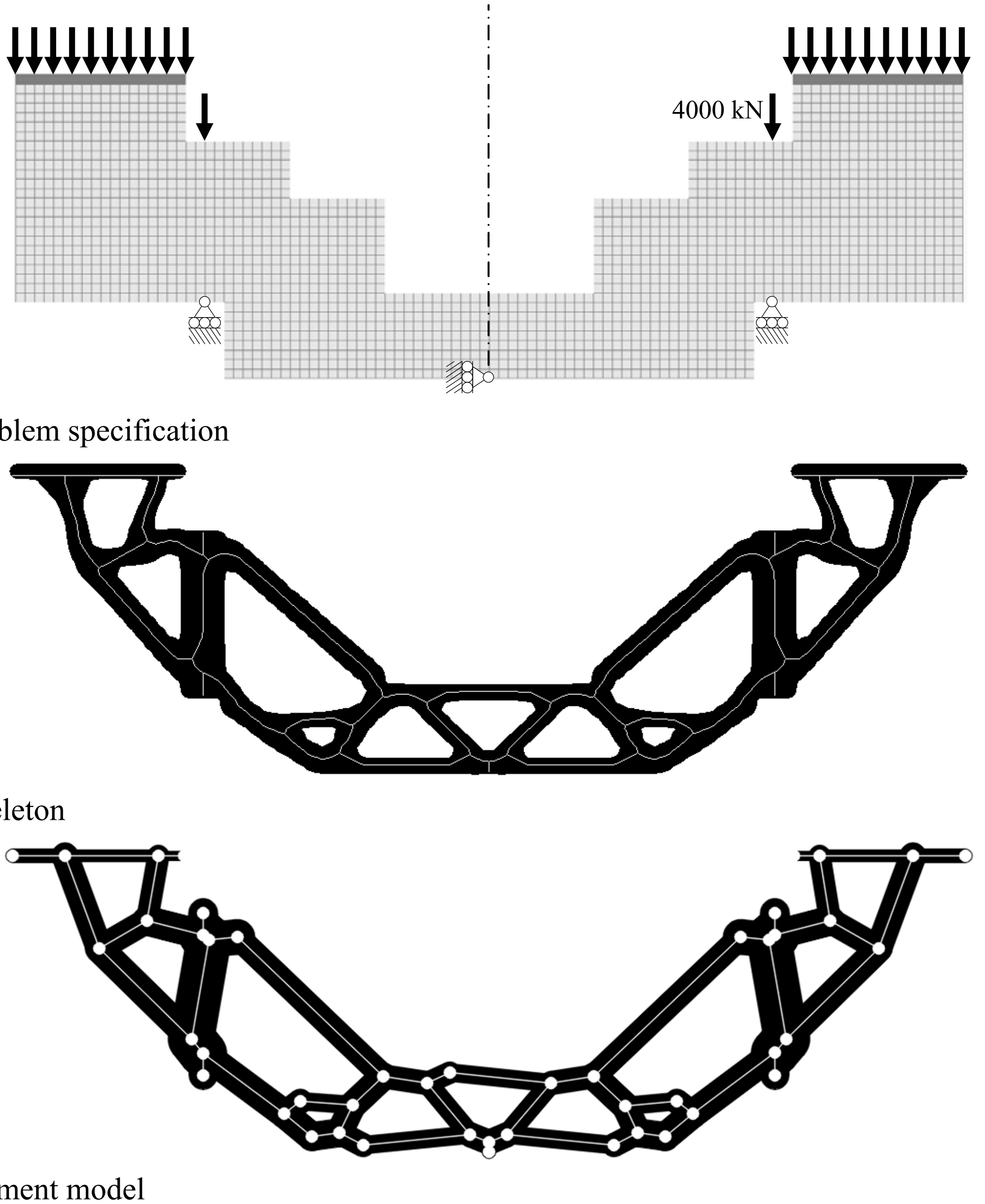


a) Problem specification

b) Skeleton

c) Element model

Figure 4.29. Telescope load case 2

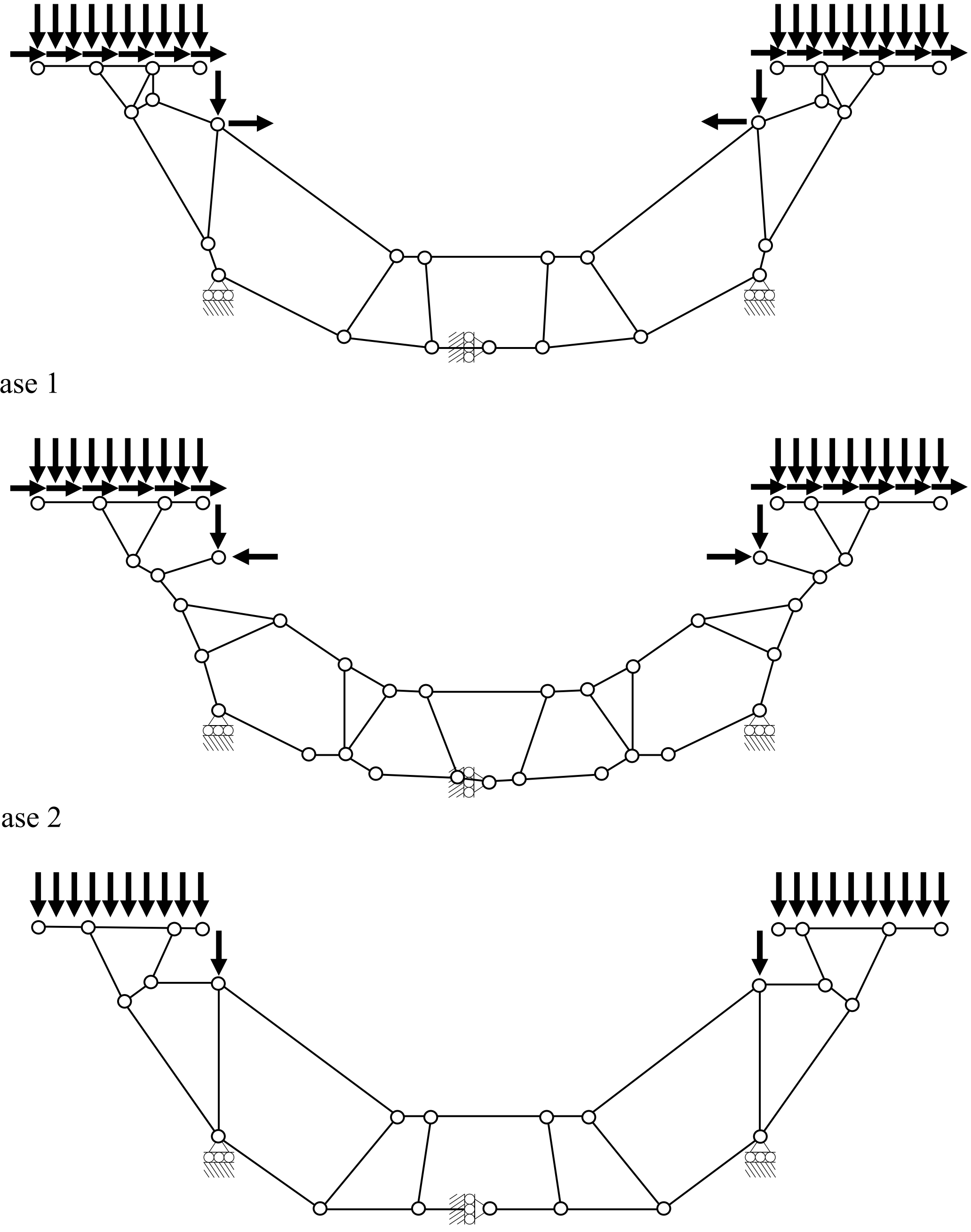


a) Problem specification

b) Skeleton

c) Element model

Figure 4.30. Telescope load case 3



a) Case 1

b) Case 2

c) Case 3

Figure 4.31. Telescope topology after Refinement Phase

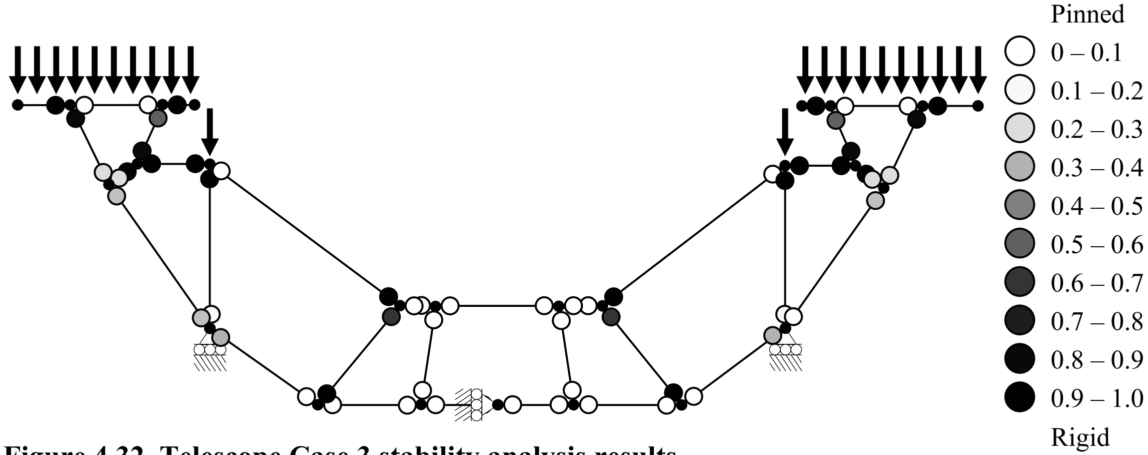


Figure 4.32. Telescope Case 3 stability analysis results

The required connection stiffness has significant implications for overall design and cost. Large optical telescope structures are typically installed at remote, high elevation sites, where working conditions are difficult. Large telescope structures are generally pre-assembled at the fabrication shop to verify the fit of the components. The rationale for trial assembly is that fabrication errors can be corrected more economically at the shop than in the field. For telescopes in the 30-meter class and larger, trial assembly is a very significant part of the overall fabrication budget. In order to reduce costs, there is pressure to reduce the amount of trial assembly required. Large telescope structures often use tubular members with bolted flange connections, as shown in Figure 4.33. To ensure predictable and repeatable performance of the structure, flange connections require accurate alignment, resulting in relatively uniform contact pressure and bolt tension. To verify the alignment of bolted flange connections, this type of connection requires trial assembly, which often incorporates some final adjustment and fit-up.

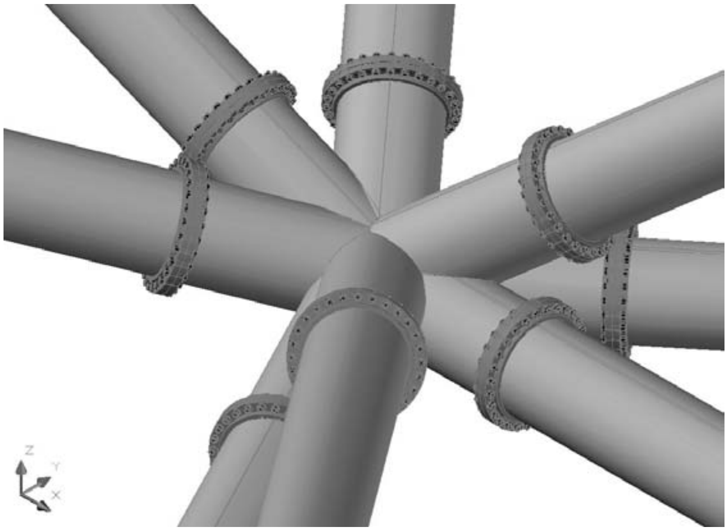


Figure 4.33. Telescope bolted flange connection

As an alternative to flange connections, bolted shear connections can be employed. Shear connections allow a larger degree of field adjustment than flange connections, and may be used strategically to reduce the size of the trial assembly units. Connection stiffness requirements similar to those presented in Figure 4.32 may be used to guide the selection of joint designs to reduce overall connection costs and trial assembly costs.

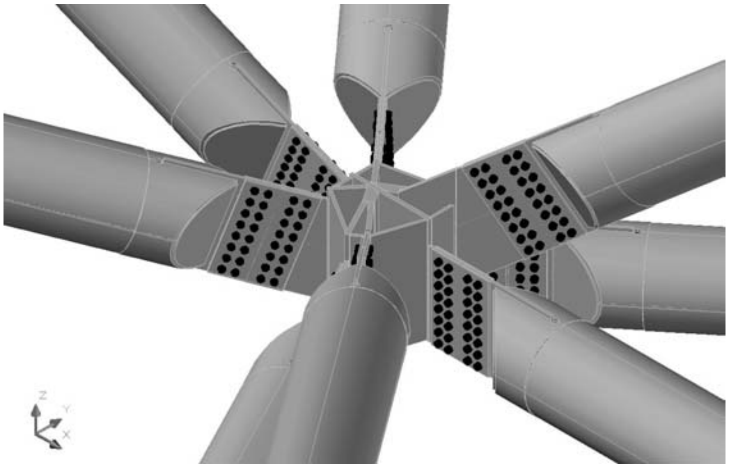
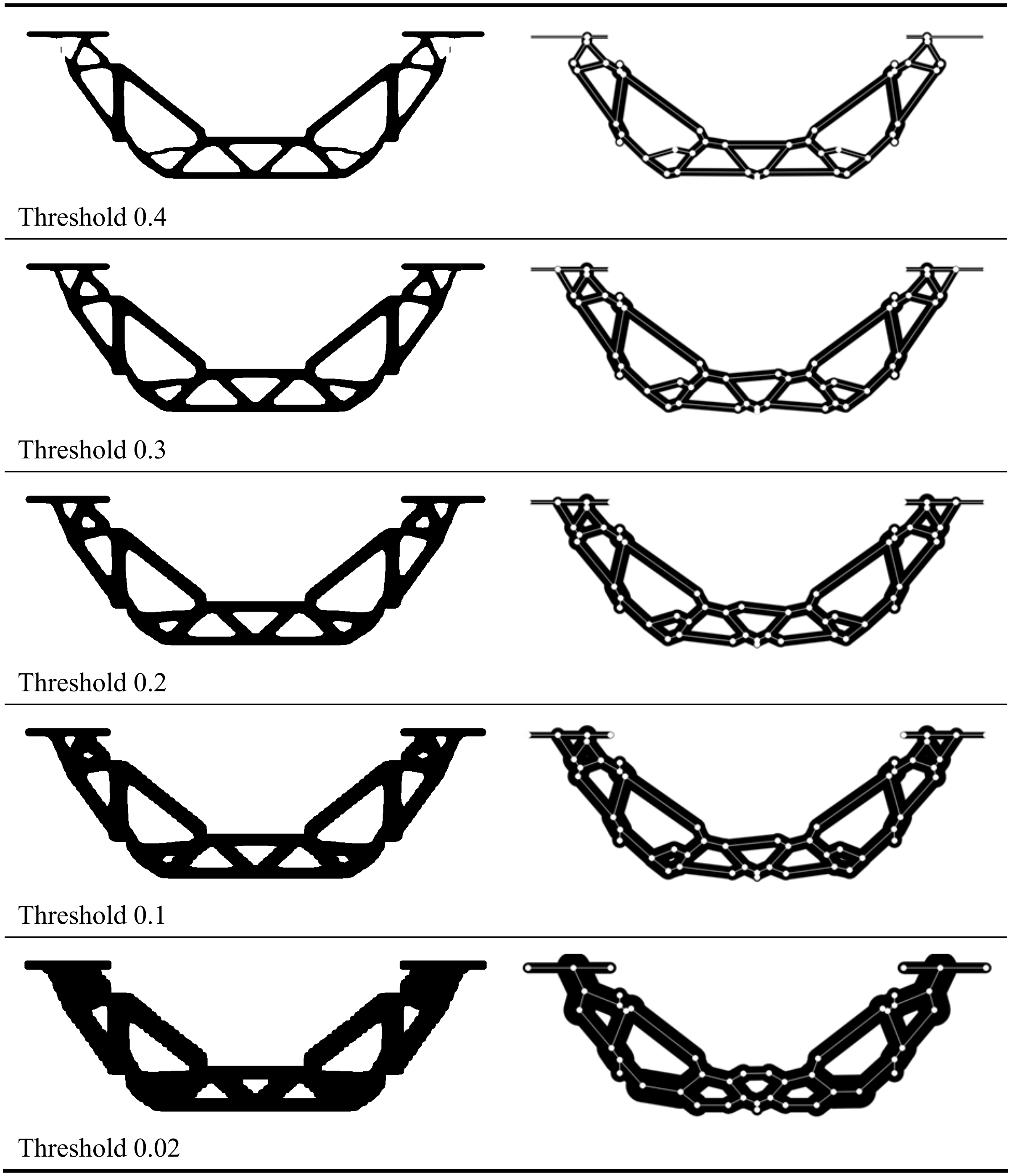


Figure 4.34. Telescope bolted shear connection

Multiple design configurations may be generated by modifying the parameters controlling topology synthesis. Figure 4.35 shows a series of designs produced by varying the threshold levels used in converting topology optimization output to a binary image. Figure 4.36 shows a range of designs produced by varying the volume fraction used in topology optimization. The topology optimization produces more “truss-like” structures for low volume fractions and high thresholds. For higher volume fractions, the members become thicker and bending stiffness becomes more significant. If the volume fraction is too high, then voids are filled and large solid areas are produced. If the volume fraction is too low or the threshold too high, then areas become disconnected and relatively weak or unstable structures may result. For most of the examples presented here, volume fractions between 0.1 and 0.3, and a threshold of 0.2 were used. For practical topology optimization, the output generally includes a range of different grayscale values. In the telescope example, the central portion of the structure contains large, heavy members, while smaller members are required under the cantilevered instrument platforms. When a single threshold value is used, the threshold value is usually determined by the finer structural elements, represented by the lighter grayscale values, in order to avoid losing such structures during binary conversion. The result is that the darker elements become thicker when converted to binary format, and carry a larger proportion of bending than the lighter elements.

Each design is processed using the techniques of this framework. Designers select the most appropriate concepts for detail design work, based on both quantitative and qualitative evaluation criteria.



