UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

ÉCOLE DOCTORALE MIPTIS

Laboratoire d'Informatique (EA 6300)

THÈSE présentée par :

The Anh PHAM

soutenue le : 27 novembre 2013

pour obtenir le grade de : Docteur de l'université François - Rabelais de Tours

Discipline/ Spécialité : INFORMATIQUE

Détection robuste de jonctions et points d'intérêt dans les images et indexation rapide de caractéristiques dans un espace de grande dimension

THÈSE DIRIGÉE PAR :

RAMEL Jean-Yves	Professeur, Université François Rabelais de Tours

CO-ENCADRANTS:

DELALANDRE Mathieu	Maître de Conférences, Université François Rabelais de Tours
BARRAT Sabine	Maître de Conférences, Université François Rabelais de Tours

RAPPORTEURS :

TABBONE Salvatore-Antoine	Professeur, Université de Lorraine, France.
OGIER Jean-Marc	Professeur, Université de La Rochelle, France.

JURY :

LLADOS Josep	Professeur, Université Autonoma de Barcelone, Espagne.
TABBONE Salvatore-Antoine	Professeur, Université de Lorraine, France.
OGIER Jean-Marc	Professeur, Université de La Rochelle, France.
KISE Koichi	Professeur, Université Osaka Prefecture, Japon.
RAMEL Jean-Yves	Professeur, Université François Rabelais de Tours
DELALANDRE Mathieu	Maître de Conférences, Université François Rabelais de Tours
BARRAT Sabine	Maître de Conférences, Université François Rabelais de Tours

Acknowledgments

"*Gratitude is the sign of noble souls.*"
– Aesop

First of all, I would like to present my warmest regards to my advisors: Dr. Mathieu Delalandre, Dr. Sabine Barrat, and Professor Jean-Yves Ramel. I am delighted and I feel privileged to be supervised by you all. All the achievement that I have obtained during my PhD are due to your patience, openness and devotion provided to me during the past three years. I appreciate so very much all the open discussion that we have exchanged with each other through a huge number of reading groups during this thesis. I feel blessed to keep forever all your kind guidance, great efforts, and considerable enthusiasm at the bottom of my heart.

I am grateful to the reviewers for taking time to give me valuable comments and suggestions on this manuscript. Despite of my exhaustive effort on fulfilling this work, there are still a number of issues that need to be pointed out and corrected by you. Your comments and feedback shall definitely make this dissertation improved considerably.

I musk thank the administration staff of Laboratory of Computer Science (François Rabelais University of Tours) for their pretty kind assistance and support during three years of my thesis research. Thank you very much for making easier all the administrative works involved in my thesis and part of my life here in Tours. In particular, on behalf of Vietnamese students at Polytech'Tours, I would like to present my special thanks to Professor Jean-Charles Billaut for his openness, kindness, and infinite help and support. I am definitely sure that you all are very friendly, open-hearted, and helpful people that I have ever had the change of knowing you.

Thank you very much all my friends. When I first came here, everything is new and strange for me, but you have helped me a lot. You make me more and more familiar to the new environment. You gave me the chance to participate in our excellent PhD student network in which we can freely discuss about our work and share our ideas together. I realize that you all are very friendly, intelligent, and humorous people. I sincerely hope we shall have the chance to meet together in the future.

At last, I would like to thank the people from my personal side. I am thankful to Vietnam International Education Development (VIED)[1] for awarding scholarship to fulfill

[1] www.vied.vn

my PhD research. Many thanks to my affiliation in Vietnam, Hong Duc University (HDU)[2], for making all the arrangements to allow me to concentrate entirely on my PhD research abroad. Big thanks to Vietnamese Student Association in Tours and Blois (AEViVaL)[3] for your support and promoting me as a vice-president during two years in the past. Thanks a million to the association of Touraine-Vietnam[4] for your considerable encouragement and assistance. Special thanks to my little family, parents and brothers. Your love and moral support actually made me go through the crucial and difficult phases of this thesis. Thank you so very much!

November 27th, 2013 (Tours, France).

The Anh Pham

[2] www.hdu.edu.vn

[3] Association des Etudiants Vietnamiens du Val de Loire (AEViVaL): www.aevival.fr

[4] www.touraine-vietnam.fr

Résumé

Les caractéristiques locales sont essentielles dans de nombreux domaines de l'analyse d'images comme la détection et la reconnaissance d'objets, la recherche d'images, etc. Ces dernières années, plusieurs détecteurs dits locaux ont été proposés pour extraire de telles caractéristiques. Ces détecteurs locaux fonctionnent généralement bien pour certaines applications, mais pas pour toutes. Prenons, par exemple, une application de recherche dans une large base d'images. Dans ce cas, un détecteur à base de caractéristiques binaires pourrait être préféré à un autre exploitant des valeurs réelles. En effet, la précision des résultats de recherche pourrait être moins bonne tout en restant raisonnable, mais probablement avec un temps de réponse beaucoup plus court. En général, les détecteurs locaux sont utilisés en combinaison avec une méthode d'indexation. En effet, une méthode d'indexation devient nécessaire dans le cas où les ensembles de points traités sont composés de milliards de points, où chaque point est représenté par un vecteur de caractéristiques de grande dimension.

Malgré le succès des nombreuses méthodes proposées dans la littérature pour la mise en place de tels détecteurs, aucune approche robuste de détection au sein des images de trait ne semble exister. Par conséquent, la première contribution de cette thèse est de proposer une telle approche. Plus précisément, une nouvelle méthode de détection de jonctions dans les images de trait est présentée. La méthode proposée possède plusieurs caractéristiques intéressantes. Tout d'abord, cette méthode est robuste au problème de déformation des jonctions. De plus, cette méthode peut détecter plusieurs jonctions dans une même zone, supportant ainsi les cas de détection multiple. Ensuite, les jonctions sont détectées avec peu d'erreurs de localisation, caractérisant ainsi la précision de la méthode. La méthode proposée a également une faible complexité algorithmique, lui permettant ainsi de supporter des applications à fort coût de calcul comme la localisation, la recherche ou la reconnaissance de symboles. Enfin, elle est invariante aux transformations géométriques habituelles (rotation, changement d'échelle et translation) et robuste aux déformations communes rencontrées dans les images de documents (comme le bruit d'impression, la basse résolution et artefacts de compression).

Des expériences approfondies ont été menées pour étudier le comportement de la méthode proposée. Celle-ci a été comparée à deux méthodes référentes de l'état de l'art. Les résultats ont montré que la méthode proposée surclasse significativement les approches de l'état de l'art. De plus, cette méthode s'est avérée utile pour la réalisation d'applications de plus haut-niveau. En effet, une application de localisation de symboles a été développée, démontrant que les jonctions détectées pouvaient être un support essentiel à l'extraction des autres primitives graphiques composant le document, permettant ainsi une localisation

et une reconnaissance robustes des symboles.

La seconde contribution de cette thèse traite de l'indexation de caractéristiques. Les méthodes de recherche de plus proches voisins rapides sont devenues un besoin crucial pour de nombreux systèmes de recherche ou de reconnaissance. Bien que de nombreuses techniques d'indexation aient été proposées dans la littérature, leurs performances de recherche restent limitées à certains domaines d'application seulement. De plus, les méthodes existantes, qui sont efficaces dans le cas de la recherche approximative de plus proches voisins, s'avèrent moins efficaces pour ce qui est de la recherche exacte. Les limites de ces méthodes nous ont conduits à proposer un algorithme d'indexation avancé. L'algorithme d'indexation proposé fonctionne aussi bien pour les tâches de recherche approximative que de recherche exacte de plus proches voisins. Des expériences approfondies ont été menées afin de comparer l'algorithme proposé à plusieurs méthodes de l'état de l'art. Ces tests ont montré que l'algorithme proposé améliore significativement les performances de recherche, pour différents types de caractéristiques, par rapport aux méthodes auxquelles notre algorithme a été comparé.

Enfin, les codes source des deux ont été rendus disponibles pour l'intérêt des chercheurs.

Mots clés : Détection de jonctions, caractérisation de jonctions, détection de points d'intérêt, documents graphiques, images de trait, recherche approximative de plus proches voisins, indexation de caractéristiques, arbres de clustering.

Abstract

Local features are of central importance to deal with many different problems in image analysis and understanding including image registration, object detection and recognition, image retrieval, etc. Over the years, many local detectors have been presented to detect such features. Such a local detector usually works well for some particular applications but not all. Taking an application of image retrieval in large database as an example, an efficient method for detecting binary features should be preferred to other real-valued feature detection methods. The reason is easily seen: it is expected to have a reasonable precision of retrieval results but the time response must be as fast as possible. Generally, local features are used in combination with an indexing scheme. This is highly needed for the case where the dataset is composed of billions of data points, each of which is in a high-dimensional feature vector space.

Despite the success of many local detectors in the literature, no robust approach to detect local features in line-drawing images seems to exist. Therefore, the first contribution of this dissertation attempts to bring such an approach. Particularly, a new method for junction detection and characterization in line drawing images is presented. The proposed approach has many favorable features. First, it is robust to the problem of junction distortion. Second, it has the ability of detecting and handling multiple junctions at a given crossing zone. Third, the junctions are detected with a small error of location, highlighting like this method precision. Fourth, the proposed approach is time-efficient supporting different time-critical applications such as symbol spotting/retrieval/recognition. Finally, it is stable to common geometry distortions (e.g., rotation, scaling, and translation) and can resist some typical noise in document images (e.g., produced by scanners, re-sampling or compression algorithms) with a satisfactory level.

Extensive experiments were performed to study the behavior of the proposed approach. Comparative results were also provided where the proposed approach gives much better results than two other state-of-the-art methods. Furthermore, the usefulness of the detected junctions is shown at application level. For this concern, an application to symbol localization is developed. This application shows that the junction features are useful, distinctive, and can be used to support the problem of symbol localization/spotting in a very efficient way.

The second contribution of this thesis is concerned with the problem of feature indexing. Fast proximity search is a crucial need of many recognition/retrieval systems. Although many indexing techniques have been introduced in the literature, their search performance is limited to the application domains where a very high search precision is expected (e.g.,

$> 90\%$). Besides, the existing methods work less efficiently for the case of exact nearest neighbor search. The limitations of these methods have led us to propose an advanced indexing algorithm. The proposed indexing algorithm works really well for both the tasks of exact/approximate nearest neighbor search. Extensive experiments are carried out to evaluate the proposed algorithm in comparison with many state-of-the-art methods. These experiments clearly show that a significant improvement of search performance is achieved by the proposed indexing algorithm for different types of features.

At last, the source codes of our two contributions are made publicly available for the interest of researchers.

Keywords : Junction Detection, Junction Characterization, Junction Distortion, Topology Correction, Edge Grouping, Dominant Point Detection, Graphical Documents, Line-Drawings, Approximate Nearest Neighbor Search, Feature Indexing, Locality-Sensitive Hashing, Clustering trees.

Contents

Introduction **19**

I Junction detection in line-drawing images **29**

1 State-of-the-art in junction detection **31**
1.1 Introduction . . . 31
1.2 Junction detection in computer vision . . . 33
1.2.1 Introduction . . . 33
1.2.2 Edge-grouping-based methods . . . 35
1.2.3 Parametric-based methods . . . 44
1.2.4 Conclusions of junction detection methods in CV . . . 49
1.3 Junction detection in graphical line-drawing images . . . 49
1.3.1 Introduction . . . 49
1.3.2 Skeleton-based methods . . . 50
1.3.3 Contour-based methods . . . 59
1.3.4 Tracking-based methods . . . 66
1.3.5 Conclusions of junction detection methods in line-drawings . . . 71
1.4 Open discussion . . . 72

2 Accurate junction detection and characterization in line-drawings **75**
2.1 Introduction . . . 75
2.1.1 Pre-processing . . . 78
2.1.2 Detection of candidate junctions . . . 78
2.1.3 Distorted zone detection . . . 81
2.1.4 Junction reconstruction . . . 82
2.2 Junction characterization . . . 87
2.3 Complexity evaluation . . . 89
2.4 Experimental results . . . 90

2.4.1 Evaluation metric and protocol . . . 90
2.4.2 Baseline methods . . . 91
2.4.3 Datasets . . . 91
2.4.4 Comparative results . . . 92
2.5 Discussion . . . 99

3 Application to symbol localization 103
3.1 Introduction . . . 103
3.2 Document decomposition . . . 105
3.3 Keypoint matching . . . 107
3.4 Geometry consistency checking . . . 109
3.5 Experimental Results . . . 111
3.6 Discussion . . . 114

II Feature indexing in high-dimensional vector space 117

4 State-of-the-art in feature indexing 119
4.1 Introduction . . . 119
4.2 Space-partitioning-based methods . . . 122
4.3 Clustering-based methods . . . 128
4.4 Hashing-based methods . . . 132
4.5 Other methods . . . 136
4.6 Discussion . . . 139

5 An efficient indexing scheme based on linked-node m-ary tree (LM-tree) 141
5.1 Introduction . . . 141
5.2 The proposed algorithm . . . 142
5.2.1 Construction of the LM-tree . . . 142
5.2.2 Exact nearest neighbor search in the LM-tree . . . 145
5.2.3 Approximate nearest neighbor search in the LM-tree . . . 147
5.3 Experimental results . . . 149
5.3.1 ENN search evaluation . . . 151
5.3.2 ANN search evaluation . . . 151
5.3.3 Parameter tuning . . . 154
5.4 Application to image retrieval . . . 157
5.5 Discussion . . . 159

Conclusions 161

List of Tables

1.1 Related work for junction detection in computer vision 33
1.2 Edge-grouping-based methods for junction detection in CV 34
1.3 Performance evaluation of the junction detectors. 42
1.4 Parametric-based methods for junction detection in CV 43
1.5 Related work for junction detection in document image analysis 50
1.6 Comparison of different skeletonization approaches 51
1.7 Skeleton-based methods for junction detection 52
1.8 Contour-based methods for junction detection 60
1.9 Tracking-based methods for junction detection 67

2.1 Datasets used in our experiments 92
2.2 Comparison of the dominant point detection rates for three scenarios. 97
2.3 Report of the processing time (ms) and the number of detected junctions (in brackets). 98

3.1 Dataset used for symbol spotting in GREC2011. 112
3.2 Experimental results of our system (%). 112
3.3 The detail of the SESYD (floorplans) dataset. 113
3.4 The results of our system for the SESYD (floorplans) dataset. 114
3.5 Comparision of recent methods for symbol localization on the SESYD (floorplans-01) dataset. 114

4.1 Indexing methods based on space partitioning in $\Re^k$ vector space 121
4.2 Indexing methods based on clustering in $\Re^k$ vector space. 129
4.3 Hashing-based indexing methods in $\Re^k$ vector space. 133
4.4 Other indexing methods in $\Re^k$ vector space. 137
4.5 Summary of different approaches for feature indexing in vector space 140

5.1 Report of search time and fraction of searched points for the 5-NN LM-trees. 159

List of Figures

1 A typical architecture of a real time image processing system. 21

2 Example of the SURF keypoints detected for: (a) a natural image; (b) a line-drawing image; (the image (a) is reprinted from [Bay et al., 2008]). . . . 23

3 Example of the detected junction points (small red dots) and junction arms (green straight lines) for a line-drawing image. 24

4 Example of the junction points and junction characterization in the CV field; (reprinted from [Xia, 2011]). 24

5 Example of multiple junction detection: (a) an input image containing a X-crossing zone and its skeleton (thin white lines); (b) the vectorization presentation of (a) with three detected junctions; (reproduced from [Hilaire and Tombre, 2006]). 25

1.1 The detected junctions (small red dots) and junction arms (yellow lines); (reprinted from [Bergevin and Bubel, 2004]). 32

1.2 Different approaches for junction detection in CV and DIA. 32

1.3 Illustration of the process of branch extraction and grouping: (A) a ROI in input image; (B) image decomposition using PCA; (C) the candidate branches and junction; (reproduced from [Bergevin and Bubel, 2004]). . . . 36

1.4 The detected junctions (small red dots) and junction arms (white thin lines); (reprinted from [Bergevin and Bubel, 2004]). 36

1.5 Illustration of orientation vector propagation and curvature update: (a) original orientation vectors; (b) $\vec{v_i}$ is the orientation vector at the low curvature endpoint P_i $(i = 1, 2)$, and propagation of orientation vector starting from P_1; (c) update of the curvature for every line point (e.g., P_3) between the two point P_1 and P_2 using the difference in direction of the two vectors: $\vec{v_1}$ and $\vec{v_2}$ (e.g., dash lines); (reproduced from [Deschênes and Ziou, 2000]). . . 37

1.6 (a) Vector fields (gray short bars) and streamlines (green lines); (b) the partial distance between two streamlines at a relative tracking size s; (c) local scale estimation; (reproduced from [Faas and van Vliet, 2007]). 39

1.7 (a) An input image with a Y-junction; (b) three anchor points q_1, q_2, q_3; (reprinted from [Laganiere and Elias, 2004]). 39

1.8 (a) An original image; (b) the detected contours (thin lines) and junctions (asterisk marks); (reprinted from [Maire et al., 2008]). 41

1.9 (a) An input image; (b) the detected corners and junctions (small black dots on the right figure); (reprinted from [Förstner, 1994]). 45

1.10 Illustration of a piecewise constant model of a 3-junction: (a) the junction model in the image plane; (b) the junction model formulated as a piecewise constant function; (reproduced from [Parida et al., 1998]). 45

1.11 Examples of detected junctions: (a) 2-junction points; (b) 3-junction points; (reprinted from [Parida et al., 1998]). 47

1.12 (a) An ideal L-junction point; (b) the ideal response of a 1D profile for the L-junction point; (reproduced from [Sluzek, 2001]). 47

1.13 (a) The skeleton of an input image; (b) the lines 1 and 2 are grouped (similar to the lines 3 and 4) based on their similar spatial continuity (dash line); (reproduced from [Nagasamy and Langrana, 1990]). 52

1.14 (a) The skeleton of an input image; (b) the initial dominant points; (c) the remaining dominant points after primitive fitting; (reproduced from [Nagasamy and Langrana, 1990]). 53

