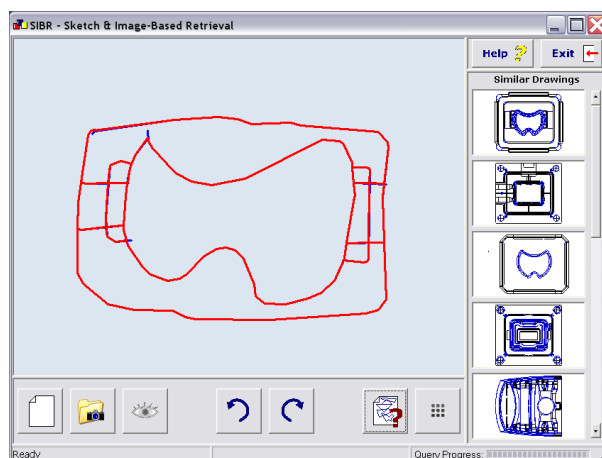




UNIVERSIDADE TÉCNICA DE LISBOA INSTITUTO SUPERIOR TÉCNICO



Sketch and Image Based Retrieval of Technical Drawings

Alfredo Manuel dos Santos Ferreira Jr.
(Licenciado)

Dissertation for the degree of Master of Science in
Information Systems and Computer Engineering

Adviser: Doutor Joaquim Armando Pires Jorge

Co-adviser: Doutor Manuel João Caneira Monteiro da Fonseca

Chairman: Doutor Joaquim Armando Pires Jorge

Members: Doutor Nuno Manuel Robalo Correia
 Doutor Mário António da Silva Neves Ramalho
 Doutor Manuel João Caneira Monteiro da Fonseca

June 2005

Sketch and Image Based Retrieval of Technical Drawings

Alfredo Manuel dos Santos Ferreira Jr.

A Thesis submitted to the Graduate School
for the degree of
Master of Science in
Information Systems and Computer Engineering

Department of
Information Systems and Computer Engineering
Instituto Superior Técnico

Adviser

Prof. Doutor Joaquim Armando Pires Jorge

Professor Auxiliar com Agregação from the
Department of Information Systems and Computer Engineering
Instituto Superior Técnico
Technical University of Lisbon

This work was funded in part by the Portuguese Foundation for Science and Technology, project 34672/99 and the European Commission, project SmartSketches IST-2000-28169.

Alfredo Ferreira Jr.

<http://immi.inesc-id.pt/~afj>

alfredo.ferreira.jr@inesc-id.pt

Resumo

Nesta dissertação, proponho um método para especificar as interrogações, que usa simultaneamente esboços e imagens digitalizadas. Desta forma conseguimos tirar partido não só da memória visual dos utilizadores e da sua habilidade para desenhar, mas também de desenhos já impressos para rapidamente especificar interrogações complexas.

Para tal, desenvolvi mecanismos de vectorização, bem como algoritmos de detecção de polígonos, simplificação e extracção de características de desenhos vectoriais. Estes foram integrados numa moldura para recuperação baseada em conteúdos que fornece uma estrutura de indexação e suporta todo processo de pesquisa, produzindo deste modo uma solução funcional para recuperação de desenhos técnicos combinando esboços e imagens. Com base em testes realizados com utilizadores conseguiu-se validar o paradigma proposto e a eficiência dos algoritmos, sendo possível afirmar que a combinação de esboços e imagens para especificar interrogações agradou aos utilizadores e que estes consideraram o sistema rápido e preciso.

Palavras-Chave

Recuperação Baseada no Conteúdo

Recuperação Usando Esboços

Vectorização de Imagens

Detecção de Polígonos

Simplificação de Desenhos

Extracção de Características

Abstract

In this dissertation, I propose a query specification scheme, where digital images are combined with sketches, after vectorization, taking advantage not only of users' visual memory and their ability to sketch but also of existing paper drawings to quickly specify complex queries.

To create my retrieval system, I used an existing framework for content based retrieval which provides an indexing structure and supports the matching process. I developed vectorization mechanisms and simplification, polygon detection and feature extraction algorithms. Then, these algorithms were integrated in the framework in order to create a functional solution for retrieval of technical drawings using both images and sketches. This system was evaluated by users to validate the interaction paradigm and the efficiency of algorithms. From these tests I concluded that users liked using sketches and images to specify queries and considered the system both accurate and fast.

Keywords

Content-Based Retrieval

Sketch-Based Retrieval

Image Vectorization

Polygon Detection

Drawing Simplification

Feature Extraction

To my Mother,
In Memoriam

Acknowledgements

First, I would like to express thanks to my adviser Professor Joaquim A. Jorge and co-adviser Professor Manuel J. Fonseca, for their continuous support during my MSc. Professor Joaquim A. Jorge thank you for always be there to meet and talk about my ideas and to provide me with good guidance. Your constant comments and the confidence you showed to me and to my research contributed to improve the quality of the developed work. Professor Manuel J. Fonseca thanks for your friendship, for your interminable patience and for your endless trust even when things seem not to have an end.

In second place, I would like to show gratitude to all the members of the IMMI group for their friendship and help, specially to Filipe Dias for being a good friend and someone you can always rely on, Bruno Araújo for his wide knowledge on Computer Graphics and his availability, Frederico Figueiredo for his different opinions and for the stimulating discussions, and Paula Caetano.

Next, I would like to credit the people at CENTIMFE that made possible a collaboration with the mould industry during the development of this work. Their ideas were a valuable contribute to the work presented in this dissertation.

Finally, and the last ones are always the first, I would like to thank my wife Sandra who supported me unconditionally and my son Tiago who gave me additional motivation to finish my work. Also, a very special thank to my aunt Bernardete for her support.

Lisboa, June 2005

Alfredo Manuel dos Santos Ferreira Jr.

Contents

Resumo	i
Abstract	iii
Acknowledgements	vii
Contents	ix
List of Figures	xv
List of Tables	xviii
Glossary	xxi
1 Introduction	1
1.1 Context of the Present Work	3
1.2 Extracting Features from Drawings and Sketches	3
1.3 Contributions	4
1.4 Publications	5
1.5 Dissertation Outline	6
2 Related Work	9
2.1 Content-Based Retrieval	9
2.2 Working with digitized drawings	14
2.3 Vectorization of engineering drawings	15
2.3.1 Hough Transform	16
2.3.2 Thinning based methods	18
2.3.3 Contour based approach	20

2.3.4	Run-graph based methods	21
2.3.5	Mesh pattern based methods	23
2.3.6	Sparse pixel based approaches	23
2.3.7	Discussion on Image Vectorization techniques	24
2.4	Curve Simplification	25
2.4.1	Discussion on Curve Simplification	28
2.5	Summary	29
3	Sketch and Image Based Retrieval of Engineering Drawings	31
3.1	Approach Overview	32
3.1.1	Use Scenarios	33
3.1.1.1	Scenario A: Creating a Mould	33
3.1.1.2	Scenario B: Looking for a Part	34
3.1.2	System Architecture	34
3.2	Image Vectorization	36
3.2.1	Contrast Enhancement	37
3.2.2	Edge Enhancement	38
3.2.3	Binarization	40
3.2.4	Binary Noise Reduction	43
3.2.5	Thinning	44
3.2.6	Polygonization	45
3.3	Feature Extraction	46
3.3.1	Simplification	47
3.3.1.1	Line Set Simplification	48
3.3.1.2	Polygon Simplification	50
3.3.2	Polygon Detection on a Vector Drawing	56
3.3.2.1	Removal of Line Segment Intersections	56
3.3.2.2	All Cycles of a Graph	58
3.3.2.3	Minimum Cycle Basis of a Graph	59
3.3.2.4	Polygon Construction	60
3.3.2.5	Polygon Detection Outline	61
3.3.2.6	Experimental Results on Polygon Detection	62
3.3.3	Topological Information Gathering	62
3.3.4	Geometrical Feature Extraction	64

3.4	Indexing, Query and Matching	66
3.4.1	Classification	66
3.4.2	Retrieval	67
3.5	Summary	68
4	Prototypes	69
4.1	Sketch and Image Based Retrieval Prototype	69
4.1.1	Architecture	70
4.1.1.1	User Interface Tier	70
4.1.1.2	Application Core Tier	72
4.1.1.3	File Management Tier	73
4.1.2	Graphical User Interface	74
4.1.2.1	Using Sketches	75
4.1.2.2	Using Images	77
4.1.2.3	Mixing Sketches and Images	79
4.2	Database Builder	79
4.2.1	Architecture	80
4.2.2	Graphical User Interface	81
4.3	Summary	82
5	Experimental Results	83
5.1	Sketching Experiment	84
5.1.1	Sketch Reader Prototype	84
5.1.2	Users	85
5.1.3	Experiment Steps	86
5.1.4	Technical Drawings	86
5.2	Sketches	87
5.2.1	First Sketching Session	87
5.2.2	Second Sketching Session	89
5.2.3	Analysis of the Sketching Experiment	91
5.2.4	Conclusions from sketching experiment	92
5.3	Preliminary Usability Evaluation	93
5.3.1	Users	93
5.3.2	Usability Test Session	93
5.3.3	Drawings Database	94

5.3.4	Queries	95
5.3.5	Final Questionnaire	96
5.3.6	Sketches	96
5.3.6.1	Basic Drawings	96
5.3.6.2	Simple Technical Drawings	98
5.3.7	Test Analysis	99
5.3.8	Questionnaire analysis	101
5.3.9	Conclusions from preliminary usability tests	101
5.4	Final Usability Evaluation	102
5.4.1	Users	102
5.4.2	Usability Test Session	103
5.4.3	Drawings Database	104
5.4.4	Queries	104
5.4.5	Final Questionnaire	105
5.4.6	Sketches	105
5.4.7	Test Analysis	108
5.4.8	Questionnaire Analysis	111
5.4.9	Conclusions from final usability tests	112
5.5	Discussion	112
5.6	Summary	114
6	Conclusions and Future Work	117
A	Sketches from Sketching Experiment	121
A.1	First Sketching Session	123
A.2	Second Sketching Session	124
B	Testing Protocols	125
B.1	Protocol for Preliminary Usability Evaluation	127
B.2	Protocol for Final Usability Evaluation	132
C	Questionnaires	137
C.1	Questionnaire of the preliminary evaluation tests	139
C.2	Questionnaire of the final evaluation tests	143
D	Databases	147

D.1	Basic Drawings	149
D.2	Simple Technical Drawings of Mould Plates	150
D.3	Technical Drawings of Parts	151
E	Sketched Queries	153
E.1	Preliminary Usability Evaluation	155
E.1.1	User A	155
E.1.2	User B	156
E.1.3	User C	158
E.2	Final Usability Evaluation	159
E.2.1	User A	159
E.2.2	User B	162
E.2.3	User C	164
E.2.4	User D	166
E.2.5	User E	168
E.2.6	User F	171
	Bibliography	175

List of Figures

1.1	Overview of architecture for a engineering drawings indexing and retrieval solution. The main contribution of the present work is the feature extraction component. Fonseca’s framework is represented within the dashed boundaries.	2
2.1	Screen-shots of Gross’s Electronic Cocktail Napkin, querying the Great Buildings Collection.	10
2.2	Screen-shots of sketched query (a) and corresponding results (b) yielded by the 2D-PIR prototype.	11
2.3	Screen-shots of a query on S3 (a) and correspondent results(b).	12
2.4	Screen-shot of Princeton Search Engine for 3D Models.	13
2.5	Screen-shot of Bajavista.	14
2.6	Example of Hough transform using trigonometric parameters: (a) Image with four lines; (b) corresponding Hough space. Each peak corresponds to a line. The dominant peak (the topmost) represents the longer line. . . .	17
2.7	Peeling iterations: (a) original image; (b), (c), (d) intermediate results after first, second and third iterations; (e) skeleton.	19
2.8	Thinning using distance transform: (a) original image; (b) distance transform; (c) skeleton.	20
2.9	Example of skeleton sampling: (a) parallel edges; (b) almost parallel edges; (c) cross-junction.	21
2.10	Simplified example of run graph representation: (a) vertical partitioning; (b) horizontal partitioning; (c) computed graph.	22
2.11	Example of Orthogonal Zig-Zag algorithm.	24
2.12	Example of regular sub-sampling using $n = 3$. Original curve (left) and its caricature (right).	26
2.13	Example of curve simplification using Douglas-Peucker. Original curve (left) and its caricature (right).	26
3.1	Searching a mould component using a sketched-query.	33

3.2	Retrieving a mould component and integrating it into a drawing.	34
3.3	System architecture of proposed solution. Feature Extraction and Image Vectorization components are my main contributions, while remaining components are provided by Fonseca’s framework.	35
3.4	Image Processing Steps.	36
3.5	Original gray-scale image (a) and yielded result after contrast and edge enhancement (b).	38
3.6	Binary Image obtained using fixed threshold (a) and region averaging (b).	40
3.7	Image of a digitized engineering drawing (a) and its intensity histogram (b). At right of the histogram the peak is formed the background while the small peak at left corresponds to foreground. Threshold value should be just at left of the larger peak.	41
3.8	Evolution of image detail during processing.	42
3.9	Pictorial example of speckle noise reduction: original image (a) underwent several iterations of the noise reduction algorithm based on the kFill filter, which yielded the final image (b).	43
3.10	Freeman chain coding: chain direction codes (a), and an example of line structure with starting coordinates and direction codes. The resulting Freeman code for this image is: (0, 0)3, 2, 2, 2, 2, 1, 5(3, 1)4, 4, 5, 1	45
3.11	Thinned binary image (a) and vector version of drawing (b).	46
3.12	Feature extraction block diagram decomposition.	47
3.13	Arrangement of segments before (a) and after (b) traditional snap rounding.	48
3.14	Arrangement of segments with a vertex very close to a non-incident line after (b) snap rounding.	49
3.15	Results produced by Snap Rounding (b) and Iterated Snap Rounding (c) when applied to a set of line segments (a).	50
3.16	Example of polygon detection on a simple drawing: (a) original set of lines and (b) detected polygons with vertices marked.	50
3.17	Example of polygon detection and coherence simplification on a simple drawing: (a) detection algorithm yields desired result and (b) an heuristic simplification must be applied to set of detected polygons.	51
3.18	Example of boolean operations over polygons: (a) original set of two polygons, (b) intersection result and (c) exclusive-or result.	53
3.19	Example of small polygon removal: (a) original set of polygons, (b) after small polygon identification, (c) after merging of small polygons and (d) simplified polygons.	55
3.20	Set Φ of line segments (a) and graph G induced by Φ (b).	57
3.21	A planar graph with a exponential number of cycles	58

3.22	Shortest cycle basis Γ of graph G (a) and set Θ , constructed from Γ , of polygons detected in original set of line segments Φ	60
3.23	Set of simplified polygons (left) and correspondent topology graph (right).	63
3.24	SIBR geometric feature vector used in construction of geometry matrix. .	65
3.25	Block diagram for the matching process.	67
4.1	Conceptual view of Sketch and Image Based Retrieval Prototype architecture.	70
4.2	Example of single stroke (top) and multi-stroke (bottom) auto-completion.	71
4.3	Screen-shot of Graphical User Interface of SIBR.	75
4.4	Using delete gesture to remove a stroke.	76
4.5	Selecting part of a sketch	77
4.6	Screen-shot of image processing window	78
4.7	Mixing images and sketches to retrieve a technical drawing.	79
4.8	Conceptual view of Database Builder prototype architecture.	80
4.9	Screen-shot of Database Builder prototype during a classification process.	81
5.1	Sketch Reader: sketching prototype application	84
5.2	User during a sketching session.	85
5.3	Top view of drawings D1 (a) and D2 (b).	87
5.4	Drawings D3 (a), D4 (b) and D5 (c).	87
5.5	Sketches representing drawing D3 produced by user B in first (a) and second (b) sketching sessions.	90
5.6	Time comparison.	92
5.7	Detail of a sketch made during second session (a), same detail of the original drawing (b) and amplified sketch of a small circle (c).	92
5.8	Basic drawings to search in the database.	95
5.9	Simple technical drawings to search in the database.	95
5.10	Sketch for Q1 made by user A and returned drawings.	97
5.11	Sketches made by user C for Q3 (left) and Q4 (right).	97
5.12	Sketch of two concentric circles.	98
5.13	Detail of the sketch made by user A for Q9.	98
5.14	Original drawing (Q10) and sketches performed by each user.	99
5.15	Overall position of the desired drawing in the results list.	100
5.16	Number of sketches drawn before finding the correct result.	100
5.17	Time spent performing queries.	101

5.18	User and observer during the training phase.	103
5.19	Technical drawings selected as queries.	104
5.20	Sketched queries made by user A for Q3.	105
5.21	Sketch S2C (a) made by user C for Q2 and details of sketch (b) and drawing (c).	106
5.22	”Good” sketches made for Q4 that returned successful results.	107
5.23	Sketches made for Q4 that returned unsuccessful results.	107
5.24	Detail of second SBR prototype showing the ”Show all results” button and part of results area scroll bar	108
5.25	Overall position of the desired drawing in the results list.	109
5.26	Average overall position of query results grouped by user experience. . .	110
5.27	Pie charts representing query results distribution (a) and number of sketches drawn before finding a correct result (b).	111

List of Tables

3.1	Results of algorithm tests	62
3.2	List of relevant geometrical features.	65
5.1	Sketching times for the first session	89
5.2	Sketching times for the second session	91

Glossary

AABB

Bounding Box aligned with the axes of the coordinate system. Axis-aligned bounding boxes are simpler to test for intersection than oriented bounding boxes but have the disadvantage that they cannot be rotated.

binarization

process to convert gray-scale images into binary images.

KNN

K-Nearest Neighbor queries return the K closest answers according to given distance metric in the database with respect to a query point.

MCB

A minimum cycle basis in an undirected graph G is a set of simple cycles whose incidence vectors span the cycle space of G and whose overall edge sum is minimal.

OBB

The smallest cuboid that encloses an object or set of objects.

OZZ

Orthogonal Zig-Zag Vectorization algorithm that samples the image sparsely.

PCC

Extension of Freeman chain code designed to preserve information on branching and junction topology.

SBR

Preliminary Sketch-Based Retrieval approach developed during the work described in this dissertation.

SIBR

Sketch and Image Based Retrieval approach proposed in this dissertation to retrieve engineering drawings through combining images and sketches.

SPV

Sparse Pixel Vectorization algorithm based on the OZZ algorithm.

Vectorization

Process to convert raster images into vector format.

1

Introduction

Over the past few decades, the widespread use of CAD applications has resulted in large numbers of engineering drawings in diverse domains such as architectural or industrial design. However, current systems provide only conventional database queries or direct-manipulation mechanisms to retrieve such drawings.

From task analysis and informal conversations with draftspeople we found out that they often re-use data from previous projects, publications and libraries of ready-to-use components. Even though reusing drawings may save time, searching for them is usually slow and problematic, requiring browsing through large directories or navigating through a maze of menus and dialogs in the case of component libraries. Unfortunately, the popularity of CAD systems, while making it easier to create and edit drawings, exacerbates this problem, insofar as the number of projects and drawings grows enormously, without adequate search mechanisms to support retrieving these documents.

Thus, there is a need for devising techniques to automatically classifying and retrieving drawings based on their content. Some approaches to classification and retrieval of drawings use textual databases to organize the information [2, 16]. These classify drawings by keywords and additional information, such as designer name, style, date of creation/modification and a textual description. However, solutions based on textual queries are not satisfactory, because they force designers to know in detail the meta-data used to characterize drawings. Moreover, such textual information requires time and effort to be correctly introduced.

In contrast to textual methods, a generic approach to the problem of retrieving drawings was presented by Fonseca in his PhD thesis [34]. He proposed a visual classification scheme based on shape geometry and spatial relationships, which are better suited to this

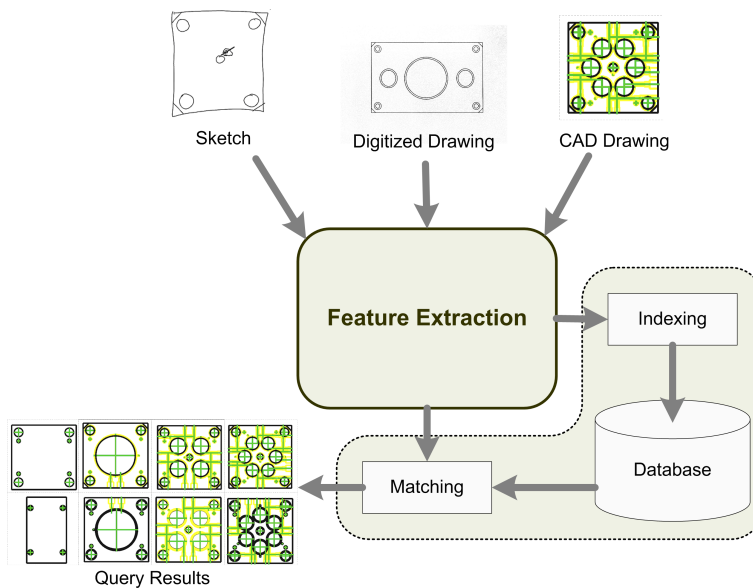


Figure 1.1: Overview of architecture for a engineering drawings indexing and retrieval solution. The main contribution of the present work is the feature extraction component. Fonseca's framework is represented within the dashed boundaries.

problem. Furthermore, he presented a novel indexing mechanism that efficiently supports large sets of drawings.

However, to use the framework presented by Fonseca with engineering drawings in an effective manner, it becomes necessary to classify such drawings correctly. Since technical drawings are often composed by hundreds or thousands of entities, extracting visual features from them is not straightforward or trivial.

In this dissertation I present an approach to retrieve engineering drawings from large databases by combining sketches and images. To that end, I propose an effective methodology to classify technical drawings or sketches and a technique to vectorize scanned engineering drawings yielding sketchable queries. This technique was integrated with Fonseca's framework to create an efficient method for engineering drawing indexing and retrieval. Figure 1.1 presents an overview of the proposed architecture to achieve this goal.

1.1 Context of the Present Work

The work presented in this dissertation was developed at Intelligent Multimodal Interfaces research group¹, within the *SmartSketches*² European project. *SmartSketches* was developed by a consortium joining three European countries with a well balanced mix of University, Industry and end-user partners. Its main objective was to introduce novel computer-based tools at the initial stage of design.

One problem addressed under *SmartSketches* was the lack of efficient techniques for engineering drawing retrieval. My research in this area focused on classification and retrieval of engineering drawings for the mould industry and provided the context to the present dissertation.

Throughout this dissertation I will use “we” to describe group work, while “I” refers to my contributions.

1.2 Extracting Features from Drawings and Sketches

In order to classify mould drawings, I developed a method to extract geometric and topological features from engineering drawings and sketched-queries, which yields the corresponding topology graphs and geometry descriptors. This process was integrated with Fonseca’s framework to yield a system for indexing, searching and retrieving 2D engineering drawings. Additionally, I created the user interface to the whole system and tested it with users, to evaluate both the system usability and the performance of classification and retrieval algorithms.

Feature extraction comprises a set of steps for simplifying drawings and sketches and detecting shapes. Thus, I developed simplification algorithms for drawings and queries as well as a novel³ simple polygon detection algorithm. This algorithm yields a set of

¹The IMMI group, at INESC-ID (Lisbon, Portugal), does research in novel user interaction paradigms and applications for Design and Manufacturing applications. The group has focused its activities on Calligraphic Interfaces, but also work on other interaction modalities such as speech, sound and vision. More information about IMMI group can be found in the web site <http://immi.inesc-id.pt>.

²More information about the *SmartSketches* project can be found at <http://smartsketches.inesc-id.pt>.

³Unlike image processing, where data consist of raster images, the proposed polygon detection algorithm deals with drawings in vector format, consisting of line segments. This requires a completely different

shapes that will be considered when classifying the drawing or query. I extract geometric features from these shapes using CALI, a powerful scribble recognizer [39]. To construct the topology graph, needed by Fonseca's framework, I developed a simple algorithm that computes topological relationships among detected polygons.

1.3 Contributions

The main outcome of my work is an approach to automatic classification and interactive retrieval of engineering drawings by combining images and sketches to specify queries. Additionally, I devised a novel polygon detection and simplification algorithm and implemented methodologies for simplification and vectorization of technical drawings were implemented. Moreover, I studied the way users sketch queries, in order to understand how they perceive and represent visual elements of a drawing. My research has yielded the following contributions:

- **A method for simplification of technical drawings.**

Efficient classification and retrieval of engineering drawings requires effective simplification methods. This is because technical drawings are often composed by thousands of elements, most of which are not relevant for classification. To this end, I devised simplification methodologies that remove irrelevant features from drawings, in two separate steps. Initially, drawings are considered as a set of line segments and simplified using an iterated snap rounding method. After detecting polygons from the remaining segments, a small polygon removal procedure and a coherence normalization heuristic are used to further simplify the drawings.

- **An algorithm for polygon detection from vector drawings.**

Our approach classifies technical drawings according to topological relationships of relevant shapes and their geometry. Thus, it is necessary to have an efficient way to identify these shapes, since irrelevant features are already removed by the simplification techniques. Unlike most published methods, which use raster information to

approach. Known algorithms detect polygons only on raster images. Until now, no efficient solution has been proposed to detect polygons directly from vector drawings.

detect shapes, I developed a novel algorithm to detect polygons from a set of lines in vector format. This algorithm combines several methods from computational geometry and graph theory to yield a set of non-intersecting polygons in polynomial time.

- **Heuristics for identifying relevant items in sketched queries.**

To take advantage of users' natural ability at sketching, I need to understand how they draw 2D views of parts. Moreover, any solution for classification and retrieval of engineering drawings based on visual features can only be effective if the shapes considered relevant by the classification heuristics match those users consider relevant. Therefore, we invested considerable efforts on performing tests with users and studying sketches drawn as queries. From the analysis of such sketches I was able to identify which features users consider relevant and those that can be discarded. These results were crucial to developing simplification methodologies.

- **A paradigm for drawing retrieval combining images and sketches.**

Using the framework presented by Fonseca [34], I devised a novel method for retrieving technical drawings based on content. Previous approaches to content-based retrieval of drawings based on visual features rely exclusively on sketches or images to specify queries. Some approaches use text to enrich the query, but none combine images with sketches. I developed a method that allows mixing both images and sketches to retrieve technical drawings in a more flexible manner.

1.4 Publications

The work described in this dissertation yielded five original publications accepted in peer-reviewed scientific meetings and journals, which are listed in reverse chronological order by date of publication.

1. Manuel J. Fonseca and **Alfredo Ferreira** and Joaquim A. Jorge.

Content-Based Retrieval of Technical Drawings. International Journal of Computer Applications in Technology, March 2005.

2. **Alfredo Ferreira** and Manuel J. Fonseca and Joaquim A. Jorge and M. Ramalho. *Mixing Images and Sketches for Retrieving Vector Drawings*. Proceedings of the 7th Eurographics Workshop on Multimedia, pages 69-75, Nanjing, China, Oct 2004.
3. Manuel J. Fonseca and **Alfredo Ferreira** and Joaquim A. Jorge. *Retrieving Mould Drawings by Content*. Proceedings of Rapid Product Development (RPD 2004), Marinha Grande, Portugal, Oct 2004.
4. Manuel J. Fonseca and **Alfredo Ferreira** and Joaquim A. Jorge. *Towards 3D Modeling using Sketches and Retrieval*. Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling 2004, pages 127-136, Grenoble, France, Aug 2004
5. **Alfredo Ferreira** and Manuel J. Fonseca and Joaquim A. Jorge. *Polygon Detection from a Set of Lines*. Proceedings of 12^o Encontro Português de Computação Gráfica (12^o EPCG), pages 159-162, Porto, Portugal, Oct 2003.

1.5 Dissertation Outline

The rest of this dissertation is organized in five chapters and one appendix.

Chapter 2 surveys the work related to my research, which includes content based retrieval of vector drawings. I also review the main image vectorization techniques and polygonal curve simplification methods.

Chapter 3 describes in detail my approach to sketch and image based retrieval of engineering drawings. Initially, it gives an overview, presenting use scenarios and describing the system architecture. Then, it describes a vectorization technique to convert raster images of engineering drawings to vector format. Next it details every step of the classification process, namely the simplification, polygon detection and feature extraction algorithms developed. Finally, it describes the query and matching methods.

The prototypes developed to exercise my approach are described in Chapter 4. The proposed system was divided in two distinct applications. One application performs drawing classification while the other allows users to specify queries through sketches using a calligraphic interface to yield the set of drawings similar to a sketched query.

These prototypes were evaluated by users in two distinct usability evaluation sessions. Additionally, a sketching experiment with users was performed to understand how they formulate queries for engineering drawings. The three experimental tests are described and their results analyzed in Chapter 5.

Chapter 6 presents an overall discussion of my work, presents conclusions and proposes directions for future work. I point possible paths for further research on image vectorization, drawing simplification and polygon detection.

Finally, five appendices provide additional information related to the experiments with users, described in Chapter 5. The first appendix, lists the sketches made by users during the initial sketching experiment. The next three contain the protocols, questionnaires and databases used in both usability evaluations. The last appendix exhaustively lists the sketched queries made by users during usability evaluations and corresponding results.

2

Related Work

Last decade has seen several systems for content-based retrieval of vector drawings and images. In most of these, queries can be specified by either using sketches, keywords or images. Some of them allow mixing textual keywords with sketches or images, but never mixing both sketches and images, as I suggest in the present work.

Since I intended to use digitized images to specify queries, I needed to develop an automatic method to classify these drawings. However, automatic classification of scanned engineering drawings based on visual features depends to a great extent on the quality of the extracted features. Thus, vectorization and simplification are key methods, whose accuracy determines the quality of the features.

This chapter reviews existing systems for content-based retrieval of vector drawings, examining their methods and domains of application. Then, it describes and compares existing vectorization algorithms and presents a comprehensive analysis of curve simplification techniques.

2.1 Content-Based Retrieval

Recently there has been considerable interest in querying multimedia databases by content. However, most such work has focused on image databases, as surveyed by Shi-Kuo Chang [13]. Moreover, in [99], the author analyzes several image retrieval systems which use color and texture to describe images. However, drawings in electronic format are represented in structured form (vector graphics), requiring different approaches from image-based methods, which resort to color, regions and texture as the main features to describe content. Some initial work [2, 16] attempted to index technical drawings through

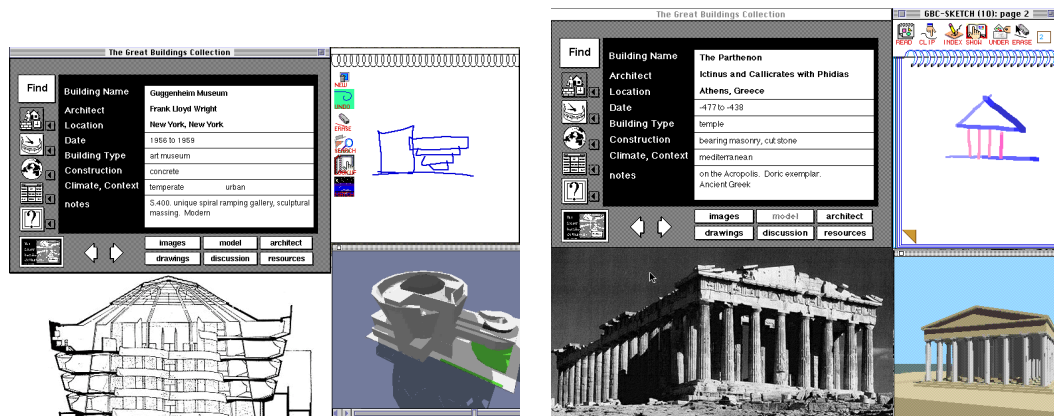


Figure 2.1: Screen-shots of Gross's Electronic Cocktail Napkin, querying the Great Buildings Collection.

textual databases. However, these fail to use the rich visual association mechanisms and designer's use of sketches to recover information.

Gross's Electronic Cocktail Napkin [45, 24, 44] is a pen based drawing environment that aims to support designing using hand drawn sketches and diagrams. This system uses a simple, trainable, on-the-fly recognition scheme to identify multi-stroke glyphs drawn using a digitizing tablet. The authors extended the Electronic Cocktail Napkin to allow visual queries and visual bookmarking in drawings databases. To that end, they addressed a visual retrieval scheme based on diagrams, to index databases of architectural drawings. To retrieve indexed drawings, users draw sketches of buildings, which are compared with annotations (diagrams) of , stored in a database and manually produced by users. Even though this system works well for small sets of drawings, the lack of automatic indexation and classification makes it difficult to scale the approach to large collections of drawings. Figure 2.1 illustrates an application of Electronic Cocktail Napkin, depicting the results of searching for buildings in a database using hand-sketched queries.

Nabil et al. [79] proposed techniques for similarity retrieval based on 2D Projection Interval Relationships (2D-PIR), including methods for dealing with rotated and reflected images. The 2D-PIR is a symbolic representation of directional as well as topological relationships among spatial objects. It combines three existing representation formalisms in a novel way to produce a unified representation of pictures, integrating both directional and topological relationships. Authors claim that their method encodes more information about spatial relationships between objects in a picture than traditional methods. Further-

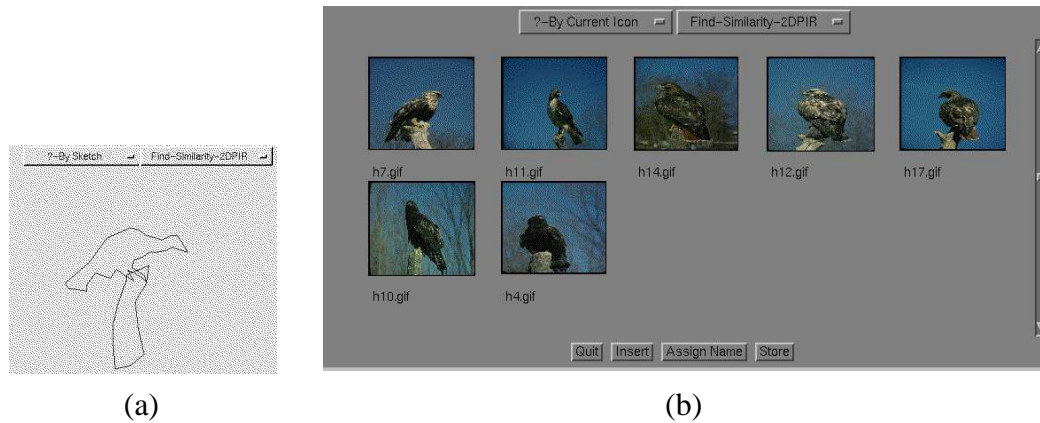


Figure 2.2: Screen-shots of sketched query (a) and corresponding results (b) yielded by the 2D-PIR prototype.

more, they suggest that it should be easy to extend this approach to three dimensional objects and even to video classification by incorporating time intervals into relationships. A later publication [80] presents a prototype image retrieval system for distributed environments based on 2D-PIR. Figure 2.2 depicts a sketched query and the results yielded by the prototype. This system transforms each image into a symbolic representation using the techniques described above and then is able to answer to sketched-queries. However, the symbolic representation of the query needs to be compared to all the symbolic representations stored in the database, making this approach difficult to scale up for large collections of images.

The Similarity Search System (S3) [8] is able to retrieve industrial CAD parts, described by their contours and thematic attributes. S3 retrieves parts using bi-dimensional polygons drawn with a graphical editor. It is also possible to use sample parts from a database, as depicted in Figure 2.3. The S3 uses four similarity algorithms to match queries against parts in the database. The first is a modified version of the Mehrotra-Gary algorithm [74] that determines angles and lengths of segments in a region. Another method is the angular profile [1], which computes angles between line segments defined by sample points from the contour. A third method is section coding which determines the portion of the contour inside each sector resulting from equally dividing the circle surrounding the polygon. Finally, a Fourier-based method [7] can detect partial similarity. Authors used special data structures and indexing algorithms, such as the R*-Tree and the X-Tree to avoid exhaustive search. However, S3 relies exclusively on matching contours, ignoring spatial relations and shape information, making this method unsuitable

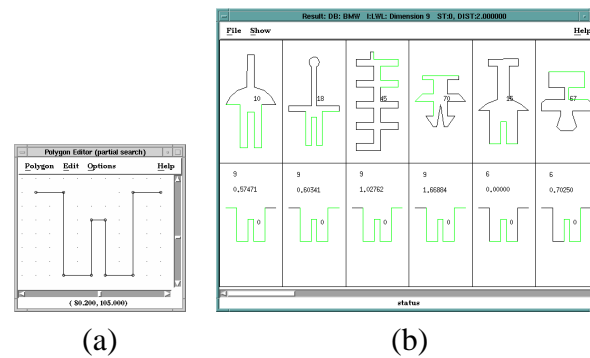


Figure 2.3: Screen-shots of a query on S3 (a) and correspondent results(b).

for retrieving complex multi-shape drawings.

Müller and Rigoll presented an approach [78, 77] to retrieve engineering drawings based on stochastic models. Engineering drawing databases can be searched using sketches or shapes which match details in drawings of mechanical parts. This technique represents drawings and queries using a pseudo 2-D Hidden Markov chain with filler models. This makes it possible to match images by specifying details and to locate such details in the retrieved images, even when queried shapes are embedded in hatching or connected to other parts in the drawing. However, the method proposed by Müller and Rigoll only allows specifying simple queries, representing a single element. Indeed, more complex queries including several elements with spatial relationships between them are not possible. Furthermore, the search mechanism is not appropriated for large collections of drawings, since they perform a sequential scan through the database comparing the query with all indexed drawings.

Leung and Chen proposed a sketch retrieval method [70] for general unstructured free-form hand-drawings stored as multiple strokes. They use shape information from each stroke and exploit geometric relationships between multiple strokes for matching. Their approach computes a matching score between a query and each sketch in the database. More recently, the authors improved their system by considering spatial relationships between strokes [69]. However, their approach still has two drawbacks. First, they use a small number of basic shapes (circle, line and polygon) to classify strokes. Second, this method is difficult to scale up to large databases, since it too relies on exhaustive search.

Funkhouser et al [41] describe a method for retrieving 3D shapes using textual key-

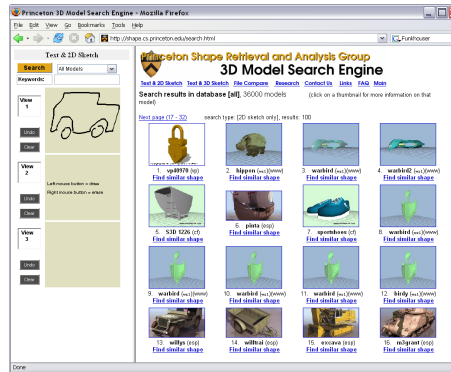


Figure 2.4: Screen-shot of Princeton Search Engine for 3D Models.

words, 2D sketched contours, 3D sketches or a combination of both. Based on this work they developed the Princeton Search Engine for 3D Models. Figure 2.4 depicts a screen-shot of a query and respective results. The authors developed a new 3D shape descriptor based on spherical harmonics that is descriptive, concise, efficient to compute and invariant to rotations. Additionally, they found that queries combining both text and shape produce better results than either one alone and that users prefer to specify queries in 2D than in 3D. The Princeton Search Engine relies on silhouettes and their fitting to projections of 3D images to retrieve models from large databases. Although this method works well when searching for a shape by its contour, it does not provide a functional solution to allow partial matches based on embedded shapes.

Bajavista is an application to index and retrieve clip-art drawings by content, using hand-sketched queries [33, 36, 35]. To that end it uses the framework for content-based retrieval from large sets of drawings developed by Fonseca [34]. This framework provides efficient indexing and matching of drawings based on their geometry and topology. Therefore, to classify existing clip-arts, Bajavista extracts geometric and topological information from them, after applying a set of simplification heuristics. The extracted information is then converted into descriptors and stored in a database. When a query is submitted to Bajavista, topology and geometry descriptors are computed and used to search the database for similar drawings. A screen-shot of Bajavista prototype is depicted in Figure 2.5, which illustrates a query (center window) and results (bottom images).

As we have seen above, the majority of the existing content-based retrieval systems focus on raster image classification and retrieval. Moreover, most published techniques

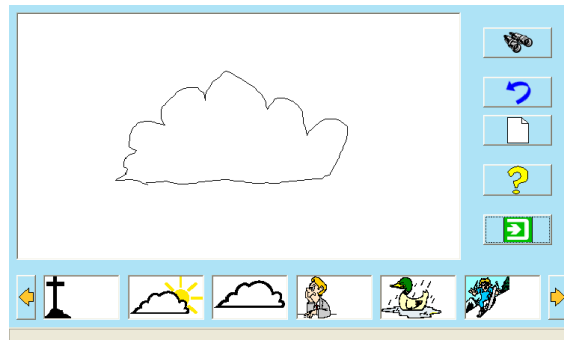


Figure 2.5: Screen-shot of Bajavista.

for retrieving vector drawings show two major drawbacks. First, they are evaluated on databases with few elements. Second, drawings stored in the database tend to be simple elements, not representative of real vector drawings. One noteworthy, the Princeton Search Engine uses large databases with complex models. However, as the authors stated, good results are produced mainly when queries combine text and shape descriptions. Another exception is the Bajavista system, which produce good results when querying large databases of clip-arts using hand-made sketches. The success of this system lies, in part, on the framework for content-based retrieval proposed by Fonseca [34]. Fonseca's framework could be considered as an improvement to Berchtold [8] and Park [89] systems. It focuses on retrieving vector drawings by privileging spatial relationships among dominant shapes and their geometry. Moreover, Fonseca's framework provides fast and efficient algorithms to perform similarity matching of sketched queries in large databases of drawings. This makes it a good foundation for for retrieval of engineering drawings proposed in this dissertation. By adopting this framework, I was able to focus my research on feature extraction from drawings and sketches.

Furthermore, Funkhouser et al [41] found out that users prefer sketching two dimensional queries, even if they are looking for 3d models. This is very important, since it supports the paradigm for query specification proposed in this dissertation.

2.2 Working with digitized drawings

During task analysis we have interviewed draftspeople from the mould industry. In these interviews one user referred that it will be convenient to use existing drawings, in

paper format to specify the queries. When we posed this idea to other users, all agree that it will be quite useful. One effective method to implement such solution is by digitizing the existing drawing using a scanner and then use the resulting image to specify the query.

However, proposed approach is based on vector information. Thus, to use digitized drawings as queries it is necessary to convert them into vector format. Raster to vector conversion, *i.e.* vectorization, of images is widely used, yet several distinct approaches exist with their own pros and cons. In next section I review the most common vectorization techniques.

2.3 Vectorization of engineering drawings

When classifying a drawing based on a printed version, we first convert the hard-copy to digital format. The quality of such digital images may vary drastically depending of the characteristics of all parts of the sensor chain and illumination technology [21]. Therefore, the process of converting those raster images to vector format will be preceded by a chain of steps to enhance image quality and reduce existing noise. That way the vectorization algorithms do not need to deal with discontinuities or extra pixels in original images, since these problems have already been solved during the pre-processing stage.

Image vectorization techniques have been deeply studied during the last decades. Most widely known solutions are more than thirty years old and still being used in several applications or have served as a base to developing improved techniques.

One of the oldest vectorization methods is the Hough Transform [95], which is still very useful for identifying clusters of points defining shapes that can be expressed parametrically, as line segments defined by their slope and zero-axis intersection. Many distinct solutions to this problem have been proposed since. Most of these methods rely on the medial axis of shapes to perform vectorization and they differ mainly in the way medial axes are determined. The other major difference is the line tracking technique, *i.e.* how points that compose the medial axis are grouped in a chain of points to determine each vector.

In this section I survey existing vectorization methods and present them according to medial axis determination methods, after a classification suggested in [109], dividing the

vectorization methods into six classes: Hough transform, thinning, contour, run-graph, mesh pattern and sparse pixels grouping.

2.3.1 Hough Transform

Theoretically, the Hough Transform can be used to detect any type of shape, regardless of its complexity. However, such general solution is not feasible since it would entail significant computation time and memory requirements. In practice, this method is used mainly to detect straight lines or curves in drawings. Still, this might be enough to process many types of engineering drawings.

I start by describing the vectorization of straight line images and then generalize to other parametric shapes. Dori [26] discusses detecting lines in binary images by transforming extended patterns into spatially compact features in parametric space. This space is often called the Hough domain, but is also referred to as the Hough space, Hough transform plane or accumulator array. By transforming a raster image to the Hough domain it is possible to avoid the difficult global detection problem and replace it with a more easily solved local peak detection problem. A line in the binary image is thus represented by a single peak in the Hough domain. This peak has coordinate values in the two parameters that describe the line, slope and intercept. When an image is composed by multiple lines the resulting Hough space will show multiple peaks each corresponding to a different line. To convert the binary image data into Hough domain it is necessary to construct lines from sets of multiple unrelated points. Straight lines are defined by the equation $y = mx + c$. Thus every line in the (x, y) plane corresponds to a point in the (m, c) plane. Each point in image plane can have an infinite number of lines that pass through it. The slope and intercepts of these infinite (x, y) plane lines correspond to a line on the (m, c) plane described by equation $c = -mx + y$.