Threshold 0.4

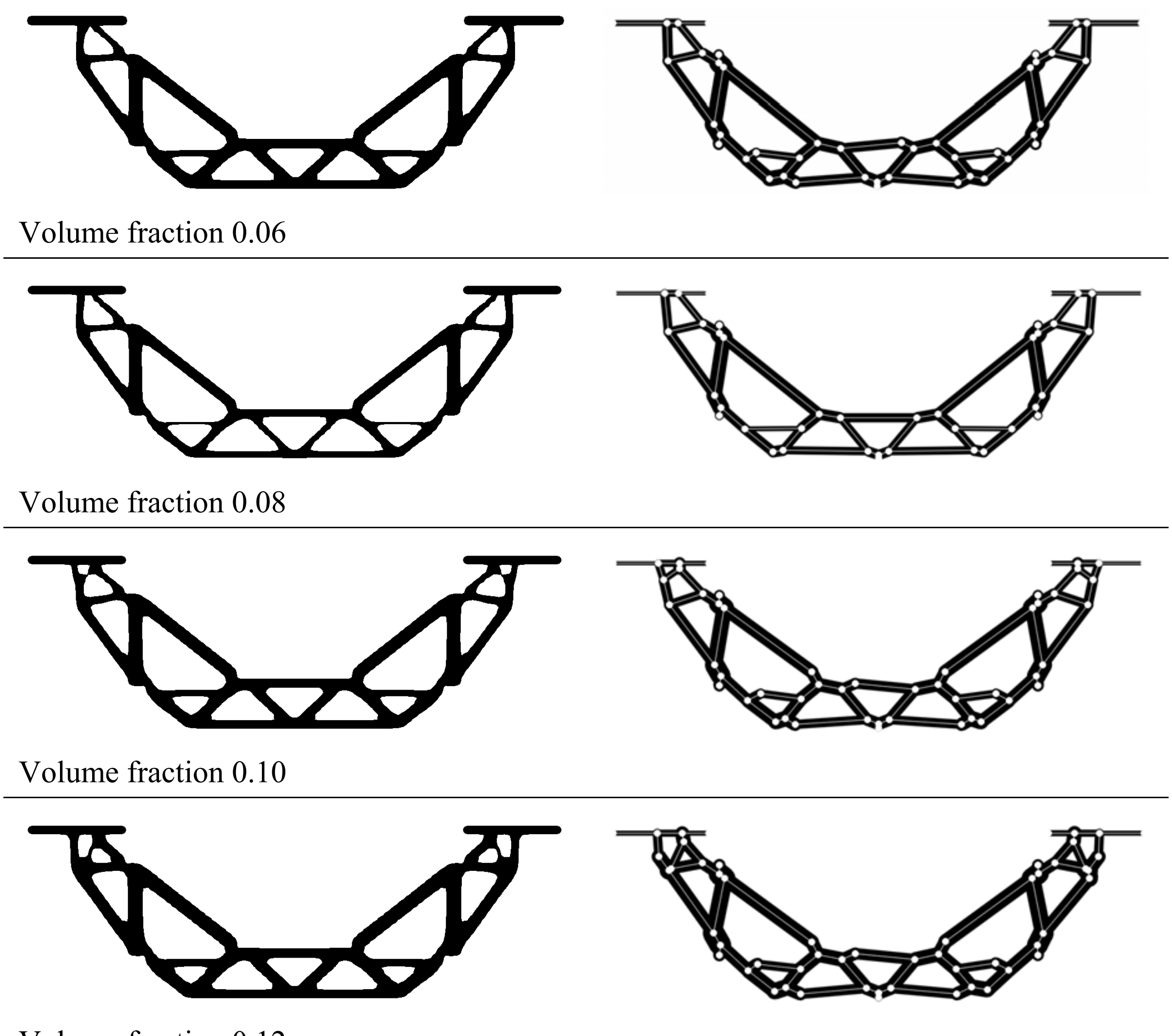
Threshold 0.3

Threshold 0.2

Threshold 0.1

Threshold 0.02

Figure 4.35. Telescope configurations for threshold variation



Volume fraction 0.06

Volume fraction 0.08

Volume fraction 0.10

Volume fraction 0.12

Figure 4.36. Telescope configurations for volume fraction variation

4.5. Roof

In a case study in architecture, conceptual designs are generated for a roof system covering the platforms of a train station. This example was inspired by the design of the Lisbon Orient train station by the architect Calatrava, shown in Figure 4.37. The problem is formulated as shown in Figure 4.38. The domain represents and area 16 meters wide by 20 meters high. A large architectural clearance area is specified above the station platforms and rails. Another clearance area defines the architectural roof line. A linear retained area is defined along the roof line, to provide support for roof cladding. The design objective is to find a series of topologies that minimize the compliance for a given structural weight.

Three load cases are studied, as presented in Figure 4.39 through 4.41. The load cases correspond to gravity load (Figure 4.39), wind load (Figure 4.40) and a combination of gravity, wind and seismic load (Figure 4.41). The critical load combination cannot be definitively established until the detailed design stage. At the conceptual stage, the critical combinations need to be estimated.

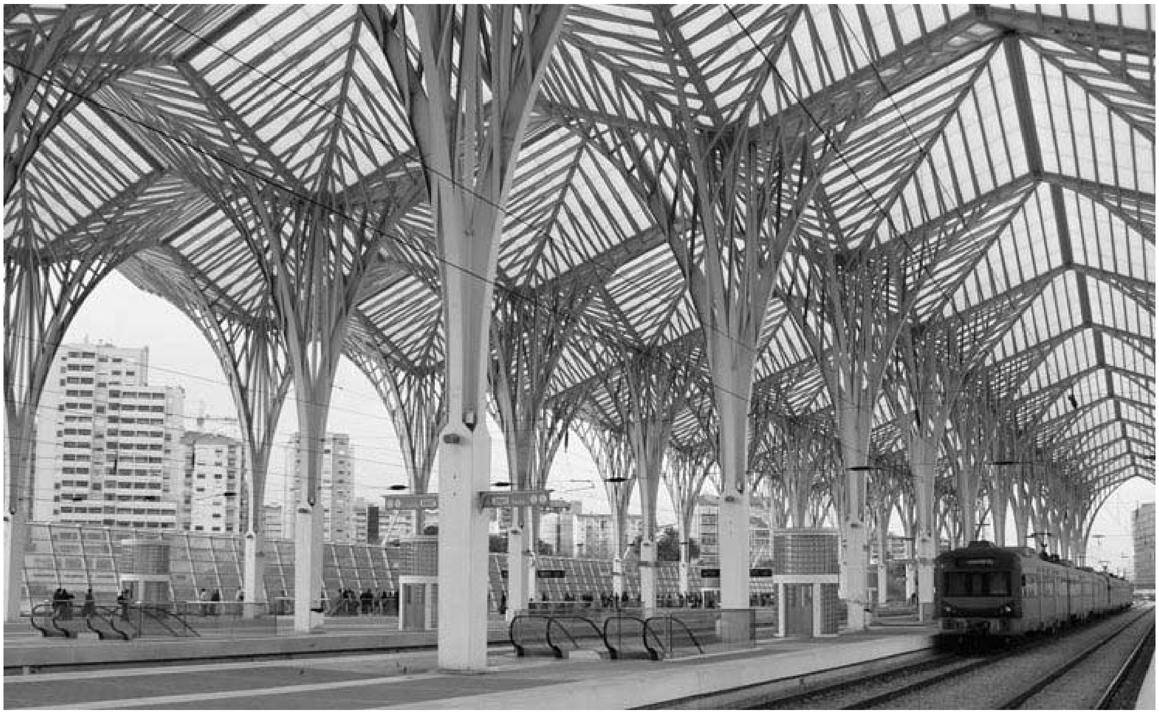


Figure 4.37. Lisbon Orient train station by architect Calatrava

As in the previous case study, the resulting topology is strongly dependent on the loading configuration. The results show that the proposed framework can generate designs that are architecturally varied and in some cases unexpected, and can be an extremely powerful tool in supporting creative conceptual design work.

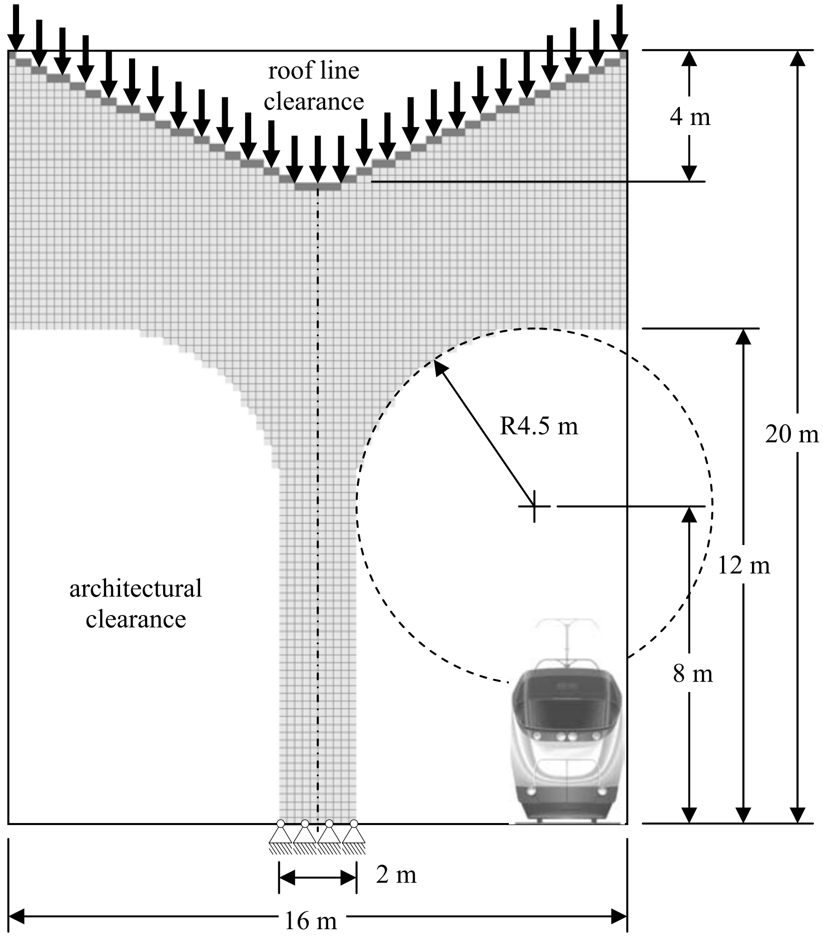
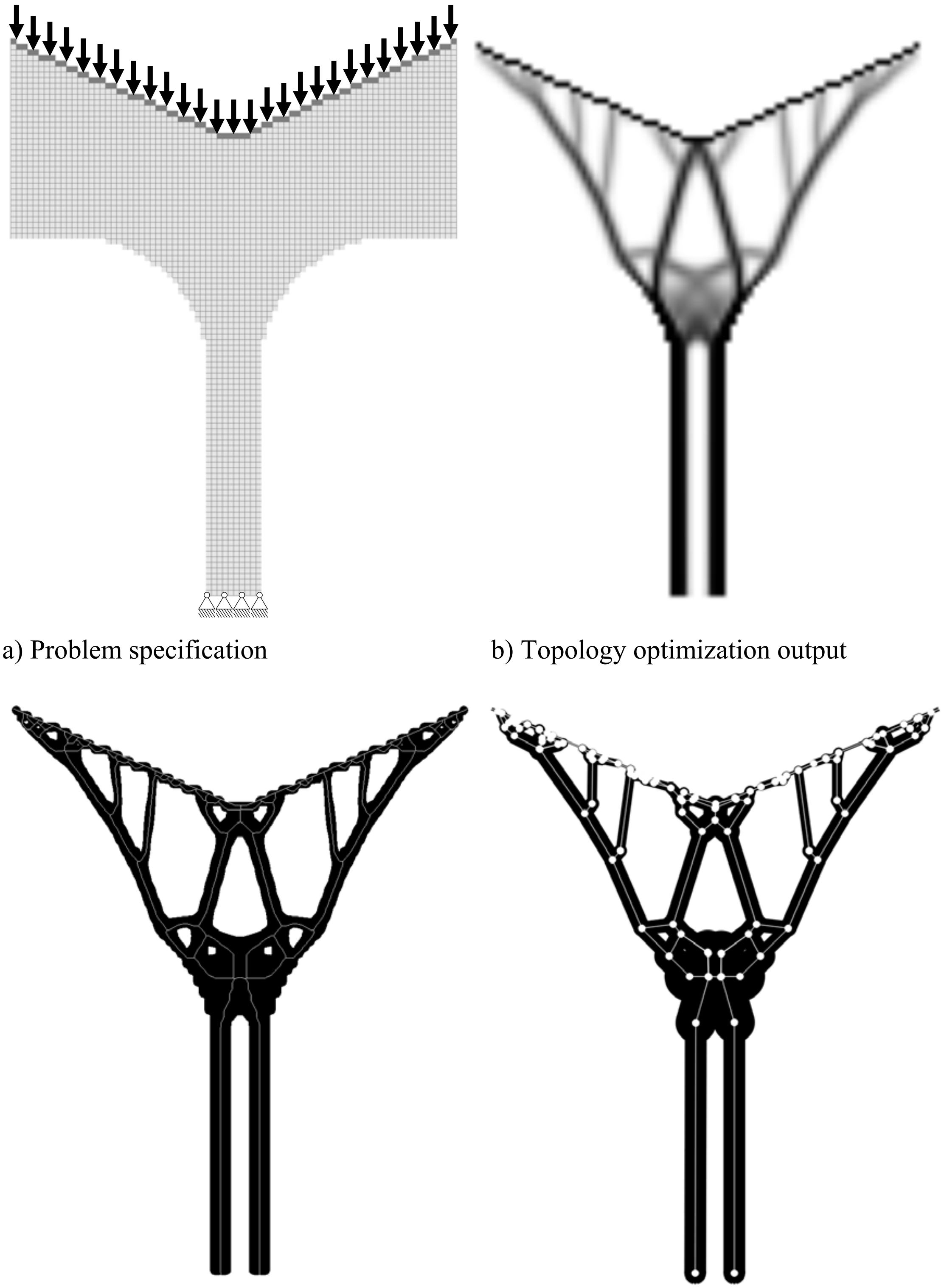