1.15 The process of refining the location of a dominant point; (reproduced from [Janssen and Vossepoel, 1997]). 54

1.16 Fraction of the obtained vectorization where the false anchor points are marked with the circles (reproduced from [Janssen and Vossepoel, 1997]). . 55

1.17 Illustration of the *Criterion A*: (a) correct fusion of two junction points P_1 and P_2; (b) wrong fusion of two junction points P_1 and P_2. 56

1.18 (a, b) The uncertainty domains (gray zones) defined for a line and a circle primitive; (c) illustration of the junction optimization process; (reproduced from [Hilaire and Tombre, 2001]). 56

1.19 Illustration of primitive grouping: (a) P joins Q by L_P; (b) P joins Q by L_p Q joins P by L_Q; (reproduced from [Hilaire and Tombre, 2006]). 57

1.20 Illustration of the process of skeleton and junction optimization: (a) an input image with its skeleton; (b) the results of skeleton segmentation process; (c) the connectivity graph; (d-f) the process of graph simplification; (g) the final graph; (f) the final skeleton and junction points; (reproduced from [Hilaire and Tombre, 2006]). 58

1.21 Another possible interpretation of the crossing structure in [Hilaire and Tombre, 2006]: (a) an input crossing structure; (b) an interpretation for the crossing in (a) where two T-junctions are constructed. 59

1.22 Illustration of the process of progressive extension of line fitting; (reproduced from [Hori and Tanigawa, 1993]). 61

1.23 Illustration of the process of correcting the junction location; (reproduced from [Hori and Tanigawa, 1993]). 62

1.24 Illustration of the process of gap filling and junction generation after contour segment pairing: (a) without common vector and (b) with common vector; (reproduced from [Han and Fan, 1994]). 63

1.25 Illustration of the process of filling gap and junction generation; (reproduced from [Fan et al., 1998]). 64

1.26 The obtained skeletal vectors and junctions in the work of [Fan et al., 1998]. The displacement of skeletal vectors and junctions are marked by the circles; (reproduced from [Fan et al., 1998]). 65

1.27 Illustration of the contour matching process (see explanation in the text); (reprinted from [Ramel et al., 2000]). 65

1.28 Construction of the structural graph (reprinted from [Ramel et al., 2000]). . 66

1.29 The branch construction process: (a) circle following; (b) branch construction; (reproduced from [Van Nieuwenhuizen and Bronsvoort, 1994]). 68

1.30 Illustration of the junction recovery process: (a) conflict of the width run and the width preservation; (b) a new invalid tracking point; (c) a new valid tracking point on the current branch; (d) a new valid tracking point on a different branch; (reproduced from [Dori and Liu, 1999]). 69

1.31 Illustration of MIC tracking process; (reproduced from [Chiang et al., 1998]). 70

1.32 Bar tracking process (a) and intersection preservation deletion (b); (reproduced from [Song et al., 2002]). 71

2.1 Overview of the proposed approach. 76

2.2 Some examples in which Teh-Chin's technique fails to correctly determine the ROS. Top row: (a) $ROS(p_i) = d$ for any point p_i on a circle with radius R, where $l_d = 2R$; (b) $ROS(p_i) = +\infty$ (i.e., p_i is the middle point of segment q_1q_2 and thus $l_{k+d} > l_k$ for any $k, d > 0$); (c) $ROS(p_i) = 1$ (i.e., $l_{k+1} = l_k$). Bottom row: expected ROS for each case in the top row. 79

2.3 (a) An original image; (b) the detected dominant points (small dots) and the corresponding local scales (small circles). 81

2.4 (a) An input image with the detected *CCDZs* (gray connected components) and reliable line segments (thin white lines); (b) the local topology defined for a *CCDZ*. 82

2.5 Incorrect convergence of the junction optimization step in [Maire et al., 2008]: (a) four line segments with the same length; (b) the detected junction, which is the same distance from the lines 1 and 2; (c) the expected junctions. . . . 83

2.6 Outline of our junction optimization algorithm. 84

2.7 (a) An image with its skeleton; (b) the *CCDZ*(s) and the reliable line segments; (c) the local topology configuration extracted for one *CCDZ* (marked by Z_d); (d) the first cycle of steps $\{1, 2, 3, 4, 5\}$: the junction J_1 is found corresponding to the cluster containing line segments $\{1, 3, 4\}$; (e) the second cycle: the junction J_2 is found corresponding to the second cluster comprised of lines $\{1, 2\}$; (f) the third cycle: the junction J_3 is found corresponding to the third cluster of $\{1, 5\}$; (g) topology correction for three junctions. 86

2.8 Detected junctions (red dots) and local scales (circles). 87

2.9 (a) A *CCDZ* cropped from Figure 2.7 with the superposition of three clusters; (b) detected junctions $\{J_1, J_2, J_3\}$; (c) the detected junctions are classified as a 3-junction even though J_2 and J_3 are constructed from two clusters, $\{1, 2\}$ and $\{1, 5\}$, respectively, each of which only contains two local line segments. . . . 88

2.10 Corresponding junction matches between a query symbol (left) and a cropped document (right). . . . 89

2.11 The evaluation strategy applied to each detector. . . . 91

2.12 Repeatability scores of three systems on rotation change (a), and scaling change (b), where location error is set at 4 pixels. . . . 93

2.13 Repeatability scores of three systems for *setA*, *setB*, *setC*, and *setD* of GREC2011. . . . 94

2.14 Junctions detected (red dots) by the proposed system for *setC* ((a) and (b)) and *setD* ((c) and (d)) of GREC11. False positives caused by context noise in *setD* are marked by dashed-line boxes. . . . 95

2.15 Repeatability scores of three systems for SESYD low resolution dataset (a), and the UMD logo dataset (b). . . . 96

2.16 Junctions detected (red dots) by the proposed system for few Chinese characters and logo images. . . . 97

2.17 Impact of the parameters E_{min} and E_{max}: the repeatability score is computed on the *setC*. . . . 98

2.18 Detected junctions for a synthetic symbol with different levels of noise. . . . 100

2.19 Detected junctions for a real musical score image. . . . 100

2.20 Detected junctions for part of an electronical image. . . . 101

2.21 Detected junctions for a mechanical-text image. . . . 102

3.1 Overview of our symbol localization system. . . . 105

3.2 Basic primitive decomposition and description. . . . 106

3.3 Keypoint-based representation of a simple document image. . . . 107

3.4 The matching process of L-keypoints (a) and A-keypoints (b). . . . 108

3.5 An example of context distortion of the detected junctions: a 3-junction **p** in (a) is distorted as a 5-junction **q** in (b). . . . 108

3.6 Geometry consistency checking: (a) the matches before checking; (b) the matches after checking. . . . 111

3.7 An example of symbol localization: a query symbol (left) and the detected instances of the query (red bounding boxes). . . . 115

3.8 The system fails to detect the symbol "outlet" (left) due to the omission of the E-keypoint and the displacement of the L-keypoint on the instance of the symbol (right). . . . 115

3.9 Few examples of the detected symbols of our system. . . . 116

4.1 Different approaches for feature indexing in vector space. 120
4.2 Illustration of data partitioning of the KD-tree in an 2D space: the resulting KD-tree (left) and the corresponding partitioned space (right). 122
4.3 Illustration of the searching process in the KD-tree: only the regions whose bounding boxes do intersect the circle are inspected. 123
4.4 Illustration of computing the lower bound using the cosine rule of the PA-tree (reprinted from [McNames, 2001]). 124
4.5 The construction of the R-tree (right) for a dataset (left); (reprinted from [Guttman, 1984]). 125
4.6 The construction of the SR-tree (reprinted from [Katayama and Satoh, 1997]).127
4.7 Illustration of the rule 1 (a) and rule 2 (b) for tree pruning; (reproduced from [Fukunaga and Narendra, 1975]) 128
4.8 Construction of the vocabulary tree with the branching factor $K = 3$ (reprinted from [Nister and Stewenius, 2006]). 130
4.9 Illustration of searching process in the multi-probe LSH algorithm: $g_i(q)$ is the hash values of the query q (green buckets), and $g_i(q) + \Delta_i$ are the adjacent buckets for probing (reprinted from [Lv et al., 2007]). 135
4.10 Example of the hashing algorithm in [Auclair, 2009]: Two points x^1 and x^2 are quite similar in a 8-dimensional space; $D(x^1)$ and $D(x^2)$ are the sorted vectors with respect to the distinctive dimensions; $Key(x^1)$ and $Key(x^2)$ are the obtained hash keys where $K = 2$ in this example; (reprinted from [Auclair, 2009]). 136
4.11 Illustration of multi-level indexing in [Roy et al., 2012]. 138
4.12 Shapeme descriptor: (a) an input image; (b) shapeme representation of (a); (reproduced from [Mori et al., 2001]). 139

5.1 Illustration of the iterative process of data partitioning in an 2D space: the 1^{st} partitioning is applied to the dataset X, and the 2^{nd} partitioning is applied to the subset X_6 (the branching factor $m = 6$). 143
5.2 Illustration of the lower bound computation. 146
5.3 Illustration of our bandwidth search with $b = 1$: X_6 is an intermediate node of the considering path, and its adjacent sibling nodes, X_1 and X_5, will also be searched; if $\mathbf{q}$ is too close to the centroid (e.g., inside the circle for the case of X_6), then all of the sibling nodes of X_6 will be searched. 149
5.4 Exact search performance for the SIFT (a) and GIST (b) features. 152
5.5 Approximate search performance for the SIFT (a) and GIST (b) features. . 152
5.6 ANN search performance as a function of the dataset size: (a) the SIFT datasets (search precision = 96%); (b) the GIST datasets (search precision = 95%). 153
5.7 Evaluation of distance error ratio for ANN search applied to SIFT (a) and GIST (b) features. 155

5.8 Search performance as a function of the pruning factor: precision is set to 95%, 90%, and 80% for both the SIFT and GIST features. 156
5.9 Several example images of ornamental drop caps in the dataset. 157
5.10 Search quality of the k-NN LM-trees (a), and search quality versus speedup of the k-NN LM-trees (b). 159
5.11 Few examples of the retrieval results: for each query in the left, the top ranked 5-NNs are showed for the ideal linear scan (the top row) and our LM-trees (the bottom row). 160

Introduction

This dissertation concerns two emerging problems of (i) junction detection and characterization in line-drawing images and (ii) feature indexing in high-dimensional vector space. The former problem is addressed in the context of graphical line-drawing images which is an active sub-area of document image analysis and understanding. The useful applications typically include symbol recognition, symbol localization and spotting, symbol indexing and retrieval, sketch-based retrieval of architectural/electronic images, etc. The later problem is addressed in a more general context of feature vector space in which its applications are broad including those in both computer vision (CV) and document image analysis (DIA) fields. This chapter aims at providing a brief introduction of the context of this thesis. Particularly, three main points are discussed including a general discussion about the CV and DIA fields; the important role of local feature detector, descriptor, and indexing; and the motivation of this work.

Introduction to CV and DIA

Computer vision is a vast and active field of computer science in which its interests are broad including the methods for acquiring, processing, analyzing, understanding, segmenting, and recognizing images from the real world [Morris, 2004]. Some typical applications of the CV field include robot navigation, security monitoring and surveillance, bio-informatics recognition system, people tracking and counting, content-based image retrieval (CBIR), gesture recognition, etc. Among various topics of the CV field, pattern recognition (PR) could be considered as a core and interdisciplinary part whose key goal is to assign a label for a given input data. Generally, pattern recognition algorithms involves in three crucial tasks of object segmentation, feature extraction, and classification. These algorithms are often categorized as structural-based and statistical-based approach. The structural-based approach involves in constructing the structural relations among the primitives. These relations can be represented by the means of syntactics and graphs. Primitive matching is then performed using a grammar parsing process [Tanaka, 1995] or a subgraph matching algorithm [Damiand et al., 2009]. In contrast to the structural pattern recognition methods, the statistical-based approach offers many powerful computation models of clustering, classification, and regression. These models have been proved to be highly efficient to perform many different tasks of object detection, spotting, and recognition. An extensive review of statistical pattern recognition methods could be found in [Jain et al., 2000]. Several attempts have been investigated to combine the advantages of both the approaches.

This is referred to graph embedding as presented in [Riesen et al., 2007, Luqman, 2012].

Document image analysis is another active research field of computer science whose interests are mainly focused on processing, understanding, and recognizing document images. These document images could be acquired by digital machines like scanners, cameras, electronic pen-input devices. Some typical applications of DIA include optical character recognition (OCR), writer identification, signature verification, mathematical formula identification and recognition, historical document retrieval, symbol detection, localization and spotting, etc. Among many different research themes of DIA, the problem of using graphic information is an intensive research activity. Generally, graphics recognition concerns assigning a class label for an input object given a list of pre-defined graphics objects. In DIA, graphic objects can be lines, arcs, arrows, logos, symbols, etc. Earlier works on graphics recognition were considered as a post-process of a vectorization system. In this context, the main goal of graphics recognition is to group the raw results from a vectorization system into different types of graphic objects [Wenyin and Dori, 1998].

Recent works on graphics recognition have moved towards symbol recognition, retrieval and spotting, and performance evaluation [Tombre, 2006]. Logos and trademarks can be considered as a special type of symbol. In this sense, interesting topics are logo detection [Zhu and Doermann, 2007], logo spotting [Rusinol and Llados, 2009a], and logo retrieval [Rusinol and Llados, 2010]. Emerging topics on this field have focused on fast logo retrieval under large-scale datasets [Jain and Doermann, 2012]. On the other hand, in line drawings a symbol can be defined as a graphical entity with a particular meaning in the context of specific application domain. Symbols can serve in different applications including document re-engineering, understanding, classification and retrieval [Rusinol et al., 2010]. A survey of existing works of symbol recognition on logical diagrams, engineering drawings, and maps was reported in [LLados et al., 2002]. Comparative results have been reported throughout a series of symbol recognition contests, which concern the aspects of performance evaluation [Delalandre et al., 2008].

Nowadays, due to the fast growing of portable digital imaging devices (e.g., cameras, smartphones, electronic camera-pen systems), the acquisition of document images becomes much more convenient, and thus the huge expansion of document data emerges. In this expeditious evolution, the real challenges in DIA have shifted to the problems of mass of data. Camera-based document image analysis (CBDAI) has emerged as a very hot research topic [Kim, 2005, Liang et al., 2005, Yin et al., 2007, Vajda et al., 2011]. Large-scale digitization and analysis of historical documents and artworks has also increasingly focused [Lins et al., 2011, Yang et al., 2011, Embley et al., 2011, Landon, 2013]. Open challenges involve the activities of constructing, indexing, and browsing of mass digitized documents. In practices, over the past decade, the interests have moved towards real time document analysis systems. Few examples of such systems include real time graphical symbol spotting in camera-acquired documents [Rusinol et al., 2013], real time camera-based document retrieval [Takeda et al., 2011], fast word spotting in large-scale digital libraries [Roy et al., 2012].

Although the DIA methods involve in handling the document images, there exists some common themes that can be reused from the well established theories in the CV field. Particularly, many powerful statistical computation models such as support vector machine

(SVM), Bayesian networks, neural network, AdaBoost, and hidden Markov model (HMM), can take part of the OCR systems [Parvez and Mahmoud, 2013], writer and signature verification systems [Liwicki et al., 2011], handwriting recognition systems [Graves et al., 2009]. Besides, the structural-based methods using graphs have been extensively used in many different areas of DIA such as clustering of ancient ornamental letters [Jouili et al., 2010], symbol spotting [Qureshi et al., 2008], symbol recognition [Llados et al., 2001], image retrieval [Jouili and Tabbone, 2012]. Moreover, since real time camera-based document analysis has emerged as an important activity in the DIA field, traditional techniques using local features and indexing in the CV field can be employed as the promising solutions [Jain and Doermann, 2012, Takeda et al., 2012].

Local feature detection, description, and indexing

To support a wide range of applications in both the CV and DIA fields, local features have been emerged as an important role in image analysis and recognition [Tuytelaars, 2007]. Especially, local features are very useful to address time-critical applications. Typically, there are three main challenges involved in processing local features: local detector, local descriptor, and indexing. Figure 1 gives an illustration of a general system architecture which is commonly employed in both the CV and DIA fields [Lowe, 2004, Leibe et al., 2006, Jain and Doermann, 2012, Takeda et al., 2012]. We shall base our discussion on this system architecture to explain in detail the context of our work and the connection of the two parts of this work. The discussion of each component of this architecture is also given in the following.

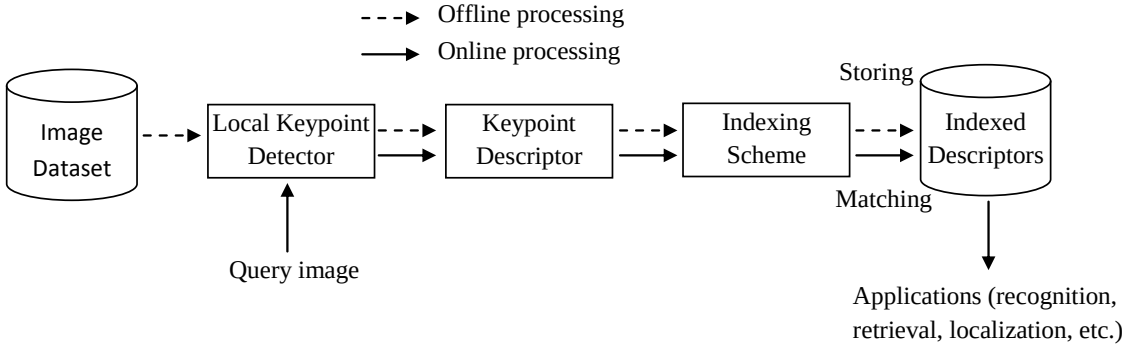


Figure 1: A typical architecture of a real time image processing system.