To allow practical implementation it is necessary to reduce the number of lines that pass through each point. To that end, the Hough space is discretized to the desired accuracy by dividing the (m, c) plane into rectangular "bins". These bins accumulate for each black pixel in the (x, y) plane the slopes and intercepts of all possible lines that can pass through it. That is, if the image contains only one line, this will map to the same coordinates in Hough space for all pixels on that line. Thus, the value of a bin in the

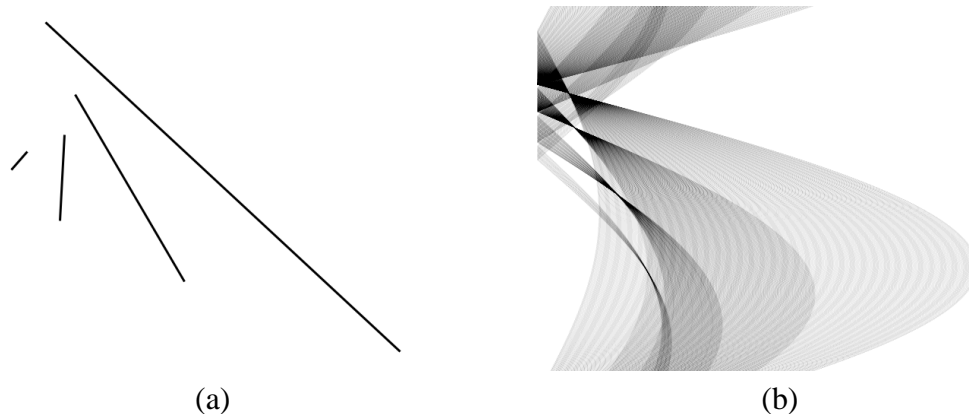


Figure 2.6: Example of Hough transform using trigonometric parameters: (a) Image with four lines; (b) corresponding Hough space. Each peak corresponds to a line. The dominant peak (the topmost) represents the longer line.

Hough space will be proportional to the length of the corresponding line. After visiting all black pixels of the binary image, it is possible to find the lines by detecting peaks on Hough domain. However, because of image noise and sampling inaccuracy it is possible that a line will be mapped into more than a single "bin" in the Hough space. If the number of existing lines is known *a priori* it is possible to limit the number of peaks to detect. On the other hand, if we just want to find significant (long) lines we can set a threshold that represents the minimum number of pixels a line must have.

Peak detection in Hough domain yields lines of infinite length instead of line segments. Furthermore, a set of collinear line segments might yield a peak similar to the one produced by a longer single line. Hence, line segments can be found by examining the overlap of an Hough-determined line to the points in the original image. A slightly different approach is referred by Seul *et al.* in [100]. They suggest parameterizing each line trigonometrically. Thus, a straight line is defined by equation $x \sin \theta + y \cos \theta = r$ and each line in the (x, y) plane corresponds to a single point in (r, θ) plane, which is considered the Hough domain. Apart from this small difference the procedure is exactly the same as before.

To detect circles in binary images the method is identical to the one used for straight line vectorization. The sought shape is parameterized by equation $(x - a)^2 + (y - b)^2 = r$, where a and b define the center of the circle and r its radius. The Hough domain becomes a three-dimensional space defined by the parameters (a, b, r) . After all black pixels in image space have been mapped to all possible points in Hough space, each peak in that

space indicates a circle with parameters equal to the coordinates of that location. Peak detection on three-dimensional Hough space makes this method a much more computationally intensive task than the detection of straight lines. Thus, the Hough transform is not a good option for detecting geometric shapes other than line segments.

Since the parameters in the Hough Transform are sampled sparsely, their precision is not too high. Hence, the shapes yielded by this technique might not be accurate. This problem cannot be eliminated, but can be minimized by using a finer grid, at the cost of increased computation time and memory.

2.3.2 Thinning based methods

Scanning an engineering drawing yields a raster image. In order to extract features from that drawing, it is necessary to convert the scanned image into vector format. One way to do that is by computing one-pixel wide skeletons of shapes and then converting them to lines using a line tracking algorithm. To calculate the skeleton of objects in raster images is generally used a technique called *thinning*. Dinnen [23] found, in the mid 1950's, that an averaging operation over a square window in a binary image with a high threshold yields a thinning of the shapes existent in original image. Since then, hundreds of articles have been published on thinning methods and their application. Among these articles, we can find several good surveys on thinning techniques [104, 20, 54, 102, 67]. Today, thinning is commonly used in a wide range of applications, such as data compression, extraction of critical features, pattern recognition or raster-to-vector conversion. In this document I am interested in studying thinning-based vectorization methods.

Since earlier vectorization systems ([31, 66, 82]), thinning methods are used as a first step of the vectorization process. These systems apply thinning methods to a binary image in order to determine one-pixel-wide medial axes of regions in the image. This skeletonized representation is then submitted to a line tracking process. This thinning process, also called skeletonisation or medial axis transformation, applies morphological transformations [47, 48] to regions on a binary-valued image to compute their one-pixel-wide skeleton. Although the thinning operation can be applied to regions of any shape, it is most suitable for elongated regions, where an obvious medial axis exists. In blob like shapes the thinning process hardly produces a correct skeleton.

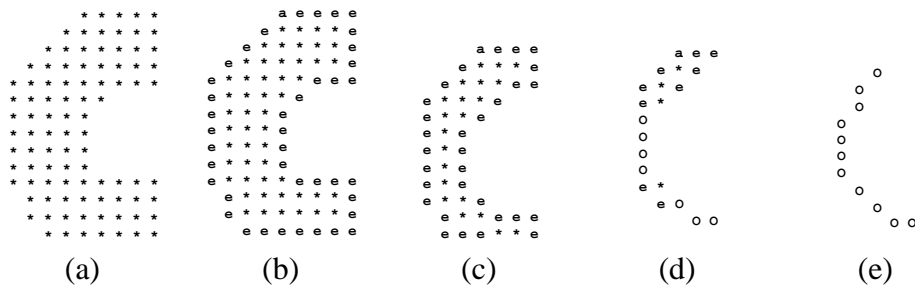


Figure 2.7: Peeling iterations: (a) original image; (b), (c), (d) intermediate results after first, second and third iterations; (e) skeleton.

Skeletons are approximations of the medial axis of the region boundary, as defined by Montanari [76], Pfaltz and Rosenfeld [92] or Davies and Plummer [20]. Based on these distinct definitions of skeleton the thinning algorithms were classified in three groups: iterative boundary erosion, distance transform and adequate skeleton.

The iterative boundary erosion, also referred as peeling, peels the region boundaries, iteratively removing one layer at a time and stopping when the region have been reduced to a one-pixel-wide skeleton. An effective way to perform this iterative thinning is by moving a square window over the image and applying a set of rules to decide wich pixels should be marked for erasure after each iteration. Hilditch [53] proposes the use of a 3×3 window and a basic set of rules to mark its center. These are described in detail by Naccache and Shinghal in [81]. The major drawbacks of iterative thinning methods are their time complexity and shape distortions at junctions that commonly occur. To reduce the impact of these problems new variations of the basic thinning algorithm were devised. For instance, Deutsch [22] uses non-square windows while O’Gorman [84] generalizes the method for $n \times n$ sized windows. Although the speed and accuracy of thinning process have improved significantly since their early implementations, the problems referred above still persist.

A distinct approach to thinning was presented by Pfaltz and Rosenfeld [91, 97], using its formal definition of skeleton described in [92]. The distance transform thinning replaces each pixel in a shape by a number indicating its minimum distance to the pixels outside that shape, as depicted in Figure 2.8(b). In the presented approach this distance is defined as the length of a 4-connected chain between the two points. After computing the distance transform they apply a local maximum determination procedure to construct the skeleton of each shape in the image. Generally, distance transform algorithms are much

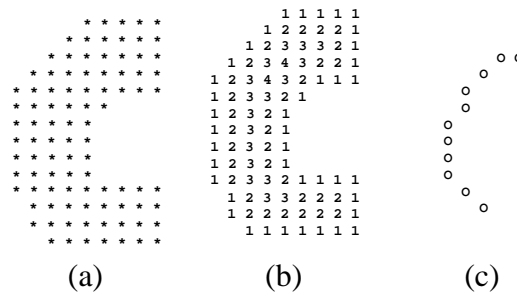


Figure 2.8: Thinning using distance transform: (a) original image; (b) distance transform; (c) skeleton.

faster than the iterative algorithms described before. However, there are no warranties that the distance transform will yield correct results. Often, the resulting skeletons are not connected, especially at junctions.

Davies and Plummer [20] defined an "adequate skeleton" and suggested a third approach for thinning, which combines skeleton points obtained by iterative and distance transform methods. This yields more accurate skeletons than iterative thinning algorithms, at the cost of more computing resources.

Unlike lines computed by the Hough Transform method, skeletons produced by thinning algorithms are still in bitmap format. To convert these to vectors is necessary to apply a line-tracking process to skeletal points. The more common line-tracking methods use the Freeman Chain Code [40]. A widely used extension of this approach, the Primitive Chain Code (PCC), was proposed by O’Gorman [86]. [100] provides a practical description of these chain coding methods.

2.3.3 Contour based approach

Contour based approaches seem to be faster than other vectorization methods. Unlike the thinning algorithms, which perform line-tracking after detecting the skeleton, contour based methods first perform a contour-tracing followed by a skeleton sampling process. This contour-tracing procedure is the most computationally intensive part of the contour-based algorithms. Since this is a common operation in computer vision, many methods have been proposed in literature. Most popular approaches to contour-tracing are described in the books of Haralick and Shapiro [48], Nalwa [83] and Seul *et al.* [100].

Jimenez and Navalon [63] suggested a skeleton sampling algorithm that considers

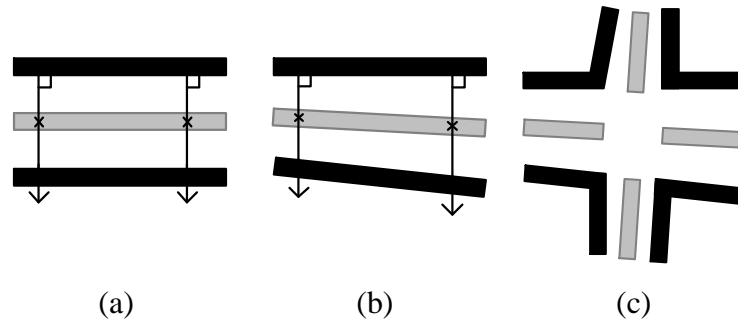


Figure 2.9: Example of skeleton sampling: (a) parallel edges; (b) almost parallel edges; (c) cross-junction.

the skeletal points as the midpoints of the perpendiculars projected from one side of the contour to the other. This algorithm behaves well with elongated shapes with parallel or almost parallel edges, *i.e.* line objects, as depicted in Figure 2.9 (a) and (b). However, it fails to detect small angle merging junctions and misjudges cross intersections, as illustrated by Figure 2.9 (c).

A similar approach for skeleton sampling was described by Shapiro *et al.* [101]. They suggest that midpoints between each pair of pixels on opposite edges of the contour be considered as a point on the skeleton. Therefore, edges must be followed in such a manner as to minimize the width of the object. As in the algorithm presented by Jimenez and Navalon [63], the approach described by Shapiro *et al.* [101] does not handle junctions perfectly.

Due to these unsolved problems in vectorizing shapes with junctions, contour-based approaches are generally unsuitable for vectorizing engineering drawings.

2.3.4 Run-graph based methods

Di Zenzo and Morelli [111] presented a graph-based approach to image representation. Their method, represents the binary image of a drawing as a list of vertical or horizontal "runs". These "runs" are maximal sequences of black pixels along vertical columns or horizontal rows. They construct a graph based on these detected runs and according to their classification:

A run adjacent to only one run in each side is a segment portion run.

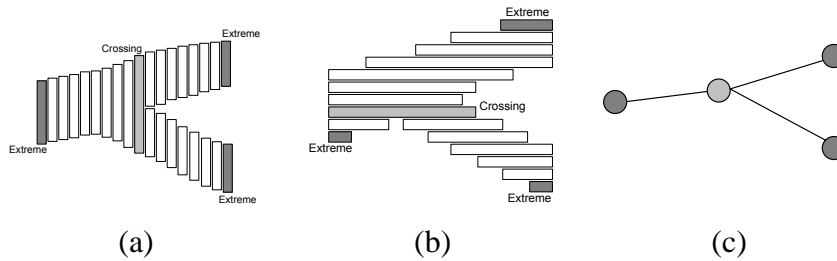


Figure 2.10: Simplified example of run graph representation: (a) vertical partitioning; (b) horizontal partitioning; (c) computed graph.

A run adjacent to more than one run on a side is a crossing point.

A run not adjacent to any run on one side is at the extreme of a segment.

A formal definition of these simple rules yields a procedure that computes a graph whose nodes derive from crossing and extreme runs and edges consist of sets of adjacent line runs. Figure 2.10 depicts this vertical (a) and horizontal (b) partitioning of a simple image. Any of these partitioning results in the graph pictured in Figure 2.10 (c). This graph represents the vectorized image, with each edge corresponding to a line segment. A improved version of this technique generalizes crossing points and extreme points as crossing areas and extreme areas. These generalized notions allow extending nodes to include runs that would otherwise belong to edges, so that the lengths of runs in an edge are approximately uniform.

More recent methods to compute run graph representations have been published, but all of them derive from the ones described before. For instance, mixed graph representations were introduced to solve the problem of associating an edge with a line segment that forms a narrow angle with the direction of runs, as in Figure 2.10 (b). In this variation, edges consist of either horizontal or vertical runs, according to the slope of the corresponding line portion, while nodes consists of vertical runs and sub-runs (parts of a run).

Boatto *et al.* [9] described a line drawing interpretation algorithm based on graphs. Their procedure takes as input the run-graph of an image and produces the corresponding skeleton, from the midpoints of the runs in edge areas.

One of the major disadvantages of run-graph based methods is the lack of accuracy in determining the junction points. Also these methods can introduce false junctions

caused by changes in run direction. Finally, these methods cannot handle curve segments correctly, since they were conceived to detect straight lines.

2.3.5 Mesh pattern based methods

Lin *et al.* [71] introduced mesh patterns to improve vectorization. They suggest dividing the image by using a specific mesh. Then they detect characteristic square patterns by analyzing the borders of each mesh unit and create a control map for the image. This control map is composed of meshes labeled according to a pattern database. Their technique detects straight lines by analyzing this control map.

Vaxivière and Tombre [108] present an extension to Lin's mesh pattern approach, which uses dynamic meshes to handle more complicated drawings. In this method, mesh units that cannot be characterized are divided into smaller known pattern meshes, whose shape may be non-square.

Mesh pattern approaches are relatively fast. This is because they sample the images sparsely and only analyze the pixels on mesh borders. As a result their running time increases linearly with image resolution. However, the ideal mesh size for each image is hard to control. Moreover, those methods fail to correctly detect complex lines such as arcs or dashed lines.

2.3.6 Sparse pixel based approaches

A different approach to vectorization that also samples the image sparsely was developed by Dori *et al.* [12, 27, 26]. Their Orthogonal Zig-Zag (OZZ) algorithm tracks the course of a one pixel wide "beam of light" which turns at right angles each time it hits the edge of an area covered by black pixels. Lines are detected by joining the midpoints of each run, as depicted in Figure 2.11. This procedure uses two passes to improve accuracy. In the first pass, it starts beams horizontally. On the second pass, it starts beams vertically. The lines detected on both passes are then combined in a single description.

The OZZ algorithm is a time-efficient method to detect straight lines due to the sparse sampling of images. However, it is difficult to correctly detect arcs with OZZ which is also highly sensible to noise. More recently, Liu and Dori [28] developed the Sparse

engineering drawings containing arcs or other curved shapes.

Finally, the SPV is a fast method that both respects line geometry, preserves line width and handles junctions and intersections accurately. Therefore, this method is better suited to perform vectorization of engineering drawings, as well as the iterative thinning approaches.

2.4 Curve Simplification

Engineering drawings in vector format are stored using a small set of primitives. These primitives range from lines and splines to more complex geometric shapes, such as polygons and arcs. In my approach I choose to convert all these shapes to polygonized curves (a chain of line segments, also called "polyline"). While this increases the number of entities in the drawing it allow me to use a uniform simplification algorithm since it only needs to handle lines.

Curve simplification, must try to reduce the number of line segments in a polyline preserving its caricature. There are three main approaches to curve simplification. The simpler takes a polygonized curve with n vertices as input and produces an approximate polyline with m vertices as output ($m \ll n$). Algorithms based on this approach are referred as *bounded-#*. Other approaches also take a polygonized curve with m vertices to yield an approximate output within a specified error ϵ tolerance. These are called *bounded- ϵ* methods. A third approach approximates the polygonized curve without any prespecified restriction. Implementations of this approach depend on the context of application and are mostly used on cartography or character outline detection [56]. Therefore, it will not be discussed further on this document.

One of the most trivial methods to simplify polylines is the regular sub-sampling algorithm [73] known as the "nth-point algorithm" in cartography. This method produces an output line using every n th point of the input and discarding the rest. The major drawback of this method is that simplification may remove critical points such as edge corners, as depicted in Figure 2.12. In this example the simplified output (right) is a straight line, which is a poor approximation of the original curve. Another disadvantage of the regular sub-sampling is that it generates unnecessary vertices on straight lines, that could

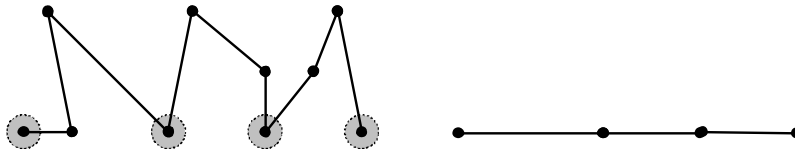


Figure 2.12: Example of regular sub-sampling using $n = 3$. Original curve (left) and its caricature (right).

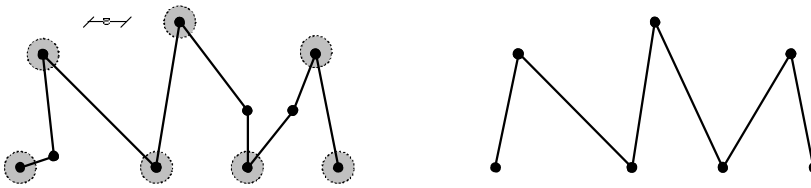


Figure 2.13: Example of curve simplification using Douglas-Peucker. Original curve (left) and its caricature (right).

be represented with only two endpoints. However, when the original polygonized curves are smooth this algorithm might produce satisfactory results. Regular sub-sampling is a very fast and simple technique that can produce, under certain conditions, good approximations. Unfortunately the result of the simplification of generic curves may be a poor caricature of the original [19].

Probably, the most widely used [50] method for curve simplification is the Douglas-Peucker algorithm [29]. This well-known technique recursively selects coordinates that fall outside a predefined bandwidth. At each step, the Douglas-Peucker algorithm attempts to approximate a sequence of points by a line segment. The farthest point from the line segment is found and if its distance to the line segment falls below a pre-defined threshold ϵ , the approximation is accepted. Otherwise the algorithm is recursively applied to the two sub-sequences before and after the farthest point. Figure 2.13 depicts the simplification of a polyline with Douglas-Peucker method. Although this technique may not be optimal, it generally yields the highest quality approximations when compared with many other heuristic approaches. According to published results [110], Douglas-Peucker Algorithm was the best at choosing critical points and generalizations produced by this algorithm were overwhelmingly deemed the best perceptual representations of the original curves. Additionally, the Douglas-Peucker method is easy to implement and the hierarchical structure of simplified polylines can be used on scale-independent carto-

graphic databases [65, 18] as well as in image processing or computational geometry. The time complexity of Douglas-Peucker algorithm depends greatly on the input curve. While $\Omega(n)$ complexity can be achieved in a best case with n vertices, the worst case running time of this algorithm is quadratic.

Hershberger and Snoeyink [51, 52] proposed an approach based on *path hulls* to speed up the Douglas-Peucker algorithm. This method uses the geometric structure of the problem to attain a worst case time-complexity of $O(n \log n)$. Since the splitting vertices of a polygonized curve must lie on its convex hull, Hershberger and Snoeyink use the *path hull data structure* to maintain a dynamic convex hull of the polygonal chain. Two convex hulls from the middle of the chain outward are computed using Melkman's hull algorithm [75]. The farthest vertex from a line is found by locating two extreme points in each hull using binary search. After splitting the curve on the farthest vertex, one of the hull computations are undone to obtain the hull from the "middle" to the farthest point and recursively approximate the sub-chain containing the middle. Then another two convex hulls for the remaining sub-chain are built and the same method is applied to find its caricature. In their paper, Hershberger and Snoeyink shown that the path hull algorithm can be between two and three times slower than the standard Douglas-Peucker method in best case situations due to extra structural information that it maintains, but it is far faster in both worst case and generic situations. However its implementation is more complex than the standard implementation of the Douglas-Peucker algorithm.

Instead of splitting the polygonized curve recursively, Ballard and Brown [4] suggest splitting at the point of highest error along the whole curve, on each iteration. This method produces highest quality results in detriment of some processing time. Additionally, if the yielded subdivision tree is saved, it allows building an approximation for any larger error tolerance very quickly [18].

Recently, Jaafar [60] presented a new approach to curve simplification using a least squares method with double tolerance (LS:DT). This technique uses the Douglas-Peucker method to identify a chain of "anchor points" that preserves the line caricature. Then, they construct a least squares line that passes trough a set of points computed from the anchor points. Using a shift tolerance it is possible to adjust these lines to enhance the generalization effects. Since least square lines are not linked together, it is necessary to establish common intersection points, which must be joined to create the simplified poly-

line. This method can minimize distortion with respect to original curve, while preserving its caricature.

Additionally to the heuristic methods referred above, algorithms for optimal approximation of polygonized curves yield the best solution to this problem. Unfortunately, these algorithms are slower than their heuristic counterparts and are usually quite complex, making them hard to implement. An optimal algorithm that performs an exhaustive search would have exponential cost. However, using dynamic programming and taking advantage of geometric properties it is possible to yield an optimal solution in polynomial time. By introducing a slight variation in error metric, Imai and Iri [59] presented an L_∞ -optimal¹ approach that yields optimal solutions in $O(n^2 \log n)$ using shortest-path graph algorithms or convex hulls.

2.4.1 Discussion on Curve Simplification

Regular sub-sampling has some restricted applications, when the input characteristics are known and fulfill certain requirements or when it is not necessary to yield a precise approximation. However it is the fastest simplification algorithm available.

The major drawback of the Douglas-Peucker algorithm and its variants is that simplification of a simple polygon can yield a self-intersecting polygonized curve. Although this problem seems quite serious, the simplicity, performance and effectiveness of the Douglas-Peucker method make up for its drawbacks. That way it is not hard to understand why this algorithm remains the most widely used curve simplification technique more than three decades after its publication.

The other existing solutions for curve simplification can produce better (even optimal) approximations and/or be faster. However the complexity of implementation prevents their use, except when Douglas-Peucker algorithms are not fast enough or when better approximations are needed.

¹To estimate the error in approximation of the initial curve, several distances from a point to a line can be applied: the Euclidean distance (L_2); the Manhattan distance (L_1); the Max distance (L_∞); the Vertical distance; the Bounding Shell distance. Furthermore, Perez and Vidal [90] proposed an incremental technique for error measurement that uses the Euclidean and Vertical distances measured to the line support of the segments that allows optimal simplification algorithms to attain faster results. For more details on approximation error measurement see [43].

2.5 Summary

This chapter presented a comprehensive survey on drawing retrieval systems, focusing on those that use sketches to specify queries. Additionally, I surveyed the most used vectorization methods in order to select the one that best suits my needs. Finally, I presented a survey on line simplification methods, identifying advantages and disadvantages of each method. The next chapter introduces my approach to retrieval of technical drawings using both images and sketches.

3

Sketch and Image Based Retrieval of Engineering Drawings

The majority of content based retrieval systems were developed to classify and search for digital images. These approaches use mainly color, texture and shape information to describe the content of pictures. However, engineering drawings, are usually colorless and existing textures are rarely relevant for drawing classification. Thus, a completely different set of features must be used to describe its content. As presented in previous chapter, some work has been published on classification and retrieval of vector drawings. However, existing content based retrieval of vector drawings methods rely exclusively on sketches, images or vector information to specify the query [8, 89, 69, 34].

My approach is based on the framework for retrieval of vector drawings proposed by Fonseca [34]. It uses spatial relationships and geometric information to classify drawings and queries and provides an effective and accurate indexing and retrieval process to search for vector drawings in large databases. Using Fonseca's framework I developed a novel approach that allows not only the use of sketches to specify queries and images to query by example but also mixing both, providing users with a powerful new way to search for vector drawings. Furthermore, I proposed methods for automatic simplification and classification of existing drawings and queries.

In this chapter I describe in detail my approach for content based retrieval of engineering drawings. Initially, I present an overview of the approach, which includes some use-case scenarios and a description of proposed architecture. Then, I describe the image vectorization along with drawing simplification and classification algorithms. Finally, I give a brief description of the indexing, query and matching mechanism.

3.1 Approach Overview

In this dissertation I present a novel approach that allows mixing images and sketches to specify queries when searching for vector drawings. It provides a powerful new way to define queries for content-based retrieval of vector drawings. It is now possible to take advantage of existing hard-copies of drawings and simultaneously explore users' visual memory and ability at sketching. Besides using only hand-sketched queries or digitized drawings to query by example, my approach allow users to add new elements to vectorized drawings by sketching new shapes or delete existing entities using simple gestures. This way, they can start with a digitized drawing and then apply editing commands to refine it.

Images and sketches are integrated by first applying a vectorization process, converting raster drawings into vector figures. To that end I selected a set of existing algorithms from computer vision. To achieve a vectorization solution which is usable interactively, performance was privileged in detriment of accuracy. Indeed, the selected algorithms were tuned to minimize user intervention during vectorization, assuring an almost automatic process, described in Section 3.2. Then users can edit the vectorized image using sketches. This way they can refine queries, select relevant parts, delete unnecessary shapes or add new visual elements.

To classify existing drawings it is necessary to extract from them visual and relationship features¹. To that end, I developed a feature extraction process capable of identify relevant features from drawings in short time. This process simplifies the drawings, in order to remove irrelevant shapes, and then, based on extracted features, it computes a geometry matrix and a topology graph for each drawing. These are then passed to Fonseca's framework, which computes the correspondent descriptor and inserts them in two different indexing structures, one for topological information and another for geometric information. The retrieval process extracts the same geometry and topology descriptors from the query that the classification process extracts from a drawing. Thus, feature extraction and descriptor computation are basically the same in retrieval and classification. However, while the later descriptors are inserted in the database, during retrieval the descriptors are used to execute a search in the database.

¹As suggested by Fonseca [34], visual features encode shape geometry while relationship features describe topological relationships among shapes on drawing.

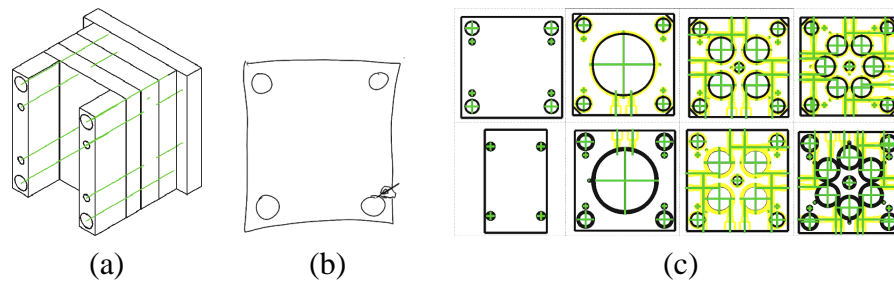


Figure 3.1: Searching a mould component using a sketched-query.

3.1.1 Use Scenarios

The proposed Sketch and Image Based Retrieval (SIBR) system can be used in a diversity of scenarios involving draftspeople, designers, CAD operators or even salespeople or storekeepers. While the first three usually use the system while creating or editing an engineering drawing for searching for other drawings or components they want to reuse, the others use the system to search for specific parts. Next, I will present distinct scenarios, focusing on two possible utilizations of my retrieval system.

3.1.1.1 Scenario A: Creating a Mould

While drawing a mould (Figure 3.1 (a)), Mary, a CAD operator, needs to include a rarely used component. Despite her large experience in this kind of task, she only knows in memory the references of the most used components. Instead of browsing in the maze of menus and options provided by the component library, Mary decides to use the SIBR system. Thus, she sketches a rough approximation of a 2D view of the ought component (Figure 3.1 (b)). The system yields a set of results (Figure 3.1 (c)) based on that query, but none of them is the desired component.

Since the ought component is not within the returned results, Mary refines the initial sketch by adding more detail (Figure 3.2 (a)). Now, the desired component is in the results yielded by the SIBR. Mary selects it (Figure 3.2 (b)) to open it with the CAD tool she is using. After confirming that that component was the one she was looking for, Mary integrates it in the mould drawing (Figures 3.2 (a) and (b)).

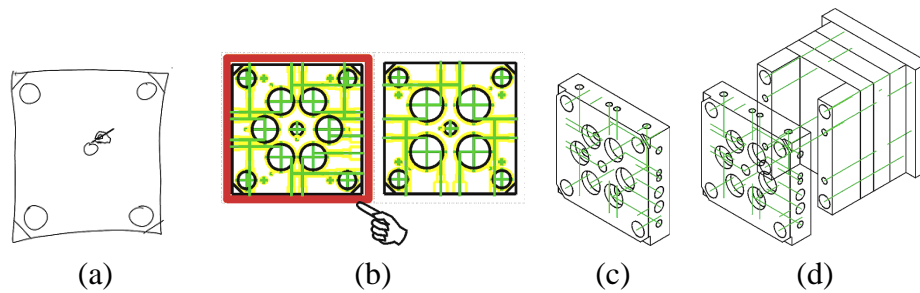


Figure 3.2: Retrieving a mould component and integrating it into a drawing.

3.1.1.2 Scenario B: Looking for a Part

John, a maintenance technician, needs a mechanical part in order to fix a broken equipment. He rushes to the huge store of the supplier and asks Jack, the storekeeper, for that part, giving him the part reference number. Unfortunately, that part is out of stock and it will not be delivered within the next weeks. However, it is possible that similar parts might solve the problem, but neither Jack have sufficient know-how to point a equivalent part neither John knows by memory references of equivalent parts.

Fortunately, John has the engineering drawing of the part he is looking for. Therefore, Jack can use the SIBR to search for similar parts. To that end, he digitizes the drawing using a flatbed scanner and submits it as a query to to SIBR system. Within a couple of minutes the system yields a set of similar drawings. By consulting the technical specifications of parts associated with each drawing, provided by the manufacturer software, John finds an equivalent part and asks Jack to give him that one.

3.1.2 System Architecture

The proposed scheme for content-based retrieval of vector drawings through images and hand-sketched queries supplies a mechanism to retrieve vector drawings, in electronic format, taking advantage of users' natural ability at sketching and existing paper drawings. Based on the framework presented by Fonseca [34], the architecture of this system for content-based retrieval, presented in Figure 3.3, is divided in two modules:

- **Classification** module, which analyzes existing drawings and maps their features onto numeric descriptors, storing them in a database.

- **Retrieval** module, which compares queries to a database of drawing descriptors to produce a set of candidate results. The vectorization of digitized drawings is also performed by this component.

In the *Classification* module, the *Feature Extraction* component yields a geometry matrix and a topology graph for each drawing. This geometrical and topological information are passed to the *Descriptor Computation* component, which creates feature vectors that are then inserted into a multidimensional indexing structure. The *Retrieval* module comprises four different components. The *Image Vectorization* component converts digitized drawings to vector format, by applying computer vision algorithms. As in the *Classification* module, the *Feature Extractor* component extracts features from sketches, vectorized drawings or from a combination of both, producing a geometry matrix and a topology graph. Those are then passed to the *Query* component which creates descriptors for this information. Finally, the *Matching* component compares descriptors from the query to those stored in the logical database, yielding a set of similar drawings.

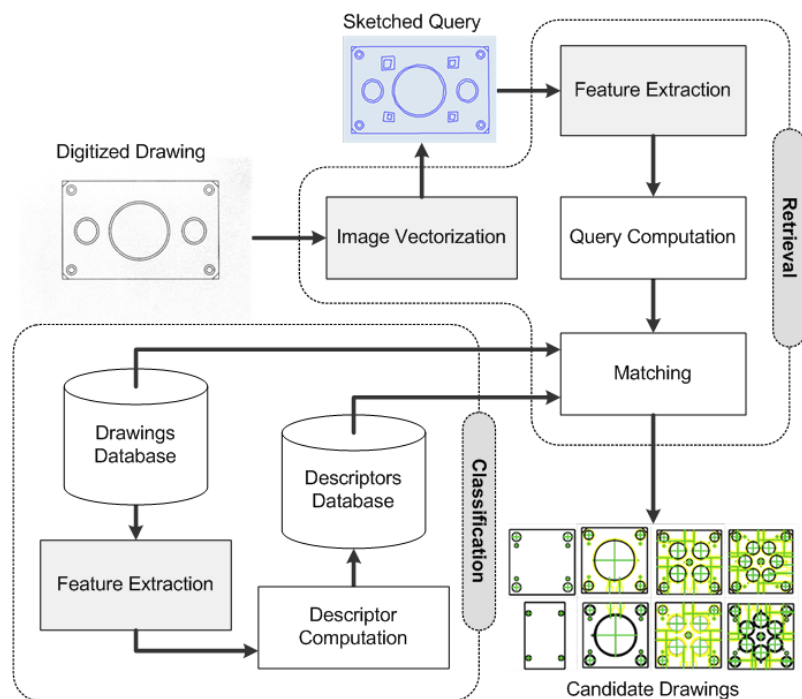


Figure 3.3: System architecture of proposed solution. Feature Extraction and Image Vectorization components are my main contributions, while remaining components are provided by Fonseca’s framework.

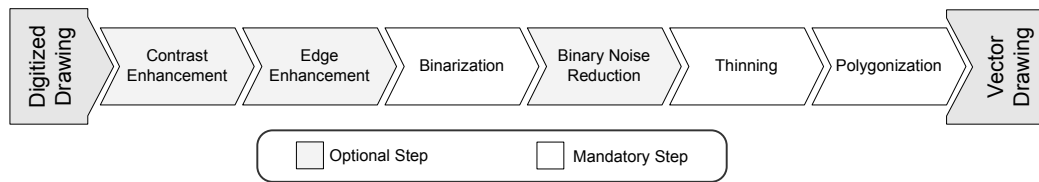


Figure 3.4: Image Processing Steps.

Next section describes in detail the process used by the *Image Vectorization* component and in following section I present the feature extraction process applied in the respective components.

3.2 Image Vectorization

When searching for a drawing or similar drawings based on a printed version, it is first converted to digital format through scanning. The quality of such digital images may vary drastically depending on the characteristics of all parts of the sensor chain and illumination technology [21]. The loss of quality incurred during image acquisition makes the vectorization a complex process. Until now no methods devised can be considered as sufficiently stable and robust to work as stand-alone "black boxes" [106]. According to Karl Tombre et al. [105], one important factor for robustness is to minimize the number of parameters and thresholds needed in the vectorization process. Thus, my approach intends to achieve an acceptable vectorization result with minimal user configuration requirements.

Proposed approach starts by performing contrast and edge enhancement, followed by a thresholding process that yields a binary version of the original image. Then, I apply a noise reduction algorithm followed by a thinning step before converting image to vector format. In this vectorization process I employed well known and widely used algorithms from Computer Vision, some of them are more than thirty years old. Hence, I will avoid an exhaustive description of these algorithms, since those techniques are extensively documented in the literature. The image processing pipeline (depicted Figure 3.4) for vectorization of engineering drawings is composed by a sequential application of the steps referred in the previous paragraph. Depending on image quality, contrast and edge enhancement might be skipped, as well as binary noise reduction.

In my work I assumed that printed drawings have no text or symbols. However, this restriction can be eliminated by applying a text/symbols/graphics segmentation method before polygonization. Ramel and Vincent [96] presented several strategies for localization and recognition of graphical entities in line drawings. Additionally, I also assumed that drawings were digitized to a raster gray-scale format and saved in non-compressed image files. If drawings are digitized in binary format, first steps can be discarded, at the cost of external pre or post processing in order to refine or connect lines after image acquisition.

3.2.1 Contrast Enhancement

Accurate visual interpretation of images depends to a great extent on their visual contrast. To improve the appearance of images and effectiveness of analysis I use histogram transformations, due to its widely proven simplicity and effectiveness. These well known methods enhance image contrast by changing the shape of the histogram and are extensively documented [58, 93, 94, 100]. If quality of original image is acceptable but not enough for an effective vectorization, an histogram expansion is applied. Histogram expansion is a straightforward and conservative linear histogram transformation that increases contrast keeping the histogram shape. To improve effectiveness of this histogram transformation in images with low contrast but with regions of low and high intensities, such as digitized versions of engineering drawings, is introduced a cut-off percentage such that middle intensities will be stretched while intensities on upper and lower tails of histogram will be compressed.

In lower quality images, I apply an histogram transformation by specifying the desired shape of the output histogram. In my approach I considered three possible slope values (m) for output histogram: a positive slope with value $m = 1$, that enhances contrast in low intensities; a negative slope with value $m = -1$, that enhances contrast in high intensities; or a zero slope ($m = 0$), corresponding to histogram equalization. Depending on brightness of original image, one of these slopes values is used in the histogram transformation.

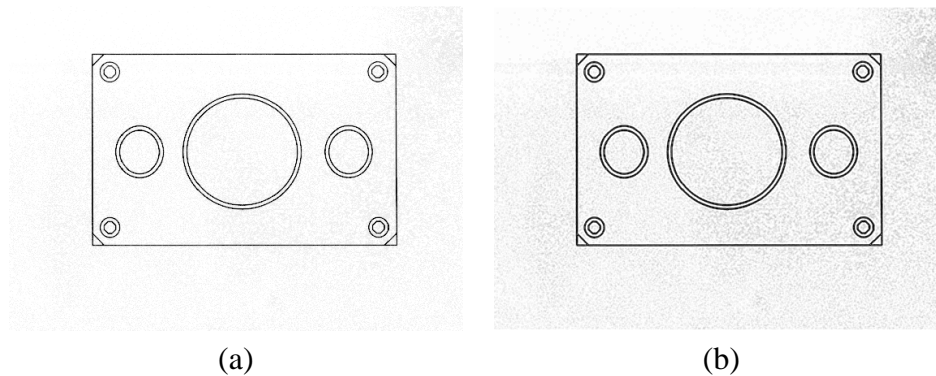


Figure 3.5: Original gray-scale image (a) and yielded result after contrast and edge enhancement (b).

3.2.2 Edge Enhancement

The main goal of the algorithms presented in this thesis is to detect lines from original images in digital format. Therefore, it is important to enhance images in order to improve the effectiveness of line detection when their quality is not good enough, as depicted in Figure 3.5. Unlike the global image enhancement provided by histogram transformations presented previously, the edge enhancement methods performs local analysis to give a crisper appearance to image, enhancing edges and local features. Edges in an image are denoted by abrupt local variations in intensity, which implies high spatial frequencies. Thus, contrast enhancement can be achieved through amplification of these high frequency components relative to background and other slowly varying intensity features.

One method to sharpen edges is simple to diminish the slowly varying intensity features. unsharp masking performs this by applying a low-pass filter and then subtract the filtered image from the original. The unsharp masking procedure is based on the assumption that edges to enhance lay on a flat (constant) background. However, this is not the most common situation, since digitized versions of engineering drawings frequently present non-uniform backgrounds. To overcome this problem, Seul *et al.*[100] suggests high-pass filtering with a pre-defined mask. These masks should have coefficients that sum to unity in order to ensure that the average intensity of the original image remains unaltered. Since engineering drawings are contains mainly lines over a distinct background, the most suitable mask is based on an operator known as the Laplacian, which is a difference operator commonly used for detecting edges in an image. Thus, I suggest using the Laplacian operator Δ as a convolution filter.

$$\Delta i(x, y) = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Unfortunately this sharpening produces, as a side effect, an enhancement of background noise. Hence, it is necessary to perform a noise reduction operation as a prior step. A common approach to generic noise reduction on gray-scale images is smoothing. To that end, convolution of the image is performed with a low-pass filter. In this technique each pixel is compared to its neighbors and if their intensity differ sharply, this difference may be reduced by adjusting the pixel's intensity.

The simplest filter used for smoothing is the uniform filter. With this filter, sharp features are smoothed through removal of intensity disparities between neighboring pixels. However, uniform filters has two major drawbacks. Both noise and small important features can be equally affected when they exhibit similar degrees of acuity. Furthermore, some ringing distortion can be introduced due to ghost edges. Using Gaussian filters for smoothing avoids major problems of uniform filters, namely by reducing the ringing distortion. Nonetheless, the degree of smoothing with a uniform filter is superior than with a gaussian filter with the same length. Thus, a larger filter should be used to achieve the same smoothing. Moreover, while the uniform filter only requires addition, using a Gaussian filter implies multiplication during convolution. Therefore, to obtain the same smoothing effect with a Gaussian filter is required higher computation time than with a uniform filter. Since engineering drawings contain a large number of lines, the most appropriate technique for noise removal on gray-scale version of digitized drawing is through smoothing with a Gaussian filter. Hence, before high-pass filtering, I perform convolution with the 3×3 Gaussian filter h_G , which yields pretty acceptable results in a reasonable amount of time.

$$h_G = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Despite the small amount of smooth produced by this filter, when compared with a similar uniform filter, the risk of image corruption due to ringing is highly reduced. Moreover, the computation time with digitized images used for retrieval of engineering drawings is similar to other steps in the vectorization pipeline.

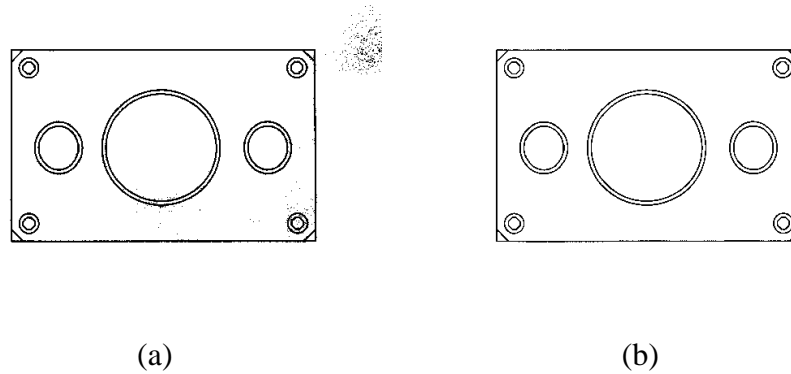


Figure 3.6: Binary Image obtained using fixed threshold (a) and region averaging (b).

3.2.3 Binarization

Shading and textures are not relevant to the presented work, since the vectorization process aims line detection on digitized engineering drawings. Therefore gray-scale images can be converted to binary images. These are composed by pixels that just admit two possible intensities, usually referred as "on" and "off" or "black" and "white". This seemingly simple conversion process is really a complex problem, since even gray-scale images that apparently contains only black lines on a white background have many more than only two intensity values, often covering almost all range of possible intensities. Therefore it is necessary to identify the range of intensities that must be considered "black" and which are "white". To that end, a threshold level is defined. The threshold level represents a intensity value above which pixels are considered "on" and below are considered "off". Thus, choice of threshold level is the most important task to perform in binarization . However, determination of this value might be difficult, especially if contrast are poor or image have a nonuniform background.

The binarization, also called thresholding, could be achieved trough several different methods. These are usually grouped in two main categories: global techniques and locally adaptive techniques. Global techniques determine a single threshold level for the image and perform conversion using that value for all pixels, ignoring their context. On the other hand, locally adaptive techniques use context information to adapt the threshold for each pixel according to its neighbors.

When images have uniform background the global techniques are usually more efficient, since there are no advantage in using the more complex locally adaptive techniques.

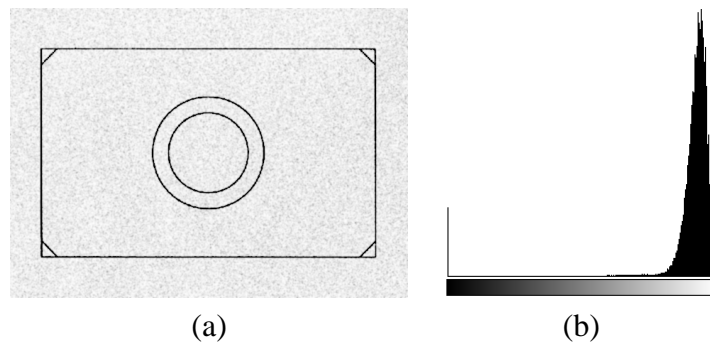


Figure 3.7: Image of a digitized engineering drawing (a) and its intensity histogram (b). At right of the histogram the peak is formed the background while the small peak at left corresponds to foreground. Threshold value should be just at left of the larger peak.

In some cases the results might even be better because threshold selection is based on overall image and not in a smaller set of pixels. However if original paper drawing quality is low or if illumination during capture were nonuniform only locally adaptive techniques can achieve satisfactory results. A detailed evaluation of locally adaptive binarization methods for gray-scale images with low contrast, variable background intensity and noise is presented by Trier and Taxt in [107].

Since in this thesis I assumed that images are acquired using a flatbed scanner that provides controlled lightning conditions, the best results are achieved, when the original paper drawing has good quality, by using global techniques. However poor quality originals, usually caused by aging or mishandling, may require locally adaptive techniques. Therefore I suggest that the choice of binarization method must be done during image processing, by selecting between a fixed thresholding technique, a global technique based on histogram shape or a locally adaptive region averaging technique.

Fixed thresholding is the most straightforward global technique. It uses a selected threshold value to perform the binarization of a gray-scale image. However, histogram shape can be used to determine the threshold value automatically. The global technique based on the histogram shape, analyzes the intensity histogram in order to find a value between the peak formed by background, dominant in engineering drawings, and the peak originated by the foreground (see Figure 3.7).