Figure 4.38. Roof problem specification



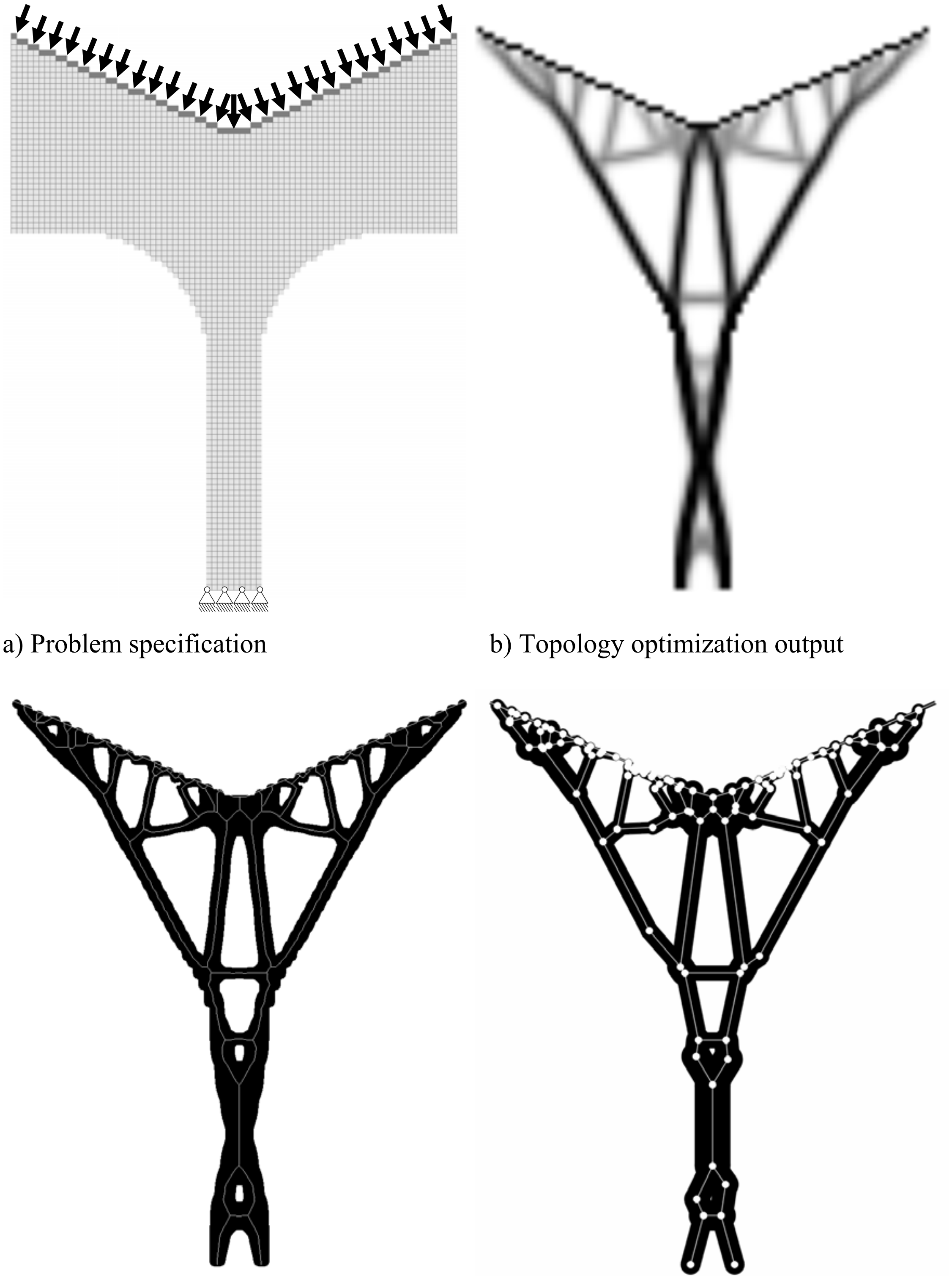
a) Problem specification

b) Topology optimization output

c) Skeleton

d) Element model

Figure 4.39. Roof load case 1



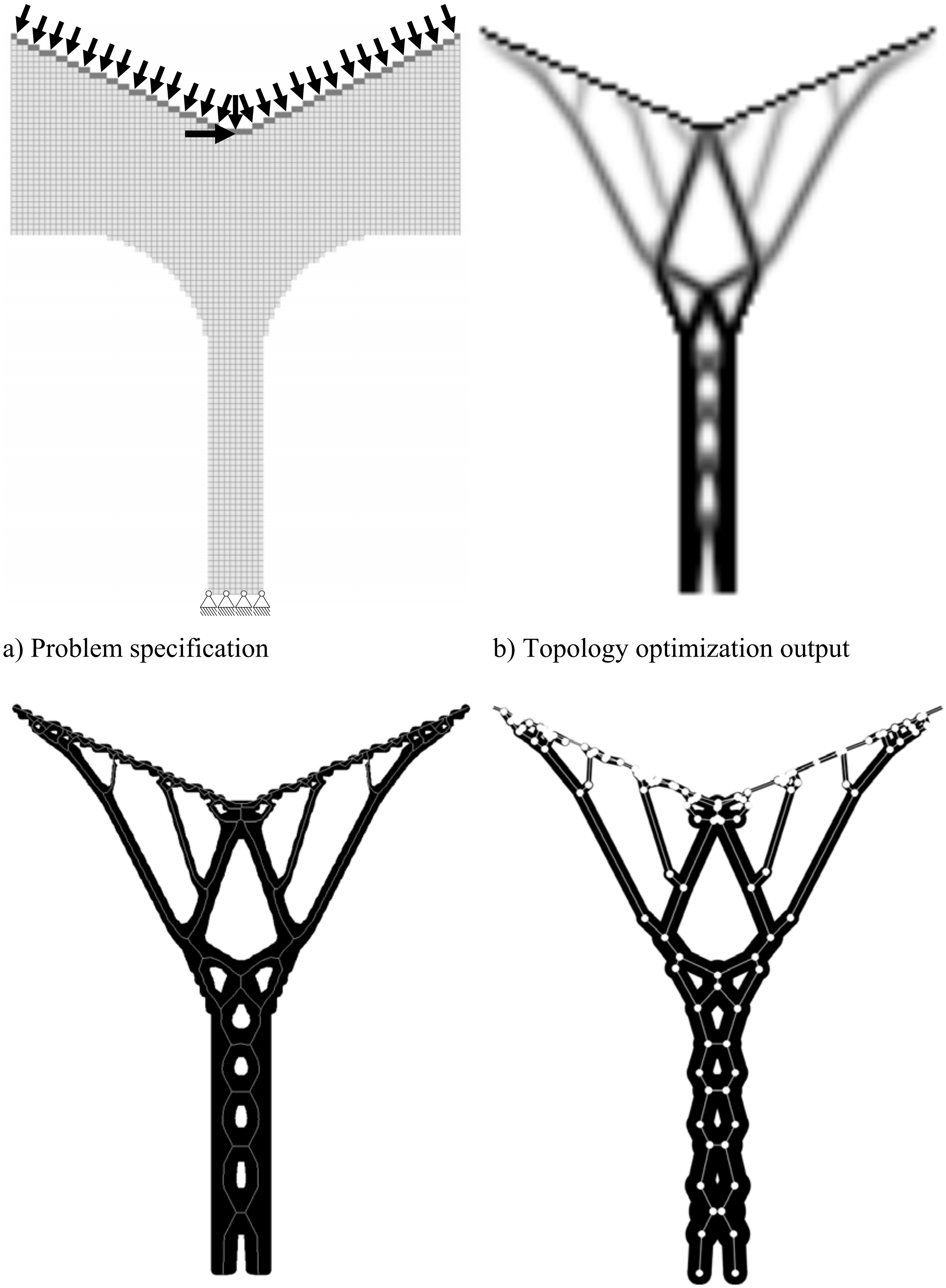
a) Problem specification

b) Topology optimization output

c) Skeleton

d) Element model

Figure 4.40. Roof load case 2



a) Problem specification

b) Topology optimization output

c) Skeleton

d) Element model

Figure 4.41. Roof load case 3

4.6. Inference

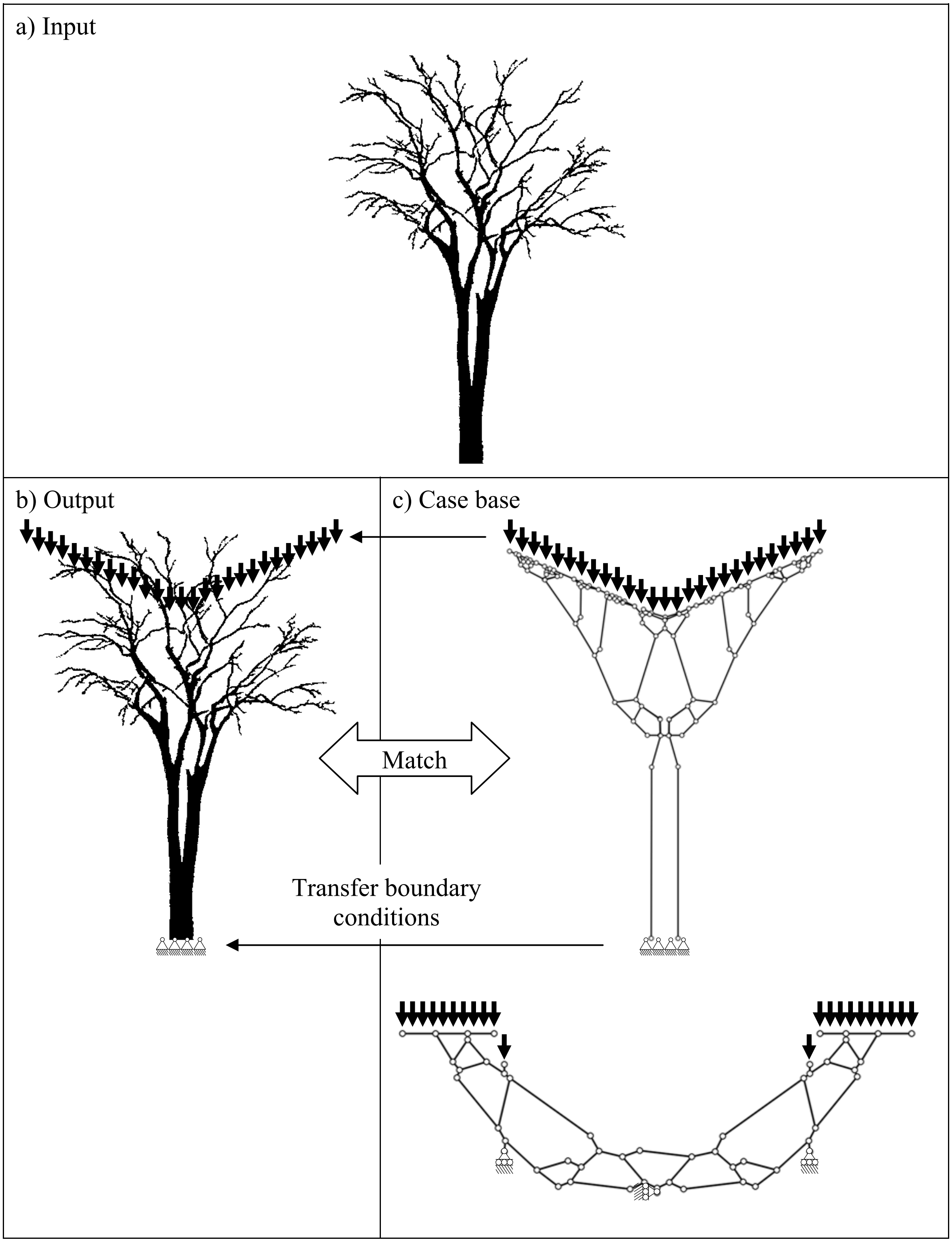


Figure 4.42. Visual inference example

To illustrate the potential application of visual case-based reasoning and visual inference, a simple example is presented in Figure 4.42. A sample image of a tree (Figure 4.42.a.) is provided as the input case, or target case. The goal is to retrieve a case from a design case base that best matches the input case. Since two structures similar in appearance may share similar design requirements, it may be possible to apply knowledge from the stored case to the input case to assist in the conceptual design of the input case. A shown in Figure 4.42.c., there are two source cases in the case base. The first case, the tree-like roof structure from the previous case study, is more similar to the input case than the second case, from the telescope example. Similarity is determined using the Object Density Map, graph matching, or other methods. The most similar cases are retrieved from the case base for the user to review and determine whether the cases are in fact similar enough to assist in the design of the input case. Stored cases can include information on loading and support conditions. As part of the retrieval process, boundary conditions for the source cases may overlaid onto the input case to determine if corresponding structure exists in the input case. Based on user judgment and the similarity of structure at boundary condition locations, boundary conditions may be transferred from the source case to the input case (Figure 4.42.b.). In this way, an arbitrary image may be interpreted as a structure, and knowledge may be gained about that image by inference from similar structures. This feature represents a powerful technique for learning about structures, and efficiently leveraging information from a database of stored experience for use in conceptual design.

Sketch input is processed in the same way as the image in Figure 4.42. This capability supports the use of natural user interfaces, including pen-based input. Using visual case-based reasoning, stored design cases similar to the sketch may be retrieved and used as the basis for conceptual design. Also, probable boundary conditions and design requirements can be inferred from target cases to assist in refining the design domain for a new conceptual design case.

5. Computer Application

This chapter describes a proof-of-concept computer application developed as part of this research. An overview of the principles guiding the development of the application is given first, followed by a description of system components and usage.

5.1. Design Principles

In order to focus research efforts on the concepts behind the framework, existing software and algorithms were used where possible. The intent in developing the application was not to produce a commercial program nor to develop a sophisticated user interface, but simply to verify proposed concepts.

5.2. Components

The main module of the application, including the user interface, was developed in Microsoft Visual C++ 2005 Express Edition, available as a free download. The main module implements image processing functions, aided by an open source computer vision library, OpenCV. The main module also calls MATLAB for image processing, linear analysis, SQP optimization, and topology optimization.

5.2.1. Topology Optimization

Topology optimization was performed in MATLAB, with an algorithm developed by [Sigmund, 2001]. The algorithm uses an optimality criteria method, and includes a mesh independency filter.

5.2.2. Image Processing

Image processing functions from the OpenCV computer vision library were used. OpenCV is open source software, originally developed by Intel, and now released under the terms of the BSD. The functions used from OpenCV were limited to image resizing and scaling, thresholding, erosion and dilation, distance transformation, pixel access, and file access. MATLAB was used for additional image processing.

5.2.3. Geometric Optimization

The Sequential Quadratic Programming algorithm from the MATLAB Optimization Toolkit was used for geometric and stability optimization. Linear structural analysis in MATLAB was done using the G2 Matrix Structural Analysis software [Fenves, 1999].

5.2.4. Stability Optimization

The author implemented a G2 element with variable end releases for stability optimization.

5.2.5. Pattern Recognition

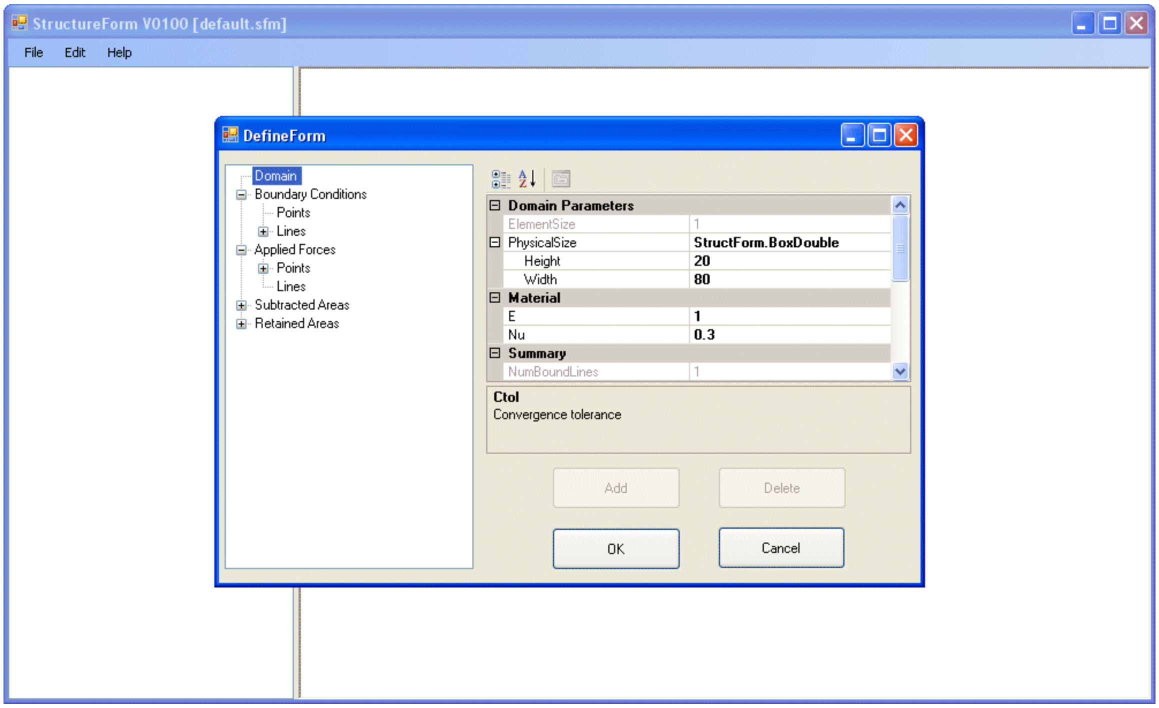
The author implemented image projections, the Object Density Map, and the Hough transform method for structural similarity in MATLAB.

5.3. Usage

5.3.1. Synthesis Phase

The Synthesis Phase is initiated by defining the design domain. Figure 5.1 shows the domain definition dialog box, which contains fields for the overall domain dimensions, the subtracted and retained areas within the domain, and the boundary conditions. The dialog box also contains the parameters controlling topology synthesis, such as the volume fraction and convergence tolerance for topology optimization. For a simple example as shown in Figure 5.1, the minimum input is the length and width of the domain rectangle, a support boundary condition, and an applied load case. After the domain parameters are entered in the dialog box, an image of the design domain is displayed, as shown in Figure 5.2.

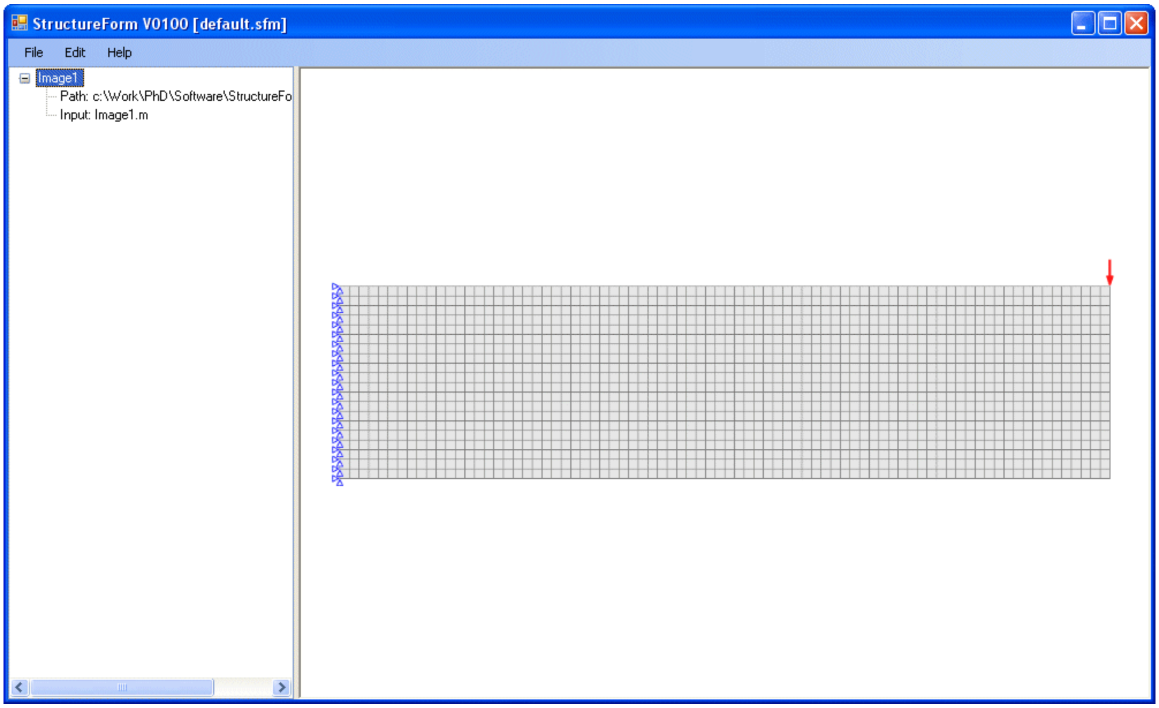
More complex design domains may be specified, as shown in Figure 5.3, which contains a number of applied loads, support boundary conditions, and subtracted areas. This figure shows the domain definition for the Telescope example in Chapter 4. The resulting domain figure is displayed in Figure 5.4.



Domain Parameters	
ElementSize	1
PhysicalSize	**StructForm.BoxDouble**
Height	**20**
Width	**80**
Material	
E	**1**
Nu	**0.3**
Summary	
NumBoundLines	1

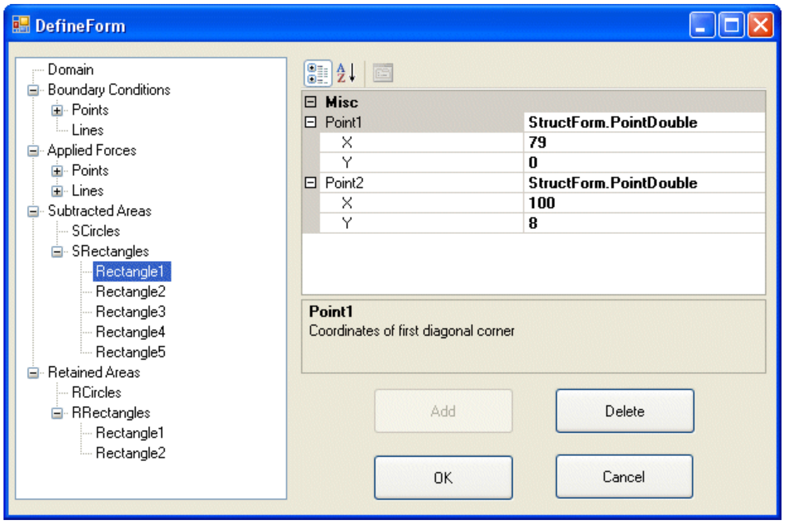
Ctol
Convergence tolerance

Figure 5.1. Domain definition form



Path: c:\Work\PhD\Software\StructureFo
Input: Image1.m

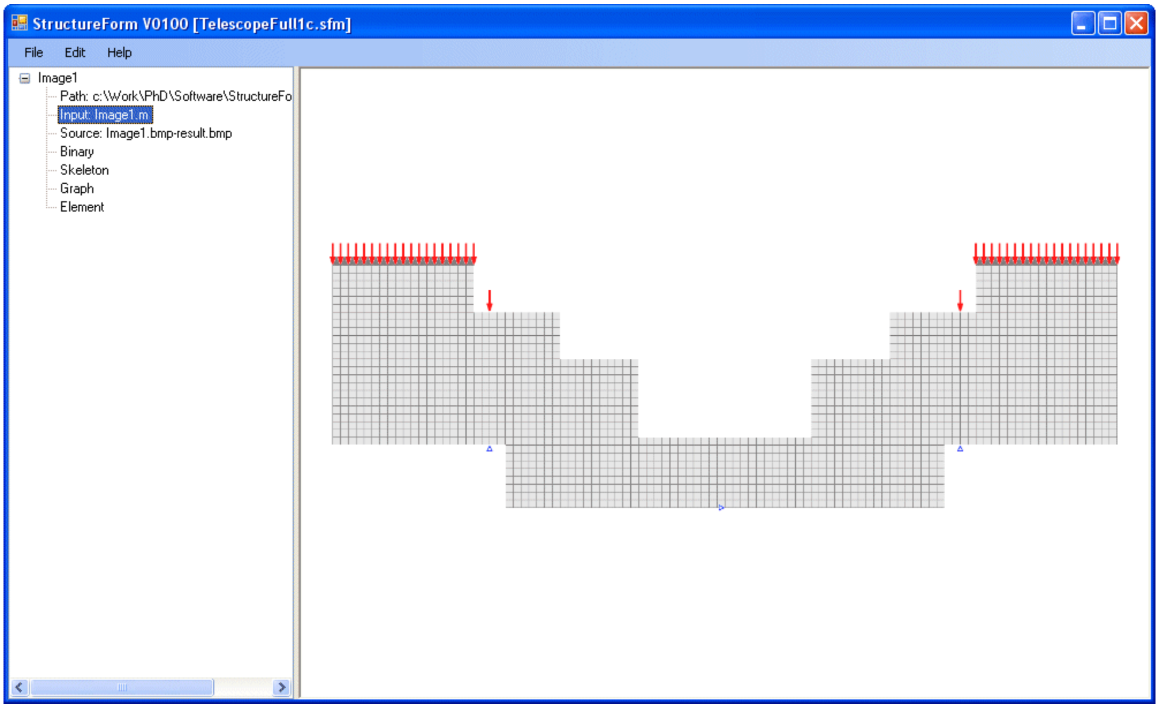
Figure 5.2. Domain image



Misc	
Point1	**StructForm.PointDouble**
X	**79**
Y	**0**
Point2	**StructForm.PointDouble**
X	**100**
Y	**8**

Point1
Coordinates of first diagonal corner

Figure 5.3. Domain definition for Telescope example



Path: c:\Work\PhD\Software\StructureFo
Input: Image1.m
Source: Image1.bmp-result.bmp

Figure 5.4. Domain image for Telescope example

Topology optimization is initiated using the menu bar of the main application, as shown in Figure 5.5.

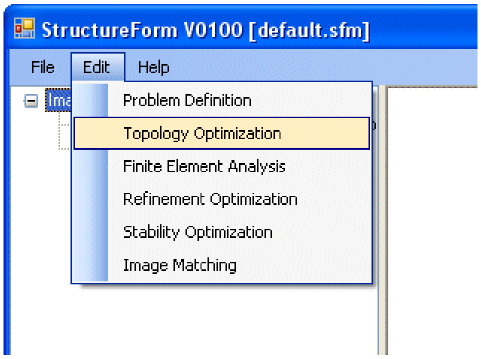


Figure 5.5. Running topology optimization

Topology optimization is executed interactively, and the user can view the emergence of structural form frame-by-frame. Some sample frame captures for a typical example are given in Figure 5.6. The frames are displayed in a MATLAB window, and are produced by subroutines developed by [Sigmund, 2001]. When the topology optimization routine converges, the results are displayed in the main application window (Figure 5.7). Image processing routines are applied automatically to the topology optimization output. The output is converted to binary format (Figure 5.8), the skeleton is calculated, the skeleton paths are traced and a vector skeleton is generated (Figure 5.9). Member widths are calculated, and the resulting element model is created (Figure 5.10). These different views or representations of the structure are accessible through a tree-like index, called the *navigation window*. Views are displayed in the *image window*, so different views are easily compared by scrolling vertically through the tree.

The main application supports multiple design configurations, as illustrated in Figure 5.11. In the navigation window, the top level items represent the design configuration, and the lower level items access the different views of the design configuration. Figure 5.11 shows a navigation window with two configurations, with the binary image of the second design configuration displayed in the image window.

Once the element model has been created, a finite element analysis may be performed. The loads and boundary conditions are transferred from the domain definition to the element model, and a finite element analysis is run. The resulting deflected shape is then included as a view in the navigation window, as shown in Figure 5.12.