The first component is served as local keypoint detector. Local keypoints or local features have emerged as a crucial research domain in the literature [Tuytelaars, 2007]. Over the years, a large number of methods for local feature detection have been proposed and widely used in various areas of activity such as image registration and object detection, matching, recognition and retrieval. A *local interest point* or *keypoint*, by definition in the CV field [Tuytelaars, 2007], is a point-like feature in image, which differs from its immediate neighborhood in terms of some image properties such as intensity, color, orientation, texture, curvature. Keypoint detection is one of the most important topics in the CV field

since keypoints are considered as useful features to address the problems of image matching, object recognition and spotting, etc. For that reason, many techniques have been proposed in the literature to detect different types of keypoint. Some examples of well-known keypoint detectors are Harris-Corner [Harris and Stephens., 1988], SIFT [Lowe, 2004], LOG [K.Mikolajczyk and Schmid, 2001], SURF [Bay et al., 2008].

Once the keypoints are detected, each keypoint is characterized by a descriptor as indicated by the second component in Figure 1. Typically, a descriptor for a given keypoint is constructed by extracting the local features within a local neighborhood. The common local features are intensity, color, texture, orientation and shape distribution. The features extracted for a given keypoint are then often represented by a feature vector or keypoint descriptor. Some well-known examples of different types of descriptors are shape-based descriptors (e.g., ShapeContext [Belongie et al., 2002], Shapeme [Mori et al., 2001]), binary-based descriptors (e.g., BRIEF [Calonder et al., 2010], ORB [Rublee et al., 2011]), and real-value descriptors (e.g., GLOH [Mikolajczyk and Schmid, 2005], SIFT [Lowe, 2004]). It is highly expected for a keypoint descriptor to meet the following basic properties: rotation-, scale-, and translation-invariant. Depending on the application domains, other complicated properties which might be needed are illumination-, affine- and viewpoint-invariant. Furthermore, one of the most critical property of a keypoint descriptor is the dimensionality. It is highly desired to design a keypoint descriptor such that it is distinctive enough while retaining the dimensionality as low as possible. The reason of this is to gain efficiency while performing the keypoint matching procedure. A reasonable dimensionality of a local keypoint descriptor would be, perhaps, less than one hundred.

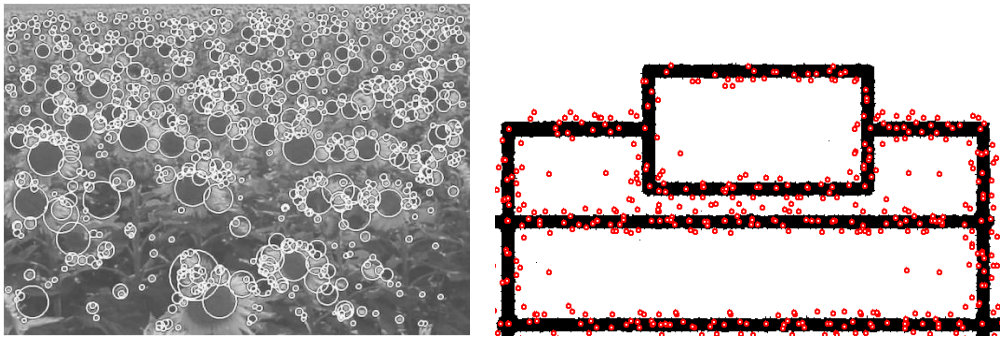
The third component aims at addressing the problem of fast matching to support time-critical applications. Although the processing time can be addressed by reducing the dimensionality of the keypoint descriptor, this process may not significantly decrease time complexity of the system. Moreover, in many applications, the descriptor's dimensionality could be sufficiently high (e.g., > 100) in order to make the descriptors highly discriminative. Therefore, a promising solution for fast matching of the descriptors would be the use of an efficient indexing scheme. Basically, an indexing algorithm is formulated as the task of reorganization of the data, such that it is able to produce quickly the query of proximity search [Beis and Lowe, 1997]. Some efficient indexing schemes in the literature are hashing-based approach (e.g., LHS [Indyk and Motwani, 1998], Kernelized LSH [Kulis and Grauman, 2009]), clustering-based approach (e.g., vocabulary K-means tree [Nister and Stewenius, 2006], randomized K-medoids trees [Muja and Lowe, 2012]), and space-partitioning-based approach (e.g., KD-tree [Friedman et al., 1977], principal axis tree [McNames, 2001]).

A new contribution on junction detection and characterization in line-drawings

Even though a large number of methods have been introduced to deal with different problems of keypoint detector, descriptor and indexing, these methods might not work well for a particular kind of binary line-drawing images. Two main reasons are attributed to this fact. First, it is very often the case that graphical line-drawing images are produced

in the binary form in which the segmentation of background and foreground is obviously clear. Second, the crucial content of graphical line-drawing images is constituted by line-like objects or elongated shapes in which the line thickness is rarely thin (i.e., 1-pixel). Particularly, a typical shape in line-drawing images would be composed of a set of adjacent black pixels. Such a shape could be treated as a homogeneous region in CV. Hence, edge detection in line-drawing images is a trivial task and the transition between the background and the foreground is high and virtually similar in magnitude for every edge point. As a result, the common keypoint detectors in the CV field, like blob-point detector for instance, would produce a significant increase of the detected keypoints. Furthermore, the distinctiveness of the detected keypoint would be highly dropped because of the poor distribution of the local features (e.g., only two values: foreground and background) around the detected keypoints.

To illustrate these issues, Figure 2 provides an example of the detected SURF keypoints for a typical natural image (a) and a line-drawing image (b). It was showed in [Bay et al., 2008] that the detected SURF keypoints for natural images are distinctive and robust. However, from Figure 2 (b), it is evident that lot of keypoints are detected when applying the SURF detector for a line-drawing image even the image is composed of few details.



(a) (b)

Figure 2: Example of the SURF keypoints detected for: (a) a natural image; (b) a line-drawing image; (the image (a) is reprinted from [Bay et al., 2008]).

The arguments above raise an interesting question: what would be the suitable features for such kind of graphical line-drawing images? The answer is naturally related to the content of line-drawing images which are mainly composed of line-like objects. Hence, the expected features would be basic primitives including straight lines, arcs, circles, curves, quadrilaterals, etc. In fact, there is a strong evidence that much of work has been performed to extracted these primitives in the literature [Han and Fan, 1994, Dori and Liu, 1999, Song et al., 2002, Hilaire and Tombre, 2006]. Furthermore, these primitives have been extensively employed as interesting features to address different applications such as engineering drawings recognition [Yan and Wenyin, 2003] and symbol recognition/retrieval/spotting [Dosch and LLados, 2004, Qureshi et al., 2008, Rusinol et al., 2010]. Among many different types of basic primitives, junction point is an important feature in line-drawing images

[Hilaire and Tombre, 2006, Zuwala and Tabbone, 2006]. Junction point serve as a "bridge" point connecting different primitives. Therefore, the accurate detection of other primitives is highly dependent on the correct detection of junction points. From Figure 3, it can be noted that these junction points can provide effective and efficient features for describing such a line-drawing document image.

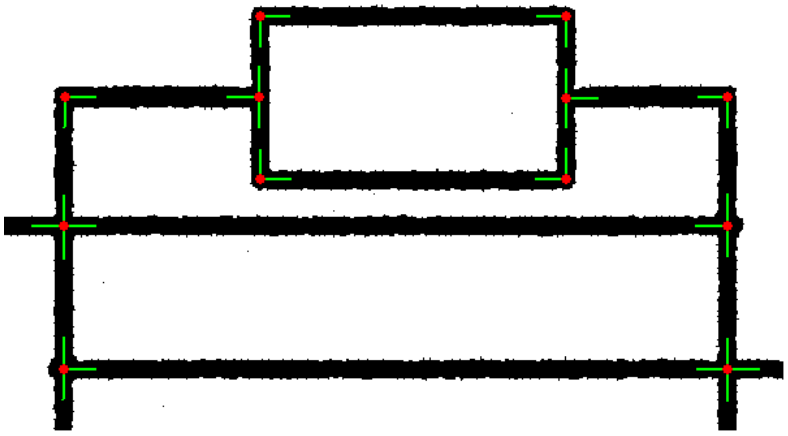


Figure 3: Example of the detected junction points (small red dots) and junction arms (green straight lines) for a line-drawing image.

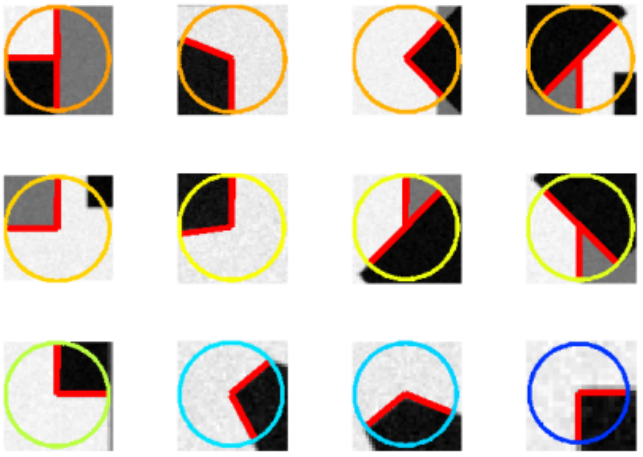


Figure 4: Example of the junction points and junction characterization in the CV field; (reprinted from [Xia, 2011]).

By definition in the CV field, a junction point is defined as the place of intersection of at least two different edge directions in the image [Bergevin and Bubel, 2004]. In addition to the detected junctions, it is desired to extract the junction parameters or junction characteristics such as the number of branches of each junction, the strength and orientation of each junction branch, and the local scale of the junction. It is worth mentioning that there is a close relationship between the junction points and corner points. Although the

terminologies of junction and corner are often interchangeably used in the CV field with a little distinction, there in fact exists some important differences. In the first place, a junction point could be considered as a corner point (i.e., L-, X-, Y-, T-junction), but the converse is not necessarily true (e.g., an endpoint is most likely to be detected as a corner point but this point is clearly not a junction point). In the other place, the characteristic of each junction point also distinguishes a junction detector from a corner detector. As argued in [Laganiere, 2004], a corner detector works as a two-stage process of estimating a measure of corner response for every pixel, followed by a second stage of thresholding to select the strongest corner candidates. On the contrary, a junction detector should simultaneously perform both the tasks of detection and characterization or classification of junction points.

Different approaches have been introduced to deal with the detection of junction points. In the first place, the CV methods detect junction points as the meeting points of edge lines. Such methods are not well adapted to graphical line-drawing images because of several main reasons. First, few works [Faas and van Vliet, 2007, Bergevin and Bubel, 2004, Deschênes and Ziou, 2000] discuss about junction characterization, which is very important for junction features. Second, all of these methods are limited to single junction detection, whereas it is very often for line-drawings to encounter a crossing zone which contains multiple junctions as illustrated in Figure 5. Most importantly, the perception of a junction point in the context of graphical line-drawing images is quite different from that in the CV field. In line-drawings, junctions are treated as the intersections of median lines, whereas the CV methods detect junctions as the intersections of the edges. This point, in fact, makes the CV methods unsuitable.

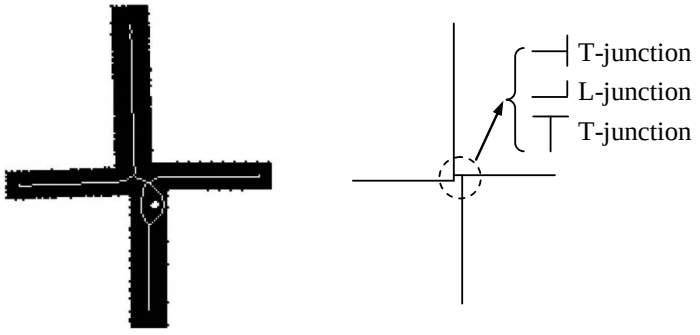


Figure 5: Example of multiple junction detection: (a) an input image containing a X-crossing zone and its skeleton (thin white lines); (b) the vectorization presentation of (a) with three detected junctions; (reproduced from [Hilaire and Tombre, 2006]).

On the other place, the problem of junction detection for graphical line-drawing images has been mainly embedded in a vectorization system [Fan et al., 1998, Song et al., 2002, Hilaire and Tombre, 2006]. The obtained results are therefore subjected to the error-prone caused by the raster-to-vector conversion process. Moreover, such a vectorization system has been known to be sensitive to setting parameters, and present difficulties when heterogeneous primitives (e.g., straight lines, arcs, curves, and circles) appear within a single

document. Knowledge about the document content (objects, entities, document layout) must be included and derived using an appropriate strategy to make the system more robust at the cost of adaptability.

Considering all the aforementioned points, it can be noted that introducing a suitable and efficient junction detection approach dedicated to line-drawing images would be highly appreciated. Therefore, the first contribution of this dissertation is an attempt to bring a robust and accurate approach for detecting and characterizing the junction points in line-drawings. The proposed approach is evaluated using a wide corpus set of line-drawings. Extensive performance evaluation is performed to study the behaviors of the proposed system at different aspects. Comparative results are provided to demonstrate the strengths and weaknesses of our system. Additional experiments at the application level are also given to prove that our junction detector is robust and discriminated enough to be used in the applications of symbol localization and spotting.

A new contribution on feature indexing

As illustrated in Figure 1, it is very often the case that the detected keypoints would be associated with some features or local descriptors. Typically, these feature vectors are computed using statistical histogram of local features within the neighborhoods around the detected keypoints. In order to be highly distinctive for differentiating different objects, the dimensionality of a feature vector must be sufficiently high. However, this raises an expensive computation cost for the subsequent steps of matching the feature vectors and object retrieval/recognition. An efficient solution for this problem is known as feature indexing. Hence, the second contribution of this dissertation is concerned with the problem of indexing in feature vector space. Although a large number of indexing algorithms have been proposed in the literature, few of them (e.g., randomized KD-trees [Muja and Lowe, 2009], hierarchical K-means tree [Muja and Lowe, 2009], randomized clustering trees [Muja and Lowe, 2012], and LHS-based schemes [Lv et al., 2007, Kulis and Grauman, 2009]) have been well validated on extensive experiments to give satisfactory performance of approximate nearest neighbor (ANN) search on some specific benchmarks. However, the search performance of these indexing algorithms is still limited to the cases (e.g., word spotting, face detection, document retrieval) where a pretty high search precision is desired (e.g., $> 90\%$). Especially, for some applications (e.g., logo recognition, handwriting identification, signature verification) where exact nearest neighbor (ENN) search is required, these algorithms give little or even no better search performance compared to the brute-force search.

To overcome such shortcomings, an advanced indexing algorithms in feature vector space is proposed as the second main contribution in this dissertation. Particularly, a new and efficient indexing algorithm is presented based on a linked-node m-ary tree (LM-tree) structure. The proposed indexing algorithm works really well for both the ENN and ANN search. Extensive experiments show that the proposed algorithm gives a significant improvement of search performance, compared to the state-of-the-art indexing algorithms including randomized KD-trees, hierarchical K-means tree, randomized clustering trees, and multi-probe LHS scheme. To further demonstrate the outstanding search performance

of the proposed indexing algorithm, an application of image retrieval is developed. In this work, a wide corpus set of historical books of ornamental graphics is used to stress our indexing system. Performance evaluation, carried out on this dataset, show the outstanding search efficiency of the indexing system.

Organization of the thesis

The rest of this dissertation is organized into two main parts and one conclusion chapter. A brief introduction of these parts is given hereafter.

- **Part 1**: In this part, the contribution on junction detection and characterization in line-drawing images is presented in three chapters as follows:
  - Chapter 1: A detailed review of state-of-the-art methods in junction detection is presented in this chapter. The main characteristics of each method, the connection among them, and the missing issues requiring further treatments are carefully discussed.
  - Chapter 2: The proposed approach for junction detection in graphical line-drawing images is described in great detail in this chapter. Complexity evaluation of the proposed system is included. Extensive experiments are performed to study the behaviors of the proposed system in comparison with other methods. Finally, the main contributions of the proposed junction detector are summarized and the shortcomings are pointed out to identify further extensions.
  - Chapter 3: In this chapter, the usefulness of the proposed junction detector is evaluated at application level. For this purpose, an application to symbol localization in line-drawing images is investigated. This application shows that the junction features are useful and distinctive, and support the problem of symbol localization/spotting in a very efficient way.

- **Part 2**: In this part, two chapters are presented for the second contribution of this thesis on feature indexing. These chapters are shortly described as follows.
  - Chapter 4: A deep review on state-of-the-art methods for feature indexing in high-dimensional feature vector space is presented. The main ideas, advantages, and weaknesses of each method are thoroughly studied. The highlight of an advanced contribution for feature indexing is given as the conclusion of this chapter.
  - Chapter 5: In this chapter, a novel indexing algorithm called linked-node m-ary tree (LM-tree) is presented. The main processes of tree construction and tree traverse are carefully described to deal with both the ENN and ANN search. Extensive experiments are carried out in comparison with the state-of-the-art indexing schemes. At last, an application of image retrieval is provided to demonstrate the effectiveness and efficiency of the proposed indexing algorithm.

- **Conclusions**: In this chapter, the main contributions of the dissertation are summarized. The merits and limitations of this work are also reviewed. Finally, possible lines of improvement are given for future work.

Part I

Junction detection in line-drawing images

Chapter 1

State-of-the-art in junction detection

This chapter presents a deep review of state-of-the-art methods for junction detection. The merits and weaknesses of each method are discussed in great detail. The interconnections among them are evaluated and the potential link to our subsequent contribution is also justified.

1.1 Introduction

In the CV field, the junction points are often detected by finding the prominent points in the image at which the boundaries of the adjacent regions meet as illustrated in Figure 1.1. The edges meeting at a junction point are regarded as the arms of the junction and are used to characterize junctions into different types such as L-, T-, or X-junction.