Region averaging is a widely used locally adaptive technique. Theoretically, it subtracts a nonuniform background from the original image and performs thresholding on the uniform result. In practice, it works by first calculating a running region average, by

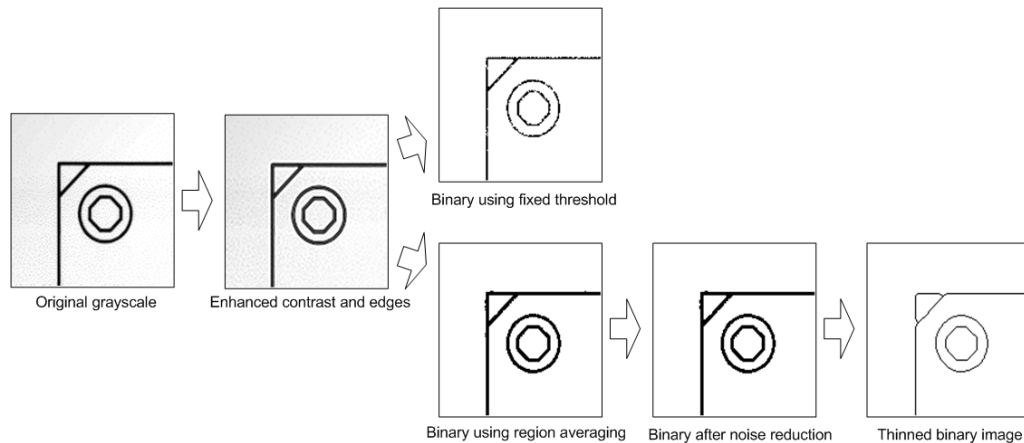


Figure 3.8: Evolution of image detail during processing.

comparing each pixel to its local average and setting it to "ON" or "OFF" accordingly. The effectiveness of region averaging technique depends on the size of the region used for calculating the local averages. This value must be adjusted to reflect the expected size of foreground features. Since I am interested in lines, this region should be relatively small. Best results were obtained with a region size $k = \frac{1}{50}l_d$, where l_d is the length of the image diagonal.

In proposed solution, the choice between global and locally adaptive techniques is made by the user during the vectorization process. This implies that users must be able to identify, by looking at the digitized image, which is the best technique. However, it must always be possible to use the trial and error approach to choose the more effective method, since results of each image processing step should be displayed and last action could be quickly undone in order to perform a different one. Figure 3.8 illustrates a vectorization of a drawing, depicting the evolution of part of the drawing. In this case the user has an original image in grayscale format, performed both contrast and edge enhancement steps and then executed a binarization using fixed threshold. However, the yielded binary image has low quality. Therefore, the user went back one step and tried a different binarization method, which produced a better binary version of digitized drawing. Then, user applied the noise reduction and thinning steps, obtaining a good final result.

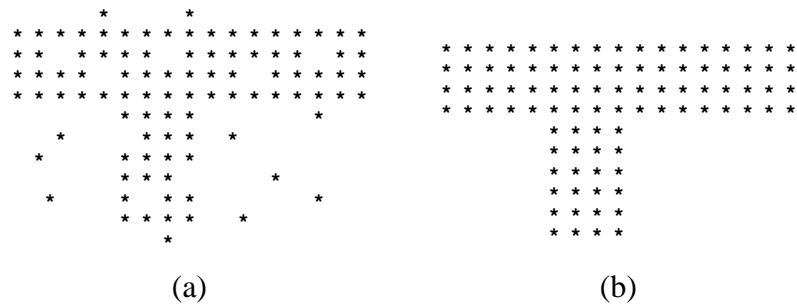


Figure 3.9: Pictorial example of speckle noise reduction: original image (a) underwent several iterations of the noise reduction algorithm based on the kFill filter, which yielded the final image (b).

3.2.4 Binary Noise Reduction

Unappropriate storage, mishandling and aging of paper drawings can seriously degrade the quality of the original document. Additionally, the acquisition process can introduce noise on the captured images. Although image processing steps performed earlier attenuated some noise on image, salt-and-pepper noise (also called speckle noise) is usually present on image yielded by binarization.

Speckle noise removal on binary images improves greatly the reliability and robustness of thinning and line detection algorithms (see Figure 3.9). However, complete noise removal is hard to achieve in real time applications. In proposed approach I aim to perform just an efficient noise reduction that executes in a short period of time. To reduce speckle noise in binary images the most common method consists in removing only single pixel noise. To that end, is used a straightforward technique which applies a 3×3 mask to each pixel, comparing it with its eight neighbors and changing its state accordingly. Unfortunately, this simple technique fails on removing larger noise features and even single pixel noise in the boundary of regions might persist.

Although more complex, the binary noise reduction algorithm based on kFill filter, proposed by O’Gorman [85], is able to reduce both isolated and border noise up to a selected size. Additionally, it allows control of corner rounding and length shortening effects. Furthermore, this algorithm enables the connectivity persistence, by providing the possibility of ensuring that filling operation does not join two disconnected regions neither separate connected regions.

Therefore, by considering characteristics of engineering drawings, the algorithm proposed by O’Gorman seems appropriate for binary noise removal during vectorization. Configured to retain 90° corners and keep the connectivity, this algorithm performs an efficient binary noise reduction on digitized technical drawings.

3.2.5 Thinning

Since purpose of proposed vectorization process is to find a sketchable version of original drawing and I am concerned mainly with topology and geometry, the thickness of lines that composes the original drawing is not important. I am only interested in its one pixel wide skeleton. These skeleton represents an approximation to medial axis of lines in original drawing and is used by the line tracking algorithm to construct the vector version of drawing. In proposed approach I consider that drawings does not contain filled shapes, only lines and arcs. However the presented work can be extended to consider such elements by using a segmentation method that separates the filled shapes from lines and then performing contour detection on the filled elements instead of thinning.

Lam et al made comprehensive reviews [67, 68] of several skeletonisation methods and conclude that there are no general thinning algorithm. In fact, to obtain the best results, advantages and drawbacks of distinct thinning techniques must be compared to select the more appropriate method for each situation.

The algorithm proposed by Hu and Li [57] is able to determine the intersection pixel with precision, but fails on maintaining line connectivity, which make it unappropriate for my intents. Jansen and Vossepoel [61] proposes a method based on iterating a sequence of vectorization and morphological operations until a fitting criteria is met, but since I do not aim to achieve a precise version of original image such complexity is dispensable.

The thinning algorithm proposed by O’Gorman [84] is both simple and effective in computing the skeleton of elongated segments, such as line segments or arcs. The major drawback of this method is the lack of precision when determining line crossings and corners of thick lines. However, such precision is not relevant in the present vectorization process. Thus, due to its simplicity and efficiency in thinning lines and arcs, the O’Gorman algorithm were chosen to implement the thinning step in the proposed vectorization pipeline. An overview of thinning techniques was presented in Section 2.3.2.

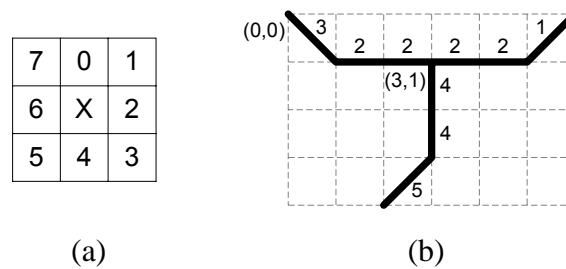


Figure 3.10: Freeman chain coding: chain direction codes (a), and an example of line structure with starting coordinates and direction codes. The resulting Freeman code for this image is: $(0, 0)3, 2, 2, 2, 2, 1, 5(3, 1)4, 4, 5, 1$

3.2.6 Polygonization

Since my classification mechanism is based on shape geometry and spacial relationships, some loss of detail during this process is acceptable and does not greatly affect the query results. This is crucial when deciding which vectorization method to use. Therefore, I use an adaptation of the primitives chain-coding method proposed by O’Gorman [86] followed by a polygonization procedure.

The thinning process yields a line image, which is composed by single-pixel width lines. These lines are indeed chains of ON pixels that are connected from one pixel to neighboring pixels. To transform these lines from pixel space to its concise representation chain coding techniques are commonly used. These methods provides a lossless transformation, preserving all topological and morphological information.

Freeman chain coding [40] is a popular and effective method for analyzing simple line images. This algorithm searches the image in raster order for the first ON pixel and stores its coordinate location. Then its eight-connected neighbors (Figure 3.10 (a)) are examined and the direction of the first ON neighbor is stored. Next the neighbors of the later pixel are analyzed in the same way, coding the chain iteratively until no more connected ON pixels exist. When each pixel coordinates are stored it is erased (set to OFF). This process repeats through the last ON pixel of the image. However, branches and crossings are not considered by the Freeman chain coding. Thus, composite lines joined at junctions in thinned image are represented as separate lines in the result produced by this algorithm, as depicted in Figure 3.10 (b)).

O’Gorman proposed [86] an extension for the Freeman chain code. It was designed to

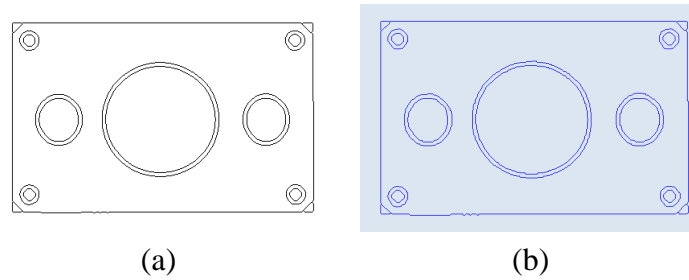


Figure 3.11: Thinned binary image (a) and vector version of drawing (b).

preserve information on branching and junction topology. The Primitives Chain Coding (PCC) presented by O’Gorman, introduces codewords to represent not only the connection direction but also other line features such as start and end points of lines, bifurcations and cross junctions. Likewise previous method, to generate a PCC representation of a line image a raster-order search is performed until an ON pixel is found and then its location is stored. Additionally, a PCC code for a start or break feature is also stored. The PCC chain coding starts from this point by analyzing on each iteration sets of up to three successive pixels which are PCC coded and then erased. Any features encountered are coded, branches are followed to their ends and loops are followed to their initial coding location. This coding process ends when the last pixel of the line structure is coded and erased. Then the raster search continues from the starting position of the previous line structure, repeating the coding process for any ON pixel found in the image.

I adapted this well-known primitive chain coding method in order to construct sets of linked points instead of codes, thus identifying branches and intersections. Each of these sets yields an approximation to a smooth curve, which can be represented by a set of connected line segments. To compute this set I used a polygonization procedure based on the Douglas-Peucker algorithm [29], that reduces the number of points within a given threshold, as described in Section 2.4. Consequently, the presented method produces an approximate vector version of the original drawing composed by a set of line segments, as depicted in Figure 3.2.6.

3.3 Feature Extraction

Content-based retrieval of pictorial data, such as digital images, drawings or graphics, uses features extracted from the corresponding picture. Typically, two kinds of features

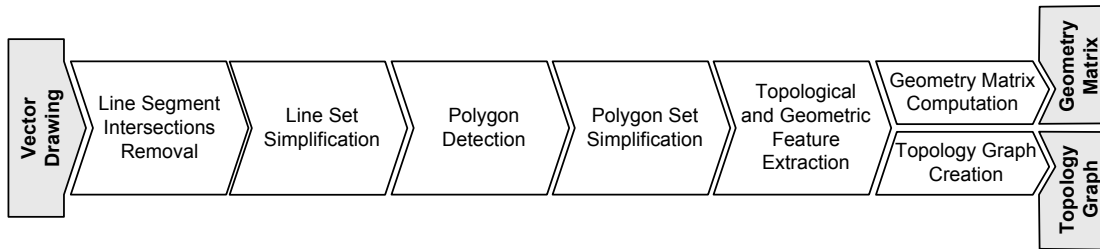


Figure 3.12: Feature extraction block diagram decomposition.

are used; visual features (such as color, texture and shape) and relationship features (topological and spatial relationships among objects in a picture). However, in the context of this work, engineering drawings, color and texture are irrelevant features and only topological relationships and geometric features are considered to make presented approach less restrictive.

Proposed feature extraction process drawings through a set of stages until they are mapped into a geometry matrix and a topology graph. These are then submitted to Fonseca's framework in order to produce two feature vectors, from which are computed a set of descriptors using spectral information. These descriptors may be inserted on the main indexing structure during classification or used to query the database when searching for similar drawings. Figure 3.12 depicts the block decomposition of the proposed feature extraction process. First, I apply a line segment intersection removal to the drawing, which is considered as a set of line segments. Then, I apply a line simplification step, followed by a polygon detection and another simplification step. Finally, topological and geometric features are extracted from simplified set of polygons. These features are then converted into a topology graph and a geometry matrix, respectively.

3.3.1 Simplification

The success of my approach content-based retrieval depends in great extend of the quality of the extracted features and these depends on accurate shape detection algorithms and efficient simplification methods. Regarding drawing simplification, the presented solution includes two distinct steps. The two simplification steps eliminate useless shapes from drawing before extracting its features. Most technical drawings contain detailed descriptions of objects, which are not necessary for a visual search and increase the cost

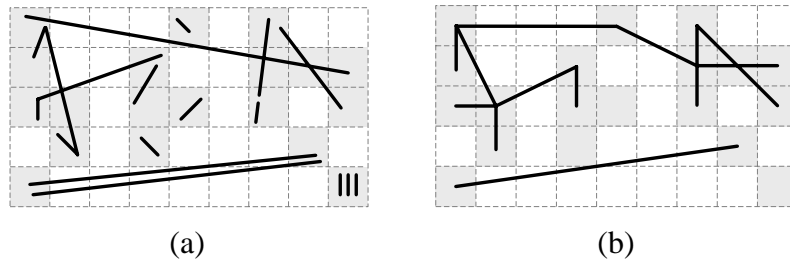


Figure 3.13: Arrangement of segments before (a) and after (b) traditional snap rounding.

of searching. Thus, I try to remove visual details (i.e. small-scale features) while retaining the perceptually dominant elements and shapes in a drawing. To that end, I first simplify vector information, producing a set of lines. After detecting polygons from these set of lines, I simplify the set of polygons produced by the detection algorithm.

3.3.1.1 Line Set Simplification

To simplify the initial set of lines I first apply a snap rounding algorithm. This is a well known method that creates fixed-precision sets of line segments from arbitrary precision vectors. In my approach I use this method not only to ensure a finite-precision approximation to the original drawing, but also to produce a simplified version, where small features are discarded.

The snap rounding method tiles the plane where the line segments lay with a grid of unit squares. These squares are usually called *pixels*. A pixel is considered *hot* if it contains one or more vertices of lines. After all line segments has been visited and all hot pixels has been found, each vertex of the arrangement is replaced by the center of the hot pixel containing it. Then each edge is replaced by a polygonal curve with vertices on the center of the hot pixels crossed by the original edge. Figure 3.13 depicts an example of traditional snap rounding on a arrangement of lines.

Goodrich *et al.* [42] presented a deterministic plane-sweep algorithm that performs a robust snap rounding approximation of a set of line segments. They also give a simpler randomized incremental construction with running time equivalent to the deterministic version. The method presented by Goodrich *et al.* preserves the more important topology of lines and runs in sub-linear time. Unfortunately, even the randomized algorithm is not easy to understand and implement.

On the other hand, the traditional snap rounding procedures will produce arrangements of lines with its vertices well separated but the edges might be very close to non incident vertices, as depicted in Figure 3.14. In this example, vertices of segments s and t in original set of lines are replaced by the center of the hot pixels that contains them, producing s' and t' respectively. In the rounded segments the distance between the rightmost vertex of segment s' and the segment t' are extremely small. However, with traditional snap rounding no further simplification will be made.

To overcome this problem, Halperin and Packer [46] proposed an augmented procedure, the iterated snap rounding, which rounds a set of lines such that each vertex is at least half the width of a pixel away of any non-incident edge. Despite possible degradation of quality of the approximation, this algorithm also preserves the topology of original arrangement of lines and the authors presented a straightforward implementation of the proposed method.

In Figure 3.15 is illustrated an example of two different rounded arrangements of a single set of line segments. The one produced by a traditional snap rounding algorithm leaves all lines disconnected, while the segments in the arrangement yielded by the iterated snap rounding are connected. Considering that my algorithm uses closed shapes (polygons) to classify drawings and sketched queries, it is obvious that, for my purposes, the iterated snap rounding methods produce more suitable results.

Therefore, I decided to use the iterative snap rounding method presented by Halperin and Packer to achieve an effective line set simplification, since there are need high quality on approximation in my approach, but only geometry and topology preservation, which is clearly accomplished by this method. Although iterated snap rounding algorithm was

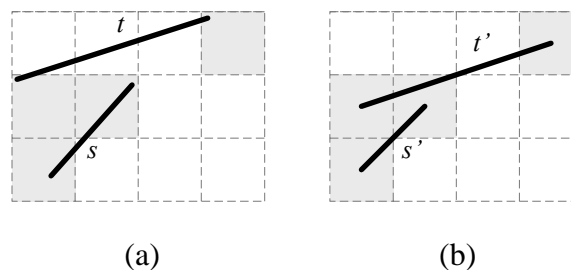


Figure 3.14: Arrangement of segments with a vertex very close to a non-incident line after (b) snap rounding.

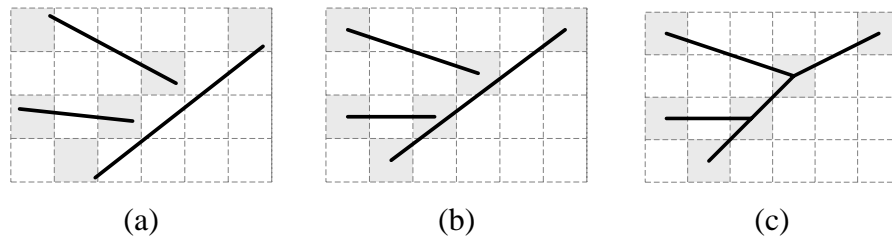


Figure 3.15: Results produced by Snap Rounding (b) and Iterated Snap Rounding (c) when applied to a set of line segments (a).

conceived to convert arbitrary precision sets of line segments into fixed precision representations, it fulfills my needs for initial line set simplification of vector drawings. My main concern, the topology and geometry of relevant shapes are preserved by this algorithm. Moreover, most irrelevant shapes are discarded by it. The resulting set of polygonal curves are then converted in a set of line segments that will be used as input to the polygon detection algorithm.

3.3.1.2 Polygon Simplification

The novel polygon detection algorithm described in Section 3.3.2 is used, in proposed approach, to identify shapes on drawings. This algorithm yields a set of minimal polygons. It contains the polygons with minimal number of edges that can be constructed from a given set of lines. For instance, when the polygon detection algorithm takes the simple set of lines depicted in Figure 3.16 (a) as input it will yield a set of two minimal polygons, one with four edges and other with six edges, as shown in Figure 3.16 (b). However, the detection algorithm considers the intersection points as vertices of the polygon even if edges are collinear. Thus, the detected rectangle is indeed a polygon with six edges, but the two horizontal edges on top are collinear, as well as the two vertical edges on left.

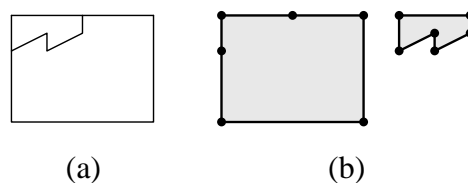


Figure 3.16: Example of polygon detection on a simple drawing: (a) original set of lines and (b) detected polygons with vertices marked.

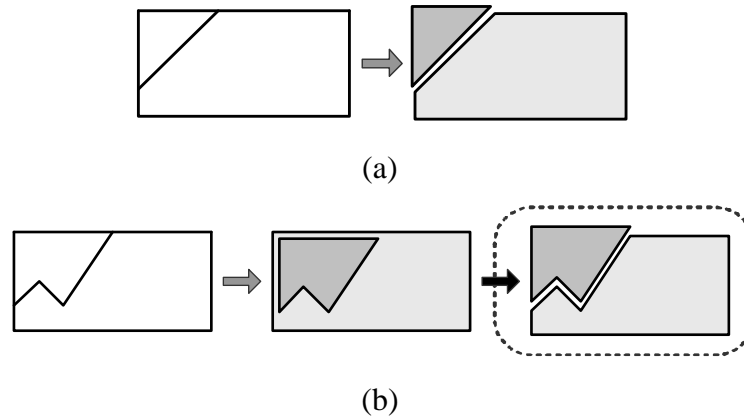


Figure 3.17: Example of polygon detection and coherence simplification on a simple drawing: (a) detection algorithm yields desired result and (b) an heuristic simplification must be applied to set of detected polygons.

Notwithstanding, this kind of results does not fulfill my needs, since drawings must be classified according to topology and this way two similar drawings can have different topologies just because one of a different number of edges. An example of such problem is depicted in Figure 3.17. Drawings represented by line sets depicted in (a) and (b) have similar topology. However, polygon detection algorithm yields sets of polygons with completely different topology. In the first one there are two adjacent shapes, while in the second there is one polygon including the other.

To overcome the topology incoherence in results yielded by polygon detection algorithm, I developed a coherence simplification heuristic that privileges adjacency between polygons, by avoiding inclusion of adjacent shapes. Therefore, when one polygon is included in another and shares some edges with it I transform these polygons in two adjacent polygons, as show in Figure 3.17 (b).

I developed a simple algorithm, INCLUSION-SIMPLIFICATION, that performs this coherence simplification in polynomial time. This algorithm compares all polygons and when two of them shares at least on edge it checks if one of them are included in the other. When one polygon includes the other the algorithm performs two boolean operations in order to determine the polygons: intersection and exclusive-or.

```

INCLUSION-SIMPLIFICATION( $\Theta$ )
1  for each  $P$  in  $\Theta$ 
2  do for each  $O$  in  $\Theta$ 
3      do if  $P \neq O$ 
4          then REMOVE-INCLUSION( $P, O, \Theta$ )
5  return  $\Theta$ 

```

```

REMOVE-INCLUSION( $P, O, \Theta$ )
1  if ADJACENT( $P, O$ )
2      then if ( $P \subseteq O$  OR  $O \subseteq P$ )
3          then
4              Remove  $P$  from  $\Theta$ 
5              Remove  $O$  from  $\Theta$ 
6              append ( $P \cap O$ ) to  $\Theta$ 
7              append ( $P \mathbf{XOR} O$ ) to  $\Theta$ 
8          return return

```

The INCLUSION-SIMPLIFICATION algorithm changes the set of polygons Θ in order to remove all inclusions of polygons that share at least one edge. This algorithm sweeps all polygons invoking for each one the REMOVE-INCLUSION procedure. This procedure compares each polygon with all other checking for adjacency. If an adjacent polygon is found it is necessary to see if it included or includes the given polygons. When it happens both polygons are removed from the polygon set and the polygons produced by the intersection and exclusive-or operations are appended to it.

The two boolean operation over polygons play an important role in the INCLUSION-SIMPLIFICATION algorithm. After the detection of a pair of polygons to simplify it is necessary to perform boolean operations on that pair in order to compute the disjoint polygons. Figure 3.18 depicts the result of applying an intersection (b) and an exclusive-or (c) to a simple set of two polygons (a), in which the darker polygon includes the lighter one. In this particular case the exclusive-or operation yields two distinct polygons. Thus, simplification algorithm will produce the three polygons depicted in (b) and (c).

Before removing the undesired inclusions using the method described above, I must discard the small polygons which are irrelevant for classification in order to simplify the drawing and speed up the whole process by reducing the number of polygons on drawing.

To discard irrelevant small polygons I developed a small polygon removal heuristic, which detects small polygons on drawings and removes or merges them according to their

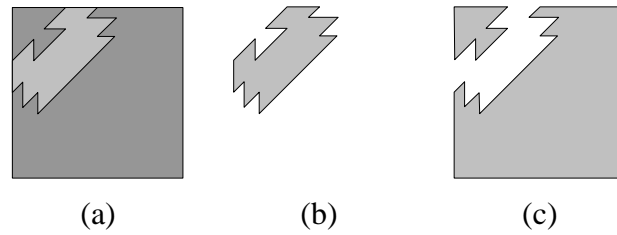


Figure 3.18: Example of boolean operations over polygons: (a) original set of two polygons, (b) intersection result and (c) exclusive-or result.

adjacency relationships with others.

Initially, a threshold value (ϵ) must be specified to determine which polygons are considered small. In my approach, this value is relative to the size of the Axis-Aligned Bounding Box (AABB) of the whole drawing. Threshold value is $\epsilon = (1/n) \times l$, where l is the length of the larger edge of the AABB and n determines the level of simplification desired ($n \geq 2$). High values of n results in almost no simplification and are useful if the goal is to remove only very small details. On the other hand, if $n \leq 5$ the number of removed polygons will be large and only the dominant shapes will be preserved. With ϵ specified it is possible to identify small polygons on drawing. To avoid complex calculus I do not use the polygon P itself but its Oriented Bounding Box (OBB),referred as OBB_P , when searching for small polygons. A polygon P is considered small if the area of OBB_P is smaller than ϵ^2 or if length of shorter edge of OBB_P is less than ϵ . These conditions are formalized in procedure IS-POLYGON-SMALL, which receives a polygon P , computes its oriented bounding box (OBB_P) and returns *true* if P could be considered small or *false* otherwise.

```

IS-POLYGON-SMALL( $P$ )
1   $OBB_P = \text{ORIENTED-BOUNDING-BOX}(P)$ 
2  if AREA( $OBB_P$ ) <  $\epsilon^2$ 
3    then return true
4  if SHORTER-EDGE( $OBB_P$ ) <  $\epsilon$ 
5    then return true
6  return false

```

After all polygons have been checked against the conditions listed above, is created a set of small polygons (SSP), containing the polygons that satisfy any of these conditions. The detection of small polygons is performed by the DETECT-SMALL-POLYGONS algorithm which yields the SSP .

```

DETECT-SMALL-POLYGONS( $\Theta$ )
1   $SSP =$  empty set for each  $P$  in  $\Theta$ 
2  do if IS-POLYGON-SMALL( $P$ )
3      then Add  $P$  to  $SSP$ 
4
5  return  $SSP$ 

```

Polygons on SSP are then analyzed to see which of them must be removed. This is done in three sequential steps. The first compares each polygon with all others on SSP within this set, producing a set of adjacent polygons (SAP_{SSP}) and the joins adjacent polygons into only one.

```

MERGE-SMALL-POLYGONS( $\Theta$ )
1   $SSP =$  DETECT-SMALL-POLYGONS( $\Theta$ )
2  for each  $P$  in  $SSP$ 
3  do
4       $SAP_{SSP} =$  ADJACENT-POLYGONS( $P, SSP$ )
5      for each  $P'$  in  $SAP_{SSP}$ 
6      do
7           $P =$  MERGE( $P, P'$ )
8          Remove  $P'$  from  $SSP$ 
9
10 return  $SSP$ 

```

Next, polygons that no longer are considered small are removed from SSP and the remaining polygons on SSP are compared against all others to check for adjacency. When a small polygon is adjacent to a larger one, they are merged and it is removed immediately from SSP .

```

MERGE-POLYGONS( $\Theta, SSP$ )
1   $SSP =$  DETECT-SMALL-POLYGONS( $SSP$ )
2  for each  $P$  in  $SSP$ 
3  do
4       $SAP_{\Theta} =$  ADJACENT-POLYGONS( $P, \Theta$ )
5      for each  $P'$  in  $SAP_{\Theta}$ 
6      do
7           $P =$  MERGE( $P', P$ )
8          Remove  $P$  from  $SSP$ 
9          Remove  $P$  from  $\Theta$ 
10
11 return  $SSP$ 

```

At the end, all polygons that remains in SSP are simply removed from the original

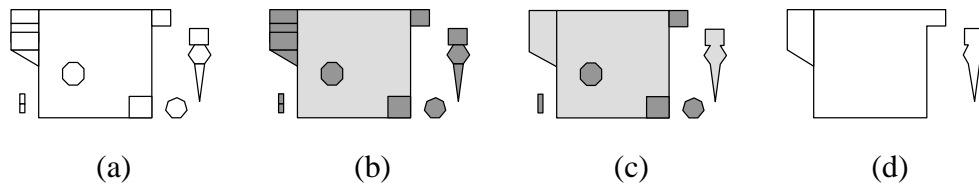


Figure 3.19: Example of small polygon removal: (a) original set of polygons, (b) after small polygon identification, (c) after merging of small polygons and (d) simplified polygons.

set of polygons, Θ . The overall simplification by removal of small polygons is detailed in REMOVE-SMALL-POLYGONS.

```

REMOVE-SMALL-POLYGONS( $\Theta$ )
1   $SSP = \text{MERGE-SMALL-POLYGONS}(\Theta)$ 
2   $SSP = \text{MERGE-POLYGONS}(\Theta, SSP)$ 
3   $\Theta = \Theta \setminus SSP$ 
4  return

```

Figure 3.19 illustrates an example of a small polygon removal on a simple drawing. In Figure 3.19 (a) is depicted the original drawing, composed by a set of polygons yielded by the polygon detection algorithm presented in Section 3.3.2. Classification of polygons according to their size is depicted in Figure 3.19 (b), where the polygons considered small are represented in a darker gray. After merging of small polygons has been completed, the set of small polygons were reduced, as depicted in Figure 3.19 (c), and they will be merged with the larger ones or discarded. Finally, the drawing with three polygons that results from simplification of the original set of fourteen polygons are presented in Figure 3.19 (d). In this example, the two small adjacent rectangles at bottom left are merged into only one but the resulting polygons remains classified as a small polygon. Since it is isolated, the last step of this removal procedure discards this shape, which does not appear in simplified version of drawing. Similarly, the octagon inside the large polygon are also discarded at the end. On the other hand, the small square the larger one is merged with it, because they are adjacent. But, since the large square includes the smaller, the merge of this two shapes yields the larger. Unlike the two small rectangles referred above, the three adjacent small rectangles and one small triangle are merged into a new polygon that is no longer considered small. Thus, it becomes a relevant shape that will not be removed or merged with adjacent large square. Likewise, the square, hexagon and triangle at right, are also merged into only one shape which is maintained since it does not

satisfies any of the conditions necessary to be classified as a small polygon. Finally, the small square that lays outside the larger one, adjacent to it, is merged with it, producing the larger shape that composes the simplified drawing.

3.3.2 Polygon Detection on a Vector Drawing

Unlike image processing, where data consist of raster images, the polygon detection algorithm described in this section deals with drawings in vector format, consisting of line segments. This requires completely different approaches, such as described next.

To perform polygon detection from a set of line segments I divide this task in four major steps. First line segment intersections were detected and removed using the Bentley-Ottmann algorithm [64] and then the resulting set of lines is simplified by applying the algorithm described in Section 3.3.1.1. Next step creates a graph induced by the drawing, where vertices represent endpoints or proper intersection points of line segments and edges represent maximal relatively open subsegments that contain no vertices. The third step finds the Minimum Cycle Basis (MCB) [103] of the graph induced in previous step, using the algorithm proposed by Horton [62]. Last step constructs a set of polygons based on cycles in the previously found MCB. This is straight-forward if I transform each cycle into a polygon, where each vertex in the cycle represents a vertex in the polygon and each edge in the cycle represents an edge in the polygon.

3.3.2.1 Removal of Line Segment Intersections

In a vector drawing composed by a set of line segments there might exist many intersections between these segments. To detect polygonal shapes proper segment intersections must be removed, thus creating a new set of line segments in which any pair of segments share at most one endpoint. Hence, the first step of my approach to detect polygons in vector drawings consists in detecting all M intersections between N line segments in a plane. This is considered one of the fundamental problems of Computational Geometry and it is known that any algorithm, within the model of algebraic decision tree, have a lower bound of $\Omega(N \log N + M)$ time to solve it [6, 14].

In [3] Balaban proposes two algorithms for finding intersecting segments, a deterministic and asymptotically optimal for both time $O(N \log N + M)$ and space $O(N)$ algo-

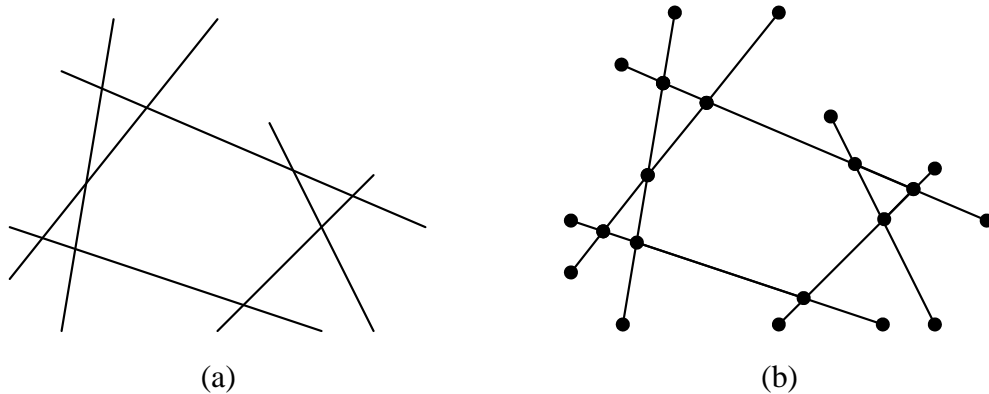


Figure 3.20: Set Φ of line segments (a) and graph G induced by Φ (b).

rithm and a simpler one that can perform the same task in $O(N \log^2 N + M)$ -time. Before that, Chazelle and Edelsbrunner [14] reached a time optimal algorithm $O(N \log N + M)$ with space requirement of $O(N + M)$. The randomized approach devised by Clarkson and Shor [15] produced an algorithm for reporting all intersecting pairs that requires $O(N \log N + M)$ time and $O(N)$ space.

In 1979 Bentley and Ottmann proposed an algorithm that solved this problem in $O((N + M) \log N)$ time and $O(N + M)$ space [64]. This algorithm is the well-known Bentley-Ottmann algorithm and after more than 20 years it is still widely adopted in practical implementations because it is easy to understand and implement [87, 55].

In realizing that this is not the most complex part of my approach, I decided to use the Bentley-Ottmann algorithm, since its time and space complexity is pretty acceptable for my purposes and its published implementations are quite simple.

The next step is to remove all proper intersections between line segments, dividing each intersected segment in sub-segments without proper intersections, only sharing endpoints. In order to find and remove intersections, performing at once the two presented steps, I used a robust and efficient implementation of the Bentley-Ottmann algorithm, described by Bartuschka, Mehlhorn and Naher [5] that computes the planar graph induced by a set of line segments. Their implementation, represented in this document by COMPUTE-INDUCED-GRAPH, computes the graph G induced by set Φ in $O((N + M) \log N)$ time. Since this algorithm is quite long I choose not to present it here. I refer readers to [5] for a detailed description.

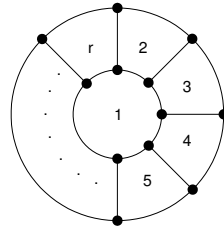


Figure 3.21: A planar graph with a exponential number of cycles

In this implementation the vertices of G represent all endpoints and proper intersection points of line segments in Φ , and the edges of G are the maximal relatively open subsegments of lines in Φ that do not contain any vertex of G . The major drawback of this implementation lies in that parallel edges are produced in the graph for overlapping segments. Since set Φ is a result of the line set simplification process described in Section 3.3.1.1 of it does not contains such segments. Considering, for example, the set Φ shown in Figure 3.20 (a), COMPUTE-INDUCED-GRAPH will produce the graph G , depicted in Figure 3.20 (b), where each edge represents a non-intersecting line segment.

3.3.2.2 All Cycles of a Graph

Detecting polygons is similar to finding cycles on the graph G produced in the previous step. The first known linear-time algorithm for listing all cycles of a graph was presented by Syslo [103]. This algorithm requires $O(V)$ space and $O(V \times C)$ time, where V is the number of vertices and C the number of cycles in G . Later Dogrusöz and Krishnamoorthy proposed a vector space algorithm for enumerating all cycles of a planar graph that runs in $O(V^2 \times C)$ time and $O(V)$ space [25]. Although asymptotically slower, this algorithm is much simpler than Syslo's and is amenable to parallelization.

Unfortunately, the total number of cycles in a planar graph can grow exponentially with the number of vertices [72]. An example of this situation is the graph presented in Figure 3.21. In this case, the number of cycles, including the interior region numbered 1, is $O(2^r)$ with $r = k/2 + 1$, where k is the number of vertices, since one can choose any combination of the remaining regions to define a cycle [25]. This is why it is not very feasible to detect all polygons that can be constructed from a set of lines. In this approach, I choose just to detect the minimal polygons, those that have a minimal number of edges and cannot be constructed by joining other minimal polygons.

3.3.2.3 Minimum Cycle Basis of a Graph

Considering that I just want to detect the minimal polygons this can be treated as searching for a MCB. So, the second step of my approach consists in obtaining a MCB of graph G . A cycle basis is defined as a basis for the cycle space of G which consists entirely of elementary cycles. A cycle is called elementary if it contains no vertex more than once. The dimension of the cycle space is given by the *cyclomatic number*² ($\nu = E - V + P$) [30, 11], where E is the number of edges and V the number of vertices in G and P is the number of connected components of G .

Horton presented the first known polynomial-time algorithm to find the shortest cycle basis of a graph, which runs in $O(E^3V)$ time [62] or in $O(E^4)$ on simple planar graphs [49], which is the case. While asymptotically better solutions have been published in the literature, the Bentley-Ottmann algorithm is both simple and usable for my needs. The pseudo-code of this algorithm is listed in MINIMUM-CYCLE-BASIS and shortly described below. A further detailed description of this algorithm and concepts behind it can be found in [62].

```

MINIMUM-CYCLE-BASIS( $G$ )
1   $\Gamma \leftarrow$  empty set
2   $\Pi \leftarrow$  ALL-PAIRS-SHORTEST-PATHS( $G$ )
3  for each  $v$  in VERTICES( $G$ )
4  do for each  $(x, y)$  in EDGES( $G$ )
5      do if  $\Pi_{x,v} \cap \Pi_{v,y} = \{v\}$ 
6          then  $C \leftarrow \Pi_{x,v} \cup \Pi_{v,y} \cup (x, y)$ 
7              add  $C$  to  $\Gamma$ 
8  ORDER-BY-LENGTH( $\Gamma$ )
9  return SELECT-CYCLES( $\Gamma$ )

```

The ALL-PAIRS-SHORTEST-PATHS finds the shortest paths between all pairs of vertices in graph G and can be performed in $O(V^3)$ time and $O(V^2)$ space using Floyd-Warshall or Dijkstra algorithms [17]. ORDER-BY-LENGTH orders the cycles by ascending length and can be implemented by any efficient sorting algorithm. This is a non-critical step because it has a $O(V\nu \log V)$ upper bound in time complexity, which is insignificant in comparison with other steps of this algorithm.

²The cyclomatic number is the smallest number of graph edges which must be removed from a graph of E edges and V vertices such that no cycle remains.

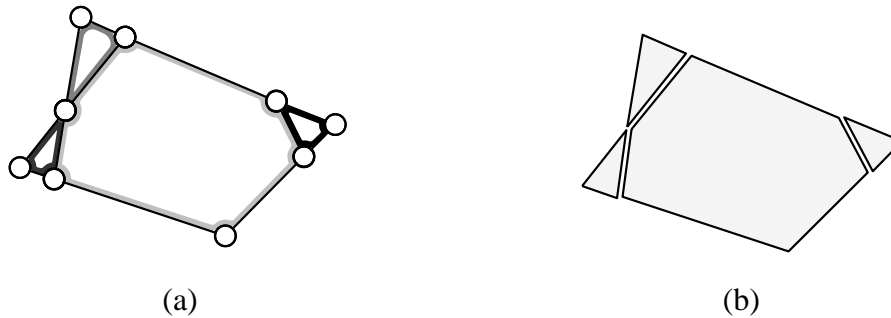


Figure 3.22: Shortest cycle basis Γ of graph G (a) and set Θ , constructed from Γ , of polygons detected in original set of line segments Φ

In **SELECT-CYCLES** I use a greedy algorithm to find the MCB from Γ set of cycles. To do this Horton [62] suggests representing the cycles as rows of a 0-1 incidence matrix, in which columns correspond to the edges of the graph and rows are the incidence vectors of each cycle. Gaussian elimination using elementary row operations over the integers modulo two can then be applied to the incidence matrix, processing each row in turn, in ascending order of the weights of cycles, until enough independent cycles are found. This step dominates the time complexity from other steps, since it takes $O(EV^2V)$ time. Knowing that G is always a simple planar graph I can conclude that as a whole the **MINIMUM-CYCLE-BASIS** algorithm has a worst case upper bound of $O(EV^2V) = O(E^3V) = O(E^4)$ operations and a space requirements of $O(V^2)$. Figure 3.22 shows an example of Γ , the set of cycles resulting from applying the **MINIMUM-CYCLE-BASIS** to graph G shown in Figure 3.20.

3.3.2.4 Polygon Construction

The last step of my approach to detect polygons in vector drawings consists in constructing a set Θ of polygons from the MCB. An algorithm to perform this operation can easily run in $O(CV)$ time, where C is number of cycles in MCB. Such an algorithm is listed in **POLYGONS-FROM-CYCLES** which returns a set Θ of polygons.

```

POLYGONS-FROM-CYCLES( $\Gamma$ )
1  $\Theta \leftarrow$  empty set
2 for each  $C$  in  $\Gamma$ 
3 do  $P \leftarrow$  new polygon
4   for each  $v$  in VERTICES( $V$ )
5     do add vertex  $v$  to  $P$ 
6     add polygon  $P$  to  $\Theta$ 
7 return  $\Theta$ 

```

Figure 3.22 illustrates the resulting set Θ of polygons generated by applying POLYGONS-FROM-CYCLES to Γ depicted in Figure 3.22.

3.3.2.5 Polygon Detection Outline

I can now outline DETECT-POLYGONS. This algorithm is able to detect a set Θ of polygons from a initial set Ψ of line segments. To perform this task I pipelined the algorithms referred previously for line segment intersection removal, MCB finding and cycle-to-polygon conversion. The first step consists in removing from Ψ all segment intersections, inducing a graph G . Next it is necessary to find Γ , a Minimal Cycle Bases of G . Finally, a set of polygons Θ must be constructed from Γ .

```

DETECT-POLYGONS( $\Psi$ )
1  $G \leftarrow$  COMPUTE-INDUCED-GRAPH( $\Psi$ )
2  $\Gamma \leftarrow$  MINIMUM-CYCLE-BASIS( $G$ )
3  $\Theta \leftarrow$  POLYGONS-FROM-CYCLES( $\Gamma$ )
4 return  $\Theta$ 

```

As referred in section 3.3.2.1, COMPUTE-INDUCED-GRAPH runs in $O((N+M) \log N)$ time and $O(N+M)$ space. The SHORTEST-CYCLE-BASIS runs in $O(V^4)$ operations and has a space requirement of $O(V^2)$, making this the critical step in the complexity of this algorithm, since the POLYGONS-FROM-CYCLES just needs $O(CV)$ time.

Since the number V of vertices in the graph is no greater than the sum of line endpoints ($2 \times N$) with detected intersections M , I can then conclude that the proposed algorithm has time and space complexities of $O(V^4) = O((N+M)^4)$ and $O(V^2) = O((N+M)^2)$, respectively.

<i>Lines</i>	<i>Intersections</i>	<i>Nodes</i>	<i>Edges</i>	<i>Time (ms)</i>
6	9	21	24	10
36	16	58	68	50
167	9	169	177	3986
286	47	389	376	8623
518	85	697	679	36703
872	94	1066	10050	128995
2507	10	2407	2526	1333547

Table 3.1: Results of algorithm tests

3.3.2.6 Experimental Results on Polygon Detection

The polygon detection algorithm proposed here was implemented in C++ and tested in a Intel Pentium III 1GHz 512MB RAM computer running Windows XP . I tested the algorithm with sets of line segments created from simple vector drawings, technical drawings of mechanical parts and hand-sketched drawings. Table 3.1 presents the results obtained from these tests.

Based on these results I conclude that performance is acceptable for on-line processing in sets with less than three-hundred lines like hand-sketches or small-size technical drawings. If the line set have about 2500 lines the algorithm will take more than twenty minutes to detect the polygons. Still this remains a feasible solution for batch processing of medium-size technical drawings or simplified large technical drawings.

3.3.3 Topological Information Gathering

Following the suggestion made by Fonseca [34], only two topological relationships are used to describe the topology of shapes in the drawings, inclusion and adjacency. While these relationships are weakly discriminating, they do not change with rotation and translation, allowing unstrained drawing classification.