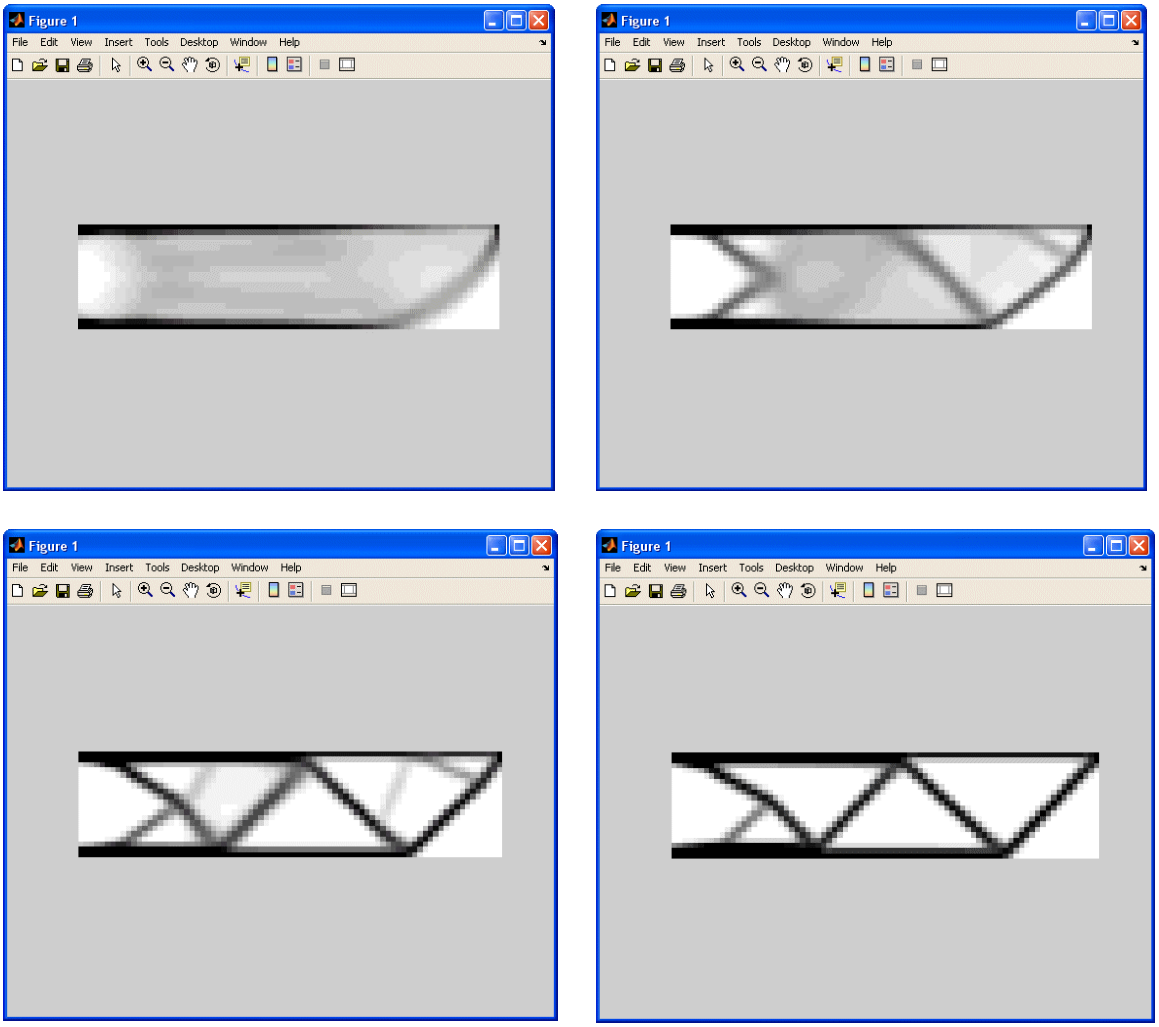


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help

Figure 5.6. Topology optimization frames

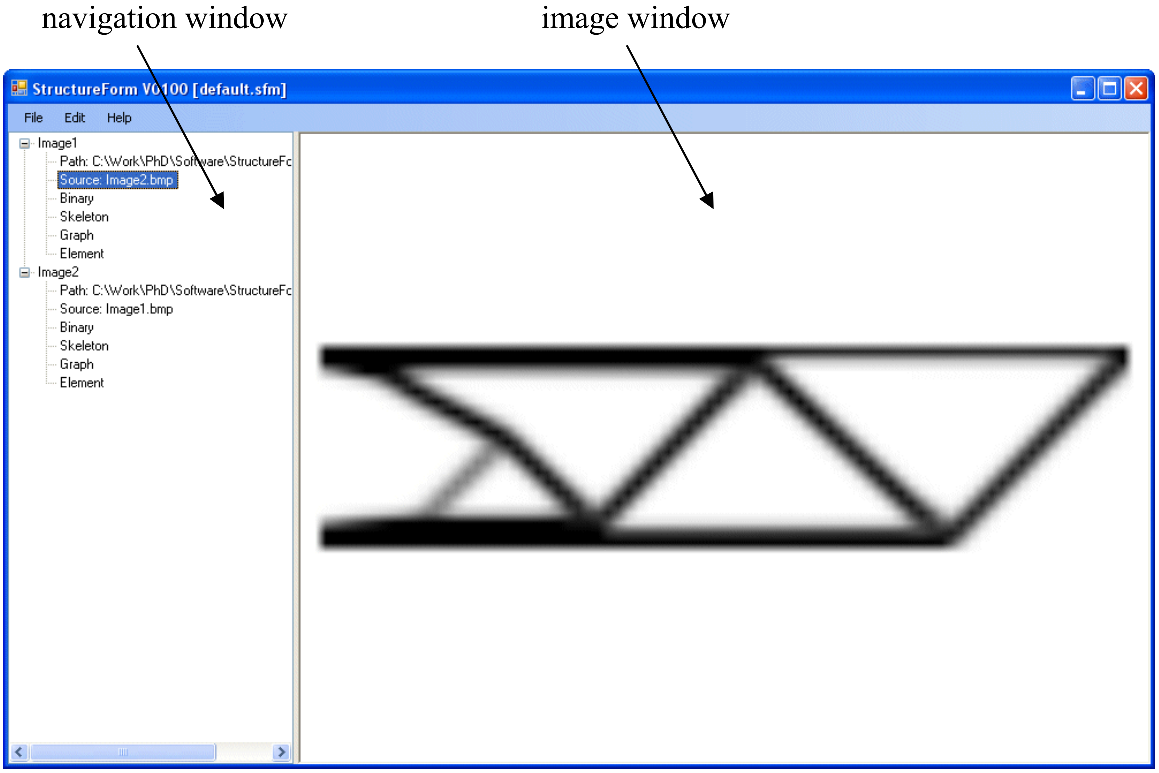


Figure 5.7. Topology optimization output

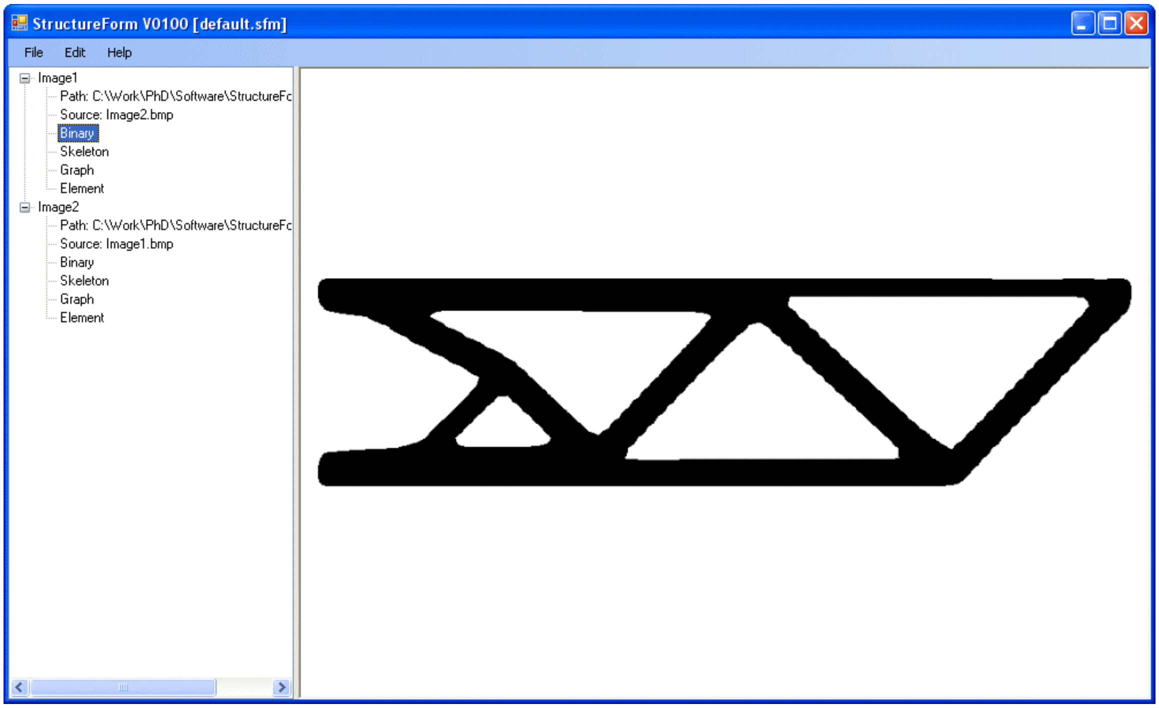
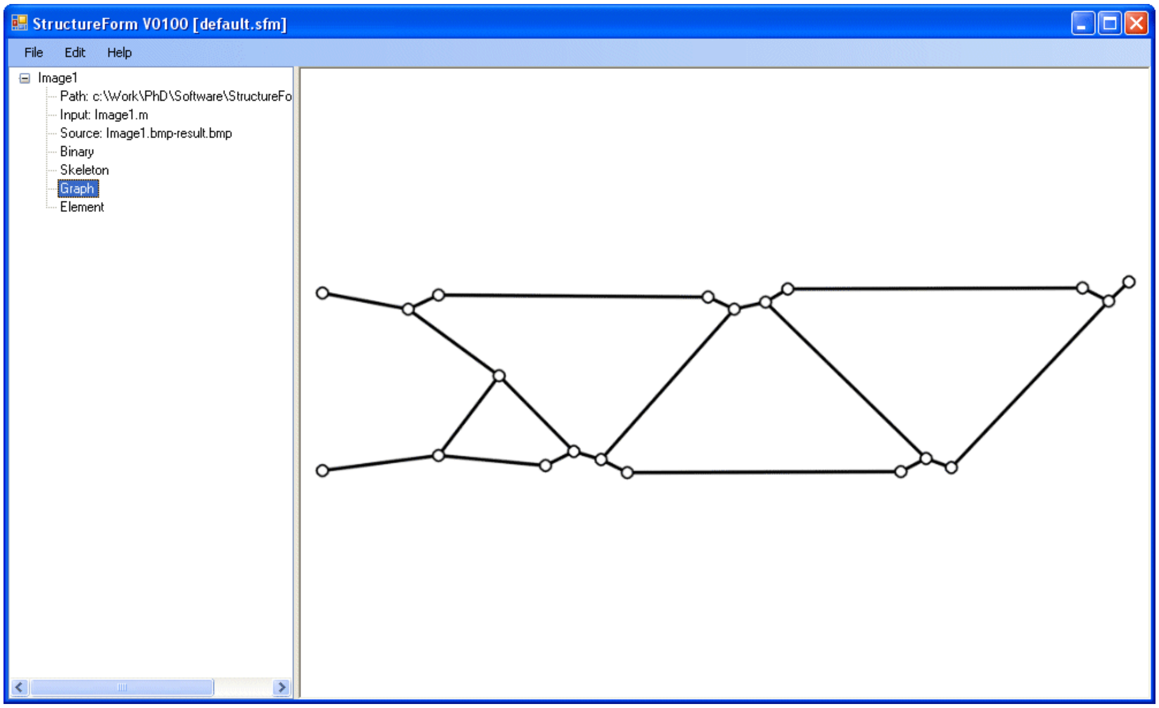
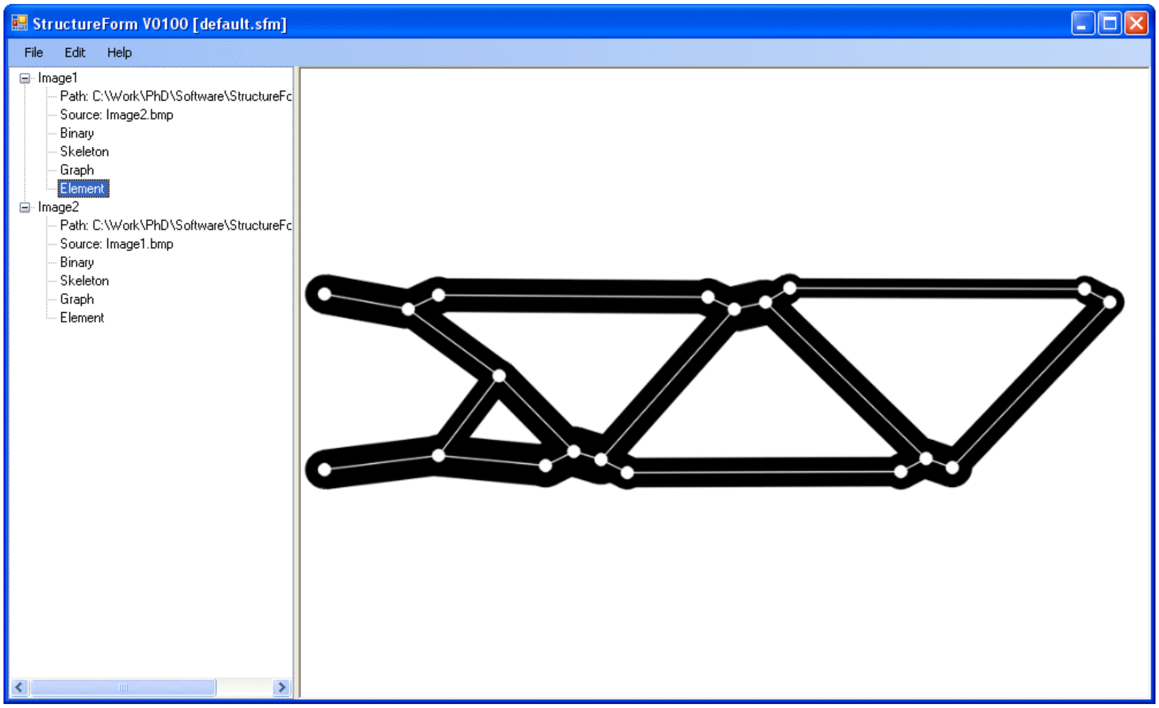


Figure 5.8. Binary image of topology optimization output



```
Image1
  Path: c:\Work\PhD\Software\StructureFo
  Input: Image1.m
  Source: Image1.bmp-result.bmp
  Binary
  Skeleton
  Graph
  Element
```

Figure 5.9. Vector skeleton view



```
Image1
  Path: C:\Work\PhD\Software\StructureFc
  Source: Image2.bmp
  Binary
  Skeleton
  Graph
  Element
Image2
  Path: C:\Work\PhD\Software\StructureFc
  Source: Image1.bmp
  Binary
  Skeleton
  Graph
  Element
```

Figure 5.10. Element view

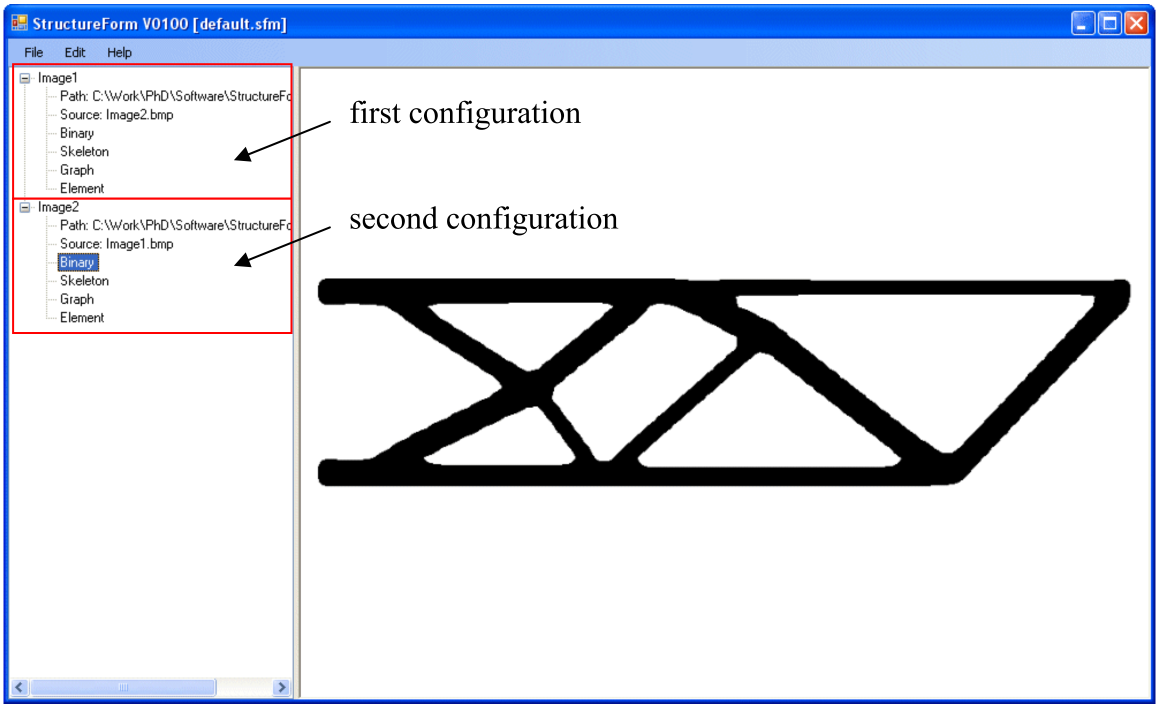


Image1
- Path: C:\Work\PhD\Software\StructureFo
- Source: Image2.bmp
- Binary
- Skeleton
- Graph
- Element