It has been shown in the literature [Parida et al., 1998, Hansen and Neumann, 2004, Biederman, 1986] that junction/corner points are important features for image analysis. They are critical features for object recognition as suggested in [Parida et al., 1998]. In the work presented in [Hansen and Neumann, 2004, Biederman, 1986], the authors studied and demonstrated the importance of corner and junction points for human object recognition in a number of psychophysical experiments. Especially, the work in [Biederman, 1986] showed that object perception of line drawings is severely degraded when corners or high curvature points are removed. In contrast, this perception is largely preserved when the parts of low curvature are eliminated. Junction features have been used to address different applications in the literature. [Mordohai and Medioni, 2004] employed junction inference for figure completion. [Rubin, 2001] performed a deep study of junction features for surface completion and contour matching. The use of junction features for robotic weld seam inspection has

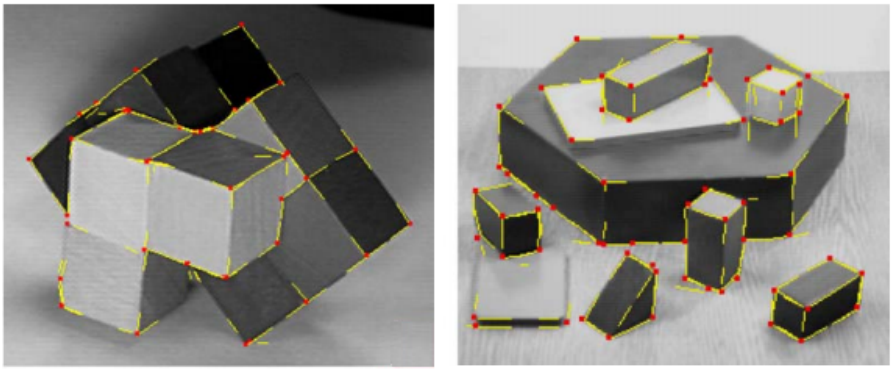


Figure 1.1: The detected junctions (small red dots) and junction arms (yellow lines); (reprinted from [Bergevin and Bubel, 2004]).

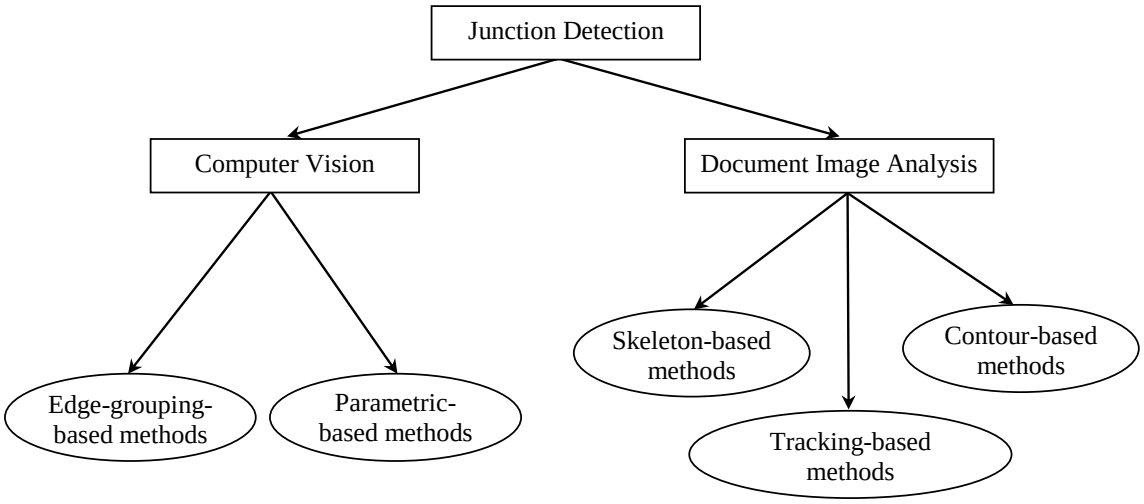


Figure 1.2: Different approaches for junction detection in CV and DIA.

been proposed in [Sluzek, 2001]. Various works [Liu et al., 1999, Lin and Tang, 2002] for stroke extraction and Chinese character recognition have been performed based on junction detection. A comprehensive study of the role of junction features can be found in [Hansen and Neumann, 2004].

Despite abundance of the methods proposed in the CV field [Bergevin and Bubel, 2004, Köthe, 2003, Maire et al., 2008, Parida et al., 1998, Deschênes and Ziou, 2000] to detect junctions, these methods can not directly applied to a particular kind of graphical line-drawing images. The main reason is realized on the fact that the definition of junction is different from that in the CV field. Particularly, a junction point is treated as the intersection of at least two line-like primitives and the problem of junction detection is usually formalized as finding the intersections of median lines in images. However, some specific techniques in CV may be adapted to be part of a junction detector in line-drawings. Therefore, it is of crucial important to identify such possible favourable techniques of the CV for junction detection.

In line-drawing images, junction points are processed with the regard to median lines. Unfortunately, median line extraction is not a trivial task because of the well-known problem of skeleton/junction distortion. One important reason of this problem is due to the line thickness of the shapes. Line-drawings are often acquired using digitization devices and thus the line thickness is rarely perfect (e.g., 1-pixel thickness). Consequently, the median lines extracted at crossing zones are distorted leading to the false formalization of junction points. For these reasons, dedicated methods [Dori and Liu, 1999, Hilaire and Tombre, 2006, Liu et al., 1999, Song et al., 2002] for junction detection in line-drawings have been mainly considered as a post-processing of a vectorization process.

This chapter describes state-of-the-art methods for junction detection including those for natural images in CV and graphical line-drawing images in DIA. Our goal here is to identify the potential techniques in CV that may be adapted to be part of a junction detector in line-drawings, and to make the connection between the both fields. These methods are briefly outlined as shown in Figure 1.2 depending on the application domains and groups of methods. For the CV-based methods, the junction detectors can be classified into two main approaches: edge grouping and parametric modeling. The other methods are dedicated to line-drawing images including skeleton-based, contour-based and tracking-based approaches. In the following sections, we will discuss the most representative works for each of these classes.

1.2 Junction detection in computer vision

1.2.1 Introduction

A number of techniques have been proposed in CV field to deal with the problem of junction detection. These methods are often classified [Parida et al., 1998] into two categories: edge-grouping-based and parametric-based. The former approaches are typically composed of two stages. The first stage detects the edges in image by computing one or several intensity-based gradient maps. The obtained edge maps are grouped in the second stage to generate hypotheses about junction parameters (e.g., junction branches and junction size). For the parametric-based methods, a junction model is first defined for representing the junction characteristics. Next, the junction characteristics are derived by fitting them to some energy functions. A brief list of the methods for junction detection in CV is presented on Table 1.1, and they are detailed in the following sections.

Table 1.1: Related work for junction detection in computer vision

Approach	References
Edge grouping	[Deschênes and Ziou, 2000, Köthe, 2003, Maire et al., 2008, Bergevin and Bubel, 2004, Laganiere and Elias, 2004, Xia, 2011]
Parametric-based	[Förstner, 1994, Parida et al., 1998, Sluzek, 2001, Tabbone et al., 2005, Kalkan et al., 2007]

Table 1.2: Edge-grouping-based methods for junction detection in CV

Method	Scale invariant	Multi-detection	Junction characteristics	Edge and branch detection	Branch grouping	Performance evaluation
Bergevin and Bubel, 2004	No	No	Yes	Local orientation gradient filters, binary tree representation, spatial dispersion and occupancy rate	Adaptive distance threshold	None[a]
Deschênes and Ziou, 2000	No	No	No	Local curvature, orientation propagation	High curvature point detection, threshold-dependent	None
Köthe, 2003	No	No	No	Structural and boundary tensor	Local maxima of the intrinsically 2D energy function, threshold-dependent	None
Faas and van Vliet, 2007	Partial	No	Yes	Local vector field of structural tensor, streamline divergence	Centroid of intersection region	None
Laganiere and Elias, 2004	No	No	No	Local directional maxima of gradients (radial lines)	#Radial lines ≥ 2 and radial line strength $>$ threshold	None
Maire et al., 2008	No	No	No	Local features (e.g., brightness, color, texture gradients) and global features (e.g., generalized eigenvectors)	EM-based optimization, fixed threshold of local window size	BSDS dataset, F-score = 0.41
Xia, 2011	Yes	Yes	Yes	Local normalization of gradients	*a contrario* theory-based decision, adaptive threshold selection	BSDS dataset, F-score = 0.37

[a]Actually, these experiments have been performed on few test images as illustrative example without characterization of the method.

1.2.2 Edge-grouping-based methods

These methods are typically composed of two main stages. The first stage extracts different edge maps in image by computing one or several intensity-based gradient maps.

The obtained edge maps are grouped in the second stage to generate hypotheses about junction branches including the strength, orientation, and the number of the radial lines. Next, junction points are reconstructed as the meeting points of the junction branches. A summary of these methods are presented on Table 1.2 (page 34) along with the merits and limitations of each method. We have based our discussion on different criteria as follows:

- Scale invariant: the ability of detecting the junctions at different scales.
- Multi-detection: the ability of detecting multiple junctions at a given zone.
- Junction characteristics: the ability of extracting junction parameters including the number of junction branches, the strength and orientation of each branch, etc.
- Edge and branch detection: how the method detects the edge and junction branches.
- Branch grouping: how the edges are merged to form the junctions.
- Performance evaluation: which datasets and metrics are used to evaluate the performance of the method.

In [Bergevin and Bubel, 2004], the edge points are first detected using several local oriented-based filters [Heitger, 1995]. The detected edge points, which are of similar amplitude and orientation, are grouped to form the potential junction branches. Next, a branch grouping algorithm is presented applying to each region of interest (Figure 1.3 (A)). This algorithm constructs a binary tree for hierarchically representing the local structure of the edge points within the region of interest. At each level of the tree, the highest variance axis obtained from principal component analysis (PCA) is selected to partition the edge points into two sub-sets (Figure 1.3 (B, G_{11}, G_{21})). This continues as long as the split axis of the underlying subset satisfies the stopping criteria: it has a low spatial dispersion and high occupancy rate. The spatial dispersion measures the uniform distribution of the edge points and the occupancy rate measures the connectivity of the edge points projected on the splitting axis.

Once the tree is constructed, the potential junction branches are detected as the splitting axes obtained during the tree's construction (Figure 1.3 (G_{22}, G_{32})), whereas the rest of data are considered as noise (Figure 1.3 (G_{31})). The constructed branches are finally analyzed to make a hypothesis about junction point (Figure 1.3 (C)). To do so, the junction branches are first segmented as constant curvature primitives corresponding to straight line segments and arc segments. An adaptive threshold computed from contour density and average primitive fitting error is used to select the branches that form a junction point centered on the region of interest under consideration. Junction characterization is also obtained in this stage by determining the characterized primitives passing through the junction location. The junction position is then further improved to obtain a sub-pixel accuracy using an adaptation of the junction localization operator in

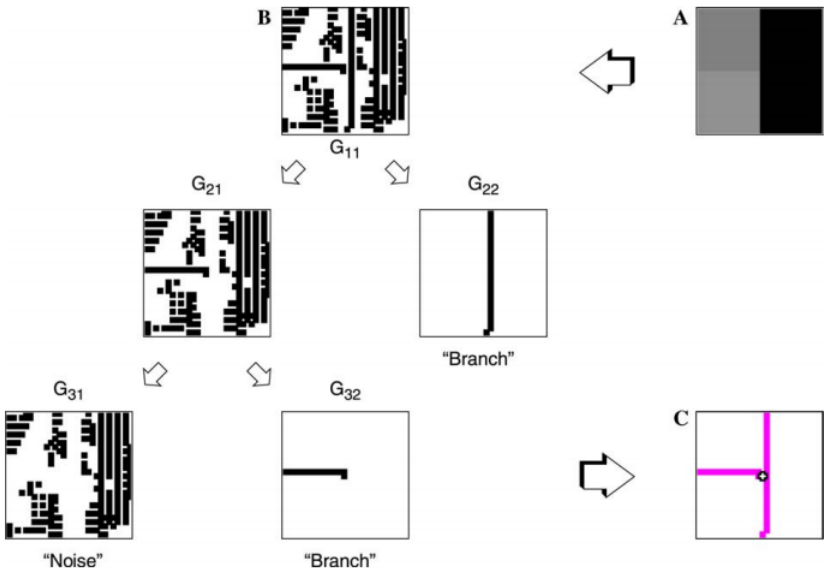


Figure 1.3: Illustration of the process of branch extraction and grouping: (A) a ROI in input image; (B) image decomposition using PCA; (C) the candidate branches and junction; (reproduced from [Bergevin and Bubel, 2004]).

[Förstner and Gülch, 1987]. The demonstration of the proposed method on several simple images are interesting but it is sensitive to a number of predefined parameters: the size of the region of interests, and the thresholds of classifying the low spatial dispersion and high occupancy rate. Figure 1.4 shows some examples of the detected junctions.

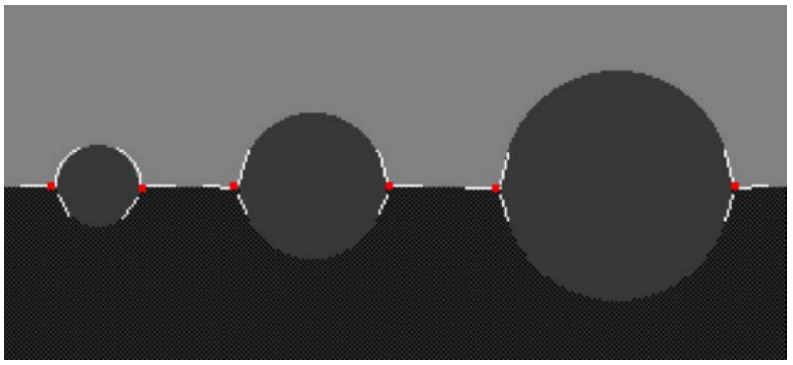


Figure 1.4: The detected junctions (small red dots) and junction arms (white thin lines); (reprinted from [Bergevin and Bubel, 2004]).

[Deschênes and Ziou, 2000] presented a method for detecting specific junction types in gray-level images including L-, X-, Y-, and T-junction, and line terminations. Given the lines extracted from input image using some basic edge operator, the proposed method computes the local curvature measure at each point and then partitions the line points into two classes (i.e., low curvature point and high curvature point) based on a fixed threshold

t_c. Next, a new concept of *low curvature endpoint* is defined as a line point which has a low curvature and belongs to a local neighborhood of a high curvature point. In Figure 1.5, P_0 is a high curvature point, and P_1 and P_2 are two low curvature endpoints. Starting from every low curvature endpoint, the propagation of orientation vector is applied to update the curvature of every line point (e.g., P_3 in Figure 1.5(c)). This is accomplished by compensating a weight computed as the difference in curvature of the orientation vectors of the starting low curvature endpoint and another low curvature endpoint (e.g., $\vec{v_1}$ and $\vec{v_2}$ in Figure 1.5(c)). Particularly, the curvature is updated for every line point $P(x, y)$ between the two endpoints P_1 and P_2 as follows:

$$C(P) = 1 + C_0(P) + s(\vec{v_1}, \vec{v_2}) \tag{1.1}$$

where $C_0(P)$ is the old curvature at P, and $s(\vec{v_1}, \vec{v_2})$ is computed as:

$$s(\vec{v_1}, \vec{v_2}) = 1 - |\frac{\vec{v_1} \cdot \vec{v_2}}{||\vec{v_1}||||\vec{v_2}||}| \tag{1.2}$$

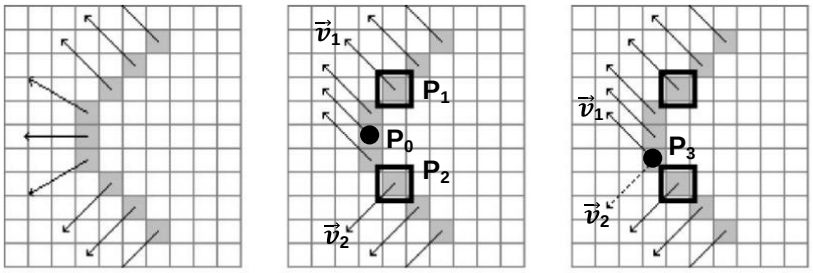


Figure 1.5: Illustration of orientation vector propagation and curvature update: (a) original orientation vectors; (b) $\vec{v_i}$ is the orientation vector at the low curvature endpoint P_i $(i = 1, 2)$, and propagation of orientation vector starting from P_1; (c) update of the curvature for every line point (e.g., P_3) between the two point P_1 and P_2 using the difference in direction of the two vectors: $\vec{v_1}$ and $\vec{v_2}$ (e.g., dash lines); (reproduced from [Deschênes and Ziou, 2000]).

After that, the junctions and line terminations are identified by extracting the local maxima of the updated curvatures within a given neighborhood. The results presented on several real and synthetic images show that the proposed method is able to localize accurately the desirable junctions. Several weaknesses of this method are handling the multi-endpoints in a given neighborhood and handling the size of the neighborhood. In addition, some true junctions might be missed as only the point having maximum curvature is selected among the local maxima of each neighborhood. The classification of low/high curvature point is sensitive to the value of t_c.

[Köthe, 2003] introduced an integrated system for edge and junction detection with boundary tensor computed based on the responses of polar separable filters. A boundary tensor is a 2×2 matrix constructed from a combination of a structure tensor T_{odd} (i.e., first-order partial derivatives) and a Hessian matrix T_{even} (i.e., second-order partial derivatives)

as follows:

$$T_{boundary} = T_{odd} + \frac{1}{4}T_{even} = \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix} \tag{1.3}$$

Such a tensor can be decomposed into two components:

$$T_{boundary} = T_{edge} + T_{junction} = (\lambda_1 - \lambda_2)\vec{e_1}\vec{e_1}^T + \lambda_2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.4}$$

where λ_1, λ_2 are the eigenvalues of $T_{boundary}$, and $\vec{e_1}$ is the unit eigenvector associated with λ_1.

The first component, T_{edge}, encodes the 1-dimensional features (i.e., edge strength and edge orientation) of the current point, while the second component, $T_{junction}$, encodes the 2-dimensional properties such as junction or corner features. Therefore, the junctions can be detected as those points corresponding to the peaks of the second component ($T_{junction}$). This boundary tensor is rotation invariant and provides satisfactory detection rate for some common junction types such as L-, T- and X-junction. However, this approach is subjected to several weaknesses, being scale-dependent and poor localization of the detected junctions.