After a recursive decomposition of drawing according to relationships referred above, I construct a topology graph that represents these relationships among relevant shapes. This graph is submitted to Fonseca's framework which will use it for computing topological descriptors for that drawing. Figure 3.23 illustrates two important results produced during the feature extraction process, the simplified set of detected polygons and the re-

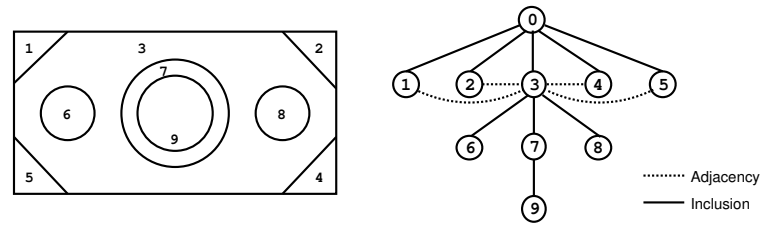


Figure 3.23: Set of simplified polygons (left) and correspondent topology graph (right).

spective topology graph, yielded by the topological relationship extraction. To extract topological relationships I use a simple but efficient sweep algorithm that compares each polygon with the ones within a specified range to check for any relationship among them. This procedure is divided in three distinct stages.

```

TOPOLOGY-EXTRACTION( $\Theta$ )
1   $\xi \leftarrow$  CONSTRUCT-EVENT-QUEUE( $\Theta$ )
2   $SL =$  empty set of polygons
3  for each  $e$  in  $\xi$ 
4  do
5    if  $e$  is a leftmost event
6    then
7       $P \leftarrow$  polygon referenced by  $e$ 
8      for each  $P_{SL}$  in  $SL$ 
9      do
10     COMPARE-POLYGONS( $P, P_{SL}$ )
11     Add  $P$  to  $SL$ 
12    if  $e$  is a rightmost event
13    then
14      $P \leftarrow$  polygon referenced by  $e$ 
15     Remove  $P$  from  $SL$ 
16   $G \leftarrow$  CONSTRUCT-TOPOLOGY-GRAPH( $\Theta$ )
17  return  $G$ 

```

The **TOPOLOGY-EXTRACTION** algorithm starts by creating an ordered event queue (ξ) which contains events corresponding to the xx -coordinates of leftmost and rightmost vertices of all polygons. Each element in ξ contains a flag indicating if it corresponds to a left or a rightmost point and a reference to the respective polygon. The event queue is ordered by the x -coordinate of the vertex and when this value is the same for several vertices the leftmost vertices are placed before the rightmost vertices, making it easier to determine which polygons must be compared. Simultaneously, is created an empty set of polygons to represent the sweep line (SL) where "active" polygons will be stored.

In a second step, the algorithm sweeps the event queue ξ and for each event e it adds or removes polygons from the sweep line accordingly and compares, before adding, the respective polygon with the "active" ones - the polygons on the sweep line structure - which can eventually have an adjacency or inclusion relationship with it. To allow a fast selection of candidate polygons, are considered for comparison the ones that have more than one vertex inside or over the border of the axis aligned bounding box.

During the comparison of polygons (COMPARE-POLYGONS), each pair is checked for adjacency and inclusion by applying common computational geometry methods, described in [88]. When any relationship is found among two shapes, it is stored within the polygon structure for later graph construction.

The third stage of the relationship extractor algorithm processes the relationships gathered during polygon comparison and constructs (CONSTRUCT-TOPOLOGY-GRAPH) the respective topology graph, depicted in Figure 3.23. In this graph each node represents a polygon. When a polygons includes another the node of the first is parent of the node of the second. Adjacency relationships among polygons produce sibling relationships in the respective graph nodes.

The graph yielded by the topological relationships extractor algorithm is then submitted to Fonseca's framework to be converted in topological descriptors using the methods described by Fonseca in [34].

3.3.4 Geometrical Feature Extraction

Fonseca and Jorge proposed a [38] a simple, fast and compact approach to recognize geometric shapes through the analysis of the convex hull of the shape. They identified a set of relevant features that when properly combined provide an effective method to describe geometric shapes for recognition.

To determine these features, Fonseca and Jorge calculate the convex hull (ch) of the shape and then compute four special polygons. The first two are the largest triangle (lt) and quadrilateral (lq) inscribed in the convex hull [10]. The third special polygon is the non-aligned smallest area enclosing rectangle (er). Then they calculate areas and perimeters of these special polygons, as well as width and height of er and the perimeter of original polygon. These relevant features are listed in Table 3.2.

<i>Feature</i>	<i>Description</i>
A_{ch}	Area of the convex hull
A_{er}	Area of the enclosing rectangle
A_{lq}	Area of the largest quadrilateral
A_{lt}	Area of the largest triangle
H_{er}	Height of the (non-aligned) enclosing rectangle
P_{ch}	Perimeter of the convex hull
P_{er}	Perimeter of the enclosing rectangle
P_{lq}	Perimeter of the largest triangle
T_l	Total length, <i>i.e.</i> perimeter of original polygon
W_{er}	Width of the (non-aligned) enclosing rectangle

Table 3.2: List of relevant geometrical features.

Furthermore, they combine these geometric features to produce a feature vector. These combinations are mainly ratios between carefully selected features [37]. From these combinations I selected the more relevant to classify shapes for retrieval and create with them the SIBR geometric feature vector, depicted in Figure 3.24.

$$\left[\frac{P_{ch}}{T_l} \quad \frac{A_{ch}}{P_{ch}^2} \quad \frac{H_{er}}{W_{er}} \quad \frac{A_{lq}}{A_{er}} \quad \frac{A_{ch}}{A_{er}} \quad \frac{A_{lq}}{A_{ch}} \quad \frac{A_{lt}}{A_{lq}} \quad \frac{A_{lt}}{A_{ch}} \quad \frac{P_{lq}}{P_{ch}} \quad \frac{P_{lt}}{P_{ch}} \quad \frac{P_{ch}}{P_{er}} \right]$$

Figure 3.24: SIBR geometric feature vector used in construction of geometry matrix.

Thus, to determine the geometrical features from drawings and sketches I suggest the use of a geometry matrix containing geometrical features of detected polygons. Each line of this matrix contains a feature vector describing the geometry of a single polygon. Using features proposed by Fonseca and Jorge to describe each polygon I was able to produce a matrix that effectively describes the geometry of vector drawings and sketches.

Therefore, construction of geometry matrix is based on SIBR geometric features vectors from all detected polygons. To that end, the GEOMETRY-EXTRACTION algorithm I devised iterates through all detected polygons (stored in Θ) and computes geometric features of each polygon. The COMPUTE-FEATURE-VECTOR method uses the CALI³ library [39], which were developed by Fonseca and Jorge to implement their approach on shape recognition. This method determines the geometric feature vector V_{geom} of a given

³CALI is a software library for the development of Calligraphic Interfaces centered mainly on a simple Shape Recognizer. This recognizer is a fast, simple and compact approach to identify scribbles (multi-stroke geometric shapes) drawn with a stylus on a digitizing tablet. In my approach I only use one of the several functionalities provided by this library. Considering a polygon as a single scribble, I submit it to CALI in order to extract its features.

polygon P by submitting it to CALI recognizer and then selecting the interesting features. Then, vector V_{geom} is added to the geometry matrix M_{geom} as a new line.

```

GEOMETRY-EXTRACTION( $\Theta$ )
1   $M_{geom}$  = empty geometry matrix
2  for each  $P$  in  $\Theta$ 
3  do
4     $V_{geom}$  = COMPUTE-FEATURE-VECTOR( $P$ )
5    Add  $V_{geom}$  as a new line in matrix  $M_{geom}$ 
6  return  $M_{geom}$ 

```

At the end of this iterative process, the geometry matrix M_{geom} is a $N \times 11$ matrix, where N is the number of detected polygons and eleven is the number of selected features to describe shape geometry. This geometry matrix describes the geometry of the detected polygon set, *i.e.* the geometry of the drawing or sketch.

3.4 Indexing, Query and Matching

In previous section I described the feature extraction process which yields two structures: topology graph and geometry matrix. These structures are used not only to classify existing drawings but also to describe sketched queries. Thus, the feature extraction process is exactly the same in classification and retrieval.

3.4.1 Classification

During classification of existing drawings, each drawing underwent a feature extraction process. Resulting topology graph and geometry matrix must then be stored in a database which will be queried later. To that end, I used the indexing mechanism provided by Fonseca's framework. Therefore, both graph and matrix are passed to Fonseca's framework, which computes the correspondent descriptors and inserts them in two indexing structures, one for topological information and another for geometric information. Descriptor computation is based on spectral information and indexing uses a NB-Tree structure. Both subjects are beyond the scope of this work. They were developed within Fonseca's framework and we refer the reader to Fonseca's PhD [34] for detailed information.

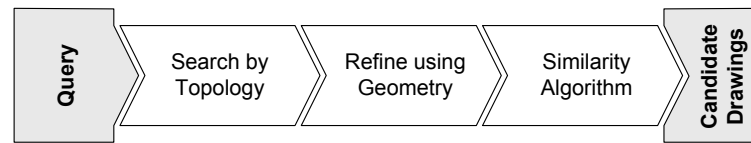


Figure 3.25: Block diagram for the matching process.

3.4.2 Retrieval

The proposed SIBR system includes a calligraphic interface to support the specification of hand-sketched queries. Moreover, digitized drawings can be used to specify queries. Digitized drawings could be edited, using sketches to add, remove or change visual elements. Thus, any query in the SIBR system is considered a sketched query even if it originates on a hard-copy version of the ought drawing.

Sketched queries underwent the feature extraction process in order to extract the spatial relationships, construct the topological graph and compute the geometry matrix. Like in classification, the graph and matrix produced by the feature extraction are submitted to Fonseca's framework to compute the corresponding multidimensional descriptors. These descriptors will be used in the matching process as a query to the respective indexing structure. However, computing the similarity between a hand-sketched query and all drawings in a database can entail prohibitive costs especially when considering large sets of drawings. To speed up searching, I divide my matching scheme in a three-step procedure as shown in Figure 3.25. The first step uses the topology descriptor to search for topologically similar drawings, working as a first filter to avoid unnecessary geometric matches between false candidates. In the second step I use geometric information to further refine the set of candidates. Finally, I apply a comparison method to get a measure of similarity between the sketched query and drawings retrieved from the database.

The matching procedure first ranks drawings in the database according to topological similarity to the sketched query. This is accomplished, within Fonseca's framework, by performing a KNN⁴ query to the topology indexing structure, using the respective descriptor computed from the sketched query.

Results returned by the topology indexing structure represent a set of descriptors sim-

⁴Given a point query Q in multi-dimensional space, K-Nearest Neighbor (KNN) queries return the K closest answers according to given distance metric in the database with respect to Q . For more details on KNN queries, I refer readers to [98].

ilar to the query descriptor. Each returned descriptor correspond to a specific graph or subgraph stored in the topology database, which will be used in the geometry matching. This first filter based on topology reduces drastically the number of drawings to compare, selecting only drawings with a high probability of being similar to the sketched query.

3.5 Summary

In this chapter I described an integrated solution for retrieval of engineering drawings using sketches and images. My approach uses the framework for content-based retrieval presented by Fonseca [34] and incorporate his matching algorithms and indexing structure with my vectorization, simplification and feature extraction algorithms.

First, I presented an overview of my approach. Then I described in detail a vectorization methodology that converts digitized drawings in raster format into a set of line segments with minimal user intervention.

In the detailed description of feature extraction I started by describing the iterated snap rounding algorithm used for line set simplification. Then, I presented the polygon simplification algorithms that eliminates small polygons by merging them with others or by purely discarding them. These polygon simplification algorithms also removes undesirable inclusions. Next I described a novel algorithm for polygon detection on a set of line segments. Then, I present the algorithm devised for topological information gathering, that constructs a graph describing the topological relationships among detected shapes. Next, I described the geometric feature extraction procedure that uses feature vectors from detected shapes to produce a geometry matrix describing the geometry of the set of detected polygons.

Finally, I presented the classification and retrieval processes. These uses the feature extraction described earlier to compute the topology graph and geometry matrix for drawings and queries, respectively. Then, this topological and geometrical information is submitted to Fonseca's framework for insertion, when classifying, or searching, when retrieving, in the indexing structure.

4

Prototypes

To exercise the retrieval approach presented in previous chapter I developed two distinct prototypes. The Sketch and Image Based Retrieval (SIBR) prototype processes sketched queries and matches candidate drawings against the query, selecting and displaying results. This prototype allows using not only sketches to specify queries, but also digitized images or a mix of both. Additionally, I create the Database Builder prototype, which implements the classification part of presented approach, classifying existing drawings and producing a logical database with descriptors. In this chapter I will describe these prototypes.

4.1 Sketch and Image Based Retrieval Prototype

In order to create a retrieval system that takes advantage of users' natural ability at sketching and drawing, I developed an application to retrieve technical drawings from the mould industry, through hand-sketched queries and digitized drawings. SIBR system retrieves sets of technical drawings by contents from large databases.

The SIBR prototype uses a calligraphic interface to allow specifying queries through sketches. Moreover, it is able to convert raster images into vector drawings and use these as queries. Users can add new elements to vectorized drawings by sketching new shapes or they can delete entities by using a simple gesture command. This way, they can start with a digitized drawing and then apply editing commands to refine it.

In some cases, digitized drawings can be very complex. In other situations users may not want to search for the entire drawing. To these end, SIBR system makes it possible to use part of a sketch or vectorized drawing as a query. To accomplish this, SIBR prototype

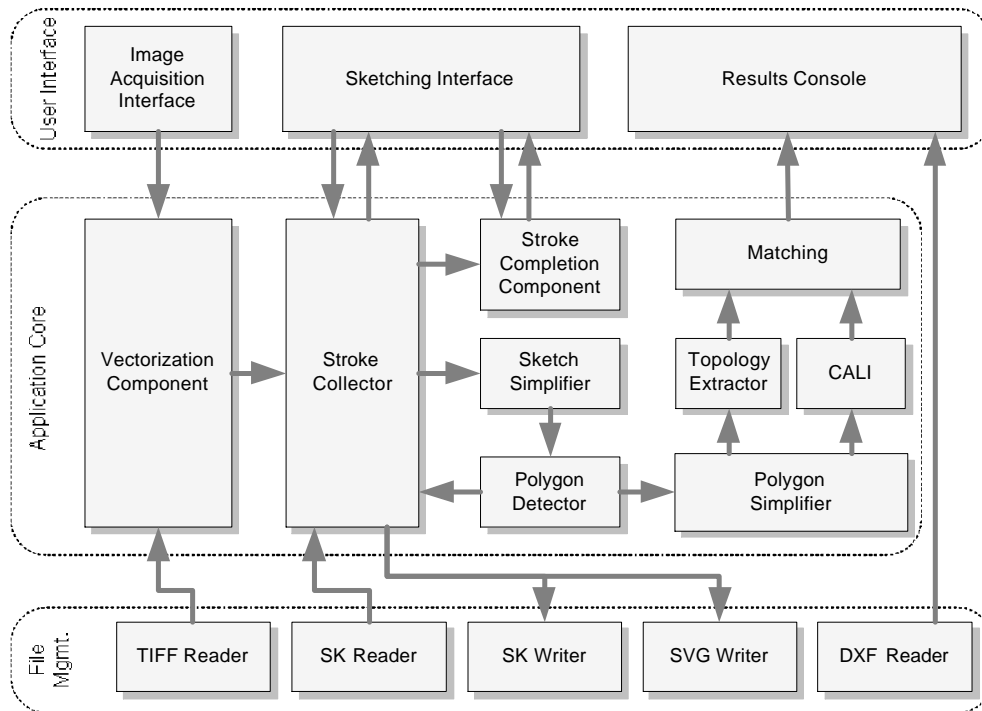


Figure 4.1: Conceptual view of Sketch and Image Based Retrieval Prototype architecture.

provides tool for selecting a region on the sketch and submitting it as a query. Thus, it is possible to discard visual elements without deleting them, making further searches easier.

4.1.1 Architecture

To ensure modularity, SIBR prototype was developed based on a three tier architecture which comprises user interface, application core and file management. Figure 4.1 depicts a conceptual view of SIBR architecture. The top tier is responsible by all interaction with users, the bottom tier manages the interactions with the file system and the application core tier contains the components that implement all the algorithms used in the retrieval process.

4.1.1.1 User Interface Tier

The user interface tier of the SIBR prototype is composed by three distinct components: sketching interface, results console and image acquisition interface. The first captures the strokes drawn by users and generates the visual feedback that are given to

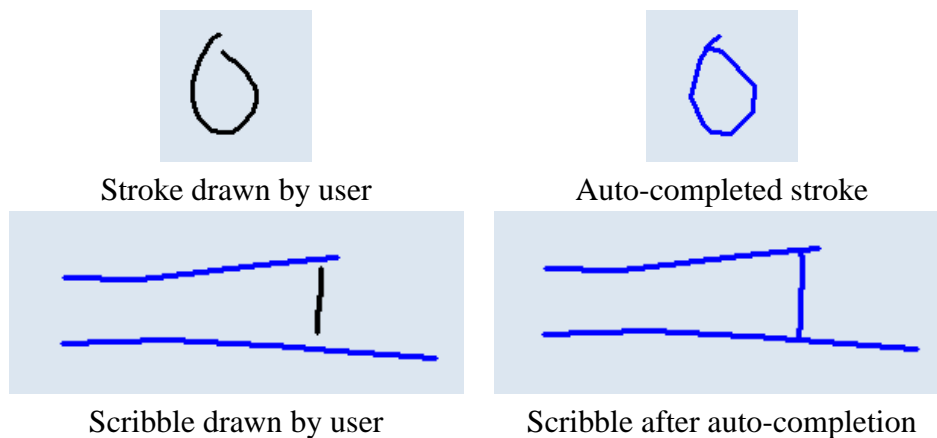


Figure 4.2: Example of single stroke (top) and multi-stroke (bottom) auto-completion.

users, such as identifying detected shapes that will be used by the query. To give visual feedback about the polygons detected on the sketch, the Polygon Detector component sends information about detected polygons to the Sketching Interface.

A common complain made by users during the preliminary evaluation tests was the difficulty in drawing closed shapes. Frequently they had to delete and repeat the stroke because it was not closed, and the system ignores it. To overcome this problem I included two auto-completion features in the SIBR prototype. These features, the single stroke completion and multi-stroke completion are implemented by the Stroke Completion component. In the first one the component analyzes the stroke alone and in the second it compares it with existing strokes and tries to create a closed shape. Figure 4.2 depicts examples of single stroke and multi-stroke auto-completion.

Results console component manages the drawings that must be returned to the user as a result of a submitted query. To that end, it receives a set of drawing identifiers from the Matching component, along with information about similarity between the submitted query and each result. Using the received identifiers and working in conjunction with the DXF Reader component, the Results Console is able to display the respective drawings to the user.

From informal conversations with users during the first evaluation tests I infer that searches based on printed drawings will be useful. Therefore, I included in this prototype a functionality that allows retrieving drawings through the use of scanned images of technical drawings. A scenario for using this new functionality occurs when users have a

printed hard copy of the drawing and want to find similar results stored in the database.

The image acquisition interface component implements the interaction with users when using digitized drawings to specify queries. This component gives to users control over the vectorization process, allowing them not only to indicate the source image but also to follow the vectorization steps, changing some parameters if needed.

4.1.1.2 Application Core Tier

Each stroke captured by the sketching interface is sent initially to stroke completion component and then, after it has been processed there, is sent to the stroke collector component. This component manages all strokes drawn by the user, constructing a sketch with them. Besides working with strokes made directly by users, stroke collector can also use sketches drawn previously and stored in files. To that end, this component can receive a complete sketch from the SK reader component or send the stored set of strokes to be saved by one of the writer components. The sketch is stored as a set of polygonal curves representing each one a stroke drawn by users. When users trigger a query process, the sketch stored in the stroke collector are sent to the sketch simplifier component to start processing.

The sketch simplifier component implements the algorithms for line set simplification described in previous chapter. This component tries to reduce the number of points that define the sketch, maintaining its relevant features. The simplified version of the sketch is then sent to the polygon detector component.

Polygon detector component receives the simplified sketch, in the form of several sets of polygonal lines, and processes it to detect polygons. The results of this computation are sent simultaneously to the stroke collector to produce visual feedback for users and to polygon simplification component to continue processing the sketch.

The polygon simplifier component implements the algorithm described in Section 3.3.1.2, which receives a set of detected polygons sent by the polygon detector component, processes them in order to simplify the set and finally sends the simplified set to CALI and topology extractor components.

The topology extractor component implements the algorithm for topology extraction

described in Section 3.3.3. It analyzes the simplified set of polygons sent by the polygon simplifier and extracts the topological relationships among them, building a graph with this relationships.

To determine geometrical information of each polygon in the simplified set produced by the polygon simplifier I use the CALI component, an implementation of the CALI library described briefly in Section 3.3.4 and developed within the PhD work of Manuel João da Fonseca [34].

The query component combines information produced by the topology extractor and CALI to create topology and geometry descriptors, as described in Section 3.3. These descriptors will be processed by the matching component to find drawings with similar descriptors in the database. A set of drawing identifiers is produced as a result of this processing. This set of identifiers is finally sent to the results console component, which is responsible for displaying the respective drawings to the user.

4.1.1.3 File Management Tier

Sketches collected by the SIBR can be saved in a file for later analysis. There are two distinct file types that can be used: the well known SVG¹ format or a proprietary file type format, which I called sk-file. Thus, I create two distinct components to write sketches on files, one for each file type.

The usage of SVG format allows a broader use of collected sketches, is possible, for instance, to open them in many different applications, including several web browsers. This way the visualization and analysis of sketches produced by users is simplified, since I can use off-the-shelf applications to accomplish it.

However, a major disadvantage of SVG format is the large amount of data needed to represent a complex sketch, producing a large XML file. Additionally, I am not satisfied with the performance of XML parser initially integrated in SIBR prototype. To reduce the file size and improve the parsing of existing files I define a proprietary format (sk-file) which is simply composed by sets of strokes stored as pairs of floating-point values representing the vertices of a polyline.

¹Scalable Vector Graphics (SVG) is a modularized language for describing two-dimensional vector and mixed vector/raster graphics in XML.

SIBR allows later analysis of sketches drawn by users and saved in the proprietary format. The SK-Reader component reads the content of a sk-file and send it has a set of strokes to the Stroke Collector. Using this feature I was able to review and edit the sketches using the SIBR, repeating the processing with improved versions of application components and comparing results. On the opposite side of my approach, it is necessary to read the original drawings. To that end, I developed a component to read DXF² files, since engineering drawings I use are all stored in this format or can be easily converted. The DXF reader component parses DXF files containing two dimensional drawings and produces a set of entities that represents the drawing. These entities are then used by the Results Console component to produce a visual representation of the drawing.

In SIBR prototype I do not implement an image acquisition component capable of directly connect to a scanner because it will add some extra development that is not within the scope of my work and will make harder to maintain the cross platform compatibility of presented prototypes. However, it is not hard to incorporate a TWAIN³ component in the Microsoft Windows version of SIBR prototype. Instead I included a TIFF⁴ reader component to import digitized drawings. Additionally, the capability of reading raster files allows a remote use of paper drawings, which can be scanned remotely and sent in a TIFF file.

4.1.2 Graphical User Interface

This prototype uses a calligraphic interface to allow the retrieval of a set of drawings similar to a hand-sketched query performed by the user. Reflecting the composition of the user interface tier, the GUI of the SIBR Prototype is divided in three main areas: sketching area, buttons area and results area, as depicted in Figure 4.3. The sketching area allows

²Drawing eXchange Format (DXF) is a native vector file format of Autodesk's AutoCAD CAD application. DXF is probably one of the most widely supported vector formats in the world.

³TWAIN is an image capture API for Microsoft Windows and Apple Macintosh operating systems. The standard was first released in 1992, and is currently ratified at version 1.9 as of January 2000. TWAIN is typically used as an interface between image processing software and a scanner or digital camera.

⁴Tag Image File Format (TIFF) is a common format for exchanging raster images between application programs. This format was developed in 1986 by an industry committee chaired by the Aldus Corporation (now part of Adobe). Microsoft and Hewlett-Packard were also on the committee. One of the more common image formats, TIFFs are common in desktop publishing, faxing and other imaging applications, since it is an lossless uncompressed image file format that produces no artifacts as is common with other image formats.

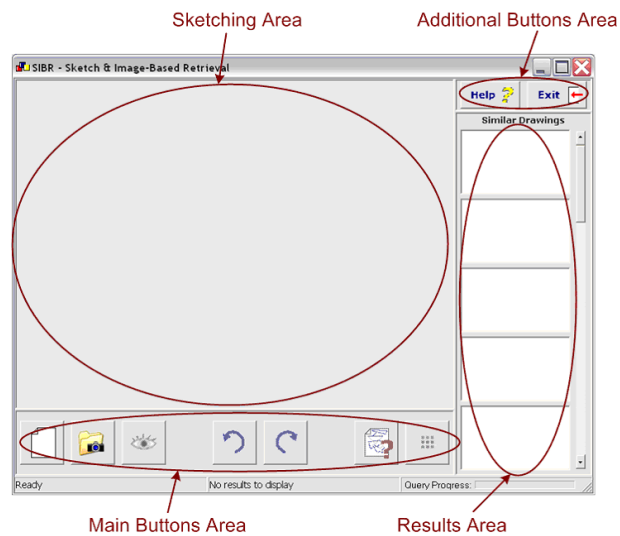


Figure 4.3: Screen-shot of Graphical User Interface of SIBR.

the specification of a query using sketches composed by a set of strokes. Additionally, the vector version of the digitized drawings is also displayed here. If users want to edit that drawing they sketch over it. It is also possible to select just part of the sketch to use as a query. At the right side of the interface, there is the results area composed a set of five blank panels, where results similar to the sketched query are displayed. Returned drawings are ordered from top to bottom, being the most similar on the top. Since each query returns twelve drawings and it is not feasible to show them all at once, only five drawings are displayed simultaneously. The scroll bar located at right allows browsing, up and down, trough the twelve drawings.

Executing a query usually takes several seconds, sometimes even a couple of minutes, depending on the complexity of the sketch and of the existing drawings and on the size of the database. At the bottom right of the window a progress bar gives some feedback about the evolution on the classification and retrieval process. Additionally, the current step of the process is also displayed on the status bar.

4.1.2.1 Using Sketches

To perform a query using only sketches, user must draw on the sketching area a rough approximation of the wanted drawing, or part of it, and submit that sketch as a query. When the ought drawing does not appear within the twelve results returned by

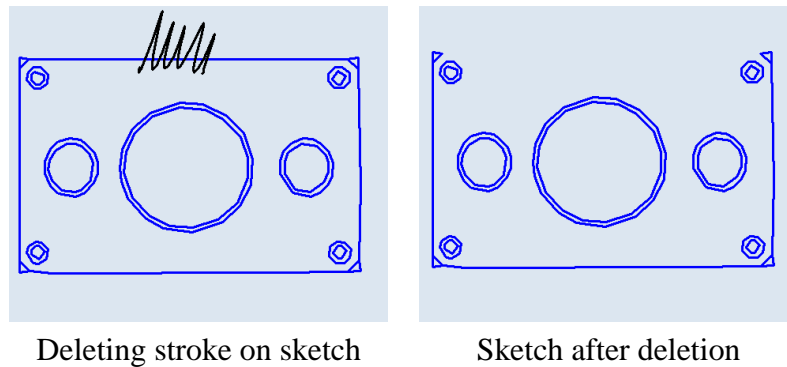


Figure 4.4: Using delete gesture to remove a stroke.

the SIBR system, it is always possible to edit the drawing, removing, adding or changing some features and then submit the new version of the sketch. This can be repeated as many times as user wishes until the desired drawing is found.

The user can, at any time, restart sketching from scratch by pressing the "New Query" button. In this case the sketching area will be deleted and drawings displayed in results area will be cleared.

While editing a sketch, the user can delete part of it. To that end, the "Delete" gesture must be used. The "Delete" gesture consists on drawing a zig-zag stroke over the shapes to delete, as depicted in Figure 4.4.

When the sketch is too complex, too big or have undesired shapes, user can select just part of it to use as a query. To accomplish this, SIBR application provides a tool for selecting a region on the sketch and submitting it as a query. Thus, it is possible to discard visual elements without deleting them making further searches easier. To select a region user must change from "Sketch Mode" to "Select Mode" and then use the pen to specify the region. To enter "Select Mode" it is necessary to press the pen over the sketching area without moving for two seconds. After entering select mode, the cursor over the sketching area will change to a cross and the stroke will appear in dashed white, as depicted in Figure 4.5.

To discard the selection, the pen must be pressed over the sketching area, maintained without moving for a couple of seconds and then lifted up. From now on the whole sketch will be used as a query. After a region has been selected by the user, the SIBR switches immediately from "Select Mode" to "Sketch Mode". Now the sketching area outside the

selected region appears with a darker background and its border is clearly identified by a dashed white line. When a region is selected, users can still sketch freely on any part of the sketching area, inside or outside of the selected region. However, only the strokes drawn inside the selected region will be considered for query purposes.

4.1.2.2 Using Images

The SIBR system allows searching for a drawing based on its paper version without using sketches. To that end, it is necessary to have a flatbed scanner connected to a computer to digitize the paper drawing. Additionally, the correspondent image acquisition software must be installed on that machine and it must be able to save the acquired images in TIFF format. Almost all scanners available will digitize drawings without problems. Even the cheaper flatbed scanner solutions will work well with SIBR for retrieval purposes.

After the drawing has been scanned and saved in a TIFF file it can be used by the SIBR application. A major advantage of this approach to use raster files instead of direct connection to scanner lays on the possibility of using remotely digitized images, which can be shared over a network or even sent by e-mail.

As soon as the image file is opened a new window is displayed (Figure 4.6). Here the users will control the vectorization process. In some steps of the vectorization, the system will ask the user about methods or parameters to apply during processing. Every one of these questions is accompanied by a suggestion. When users are not sure of which option produces the best results, they can follow the suggestion made by the system. However,

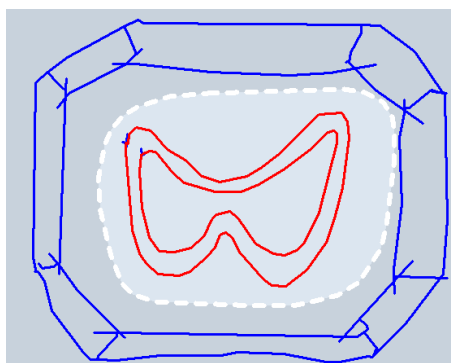


Figure 4.5: Selecting part of a sketch

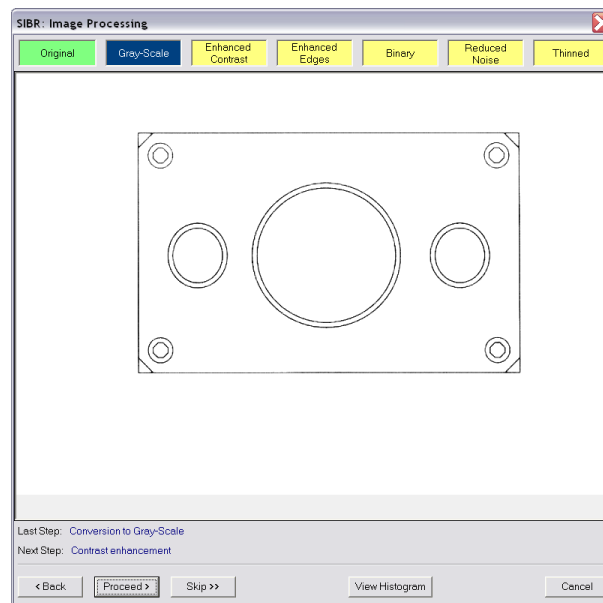


Figure 4.6: Screen-shot of image processing window

if at any time of the vectorization process the resulting image does not satisfy the users, they can undo one or more steps and then repeat them with different options.

Depending on the quality of the scanned drawing, not all steps of the vectorization process are required. Thus, the users can sometimes skip a step they consider unnecessary if it is an optional step (refer to Section 3.2, Figure 3.8 for more details on vectorization steps). This way the time spent in the vectorization process could be shortened.

Moreover, experienced users can consult the histogram of the image and use their knowledge to help them choosing properly the more efficient methods and parameter to apply during the image vectorization. On the other hand, novice users can perform their choices by simply following the suggestions made by the system or using a trial and error approach.

To allow an effective control of the vectorization process, the image-processing shows the current status of the vectorization. On the top of the window is indicated which version of the image is displayed below. Additionally, this also shows which steps were performed and which were skipped. Under the image are shown the last step performed and the following one. At the bottom of the window a set of buttons allows the control of the process.

When the vectorization is finished the result is drawn by the system on the sketching

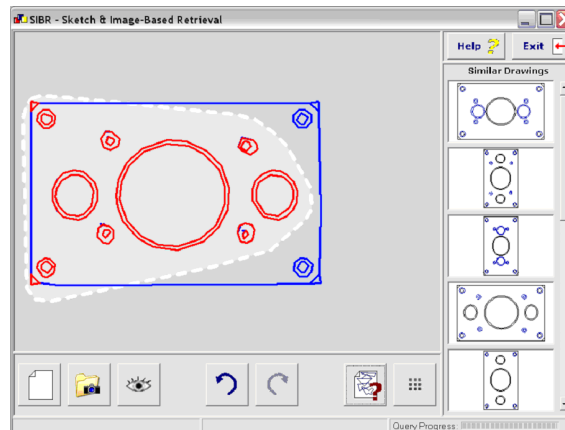


Figure 4.7: Mixing images and sketches to retrieve a technical drawing.

area and is from now on considered as a sketch. Therefore, the procedure to execute a query with it is exactly the same as with a hand-made sketch.

4.1.2.3 Mixing Sketches and Images

Since digitized images after vectorization are considered as a sketch, the users can treat it exactly the same way they treat a sketch. Thus it is possible to sketch over it, adding new shapes, delete existing ones and select only part of the sketch to use as a query.

Mixing sketches and vector drawings give the users the capacity to derive more complex queries. They can add new elements to vectorized drawings by sketching new shapes or they can delete entities by using a simple gesture command. This way, they can start with a digitized drawing and then apply editing commands to refine it. Figure 4.7 presents a query that was refined by adding shapes to the original figure in order to detail more information. The figure illustrates also a selection of part of the drawing to use as a query, without deleting the remaining elements.

4.2 Database Builder

I will now describe the prototype that implements classification part of the content-based retrieval architecture presented in Chapter 3. This application classifies existing drawings, by creating logical descriptors based on features extracted from drawings. Be-

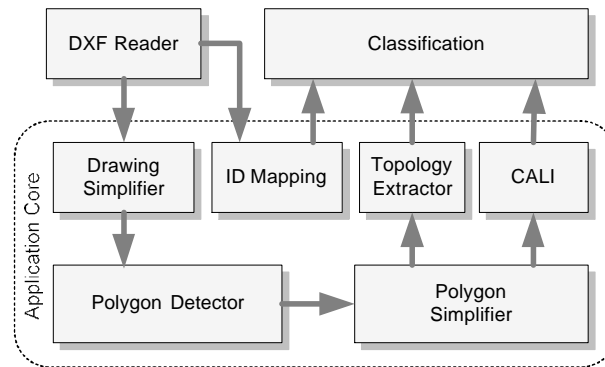


Figure 4.8: Conceptual view of Database Builder prototype architecture.

fore creating the database with the logical information (descriptors) extracted from drawings, Database Builder performs simplification steps to reduce the number of elements in a drawing, thereby speeding up both classification and query. I simplify drawings by reducing adjacent lines to one, by discarding small area polygons and by removing spurious lines. This way I get a simple version of the technical drawing while retaining its main features. This will yield less descriptors and consequently a smaller database. Furthermore, simpler drawings will be easier to match to hand-sketched queries.

The classification of a technical drawing might take from a few seconds to several minutes, depending of its complexity. Therefore, the classification of a large set of drawings may take several hours. However, since this is an operation that will be performed only once, users can execute it in batch mode. To that end, Database Builder offers the possibility of automatically classifying a set of drawings without user intervention. Users only need to specify where drawings can be found and the system will do the rest.

4.2.1 Architecture

Extracting features from existing drawings is similar to extracting features from sketches. Thus, it is obvious that some components used in the retrieval part of SIBR system are also used in the Database Builder. Therefore, the polygon detector, polygon simplifier, topology extractor and CALI components are simultaneously used by both prototypes. Even the drawing simplifier component is basically similar to sketch simplifier component used in SIBR prototype.

The Database Builder prototype have two components that are not shared with the

SIBR prototype. The ID mapping component is responsible for assigning an unique identifier to each classified drawing and store it on the database.

4.2.2 Graphical User Interface

The GUI of the Database Builder prototype, depicted in Figure 4.9, has a small form on top of the window with three fields to specify the working parameter of the classification module. In the first field users can specify the folder containing drawings to classify. If this folder has subfolders, drawings on these will be classified too. Only drawings saved in DXF format with the "dxf" extension will be considered. The database folder field specifies the folder where is located the database.

The initial ID field allows the specification of the value of the initial identifier for the drawings to classify. This identification number is necessary because the drawings stored in the database are identified by it. Since the database can already have classified drawings, the identifier of the new ones must not conflict with the existing drawings identification. Moreover, users might want to group drawings by identifier according to some criteria.

In order to give some feedback to the users during processing, Database Builder application has a log window where the status of the classification is displayed. Additionally, a log file is also saved for later analysis. After checking that values on the fields are cor-

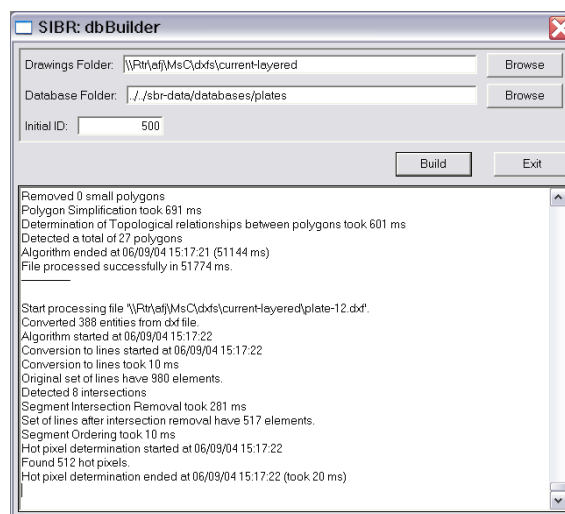


Figure 4.9: Screen-shot of Database Builder prototype during a classification process.

rect, the user can press the "Build" button to start the classification of drawings in DXF format that are in the specified folder. Descriptors of these drawings are then added to the database in specified folder.

4.3 Summary

In this chapter I presented the prototypes developed to exercise the content-based retrieval approach described in previous chapter. The Sketch and Image Based Retrieval (SIBR) prototype exercises the retrieval part of the architecture and the Database Builder implements the drawing classification part.

The SIBR prototype use a calligraphic interface to allow the specification of queries using hand-drawn sketches. It is also possible to use digitized drawings to specify queries or a mixing both images and sketches. In order to convert the digitized drawing into a vector representation, a vectorization process must be executed. During this process the user controls each step and receives constant feedback on evolution of conversion.

I also described the conceptual view of architecture for each prototype. Due to its complexity, the SIBR prototype was based on a three tier architecture, composed by the user interface, application core and file management. On the other hand, the Database Builder, being a simpler application do not need such division, since it is composed by few components. Moreover, most of these components are the same used in SIBR.

5

Experimental Results

During the development of the work presented in this dissertation three experiments with users were carried out researchers of IMMI group. These experiments were performed within the *SmartSketches* European project. All tests were made with users from CENTIMFE¹, a partner in the *SmartSketches* consortium, and took place at CENTIMFE headquarters in Marinha Grande, Portugal.

To take advantage of users' natural ability at sketching and drawing, we, at IMMI research group, first performed tests with users to understand how they sketch 2D views of parts. From the collected data and after analyzing all the information, I completed the first version of a Sketch-Based Retrieval (SBR) prototype. Then, to evaluate it and the performance of its algorithms, we made a preliminary usability evaluation ². During these tests users sketched queries, commented the returned results and answered a questionnaire.

Based on the results from these tests I improved the prototype, producing a second version with a new user interface, enhanced algorithms and new functionalities: the SIBR prototype. This new version of the system was then evaluated by six users and a larger database than in the first tests. Results from these usability tests revealed that users liked the new interaction paradigm, which combines sketches and images to specify queries. Moreover, users were pleased with the returned results and satisfied with the time they have to spend, from sketching to the final results.

As annex to this document I included the protocols and questionnaires presented to users during our usability tests, the databases used in the evaluation experiment and the complete sets of queries used and correspondent results returned by retrieval prototypes.

¹CENTIMFE is a technological training center for the Portuguese Mould Industry.

²The preliminary usability evaluation tests for the SBR prototype has already been summarized by Fonseca's in his PhD thesis [34].

5.1 Sketching Experiment

A Sketch-Based Retrieval system should be able to find a small set of drawings in a large technical drawing database through a hand-sketched query performed by users. However, to take advantage of users' natural ability at sketching and drawing, one must understand how they sketch 2D views of parts.

To accomplish this step we made an sketching experiment with users where we asked them to sketch some drawings while we collected the produced data. For this experiment we use a TabletPC with Sketch Reader installed to collect sketches drawn by users.

5.1.1 Sketch Reader Prototype

To understand how users sketch 2D views of parts I developed a sketching prototype, called Sketch Reader, which collects strokes drawn using a pen on a digitizing tablet. Using this application I were able to gather sketch information produced by them.

The user interface of the Sketch Reader Prototype, depicted in Figure 5.1, is divided in two main areas: a large sketching area above a smaller log output area. Users sketch in the sketching area and collected information is displayed in the log output area. This information is also stored in a log file for later use.

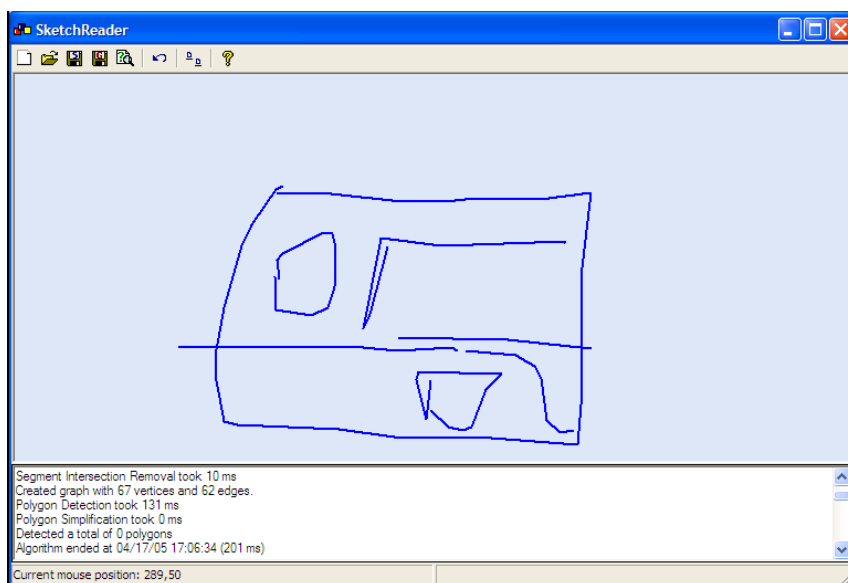


Figure 5.1: Sketch Reader: sketching prototype application

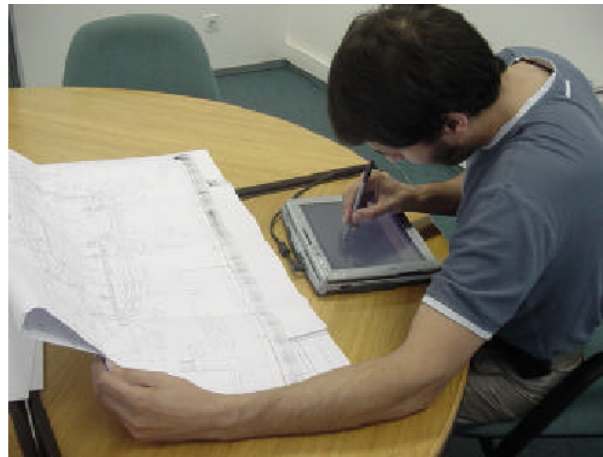


Figure 5.2: User during a sketching session.

Besides working with strokes made directly by users, this prototype can also use sketches stored in files. To that end, this application can read a previously saved sketch. Thus, sketches collected by the Sketch Reader can be saved in a file for later analysis.

This way, Sketch Reader allows later analysis of sketches drawn by users and saved in the proprietary format. Using this feature we were able to review and edit the sketches using the Sketch Reader, repeating processing of sketches drawn during the experiments for further analysis.

5.1.2 Users

To perform this sketching experiment, we involved three draftspeople from CEN-TIMFE, with a good know-how on drafting with CAD tools and with a great ability to recreate mentally the depicted parts.

These users had no prior experience in sketching queries for any kind of SBR system. This way, they were initially unbiased. However, after the first sketching session we gave them some tips on how they should sketch to produce good queries for our retrieval system.

Figure 5.2 portrays a user during a sketching session. There it is possible to see the user holding the paper drawing while drawing a sketch on our prototype.

5.1.3 Experiment Steps

The experiment was divided in two parts. In the first part we introduced some concepts about the experiment and in the second users executed a set of requested tasks. During the first part all users were present, while the second part was composed by individual sessions. The whole process can be summarized in the following steps:

- Introduction to the SBR system (all users simultaneously);
- Description of the experiment (all users simultaneously);
- First sketching session (one user at a time);
- Second sketching session (one user at a time);

Sketching sessions were photographed and videotaped, taking place without our intervention. The next steps composed each session:

1. Present the drawing of the part to the user;
2. Ask the user to sketch a 2D view of the part using a Tablet PC running a sketching prototype;
3. Save the sketch in a file for later analysis and processing;
4. Repeat steps one to three for the remaining parts.