Image2
- Path: C:\Work\PhD\Software\StructureFo
- Source: Image1.bmp
- Binary
- Skeleton
- Graph
- Element

first configuration

second configuration

Figure 5.11. Multiple design views

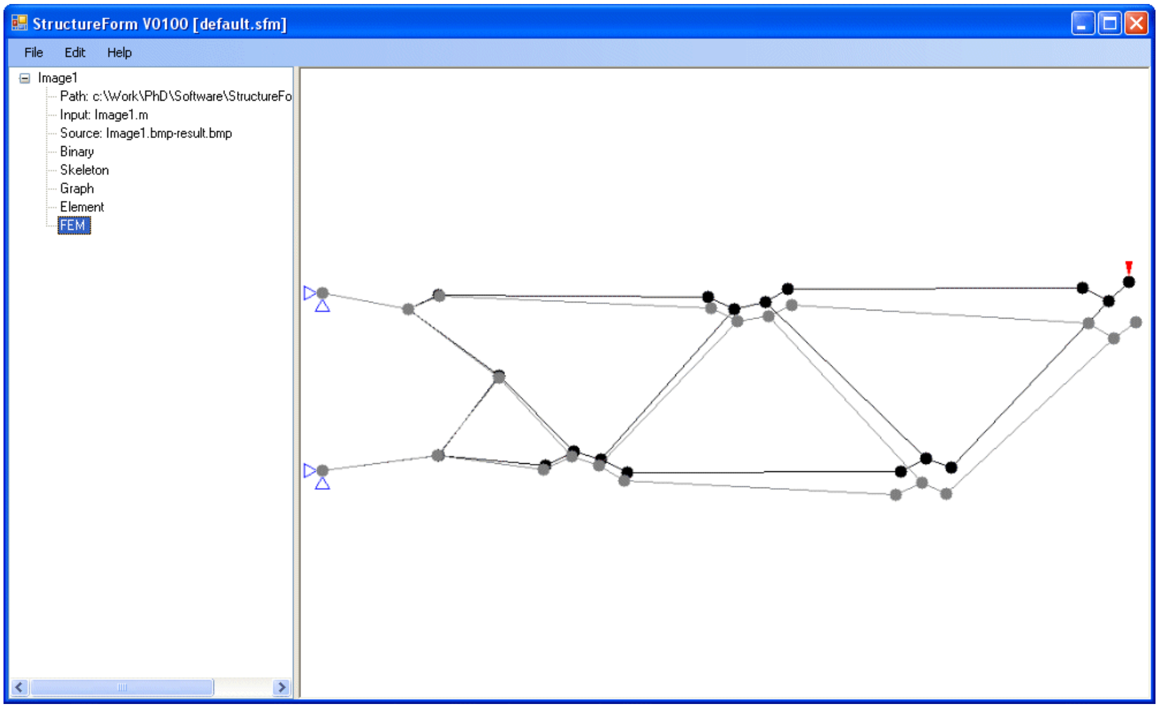
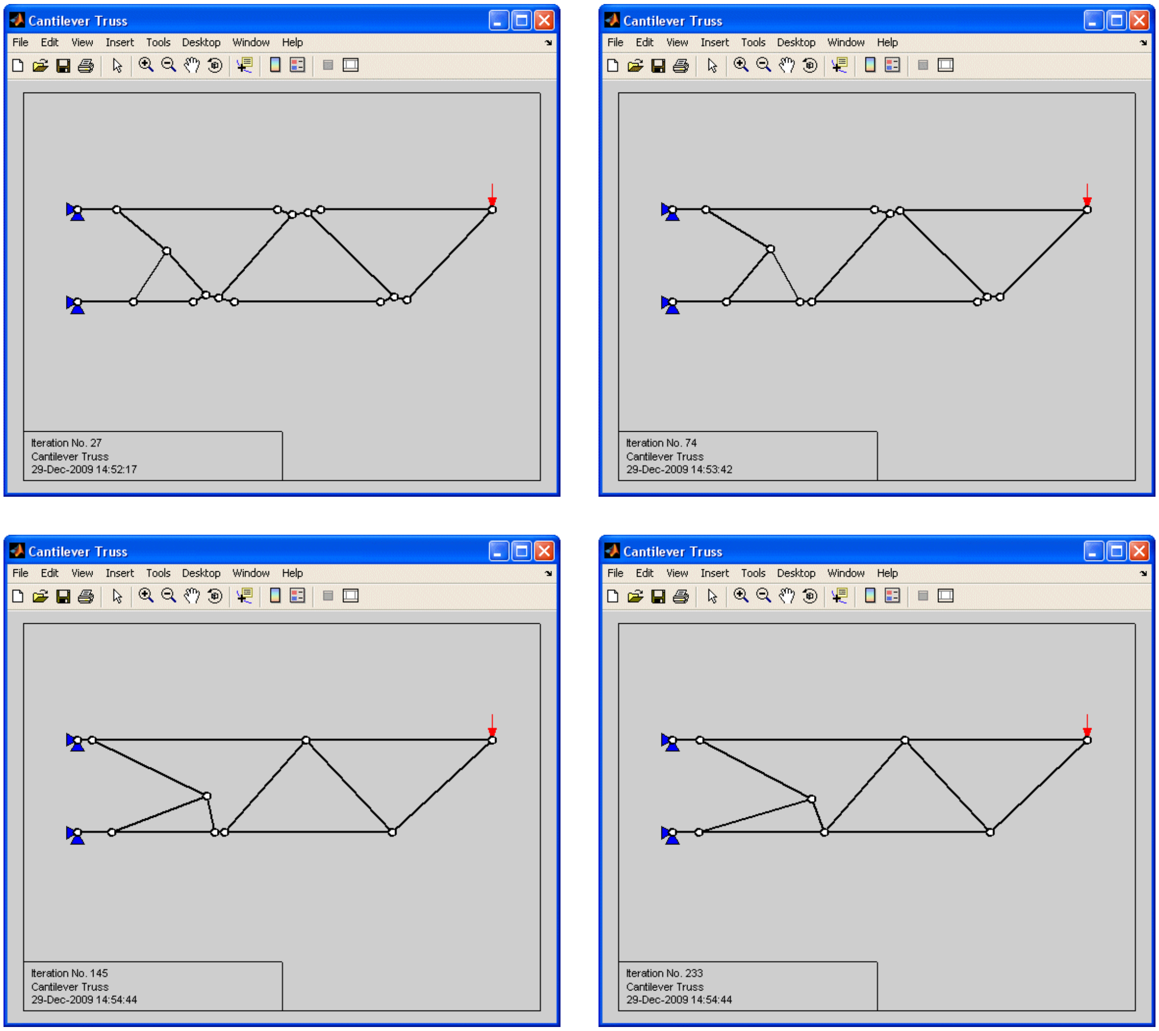


Image1
- Path: c:\Work\PhD\Software\StructureFo
- Input: Image1.m
- Source: Image1.bmp-result.bmp
- Binary
- Skeleton
- Graph
- Element
- FEM

Figure 5.12. Linear analysis results

5.3.2. Refinement Phase

As a first step in the Refinement Phase, geometric optimization is performed on the finite element model. Geometric optimization is performed in MATLAB using Sequential Quadratic Programming routines from the MATLAB Optimization Toolbox. An image of the structure is updated with each optimization iteration, allowing the user to interactively view the changes occurring in the structure. Node repositioning, node deletion and node merging are all visible as they occur. Some sample frames from geometric optimization are shown in Figure 5.13.



Iteration No. 27
Cantilever Truss
29-Dec-2009 14:52:17

Iteration No. 74
Cantilever Truss
29-Dec-2009 14:53:42

Iteration No. 145
Cantilever Truss
29-Dec-2009 14:54:44

Iteration No. 233
Cantilever Truss
29-Dec-2009 14:54:44

Figure 5.13. Geometric optimization frames

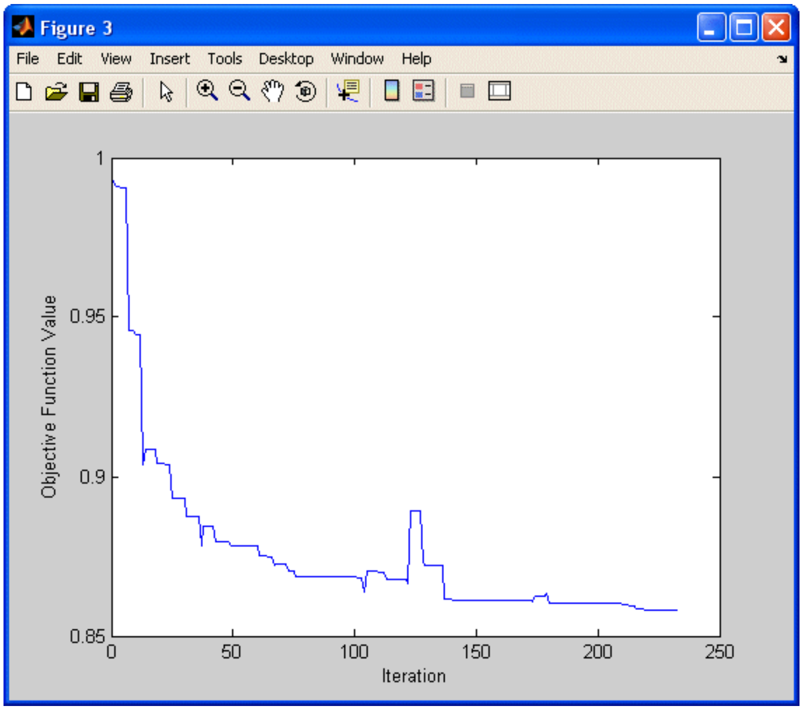
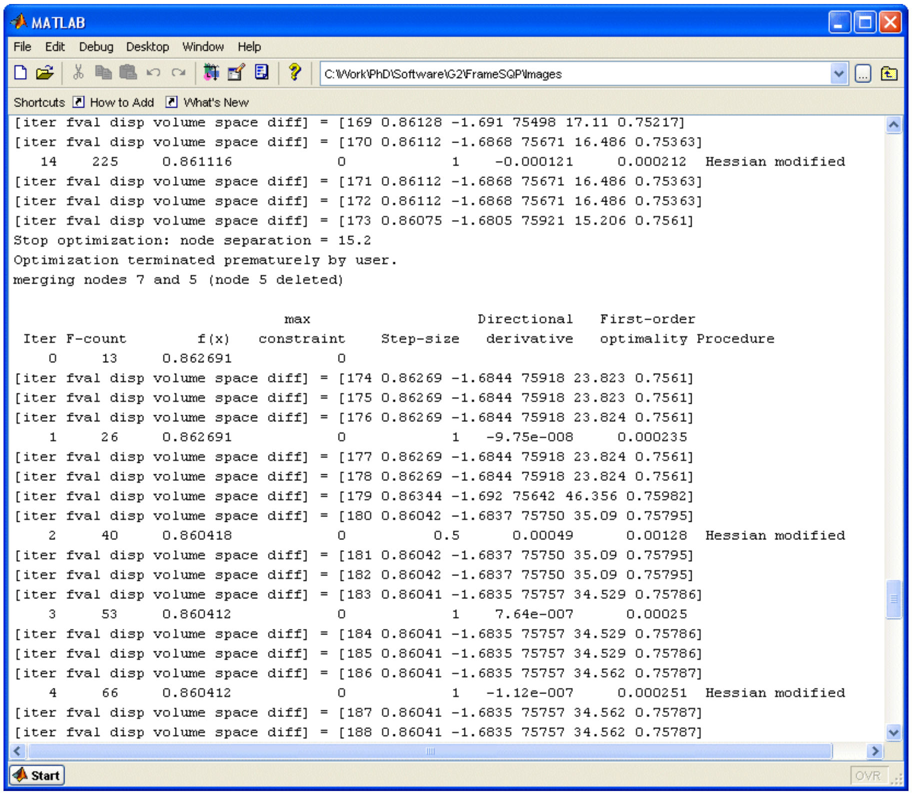


Figure 5.14. Geometric optimization objective function history


MATLAB
File Edit Debug Desktop Window Help
C:\Work\PhD\Software\G2\FrameSQP\Images
Shortcuts How to Add What's New
Start
OVR

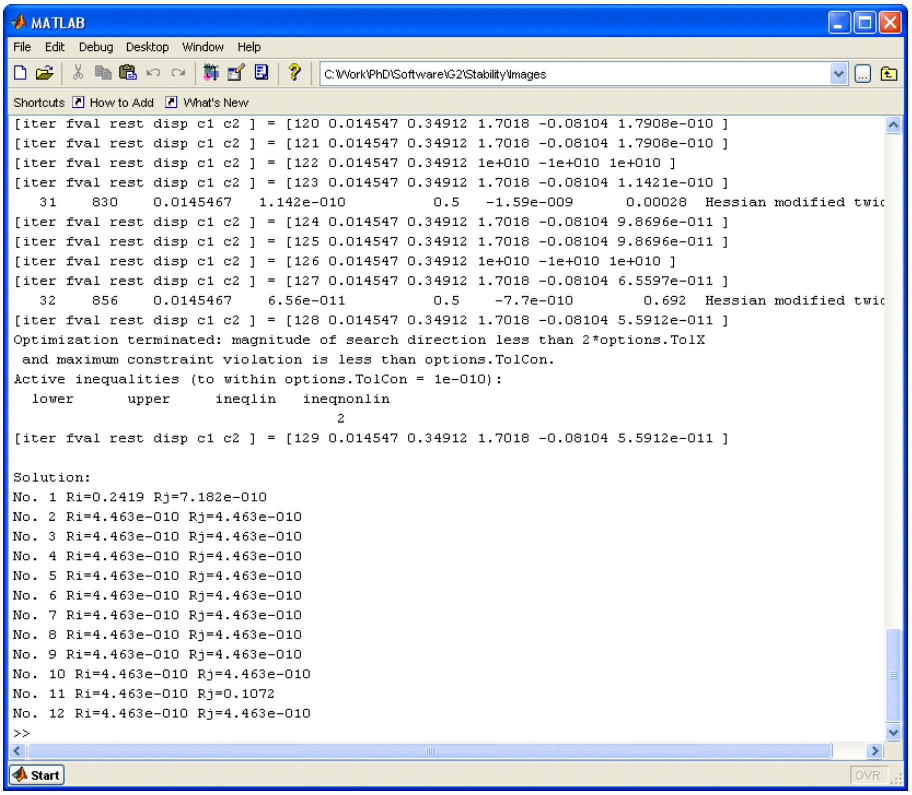
```
[iter fval disp volume space diff] = [169 0.86128 -1.691 75498 17.11 0.75217]
[iter fval disp volume space diff] = [170 0.86112 -1.6868 75671 16.486 0.75363]
   14    225      0.861116            0            1    -0.000121     0.000212  Hessian modified
[iter fval disp volume space diff] = [171 0.86112 -1.6868 75671 16.486 0.75363]
[iter fval disp volume space diff] = [172 0.86112 -1.6868 75671 16.486 0.75363]
[iter fval disp volume space diff] = [173 0.86075 -1.6805 75921 15.206 0.7561]
Stop optimization: node separation = 15.2
Optimization terminated prematurely by user.
merging nodes 7 and 5 (node 5 deleted)

                                 max                      Directional   First-order
 Iter F-count        f(x)    constraint    Step-size   derivative   optimality Procedure
    0      13      0.862691            0
[iter fval disp volume space diff] = [174 0.86269 -1.6844 75918 23.823 0.7561]
[iter fval disp volume space diff] = [175 0.86269 -1.6844 75918 23.823 0.7561]
[iter fval disp volume space diff] = [176 0.86269 -1.6844 75918 23.824 0.7561]
    1      26      0.862691            0            1   -9.75e-008     0.000235
[iter fval disp volume space diff] = [177 0.86269 -1.6844 75918 23.824 0.7561]
[iter fval disp volume space diff] = [178 0.86269 -1.6844 75918 23.824 0.7561]
[iter fval disp volume space diff] = [179 0.86344 -1.692 75642 46.356 0.75982]
[iter fval disp volume space diff] = [180 0.86042 -1.6837 75750 35.09 0.75795]
    2      40      0.860418            0          0.5      0.00049      0.00128  Hessian modified
[iter fval disp volume space diff] = [181 0.86042 -1.6837 75750 35.09 0.75795]
[iter fval disp volume space diff] = [182 0.86042 -1.6837 75750 35.09 0.75795]
[iter fval disp volume space diff] = [183 0.86041 -1.6835 75757 34.529 0.75786]
    3      53      0.860412            0            1    7.64e-007      0.00025
[iter fval disp volume space diff] = [184 0.86041 -1.6835 75757 34.529 0.75786]
[iter fval disp volume space diff] = [185 0.86041 -1.6835 75757 34.529 0.75786]
[iter fval disp volume space diff] = [186 0.86041 -1.6835 75757 34.562 0.75787]
    4      66      0.860412            0            1   -1.12e-007     0.000251  Hessian modified
[iter fval disp volume space diff] = [187 0.86041 -1.6835 75757 34.562 0.75787]
[iter fval disp volume space diff] = [188 0.86041 -1.6835 75757 34.562 0.75787]
```

Figure 5.15. Geometric optimization status

At the completion of geometric optimization, the objective function value is plotted for each iteration (Figure 5.14). Similarly, other key optimization results, such as deflection and volume are plotted. The MATLAB window contains a log of the optimization process, including information on node merging and deletions (Figure 5.15).

Stability optimization results are displayed in the MATLAB window, as shown in Figure 5.16. The results are summarized in a listing containing the release values r_i at the ends of each member (shown as `Ri` and `Rj` in Figure 5.16).


MATLAB
File Edit Debug Desktop Window Help
C:\Work\PhD\Software\G2\Stability\Images
Shortcuts How to Add What's New
Start
OVR

```
[iter fval rest disp c1 c2 ] = [120 0.014547 0.34912 1.7018 -0.08104 1.7908e-010 ]
[iter fval rest disp c1 c2 ] = [121 0.014547 0.34912 1.7018 -0.08104 1.7908e-010 ]
[iter fval rest disp c1 c2 ] = [122 0.014547 0.34912 1e+010 -1e+010 1e+010 ]
[iter fval rest disp c1 c2 ] = [123 0.014547 0.34912 1.7018 -0.08104 1.1421e-010 ]
   31     830     0.0145467    1.142e-010           0.5    -1.59e-009      0.00028  Hessian modified twic
[iter fval rest disp c1 c2 ] = [124 0.014547 0.34912 1.7018 -0.08104 9.8696e-011 ]
[iter fval rest disp c1 c2 ] = [125 0.014547 0.34912 1.7018 -0.08104 9.8696e-011 ]
[iter fval rest disp c1 c2 ] = [126 0.014547 0.34912 1e+010 -1e+010 1e+010 ]
[iter fval rest disp c1 c2 ] = [127 0.014547 0.34912 1.7018 -0.08104 6.5597e-011 ]
   32     856     0.0145467     6.56e-011           0.5     -7.7e-010        0.692  Hessian modified twic
[iter fval rest disp c1 c2 ] = [128 0.014547 0.34912 1.7018 -0.08104 5.5912e-011 ]
Optimization terminated: magnitude of search direction less than 2*options.TolX
 and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-010):
  lower      upper     ineqlin   ineqnonlin
                                      2
[iter fval rest disp c1 c2 ] = [129 0.014547 0.34912 1.7018 -0.08104 5.5912e-011 ]

Solution:
No. 1 Ri=0.2419 Rj=7.182e-010
No. 2 Ri=4.463e-010 Rj=4.463e-010
No. 3 Ri=4.463e-010 Rj=4.463e-010
No. 4 Ri=4.463e-010 Rj=4.463e-010
No. 5 Ri=4.463e-010 Rj=4.463e-010
No. 6 Ri=4.463e-010 Rj=4.463e-010
No. 7 Ri=4.463e-010 Rj=4.463e-010
No. 8 Ri=4.463e-010 Rj=4.463e-010
No. 9 Ri=4.463e-010 Rj=4.463e-010
No. 10 Ri=4.463e-010 Rj=4.463e-010
No. 11 Ri=4.463e-010 Rj=0.1072
No. 12 Ri=4.463e-010 Rj=4.463e-010
>>
```

Figure 5.16. Stability optimization results

5.3.3. Case-based Reasoning Phase

A critical element in visual case-based reasoning is image matching, used for the retrieval of similar images from the case base. Image matching using the Object Density Map (ODM) is implemented in MATLAB, where the source and target images are represented as 2D arrays. The source and target images are processed similarly. An output image is created for each image by scaling the input images sufficiently to clearly show the superimposed sampling grid and ODM grayscale values. The pixels in each grid square are counted and the sum of the pixels is used to select the ODM grayscale values. The grayscale values are plotted as triangular-shaped overlapping intervals.