[Faas and van Vliet, 2007] introduced a streamline method for junction detection and classification in gray-scale images (Figure 1.6). The proposed method first computes vector fields using the eigenvectors corresponding to the smallest eigenvalues of the structure tensors. Next, the vector fields are linked to form the streamlines. More precisely, a streamline is mathematically defined as a curve whose every point is tangent to a local vector field (Figure 1.6(a)). A new measure of partial distance $d_{pq}(s)$ between two streamlines P and Q is defined as the distance of the line connecting two points $P(s)$ and $Q(s)$ where s is a tracking size relative to some pivot point $P(0)$ or $Q(0)$ (Figure 1.6(b)). The basic idea of the junction detection algorithm is that two streamlines originating from $P(0)$ and $Q(0)$ within a local neighborhood of a junction can end up in two different directions with a significant distance $d_{pq}(s)$. By summing up the distances $d_{pq}(s)$ of the streamline passing a pivot point $P(0)$ and the other streamlines within a local neighborhood N, a new measure of streamline divergence (d_p) is derived as follows:

$$d_p = \sum_{q \in N} d_{pq} \tag{1.5}$$

The size of the neighborhood (N) is treated as the local scale and determined as the longest diameter, d_{max}, of the common zone shared by the incident streamlines (Figure 1.6(c)). The junction locations are then detected as those areas having high responses of the streamline divergence. However, such a measure of streamline divergence is quite sensitive to noise. To alleviate this matter, the authors suggested weighting these responses by the certainty of the Hessian matrix. In this way, the responses of streamline divergence corresponding to the background are degraded close to zero. The junctions are then identified as the centers of gravity of the intersection regions. This method is also subjected to several weaknesses being sensitive to the determination of the pivot points and the selection of the parameter s (i.e., $s = 7, 8$ in their experiments).

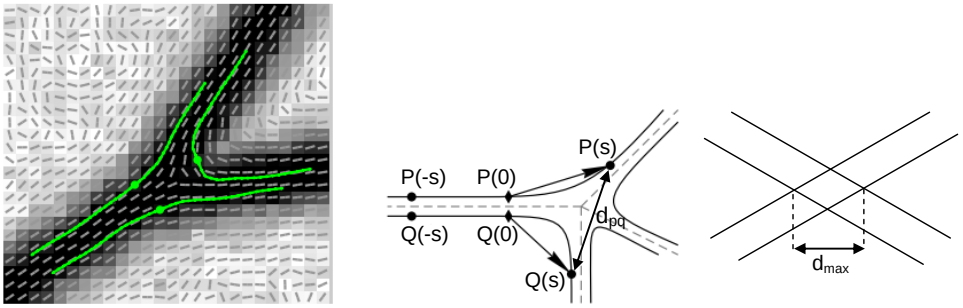


Figure 1.6: (a) Vector fields (gray short bars) and streamlines (green lines); (b) the partial distance between two streamlines at a relative tracking size s; (c) local scale estimation; (reproduced from [Faas and van Vliet, 2007]).

The work in [Laganiere and Elias, 2004] detects the junctions based on intensity gradient. The proposed approach constructs two binary images B and B^+ as follows:

- B is formed by thresholding the gradient image with a threshold t_B.
- B^+ is formed by looking for local maxima in the direction of gradient.

Next, for each point $p \in B$, a list C_A containing the anchor points $q_i \in B^+$ is constructed by choosing the point q_i lying in the boundary of the circle centered at the point p with a radius r (Figure 1.7).

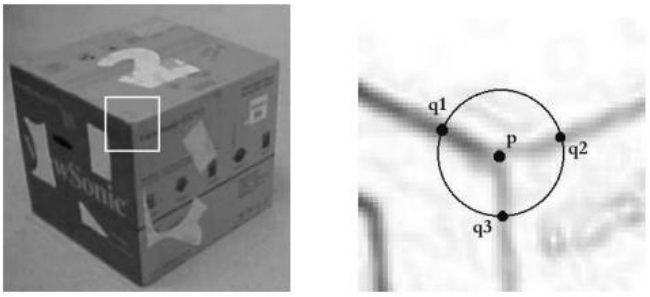


Figure 1.7: (a) An input image with a Y-junction; (b) three anchor points q_1, q_2, q_3; (reprinted from [Laganiere and Elias, 2004]).

Each line d_{qp} connecting an anchor point $q_i \in C_A$ and the corresponding center point p is treated as a candidate junction branch. A candidate branch d_{qp} is accepted as a valid junction branch if there exists a continuous path of points in B so that the distance from each point to d_{qp} is less than 1 pixel. For such a branch d_{qp}, the strength of d_{qp} is defined as the sum of squared distances of each point in B^+ to d_{qp}. The strength of d_{qp} is then normalized based on the length of d_{qp}. Next, every valid junction branch having

the strength less than a threshold t_s is discarded. Finally, the junctions are detected as the points having more than two valid junction branches. In the case of a 2-junction (i.e., junction formed by exact two branches), an additional step is applied to verify that this junction is not a straight line segment. Limited experiments have been provided to demonstrate the performance of the method. This method is sensitive to the selection of the parameters: t_B, t_s, and r.

[Maire et al., 2008] presented a new method for contour and junction detection in natural images. The contours are first detected by combining local features (i.e., brightness, color, texture gradients) and global features (i.e., generalized eigenvectors obtained from spectral clustering) to form a *globalized probability of boundary*. Next, the obtained contours are segmented into a set of straight line segments using a polygon approximation technique. A final process of line segment grouping is performed using an EM-like technique. This iterative process works based on the idea that if we know the position of the junction, the associated line segments passing through this junction could be easily identified and *vice versa*. The algorithm is therefore composed of two iterative steps: estimating an optimized location of the junction within a local neighborhood and updating the weights based on the distances of the newly derived junction to the contour segments nearby. The main idea of this junction optimization method is outlined below:

- Step 1: Estimate an optimized location J of the junction within a neighborhood N:

$$J = \arg\min_{J \in N} \sum_{i=1}^{n} w_i * d(C_i, J) \tag{1.6}$$

  where C_i is a contour segment, n is the number of the contour segments, w_i is the weight associated to the contour C_i, and $d(C_i, J)$ is the Euclidean distance from J to C_i.

- Step 2: Update the weights based on the strengths (i.e., $|C_i|$) of the contour segments:

$$w_i = |C_i| \cdot \exp((\frac{-d(C_i, J)}{\epsilon})^2) \tag{1.7}$$

  where ϵ is a distance tolerance determined empirically.

- Step 3: Repeat Step 1,2 until the junction J converges to a fixed point or until a number of iterations is done.

This method assumes that each neighborhood N contains at most one junction point and thus it is not able to detect multiple junctions, if any. Besides, it is subjected to high computation load because the size of the neighborhood must be sufficiently large to ensure that it is less impacted by the error-prone introduced by the previous step of contour detection. The authors suggested choosing the neighborhoods around the terminations of the contour segments. In addition, the reweighting step (i.e., Step 2) does not consider the weights accumulated during the previous iterations. This omission may lead to incorrect junction convergence for the case that all the contour segments has the same strength.

The proposed method has been evaluated on the BSDS benchmark dataset[1] using the F-score metric. The BSDS is composed of 300 natural images whose boundaries have been manually segmented by different human subjects. From the ground-truth segmentations, the 3-junctions were extracted as the places of intersection of at least three regions. The 2-junctions were also extracted as places of high curvature points along the contours hand-drawn by the humans. Combination of these two types of junctions is served as junction ground-truth. Precision, recall, and F-score were employed as evaluation metrics where a true detection is validated if the distance between the detected junction and the ground-truth one is smaller than 6 pixels. The obtained results are quite interesting with F-score = 0.41 compared to the human agreement of F-score = 0.47. An example of detected contours and junctions is shown in Figure 1.8.

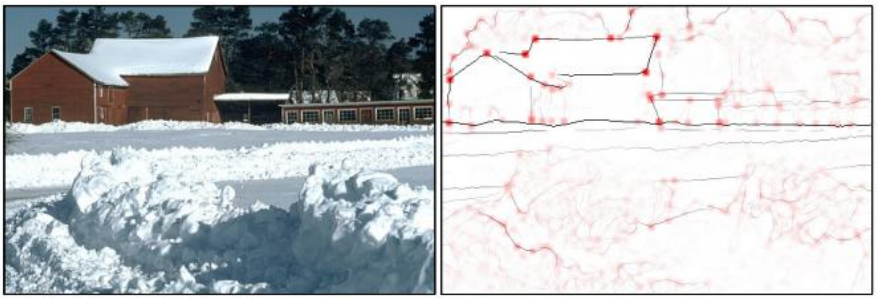


Figure 1.8: (a) An original image; (b) the detected contours (thin lines) and junctions (asterisk marks); (reprinted from [Maire et al., 2008]).

Recently, [Xia, 2011] introduced the concept of meaningful junctions based on the *a contrario* detection theory originally proposed by [Desolneux et al., 2000]. This detection theory basically states that an observed image feature is meaningful if it is unlikely to occur randomly under some null hypothesis H_0. The advantage of the *a contrario* theory is that it provides a way of automatically computing the threshold for creating a binary decision for determining a meaningful junction point. This is accomplished by controlling an expected number of false detections. Scale invariant was achieved by employing a multi-scale approach. Experiment was performed on the BSDS benchmark, and the comparative results are summarized on Table 1.3. One of the advantages of this method is that it requires fewer parameters, while detecting accurate junction points. Although the proposed method employs multi-scale approach to detect the junction, the first step of local normalization of gradient is scale-dependent (the neighborhood size was fixed at 5×5). Therefore, the detection rate is sensitive to the parameter tuning. Actually, the proposed method achieved the F-score of 0.38 with respect to the neighborhood size of 11×11.

In summary, we will highlight hereafter several key remarks. For performance evaluation, since there is a lack of standard benchmarks for the junction detection algorithms, it gives a rise of difficulties to evaluate the real performances of the existing works. Most of the existing methods performed the experiments on their own and pretty small datasets. No evaluation metrics and no comparative results were reported for performance evaluation

[1] www.cs.berkeley.edu/projects/vision/grouping/segbench/

Table 1.3: Performance evaluation of the junction detectors.

System	Dataset	F-score
[Xia, 2011]	BSDS	0.38
[Maire et al., 2008]	BSDS	0.41
[Harris and Stephens., 1988]	BSDS	0.28
Human agreement	BSDS	0.47

except the works of [Maire et al., 2008, Xia, 2011]. Typically, these experimental results were carried out by visually showing the detected junctions for few test images. Although [Maire et al., 2008] have recently provided the BSDS benchmark for the field, limited performance evaluation of different techniques has been carried out. Only [Xia, 2011] reported their results on this benchmark in comparison with the work of [Maire et al., 2008] and a classical Harris corner detector. General speaking, these works achieved quite good results with respect to the human agreement in terms of junction detection. [Maire et al., 2008] obtained better results than the two others probably due to combining local and global cues in contour detection.

It can be concluded that there are two main challenges for the edge-grouping-based junction detectors. The first one concerns the extraction of junction branches from image features such as local curvatures or local gradients [Bergevin and Bubel, 2004, Köthe, 2003, Faas and van Vliet, 2007]. However, these basic features are sensitive to contrast change and noise. To alleviate this problem, [Xia, 2011] normalizes the extracted gradients using the mean and standard deviation of local gradient within a local neighborhood. The obtained results showed that the proposed system is quite robust to the contrast change, but it is subjected to the proper setting of the size of the local neighborhood. [Maire et al., 2008] proposed a more robust approach by combining the local and global cues for contour extraction. They obtained the best results in terms of junction detection for their benchmark dataset.

The second challenge of every edge-grouping-based junction detection method is the selection of junction branches to form a junction. Typically, this is accomplished by thresholding and grouping the survived branches nearby. Consequently, this gives another difficult issue: the selection of a proper distance threshold. A high value of the distance threshold will reduces the false detections but may miss some true junctions, and vice versa. Some methods just empirically fixed these values as in the cases of [Köthe, 2003, Laganiere and Elias, 2004, Maire et al., 2008]. Interestingly, [Xia, 2011] exploited the *a contrario* detection theory for junction branch grouping provided in advance the expected number of false detections. That said, given a specific number of false detections, the junction detector can automatically decide which points are likely to be the useful junctions. [Bergevin and Bubel, 2004] addressed this matter by adaptively deriving the distance thresholds based on the local contour density and average primitive fitting error obtained previously.

Table 1.4: Parametric-based methods for junction detection in CV

Method	Scale invariant	Multi-detection	Junction classification	Junction model	Fitting criteria	Performance evaluation
Förstner, 1994	No	No	No	Corner function using average squared gradients	Local minimum of regularity measure, adaptive threshold selection	None[a]
Parida et al., 1998	Yes	No	Yes	Piecewise constant functions	Local minimum of radial intensity variation, thresholding the relative error of radial variation and fitting energy	None
Sluzek, 2001	No	No	No	1D orientation profile of intensity	Local peak of the junction profile, fixed threshold dependent	None
Kalkan et al., 2007	No	No	Yes	Harris corner function	Local minimum of intersection consistence, hard thresholding 1D orientation junction profile	None

[a]Actually, these experiments have been performed on few test images as illustrative example without characterization of the method.

1.2.3 Parametric-based methods

In the preceding section, we described the methods that detect junctions based on branch detection and branch grouping. In this section, we are going to discuss the most representative parametric-based methods for junction detection. For the parametric-based methods, a junction model is first constructed. This model formalizes the junction parameters including junction's location, junction's scale, orientation and magnitude of each junction branch, etc. Next, these parameters are derived by fitting them to one or several energy functions designed to explicitly reflect the junction model.

Table 1.4 summaries some typical parametric-based methods for junction detection. The first three evaluation criteria have the same interpretation as presented in Table 1.2 (page 34), whereas the last ones indicate the main idea of the parametric-based methods such as the definition of junction model and model fitting. The detail of each method is given in the paragraphs thereafter.

[Förstner, 1994] introduced a general framework for low level feature extraction such as corners, junctions, edge points, and circular symmetric features. Particularly, the authors introduced some local image characteristics such as the average squared gradient and regularity measure. The average squared gradient is defined as the convolution of a 2×2 squared gradient matrix to a rotationally symmetric Gaussian function. By analyzing the eigenvalues and eigenvectors of the resulting matrix, the corners and junctions can be detected as those having large values of the two eigenvalues λ_1 and λ_2.

The regularity measure is designed to estimate the location of the junctions and circular symmetric features. Given a local patch centered at p, the regularity measure $S(p, \sigma)$ is defined as follows:

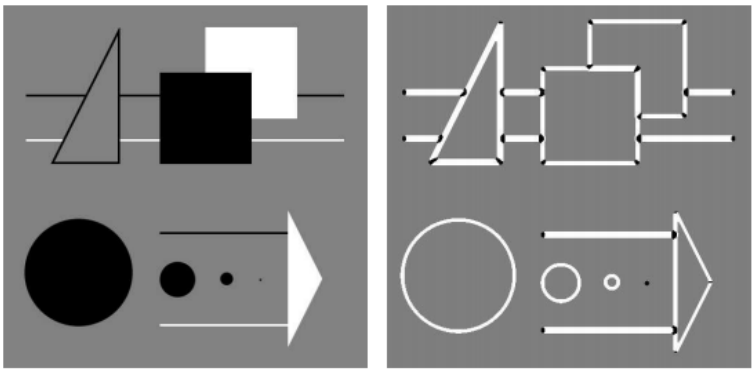
$$S(p, \sigma) = \int \int d^2(p, q) ||\nabla g(q)||^2 G_\sigma(||p - q||) dq \tag{1.8}$$

where $d(p, q)$ is the Euclidean distance between p and q, and $||\nabla g(q)||$ is the gradient magnitude at q, and G_σ is the Gaussian function.

The accurate location of the junction and corner candidates can be obtained by minimizing the regularity measure within a 3×3 window. Figure 1.9 shows an illustrative example where all the corners and junctions are correctly detected. However, in contrast to edge-grouping-based methods, all the steps of the proposed method purely rely on low-level feature extraction without incorporating high-level processing or scene knowledge analysis. Hence, the method is sensitive to noise. To alleviate this problem, image smoothing using Gaussian function can be applied, but it raises another concern of scale selection.

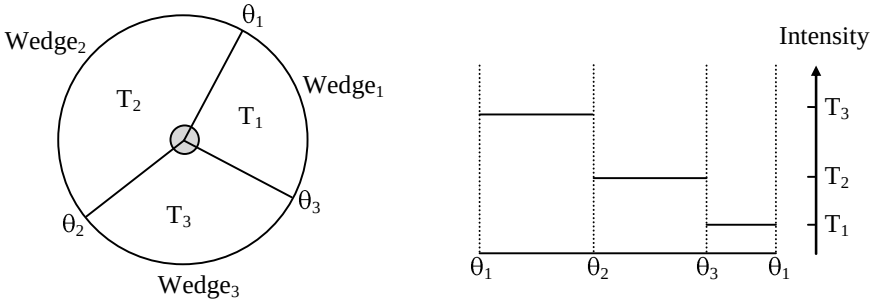
[Parida et al., 1998] formalized a junction model as a small disk of image in which the intensity values are piecewise constants in the incident homogeneous regions (i.e., the wedges) pointing towards the center of the disk (Figure 1.10). Particularly, the junction model involves the following parameters: the central point, the radius of the disk, the number of wedges, the intensity value in each wedges, and the junction branches separating two adjacent wedges.

Typically, a junction model is formalized as an energy function:



(a) (b)

Figure 1.9: (a) An input image; (b) the detected corners and junctions (small black dots on the right figure); (reprinted from [Förstner, 1994]).