5.1.4 Technical Drawings

A set of five technical drawings of mechanical parts was selected to be presented to users. All of the five drawings had at least the top view of the part. Two of them had other views and another one also had a 3D view of the part. The first drawing (D1) had all the 2D views, some 3D views and other detailed views of the part. Figure 5.3(a) presents the top view of this drawing. The second drawing, which was the most complex of the set, included the top view (illustrated in Figure 5.3 (b)), a side view and several cut views. This drawing was very rich in additional information, such as call out measurements and labels. The next three drawings were edited using a CAD application in order to remove

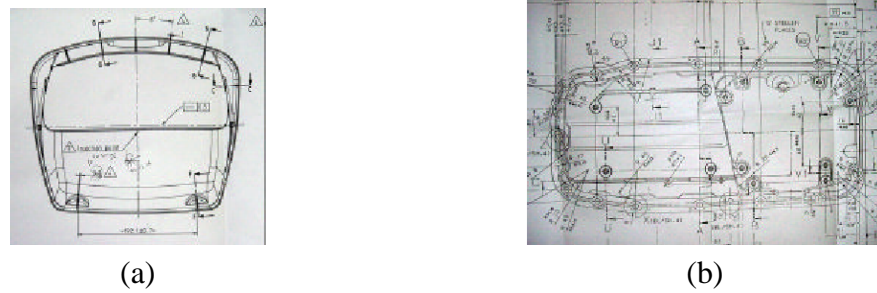


Figure 5.3: Top view of drawings D1 (a) and D2 (b).

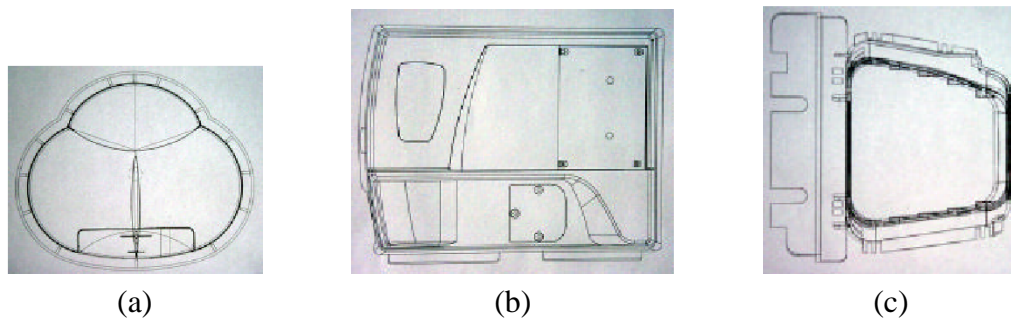


Figure 5.4: Drawings D3 (a), D4 (b) and D5 (c).

the views and unnecessary information, providing only the top view of the parts. Unlike previous drawings, which were presented in the original A0 sheet with several views and additional data, these drawings were presented in an A4 sheet with only the information depicted in Figure 5.4.

5.2 Sketches

Next I will describe sketches of the drawings presented previously, produced by users during the two sketching sessions. An exhaustive listing of sketches made by users during these tests are available in Annex A.

5.2.1 First Sketching Session

The first sketching session occurred without any suggestion on how users should perform their sketches, in order to obtain unbiased results. The main purpose of this session was to understand the way users represent naturally the technical parts, without any interference or constraints.

Unfortunately, due to some time constraints, user C did not sketch drawings D3, D4 and D5 during the first sketching session. Nevertheless, sketches made by users A and B were enough for us to achieve important conclusions.

In drawing D1 all users sketched, immediately after the contour, the larger shape at top, then the two small half-circles at bottom and finally the half-ellipse at top, adjacent to the contour. Only from this step forward they started to diverge, sketching distinct details.

During the sketch of the more complex drawing of the set, D2, users were concerned not only with the major shapes of the part, but especially with the small screw-holes. Like in all other sketches users started by drawing the contour of the part and then the inner shapes.

As before, users started sketching drawing D3 by the contour. After that, they sketched the larger inner shape and added some details carefully, paying special attention to accuracy. Both sketches are composed by a large number of strokes, which makes them quite complex, as depicted in Figure 5.5 (a) despite the apparent simplicity of drawing D3.

When presented with drawing D4 both users started sketching, as always, the contour of the part. After that each one sketched different elements, but both ended by sketching the sets of three and five small circles. In these sketches users drew almost every shape existing in original drawing with an high level of accuracy.

Although the last drawing presented to users did not represent a very complex part, it had a lot of details. Users tried to represent these details with some accuracy, namely the small "staircase" shapes and the little shapes outside the large polygons. In this particular drawing, users did not draw the contour of the whole part at the beginning, but they started by drawing the contour of what can be called the three dominant shapes: a trapezoid at right and two rectangles, a thinner one at middle and a larger one at left. After sketching these three shapes users started adding details to their representation of D5.

Table 5.1 presents the time per drawing spent by each user. We can see that there is some difference between users, but they do not take less than two minutes.

Table 5.1: Sketching times for the first session

User	Drawings				
	D1	D2	D3	D4	D5
A	05m 40s	06m 04s	04m 04s	06m 22s	06m 38s
B	02m 12s	06m 31s	02m 38s	03m 52s	03m 51s
C	02m 50s	04m 13s	*	*	*

* User C did not sketch drawings D3 to D4, therefore we have no times for him.

5.2.2 Second Sketching Session

At the beginning of the second sketching session we gave some suggestions to users on how we expect them to sketch. We told them that the main point was to make a quick sketch, without many details, representing only the main features they perceive from each drawing. Another suggestion was to avoid over-sketching, since it will just add complexity without improving the description of the part.

As a result of these suggestions sketches produced in this session were much simpler than the ones produced before and the time spent on each drawing decreased significantly.

When presented with drawing D1, users sketched them in less than a minute. Like before, they started by sketching the contour, then the larger inner shape and finally the smaller inner half-ellipses.

From a quick comparison between these sketches and those performed in previous sketching session it is clear that users only drew the main features. This way it is possible to see clearly which shapes they consider more relevant on drawing D1.

Once again, users started sketching D2 by representing its contour and then the smaller shapes. Here, user A took simplification seriously and sketched a minimal representation of the part, focusing only on the main components. On the other hand, user B depicted not only the larger shapes but also the screw-holes.

From a quick analysis of these sketches and taking in account the drawing order, we can say that they identified the same main features in D2: the contour of the part, the larger inner shape that follows the contour and the division in half of the inner shape.

The part depicted by D3 was sketched in a much simpler way on the second sketching session, as Figure 5.5 illustrates. Like in previous drawings, user A chose a simplified

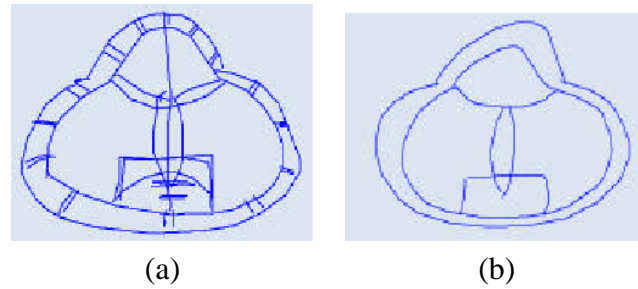


Figure 5.5: Sketches representing drawing D3 produced by user B in first (a) and second (b) sketching sessions.

representation, sketching a rough approximation of the contour and just one shape inside. The sketches produced by users B and C are extraordinarily similar, reproducing the same shapes that they consider more relevant.

It is clear from the analysis of sketches produced by users B and C to represent the part depicted in drawing D3 that they considered the contour and the three inner shapes as the more relevant elements.

The sketch of part D4 made by user A is quite simple, like all the others he did in this session. As before, all users sketched the contour of the part, followed by the inner shapes. Sketches made by user A and user C depicted basically the same shapes of the part, which were the ones that user B sketched in first place, before sketching the details. Therefore, I can state that all users identified the same relevant shapes from the current part (shapes sketched by users A and C).

To conclude the second session, we presented drawing D5 to all users. This time they sketched much simpler representations of the part, ignoring lots of small details they sketched in the first session.

To sketch this part all users adopted the same methodology, sketching first the two main shapes, the large rectangle at the left and the trapezoid at the right, and then the shape inside the trapezoid. The simplified representation sketched by user A includes an additional stroke dividing vertically the rectangle in order to recreate the two almost rectangular shapes on the left of the part. Sketches produced by users B and C had a little more detail, namely in the format of the rectangular shape.

As in first session, we measure the sketching times for every user during this second session. Table 5.1 presents the time per sketch spent by each user. Results show that users

Table 5.2: Sketching times for the second session

User	Drawings				
	D1	D2	D3	D4	D5
A	*	00m 44s	00m 21s	01m 00s	00m 39s
B	00m 58s	01m 21s	00m 31s	01m 06s	00m 45s
C	00m 46s	01m 04s	00m 27s	00m 34s	01m 02s

* User A was interrupted several times while sketching D1, therefore it was not possible to correctly measure its time.

spent less time in this session, taking no more than one and a half minute to complete their action.

5.2.3 Analysis of the Sketching Experiment

From this sketching experiment I can conclude that the use of sketched queries to find drawings in large technical drawing databases is validated by the coherence found among sketches performed by all the users of our experiment.

Additionally, all users recognized the same relevant shapes in each drawing. I also noticed that users consider the contour of the part as the most relevant shape, since they always started by sketching it.

Moreover, I found out that users' natural ability for sketching and drawing can be used naturally in a Sketch-Based Retrieval system. This is an important conclusion, since it validates the concept I present in this dissertation, from users point of view.

However, to best explore this advantage, one must provide users with some information on how proposed SBR system will expect them to sketch queries. The comparison of sketches from the two sessions shows that users produce results more suited for a SBR system when they know what is supposed to draw.

In the second session, we asked users to avoid over-sketching and to ignore details. As a result we got simpler sketches, closer to what was expected to have for queries, and reduced sketching times. While in the first session users took between two to seven minutes to sketch each drawing, in the second session they spent no more than one minute and a half. From Figure 5.6 we can also see that times are more similar between users during session two that during the first one.

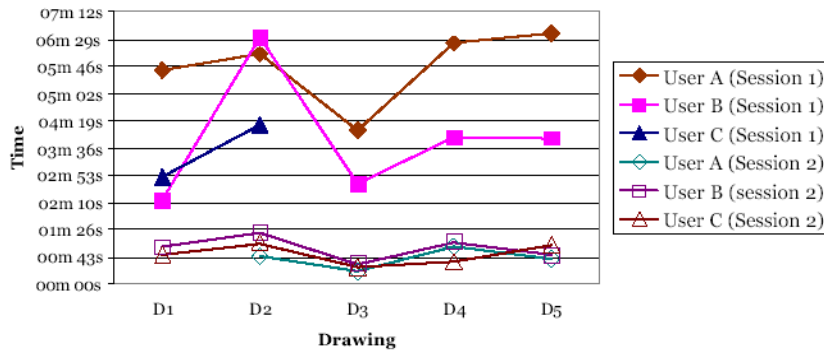


Figure 5.6: Time comparison.

Another relevant observation from this experiment was the way users sketched. They tend to leave unclosed shapes that are supposed to be closed. This situation is illustrated in Figure 5.7, where we show part of a sketch made by a user (left) and the corresponding region on the original drawing (right).

Moreover, although many sketches depicted in this document appear to have closed shapes, a closer inspection will reveal that several of them are opened. Some are clearly visible, as illustrated in Figure 5.7 (a), while others have only very small gaps. A variant of this problem appeared during the sketch of small closed shapes, namely small circles. An example of this situation is depicted in Figure 5.7 (c), where the user made an approximation of a circle leaving a gap between the two endpoints of the stroke.

5.2.4 Conclusions from sketching experiment

From the sketching sessions, I noticed that users sketch a lot of details and use too much accuracy, unless we tell them to be more objective. I also observed that all users

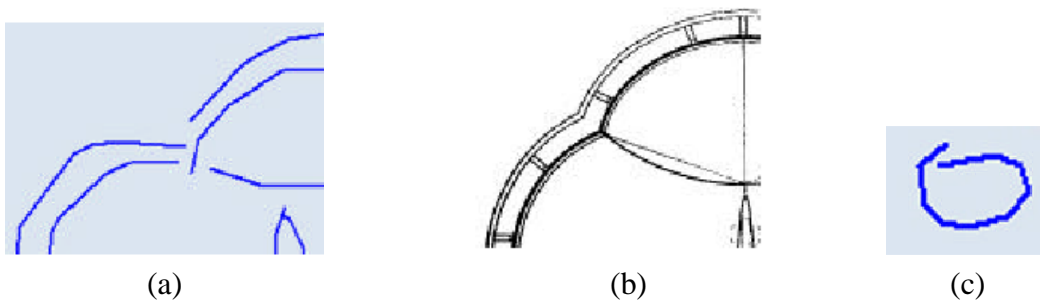


Figure 5.7: Detail of a sketch made during second session (a), same detail of the original drawing (b) and amplified sketch of a small circle (c).

identify the same relevant shapes of a drawing, validating our approach of using sketches as queries. This experiment was also important to see how users perform their sketches, providing information on how to improve and enhance the SBR prototype.

As a conclusion from this experiment I identified a couple of features and improvements to include in the SBR system. For instance, observation mentioned before unveils the necessity of an intelligent sketch editor. This editor should be able to perform auto completion of shapes during sketch.

5.3 Preliminary Usability Evaluation

This section presents the results of these preliminary usability tests of the SBR prototype. The main goals of this experiment were to test classification and retrieval algorithms and collect users' opinions about the results returned by the system. Additionally, they also evaluated the user interface, suggesting changes to improve the final version.

This experiment was also important to see how users sketch queries using a Tablet PC and a pen, and what their expectations were about the answers from the system. Collected data provided valuable information to redesign the SBR user interface, to improve the sketching method, enhance the retrieval algorithm and integrate new features.

5.3.1 Users

To perform this experiment, we involved three draftspeople from CENTIMFE, with a good know-how on drafting with CAD tools. These users have no prior experience in sketching queries for any kind of SBR system. This way, they are initially unbiased. However, after the first sketching session we gave them some tips on how they must sketch to produce "good queries" for our retrieval system.

5.3.2 Usability Test Session

Our test sessions were made entirely of individual sessions. This method was adopted to minimize the impact of the experiment in users work routine. Each individual session can be summarized in the following steps:

- Description of the experiment;
- Introduction to the SBR prototype;
- Accomplishment of tasks (querying) using two sets of queries;
- Answering to the questionnaire;
- Informal conversation about the SBR system.

During the execution of tasks users were encouraged to make comments (“think loud”) and even raise questions to observers. This form of interaction proved to be very fruitful, since we were able to collect useful information from users. The following steps composed each querying session:

1. Presentation of a set of basic drawings to the user;
2. Ask the user to sketch a query for each presented drawing, submitting it to the system and analyze the returned results;
3. Presentation of a set of simple technical drawings to the user;
4. Ask the user to sketch a query for each presented drawing, submitting it to the system and analyze the returned results.

After the querying session each user answered to a questionnaire about their previous experience in drawing, their opinion about the SBR system and about its user interface. At the end of each session we had an informal conversation with the user focusing the concepts beyond SBR the functionality of the prototype and its application to real situations in the mould industry. From these conversations we gathered relevant information, since users were not constrained by previously prepared questions.

5.3.3 Drawings Database

The drawings database used in this experiment was constructed by joining two sets of drawings, one with basic drawings and the other with simple technical drawings of plates, totalizing 78 drawings. Some instances of these drawings are simply rotated or inverted

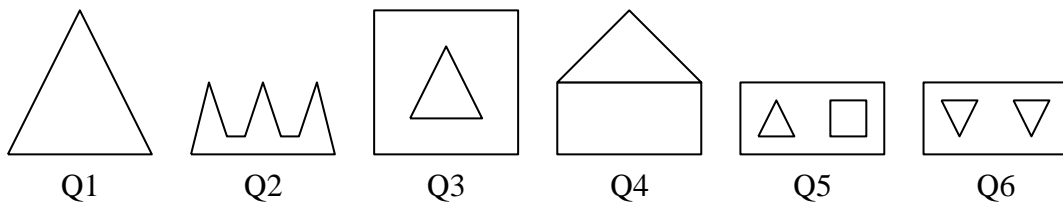


Figure 5.8: Basic drawings to search in the database.

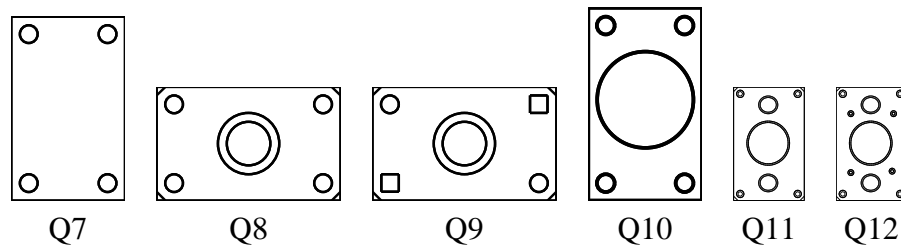


Figure 5.9: Simple technical drawings to search in the database.

versions of others in the database, because I intended to check that the retrieval algorithm is orientation independent.

The set of basic drawings has 38 elements, composed mainly by simple shapes or combinations of two or three simple shapes, as depicted in Section D.1.

The set of simple technical drawings has forty elements, which are simplifications of technical drawings of mould plates. Some plates just have small differences between them like, for example, an additional small triangular cut on the corners. Others are composed by the same shapes, but with different sizes. A complete list of drawings in this set are depicted in Section D.2.

5.3.4 Queries

In this evaluation, users were asked to search for twelve drawings in the database, using the SBR system. First, users sketched six basic drawings, depicted in Figure 5.8, and then they sketched more six simple technical drawings, listed in Figure 5.9. We choose this order of tasks, to provide some training to users while sketching basic drawings. This way, when users perform the second task they already are used to the prototype.

5.3.5 Final Questionnaire

The questionnaire is divided in three parts and can be found in Annex C. One was designed to collect general information about users experience on sketching, drawing tools usually used and input devices preferred and most used. In the second part we questioned users about the use of the SBR prototype to retrieve technical drawings. Users were asked about the time spent to get results and about the quality of results. Finally, there are a set of questions to evaluate the user interface in terms of window layout, size of the main areas and icons of buttons. The time to complete the questionnaire was approximately ten minutes.

5.3.6 Sketches

Sketches made by users to query the database are presented in this section and some of them are depicted. The Sketch-Based Retrieval Prototype stored these sketches automatically, for later analysis. All sketched-queries made by users and respective results returned by the SBR Prototype are depicted in Annex E.

5.3.6.1 Basic Drawings

Sketches drawn to search for a triangle have noticeable differences from the original shape. Among these differences we highlight the fact that users ignored angles and aspect ratio of triangles. This situation occurred in sketch S1A drawn by user A. However, our algorithm was always able to return the correct drawing within the first ten results. In this case the correct result was the first one, as depicted in Figure 5.10.

When users had to sketch a more complex shape, for instance the Chantal's Comb (Q2), it was clear that they had some difficulty in representing it at the first try. While sketching this shape, users used the "undo" command very often, because they made a lot of mistakes.

To execute queries for Q3 and Q4 users had to draw a triangle and a square or rectangle. As in Q1 the triangles sketched for Q3 and Q4 did not respect angles or aspect ratio. Moreover, users did not distinguish sketching a rectangle and a square, as illustrated in Figure 5.11, where user C sketched the rectangle and the square similarly.

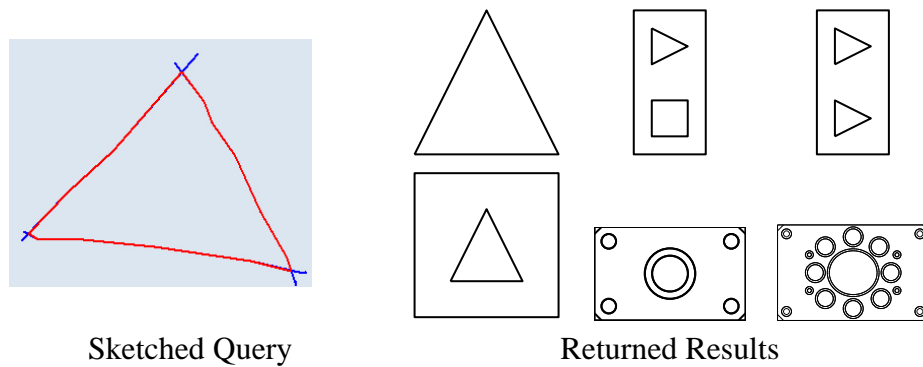


Figure 5.10: Sketch for Q1 made by user A and returned drawings.

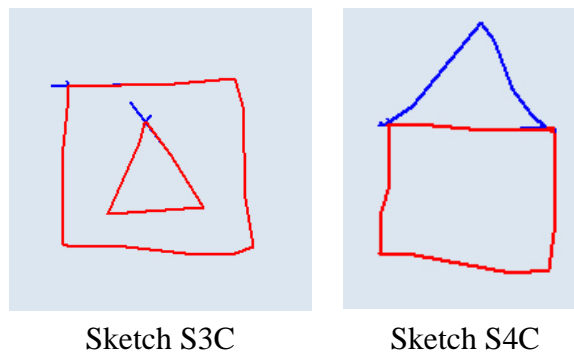


Figure 5.11: Sketches made by user C for Q3 (left) and Q4 (right).

The lack of accuracy in sketches had a major effect in the query for Q3, since the desired drawing never appeared in the first place of the returned results. On the other hand the query for Q4 produced good results despite the inaccurate sketches. In this case, sketch S4C from user C produced a good set of results, with the wanted drawing in first place.

In queries for Q5 and Q6 users searched for similar drawings with a small geometric difference, one had two triangles inside a rectangle while the other had a square and a triangle. This difference was clearly represented in sketches produced by users. Additionally to the lack of accuracy factors referred previously, users also had no concerns about shape alignments, since in original drawing triangles were aligned and have the same size, unlike the correspondent sketches.

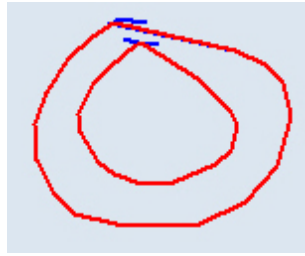


Figure 5.12: Sketch of two concentric circles.

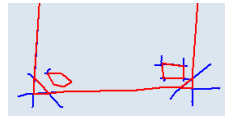


Figure 5.13: Detail of the sketch made by user A for Q9.

5.3.6.2 Simple Technical Drawings

The major problem encountered by users when sketching a query for Q7, and for the rest of the drawings, was the accuracy used to draw circles. However, after a few tries users successfully sketched what they wanted. Despite problems encountered during sketching, users were very pleased with results returned by the first SBR prototype.

Sketching a query for Q8 highlighted the problem of drawing circles, because users represented all six circles of the original drawing. The task of sketching these circles was error-prone and sketched circles were just rough approximations of a real circle, as depicted in Figure 5.12.

Despite the lack of accuracy in sketches for Q8, the SBR prototype always returned the ought drawing in first place. Moreover, the first four results were the same for the three different queries (one for each user).

The major difference between Q8 and Q9 was that users had to draw two squares instead of two circles in the corners of the part. In spite of roughly sketched circles, squares were easily identified, since users sketched them differently, as shown in Figure 5.13. Here is visible how user A sketched an inaccurate circle but drew the square more carefully. Indeed the sketched square was closer to a rectangle than to a square but the resemblance is greater than that between the sketched circle and the circle itself.

In all sketches users respected the topological relationships among shapes. Sketches

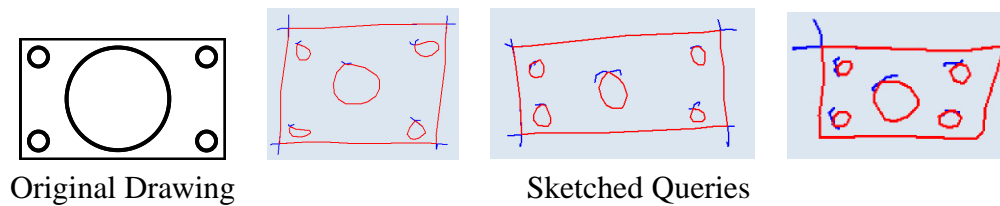


Figure 5.14: Original drawing (Q10) and sketches performed by each user.

made for query Q10 were a good example of this, because regardless the accuracy of the shapes geometry, users drew shapes taking in account their relative position. In this case all users sketched the outer rectangle and the five circles with little geometric accuracy but respecting the placement of shapes in the original drawing, as depicted in Figure 5.14.

Drawings to search for Q11 and Q12 were more complex to sketch than previous ones, because both had several circles with different sizes and some of them were concentric. Although users did not represent all the circles, our system returned the wanted drawing within the first eight results. Typically, users only sketched one of the concentric circles.

5.3.7 Test Analysis

During test sessions each user performed a set of sketched-queries, with the objective of obtaining the complete filename of each searched drawing. To achieve this, users had to locate the wanted drawing in the result list returned by the SBR prototype.

Since sketches are rough approximations of original drawings, it is acceptable that the wanted drawing is not the first returned result. Therefore, I checked the position of the desired drawing within the returned results for each query and summarized it in Figure 5.15. From the analysis of the depicted chart it is clear that in the majority of the queries, wanted drawings were in the first five returned results. Moreover, it is possible to see that in all queries, except one, the desired drawing was within the ten first results, giving some trust to the user.

These results report to successful queries: queries that produce a set of results that include the desired drawing. Unsuccessful queries occurred mainly due to incorrect polygon identification, caused by the lack of accuracy in sketches.

Another measure used to evaluate the prototype was the number of necessary sketched queries to achieve the desired drawing. The correspondent values are presented in Fig-

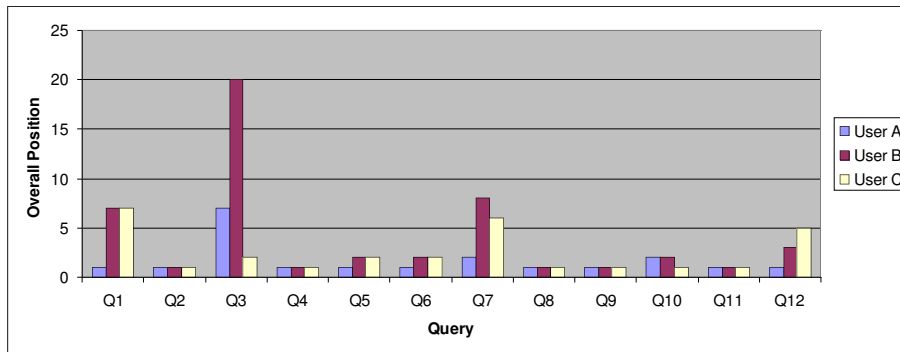


Figure 5.15: Overall position of the desired drawing in the results list.

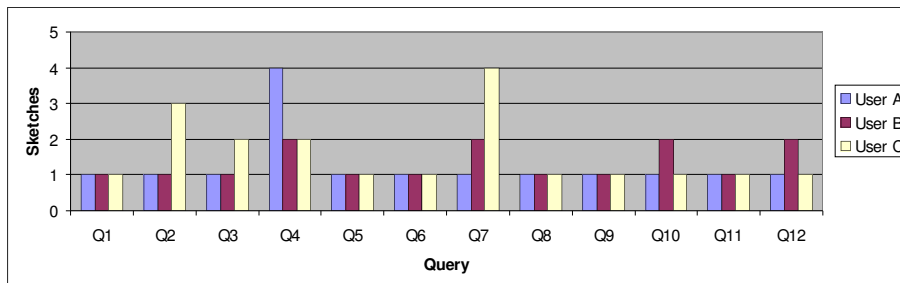


Figure 5.16: Number of sketches drawn before finding the correct result.

Figure 5.16. Looking at this chart one can observe that in the majority of the cases there is a successful query with the first sketch. However, there are some situations where users had to repeat the sketch. Most of the times another iteration was enough to achieve a successful query. Even in the worst situation users re-sketch the query only three times.

To be useful a SBR system must provide good results in a short time. To that end we measured the total time that includes the sketching and the query execution time. Values for each query are shown in Figure 5.17. Query execution took from two to ten seconds in the Tablet PC used in this evaluation. From this graphic it is clear that almost all queries take less than one minute.

Defining what is considered a short time was one of the purposes of this usability test. From informal conversations with users and analyzing their feedback during the sketching session I noticed that they were satisfied with the amount of time spent sketching and waiting for results.

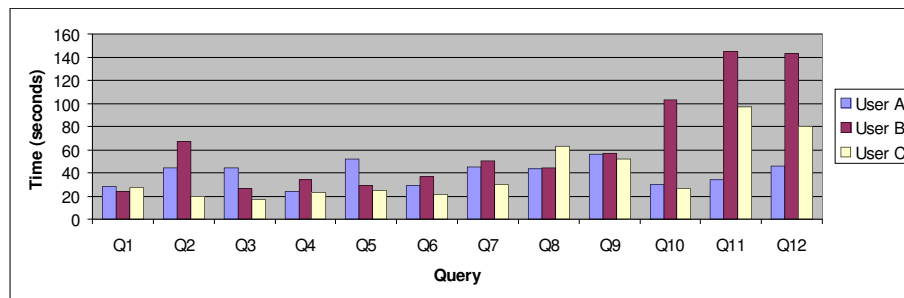


Figure 5.17: Time spent performing queries.

5.3.8 Questionnaire analysis

The first set of questions in the questionnaire (see Annex C) of these evaluation tests was meant to know previous experience of the user and what kind of input devices they use and prefer to use. From results, we can conclude that users have a good experience with CAD tools, for 2D, 3D and modeling. Moreover, we can say that users usually use the mouse as input device.

From the second part of the questionnaire, we tried to extract some feedback from users about the performance of the SBR prototype to retrieve technical drawings using sketches. Results show that users like the idea of specifying a query using sketches, that the quality of results and the time needed to find the desired drawing were very satisfactory.

Finally, from the third part of the questionnaire, we wanted to know what users think about the user interface of the SBR prototype. All users considered the drawing area too small and the buttons area too big, but they liked the general layout of the window. Regarding the presentation of results, users referred that it was difficult to locate at a glance a specific drawing among ten similar ones. They suggested displaying only five results at a time.

5.3.9 Conclusions from preliminary usability tests

In the preliminary usability evaluation we tried to evaluate the main retrieval algorithms and the user interface of the SBR prototype. To that end we involved three users from the mould industry that performed some sketching tasks to search for drawings using our SBR prototype. We also asked users to answer a questionnaire in order to identify

their profile and to get some feedback about prototype functionalities and user interface.

One of the things that I observed during the execution of tasks was that users did not care about where in the order of retrieval the intended drawing appears, the important fact being that it was there. One of the users produced this comment:

”It [the SBR system] found it [the drawing]! That is what counts!”

Although we just involved three users in our preliminary usability test, I think that the main objectives were achieved. We presented a first version of the SBR prototype to users from the mould industry and received great feedback from them. Users liked the interaction paradigm, were satisfied with the results returned by SBR prototype and were pleased with the short time they have to spend to get what they want.

From users’ comments and suggestions, and from observations, I was able to improve the prototype and algorithms. In next section I present the results of the final evaluation tests made with the evolved version of the prototype and using a larger number of users and a larger database of drawings.

5.4 Final Usability Evaluation

This chapter describes the experiment and presents the results of the final usability evaluation. The overall objective of these tests was to evaluate the changes and new features added to the new version of the prototype, developed taking into account the results from the first usability tests. A major goal of this evaluation session was to get feedback from users about the new functionalities, the revamped user interface and the quality of results.

5.4.1 Users

To perform this experiment, we conducted test sessions with five draftspeople from CENTIMFE and a designer from Aníbal Abrantes³. All of them have a good know-how on drafting with CAD tools. Three users have no prior experience in sketching queries

³Aníbal Abrantes is a pioneer company on the Portuguese mould industry. Founded in 1946, is known as ”Mould University” and had already produced about 8000 molds for more than fifty different countries



Figure 5.18: User and observer during the training phase.

for any kind of SBR system. The other three had already participated in the preliminary evaluation tests, knowing the SBR system. We gave some tips to novice users on how to sketch "good queries" for a retrieval system. Additionally, we made a brief demonstration of the new functionalities to all users, so they can take full advantage of this second version of the SBR system, the SIBR prototype.

5.4.2 Usability Test Session

Each usability test session was divided in four distinct parts. In the first part we explained the experiment and introduced the SBR prototype to users. Next, they perform four simple queries to become familiarized with the system. During the third part users executed a set of queries, searching for technical drawings. Finally, users answered a questionnaire, where we try to figure out their profile, their opinions about the prototype and their evaluation of the user interface. We also asked users, in an informal manner, about their opinions suggestions and ideas.

Users were encouraged to make comments ("think loud") and even raise questions to observers, in the training phase and during the execution of tasks. This form of interaction proved to be very fruitful, since we were able to collect useful information from users, which were not constrained by previously prepared questions.

During the querying session we presented a printed drawing to users and asked them to sketch a query for that drawing, to submit it to the system and to analyze the returned results. If the desired drawing did not appear in the set of twelve results the user was

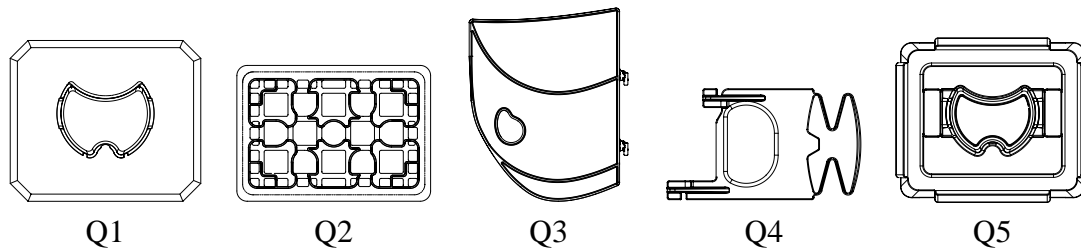


Figure 5.19: Technical drawings selected as queries.

free to enrich the query with additional shapes, delete some of the existing or start a new sketch from scratch. These steps were repeated for each of the five drawings.

5.4.3 Drawings Database

The drawing database used in this experiment was constructed by joining three sets of drawings, one with basic drawings, other with simple technical drawings of plates and another one with twenty technical drawings of parts, totalizing 98 drawings. The first two sets of the database are the same that were used in the first tests and the third set was extracted from complex mould drawings. These three sets of drawings are depicted in Annex D.

5.4.4 Queries

During these tests, drawings Q1 to Q5, depicted in Figure 5.19, were show to users in this order. After presenting each drawing to users, we asked them to search for it using the SIBR prototype and sketched queries.

We start by presenting to users one of the simplest part drawings of the database (Q1). Then a more complex drawing was presented (Q2). However, the complexity of this drawing is relative because it is composed of a set of similar shapes and with a clear symmetry. Next, we present a drawing that had a few small shapes mixed with larger ones (Q3). The penultimate drawing presented to users (Q4) had several shapes, very different among them and lots of small details. Finally, a drawing very similar with the first one but more complex was presented to users (Q5).

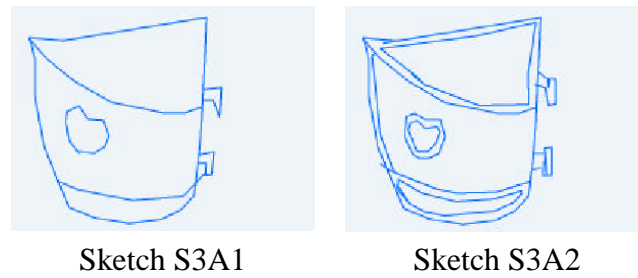


Figure 5.20: Sketched queries made by user A for Q3.

5.4.5 Final Questionnaire

In this final evaluation tests we use a questionnaire similar to the one used in the preliminary tests. The main differences are related to the prototype new functionalities, about which some questions were added. Users took approximately ten minutes to complete the questionnaire. Its content can be found in the Annex C.

5.4.6 Sketches

In this section I present the sketches made by users to query the database, the most relevant sketches and some sets of results returned by our system. The complete set of sketched queries and correspondent results are depicted in Annex E.

When searching for a drawing, the better way to find it is sketching a query depicting its more relevant features. If the user tries to perfect the sketch by adding more features, but fails to draw a precise representation of the drawing it will be less similar to it. Figure 5.20 depicts such case, where the user sketched a simple query (S3A1: first sketch made by user A when searching for query Q3) with a few strokes and the correct drawing were returned within the top ten results but, when the sketched drawing was changed by adding more features (S3A2), the correct drawing were returned only in the 12th position, because the new sketch is less accurate.

However, even the main features need to be represented with some precision. If only a rough approximation is sketched, the results might be unsatisfactory. This way it is possible that two apparently similar sketches produce completely different results.

Nevertheless, a sketched query for a drawing does not need to have all elements. Each drawing has a set of relevant shapes that characterizes it. If the user is able to identify such

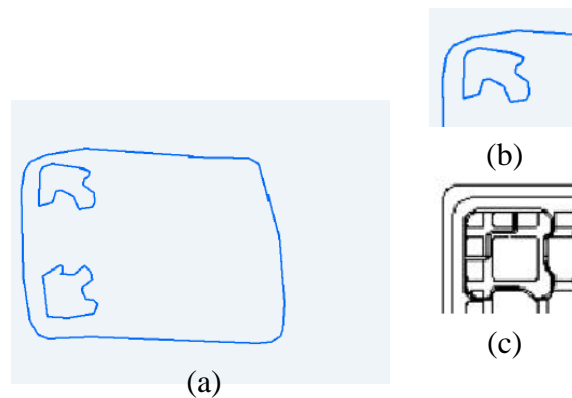


Figure 5.21: Sketch S2C (a) made by user C for Q2 and details of sketch (b) and drawing (c).

shapes it will be enough to sketch them in order to obtain successful search results.

Sketch S2C, depicted in Figure 5.21 (a), illustrates an example of a situation where a good capacity to understand which shapes distinguishes a drawing from all others enables the user to quickly sketch a simple query that produces a good result. In this case was sketched the outer rectangular shape and two inner shapes. User C has been especially careful when sketching the inner shapes, easily perceived by comparing details, depicted in Figure 5.21, of sketch drawn (b) and wanted drawing (c).

The methodology for specification of queries used in sketch S2C proved well with proposed algorithms. In this case, the SIBR system returned ought drawing in first place, which is an excellent result for a sketch with only three strokes that was meant to describe a very complex drawing.

Therefore, to sketch a query that produces good results it is important that the user is able to identify relevant shapes and then sketch them with some precision. If relevant shapes are sketched roughly the quality of the results may decay. Sketches made by users when searching for drawing Q4 demonstrate that.

Several different shapes with small details compose drawing Q4. Consequently, the identification of relevant shapes is not straightforward and the irregularity of most shapes makes precise sketching difficult. However, users A and D were able to sketch successful queries at first attempt (see Figure 5.22, sketches S4A and S4D) while user E needed five iterations before submitting a successful query to the system, depicted in Figure 5.22, sketch S4E5.

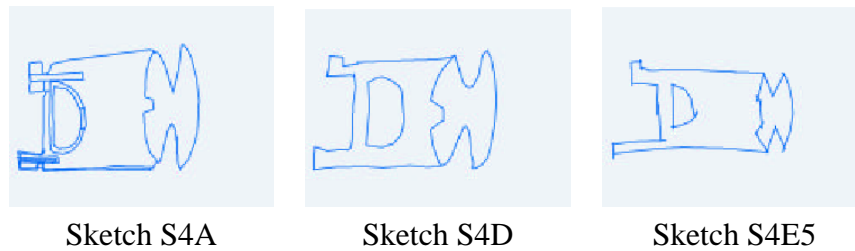


Figure 5.22: "Good" sketches made for Q4 that returned successful results.



Figure 5.23: Sketches made for Q4 that returned unsuccessful results.

User A sketched Q4 accurately (S4A), representing all main features of the drawing with precision. As a result of this careful sketching, the SIBR system returned the wanted drawing in second place, compensating the large number of strokes used and the time spent. On the other hand, user D focused on distinctive features of wanted drawing and sketched them precisely with few strokes (S4D). Such query returned the correct result among the top ten. Similar result was achieved when sketch S4E5 were submitted to the SIBR system, which is not surprising since both users identify the same relevant shapes and sketch them similarly.

Problems commonly arise when users are unable to sketch relevant shapes with a minimum accuracy or when they do not even identify correctly the relevant shapes. Some examples of such situations are illustrated in Figure 5.23. When the depicted sketched queries were submitted to our SIBR system the wanted drawing was not among the top twelve returned results.

User C tried three different queries for Q4 before sketching S4C4. Despite a correct identification of relevant shapes in all four queries, the lack of precision in sketching yielded only unsuccessful queries. As a result of these misses, user decided to give up looking for drawing Q4 in the database, assuming that it does not exist there.

A different problem occurs with user E, which initially fails to identify the more

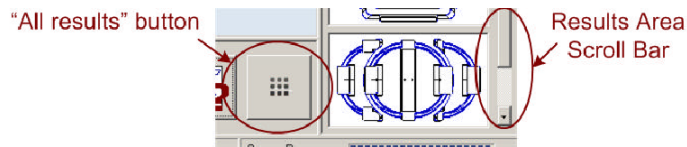


Figure 5.24: Detail of second SBR prototype showing the "Show all results" button and part of results area scroll bar

relevant shapes on Q4. Consequently, only two shapes compose sketch S4E2, one inner half circle and an outer contour. Despite the inner half-circle could be considered as accurate the outer shape is too distinct from the contour of the wanted drawing. Moreover, drawing Q4 is clearly divided in two adjacent outer shapes, but user E ignores that fact and only sketches one outer shape.

Since using S4E2 as a query does not yielded successful results, user E added more features to it and produced sketch S4E4. Now the more relevant features are already identified, but too roughly sketched to obtain good results. Although its failed fourth attempt, user E decided to make other iteration. Understanding that adding features or making small changes to current sketch will not be enough, user E started a new one from scratch. This sketch contains the three relevant shapes of Q4 represented with good accuracy. Submitting this fifth sketch as a query to the SBR system yielded ought drawing among the top ten results.

5.4.7 Test Analysis

During these test sessions each user performed a set of sketched-queries, with the objective of identifying each searched drawing in the results area of the prototype or alternatively in the "All Results" window. We choose to allow these two options because all results can be consulted either using the scroll-bar on results all area or clicking on a control button that opens the "All Results" window, which displays simultaneously all twelve returned results.

Observing users behavior when the wanted drawing was not in the top five results, we noticed that they prefer to open the "All Results" window rather than use the scroll bar to change the displayed drawings in the results area. A possible reason for this might be the relative small width of the scroll bar, when compared to the button size (see Figure 5.24).

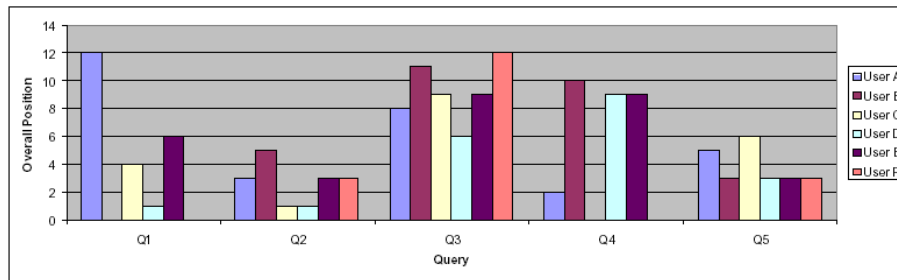


Figure 5.25: Overall position of the desired drawing in the results list.

Moreover, some users complained about the short width of the scroll bar and the difficulty they had when trying to use it. It is well known among user interface designers that small targets are difficult to click and users prefer to use the larger ones. Therefore, the correct solution will be to increase the width of the scroll bar, or even remove that feature, since users prefer to view all twelve results in a separate window when sought drawings are not among the top five.

Additionally, I checked the position of the desired drawing within the returned results for each query and summarized it in Figure 5.25. In this chart unsuccessful queries are ignored since it only considers queries that returned the wanted drawing among the top twelve results.

Contrary to the preliminary evaluation tests, wanted drawings were not in the top five returned results for the majority of the queries. Now the overall position of the desired drawing in the results list is distributed more evenly. A possible cause for this might be the fact that now we have complex drawings in the database. Since complex drawings share features, a specific query might be similar to more than one drawing.

However, 47% of all submitted queries returned the desired drawing in the top five results, as depicted in Figure 5.27 (b). If I only consider successful queries, then more than a half of them yielded the wanted drawing in the top five results.

In the current analysis I consider a miss when the user was unable to submit a query that returns the desired drawing among the top twelve results. During this second evaluation tests the retrieval system returned the correct result for almost all queries. Only three times the SIBR prototype fails to find the desired drawing. Two of these misses were when searching for Q4, which proved to be a problematic drawing due to its complexity and to the difficulties users had in identifying its relevant features.

While three users made its first contact with the SBR system during these second evaluation tests, the other three had already participated in the first evaluation tests of the SBR system. Therefore, we decided to classify users in two different categories according to its experience using an SBR system. Users C, D and E already know the system and are acquainted with sketching queries. On the other hand, users A, B and F have no prior experience in using sketches to search for technical drawings.

Considering users experience, I analyzed the results yielded by submitted queries. In Figure 5.26 we depict the average overall position for each query, grouped by user experience. It is possible to see that queries made by experienced users produce better results, in average, than the ones submitted by first-time users.

Additionally we measured the number of necessary sketched-queries to achieve the desired drawing. Trough the analysis of obtained data I concluded that in more than fifty percent of the cases the wanted drawing was obtained with only one query (see Figure 5.27 (b)). Moreover, almost all successful results were achieved with less than three queries. Using the queries and the database described above, the SIBR system was able to find the ought drawing in 90% of the queries, which satisfies users (see next section).

During these tests we choose not to measure the time each user took sketching, since we prefer to gather their comments and it will be difficult to account time while talking.