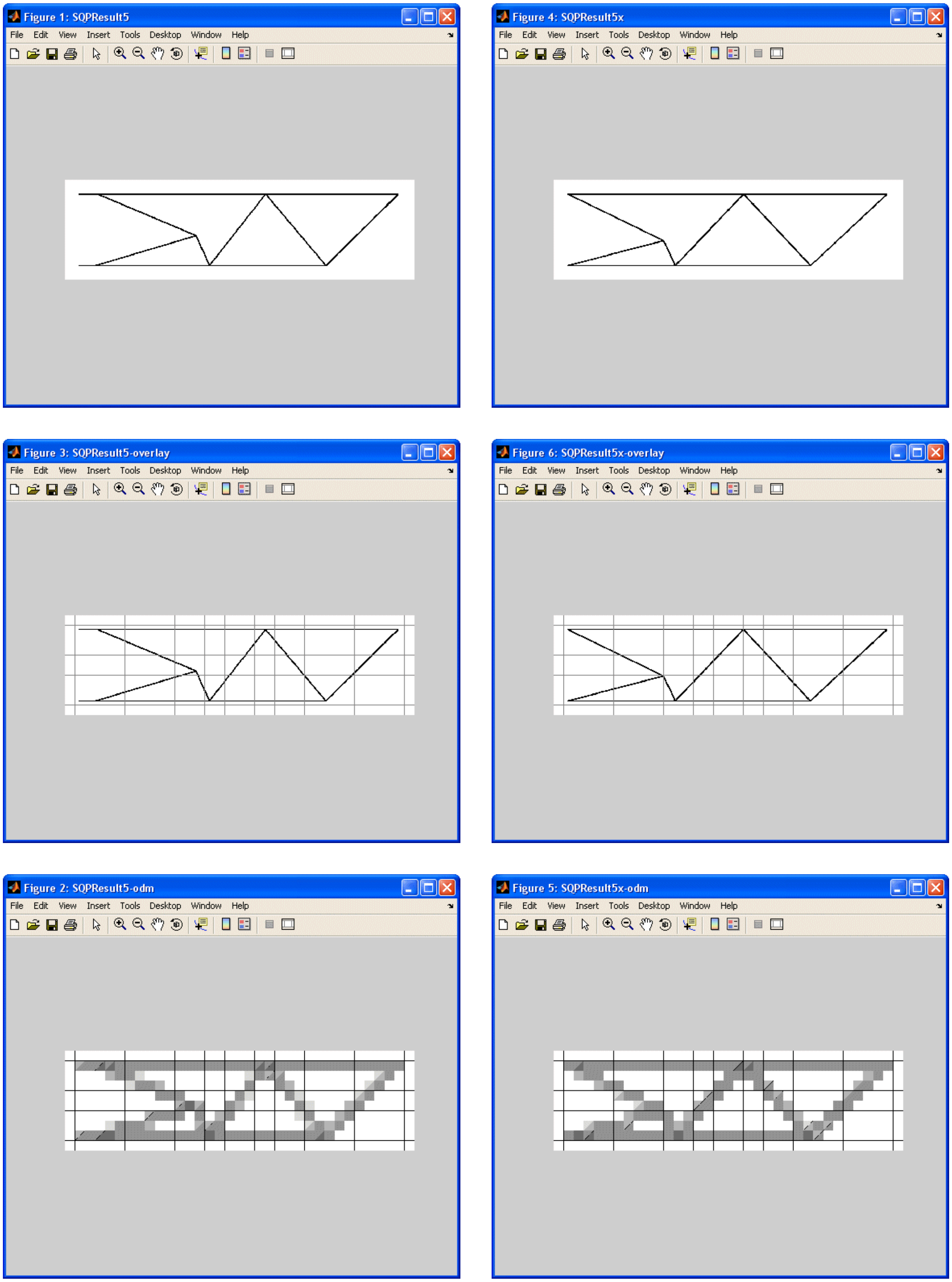
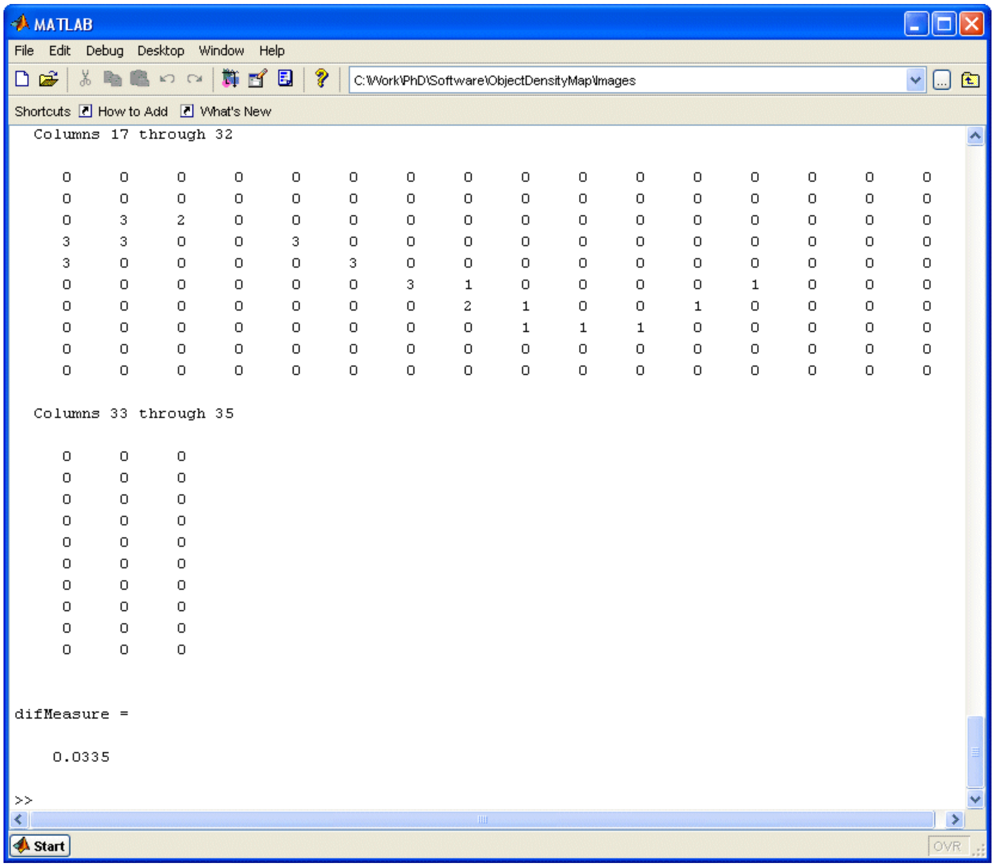


Figure 1: SQPResult5
Figure 4: SQPResult5x
Figure 3: SQPResult5-overlay
Figure 6: SQPResult5x-overlay
Figure 2: SQPResult5-odm
Figure 5: SQPResult5x-odm
File Edit View Insert Tools Desktop Window Help

Figure 5.17. Object density map

Image matching results are presented as shown in Figure 5.17. In this figure, the left and right columns of images correspond to the source and target images. The three images shown in sequence from top to bottom are 1) the input image, 2) the input image with superimposed sampling grid and 3) the ODM output. Further details of the image matching are provided in the MATLAB window (Figure 5.18), which displays a 2D array showing the numerical difference between corresponding grid square values. Finally, the MATLAB window displays the overall difference between images as a single, real-valued number (shown as `difMeasure` in Figure 5.18).


MATLAB
File Edit Debug Desktop Window Help
C:\Work\PhD\Software\ObjectDensityMap\Images
Shortcuts
How to Add
What's New
Start
OVR

```
  Columns 17 through 32

     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     3     2     0     0     0     0     0     0     0     0     0     0     0     0     0
     3     3     0     0     3     0     0     0     0     0     0     0     0     0     0     0
     3     0     0     0     0     3     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     3     1     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     2     1     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     1     1     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 33 through 35

     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0


difMeasure =

    0.0335

>>
```

Figure 5.18. Object density map results

6. Conclusions

This chapter summarizes the main features of the proposed framework, discusses general features and specific research contributions, comments on how well overall research objectives were satisfied, and gives recommendations for further research.

This dissertation describes a computational framework for assisting the structural designer during the conceptual design phase. A central idea in the framework is that the form of structures holds valuable information which can support visual reasoning at different levels of abstraction. The framework supports both the generation of new concepts as well as the use of past designs. In generation mode, a top-down process is first used to synthesize conceptual forms. Forms may be synthesized from a simple set of inputs representing a plain block of material, or from a complex set of geometric constraints representing an expressive shape. The form is then decomposed or deconstructed into a set of primitives and their relationships, a representation that supports high-level reasoning using techniques from linguistics and pattern recognition. The decomposition into primitives mirrors the fabrication process, where pieces are assembled to build a structure. The decomposition thus supports intelligent decision-making about potential detail design, fabrication and assembly options at the conceptual stage. For design reuse, the framework supports visual case-based reasoning as a means for retrieving similar designs. A novel feature of the framework is that arbitrary images of shapes may be interpreted as structures by using visual similarity to infer potential boundary conditions, functionality, and behaviour for those shapes.

6.1. Main Features

The main features of the work are summarized in the following paragraphs.

Visual reasoning

The use of visual reasoning methods is central to this research. The objective of conceptual structural design is to create forms. For human designers, this is a highly visual and symbolic process, particularly in the early stages of conceptual design. Formal methods for reasoning with images, shapes and patterns are effective in a system that provides computational support for conceptual design. Many existing conceptual design systems reason with concepts represented by textual information or a limited set of spatial relationships. The framework is specifically developed to leverage the information content of structural forms in all phases.

Automation
The framework has been developed to assist the designer, rather than generate design concepts in a fully automatic process. The most important creative force during conceptual design is the human designer. The framework mirrors the approach of experienced designers, who rely heavily on visual and symbolic information, and the store of past experience.

Synthesis
Conceptual design synthesis is a creative process. A key objective is to synthesize new designs that uniquely reflect the particular requirements of a given design problem. Sometimes the results are unexpected, which is a characteristic of all creative processes. The framework uses topology optimization methods to create new forms, sometimes from a minimal set of inputs. The framework also supports the ability to respond to an expressive and complex set of geometric constraints, which may arise in architectural design, for example. Through case-based reasoning, the new concepts can be refined with the benefit of experience from past design work.

Speed
Using a minimal amount of user input, the proposed methodology can be used to rapidly transform problem requirements into a concept-level finite element model. The model is developed using optimization techniques which simplify the geometry and guarantee stability. The model can be used directly in detailed, domain specific optimization procedures. Retrieval and reuse of past design information using case-based reasoning methods also reduces the time to develop design concepts.

Abstraction
Viewing concepts at multiple levels of abstraction is important during conceptual design. Conceptual design is characterized by incomplete and uncertain information, and detailed evaluation methods may be inefficient and ineffective. Similarity between concepts depends on the abstraction level at which they are viewed. Measures of similarity, which are essential to inferential learning, are more effective when they handle different abstraction levels. Multiple abstraction levels are featured throughout the framework. Images are represented at different levels of resolution. Symbolic representations are extracted from images through incremental changes in the abstraction level. Comparative finite element analysis techniques are used, as opposed to methods seeking absolute performance measures. Nondimensional or scale-invariant measures are used where possible to compare designs.

General optimization methods
The framework utilizes commercially-available mathematical optimization algorithms. This reduces the time needed to program specific optimization algorithms, which may be an important practical consideration. Optimization methods can readily be updated as more efficient ones become available. Mathematical optimization algorithms require a clear, concise, and general problem formulation, supporting independent verification and future research.

Multiobjective optimization
The framework directly supports the development of practical and cost-effective design concepts with multiobjective optimization. The framework emphasizes simplicity as a design objective, and decomposes the structure into components in a way that mirrors the fabrication process. Design, fabrication and assembly related constraints that apply to components and connections can readily be incorporated.

6.2. Contributions

This research draws heavily on the work of others in the fields of structural engineering, mechanical engineering, image processing and pattern recognition. Contributions to the conceptual design of structures are summarized here.

General topology optimization for large-scale structures
This framework proposes a method for efficiently generating discrete topology during conceptual structural design. The framework is particularly suited to the design and fabrication of large-scale skeletal structures. Such structures consist of an assemblage of discrete members, and connections between the members have a significant impact on the overall performance and cost of the structure. The framework describes a general method for generating topology for a wide range of potential applications using a minimal amount of input. The system is useful for a wide range of structural applications, including scientific instruments, industrial equipment supports, bridges and buildings.

A large number of computer applications for structural topology optimization are described in the literature. Continuum topology optimization methods are used for the design of plate, shell and solid structures. Truss and frame topology optimization methods are used for the design of skeletal structures. For truss topology optimization, one of the most general methods is the ground structure approach, which starts with a large number of potential design configurations. This method is known to be computationally demanding, particularly for a fully connected ground structure with fine node spacing. In practical applications, the ground structure is therefore relatively coarse, and must be carefully selected based on both domain- and problem-specific considerations. Truss topology optimization is complicated by the fact that even small changes in topology can lead to large differences in stiffness, limiting the use of classical gradient-based optimization techniques. Global optimization methods such as evolutionary computing have been found to be effective for discrete topology optimization. For design problems that are highly nonlinear or nonconvex, global optimization methods may be the only practical solution technique. Methods that use randomly generated ground structures to generate initial populations represent some of the most flexible and efficient evolutionary computing techniques for conceptual structural design. Such methods are particularly useful when the search space is complex or poorly-understood, and they can be used as a tool to explore and gain a better understanding of that space. If the search space contains structure that can be exploited by special-purpose search techniques, the use of evolutionary methods is generally computationally less efficient [De Jong, 1990] than those techniques. The framework proposed here is such a special-purpose technique that takes advantage of the underlying principles of engineering mechanics to efficiently generate design concepts.

Continuum topology optimization methods are efficient, well-established, and commercially available. The major drawback is that the output of such methods is not directly suitable for fabrication, particularly at scales larger than a few meters. Recent research has tended to focus on applications to microtechnology, nanotechnology, and material design. The framework presented in this research leverages the efficiency of continuum optimization methods, and extends their range of applicability to large scale structures. Although other research has been done in this area, the work proposed here represents a wider approach that integrates topology generation with visual case-based reasoning and visual inference methods. Also the this work proposes an efficient method for generating stable skeletal structures.

The conceptual design of buildings has been extensively covered in the literature, given the obvious economic implications. Computer systems that support conceptual building design generally use geometric reasoners with limited capabilities. The reasoners are restricted to the relatively simple spatial relationships found in economical buildings based on a rectilinear grid system with conventional column and beam construction. Although there are many potential applications for such systems, they are generally not suitable for the free-form, curvilinear construction that is common in modern architecture. This research proposes a system that is useful for a wide range of structural applications.

Automated generation of stable skeletal structures

Although methods for verifying structural stability are well established, efficient methods for generating stable structures are not. In truss topology optimization, stability is generally ensured using heuristics and generate-and-test methods. A common heuristic is to add sufficient members to ensure that all polygonal cells are triangular. Generate-and-test refers to the generation of a large number of different topological configurations, and filtering out the ones that are unstable. This work proposes an efficient method for generating stable skeletal structures using classical mathematical optimization methods. The stability optimization method presented here produces information that directly supports the detailed analysis and design of economical connections between members.

Visual case-based reasoning
Existing CBR systems for conceptual design use primarily textual attributes or simplified geometric attributes. Although sophisticated text-based case retrieval methods have been developed, these methods cannot fully represent the visual and graphical information that is an important part of conceptual structural design. The framework described here applies visual case-based reasoning techniques to conceptual structural design. Such methods have been applied to mechanical engineering and other fields, but there are apparently no existing applications to conceptual structural design.

Pattern recognition
Given the importance of visual and graphical information during conceptual design, it is remarkable that few computational tools for conceptual structural design exploit this information. In the fields of image processing and pattern recognition, there are well-established, rigourous techniques for manipulating graphical information. Such techniques have been applied for many years in areas such as medical imaging, remote sensing, and maufacturing, but few of these techniques have been applied to conceptual structural design. The framework presented here makes extensive use of these methods to generate conceptual designs and reason with those designs at relatively high levels of abstraction.

Structural similarity
This research introduces measures of similarity between structural design concepts using image processing methods. The framework represents images of structures at various levels of abstraction, which is key to identifying similarities between complex, dissimilar forms. The methods used include a Hough transform approach, discrete projections, and the Object Density Map.

Visual inference
Using structural similarity measures, the characteristics of one structural form can be inferred from those of another. In the framework, images containing arbitrary forms can be interpreted as structures. Similarity between the forms can be use to infer boundary conditions, design constraints and functionality from other structures.

Sophisticated graphical user interfaces have been developed in fields such as architecture and industrial design, where natural, pen-based applications simulate sketching, clay modeling and other creative form-finding methods. Unfortunately, such applications do not recognize content such as the meaning of elements and relationships between them. Symbolic sketch recognition computer programs have been developed in architecture and mechanical design. For mechanism design, there are diagrammatic reasoning systems that convert unlabeled line drawings into a description of a physical system. The framework described here is capable of providing high level descriptions of the functionality and performance of structures described by a sketch. No similar applications in the field of conceptual structural design are apparent in the literature.

Using methods similar to those used to process sketches, the framework has the capability to process photographs and other images of structures, and generate high level descriptions of the probable functionality and a performance of the structures represented in the images.

6.3. Objectives

The main objective of this research was to develop computational techniques to support the conceptual design of structures by enabling the rapid generation and evaluation of new designs, and by facilitating the reuse of past designs. The framework described here accomplishes that objective. To generate or synthesize new designs, the framework uses a combination of mathematical optimization, image processing and pattern recognition methods. The reuse of past designs is implemented using visual case-based reasoning methods. During synthesis, forms are initially created using topology optimization methods; these forms are processed to extract high level information that supports further structural optimization, including the assessment of stability and relative cost. The high level information is used to describe, classify and store conceptual forms for case-based reasoning. These techniques were implemented in a proof-of-concept computer application, and several examples were presented that illustrate the effectiveness of the proposed framework.

The specific objectives of this research have largely been accomplished, as discussed in the following paragraphs.

Apply visual case-based reasoning to conceptual structural design

A central feature of visual case-based reasoning is the retrieval of cases on the basis of the similarity of form. This research effectively implements a method for evaluating similarity between structural concepts. Reuse and modification of past design cases uses existing state-of-the-art case-based reasoning procedures.

Develop a framework for general conceptual structural design

Many existing conceptual design systems have been developed for buildings with simplified geometry based on rectilinear grids. The framework proposed here is useful for a wide range of structural applications, including scientific instruments, industrial equipment supports, and geometrically complex bridges and buildings.

Incorporate natural user input in conceptual structural design

The framework proposed here accepts natural input in the form of sketches. The framework uses sketches in two different modes. In the first mode, sketches are directly converted to finite element models which are subsequently processed by the framework to simplify form and ensure stability. In the second mode, sketches may be used to retrieve structural designs with similar form from a database of design cases. Simplification in the first mode produces a more abstract representation of the structure that supports a more effective search for similar designs in the second mode.

Apply visual inference to conceptual structural design
The research described here presents a method for determining visual similarity between two structural design concepts. Evaluating similarity between concepts is a key requirement for applying the process of inference.

Rapidly generate discrete topology in conceptual structural design
Case studies show that the framework presented here is capable of generating discrete structural topology more efficiently than some of the most efficient existing methods. The rapid generation of discrete topology is particularly important for the conceptual design of large structures with consideration to practical fabrication methods.

Extend continuum topology optimization to large scale structures
The framework described here uses efficient, well-established, and commercially available continuum topology optimization methods, which are currently more suited to relatively small-scale fabrication. The framework successfully applies these methods to large scale structures such as bridges, architectural structures, and large telescopes.

Establish an efficient method for developing a stable discrete structure
This work presents an efficient method for generating stable skeletal structures using classical mathematical optimization methods.

Generate and reason with conceptual designs at a high level of abstraction
The framework employs techniques of image processing and pattern recognition to manipulate graphical information at high levels of abstraction. The use of multiple abstraction levels is emphasized throughout the description of the framework.

Revisiting the main hypothesis, that “patterns exist that relate structural forms to design requirements”, this research has developed methods to process structural forms and identify patterns. The secondary hypothesis, that “a formal language exists to describe structural form,” is more ambitious, and verification requires further study using methods like the ones described in this work.

6.4. Recommendations for Further Work

The framework described here introduces several novel concepts in the conceptual design of structures. Further work is required to validate the overall framework in a practical, multi-user setting. A proof-of-concept software application which validates the major components of the framework, is also described here. Further work would be required to develop the proof-of-concept application into a commercial application. Other recommendations for further work are outlined in the following list.

Visual case base of structural designs
A practical structural design case base containing visual information would be useful in validating several aspects of the framework. A large database of design cases should be used to evaluate the practical effectiveness of case retrieval. Also, such a case base should be used to further test the inference of boundary conditions and other information from stored cases to arbitrary images, such as sketches and photographs.