(a) (b)

Figure 1.10: Illustration of a piecewise constant model of a 3-junction: (a) the junction model in the image plane; (b) the junction model formulated as a piecewise constant function; (reproduced from [Parida et al., 1998]).

$$E = \lambda \Re + \overline{E} \tag{1.9}$$

The term $\Re$ (i.e., Equation 1.10) is regarded as the radial variation of intensity, and is constructed as the sum of weighted squared gradients within a local neighborhood. The gradient at a given point (r, θ) is weighted by the squared distance of the center of the neighborhood to that point, where (r, θ) is polar coordinate of the point.

$$\Re(r) = \int_0^r \int_0^{2\pi} s^2 (\frac{\partial I}{\partial s})^2 ds d\theta \tag{1.10}$$

The $\Re(r)$ behaves similar to that of the regularity measure in [Förstner, 1994]: it moves towards a minimum value at a junction location. Therefore, the precise location of a junction can be derived by finding the point which has a minimum $\Re(r)$ within a local neighborhood. Also, the scale of a junction can be estimated by thresholding the relative error $e_s(r)$ defined as follows:

$$e_s(r) = \frac{\Re(r)}{\Re(r+1)} < \tau_{\varpi} \tag{1.11}$$

where τ_{ϖ} is a fixed threshold.

The term $\overline{E}$ is used to estimate the junction characteristics including the number of junction branches (n), the orientation of each branch θ_i with $i \in \{1, \ldots, n\}$, and the intensity T_i in each junction wedge. These junction parameters are derived using the following procedure:

- Fix the number of junction branches, say that n.
- Uniformly sample the range $[0, 2\pi]$ into d regular intervals.
- Explore all possible discretized sets of the angles θ_{k_i} with $i = 1, \ldots, n$, and look for the solution that minimize the fitting energy $\overline{E}^n$.

In the procedure above, the fitting energy $\overline{E}^n$ is formalized as the sum of squared differences of the data intensities and the fitted values T_i(s). Assume that the orientations θ_{k_i}(s) are known, the fitted intensity T_i can be estimated as the average value of the data in the corresponding wedge. However, it still needs to determine the optimized number of junction branches. This is accomplished by varying the value n and measuring the decrease in the optimized energy $\overline{E}^n$. The optimized number of junction branches is derived if the following relative error (r^n) is small enough:

$$r^n = \frac{\overline{E}^{n+1}}{\overline{E}^n} < \tau_c \tag{1.12}$$

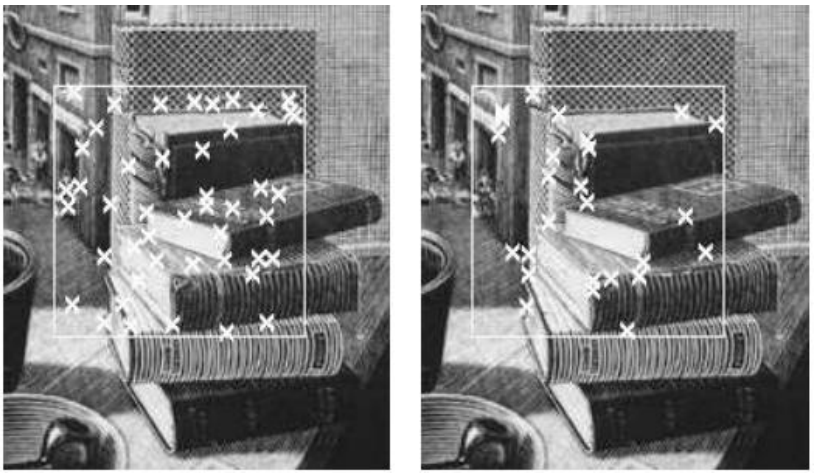
where τ_c is a fixed threshold. In their experiments, with a fixed setting for the two thresholds: $d = 16$, $\tau_{\varpi} = 2.1$ and $\tau_c = 0.4$, the authors provided some illustrative results using several synthesized and real images (Figure 1.11).

[Sluzek, 2001] formalized a junction model as a summation function $g(\theta)$ of image intensity along a given angular direction θ within a local circular window of size R ($15 < R < 30$).

$$g(\theta) = \int_{\alpha=-\pi}^{+\pi} \int_{r=0}^{R} I(r\cos\alpha, r\sin\alpha)\delta(\alpha - \theta) dr d\alpha \tag{1.13}$$

where (r, α) is the polar coordinate of a point, and $\delta(x)$ is equal to either 1 or 0 taking into account $x = 0$ or $x \neq 0$.

The local function $g(\theta)$ is computed for every edge point yielding a 1D profile representing the characteristics of a junction, if any. At a junction location (Figure 1.12), such a 1D profile presents several spikes and the position of spikes indicates the geometry of the junction. Identification of spikes is not an easy task since it is a highly threshold-dependent process. The authors suggested two-step normalization process for dealing with this issue. The first step is to normalize the response of $g(\theta)$ inversely proportional to the number of pixel in the local window, and the second step is to rescale this response proportionally to



(a) (b)

Figure 1.11: Examples of detected junctions: (a) 2-junction points; (b) 3-junction points; (reprinted from [Parida et al., 1998]).

the number of junction branches. A final process of template matching is applied to measure the dissimilarity scores between the junction profile and the model junction profiles. By thresholding these dissimilarity scores, the junction is either accepted or rejected. Few test examples were used to demonstrate the performance of the proposed method.

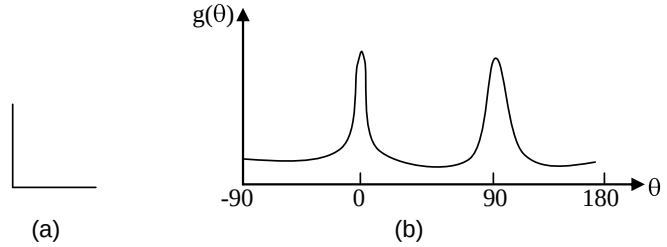


Figure 1.12: (a) An ideal L-junction point; (b) the ideal response of a 1D profile for the L-junction point; (reproduced from [Sluzek, 2001]).

[Kalkan et al., 2007] proposed two new improvements enabling a classical corner detector (Harris [Harris and Stephens., 1988], SUSAN [Smith and Brady, 1995], etc.) to be a semantic junction detector. The first advancement concerns improving the location of the detected corners. This is accomplished by designing a new measure of *intersection consistence* (IC), which is similar in spirit to the function $\Re(r)$ in [Parida et al., 1998]:

$$IC(p_c) = \int_{p \in P} c_{i1D}(p)^2 (1 - d(l_p, p_c)/d(p, p_c)) dp \tag{1.14}$$

where the involved parameters are described as follows:

- p_c is the center of some local image patch P;

- p is an image pixel in P;
- c_{i1D} is the gradient strength of p;
- l_p is the line passing p with a slope of θ_p computed as the local orientation at p;
- $d(x, p_c)$ is the Euclidean distance between p_c and x.

The $IC(p_c)$ could be interpreted as a measure of how the pixels in P pointing towards the center p_c. Therefore, $IC(p_c)$ will be high if P contains one local line passing p_c or p_c is a junction formed by the intersection of several edge lines. Thanks to this characteristic, the improved location of a corner q is achieved by looking for a point having local maximum IC within a local neighborhood of q. The second improvement concerns the estimation of junction characteristics such as junction branches and the strength of the branches. This is accomplished by clustering and thresholding an energy histogram over the junction-relative orientations (i.e., essentially similar to the function $g(\theta)$ in [Sluzek, 2001]). The experimental results showed a significant improvement of junction localization and junction characteristic compared to the Harris and SUSAN corner detectors.

[Tabbone et al., 2005] carried out an in-depth study of the behavior in the scale space of the extrema of Laplacian of Gaussian (LoG). Different junction models have been investigated including linear junction models, non-linear junction models, and linear junction multi-models. Here, a linear model is concerned with the straight edges forming the junction, whereas a non-linear model is associated to the curved straight edges. For the linear junction multi-models, two models are considered including infinite and finite models. They are called multi-models as each contains several adjacent L-junctions. Based on these behaviors, several key conclusions have been drawn:

- Each of these junction models contains at least one LoG extremum point inside the junction sectors;
- The LoG extrema inside the sectors of a junction model are very robust even if the junction model is distorted by geometric transformation and illumination change;
- These LoG extrema can be good starting points to detect junction points.

Although the authors concluded that the LoG extrema are reliable to detect junction points, they did not provide any hints on how to detect the junctions from these extrema. Furthermore, it was concluded that each junction model provides one or several extrema inside the junction sectors but the converse is not studied. More precisely, the LoG extrema appear not only inside the junction sectors but also at different locations. This indeed raises a real challenge as we do not know yet a criterion to justify whether a given LoG extremum point does appear inside a junction sector nearby or it does not.

In summary, the parametric-based methods work from a junction model following model fitting for junction optimization. The junction model can be constructed as a corner model [Förstner, 1994, Kalkan et al., 2007], a piecewise constant function [Parida et al., 1998], or a local 1D orientation profile [Sluzek, 2001]. For the model fitting, [Parida et al., 1998,

Kalkan et al., 2007] employed the same criterion (i.e., regularity measure) originally developed by [Förstner, 1994]. This criterion measures the regularity distribution or intersection consistency of local intensities within a neighborhood. However, the original work of [Förstner, 1994] is limited to simple junction detection without junction characterization, whereas the later works presented robust techniques to estimate junction parameter using dynamic programming [Parida et al., 1998] or semantic interpretation [Kalkan et al., 2007]. Automatic scale selection was addressed in [Parida et al., 1998], but not discussed in [Förstner, 1994, Kalkan et al., 2007, Sluzek, 2001].

1.2.4 Conclusions of junction detection methods in CV

To conclude this section, we shall highlight hereafter several key remarks for both the merits and weaknesses of the existing methods. At first, most of these methods are scale-dependent, and restricted to limited experiments without comparative evaluation with other methods. Experimental results have been mainly performed by giving some illustrative results of the detected junctions for few images. Limited comparisons have been carried out merely using the F-scores on the BSDS benchmark [Maire et al., 2008, Xia, 2011]. Few works [Faas and van Vliet, 2007, Bergevin and Bubel, 2004, Deschênes and Ziou, 2000] discuss junction characterization, which is very important for junction features. Furthermore, the CV methods for junction detection are too much dependent on the analysis and extraction of junction branches. Most of these methods address this matter using the local features and thus are sensitive to noise. The use of combined local and global features has been proposed in [Maire et al., 2008] to make this stage much more robust. Besides, a number of parameters are needed for these methods. Interestingly, the use of the powerful *a contrario* detection theory for junction branch grouping has been shown to give quite good results. Finally, all of these methods are limited to single junction detection in the senses that the considering region of interest contains exactly one junction point.

1.3 Junction detection in graphical line-drawing images

1.3.1 Introduction

In contrast to the definition of the junction point in CV, the junction points in line-drawings are treated as the meeting points of the median lines. However, the task of extracting median lines is not trivial. To be more specific, the major problem is the sensitiveness to small perturbations of the shape boundary and the line thickness of the shapes. This matter results in unwanted skeleton branches which pose another serious problem of junction distortion. That is, both the position of skeleton and junction points are not correctly extracted. Despite of these weaknesses, several methods for junction detection are skeleton-based. The methods start from skeleton and incorporate high-level post-processing steps to correct the junction location. Other works, which try to avoid the problems resulting from skeletonization step, are based on contour matching and element tracking. Table 1.5 summaries these approaches and the details are discussed in the next sub-sections.

Table 1.5: Related work for junction detection in document image analysis

Approach	References
Skeleton-based	[Nagasamy and Langrana, 1990], [Janssen and Vossepoel, 1997], [Liu et al., 1999], [Hilaire and Tombre, 2001], [Hilaire and Tombre, 2006]
Contour matching	[Hori and Tanigawa, 1993, Han and Fan, 1994, Lee and Wu, 1998], [Fan et al., 1998, Ramel et al., 2000]
Element tracking	[Van Nieuwenhuizen and Bronsvoort, 1994, Chiang et al., 1998, Dori and Liu, 1999, Song et al., 2002]

1.3.2 Skeleton-based methods

In shape analysis, the term "skeleton" of a shape is referred to a thin representation of the shape that is equidistant to its boundaries. As argued by [Bai et al., 2007], the skeleton of a shape should meet the following properties:

- Preserve the topology of the original shape,
- Extract accurately the location of skeleton points,
- Robust to small distortions of contours and geometric transformations,
- Reversible,
- Represent significant visual parts of the shape.

A comprehensive survey for these methods can be found in [Lam et al., 1992]. Some common approaches for skeletonization are based on iterative thinning [Rosenfeld, 1975, Kwok, 1988], Voronoi diagrams [Näf et al., 1997, Ogniewicz and Ilg, 1992], veinerization [Deseilligny et al., 1998], and distance transform [Niblack et al., 1990, di Baja, 1994]. Table 1.6 summarizes the main features of the traditional skeletonization approaches. Iterative thinning algorithms extract the skeleton by recursively removing the outer boundaries of the object. The resulting skeleton is connected and preservers the topology of the original shape, but it is sensitive to small distortions of boundaries. The veinerization-based technique [Deseilligny et al., 1998] constructs a graph containing rich topological information of the object that is useful to extract different levels of skeleton. The final results are depending on specific applications. For instance, one can select a particular parameter setting to obtain skeleton satisfying one or few specific properties. The Voronoi diagram methods partition the image into regions, each of which is closer to a particular pivot point than others. The pivot points are selected on the boundaries of the object. The skeleton is obtained by extracting the boundaries dividing the regions. The distance transform based methods calculate the shortest distance of each foreground pixel to the background. The skeleton is extracted by finding the pixels having local maximal distance transform.

Consequently, the resulting skeleton is not connected, and an additional step of linking the skeleton points must be incorporated.

Table 1.6: Comparison of different skeletonization approaches

Approach	Geometry invariance	Reversible ability	Junction accuracy	Complexity
Iterative thinning	Yes	No	Low	High
Voronoi diagrams	Yes	Yes	Low	High
Veinerization	Yes	Yes	Medium	High
Distance transform	Partial	Yes	Medium	Low

Despite a large number of existing methodologies for skeletonization, it is difficult to obtain a satisfactory skeletonization algorithm as argued by [Deseilligny et al., 1998]. Recent advanced methods for skeletonization are presented by [Bai et al., 2007, Aslan et al., 2008, Shen et al., 2011, Ward and Hamarneh, 2010]. [Bai et al., 2007] proposed a new skeletonization method using contour partitioning obtained by discrete curve evolution (DCE). The DCE algorithm starts from a polygonal approximation of the contours and iteratively simplifies the boundary polygon. [Aslan et al., 2008] extracted the disconnected skeleton segments for a single shape at a coarse scale. This method successively smoothes the contours until the resulting shapes are simple enough. The skeleton segments are then detected as the symmetry points from the smoothed curves. To make the system more robust to contour distortion, some skeletonization methods [Ward and Hamarneh, 2010, Shen et al., 2011] introduce the concept of branch significance measure to prune the spurious skeleton branches. Although the obtained results are promising, these methods are not related to the junction detection problem and are thus not suitable when dealing with complex and distorted junction in line-drawing images.

In summary, despite a large number of existing methodologies, it is difficult to obtain a satisfactory skeletonization algorithm satisfying the aforementioned properties. Besides, the major problem of skeleton distortion and junction distortion at crossing or irregular zones has not been thoroughly addressed. For these reasons, the skeleton-based methods for junction detection need to incorporate high-level processing steps to correct the location of junction points [Liu et al., 1999, Hilaire and Tombre, 2006]. Such methods work by extracting the skeletons of input image, following a skeleton segmentation and pivot point extraction (e.g., high curvature points). Next, the obtained skeleton segments and pivot points are refined by incorporating some heuristic merging rules. Some specific post-processing steps are applied to handle particular junction areas. Finally, the junction points are detected as the intersections of the skeleton segments, in combination with the retaining pivot points. Table 1.7 briefly describes such methods where the first three criteria have the same meaning as those in the Table 1.2 (page 34), and the two last ones indicate the ways of computing the thresholds for junction merging and junction treatment.

[Nagasamy and Langrana, 1990] proposed a method dedicated to preprocessing and vectorizing engineering drawings. At first, some basic steps such as noise removal and hole

Table 1.7: Skeleton-based methods for junction detection

Features	Nagasamy and Langrana, 1990	Janssen and Vossepoel, 1997	Liu et al., 1999	Hilaire and Tombre, 2001, 2006
Scale invariant	No	No	No	Yes
Multi-detection	No	No	Yes	Yes
Junction characterization	No	No	Partial	Yes
Threshold for junction merging	Fixed	Fixed	Fixed	Fixed
Junction treatment rule	Spatial continuity	Maximal morphology	Criterion A	Topological correction

filling are applied using morphological filters. Next, thickness layer separation is employed to separate the thick objects (e.g., filled areas, arrows, blobs) from the thin line structures from which the skeleton is extracted. At a crossing zone, the skeleton segments are merged to preserve the spatial continuity of the relevant lines. As illustrated in Figure 1.13, the step of merging the skeleton segments is accomplished by first identifying the incident lines at a given junction and then pairing the lines based on the continuity criteria such as the slope and local curvature at the junction.

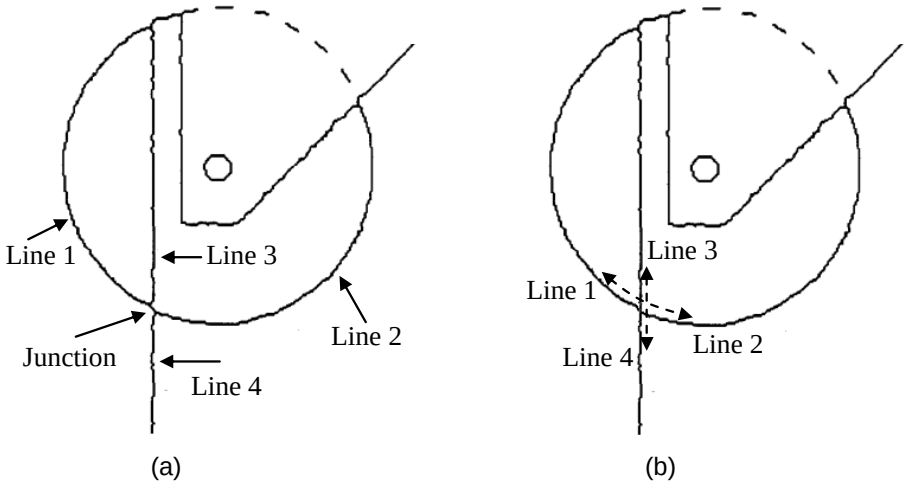


Figure 1.13: (a) The skeleton of an input image; (b) the lines 1 and 2 are grouped (similar to the lines 3 and 4) based on their similar spatial continuity (dash line); (reproduced from [Nagasamy and Langrana, 1990]).