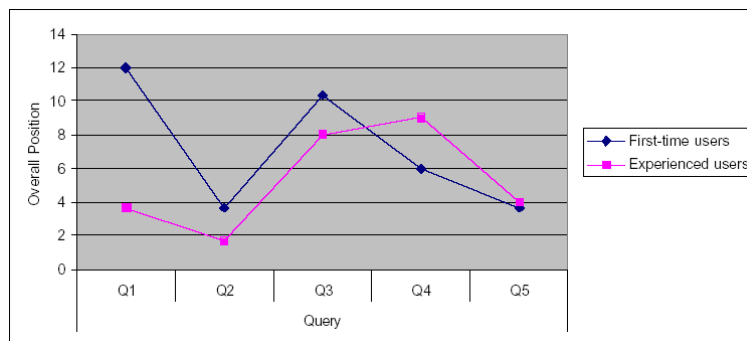


Figure 5.26: Average overall position of query results grouped by user experience.

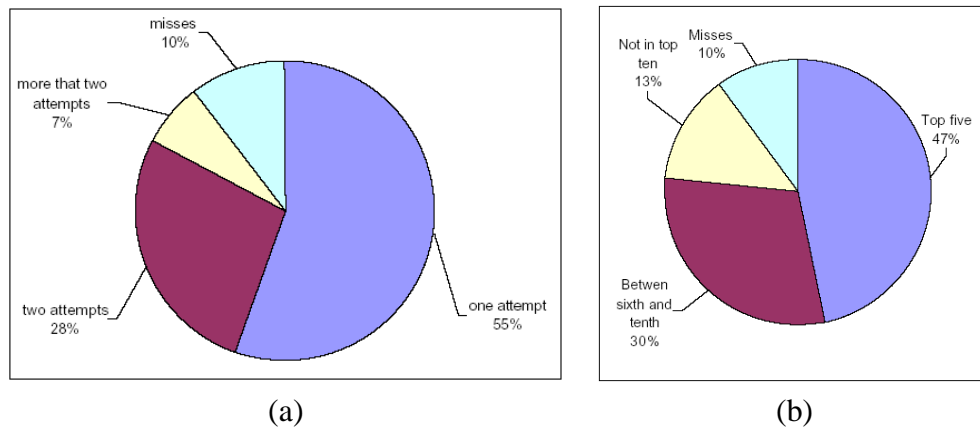


Figure 5.27: Pie charts representing query results distribution (a) and number of sketches drawn before finding a correct result (b).

5.4.8 Questionnaire Analysis

After each sketching session we ask users to answer to a questionnaire . Due to time restrictions, the last user skipped this step of the evaluation tests. Therefore we only have five answered questionnaires.

The questionnaire, listed in Annex C, was divided in three parts. In the first part we intend to gather information about users' experience. From obtained results, we know that all subjects are CAD users, with experience in both 2D and 3D design. Additionally, we also know that all users have previous experience with sketching tools, but not in their everyday work.

From the answers given by users when asked about input devices we conclude that keyboard and mouse are the most used. This conclusion is coherent with the fact that CAD is the most commonly used tool, since the interaction with this kind of system is made mainly trough keyboard commands and mouse point-and-click operations.

Despite the common use of keyboard, when we ask the users about which device they prefer it was not pointed by them. Users clearly prefer to use the mouse as input device, but the current computer aided design tools force them to use intensively the keyboard.

The second set of questions aims to extract some feedback from users about the concept of sketch-based retrieval and the performance of the prototype. Results showed that users are very pleased with the possibility of using sketches to retrieve technical drawings and are even more interested in the new feature of using existing drawings to help in query

specification.

Finally, the last set of questions focused on the user interface. According to obtained results, users are pleased with the user interface of this final prototype. The main complains are about the icons used on the "Show All Results" and "Acquire Image" buttons, the absence of clues about the delete gesture and the selection method.

5.4.9 Conclusions from final usability tests

In this chapter, I have described the usability tests for the second Sketch-Based Retrieval prototype. The improved retrieval algorithm and the revamped interface were meant to be evaluated by designers or draftspeople. To that end we involved six users from the mould industry that performed some sketching tasks to search for drawings using the SIBR prototype. After each sketching session we asked users to answer a questionnaire in order to identify their profile and to get some feedback about prototype functionalities and user interface.

We intend to involve a large number of users in these tests, but it proved to be an impossible task, since it is very hard to find designers or draftspeople available to engage in this kind of activities. However, we consider that our main goal for this user evaluation was achieved. Users were satisfied with the interaction paradigm and with the concept of using sketches to retrieve technical drawings. Moreover, they were very pleased with the new functionalities provided by the final prototype, namely the query using drawings on paper and the query-by-example.

Suggestions and comments made by users during these tests in conjunction with our observations will lead to improvements in this final prototype and in the retrieval algorithms, pointing out directions for future work.

5.5 Discussion

During the development of a system for retrieving technical drawings using sketches and images we invested on a strong collaboration between the research team and the final users. To that end we organized three experiments involving draftspeople from the mould industry.

Initially, a sketching experiment took place at CENTIMFE on July 2003. With this experiment we intended to study the way users sketch technical parts, in order to figure out how our retrieval system can take advantage of users' natural ability at sketching. Using a sketching prototype developed specifically for this experiment and involving three draftspeople we gathered important information that was used to create the first version of our Sketch Based Retrieval prototype.

From the analysis of the sketches made by users during the sketching sessions I noticed that all users identified the same relevant shapes of a drawing. This fact validates proposed approach of using sketches to specify queries on a database of technical drawings. Additionally, we observed that some users sketch accurately and with lots of details, unless we tell them to be less precise and ask them to sketch only what they consider relevant. Therefore, users will need some training before using efficiently the SIBR system.

Employing the information gathered in the sketching experiment, I developed the SBR prototype. In September 2003 we organized a preliminary usability evaluation to this prototype. With this experiment we tested the performance and accuracy of retrieval algorithms and evaluated the user interface. This experiment involved the same three draftspeople that engaged the previous one. They performed some sketching tasks to search for drawings using our SBR prototype and then answered to a questionnaire.

The first usability evaluation revealed that users were pleased with the results associated with their sketched queries. Moreover, from informal conversations and questionnaire analysis we noticed that users were satisfied with the amount of time spent sketching and waiting for the results. However, the drawings and queries used in these tests have relatively low complexity.

Users' comments and suggestions alongside with observations we made during the first user evaluation lead to major changes in the prototype. A revamped user interface and a set of new features were incorporated in the final SIBR prototype. Furthermore, a completely renewed feature extraction component was developed. The second prototype underwent a usability evaluation in July 2004. This second usability evaluation was similar to the first one, but now we used a larger database with more complex drawings and a new prototype with additional functionalities.

Analysis of the information gathered during the second usability evaluation tests

showed that we achieved a successful evolution from the first to the second prototype. Despite some complaints about the "select region" procedure, users were generally pleased with the interface. Besides, the improved retrieval algorithms returned accurate results within an acceptable time when searching for complex drawings in the database.

Unfortunately, we were unable to construct a really large database, with several hundreds of complex drawings, since we only have restricted access to technical drawings from the mould industry. Currently our database contains about one hundred drawings and only twenty of them are real technical drawings of mould parts.

Although results from usability evaluation tests were very promising, I think that the number of users involved and the size of the database used did not come up to my expectations. I would like to have had performed tests with thousands of drawings and dozens of users, to state with more confidence that proposed system has good performance and accuracy for large sets of drawings.

5.6 Summary

In this chapter I presented the three user tests performed during the development of my work. In the first tests I studied how real users sketch queries to search for technical drawings. During these initial tests we gathered sketches made by draftspeople and ask them some feedback about the concept of retrieve drawings using sketches. At the end, we conclude that users were pleased the idea of sketch-based retrieval. In addition, we assert that they were able to quickly sketch a query for a complex engineering drawing. From the analysis of collected sketches we identified the features of an engineering drawing that users consider relevant to draw in a query.

Based on results of these first sketching experience, I developed a prototype that was able to retrieve vector drawings from a database. A preliminary usability evaluation were performed in order to test classification and retrieval algorithms and collect users' opinions about the results returned by the system and suggestions for improvements. Users liked the interaction paradigm, were satisfied with the results returned by the SBR prototype and were pleased with the short time they have to spend to get what they want.

From comments and suggestions provided by users, and from our observations, I was

able to improve the prototype and algorithms, producing the final SIBR application. This prototype was also evaluated by users. In this final usability evaluation we engaged six draftspeople and use a database with little less than one hundred drawings.

Despite some minor complaints regarding some interaction methods, users were pleased with the interface and with the results yielded by the retrieval algorithms. From this final tests we concluded that this paradigm for retrieval of technical drawings will work in a real environment and that our algorithms are able to fulfill users' expectation regarding speed and accuracy.

6

Conclusions and Future Work

This chapter presents the final conclusions, identifying the most relevant contributions of my work and discussing the strengths and weaknesses of algorithms and methodologies presented in this thesis. In addition I point possible directions for future work in each field.

My main goal was to develop a set of techniques to retrieve technical drawings from databases using sketches as queries. As a result of task analysis and interaction with them, this goal was broadened to allow query specification by combining images with sketches. To achieve it I took advantage of the framework presented by Fonseca in his PhD work [34] to retrieve technical drawings from large databases, focusing my research on algorithms and usability issues postponing efficiency concerns to future work.

In order to yield a novel method for retrieving 2D engineering drawings, I developed techniques to extract visual features from technical drawings and sketches and integrate them with Fonseca's framework for retrieval in large sets of drawings. Thus, my thesis focused on vectorization and simplification of engineering drawings, shape detection and feature extraction.

During this research I devised a methodology for vectorizing engineering drawings that proved to be simultaneously effective and accurate enough. Although this technique uses existing algorithms, the proposed combination of these minimizes user intervention, a desirable goal to perform line simplification, producing good results with simple technical drawings digitized using a flatbed scanner. However, this approach proved difficult to scale to very complex drawings, which take too much time to perform the raster to vector conversion "on-the-fly". Since vectorization is used to specify queries interactively, ideally this process should not take more than a few seconds. Moreover, if scanned drawings

have additional elements such as text, symbols or dimensioning notes, this method would need to perform text/symbols/graphics segmentation, which could be easily added to the vectorization pipeline. Although several algorithms exist for this purpose, their inclusion would add extra complexity to this approach without commensurable returns. Thus, for this work I assumed that scanned drawings have no additional elements, which although acceptable to prove the concept should be revised in a fully functional prototype. Therefore, further improvements regarding strategies for segmentation and recognition of graphical entities in engineering drawings should be studied and integrated in a final application. Moreover, developing a stable and robust stand-alone vectorization process would be a major advance, since no currently known methods can work that way [106].

The curve simplification algorithms integrated in this work were not developed specifically for this purpose. Instead I reused existing ones, while performing only the necessary adaptations. Indeed, there are many distinct solutions for line simplification available today, covering a wide range of applications. However, there is room for improvement. Further research should focus on context sensitive simplification of engineering drawings aimed at classification for sketch-based retrieval.

Despite the considerable body of work in shape detection from raster images, I could not find an algorithm to detect polygons in sets of line segments. Thus, I developed a novel technique to solve this problem in polynomial time and space. I consider this algorithm one of the most important results from my thesis work [32]. It identifies all minimal polygons that can be constructed from a given set of line segments using well-known and simple to implement algorithms to find and remove intersections between line segments and to find a MCB of a graph, instead of using theoretically more efficient but less simple to code methods. For instance, we could adapt the method presented by Hartvigsen and Hardon [49] to find the MCB on planar graphs or the optimal algorithm for detecting line segment intersections in a plane presented by Chazelle and Edelsbrunner [14].

The whole approach to interaction constitutes an important result of my work, since it represents a new interaction paradigm for retrieval of engineering drawings. Three experiments with users, performed during the development of this work, validate the approach and support the concept of mixing sketches with printed images to retrieve existing drawings from databases. Unfortunately the number of designers and draftspeople involved were below my expectations. For instance, at the final usability evaluation only six users

participated in the experience. However, they were generally very pleased with the concept of mixing images and sketches for retrieving technical drawings. Furthermore, most users showed a keen interest on having such a tool available in their daily work and this is the subject of ongoing development. Usability tests revealed that this approach is indeed suited to quickly find drawings using sketches. Moreover, the results yielded by queries shown that the proposed methodology is effective. Further testing with more users and larger databases would yield more conclusive data. However, the indexing structure, that could be a possible bottleneck during retrieval, has shown good performance for data sets with around one million elements [34]. Because of this, I believe that the present approach has the potential to scale up to realistic more settings.

This thesis proposed a solution that combines images and sketches to retrieve technical drawings from large databases. To that end, I used the framework for retrieval in sets of drawings developed by Fonseca [34]. While the presented prototype applications have been developed mostly as concept demonstrators, the strong feedback received from draftspeople makes us confident that they can evolve into a fully functional system integrated in a production environment.

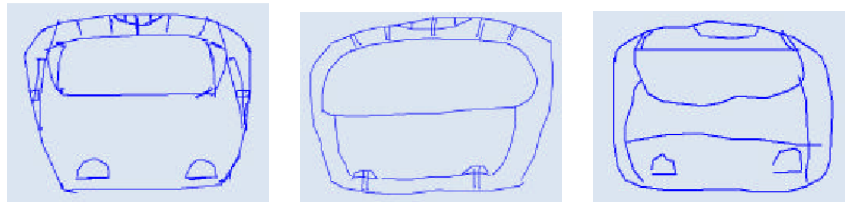
A

Sketches from Sketching Experiment

In this annex I present the sketches made by users during the sketching experiment that took place in CENTIMFE. This experiment was divided in two distinct sessions, as described in Section 5.1. Thus, sketches presented here respect that division.

A.1 First Sketching Session

Sketches representing drawing D1

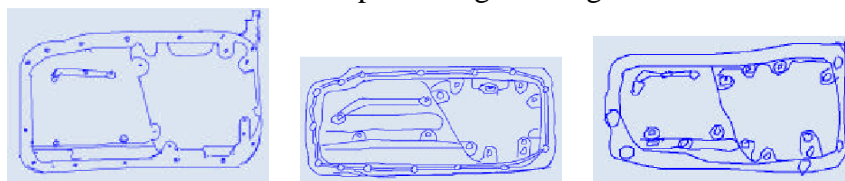


User A

User B

User C

Sketches representing drawing D2

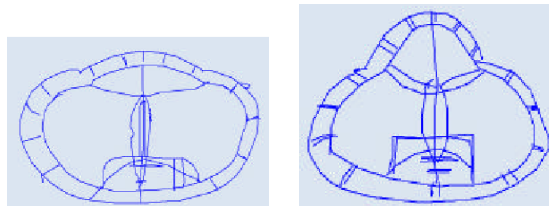


User A

User B

User C

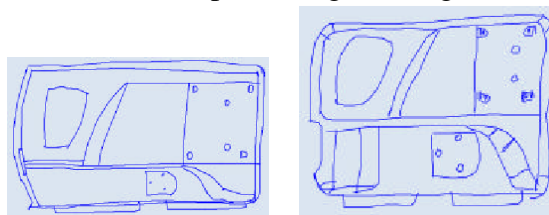
Sketches representing drawing D3



User A

User B

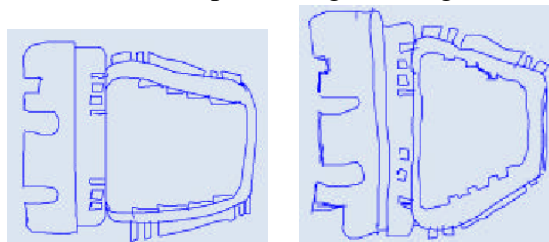
Sketches representing drawing D4



User A

User B

Sketches representing drawing D5

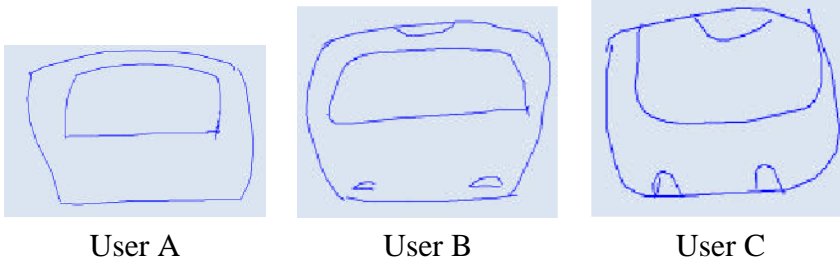


User A

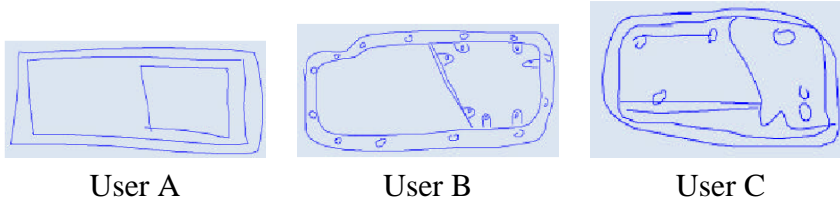
User B

A.2 Second Sketching Session

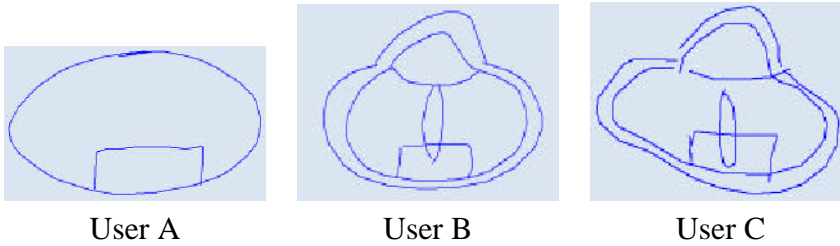
Sketches representing drawing D1



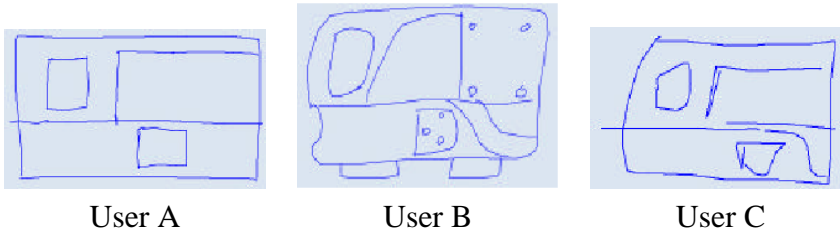
Sketches representing drawing D2



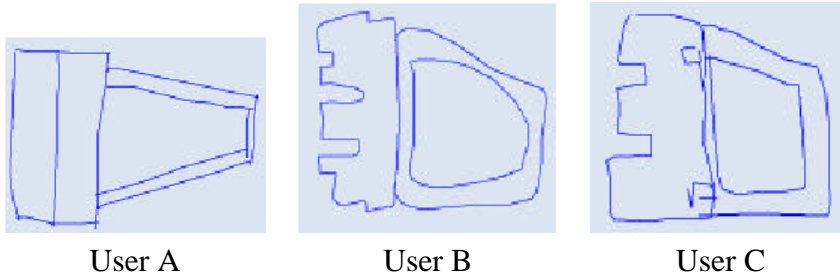
Sketches representing drawing D3



Sketches representing drawing D4



Sketches representing drawing D5



B

Testing Protocols

This annex presents the protocols presented to users during the two usability evaluations, performed at CENTIMFE. First I present the protocol used in the preliminary usability evaluation. Next, I present the protocol for the final usability evaluation.

B.1 Protocol for Preliminary Usability Evaluation

Sketch-Based Retrieval Prototype

Usability evaluation session

Thank you for having accepted to participate in this experiment. Its main objectives are the evaluation of the underlying ideas of our Sketch-Based Retrieval prototype and the validation of its algorithms. This prototype allows the retrieval of technical drawings using sketches to specify the desired drawing.

The schedule of this session is described in the following table, indicating the estimated time for each of the foreseen tasks, with an expected total time of about 1 hour and 30 minutes.

1	Experiment Description	10 m
2	Accomplishment of tasks using our prototype	50-60 m
3	Questionnaire about users, prototype and tasks	10 m

This experiment intends to evaluate the utility and the effectiveness of our prototype, so, all comments and suggestions are welcome. “Think loud” during task execution and do not feel inhibited to point out negative or positive aspects, of the prototype.

To finish, we would like to thank you the time and effort spent.

Consent form

Part of this evaluation session will be videotaped and we would like to include some excerpts in a small film about the system. Please indicate whether you authorize or not the diffusion of the excerpts where you appear:

Yes

No

Name: _____

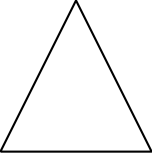


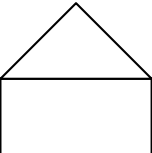
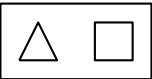
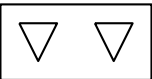
Signature: _____

First set of Tasks

(Simple Drawings)

Duration: 20 min

Please perform the following 6 (six) queries using the Sketch-Based Retrieval System and comment the returned results. Do not hesitate in making comments in loud voice, or in asking for help whenever necessary. Start the construction of each model only after you have been told to do so.

Query	Comments
 Q1	
 Q2	
 Q3	
 Q4	
 Q5	
 Q6	


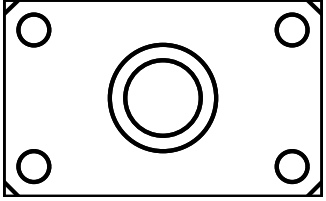
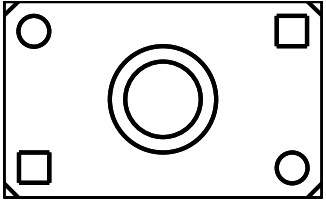
Second set of Tasks

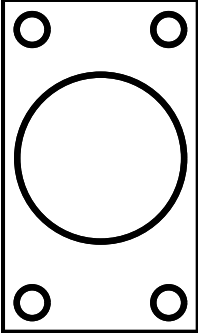
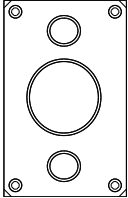
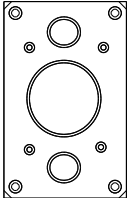
(Plate Drawings)

Duration: 30 min

Please perform the following 6 (six) queries using the Sketch-Based Retrieval System and comment the returned results. Do not hesitate in making comments in loud voice, or in asking for help whenever necessary.

Start the construction of each model only after you have been told to do so.

Query	Comments
 <p data-bbox="352 1352 400 1391">Q7</p>	
 <p data-bbox="352 1637 400 1675">Q8</p>	
 <p data-bbox="352 1921 400 1960">Q9</p>	

Query	Comments
 <p data-bbox="347 790 405 824">Q10</p>	
 <p data-bbox="347 1077 405 1111">Q11</p>	
 <p data-bbox="347 1361 405 1395">Q12</p>	

B.2 Protocol for Final Usability Evaluation

Sketch and Image Based Retrieval Prototype

Usability evaluation session

Thank you for having accepted to participate in this experiment. Its main objective is the evaluation of the underlying ideas of our Sketch and Image Based Retrieval prototype. This prototype allows the retrieval of technical drawings using sketches and images to specify the desired drawing.

The schedule of this individual session is described in the following table, indicating the estimated time for each of the foreseen tasks, with an expected total time of about 50 minutes.

1	Experiment Description and Training	10 m
2	Accomplishment of tasks using our prototype	30 m
3	Questionnaire about users, prototype and tasks	10 m

This experiment intends to evaluate the utility and the effectiveness of our prototype, so, all comments and suggestions are welcome. “Think loud” during task execution and do not feel inhibited to point out negative or positive aspects, of the prototype.

To finish, we would like to thank you the time and effort spent.

Consent form

Part of this evaluation session will be videotaped and we would like to include some excerpts in a small film about the system. Please indicate whether you authorize or not the diffusion of the excerpts where you appear:

Yes

No

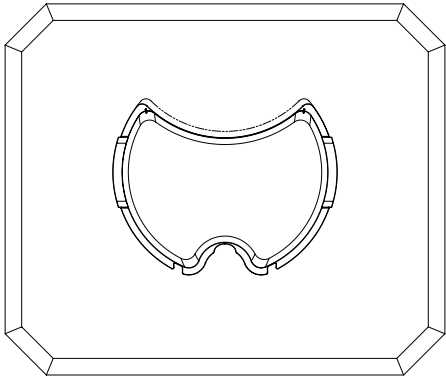
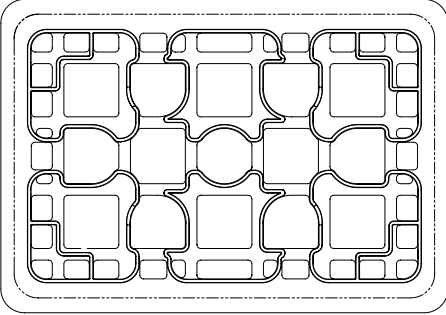
Name: _____

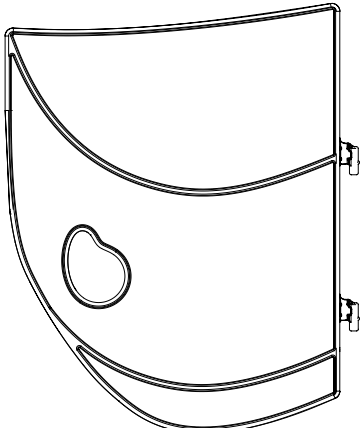
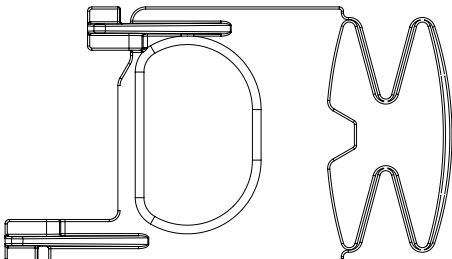
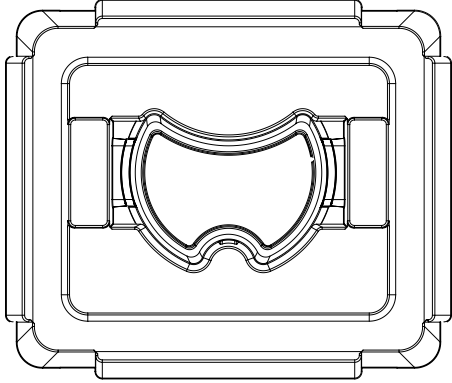
Signature: _____

Sketch-Based Retrieval Tasks

Duration: 30 min

Please search for the following five drawings using the Sketch-Based Retrieval System and comment the returned results. Do not hesitate in making comments in loud voice, or in asking for help whenever necessary. Start the construction of each query only after you have been told to do so.

Query	Comments
 <p data-bbox="464 1207 507 1240">Q1</p>	
 <p data-bbox="464 1617 507 1650">Q2</p>	

Query	Comments
 <p data-bbox="566 817 614 851">Q3</p>	
 <p data-bbox="566 1176 614 1209">Q4</p>	
 <p data-bbox="566 1657 614 1691">Q5</p>	

C

Questionnaires

This annex presents the questionnaires presented to users during usability evaluation tests. First, I present the questionnaire of the preliminary evaluation tests. Next, I present the questionnaire used in the final evaluation tests.

C.1 Questionnaire of the preliminary evaluation tests

Questionnaire

General questions

- 1 - Do you have previous experience on free-hand sketching? Yes No
- 2 - What kind of drawing tools do you usually use? (You can select more than one option)
- a) CAD 2D
 - b) CAD 3D
 - c) Modeling
 - d) Other: _____
- 3 - How long have you been using drawing tools? _____
- 4 - What input devices have you already used? (You can select more than one option)
- a) Mouse
 - b) Pen and tablet
 - c) 3D Mouse
 - d) Keyboard
 - e) Other: _____
- 5 - What input device do you use most for drawing? (You can select more than one option)
- a) Mouse
 - b) Pen and tablet
 - c) 3D Mouse
 - d) Keyboard
 - e) Other: _____
- 6 - What input device do you prefer for drawing? (You can select more than one option)
- a) Mouse
 - b) Pen and tablet
 - c) 3D Mouse
 - d) Keyboard
 - e) Other: _____

Questions about the Prototype

Please characterize the adaptation of our Sketch-Based Retrieval prototype to retrieve technical drawings, according to:

1 - Use of sketches to specify queries. Bad Excellent

Comments:

2 - Number of iterations to achieve the wanted result. Bad Excellent

Comments:

3 - Total time to get the wanted result. Bad Excellent

Comments:

4 - Quality of results. Bad Excellent

Comments:

5 - Critics and suggestions to the Sketch-Based Retrieval prototype:

Questions about the User Interface

Please characterize the user interface of our Sketch-Based Retrieval system, according to:

1 - Size of the Sketching area. Too Small OK Too Big

2 - Size of the Result area. Too Small OK Too Big

3 - Size of the Buttons area. Too Small OK Too Big

4 - Layout of the window. Are the three areas well distributed in the window?

Yes No

5 - Quality of Button Icons

a) Quit button Bad Excellent

Comments:

b) Help button Bad Excellent

Comments:

c) Query button Bad Excellent

Comments:

d) Undo button Bad Excellent

Comments:

e) New button Bad Excellent

Comments:

f) Up button Bad Excellent
Comments:

g) Down button Bad Excellent
Comments:

6 - Functionality Bad Excellent
Comments:

7 - Critics, suggestions or new functions to integrate in the user interface of the prototype:

C.2 Questionnaire of the final evaluation tests

Questionnaire

General questions

- 1 - Do you have previous experience on free-hand sketching? Yes No
- 2 - What kind of drawing tools do you usually use? (You can select more than one option)
- a) CAD 2D
 - b) CAD 3D
 - c) Modeling
 - d) Other: _____
- 3 - How long have you been using drawing tools? _____
- 4 - What input devices have you already used? (You can select more than one option)
- a) Mouse
 - b) Pen and tablet
 - c) 3D Mouse
 - d) Keyboard
 - e) Other: _____
- 5 - What input device do you use most for drawing? (You can select more than one option)
- a) Mouse
 - b) Pen and tablet
 - c) 3D Mouse
 - d) Keyboard
 - e) Other: _____
- 6 - What input device do you prefer for drawing? (You can select more than one option)
- a) Mouse
 - b) Pen and tablet
 - c) 3D Mouse
 - d) Keyboard
 - e) Other: _____

Questions about the Prototype

Please characterize the adaptation of our Sketch-Based Retrieval prototype to retrieve technical drawings, according to:

1 - Use of sketches to specify queries. Bad Excellent

Comments:

2 - Use of drawings on paper to specify queries. Bad Excellent

Comments:

3 - Number of iterations to achieve the wanted result. Bad Excellent

Comments:

4 - Total time to get the wanted result. Bad Excellent

Comments:

5 - Quality of results. Bad Excellent

Comments:

6 - Critics and suggestions to the Sketch-Based Retrieval prototype:

Questions about the User Interface

Please characterize the user interface of our Sketch-Based Retrieval system, according to:

1 - Size of the Sketching area. Too Small OK Too Big

2 - Size of the Result area. Too Small OK Too Big

3 - Size of the Buttons area. Too Small OK Too Big

4 - Layout of the window. Are the three areas well distributed in the window?

Yes No

5 - Quality of Button Icons

a) Quit button Bad Excellent

Comments:

b) Help button Bad Excellent

Comments:

c) New button Bad Excellent

Comments:

d) Open Image Bad Excellent

Comments:

e) View Image Bad Excellent

Comments:

f) Undo button Bad Excellent
Comments:

g) Redo button Bad Excellent
Comments:

h) Query button Bad Excellent
Comments:

j) Show All Results button Bad Excellent
Comments:

6 - Editing Options Bad Excellent
(Delete, Select, Add, Query-by-Example)
Comments:

7 - General Functionality Bad Excellent
Comments:

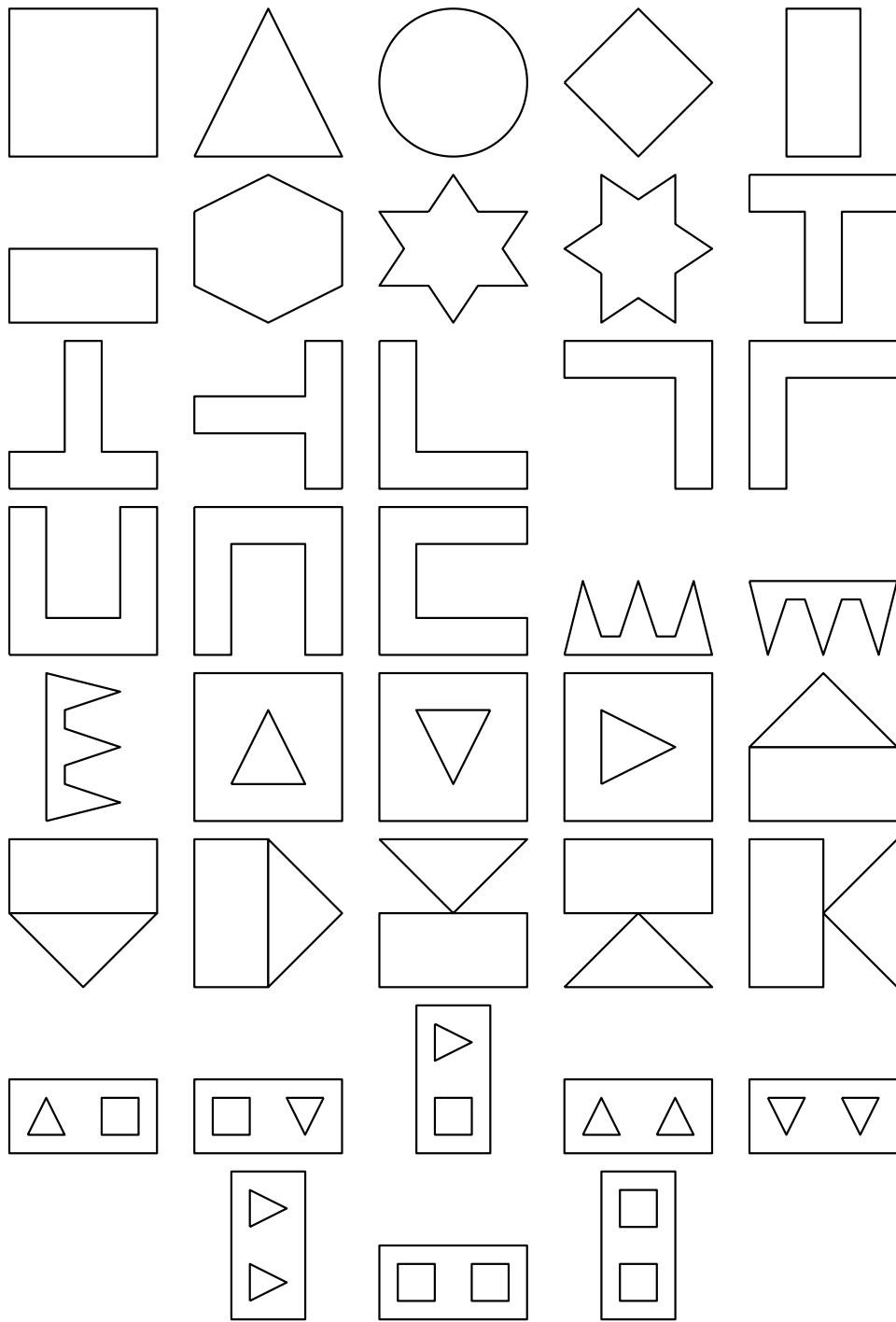
8 - Critics, suggestions or new functions to integrate in the user interface of the prototype:

D

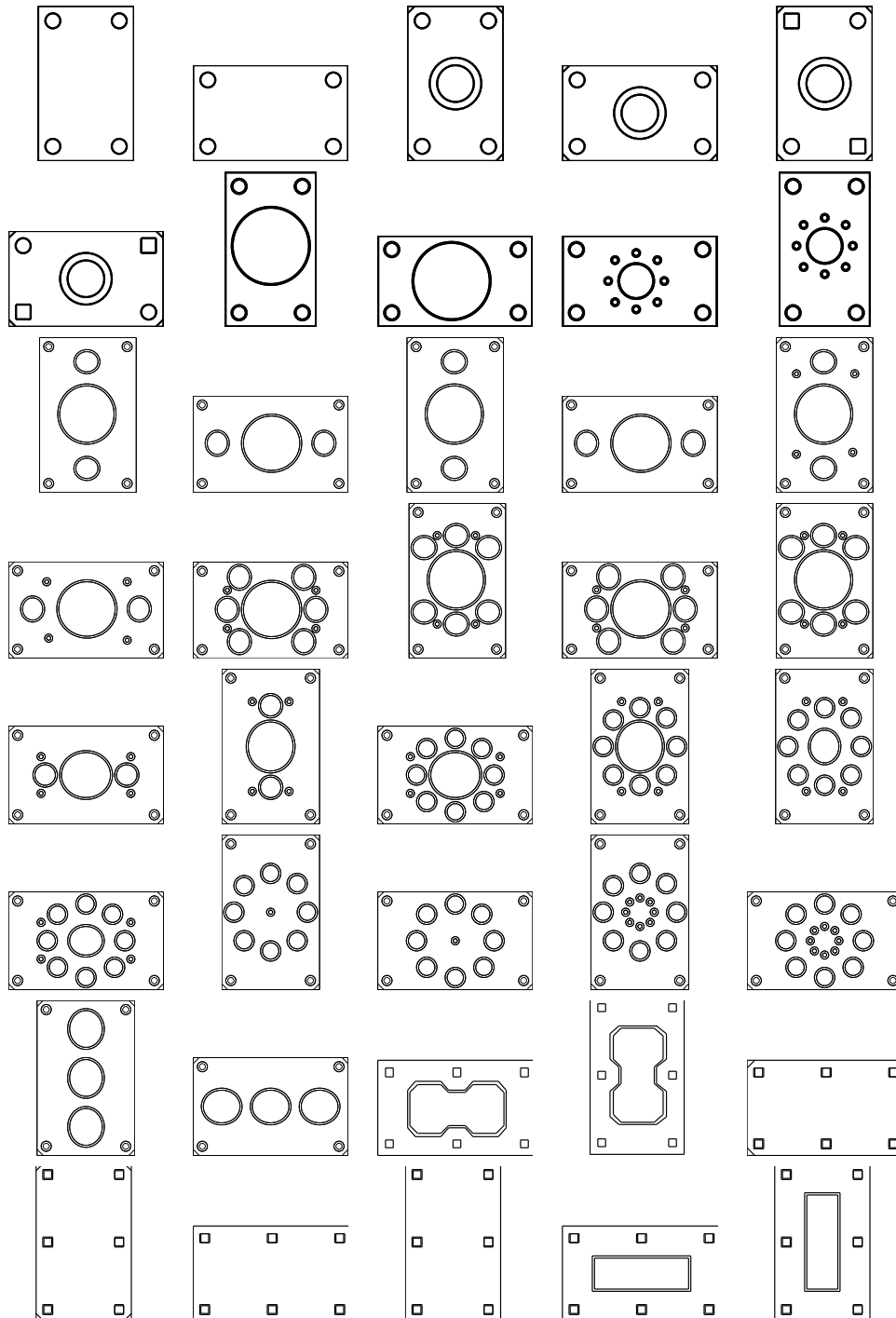
Databases

In this annex I present the databases used in usability evaluations. These databases consisted in vector drawings previously classified. The drawings were grouped according to its complexity. Thus, I used a set of thirty eight basic drawings, a set with forty simple technical drawings of mould plates and a set with twenty technical drawings of real parts. While the first two sets were used in the preliminary usability evaluation, only in final tests all the databases were used.

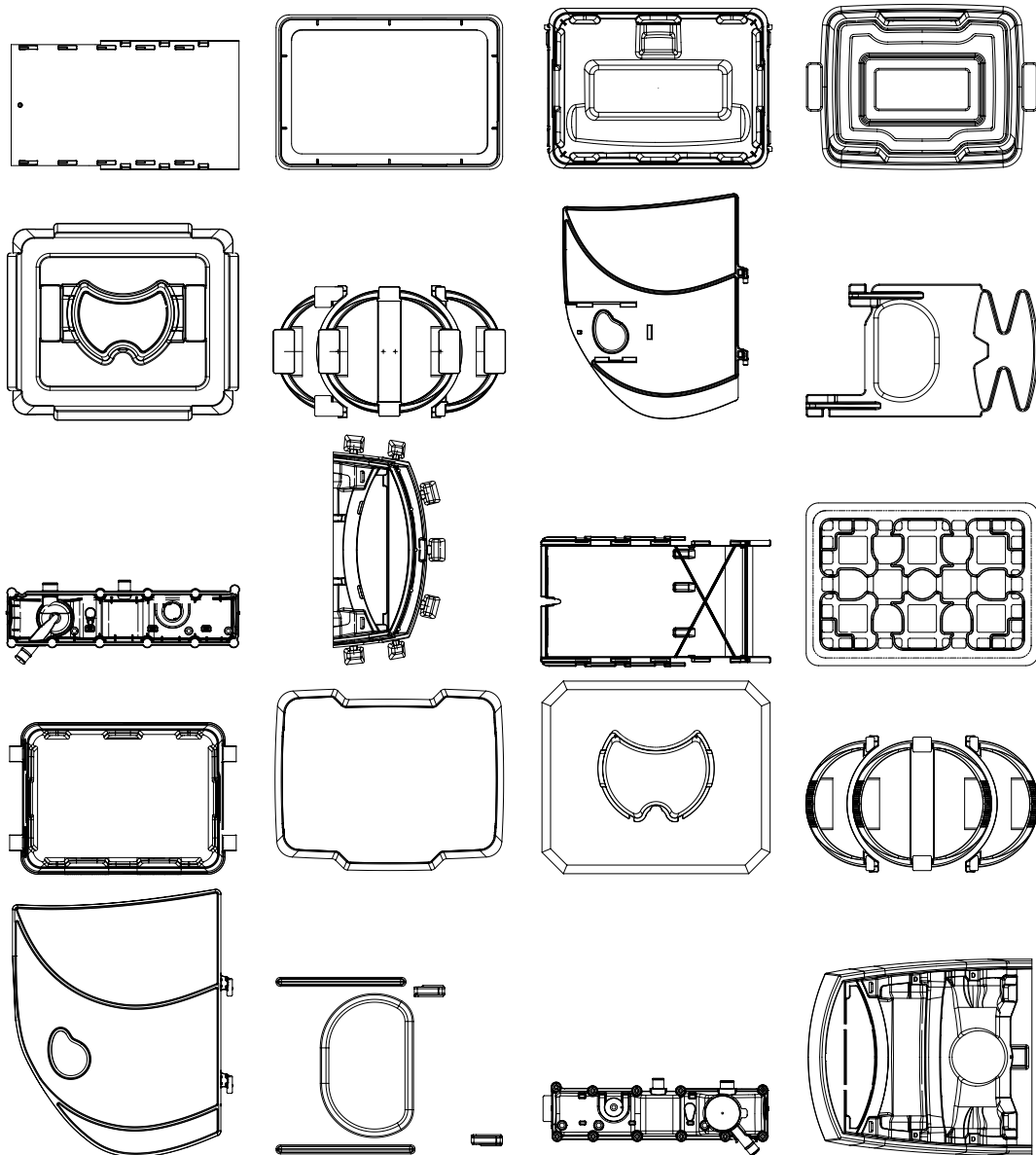
D.1 Basic Drawings



D.2 Simple Technical Drawings of Mould Plates



D.3 Technical Drawings of Parts



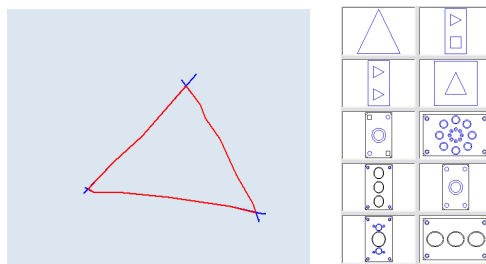
E

Sketched Queries

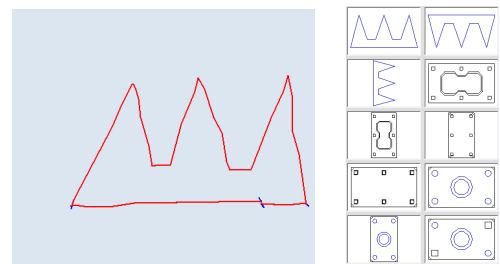
In this chapter I present the queries to the database sketched by users during usability evaluations. Additionally, for each query are also displayed the set of candidate drawings returned by the prototype.