Sensitivity Analysis
Although some effort was made in this work to evaluate the sensitivity of the derived forms to changes in input and control parameters, further sensitivity analysis is required. Sensitivity analysis may be used to improve control parameter settings, and to assist the user in efficiently exploring the input parameter space. Control parameters include the binary conversion threshold for topology optimization grayscale output, the volume fraction for topology optimization, and curve-fitting parameters for skeleton generation. Sensitivity to changes in input parameters such as boundary conditions and loads should be investigated further, with the possibility of developing qualitative and quantitative measures to describe the sensitivity of form to input values. Further research could be conducted into the sensitivity of performance to form. For example, is it possible to understand the general conditions under which large changes in topology produce relatively small changes in performance?

Automatic Control Parameters
Using the results of sensitivity analysis, schemes could be established for automatic adjustment of control parameters such as the binary conversion threshold. For example, the threshold could be automatically adjusted to reliably produce skeleton structures with the expected overall characteristics, such as connectivity.

Extension to Three Dimensions
Both topology optimization and pattern recognition methods are well established in 3D. Further work is required to extend the framework to accommodate 3D design, and to test the performance with a range of practical problems.

Human Interaction
Further work is required to understand how effective such a framework would be in supporting multiple human users during conceptual design. For example, architects, owners and engineers participate in the conceptual design of buildings, and each participant evaluates concepts using different quantitative and qualitative criteria. The interactive navigation, exploration, and selection of designs from a set of retrieved cases should be investigated further. A wide range of software and hardware tools have been developed to support the intuitive browsing and organization of visual information, and these may be used in conceptual structural design. Support for natural interfaces such as pen-based input could be expanded. For example, intelligent sketch recognition could be used to differentiate between structure, loads and support conditions in a schematic sketch of a structure.

7. Bibliography

Adeli, H., and Cheng, N.T., 1993. Integrated Genetic Algorithms for Optimisation of Space Structures. *Journal of Aerospace Engineering*, 6(4):315-328.

AISC, 1989. *Manual of Steel Construction: Allowable stress design*, 9th ed., American Institute of Steel Construction.

Alexander, C., 1964. *Notes on the synthesis of form*, Harvard University Press.

Arciszewski, T., Bloedorn, E., Michalski, R.S., Mustafa, M., Wnek, J., 1994. Machine learning of design rules: methodology and case study. *ASCE J. Comp. Civ. Engrg.* 8(2):286-309.

Azid, I.A. and Kwan, A.S.K., 1999. A layout optimisation technique with displacement constraint. In Topping and Kumar (eds.), *Optimization and Control in Civil and Structural Engineering*, Civil-Comp UK 1999, 71-77.

Azid, I.A., Kwan, A.S.K. and Seetharamm, K.N., 2002. An evolutionary approach for layout optimization of a three-dimensional truss. *Structural and Multidisciplinary Optimization*, 24(4): 333-337.

Bailey, S.F., and Smith, I.F., 1994. Case-based preliminary building design. *ASCE Journal of Computing in Civil Engineering*, 8(4):454-467.

Baldock, R., Shea, K., Eley, D., 2005. Evolving optimized braced steel frameworks for tall buildings using modified pattern search. *ASCE Conference on Computing in Civil Engineering*, Cancun, Mexico.

Baldock, R., and Shea, K., 2006. Structural Topology Optimization of Braced Steel Frameworks Using Genetic Programming. *European Group for Intelligent Computing in Engineering, 13th EG-ICE Workshop*, 25-30 June 2006 Monte Verità, Ascona, Switzerland.

Barrow, H.G., and Popplestone, R.J., 1971. Relational descriptions in picture processing. *Machine Intelligence*, 5:377-396.

Bibliography

Bendsøe, M.P., and Kikuchi, N. 1988. Generating Optimal Topologies in Structural Design Using a Homogenization Method. *Computer Methods in Applied Mechanics and Engineering*, 71:197-224.

Bendsøe, M.P. 1989. Optimal Shape Design as a Material Distribution Problem. *Structural Optimization*, 1:193-202.

Bendsøe, M.P., and Sigmund, O. 2004. *Topology Optimization. Theory, Methods and Applications*, 2nd Ed., Springer-Verlag, Berlin.

Bentley, P. and Wakefield, J., 1996. The Evolution of Solid Object Design Using Genetic Algorithms. *Modern Heuristic Search Methods*, 197-211.

Blake, R.E., 1994. Partitioning graph matching with constraints. *Pattern Recognition*, 27(3):439-446.

Bloch, I., 1999. On fuzzy distances and their use in image processing under imprecision. *Pattern Recognition*, 32(11):1873-1895.

Blum, H. 1967. A transformation for extracting new descriptions of shape. *Models for the Perception of Speech and Visual Form*, MIT Press, 362-380.

Blum, H. 1973. Biological shape and visual science. *Journal of Theoretical Biology*, 38:205-287.

Börner, K., Pippig, E., Tammer, E.-C., Coulon, C.-H., 1996. Structural Similarity and Adaptation. In *Proceedings, Advances in case-based reasoning: Third European Workshop, EWCBR-96*, Lausanne, Switzerland, Nov. 14-16, 1996, I. Smith, B. Faltings, eds.

Boothroyd, G., Dewhurst, P., Knight, W., 1994. *Product Design for Manufacture and Assembly*, Marcel Dekker.

Borgefors, G. 1986. Distance Transformations in Digital Images. *Computer Vision, Graphics and Image Processing*, 34:344-371.

Bremicker, M., Eschenauer, H.A., Post, P.U., 1990. Optimization Procedure SAPOP – A General Tool for Multicriteria Structural Designs. In *Multicriteria Design Optimization: Procedures and Applications*, H. Eschenauer, J. Koski, A. Osyczka, eds., Springer-Verlag, Berlin.

Bremicker, B., Chirehdast, M., Kikuchi, N., Papalambros, P.Y., 1991. Integrated Topology and Shape Optimization in Structural Design. *Mechanics of Structures and Machines*, 19(4):551-587.

Bunke, H., 1998. Error-tolerant graph matching: a formal framework and algorithms. In *Advances in Pattern Recognition*, A. Amin, D. Dori, P. Pudil and H. Freeman, eds., LNCS 1451, Springer Verlag, 1-14.

Calladine, R.B., 1978. Buckminster Fuller's "Tensegrity" structures and Clark Maxwell's rules for the construction of stiff frames. *International Journal of Solids and Structures*, 14:161-172.

Calladine, C.R., and Pellegrino, S., 1991. First-order infinitesimal mechanisms. *International Journal of Solids and Structures*, 27(4):505-515.

Censor, Y., 1977. Pareto Optimality in Multiobjective Problems. *Appl. Math. Optimiz.*, 4:41-59.

Cetin, O.L., and Saitou, K., 2004. Decomposition-Based Assembly Synthesis for Maximum Structural Strength and Modularity. *ASME Journal of Mechanical Design*, 126:244-253.

Chapman, C., Saitou, K. and Jakiela, M., 1993. Genetic algorithms as an approach to configuration and topology design. *Advances in Design Automation*, 65:485-498.

Chapman, C.D., Saitou, K., and Jakiela, M.J., 1994. Genetic algorithm as an approach to configuration and topology design. *Journal of Mechanical Design*, 116, 1005-1012.

Chickermane, H., and Gea, H.C., 1997. Design of Multi-component Structural Systems for Optimal Layout Topology and Joint Locations. *Engineering with Computers*, 13:235-243.

Chirehdast, M., and Papalambros, P., 1991. A note on automated detection of mobility of skeletal structures. *Tech. Report UM-MEAM-91-09*, Department of Mechanical Engineering and Applied Mechanics, The University of Michigan, Ann Arbor, MI.

Chirehdast, M., Linder, B., Yang, J., and Papalambros, P., 1992. Concurrent Engineering in Optimal Structural Design. In *Concurrent Engineering: Automation, Tools, and Techniques*, A. Kusiak, ed., John Wiley and Sons, New York.

Chirehdast, M., Gea, H.-C., Kikuchi, N., Papalambros, P.Y., 1994. Structural Configuration Examples of an Integrated Optimal Design Process. *ASME Journal of Mechanical Design*, 116(4):997-1004.

Coello, C.A.C., 2006. Evolutionary Multi-Objective Optimization: A Historical View of the Field. *IEEE Computational Intelligence Magazine*, February, pp. 28-36.

Connelly, R., and Whiteley, W., 1992. The stability of tensegrity frameworks. *International Journal of Space Structures* 7(2):153-163.

Cook, S.A., 1971. The complexity of theorem-proving procedures. In *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, 151-158.

Corneil, D.G., and Gotlieb, C.C., 1970. An efficient algorithm for graph isomorphism. *Journal of the Association for Computing Machinery*, 17:51-64.

Coulon, C.-H., and Steffens, R., 1994. Comparing fragments by their images. In *Similarity Concepts and Retrieval Methods*, A. Voβ, ed., Fabel-Reports, Volume 13, GMD, Sankt Augustin, 36-44.

Cross, A.D.J., Wilson, R.C., Hancock, E.R., 1997. Inexact graph matching using genetic search. *Pattern Recognition*, 30(6):953-970.

Da Cunha, N.O., and Polak, E., 1967. Constrained Minimization Under Vector-valued Criteria in Finite Dimensional Spaces. *J. Math. Anal. Appl.* 19:103-124.

Danielsson, P.-E. 1980. Euclidean Distance Mapping. *Computer Graphics and Image Processing* 14(3):227-248.

Dantzig, G.B., 1963. *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press.

Davies, T.H., 1968. An extension of Manolescu's classification of planar kinematic chains and mecanisms of mobility M>=1 using graph theory. *Journal of Mechanisms*, 3:87-100.

De Jong, K.A., 1990. Introduction to the second special issue on genetic algorithms. *Machine Learning*, 5(4):351-353.

De Kleer, J., 1977. Multiple representations of knowledge in a mechanics problem-solver. In *Proceedings, Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA.

De Kleer, J., 1979. The origin and resolution of ambiguities in causal arguments. In *Proceedings, Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, 197-203.

de Silva Garza, A.G., Maher, M.L., 1996. The Adaptation of Structural System Designs using Genetic Algorithms. *Proc. Int. Conf. Information Technology in Civil and Structural Engineering Design*, Elsevier Science, New York, 189-196.

Deng, H., Kwan, A.S.K., 2005. Unified classification of stability of pin-jointed bar assemblies. *International Journal of Solids and Structures*, 42 (15), 4393-4413.

di Ruberto, C., and Dempster, A., 2001. Attributed skeleton graphs using mathematical morphology. *Electronics Letters*, 37(22):1325-1327.

Domeshek, E.A., and Kolodner, J.L., 1991. Toward a case-based aid for conceptual design. *International Journal of Expert Systems*, 4(2):201-220.

Domeshek, E.A., and Kolodner, J.L., 1992. A Case-Based Design Aid for Architecture. In *Artificial Intelligence in Design '92*, J. Gero, ed., Kluwer Academic Publishers, Boston, 497-516.

Dorn, W.S., Gomory, R.E., Greenberg, H.J., 1964. Automatic design of optimal structures, *J. de Mecanique*, 3:25-52.

Dougherty, R.D. and Lotufo, R.A. 2003. *Hands-on Morphological Image Processing*, SPIE Press, Bellingham, WA, USA.

Duc, B., Fischer, S., Bigun, J., 1999. Face authentification with Gabor information on deformable graphs. *IEEE Transactions on Image Processing*, 8(4):504-516.

Duda, R.O., Hart, P.E., 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Comm. ACM*, 15:11–15.

Eisfeld, M., and Scherer, R., 2003. Assisting conceptual design of building structures by an interactive description logic based planner. *Advanced Engineering Informatics*, 17:41-57.

Eshera, M.A., and Fu, K.S., 1984. A graph distance measure for image analysis. *IEEE Trans. SMC*, 14:398-408.

Eshera, M.A. and Fu, K.S., 1986. An image understanding system using attributed symbolic representation and inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):604-617.

Falkenhainer, B., Forbus, K.D., Gentner, D., 1990. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1-63.

Fan, K., Lu, J., Chen, G., 1998. A feature point clustering approach to the recognition of form documents. *Pattern Recognition*, 31(9):1205-1220.

Farmer, S.-J., 1999. *Probabilistic graph matching*. Unpublished manuscript. University of York, UK.

Fenves, S.J., Flemming, U., Hendrickson, C., Maher, M.L., Quadrel, R., Terk, M., Woodbury, R., 1994. *Concurrent computer-integrated building design*, Prentice-Hall, Englewood Cliffs, NJ.

Fenves, S.J., Rivard, H., Gomez, N., Chiou, S.-C., 1995. Conceptual Structural Design in SEED. *Journal of Architectural Engineering*, 1(4):179-186.

Fenves, G.L., 1999. *G2 - Matrix Structural Analysis with Matlab*, Version 0.1 [software]. University of California, Berkeley, CA.

Fenves, S.J., Rivard, H., Gomez, N., 2000. SEED-Config: a tool for conceptual structural design in a collaborative building design environment, *Artificial Intelligence in Engineering*, 14:233-247.

Ferguson, R.W. and Forbus, K.D., 2000. GeoRep: A flexible tool for spatial representation of line drawings. In *Proceedings, AAAI-2000*, Austin, Texas, AAAI Press.

Firkins, A., and Hemphill, D., 1990. Fabrication cost of structural steelwork. *Steel Construction*, Australian Institute of Steel Construction, 24(2):2-14.

Flemming and Woodbury, 1995. Software Environment to Support Early Phases in Building Design (SEED): Overview. *Journal of Architectural Engineering*, 1(4):147-152.

Flemming, U., Aygen, Z., Tsai, J., 1996. A2: an architectural agent in a collaborative engineering environment. *Technical Report 48-38-96*, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh.

Fletcher, R., 1987. *Practical Methods of Optimization*, John Wiley and Sons.

Fleury, C., 1993. Mathematical programming methods for constrained optimization: Dual methods. In *Structural Optimization - Status and Promise*, M.P. Kamat, ed., Vol. 150 of Progress in Astronautics and Aeronautics, AIAA, 123-150.

Fogel, L.J., Owens, A.J., Walsh, M.J., 1966. *Artificial Intelligence through Simulated Evolution*,Wiley, New York.

Foggia, P., Genna, R., Vento, M., 1999. Introducing generalized attributed relational graphs (gargs) as prototypes of args. In *Proceedings of the 2nd IAPR Workshop on Graph-based Representations (GbR99)*, Haindorf, Austria.

Franti, P., Mednonogov, A., Kalviainen, H., 2000. Hough transform for rotation invariant matching of line-drawing images. *Proceedings, 15th International Conference on Pattern Recognition, Pattern Recognition*, 4(2):389-392.

Freeman, H., 1961. On encoding arbitrary geometric configurations. *IRE Trans. Electron. Comput.* 10:260-268.

Fuller, R.B., 1975. *Synergetics*, 2nd Ed., Macmillan, NY.

Fuyama, H., Law, K., Krawinkler, H., 1997. An Interactive Computer-Assisted System for Conceptual Structural Design of Steel Buildings. *Computers and Structures*, 63(4):647-662.

Garey, M.R. and Johnson, D.S., 1979. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., NY.

Gebhardt, F., Voβ, A., Gräther, W., Schmidt-Belz, B., 1997. *Reasoning with Complex Cases*, Kluwer Academic Publishers.

Gembicki, F.W., 1974. *Vector Optimization for Control with Performance and Parameter Sensitivity Indices*, PhD Thesis, Case Western Reserve Univ., Cleveland Ohio.

Gero, J.S., 1990. Design Prototypes: A Knowledge Representation Schema for Design, *Artificial Intelligence Magazine*, 11(4):26-36.

Gero, J.S., 1996. Creativity, emergence and evolution in design, *Knowledge-Based Systems*, Elsevier Science, 9:435-448.

Gero, J.S., 2000. Computational Models of Innovative and Creative Design Processes. *Technological Forecasting and Social Change*, Elsevier Science, 64:183-196.

Gielingh, W., 1988. General AEC Reference Model. ISO TC 184/SC4/WG1 Document 3.2.2.1, TNO Report BI-88-150.

Gill, P.E., Murray, W., Wright, M.H., 1981. *Practical Optimization*. Academic Press, London.

Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H., 1984. Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints. *ACM Trans. Math. Software*, 10:282-298.

Gill, P.E., Murray, W., Wright, M.H., 1991. *Numerical Linear Algebra and Optimization*, Volume 1, Addison Wesley.

Goel, A., 1989. *Integration of case-based reasoning and model-based reasoning for adaptive design problem solving*, PhD Dissertation, Department of Computer and Information Science, The Ohio State University.

Golabchi, M., 1997. Development of an expert system to select the appropriate structural systems for large span structures. *Proceedings of the ECCE Symposium*, RIL, Finland, 248-252.

Goldberg, D.E. and Samtami, M.P., 1986. Engineering optimization via genetic algorithm. In: *Ninth Conference on Electronic Computation*. New York, ASCE, pp. 471-482.

Grierson, D.E. and Pak, W.H., 1993. Optimal sizing, geometrical and topological design using a genetic algorithm. *Structural Optimization* 6:151-159.

Grierson, D. and Khajehpour, S., 2002. Method for Conceptual Design applied to Office Buildings. *Journal of Computing in Civil Engineering*, ASCE, 16(2):83-103.

Griffiths, D.R. and Miles, J.C., 2003. Determining the optimal cross-section of beams. *Advanced Engineering Informatics*, 17:59-76.

Gross, M.D. and Do, E.Y.-L., 1995. Diagram query and image retrieval in design. In *Proceedings, Second International Conference on Image Processing*, Crystal City, VA, IEEE Computer Society Press.

Gross, M.D., 1996. The Electronic Cocktail Napkin - a computational environment for working with design diagrams. *Design Studies*, 17:53-69.

Gruebler, M., 1885. Allgemeine eigenschaften der zwanglaufigen ebenen kinematischen ketten. *Civilingenieur*, 29:167-200.

Hajela, P. and Lee, E. 1995 Genetic algorithms in truss topological optimization. *Int. J. Solids Structures*, 32(22):3341-3357.

Hamda, H., Jouve, F., Lutton, E., Schoenauer, M., Sebag, M., 2002. Compact unstructured representations for evolutionary topological optimum design. *Applied Intelligence*, 16:139-155.

Han, S.P., 1977. A Globally Convergent Method for Nonlinear Programming. *J. Optimization Theory and Applications*, 22:297.

Hancock, E.R. and Kittler, J., 1990. Edge-labeling using dictionary-based relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):165-181.

Haralick, R.M., Shanmugam, K., Dinstein, I., 1973. Textural Features for Image Classification. *IEEE Trans., Systems, Man, and Cybernetics*, 3(6):610-621.

Harty, N., and Danaher, M., 1994. A knowledge-based approach to preliminary design of buildings. *Proceedings of the Institution of Civil Engineers*, 104:135-144.

Hauser, M., and Scherer, R.J., 1997. Application of intelligent CAD paradigms to preliminary structural design. *Journal of Artificial Intelligence in Engineering*, 11(3):217-229.

Hofmeyer, H., Rutten, H., Fijneman, H., 2006. Interaction of spatial design and structural design, an automated approach. *Journal of Design Studies* 27(4):423-524.

Holland, J.H., 1975. *Adaptation in Natural and Artifical Systems*. The University of Michigan Press, Ann Arbor, MI.

Hopcroft, J.E., and Wong, J.K., 1974. Linear time algorithm for isomorphism of planar graphs. *Proc., 6th Annual ACM Symposium on Theory of Computing*, 172-184.

Hough, P.V.C, 1959. Machine Analysis of Bubble Chamber Pictures. *Proc., Int. Conf. High Energy Accelerators and Instrumentation*.

Hu, M.-K., 1962. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179-187.

Huet, B., Hancock, E.R. 1999. Shape recognition from large image libaries by inexact graph matching. *Pattern Recognition Letters*, 20:1259-1269.

Jain, D., Krawinkler, H., Law, K., 1991. *Logic-based Conceptual Structural Design of Steel Office Buildings*, Technical Report, Center for Integrated Facility Engineering, Stanford University, CA.

Jenkins, W.M., 1991. Towards Structural Optimization via the Genetic Algorithm. *Computers and Structures* 40(5):1321-1327.

Jensen, E.D., 1992. Topological *structural design using genetic algorithms*, PhD Dissertation, Purdue University, Lafayette, IN.