In the next stage, each skeleton line is vectorized by a two-step procedure. The first step detects the dominant points from the skeleton line as illustrated in Figure 1.14 (a-b), and the second step tries to fit the curve segments bounded between two successive dominant points to the arc primitives, if any (Figure 1.14 (c)). The rest of the skeleton line, which can not be fitted by some arc primitives, is fitted to one or several straight line primitives. Part of the dominant points retained after the vectorization process can be treated as 2-junction

points (e.g., like L-junctions) as shown in Figure 1.14 (c). Since the distorted skeleton parts appearing across the junctions are still included in the skeleton lines, the fitting-based vectorization process is affected. Besides, the location of the obtained junctions is displaced since no correction of junction location is performed.

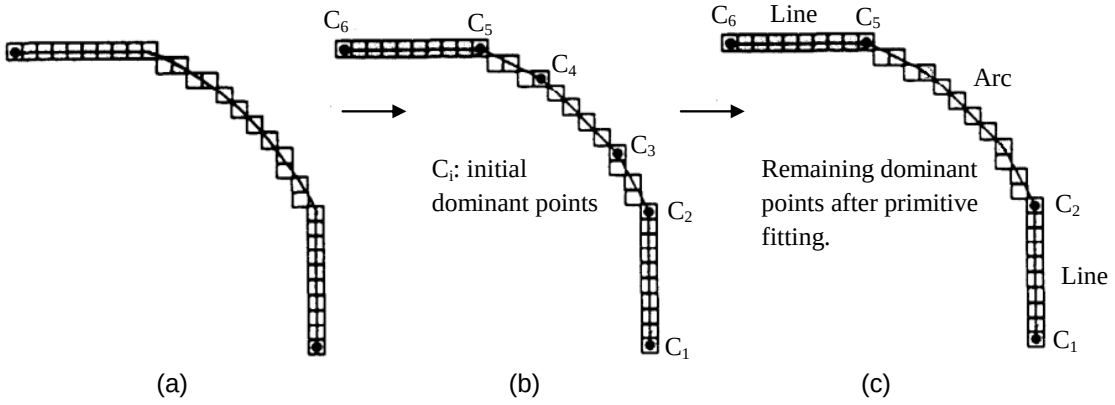


Figure 1.14: (a) The skeleton of an input image; (b) the initial dominant points; (c) the remaining dominant points after primitive fitting; (reproduced from [Nagasamy and Langrana, 1990]).

[Janssen and Vossepoel, 1997] presented a coarse-to-fine approach for obtaining the vectorization and anchor points of input images. First, the skeleton of an input image is coarsely vectorized using a polygon approximation technique [Douglas and Peucker, 1973] with a large threshold t_{dp_coarse}. This step will result in a set of dominant points including the endpoints, corners, and junctions. Next, these dominant points are refined by moving them to "correct" location using the *maximal threshold morphology*. This process of refining the location of a dominant point is illustrated in Figure 1.15, where (a) is an original image, and (b) is its coarse vectorization results including two line segments (black thin lines) and an initial pivot point (red dot). The process of refining the dominant point is summarized as follows:

- Construct a structuring element corresponding to the current location of the dominant point and the two line segments within a local window. The structuring element contains three values $\{1, 0, -1\}$ corresponding to the black pixels, white pixels, and gray pixels as shown in Figure 1.15(c). This structuring element is constructed in a particular way to reflect the approximate shape of the input image given a current skeleton vector.

- Search for a black point within the local window which corresponds to the maximal threshold morphology. To be specific, the structuring element is convolved with the original image and the maximal value is set as the threshold for selecting the output points. In Figure 1.15(d), there is only one point whose value is equal to the maximal threshold.

- Update the obtained point as the new dominant point and repeat the two steps above until the dominant point converges to a fixed location. That is, the distance of the

dominant points found during two successive iterations is less than a fixed threshold (Figure 1.15(f)).

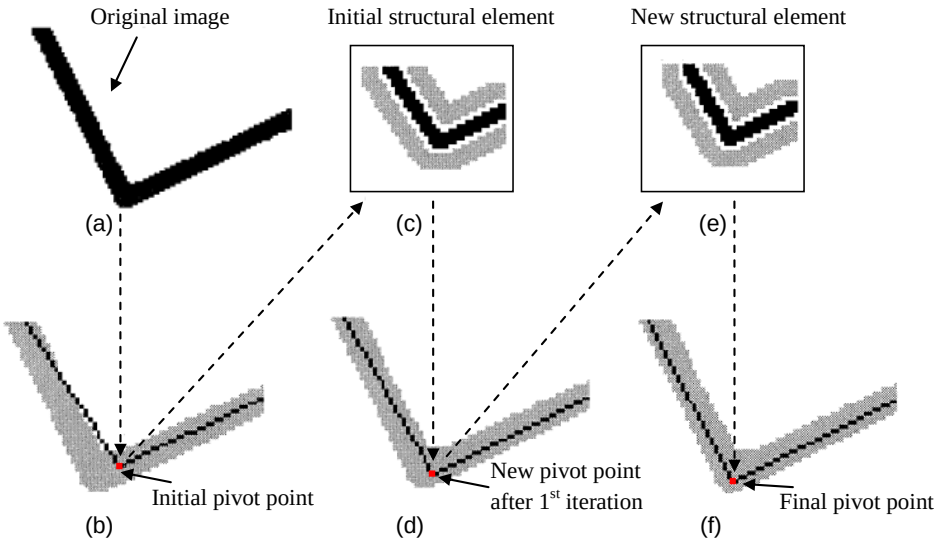


Figure 1.15: The process of refining the location of a dominant point; (reproduced from [Janssen and Vossepoel, 1997]).

After that, the corrected dominant points so-called anchor points are used to correct the subsequent step of refining the vectorization results. This is accomplished by re-performing the polygonal approximation step of the skeleton with a sufficiently small threshold t_{dp}, following a post-processing step of removing false dominant points. The post-processing step works based on several heuristic rules:

- A dominant point is removed if the distance to the closest anchor point is less than a fixed threshold;
- Two T-junctions are merged to form an X-crossing if their distance is less than a fixed threshold;
- A link dominant point is discarded if the bend angle is small enough.

Since these rules are mainly relying on several fixed thresholds, the obtained results are sensitive to the parameter setting. Figure 1.16 gives an example of the results obtained by applying these rules. From Figure 1.16, it may be noted that many false anchor points were detected.

[Liu et al., 1999] introduced a new set of feature points, S_n, computed based on the crossing number $N_c(P)$ for a given binary-value pixel P as follows:

$$S_n = S_e \cup (S_b \cap S_3) \tag{1.15}$$

$$S_3 = \{P | (N_c(P) \geq 3) \; or \; (N_b(P) \geq 4)\} \tag{1.16}$$

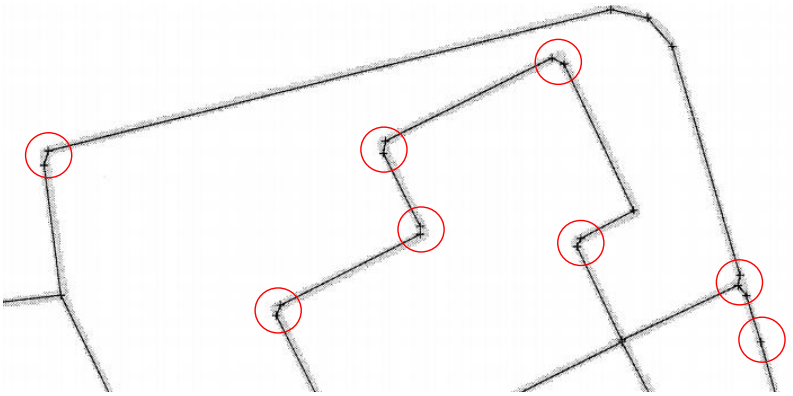


Figure 1.16: Fraction of the obtained vectorization where the false anchor points are marked with the circles (reproduced from [Janssen and Vossepoel, 1997]).

$$N_c(P) = \frac{1}{2}\sum_{0}^{8} |P_{i+1} - P_i| \tag{1.17}$$

where S_e is the set of endpoints, S_b is the set of black points, S_3 is the set of crossing points, and $N_b(P)$ is the number of the black adjacent points of a black point P.

The set S_n, in conjunction with the dominant points obtained from a polygonal approximation step, are represented by a geometrical graph. This graph is then used in accordance with a criterion, called *Criterion A*, to correct spurious junction points (i.e., fork points in the notation of their paper). The basic idea of the *Criterion A* is that two junctions P_1 and P_2 are merged if there exists a point P^* belonging to P_1P_2 such that for every branch B originating from P_i $(i = \{1, 2\})$ except the branch P_1P_2:

- These is a straight line segment P^*P_b fully included in the black region of the image and the length of P^*P_b is sufficiently long; *or*
- These is a straight line segment P^*P_b fully included in the black region of the image and P_b is a termination point of the branch B.

In the experiment, the authors suggested choosing P_b as a point in the branch B. Figure 1.17 (a) gives an example where the *Criterion A* is successfully applied to merge two junction points. However, this criterion is less effective when dealing with different thickness and degraded objects. Figure 1.17 (b) illustrates such a situation where two junction points P_1 and P_2 are merged even that their location is far from each other.

The work presented in [Hilaire and Tombre, 2001] detects junctions based on topological correction of vectorization results. In their work, each skeleton branch is segmented into line and arc segments using a sampling method. Considering the segmentation of line segments, for instance, the sampling algorithm works by iteratively selecting two random points on the branch and extending the candidate line passing these two points as far as possible. This process is repeated for a number of times to find the best candidate line.

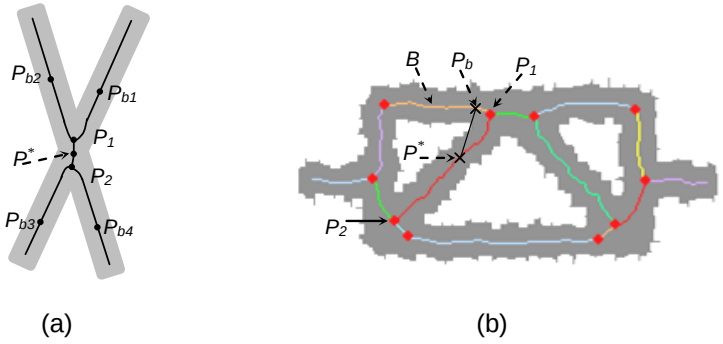


Figure 1.17: Illustration of the *Criterion A*: (a) correct fusion of two junction points P_1 and P_2; (b) wrong fusion of two junction points P_1 and P_2.

A similar process is applied to obtain the arc segments, but three random points are chosen. Each skeleton segment is then classified into either short or long primitives by simply comparing its length with the local line-thickness. Each long primitive p is then associated with an uncertainty domain $\Delta(p)$ defined as the region extended by two curves at a distant of ϵ on both sides from the that primitive (Figure 1.18).

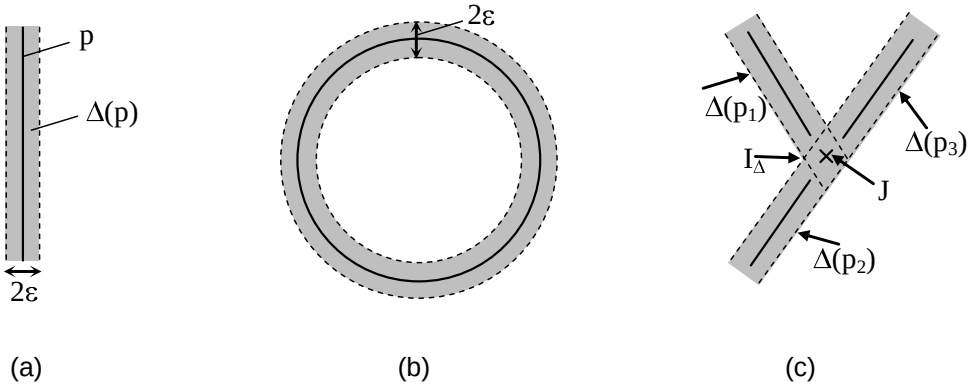


Figure 1.18: (a, b) The uncertainty domains (gray zones) defined for a line and a circle primitive; (c) illustration of the junction optimization process; (reproduced from [Hilaire and Tombre, 2001]).

Next, the long primitives are clustered into different groups by calculating the intersection zones from the uncertainty domains. Finally, each junction point J is reconstructed from the primitives in each group by minimizing the weighted distance error. Particularly, the optimized junction J is computed using the following formula:

$$J = \arg\min_{J \in I_\Delta} \sum_{i=1}^{n} w_i * d^2(p_i, J) \tag{1.18}$$

where n is the number of primitives in the considering group, $d(p_i, J)$ is the Euclidean distance from a point J to the primitive p_i, the weight w_i is set to the length of p_i, and I_Δ is the intersection area of the uncertainty domains $\Delta(p_i)$ with $i \in \{1, \ldots, n\}$ (as illustrated in

Figure 1.18 (c)). This work, as discussed by the authors themselves, has several weaknesses, including being time-consuming, being sensitive to interrupted patterns, and featuring the ambiguous step of merging junction points. In addition, the use of the uncertainty domains is too restricted and threshold-dependent as the junction points need not to lie inside the uncertainty domains.

This work has been further improved in [Hilaire and Tombre, 2006] where the main improvement lies in the step of junction and skeleton optimization. The improved work employs the same technique for skeleton segmentation as before. Next, a new criterion, which is quite similar in spirit to the *Criterion A*, is defined for merging two long primitives. This criteria is illustrated in Figure 1.19. To be more specific, a long segment P, which has two endpoints L_P and R_P, is said to join another long segment Q having two endpoints L_Q and R_Q by L_P if and if there are two discrete primitives X and Y such that the following conditions meet:

- X and Y are fully included in the black region of original image,
- $P \subseteq X$, $Q \subseteq Y$, and $X \cap Y \neq \emptyset$, and
- $\exists K \in X \cap Y$ such that $[R_P K] \cap [R_P L_P] = [R_P L_P]$, where $[R_P K] \in X$ is a discrete primitive having two endpoints R_P and K, and the same for $[R_P L_P]$.

Figure 1.19 illustrates two examples of possible joining primitives where P joins Q by L_P (Figure 1.19(a)), and P joins Q by L_p and Q joins P by L_Q (Figure 1.19 (b)).

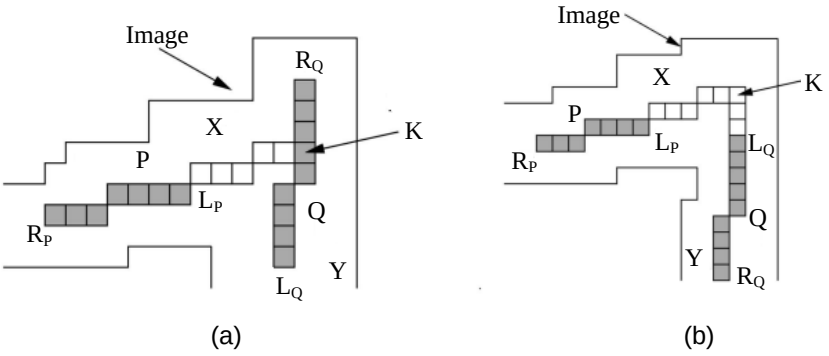


Figure 1.19: Illustration of primitive grouping: (a) P joins Q by L_P; (b) P joins Q by L_p Q joins P by L_Q; (reproduced from [Hilaire and Tombre, 2006]).

Next, the criterion is applied to achieve the global optimization of the skeleton and junction location. To accomplish this, a connectivity graph is constructed where the nodes are the primitives and an edge connects two adjacent primitives. The junction optimization algorithm traverses all possible paths in this connectivity graph, starting from a long segment and leading to either another long segment or the last segment of a sequence of the short ones. For each path traversed, the criterion above is applied to check whether the first segment could join the final one of the considering path.