E.1 Preliminary Usability Evaluation

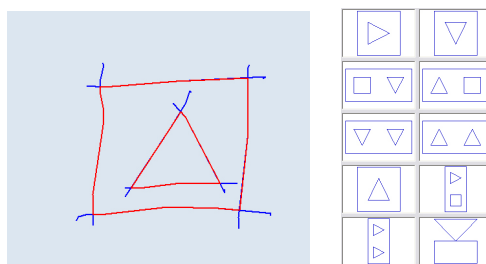
E.1.1 User A



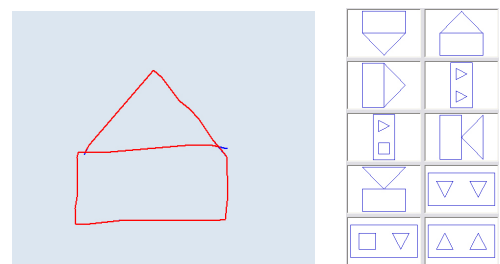
Query Q1, user A



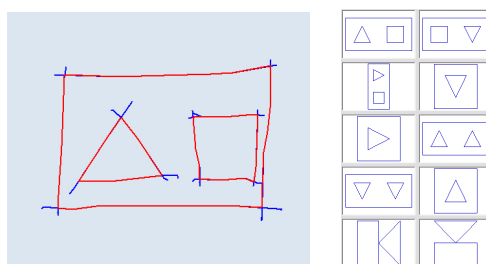
Query Q2, user A



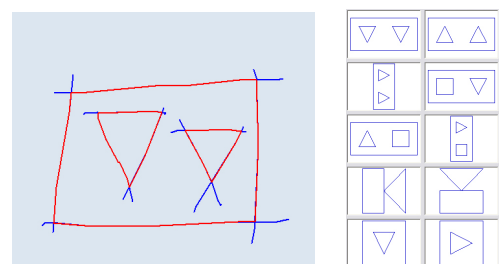
Query Q3, user A



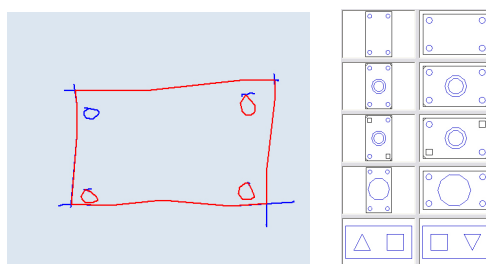
Query Q4, user A



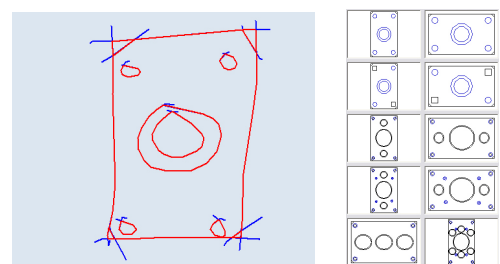
Query Q5, user A



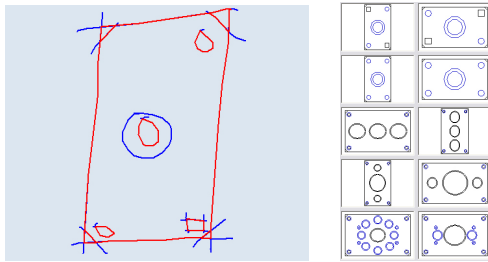
Query Q6, user A



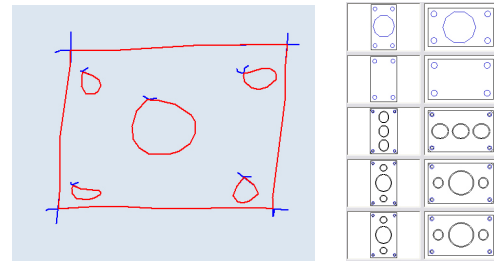
Query Q7, user A



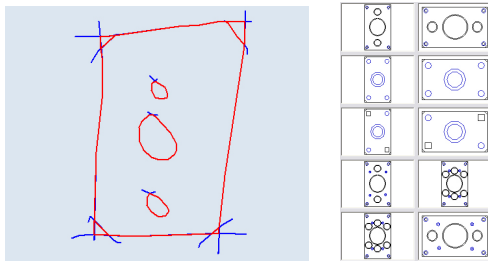
Query Q8, user A



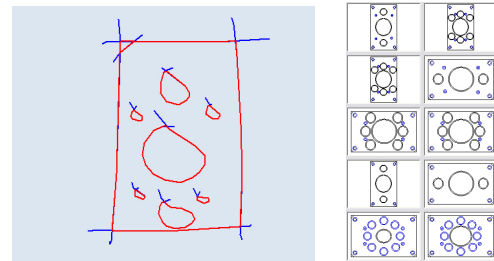
Query Q9, user A



Query Q10, user A

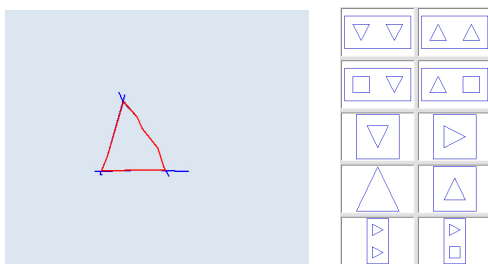


Query Q11, user A

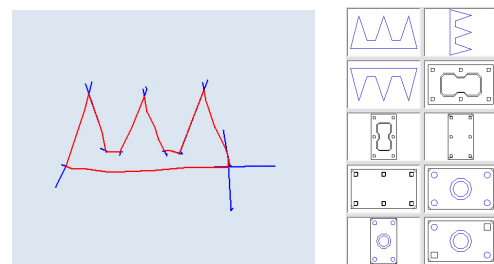


Query Q12, user A

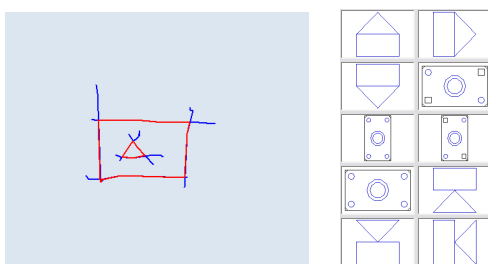
E.1.2 User B



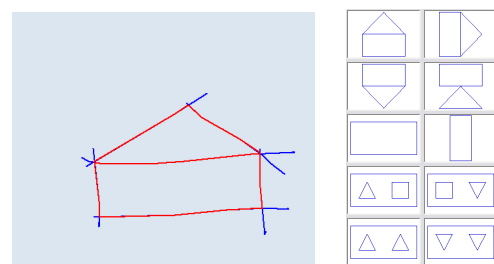
Query Q1, user B



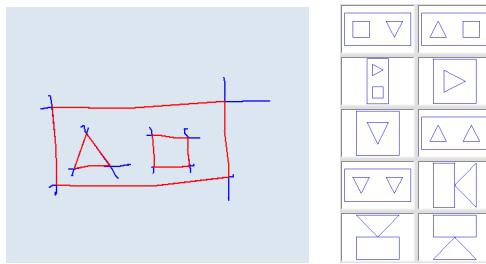
Query Q2, user B



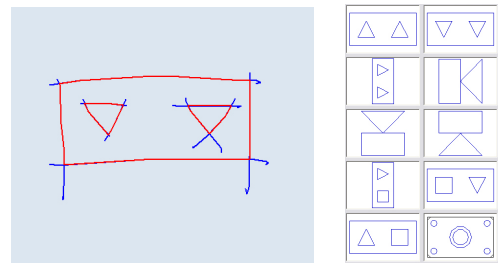
Query Q3, user B



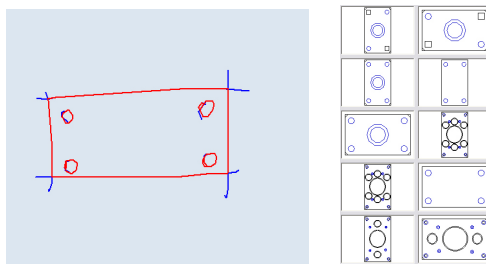
Query Q4, user B



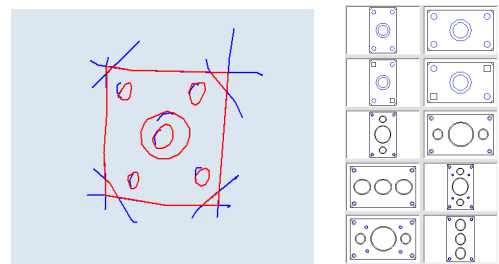
Query Q5, user B



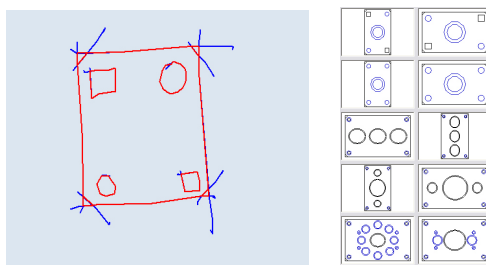
Query Q6, user B



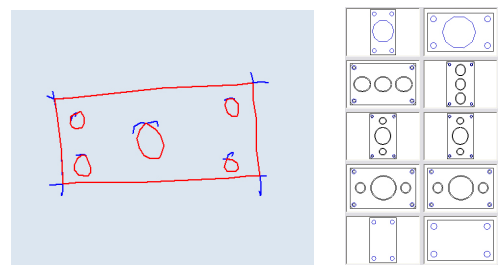
Query Q7, user B



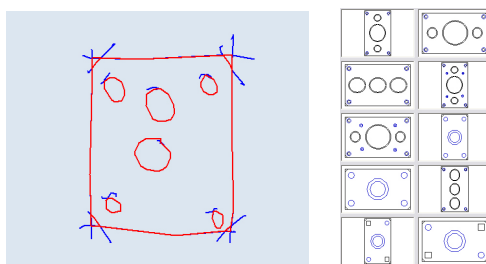
Query Q8, user B



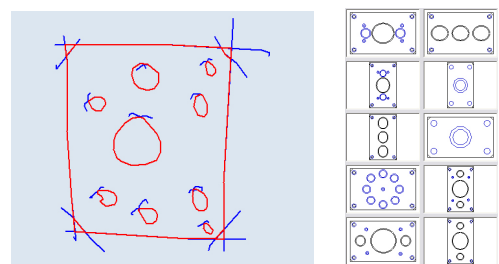
Query Q9, user B



Query Q10, user B

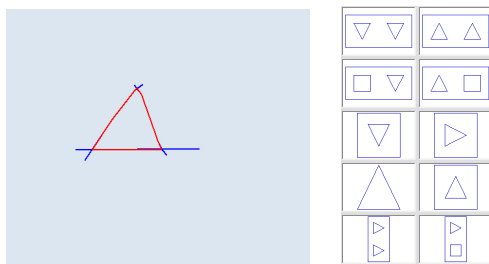


Query Q11, user B

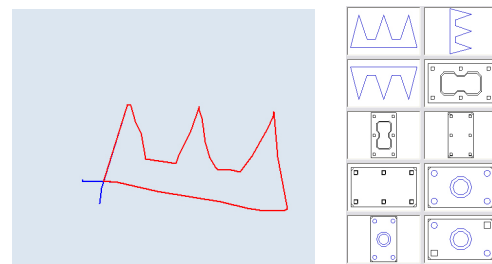


Query Q12, user B

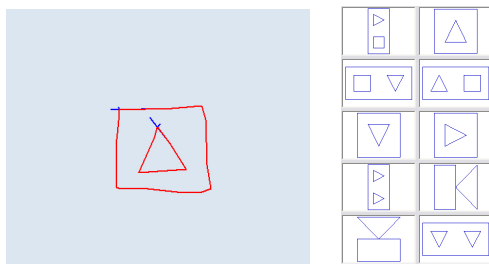
E.1.3 User C



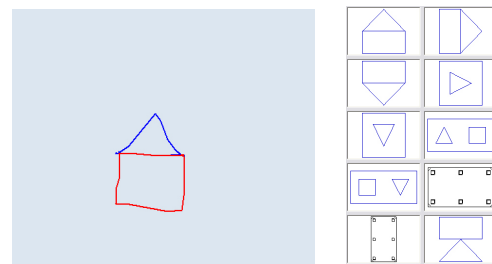
Query Q1, user C



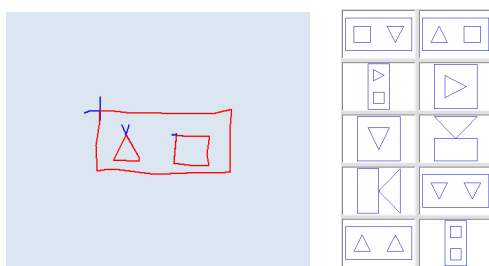
Query Q2, user C



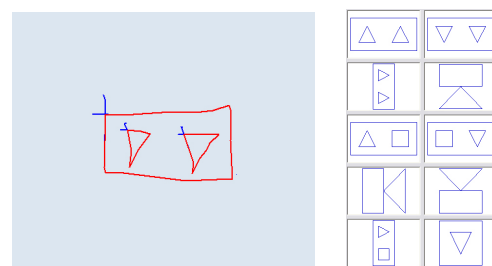
Query Q3, user C



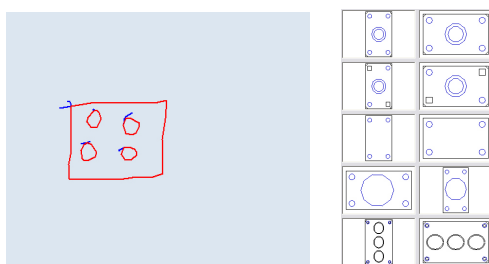
Query Q4, user C



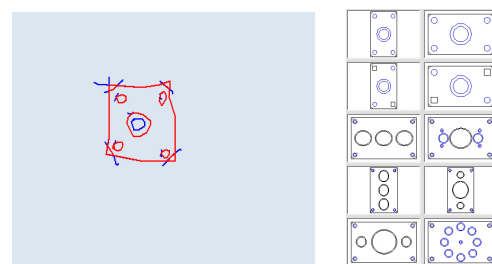
Query Q5, user C



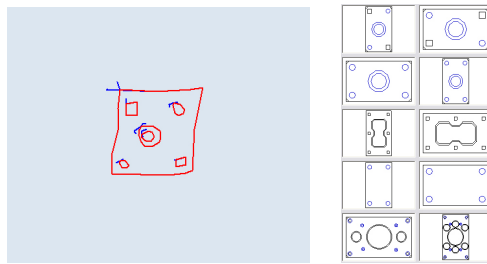
Query Q6, user C



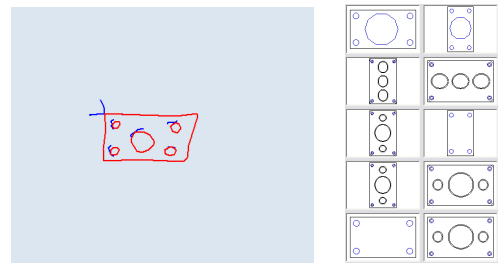
Query Q7, user C



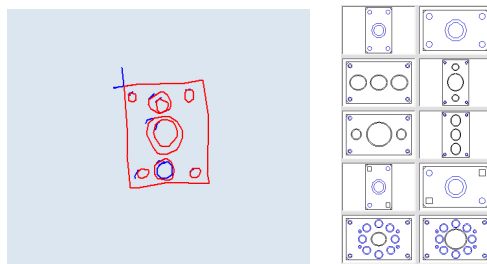
Query Q8, user C



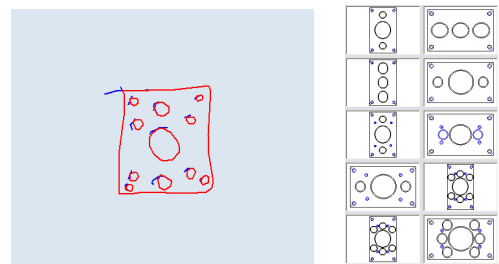
Query Q9, user C



Query Q10, user C



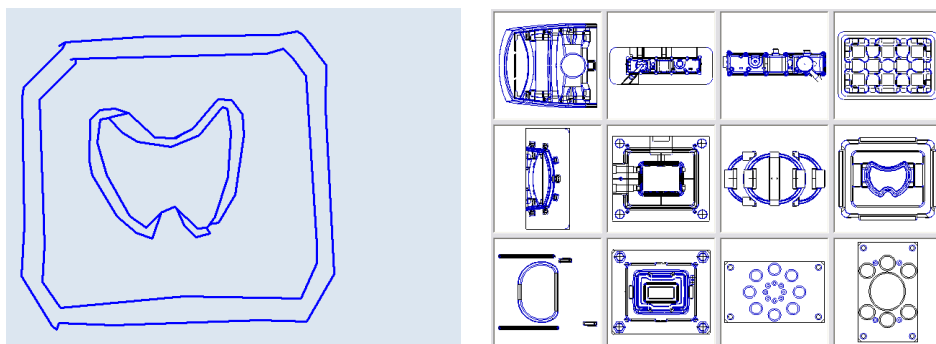
Query Q11, user C



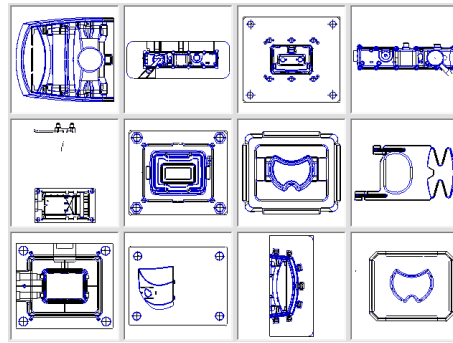
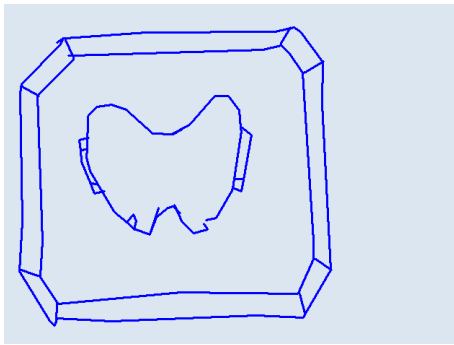
Query Q12, user C

E.2 Final Usability Evaluation

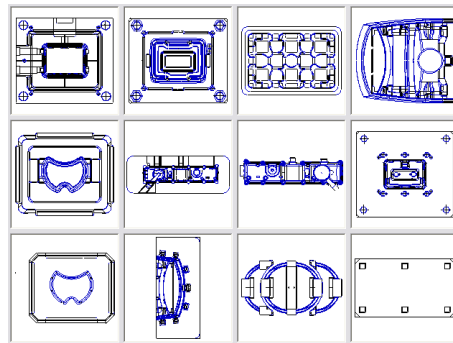
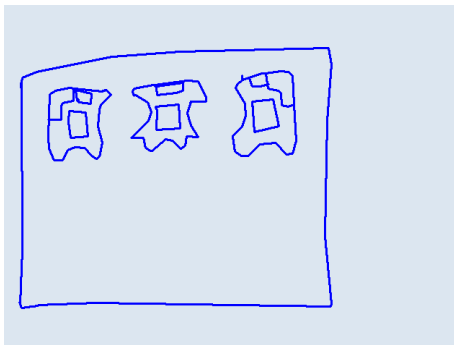
E.2.1 User A



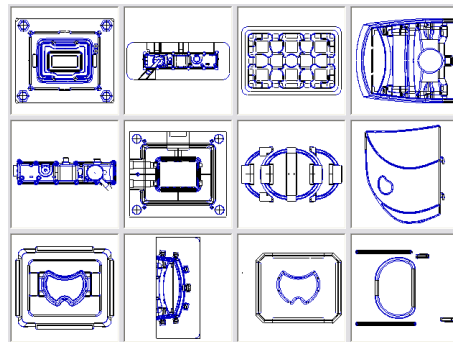
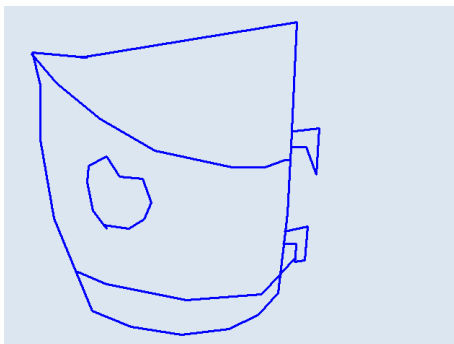
Query Q1, user A, first attempt



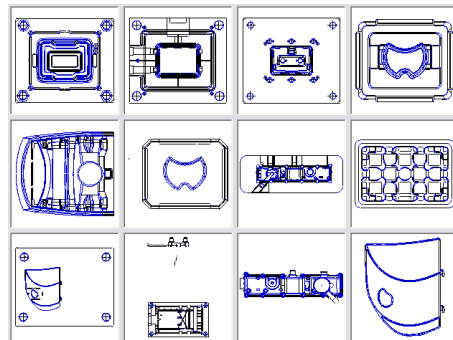
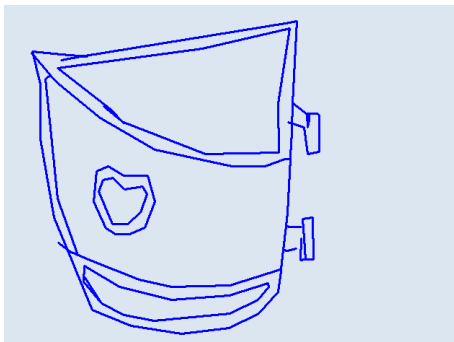
Query Q1, user A, second attempt



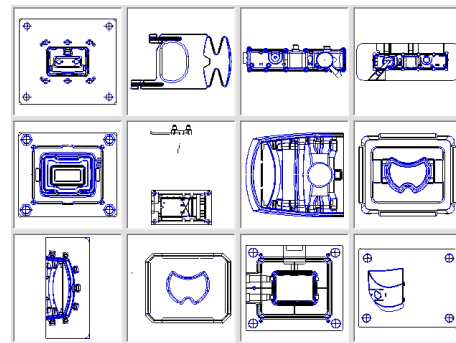
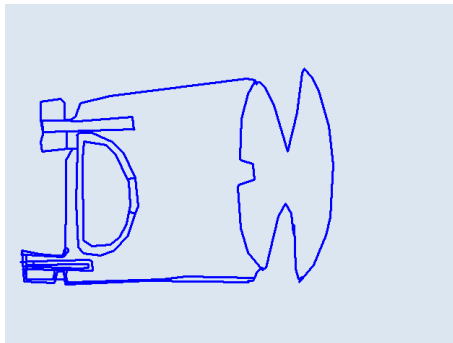
Query Q2, user A



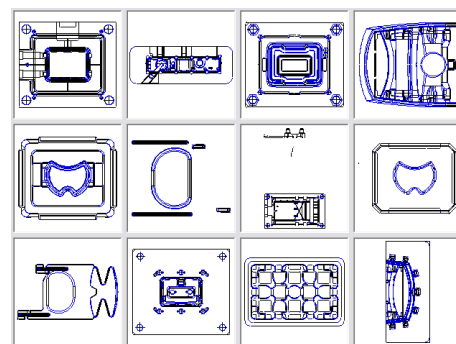
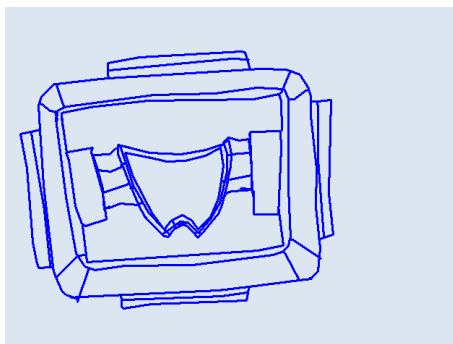
Query Q3, user A, first attempt



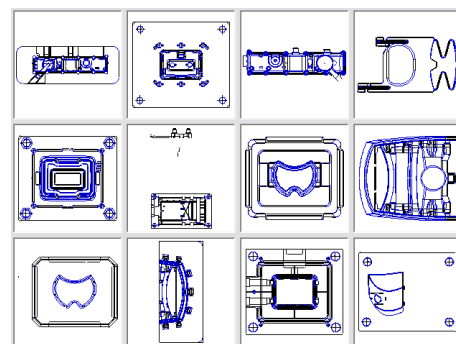
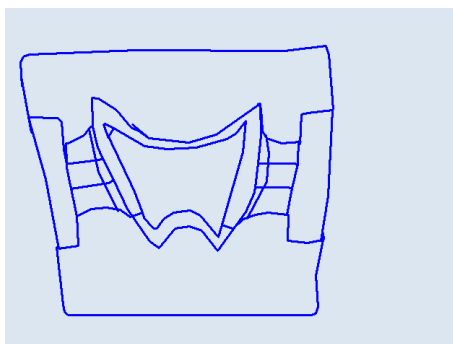
Query Q3, user A, second attempt



Query Q4, user A

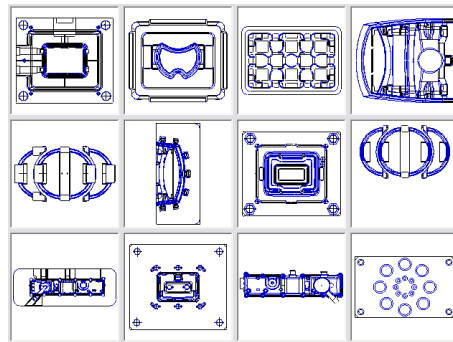
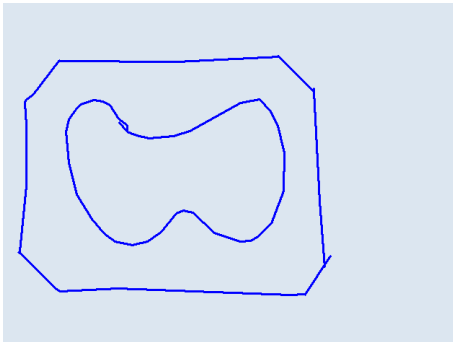


Query Q5, user A, first attempt

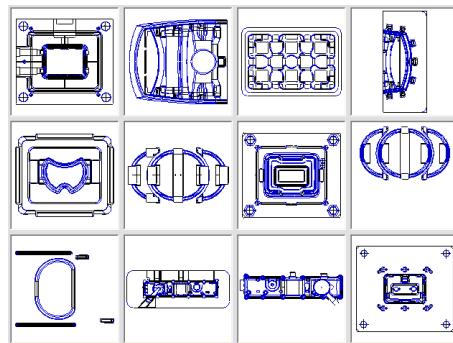
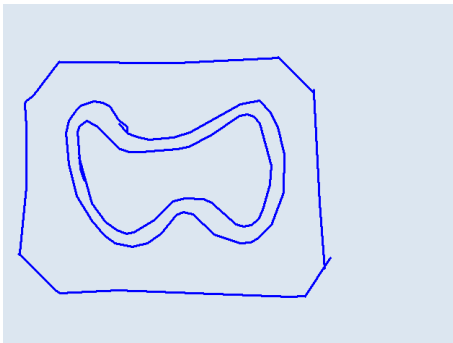


Query Q5, user A, second attempt

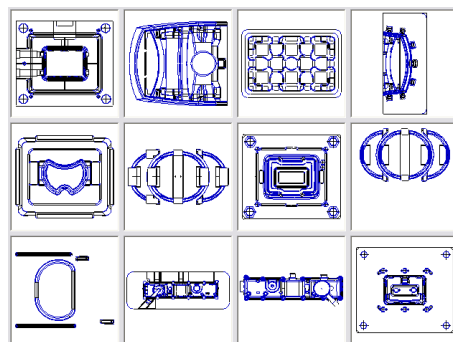
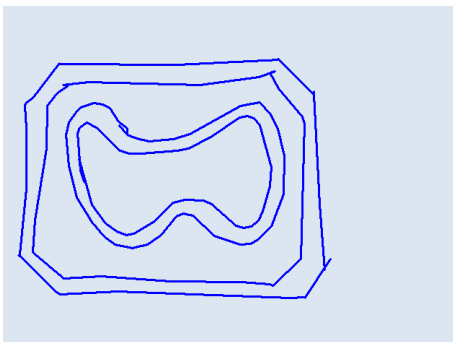
E.2.2 User B



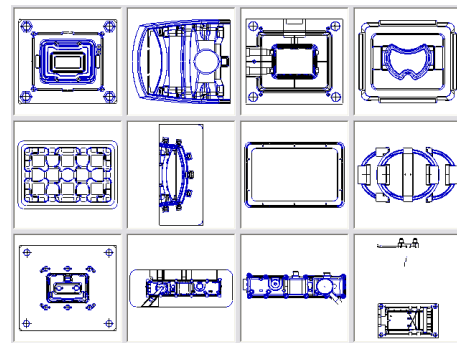
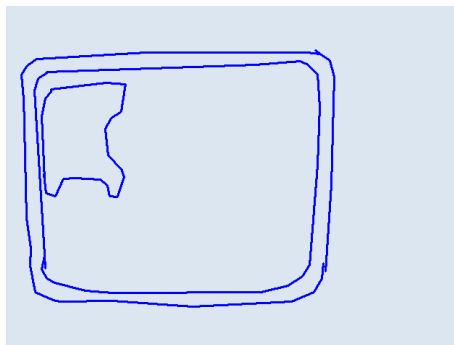
Query Q1, user B, first attempt



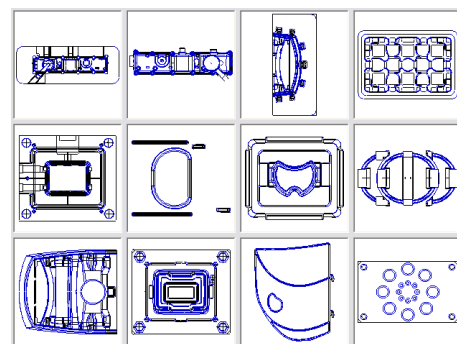
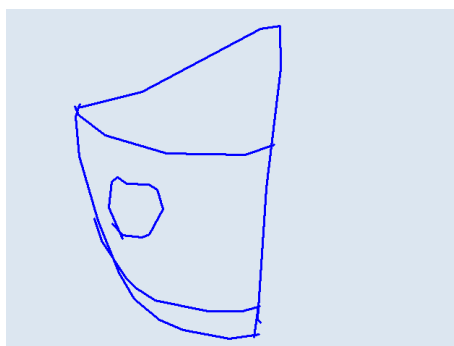
Query Q1, user B, second attempt



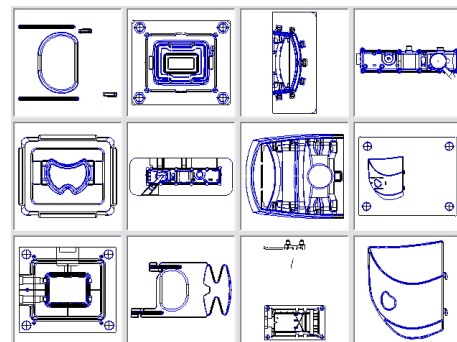
Query Q1, user B, third attempt



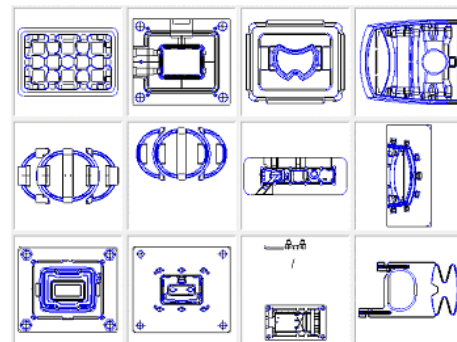
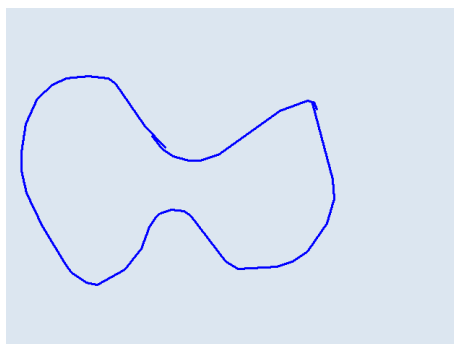
Query Q2, user B



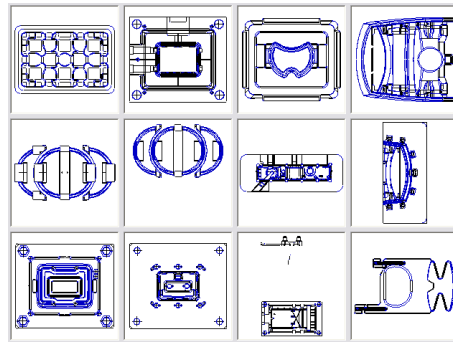
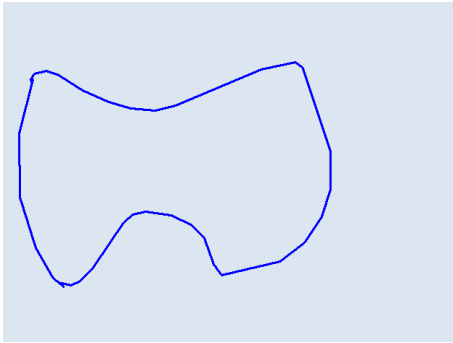
Query Q3, user B



Query Q4, user B

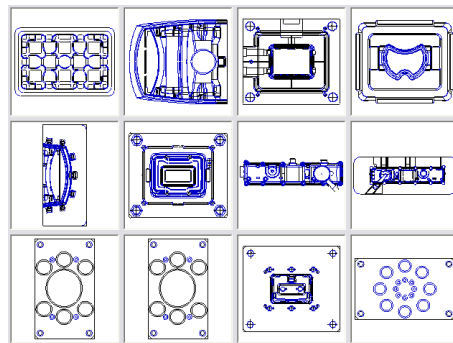
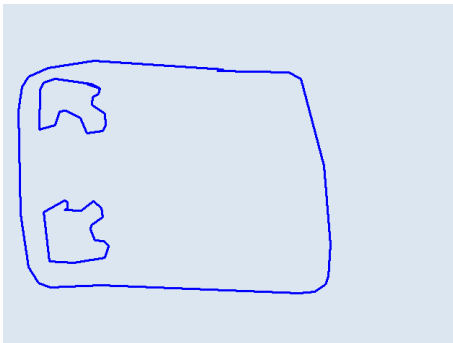


Query Q5, user B, first attempt

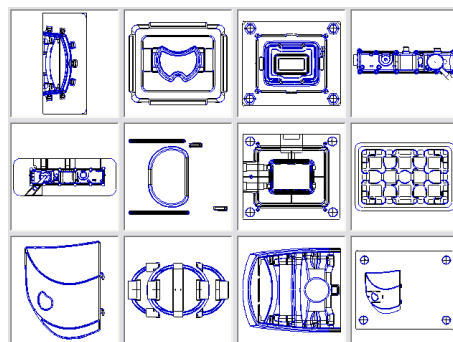
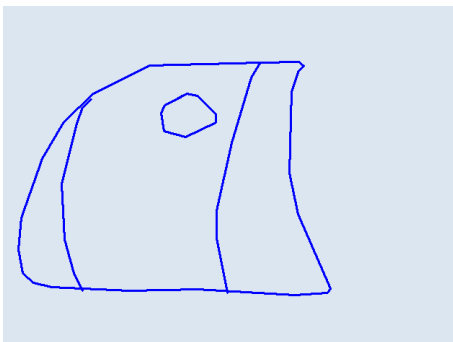


Query Q5, user B, second attempt

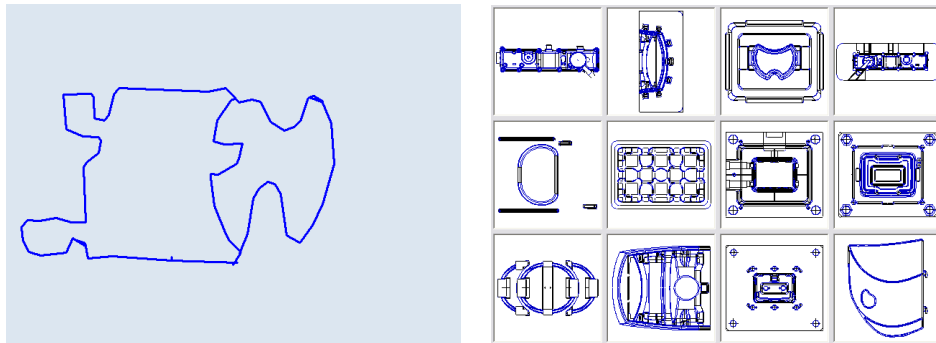
E.2.3 User C



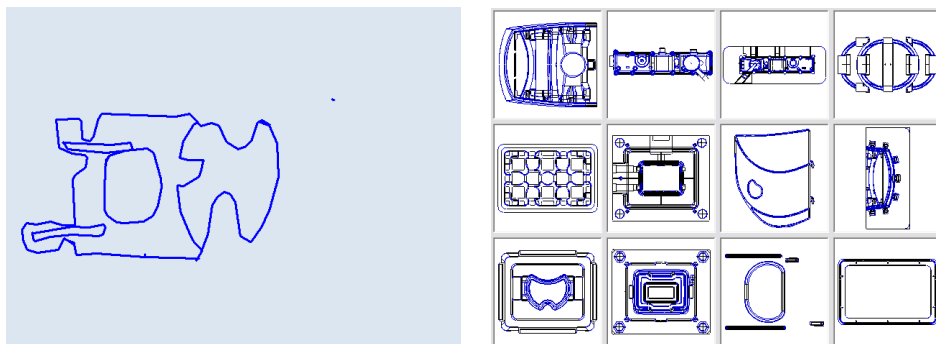
Query Q2, user C



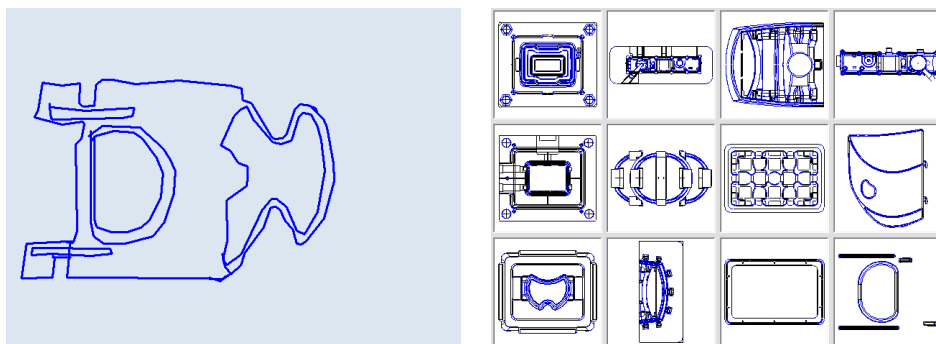
Query Q3, user C



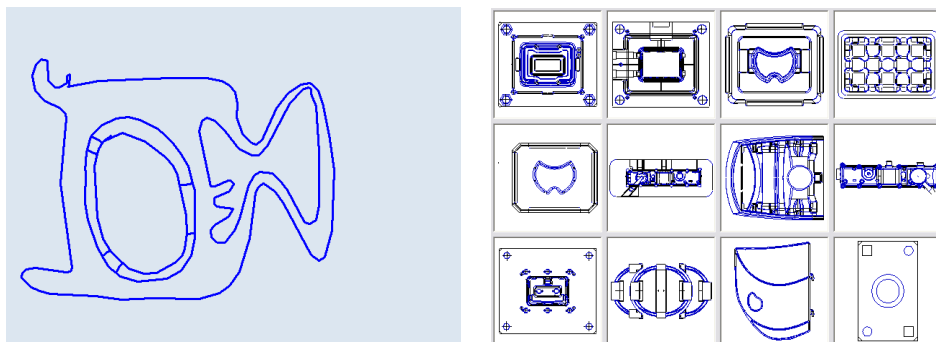
Query Q4, user C, first attempt



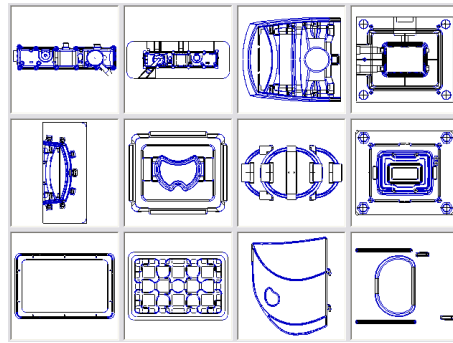
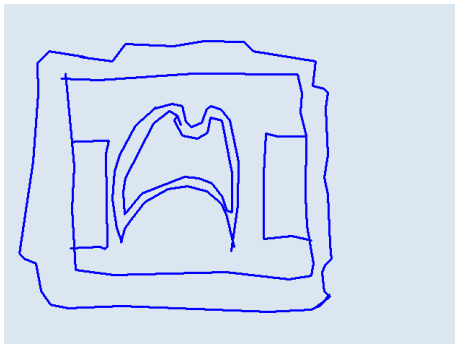
Query Q4, user C, second attempt



Query Q4, user C, third attempt

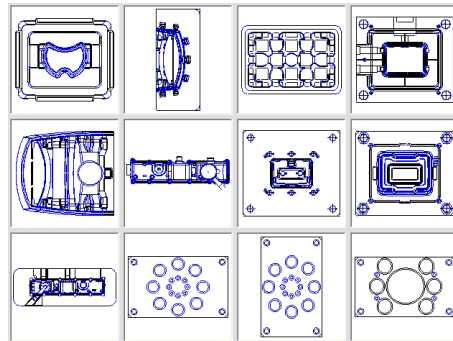
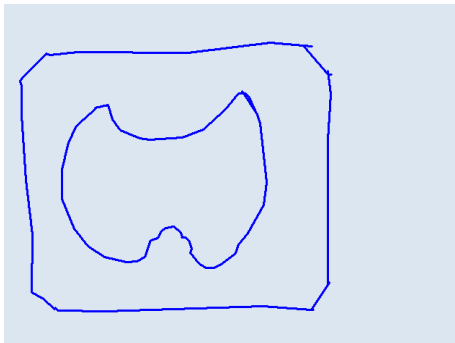


Query Q4, user C, fourth attempt

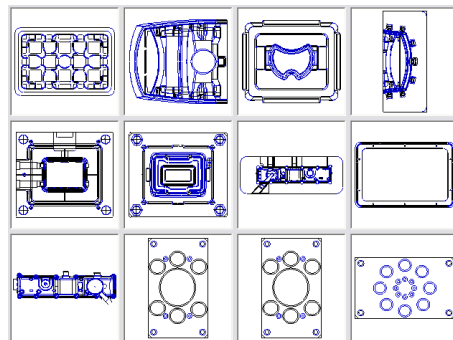
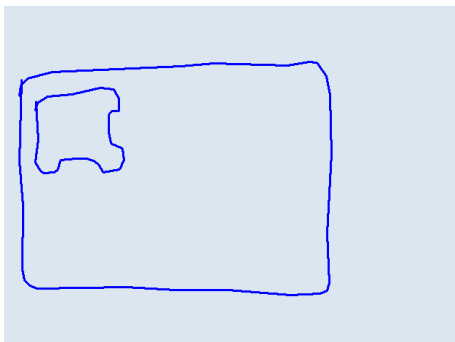


Query Q5, user C

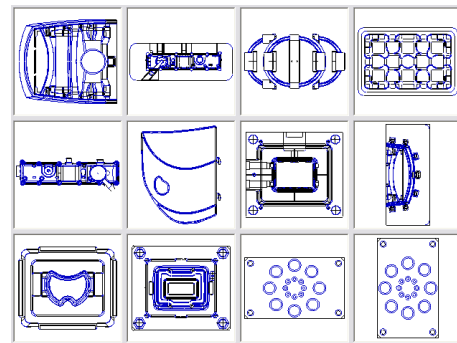
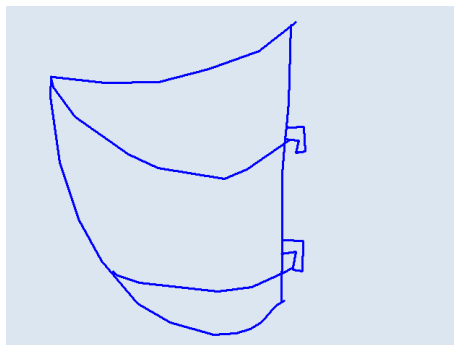
E.2.4 User D



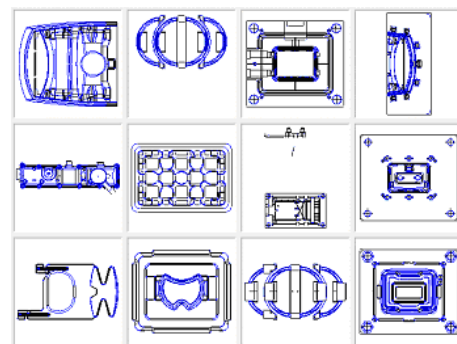
Query Q1, user D



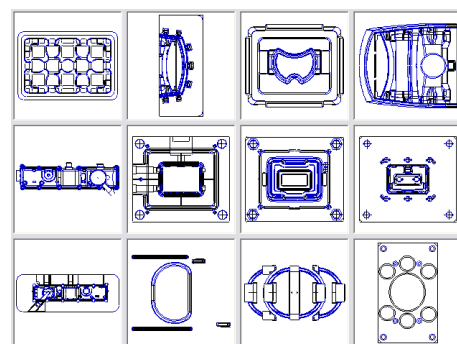
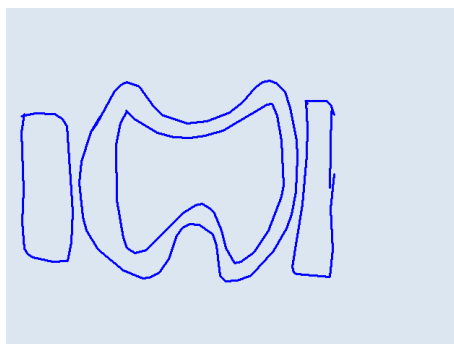
Query Q2, user D



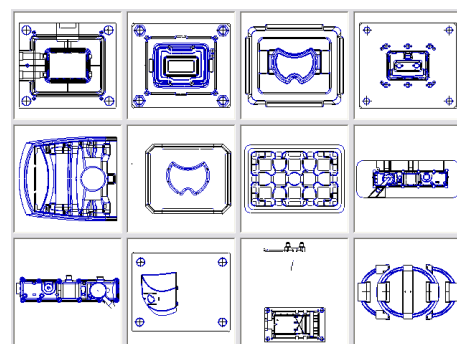
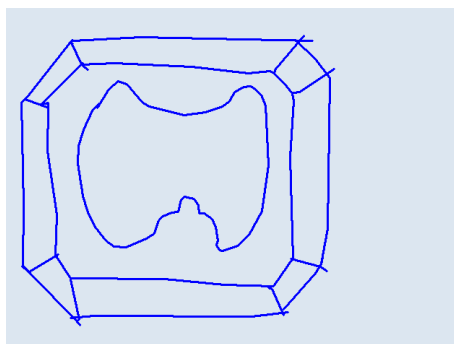
Query Q3, user D