Jurisica, I., and Glasgow, J., 2004. Applications of Case-Based Reasoning in Molecular Biology. *AI Magazine*, 25(1):85-95.

Kicinger, R., Arciszewski, T., and De Jong, K., 2003. Conceptual design in structural engineering: an evolutionary computation approach. *Proceedings of the 2nd International Specialty Conference on the Conceptual Approach to Structural Design*, Milan, Italy, July 1-2, 2003. CI-Premier PTE Ltd., Singapore, 529-536.

Kicinger, R., 2004. *Emergent engineering design: design creativity and optimality inspired by nature*. PhD Thesis, George Mason University.

Kicinger, R., Arciszewski, T., De Jong, K.A., 2005. Evolutionary computation and structural design: a survey of the state of the art. *Computers and Structures*, 83(23-24):1943-1978.

Kirsch, U., 1993. *Structural optimization: fundamentals and applications*, Springer, New York.

Kolodner, J., 1993. *Case-Based Reasoning*, Morgan Kaufmann, San Mateo, CA.

Koumousis, V.K. and Georgiou, P.G., 1994. Genetic Algorithms in Discrete Optimization of Steel Truss Roofs. *Computing in Civil Engineering*, 8(3):309-325.

Koza, J.R., 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA.

Krishnamoorthy, C.S., and Rajeev, S., 2000. *Artificial Intelligence and Expert Systems for Engineers*, CRC Press, Boca Raton.

Kuipers, B., 1984. Commonsense Reasoning about Causality: Deriving Behavior from Structure, *Artificial Intelligence*, 24:169-203.

Kuznetsov, E.N., 1975. Statical-kinematic analysis of spatial systems. In *Proc. Second Int. Conf. on Space Structures*, W.J. Supple, ed., University of Surrey, Guildford, 123-127.

Lam, L., Lee, S.W., Suen, C.Y. 1992. Thinning Methodologies - a Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869-885.

Leclerq, P., 1999. Interpretive Tool for Architectural Sketches. In *Visual and Spatial Reasoning in Design*, J. Gero, B. Tversky, eds., Key Centre of Design Computing and Cognition, University of Sydney, 1-11.

Ledley, R.S., 1964. High-speed automatic analysis of biomedical pictures. *Science*, 146(3641):216-223.

Lee, H.J., and Yoon, Y.S., 1992. Detection of rigid structure in enumerating basic kinematic chain by sequential removal of binary link string. *JSME International Journal*, 35(4):647-651.

Lee, R., and Liu, J., 1999. An automatic satellite interpretation of tropical cyclone patterns using elastic graph matching dynamic link model. *International Journal of Pattern Recognition and Artificial Intelligence*, 13(8):1251-1270.

Leung, M., Nevill, G.E., Jr., 1994. Genetic algorithms for preliminary 2-D structural design. *AIAA 2287-2291* (AIAA-94-1602-CP).

Lewinski, T., Zhou, M., Rozvany, G.I.N., 1994. Extended least-weight truss layouts. Part I: cantilever with a horizontal axis of symmetry. *Int. J. Mech. Sci*. 36:375-398.

Liang, Q.Q., Xie, Y.M., Steven, G.P., 1999. Optimal selection of topologies for the minimum-weight design of continuum structures with stress constraints. *Journal of Mechanical Engineering Science*, Proceedings of the Institution of Mechanical Engineers, UK, Part C, 23(C8):755-762.

Liang, Q.Q., Xie, Y.M., Steven, G.P., 2000. Optimal topology design of bracing systems for multistory steel frames. *J. Struct. Engrg*., 126(7):823-829.

Lin, C.-Y. and Hajela, P., 1993. Genetic search strategies in large scale optimization. *Proc. 34th AIAA/ASME/ASCE/AHS/ASC SDM Conf.*, La Jolla, CA.

Lin, T.Y., and Stotesbury, S.D., 1988. *Structural Concepts and Systems for Architects and Engineers*, 2nd Ed., Van Nostrand Reinhold, New York.

Louis, S.J., and Zhao, F., 1995. Domain Knowledge for Genetic Algorithms. *International Journal of Expert Systems*, 8(3):195-212.

Macquorn Rankine, W.J., and Millar, W.J., 1868. *Manual of Applied Mechanics*. Charles Griffin and Company, London.

Maher, M.L., Fenves, S.J., Garrett, J.H., 1988. Expert systems for structural design. In *Expert Systems in Construction and Structural Engineering*, H. Adeli, ed., Chapman and Hall, New York, NY.

Maher, M.L. and Zhang, D.M., 1993. CADSYN: A case-based design process model, *AI EDAM*, 7(2):97-110.

Maher, M.L., and Balachandran, M.B., 1994. Multimedia approach to case-based structural design. *Journal of Computing in Civil Engineering*, 8(3):359-376.

Maher, M.L., and de Silva Garza, A.G., 1997. Case-Based Reasoning in Design. *IEEE Expert*, 12(2):34-41.

Martini, K., and Powell, G.H., 1990. Geometric modeling requirements for structural design. *Engineering with Computers* 6:93-102.

Maxwell, J.C., 1864. On the calculation of equilibrium and stiffness of frames. *Phil. Mag.* 27 (4th Series), 294-299.

Messmer, B.T. and Bunke, H., 1998. Error-correcting graph isomorphism using decision trees. *International Journal of Pattern Recognition and Artificial Intelligence*, 12(6):721-742.

Meyer, S, 1995. *A Description of the Structural Design of Tall Buildings through the Grammar Paradigm*, PhD Thesis, Department of Civil Engineering, Carnegie Institute of Technology, Carnegie-Mellon University, Pittsburgh, PA.

Michell, A.G.M., 1904. The Limits of Economy in Frame Structures. *Philosophical Magazine*, Series 6, Vol. 8, No. 47, 589-597.

Mijar, A.R., Swan C.C., Arora, J.S., Kosaka, I., 1998. Continuum topology optimization for concept design of frame bracing systems. *J. Struct. Engrg.* 5:541-550.

Mlejnek, H.P., Schirrmacher, R. 1993. *An Engineer's Approach to Optimal Material Distribution and Shape Finding*, Computer Methods in Applied Mechanics and Engineering, 106(1-2):1-26.

Moore, R.E., 1966. *Interval Analysis*, Prentice Hall.

Mora, R., Bédard, C., Rivard, H., 2003. Integrated Computer-Based Approach for Conceptual Structural Design. *Proceedings of the Architectural Engineering 2003 Conference*, "Building Integration Solutions", ASCE, Mingsheng Lui and Kevin M. Parfitt, eds., September 17-20, 2003, Austin, Texas, 97-101.

Mora, R., Rivard, H., Bédard, C., 2005. From Architectural Sketch to Feasible System Solution, *Proceedings of 2005 ASCE International Conference on Computing in Civil Engineering*.

Mora, R., Bédard, C., Rivard, H., 2008. A geometric modeling framework for conceptual structural design from early digital architectural models, *Advanced Engineering Informatics*, 22:254-270.

Murawski, K., Arciszewski, T., De Jong, K., 2000. Evolutionary computation in structural design. *Engineering with Computers*, 16:275-286.

Nakanishi, Y. and Nakagiri, S. 1996. Optimization of frame topology using boundary cycle and genetic algorithms. *JSME International Journal, Series A*, 39:279-285.

Nakanishi, Y. and Nakagiri, S. 1997. Structural optimization under topological constraint represented by homology groups. *JSME International Journal, Series A*, 40:219-227.

Navinchandra, D., Sycara, K., Narasimhan, S., 1991. Behavioral Synthesis in CADET, a case-based design tool. In *Proceedings of the Seventh IEEE Conference on AI Applications*, Miami. IEEE Press.

Norton, R.L., 2003. *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*, 3rd. Ed. McGraw-Hill.

Olhoff, N., 1970. Optimal design of vibrating circular plates. *Int. J. Solids Struct*, 6:139-156.

Osherson, D.N., and Smith, E.E., 1981. On the adequacy of prototype theory as a theory of concepts. *Cognition* 9(1):35-38.

Packer, J.A., and Henderson, J.E., 2003. *Hollow Structural Sections, Connections and Trusses*, Second Ed., Canadian Institute of Steel Construction.

Pavlidis, T., 1977. *Structural Pattern Recognition*, Springer, New York.

Pedersen, P. 1992. Topology optimization of three-dimensional trusses. In *Topology Designs of Structures*, NATO ASI Series – NATO Advanced Research Workshop, Kluwer Academic Publishers, 19-30.

Pellegrino, S., Calladine, C.R., 1986. Matrix analysis of statically and kinematically indeterminate frameworks. *International Journal of Solids and Structures*, 22(4):409-428.

Pellegrino, S., 1990. Analysis of prestressed mechanisms. *International Journal of Solids and Structures*, 26(12):1329-1350.

Periaux, J., Winter, G. (eds.), 1995. *Genetic algorithms in engineering and computer science. Chichester*, UK, John Wiley.

Plaza, E., and López de Mántaras, R., 1990. A case-based apprentice that learns from fuzzy examples. In *Methodologies for Intelligent Systems*, Z. Ras, M. Zemankova, M.L. Emrich, eds., North Holland, 420-427.

Powell, M.J.D., 1978. A Fast Algorithm for Nonlinearly Constrained Optimization Calculations. *Numerical Analysis*, G.A. Watson, ed., Lecture Notes in Mathematics, Vol. 630, Springer-Verlag.

Powell, M.J.D., 1983. Variable Metric Methods for Constrained Optimization. *Mathematical Programming: The State of the Art*, A. Bachem, M. Grotschel, B. Korte, Eds., Springer-Verlag, 288-311.

Purcell, A.T. and Gero, J.S., 1998. Drawings and the design process. *Design Studies*, 19:389-430.

Rafiq, M.Y., Matthews, J.D., Bullock, G.N., 2003. Conceptual Building Design – Evolutionary Approach, *Journal of Computing in Civil Engineering*, ASCE, 17(3):150-158.

Rafiq, Y., Beck, M., Packham, I, Denhan, S., 2005. Evolutionary Computation and Visualisation as Decision Support Tools for Conceptual Building Design. *Innovation in Civil and Structural Engineering*, B.H.V. Topping (ed.), Saxe-Coburg Publications, pp. 49-74.

Rajan, S.D., 1995. Sizing, shape and topology design optimization of trusses using genetic algorithm. *Journal of Structural Engineering*, 121, 1480-1487.

Rajeev, S., and Krishnamoorthy, C.S., 1992. Discrete Optimization of Structures Using Genetic Algorithms. *The Structural Engineer*, 118(5):418-422.

Rajeev, S. and Krishnamoorthy, C.S., 1997. Genetic algorithms-based methodologies for design optimization of trusses. *Journal of Structural Engineering*, 123(3):350-358.

Rasmussen, J., and Olhoff, N., 1992. Status and Promise of Optimum Design System in Denmark. *Structural Optimization – Status and Promise*, M. Kamat, ed.

Ravi, M., and Bédard, C., 1993. Approximate methods of structural analysis and design in a knowledge-based environment. *Artificial Intelligence in Engineering*, 8(4):271-275.

Rechenberg, I., 1965. *Cybernetic Solution Path of an Experimental Problem*. Report No. 1122, Royal Aircraft Establishment, Farnborough, Hampshire, UK.

Reddy, G., and Cagan, J. 1995. An improved shape annealing algorithm for truss topology. *ASME Journal of Mechanical Design*, 117, 2A, 315-321.

Rivard, H., and Fenves, S.J., 2000a. A Representation for Conceptual Design of Buildings. *Journal of Computing in Civil Engineering*, ASCE, 14(3):151-159.

Rivard, H., and Fenves, S.J., 2000b. SEED-Config: A Case-based Reasoning System for Conceptual Building Design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 14:415-430.

Rosenfeld, A., 1969. *Picture Processing by Computer*, Computer Science and Applied Mathematics, Springer, Berlin, 196p.

Rosenman, M.A., and Gero, J.S., 1996. Modelling multiple views of design objects in a collaborative CAD environment. *Computer-Aided Design*, 28(3):193-205.

Rozvany, G.I.N., Olhoff, N., Bendsøe, M.P., Ong, T.G., Szeto, W.T., 1985. Least-weight design of perforated elastic plates. *DCAMM Report No. 306*. Technical University of Denmark, Lyngby.

Rozvay, G.I.N., 1989. *Structural design via optimality criteria*, Kluwer, Dordrecht.

Rozvany, G.I.N., 1997. Aims, Scope, Basic Concepts and Methods of Topology Optimization. In *Topology Optimization in Structural Mechanics*, G.I.N. Rozvany, ed., Springer-Verlag, Vienna.

Rozvany, G.I.N., 2009. A critical review of established methods of structural topology optimization. *Struct. Multidisc. Optim.*, 37(3):217-237.

Rozvany, G.I.N., and Zhou, M., 1991. The COC Algorithm, Part I: Cross-section Optimization or Sizing. *Computer methods in Applied Mechanics and Engineering*, 89:281-308.

Russ, J.C. 1999. *The Image Processing Handbook*, Third Ed., CRC Press.

Rutovitz, D., 1970. Centromere finding: Some shape descriptors for small chromosome outlines. *Machine Intelligence*, 5:435-562.

Sacks, R., and Warszawski, A., 1997. A project model for an automated building system: design and planning phases. *Journal of Automation in Construction*, 7(1):21-34.

Sandgren, E., Jensen, E.D., and Welton, J., 1990. Topological design of structural components using genetic optimization methods. In Sensitivity Analysis and Optimization with Numerical Methods, AMD Vol. 115, *Proceedings of the Winter Annual Meeting of the American Society of Mechanical Engineers*, Dallas, TX, 31-43.

Sauce, R., Martini, K., Powell, G.H., 1992. Object-oriented approaches for integrated engineering design systems. *ASCE Journal of Computing in Civil Engineering*, 6(3):248-265.

Schoenauer, M., 1996. Shape representations and evolution schemes. In L.J. Fogel, P.J. Angeline and T. Bäck (eds.), *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, San Diego, CA, USA.

Serra, J., 1982. *Image Analysis and Mathematical Morphology*, Academic Press, London, UK.

Shafer, G., 1976. *A mathematical theory of evidence*, Princeton University Press, Princeton, NJ.

Shapiro, L.G., and Haralick, M. 1981. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):504-519.

Shea, K., and Cagan, J., 1999. The Design of Novel Roof Trusses with Shape Annealing: Assessing the Ability of a Computational Method in Aiding Structural Designers with Varying Design Intent. *Design Studies*, 20:3-23.

Shen, Y.C., Bonissone, P.P., Feeser, L.J., 2001. Conceptual Modeling for Design Formulation. *Engineering with Computers*, 17:95-111.

Sherman, J., and Morrison, W.J., 1949. Adjustments of an inverse matrix corresponding to changes in elements of a given column or a given row of the original matrix. *Ann. Math. Statist.*, 20:621.

Shrestha, S.M., Ghaboussi, J., 1998. Evolution of optimum structures using genetic algorithm. *J. Struct. Engrg*. ASCE, 124(11):1331-8.

Sigmund, O., 1997. On the design of compliant mechanisms using topology optimization. *Mechanics of Structures and Machines*, 25(4):493-524.

Sigmund, O., 2001. A 99 line topology optimization code written in MATLAB. *Structural and Multidisciplinary Optimization*, 21:120-127.

Simon, H.A., 1969. *The Sciences of the Artificial*, Cambridge, MIT.

Sisk, G, Miles, J., Moore, C., 2003. Designer Centered Development of GA-Based DSS for Conceptual Design of Buildings. *Journal of Computing in Civil Engineering,* ASCE, 17(3):159-166.

Soh, C.K. and Yang, Y., 2001. Genetic programming-based approach for structural optimization. *Journal of Computing in Civil Engineering*, 31, 31-37.

Soibelman, L., and Peña-Mora, F., 2003. Distributed Multi-Reasoning Mechanism to Support Conceptual Structural Design. *Journal of Structural Engineering,* ASCE, 126(6):733-742.

Sriram, D., Logcher, R.D., Groleau, N., Cherneff, J., 1992. DICE: an object-oriented programming environment for cooperative engineering design. In *Artificial Intelligence in Engineering Design*, D. Sriram, C. Tong, ed., Academic Press, New York.

Sriram, D., 2006. Artificial intelligence in engineering: Personal reflections. *Advanced Engineering Informatics*, Elsevier Science, 20:3-5.

Svanberg, K., 1982. Optimal geometry in truss design. In *Foundations of Structural Optimization: A Unified Approach*, A.J. Morris, ed., Wiley, New York, 513-544.

Svanberg, K., 1987. The method of moving asymptotes - A new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24:359-373.

Svanberg, K., 1994. Global convergence of the stress ratio method for truss sizing. *Struct. Optim.*, 8:60-68.

Szeto, K., Roberts, S., MacMynowski, D., Sirota, M., Stepp, L., Gedig, M., Lagally, C., Tsang, D., 2008. TMT Telescope Structure System: Design and Development Progress Report. Proc. of SPIE, 7012-88.

Taylor, J.E., Rossow, M.P. 1977. Optimal truss design based on an algorithm using optimality criteria. *Int. J. Solids Struct.* 13, 913-923.

Thoresen, S., 2007. *An Efficient Solution to Inexact Graph Matching with Application to Computer Vision*. Doctoral thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway.

Topping, B.H.V., 1983. Shape Optimization of Skeletal Structures: A Review. *Journal of Structural Engineering*, 109(8):1933-1951.

Ullmann, J.R., 1976. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31-42.

Vassart, N., Laporte, R., Motro, R., 2000. Determination of mechanism's order for kinematically and statically indetermined systems. *International Journal of Solids and Structures*, 37, 3807-3839.

Vitruvius, 1999. *Vitruvius: Ten Books on Architecture.* Translated by I.D. Rowland with commentary by T.N. Howe, I.D. Rowland, and M.J. Dewar. Cambridge University Press, Cambridge.

Watson, I., and Perera, S., 1997. Case-based design: A review and analysis of building design applications. *AI-EDAM* 11(1):59-87.

Winston, P.H., 1975. Learning structural descriptions from examples. In *The Psychology of Computer Vision*, P.H. Winston, ed., McGraw-Hill, New York, 157-209.

Wolfram, 2009. Wolfram|Alpha computational knowledge engine, www.wolframalpha.com.

Wood, J. 1996. Invariant pattern recognition: A review. *Pattern Recognition*, 29(2):1-17.

Woodbury, M., 1950. *Inverting modified matrices*. Statistical Research Group, Princeton University, MR 42, Princeton, NJ.

Woodbury, R. and Chang, T.-W., 1995. Massing and enclosure design with SEED-Config. *Journal of Architectural Engineering*, ASCE, 1(4):170-178.

Xie, Y.M. and Steven, G.P., 1992. Shape and layout optimization via an evolutionary procedure. In Proceedings *of the International Conference on Computational Engineering Science*, Hong Kong University of Science and Technology, Hong Kong.

Yaner, P.W., and Goel, A.K., 2003. Visual Case-Based Reasoning I: Memory and Retrieval. *Proceedings of the First Indian International Conference on Artificial Intelligence*.

Yaner, P.W., and Goel, A.K., 2007. Visual Analogies at Multiple Levels of Abstraction. *Proceedings, 29th Annual Meeting of the Cognitive Science Society, CogSci-07*.

Yang, Y. and Soh, C.K., 2002. Automated optimum design of structures using genetic programming. *Computers and Structures* 80(18-19):1537-1546.

Yetis, F.A., and Saitou, K., 2002. Decomposition-Based Assembly Synthesis Based on Structural Considerations. *Journal of Mechanical Design*, 124:593-601.

Zadeh, L.A., 1963. Optimality and Nonscalar-valued Performance Criteria. *IEEE Trans. Automat. Contr.* Vol. AC-8, p.1.

Zhang, Y.Y. 1997. Redundancy of parallel thinning. *Pattern Recognition Letters*, 18:27-35.

Zhao, F., and Maher, M.L., 1988. Using analogical reasoning to design buildings. *Engineering with Computers*, 4(3):107-119.

Zhou, M., Rozvany, G.I.N., 2001. On the validity of ESO type methods in topology optimization. *Struct. Multidisp. Optim*. 21(1):80-83.