Query Q4, user D

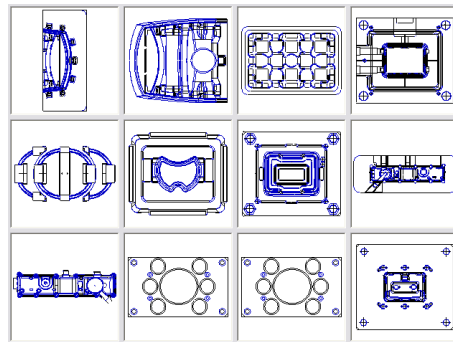
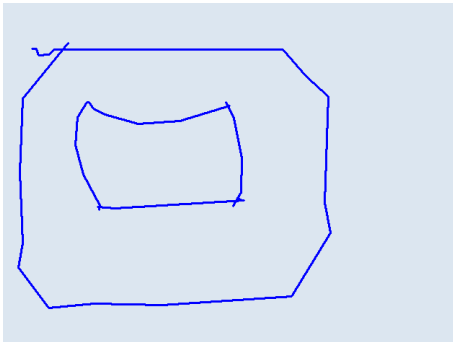


Query Q5, user D, first attempt

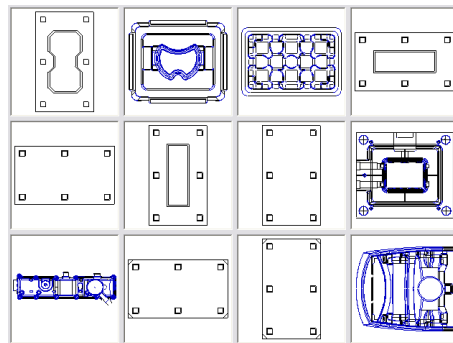
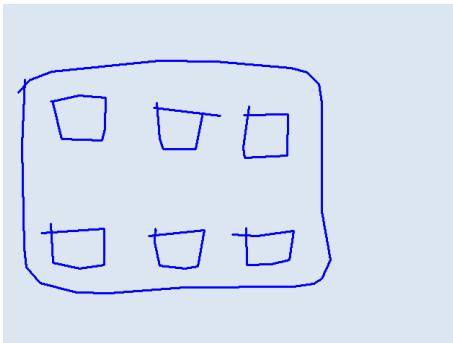


Query Q5, user D, second attempt

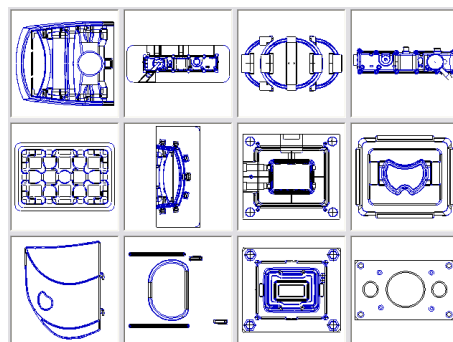
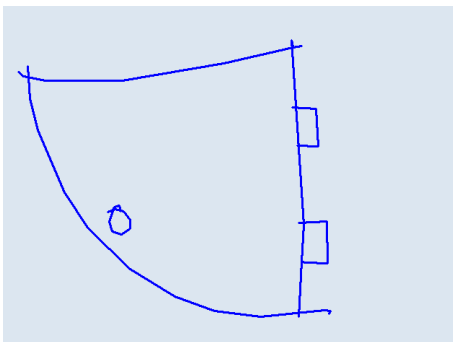
E.2.5 User E



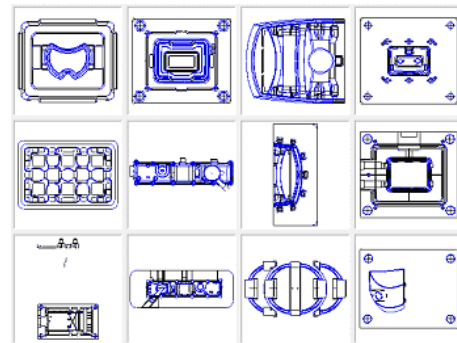
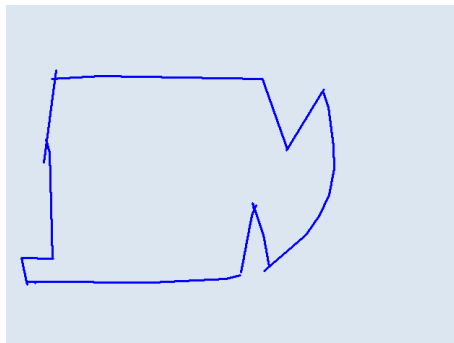
Query Q1, user E



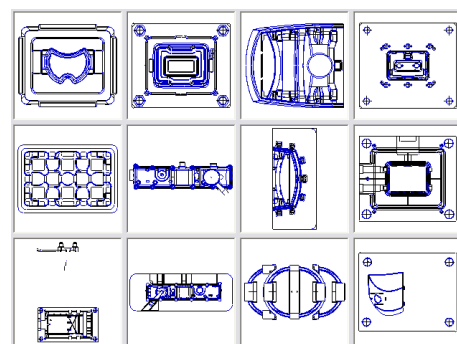
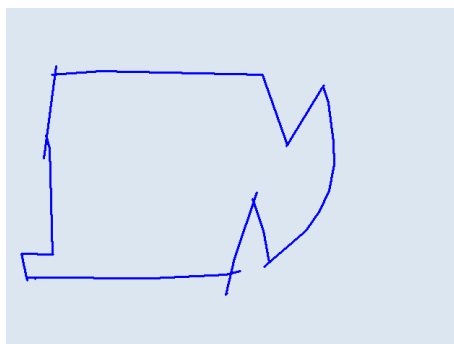
Query Q2, user E



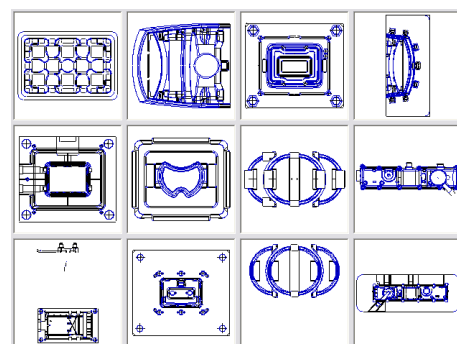
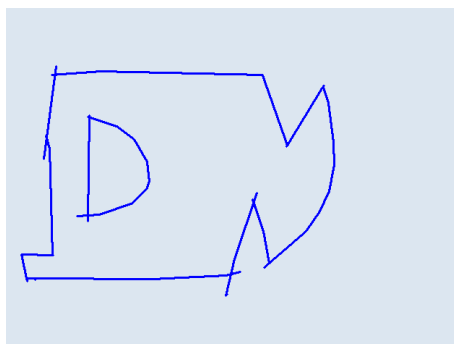
Query Q3, user E



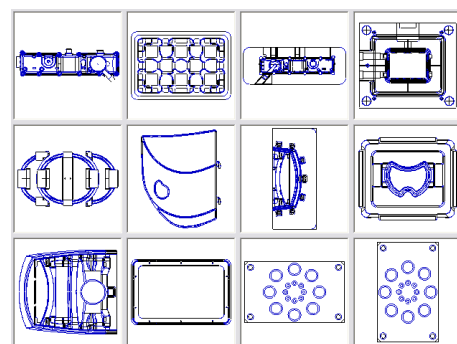
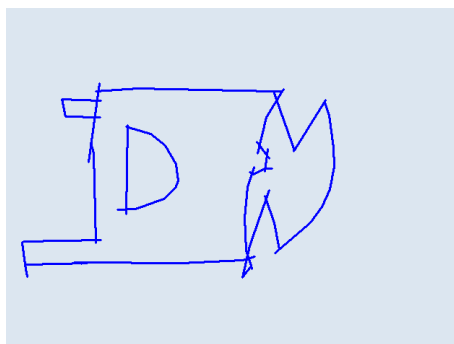
Query Q4, user E, first attempt



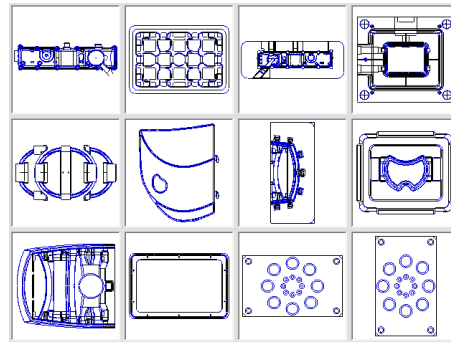
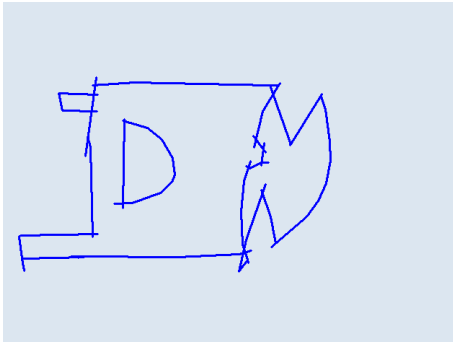
Query Q4, user E, second attempt



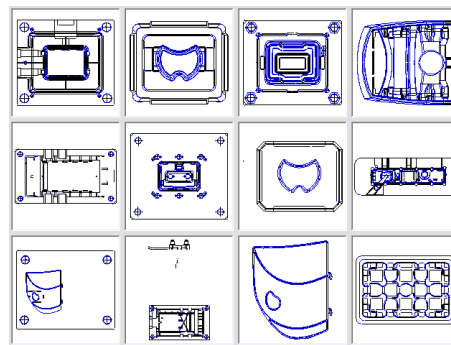
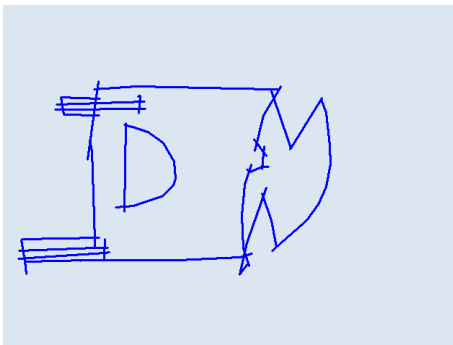
Query Q4, user E, third attempt



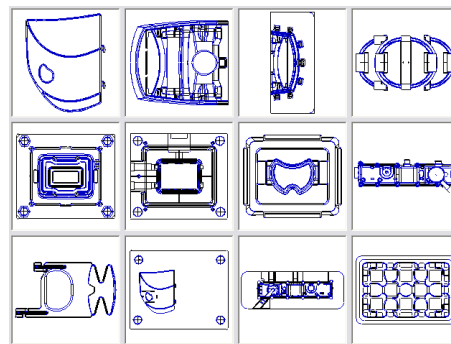
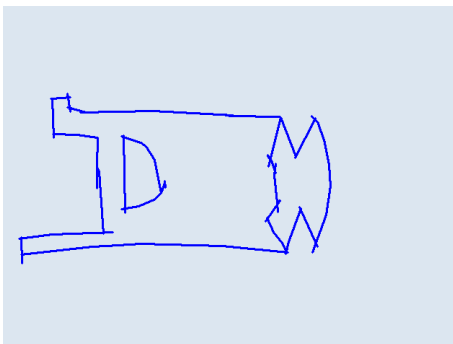
Query Q4, user E, third attempt



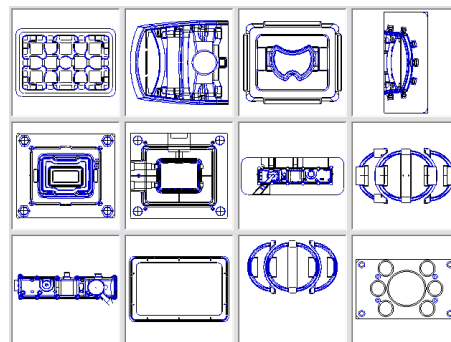
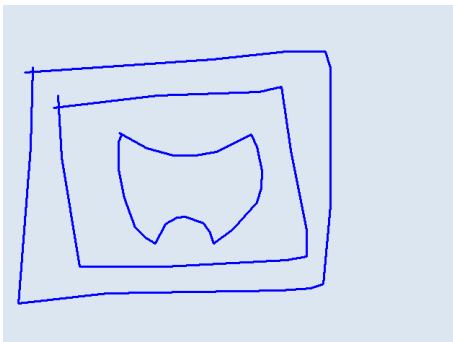
Query Q4, user E, fourth attempt



Query Q4, user E, fifth attempt

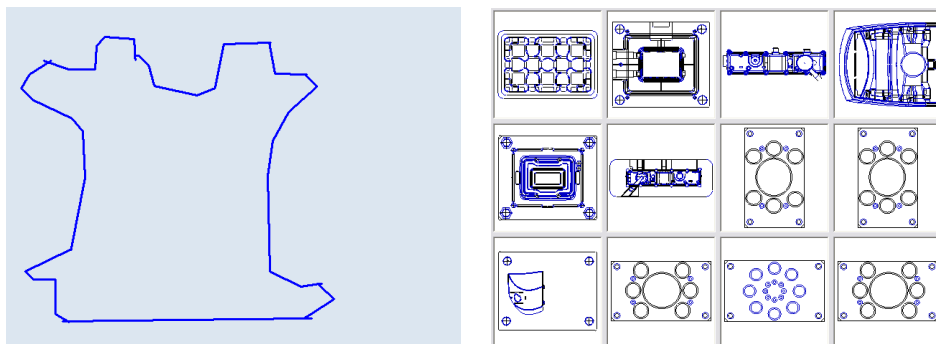


Query Q4, user E, sixth attempt

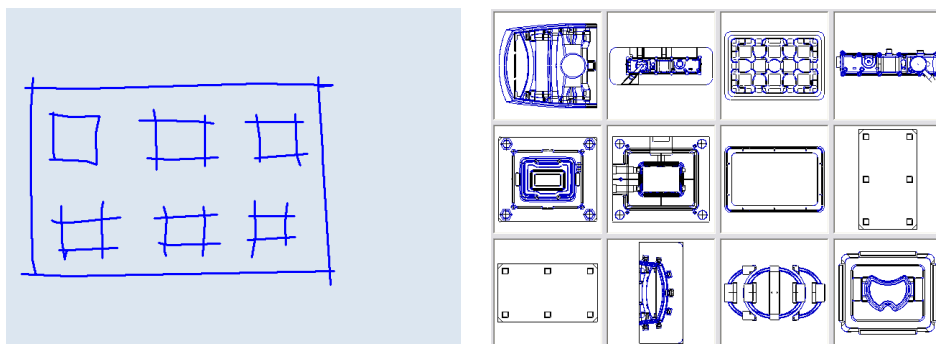


Query Q5, user E

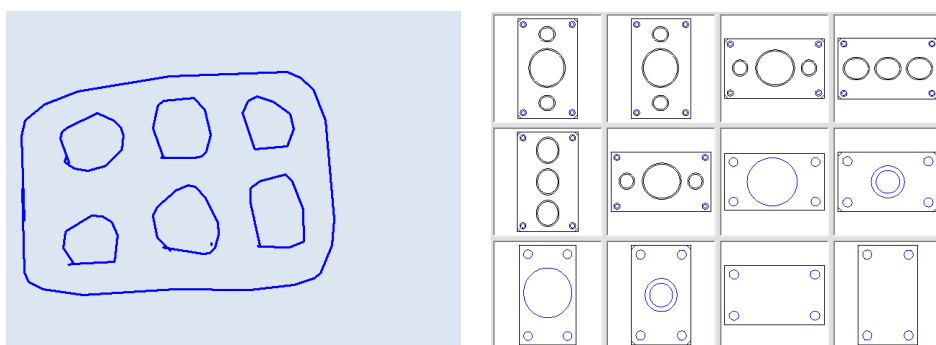
E.2.6 User F



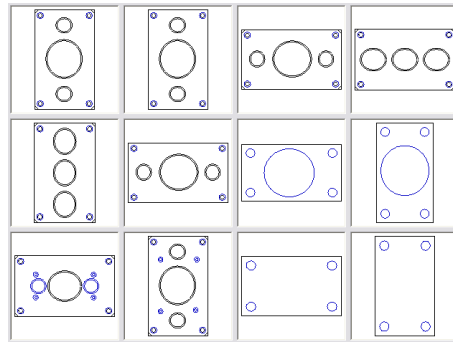
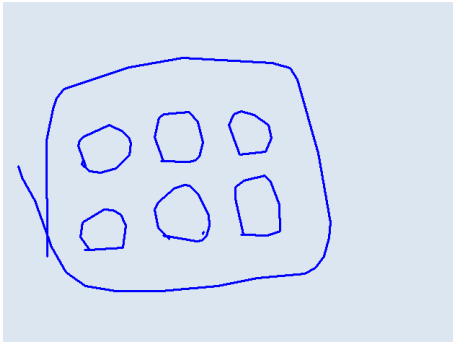
Query Q1, user F



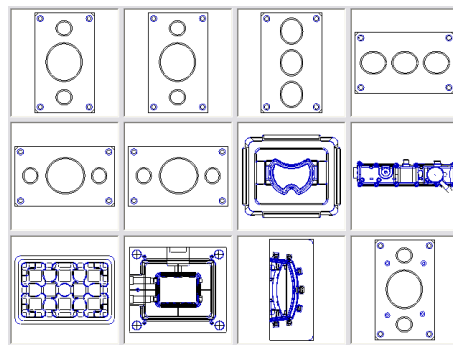
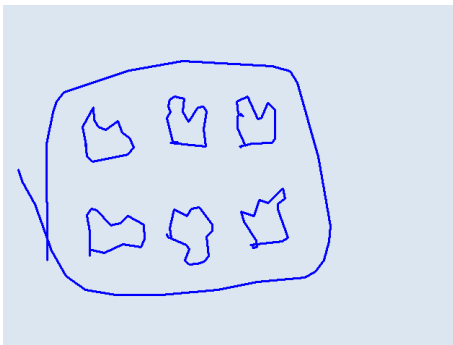
Query Q2, user F, first attempt



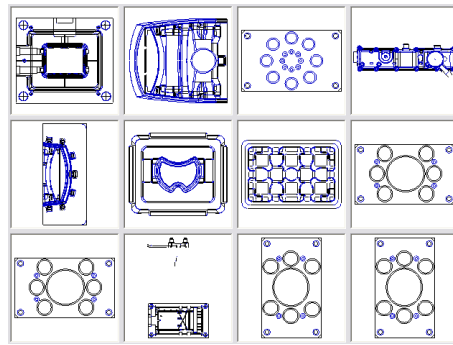
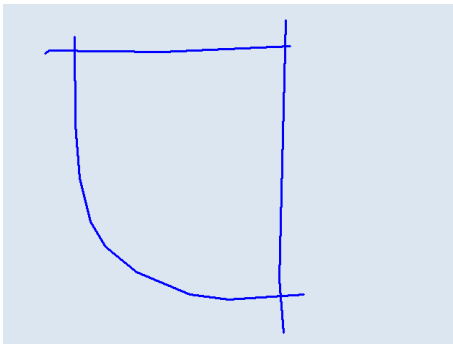
Query Q2, user F, second attempt



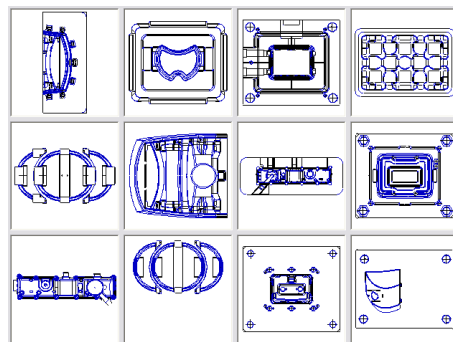
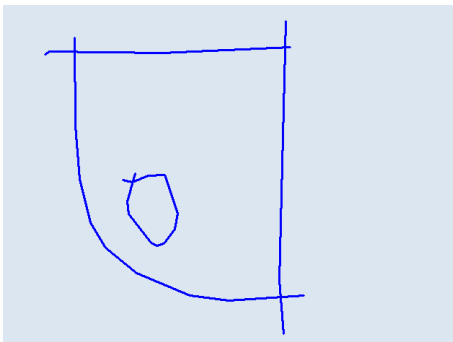
Query Q2, user F, third attempt



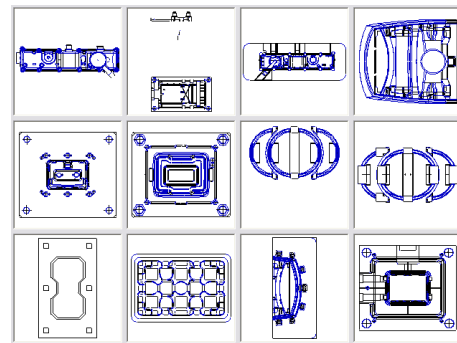
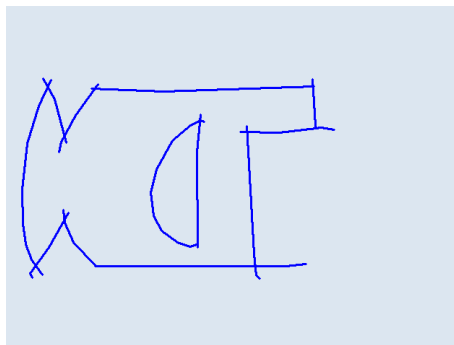
Query Q2, user F, fourth attempt



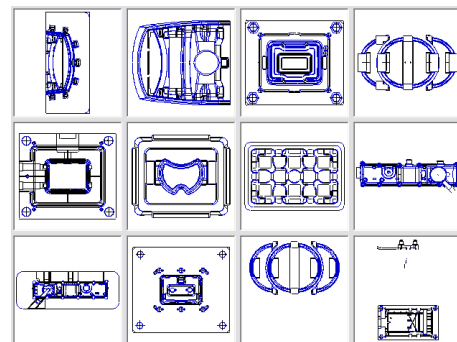
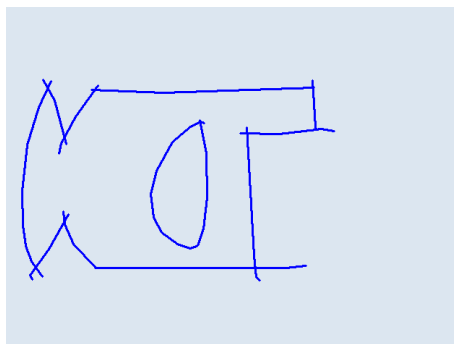
Query Q3, user F, first attempt



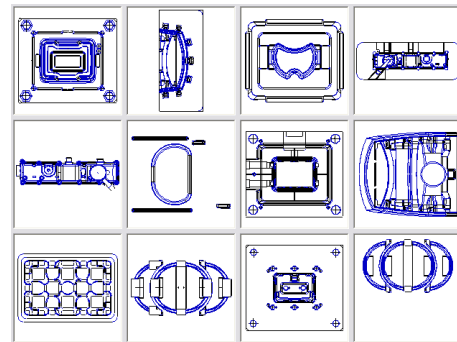
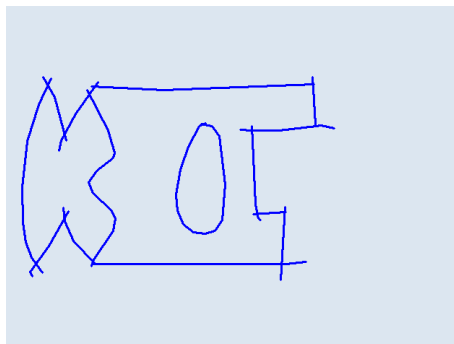
Query Q3, user F, second attempt



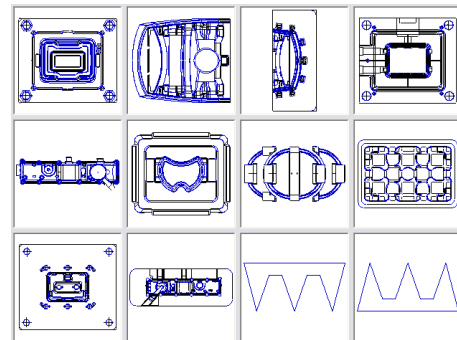
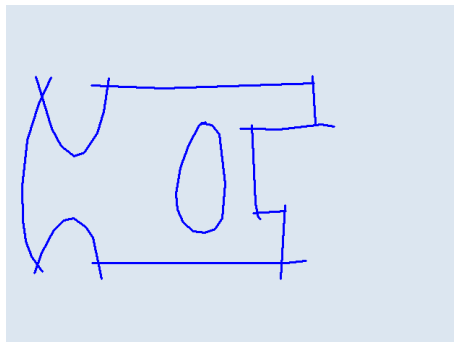
Query Q4, user F, first attempt



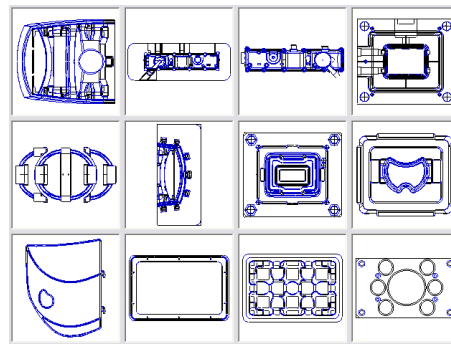
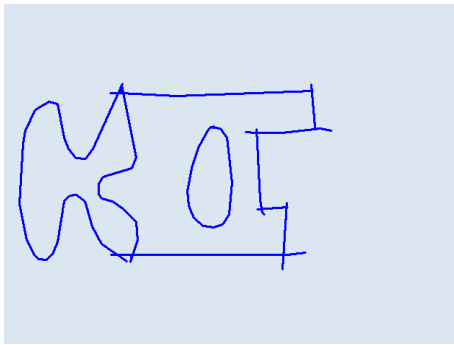
Query Q4, user F, second attempt



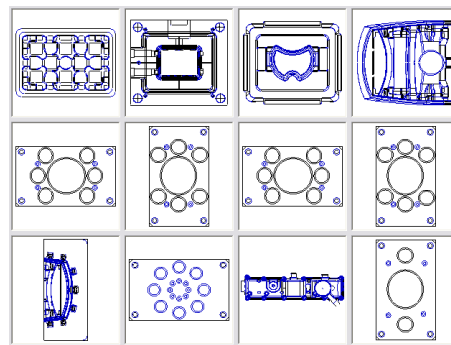
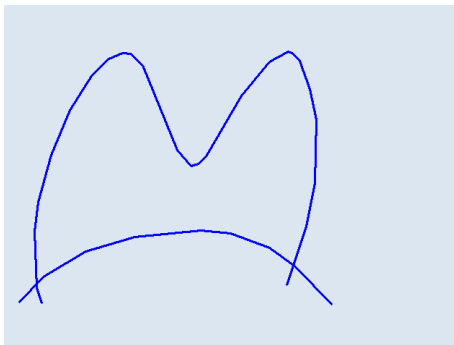
Query Q4, user F, third attempt



Query Q4, user F, fourth attempt



Query Q5, user F, fifth attempt



Query Q5, user F

Bibliography

- [1] A. Badel, J. P. Mornon, and S. Hazout. Searching for Geometric Molecular Shape Complementary Using Bidimensional Surface Profiles. *Journal of Molecular Biology*, 10:205–211, December 1992.
- [2] D. V. Bakergem. Image Collections in The Design Studio. In *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Age*, pages 261–272. MIT Press, 1990.
- [3] Ivan J. Balaban. An optimal algorithm for finding segment intersections. In *Proc. 11th Annual ACM Symposium Comp. Graph.*, pages 211–219. ACM, 1995.
- [4] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice Hall, 1982.
- [5] Ulrike Bartuschka, Kurt Mehlhorn, and Stefan Naher. A robust and efficient implementation of a sweep line algorithm for the straight line segment intersection. In *Proceedings of Workshop on Algorithm Engineering*, pages 124–135, Venice, Italy, September 1997.
- [6] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proceedings of the 15th Annual ACM Symp. Theory of Computing*, pages 80–86. ACM, 1983.
- [7] S. Berchtold, D.A. Keim, and H.P. Kriegel. Using extended feature object for partial similarity retrieval. *The International Journal on Very Large Data Bases*, 6(4):333–348, 1997.
- [8] Stefan Berchtold and Hans-Peter Kriegel. S3: Similarity in CAD Database Systems. In *Proc. of the Int. Conference on Management of Data (SIGMOD'97)*, 1997.
- [9] Luca Boatto, Vincenzo Consorti, Monica Del Buono, Francesco Melcarne, Marco Meucci, Andrea Morelli, Marco Mosciatti, Stefano Scarci, and Marco Tucci. An interpretation system for land register maps. *Computer*, 25(7):25–33, 1992.
- [10] James E. Boyce, David P. Dobkin, III Robert L.(Scot) Drysdale, and Leo J. Guibas. Finding extremal polygons. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 282–289, New York, NY, USA, 1982. ACM Press.

- [11] Augustin-Louis Cauchy. Recherche sur les polyèdres. *J. Ecole Polytechnique*, 9(16):68–86, 1813.
- [12] Ian Chai and Dov Dori. Orthogonal zig-zag: an efficient method for extracting straight lines from engineering drawings. In *Proceedings of the International Workshop on Visual Form (IWVF '91)*, Capri, Italy, May 1991.
- [13] S. K. Chang, B. Perry, and A. Rosenfeld. *Content-Based Multimedia Information Access*. Kluwer Press, 1999.
- [14] Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39:1–54, 1992.
- [15] Ken Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, ii. *Discrete and Computational Geometry*, 4:387–421, 1989.
- [16] M. Clayton and H. Wiesenthal. Enhancing the Sketchbook. In *Proc. of the Association for Computer Aided Design in Architecture (ACADIA '91)*, Los Angeles, CA, 1991.
- [17] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. MIT Press, McGraw-Hill, 2nd. edition, 1990.
- [18] Robert G. Cromley. Hierarchical methods of line simplification. *Cartography and Geographic Information Systems*, 18(2):125–131, 1991.
- [19] Robert G. Cromley. Principal axis line simplification. *Computers and Geosciences*, 18(8):1003–1011, 1992.
- [20] E.R. Davies and A.P.N. Plummer. Thinning algorithms: A critique and a new methodology. *Pattern Recognition*, 14(1–6):53–63, 1981.
- [21] Christian Demant, Bernd Streicher, and Peter Waszkewitz. *Industrial Image Processing : Visual Quality Control in Manufacturing*, chapter Chapter 8. "Overview:Image Acquisition and Illumination". Springer Verlag, 1999.
- [22] E. S. Deutsch. Thinning algorithms on rectangular, hexagonal, and triangular arrays. *Communications of the ACM*, 15(9):827–837, 1972.
- [23] G.P. Dinnen. Programming pattern recognition. In *Proc. Western Joint Computer Conf.*, pages 94–100, 1955.
- [24] Ellen Y. Do. What's in a Diagram that a Computer Should Understand? In *Proc. of The Sixth Int. Conf. on Computer Aided Architectural Design Futures (CAADF'95)*, pages 103–114. The Global Design Studio, 1995.
- [25] U. Dogrusöz and M. Krishnamoorthy. Cycle vector space algorithms for enumerating all cycles of a planar graph. Technical Report 5, Rensselaer Polytechnic Institute, Dept. of Computer Science, Troy, New York 12180 USA, January 1995.

- [26] Dov Dori. Orthogonal zig-zag: an algorithm for vectorizing engineering drawings compared with hough transform. *Adv. Eng. Softw.*, 28(1):11–24, 1997.
- [27] Dov Dori, Y. Liang, J. Dowell, and Ian Chai. Sparse-pixel recognition of primitives in engineering drawings. *Machine Vision and Applications*, (6):69–82, 1993.
- [28] Dov Dori and Wenyin Liu. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):202–215, 1999.
- [29] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10:112–122, 1973.
- [30] Leonhard Euler. *Elementa doctrinae solidorum*. *Novi Commentarii Academiae Scientiarum Petropolitanae*, 4:109–140, 1752.
- [31] Chin-Shyurng Fahn, Jhing-Fa Wang, and Jau-Yien Lee. A topology-based component extractor for understanding electronic circuit diagrams. *Comput. Vision Graph. Image Process.*, 44(2):119–138, 1988.
- [32] Alfredo Ferreira, Manuel J. Fonseca, and Joaquim A. Jorge. Polygon Detection from a Set of Lines. In *Proceedings of 12^o Encontro Português de Computação Gráfica (12th EPCG)*, pages 159–162, Porto, Portugal, October 2003.
- [33] Manuel Fonseca, Bruno Barroso, Pedro Ribeiro, and Joaquim Jorge. Sketch-Based Retrieval of ClipArt Drawings. In *Proceedings of the Advanced Visual Interfaces (AVI'04)*, Gallipoli, Italy, May 2004. ACM Press.
- [34] Manuel J. Fonseca. *Sketch-Based Retrieval in Large Sets of Drawings*. PhD thesis, Instituto Superior Técnico / Universidade Técnica de Lisboa, 07 2004.
- [35] Manuel J. Fonseca, Bruno Barroso, Pedro Ribeiro, and Joaquim A. Jorge. Retrieving ClipArt Images by Content. In *International Conference on Image and Video Retrieval (CIVR'04)*, Lecture Notes in Computer Science. Springer-Verlag, 07 2004.
- [36] Manuel J. Fonseca, Bruno Barroso, Pedro Ribeiro, and Joaquim A. Jorge. Retrieving Vector Graphics Using Sketches. In Andreas Butz, Antonio Krüger, and Patrick Olivier, editors, *Proceedings of the 4th International Symposium on Smart Graphics (SG'04)*, volume 3031 of *Lecture Notes in Computer Science*, pages 66–76. Springer-Verlag, May 2004.
- [37] Manuel J. Fonseca and Joaquim A. Jorge. *C_{AL}I*: Uma Biblioteca de Componentes para Interfaces Caligráficas. In *Actas do 9^o Encontro Português de Computação Gráfica*, Marinha Grande, Portugal, February 2000.
- [38] Manuel J. Fonseca and Joaquim A. Jorge. Experimental evaluation of an on-line scribble recognizer. *Pattern Recognition Letters*, 22(12):1311–1319, Jan 2001.

- [39] Manuel J. Fonseca, César Pimentel, and Joaquim A. Jorge. CALI: An Online Scribble Recognizer for Calligraphic Interfaces. In *Proceedings of the 2002 AAAI Spring Symposium - Sketch Understanding*, pages 51–58, Palo Alto, USA, March 2002.
- [40] Herbert Freeman. Computer processing of line-drawing images. *ACM Computing Surveys*, 6(1):57–97, 1974.
- [41] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Transactions on Graphics*, 22(1), January 2003.
- [42] Michael T. Goodrich, Leonidas J. Guibas, John Hershberger, and Paul J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 284–293. ACM Press, 1997.
- [43] Ovidiu Grigore and Remco C. Veltkamp. On the implementation of polygonal approximation algorithms. Technical Report UU-CS-2003-005, Institute of Information and Computing Sciences, Utrecht University, 2003.
- [44] Mark Gross and Ellen Do. Demonstrating the Electronic Cocktail Napkin: a paper-like interface for early design. In *Proc. of the Conf. on Human Factors in Computing Systems (CHI'96)*, pages 5–6, 1996.
- [45] Mark D. Gross. Indexing Visual Databases of Designs with Diagrams. In A. Koutamanis, H. Timmermans, and I. Vermeulen, editors, *Visual Databases in Architecture*, pages 1–14, Avebury: Aldershot, UK, 1995.
- [46] Dan Halperin and Eli Packer. Iterated snap rounding. *Comput. Geom. Theory Appl.*, 23(2):209–225, 2002.
- [47] R.M. Haralick. Performance characterization in image analysis: Thinning, a case in point. *Pattern Recognition Letters*, 13:5–12, 1992.
- [48] R.M. Haralick and L.G. Shapiro. Addison-Wesley, 1992.
- [49] David Hartvigsen and Russel Mardon. The all-pairs minimum cut problem and the minimum cycle basis problem on planar graphs. *SIAM Journal on Computing*, 7(3):403–418, August 1994.
- [50] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification. Technical report, Carnegie Mellon University, 1997.
- [51] John Hershberger and Jack Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *Proceedings of the 5th International Symposium on Spatial Data Handling*, volume 1, pages 134–143, Charleston, South Carolina, USA, 1992.

- [52] John Hershberger and Jack Snoeyink. An $o(n \log n)$ implementation of the douglas-peucker algorithm for line simplification. In *Proceedings of the 10th Annual Symposium on Computational geometry*, pages 383–384. ACM Press, 1994.
- [53] C.J. Hilditch. Linear skeletons from square cupboards. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, number 4, pages 404–420. Edinburgh University Press, 1969.
- [54] C.J. Hilditch. Comparison of thinning algorithms on a parallel processor. *Image Vision Comput.*, 1(3):115–132, August 1983.
- [55] John Hobby. Practical segment intersection with finite precision output. *Computational Geometry: Theory and Applications*, 13(4), 1999.
- [56] John D. Hobby. Polygonal approximations that minimize the number of inflections. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 93–102. Society for Industrial and Applied Mathematics, 1993.
- [57] G. Hu and Z.N. Li. An x-crossing preserving skeletonization algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:1031–1053, 1993.
- [58] Robert A. Hummel. Histogram modification techniques. *Computer Graphics Image Processing*, 4(3):209–224, September 1975.
- [59] Hiroshi Imai and Masao Iri. Polygonal approximations of a curve – formulations and algorithms. In G. T. Toussaint, editor, *Computational Morphology*, pages 71–86. Elsevier Science, 1988.
- [60] Jasmee Jaafar. Line generalization: least square with double tolerance. In C.A.Brebbia and P.Pascolo, editors, *Management Information Systems 2002: GIS and Remote Sensing*, volume 4 of *Management Information Systems*. WIT Press, 2002.
- [61] Rik D. T. Janssen and Albert M. Vossepoel. Adaptive vectorization of line drawing images. *Comput. Vis. Image Underst.*, 65(1):38–56, 1997.
- [62] J.D.Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, April 1987.
- [63] J.Jimenez and J.L.Navalon. Some experiments in image vectorization. *IBM Journal of Research and Development*, 26(6):724–734, 1982.
- [64] J.L.Bentley and T.Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, pages 643–647, 1979.
- [65] Christopher B. Jones and Ian M. Abraham. Line generalisation in a global cartographic database. *Cartographica*, 24(3):32–45, 1987.

- [66] R. Katsuri, S. T. Bow, W. El-Masri, J. Shah, J. R. Gattiker, and U. B. Mokate. A system for interpretation of line drawings. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10):978–992, 1990.
- [67] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.
- [68] Louisa Lam and Ching Y. Suen. An evaluation of parallel thinning algorithms for character recognition. *IEEE TPAMI*, 17:914–919, 1995.
- [69] Wing Ho Leung and Tsuhan Chen. Retrieval of Sketches Based on Spatial Relation Between Strokes. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'02)*, volume 1, pages 908–911, Rochester, New York, USA, September 2002. IEEE Press.
- [70] Wing Ho Leung and Tsuhan Chen. User-Independent Retrieval of Free-Form Hand-Drawn Sketches. In *Proc. of the IEEE Int. Conf. on Acoustics Speech and Signal Processing (ICASSP'02)*, volume 2, pages 2029–2032, Orlando, Florida, USA, May 2002. IEEE Press.
- [71] X. Lin, S. Shimotsuji, M. Minoh, and T. Sakai. Efficient diagram understanding with characteristic pattern detection. *Computer Vision, Graphics and Image Processing*, 30:84–106, April 1985.
- [72] Prabhaker Mateti and Narsingh Deo. On algorithms for enumerating all circuits of a graph. *SIAM Journal on Computing*, 5(1):90–99, March 1976.
- [73] Robert B. McMaster. Automated line generation. *Cartographica*, 24(2):74–111, 1987.
- [74] R. Mehrotra and J. E. Gary. Feature-Based Retrieval of Similar Shapes. In *Proceedings of the 9th International Conference on Data Engineering (ICDE'93)*, pages 108–115, Vienna, Austria, 1993.
- [75] Avraham A. Melkman. Online construction of the convex-hull of a simple polyline. *Information Processing Letters*, 25:11–12, 1987.
- [76] Ugo Montanari. Continuous skeletons from digitized images. *Journal of the ACM*, 16(4):534–549, 1969.
- [77] Stefan Müller and Gerhard Rigoll. Engineering drawing database retrieval using statistical pattern spotting techniques. In *Lecture Notes in Computer Science*, volume 1941, page 246. Springer-Verlag, 2000.
- [78] Stefan MüLlertgot and Gerhard Rigoll. Searching an engineering drawing database for user-specified shapes. In *Proc. of the Int. Conf. on Document Analysis and Recognition (ICDAR'99)*, pages 697–700, Bangalore, India, 1999.

- [79] Mohammad Nabil, Anne H.H. Ngu, and John Shepherd. Picture Similarity Retrieval Using the 2D Projection Interval Representation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):533–539, August 1996.
- [80] Mohammad Nabil, John Shepherd, and Anne H.H. Ngu. An image retrieval system for distributed system. Orlando, Florida, August 1998.
- [81] Nabil Jean Naccache and Rajjan Shinghal. An investigation into the skeletonization approach of hilditch. *Pattern Recognition*, 17(3):279–284, 1984.
- [82] Vijay Nagasamy and Noshir A. Langrana. Engineering drawing processing and vectorization system. *Comput. Vision Graph. Image Process.*, 49(3):379–397, 1990.
- [83] V.S. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley, 1993.
- [84] Lawrence O’Gorman. K x k thinning. *Computer Vision Graphics and Image Processing (CVGIP)*, 51(2):195–215, August 1990.
- [85] Lawrence O’Gorman. Image and document processing techniques for the right pages electronic library system. In *International Conference on Pattern Recognition (ICPR)*, pages 260–263, 1992.
- [86] Lawrence O’Gorman. Primitives chain code. In *Progress in Computer Vision and Image Processing (CVIP92)*, pages 167–183, 1992.
- [87] Joseph O’Rourke. *Computational Geometry in C*, chapter Section 7.7 ”Intersection of Segments”, pages 264–266. Cambridge University Press, 2nd edition, 1998.
- [88] Joseph O’Rourke. *Computational Geometry in C*. Cambridge University Press, 2nd edition, 1998.
- [89] Jong Park and Bong Um. A New Approach to Similarity Retrieval of 2D Graphic Objects Based on Dominant Shapes. *Pattern Recognition Letters*, 20:591–616, 1999.
- [90] Juan-Carlos Perez and Enrique Vidal. Optimum polygonal approximation of digitized curves. *Pattern Recogn. Lett.*, 15(8):743–750, 1994.
- [91] John L. Pfaltz and Azriel Rosenfeld. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471–494, 1966.
- [92] John L. Pfaltz and Azriel Rosenfeld. Computer representation of planar regions by their skeletons. *Communications of the ACM*, 10(2):119–122, 1967.
- [93] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart Ter Haar Romeny, and John B. Zimmerman. Adaptive histogram equalization and its variations. *Computer Vision, Graphics and Image Processing*, 39(3):355–368, 1987.
- [94] William K. Pratt. *Digital Image Processing*. John Wiley & Sons Inc., 1991.

- [95] P.V.C.Hough. A method and means for recognizing complex patterns. U.S. Patent 3,069,654, 1962.
- [96] Jean-Yves Ramel and Nicole Vincent. Strategies for line drawing understanding. In *Proceedings of the 5th IAPR Intl. Workshop on Graphics Recognition*, Barcelona, Catalonia, Spain, 2003. IAPR.
- [97] A. Rosenfeld and J.L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1(1):33–61, July 1968.
- [98] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 71–79. ACM Press, 1995.
- [99] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image Retrieval: Current Techniques, Promising Directions, and Open Issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, March 1999.
- [100] Michael Seul, Lawrence O’Gorman, and Michael J. Sammon. *Practical Algorithms for Image Analysis*. Cambridge University Press, 2000.
- [101] B. Shapiro, J. Pisa, and J. Sklansky. Skeleton generation from x–y boundary sequences. *Computer Graphics and Image Processing*, 15:136–153, 1981.
- [102] R. W. Smith. Computer processing of line images: a survey. *Pattern Recogn.*, 20(1):7–15, 1987.
- [103] Maciej M. Syslo. An efficient cycle vector space algorithm for listing all cycles of a planar graph. *SIAM Journal on Computing*, 10(4):797–808, November 1981.
- [104] H. Tamura. A comparison of line thinning algorithms from digital geometry viewpoint. In *ICPR78*, pages 715–719, 1978.
- [105] Karl Tombre, Christian Ah-Soon, Philippe Dosch, Adlane Habed, and Gérard Masini. Stable, robust and off-the-shelf methods for graphics recognition. In *Proc. of the 14th Int. Conf. on Pattern Recognition*, pages 406–408, Brisbane, Australia, 1998. IAPR.
- [106] Karl Tombre, Christian Ah-Soon, Philippe Dosch, Gérard Masini, and Salvatore Tabbone. Stable and robust vectorization: How to make the right choices. In *Proceedings of the 3rd IAPR Intl. Workshop on Graphics Recognition*, pages 3–16, Jaipur, India, 1999. IAPR.
- [107] Øivind Due Trier and Torfinn Taxt. Evaluation of binarization methods for document images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(3):312–315, 1995.
- [108] Pascal Vaxivière and Karl Tombre. Celesstin: Cad conversion of mechanical drawings. *Computer*, 25(7):46–54, 1992.
- [109] Liu Wenyin and Dov Dori. From raster to vectors: Extracting visual information from line drawings. *Pattern Analysis and Applications*, 2(1):10–21, April 1999.

-
- [110] Ellen R. White. Assessment of line-generalization algorithms using characteristic points. *The American Cartographer*, 12(1):17–27, 1985.
- [111] S. Di Zenzo and A. Morelli. A useful image representation. In *Proceedings of the Fifth International Conference on Image Analysis and Processing*, Singapore, 1989. World Scientific Publishing.