Figure 1.20 demonstrates an illustrative example of the junction optimization process. Figure 1.20 (a) is an original image and Figure 1.20 (b) is the results of the skeleton

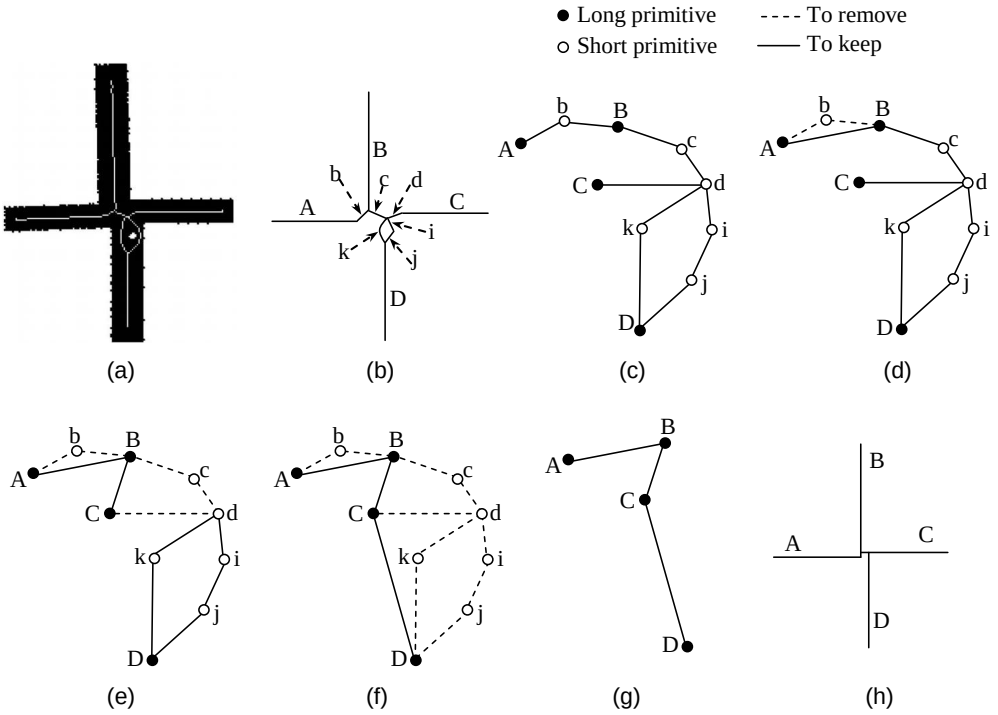


Figure 1.20: Illustration of the process of skeleton and junction optimization: (a) an input image with its skeleton; (b) the results of skeleton segmentation process; (c) the connectivity graph; (d-f) the process of graph simplification; (g) the final graph; (f) the final skeleton and junction points; (reproduced from [Hilaire and Tombre, 2006]).

segmentation step: the capital letters (i.e., A, B, C, D) indicates the long primitives, while the rest represent the short ones. The resulting connectivity graph is shown in Figure 1.20 (c) where the black nodes correspond to the long primitives and the white nodes are the short ones. In Figure 1.20 (d), we obtain the first path composing of the nodes $\{A, b, B\}$, and the second path is found later in Figure 1.20 (e) composing of the nodes $\{B, c, d, C\}$. For the first path, because the node A can join B using the aforementioned joining rule, the node b is then marked for deletion and the two nodes A, B are directly connected. This process is repeated for the other paths as shown in Figure 1.20 (e, f). The final graph is plot in Figure 1.20 (g), which corresponds to the final skeleton and junction structure as illustrated in Figure 1.20 (h). When the optimized algorithm terminates, the nodes marked for deletion with unlinked edges are deleted.

The problem with this approach concerns the ambiguous step of constructing the connectivity graph. For instance, it was not discussed in the paper why the edges (b, c), (c, k), (i, c), and (j, k) are not included in the graph in Figure 1.20 (c). Besides, as mentioned by the authors, multiple merging situations of the primitives might happen, the process of reconstruction of the junctions is thus non-deterministic. Hence, more than a single solution can be obtained depending on the order of primitive merging. Figure 1.21 illustrates such a case where another possible solution is presented for the crossing structure in Figure 1.20.

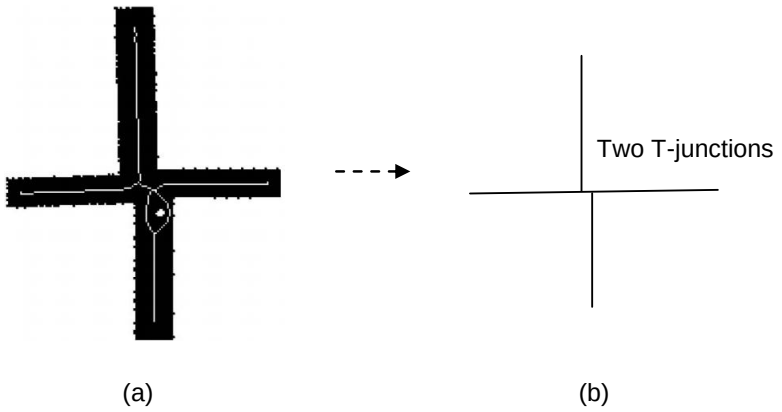


Figure 1.21: Another possible interpretation of the crossing structure in [Hilaire and Tombre, 2006]: (a) an input crossing structure; (b) an interpretation for the crossing in (a) where two T-junctions are constructed.

In summary, one of the most challenges of the skeleton-based methods is to handle the problem of skeleton distortion taken place at the crossing zones. Most of these methods addressed this problem by introducing a junction optimization step that uses some simple and heuristic rules to merge the junction candidates. Such a rule, therefore, is highly relying on the selection of several thresholds to make the decision of merging the junctions. As the chosen of these thresholds is not trivial and mainly found based on empirical trials, these methods can not perform well for different datasets. In practices, due to different levels of degradation of input images, the obtained results could be substantially affected.

1.3.3 Contour-based methods

To avoid the distortion problem resulting from the skeletonization step, some works detect junctions from contour-based vectorization results. Instead of using a traditional skeletonization technique, these works are concerned with the extraction of median lines from contours to avoid the problem of skeleton/junction distortion. These methods are typically composed of four main stages. The first stage detects contours and partitions them to obtain a set of contour segments. The second stage performs contour segment pairing to find the matching contour segments located in two opposite sides of a stroke. In the third stage, the median lines are generated from the matching contour segments. Median line correction may be applied to refine the extracted lines. The last stage is junction localization using the direction and continuity of the median lines and contour boundaries. Table 1.8 presents the representative methods of this approach. The two first criteria of multi-detection and junction characterization have the same meaning as previously used in Table 1.2 (page 34). The last three criteria explain for the rules used in the stages of contour paring, median line extraction/correction, and junction localization respectively.

In [Hori and Tanigawa, 1993], the contours and skeletons of input image are polygonally approximated to obtain two sets of skeleton and contour fragments. Next, each skeleton fragment is associated to the closest contour fragments based on some heuristic criteria.

Table 1.8: Contour-based methods for junction detection

Method	Multi-detection	Junction characterization	Contour pairing rule	Median line correction	Junction localization
Hori and Tanigawa, 1993	No	Partial	Missing	Progressive searching and fitting	Intersection of median lines sharing a same segment
Han and Fan, 1994	No	No	Direction difference, spatial distance, and contour projection overlapping	Median lines of matched contours, gap filling by extrapolating to the junction	Centroid of intersections of median lines or terminations
Fan et al., 1998	Yes	No	Direction difference, minimum spatial distance, and minimum matching cost	Median lines of matched contours, gap filling using adjacent blocks or contour continuity	Intersection of median lines
Ramel et al., 2000	Yes	Yes	Spatial distance, parallelism, and foreground density	Not used	Intersection of quadrilaterals

A line fitting process is then performed as an iterative process composing of two steps. The first step chooses a seed for line fitting as the longest unused skeleton fragment or contour segment. The selected seed is then treated as a path on a graph where the nodes are the skeleton fragments and an edge connects two adjacent skeleton fragments. Such a path will be used to find a candidate line, fitting well to the skeleton segments of the path. The second step extends the path by iteratively inserting, in one direction, a new adjacent node of the current termination node of this path. A new node is inserted to the ongoing path if two following conditions are met (Figure 1.22):

- The average distance between the endpoints of every skeleton fragment of the ongoing path and the candidate fitting line is less than a threshold;
- The fitting candidate line lies inside an enclosed region bounded by the contour fragments.

This searching step terminates when all candidate lines are visited and the longest one is considered as the best line.

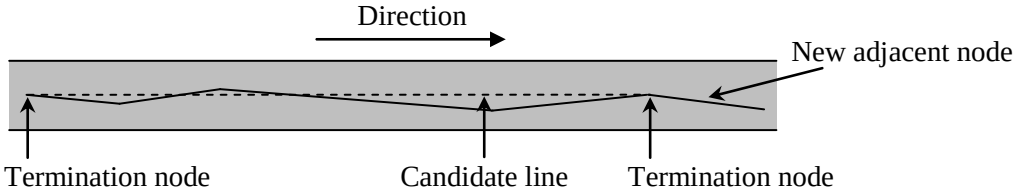


Figure 1.22: Illustration of the process of progressive extension of line fitting; (reproduced from [Hori and Tanigawa, 1993]).

A final step of correcting the junction location is performed as follows with respect to Figure 1.23:

- Find every two lines which share the same skeleton segment. For instance, in Figure 1.23, the first line is composed of two skeleton segments $\{S_1, S_2\}$, and the second one contains two skeleton segments $\{S_3, S_2\}$. These two lines share the same skeleton segment S_2.
- If the two lines share the same skeleton segment in two different directions, as presented in Figure 1.23 (a, b), the intersection of the new lines without the shared skeleton segment is treated as a new junction point. Figure 1.23 (d, e) demonstrate the junction correction results for the cases in Figure 1.23 (a, b).
- If the two lines share the same skeleton segment in the same direction, as presented in Figure 1.23 (c), these two lines are refitted by merging their termination points to produce the result (Figure 1.23 (f)).

This method can handle simple crossing zones corresponding to the T-, L-, and X-junction. For more complicated junction configurations, special post-processing is needed.

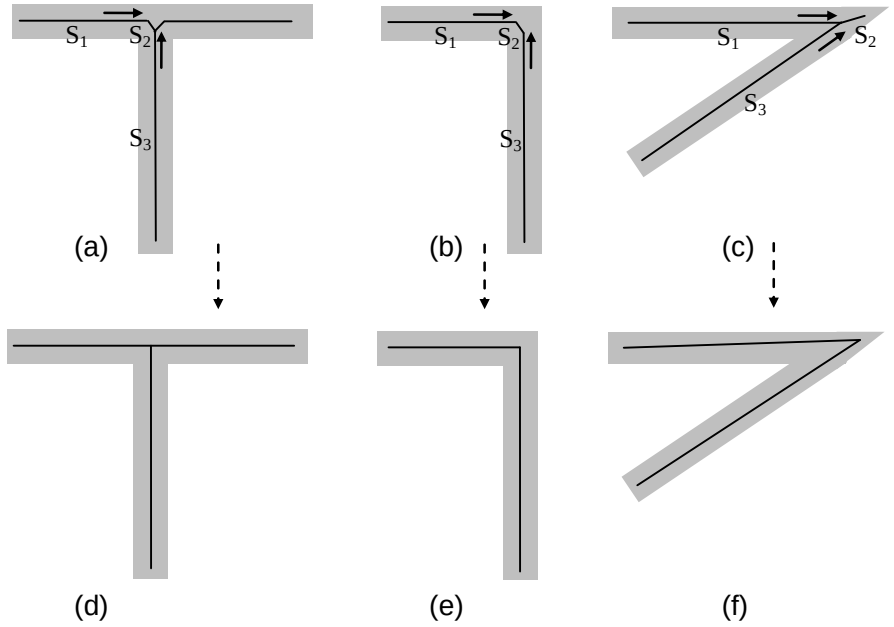


Figure 1.23: Illustration of the process of correcting the junction location; (reproduced from [Hori and Tanigawa, 1993]).

In [Han and Fan, 1994], the contours are first vectorized to obtain a set of contour segments such as straight line segments, circular arcs, and curve segments. The contour segments of the same type are then matched based on the following criteria:

- The distance between two segments is small enough;
- The overlap of the projection of the two segments on either horizontal axis or vertical axis is sufficiently high;
- The direction difference of two straight line segments is close to 0^0 or 180^0, or the difference of the radii of two circular arcs must be small enough.

All these criteria are applied using some fixed thresholds. Next, the median lines generated from the matching contour vectors, are vectorized resulting in a set of skeleton vectors. Since there is often no properly matched contour segments at the crossing zones, no median lines are generated at such location. To remedy this matter, a last process is performed to connect the gaps and generate junction points as illustrated in Figure 1.24. At first, a gap is detected as the place where there is at least two adjacent pairs of matching contour vectors. Next, gap filling and junction generation is applied. Given n adjacent contour vector pairs represented by $\{V_{i_1}, V_{i_2}\}$ $(i = 1, \ldots, n)$, a junction is constructed employing the following procedure:

1. If there is no common contour vector among the n adjacent contour vector pairs, as illustrated Figure 1.24 (a), the followings are the main steps of gap filling:

- Pick randomly a skeleton vector as a reference vector.
- Compute the intersections of the reference vector with the rest.
- A new junction point is computed as the center of the intersections.
- The skeleton lines are extrapolated to the junction point.

2. If there exist two pairs of contour vectors which share the same contour vector (e.g., V_1 in Figure 1.24 (b)), the process of gap filling is performed as follows:

   - Join the two skeleton vectors corresponding to the pair of continue collinear contour vectors.
   - Compute the intersections of the joint skeleton vector with the rest.
   - A new junction point is computed as the center of the intersections.
   - The skeleton lines are extrapolated to the junction point.

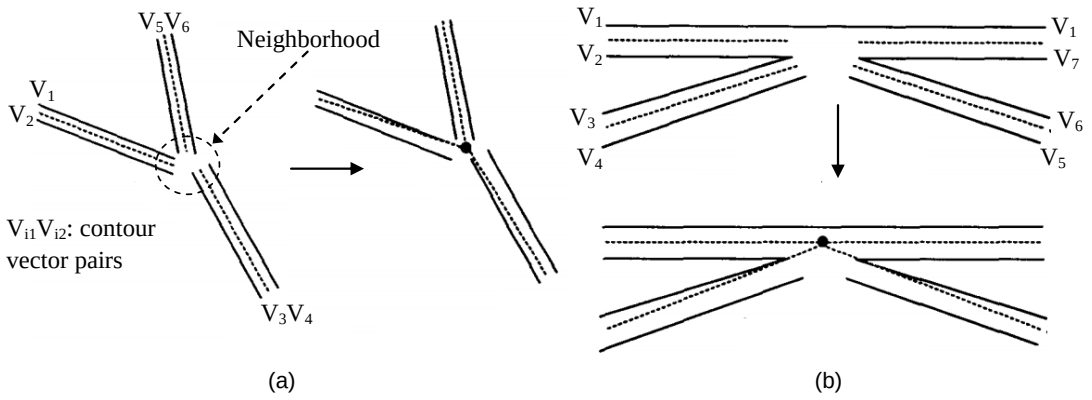


Figure 1.24: Illustration of the process of gap filling and junction generation after contour segment pairing: (a) without common vector and (b) with common vector; (reproduced from [Han and Fan, 1994]).

This work assumes that each crossing zone contains exact one meeting point or junction point. In addition, the meeting point computed in this way might be located outside the crossing zone. Furthermore, the junction location obtained for the case in Figure 1.24 (a) may differ from time to time because the reference skeleton vector is selected randomly without checking at the global point of view.

[Fan et al., 1998] presented a method to deal with the distortion of thinning-based skeletonization using block decomposition and contour vector matching. The proposed method first decomposes a binary input image into blocks and then the contours of each block are extracted and polygonally approximated to obtain a set of contour vectors. A matching process is then performed for pairing the matched contour vectors based on several heuristic criteria. Particularly, a contour vector L_j is matched with another contour vector L_i if the following conditions are met:

- The direction difference between L_i and L_j is within a range of $[90, 270]$, and
- L_j is the line having the shortest distance, $dist(L_i, L_j)$, to L_i, and having the smallest matching cost defined as $dist(L_i, L_j)/|L_i|$, where $|L_i|$ is the length of vector L_i.

Next, the median lines are generated from the matching contour vectors. The median lines are vectorized to obtain a set of skeleton vectors. A final process of gap filling is applied to connect the disjointed skeleton vectors due to the discontinuity of the blocks as illustrated in Figure 1.25. Two different types of gaps are defined in this approach and two different methods are employed for filling the gaps. These methods are illustrated in the following.

- Type 1 gap: A gap of type 1 is formed by the matching of one contour vector C_1 and two disconnected contours C_2 and C_3 as shown in Figure 1.25 (a). For such a case, gap filling is treated by linking two contour vectors corresponding to the matched contours C_1, C_2, and C_3 as illustrated in Figure 1.25 (b).
- Type 2 gap: A gap of type 2 is formed by the matching of two skeleton vectors belonging to different adjacent blocks B_1 and B_2 as shown in Figure 1.25 (c). In this case, gap filling is completed by linking these two skeleton vectors as shown in Figure 1.25 (d).

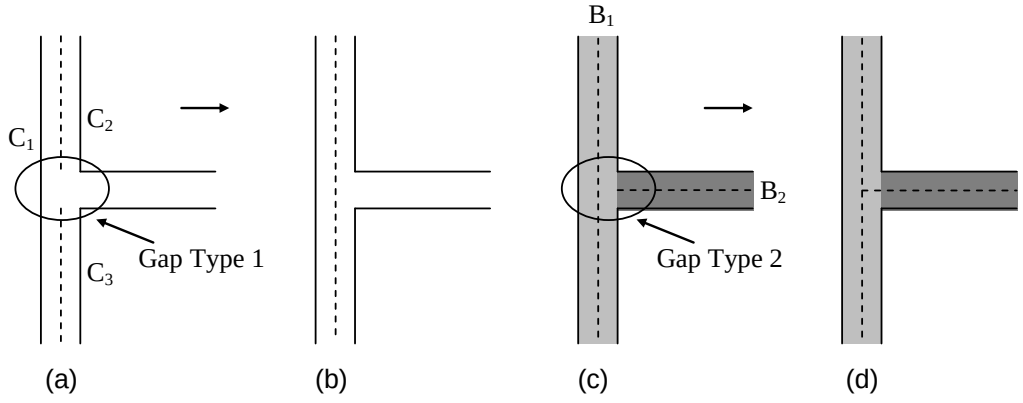


Figure 1.25: Illustration of the process of filling gap and junction generation; (reproduced from [Fan et al., 1998]).

Figure 1.26 shows an example of the obtained skeletal vectors and junction points. It can be seen from this example that the proposed method produces a noticeable displacement of the skeleton and junctions (marked by the circles). In addition, the proposed method works by incorporating various heuristic rules which would be failed to handle complicated crossing zones.

The work in [Ramel et al., 2000] formulates the junction detection problem as an identification of relations (i.e., intersection, succession, parallelism) between contour primitives. The contours of input image are first extracted and polygonally approximated to obtains a set of vectors. A quadrilateral is defined as a pair of two matched vectors on two opposite sides of a line segment. To construct the quadrilaterals, the matching algorithm picks up

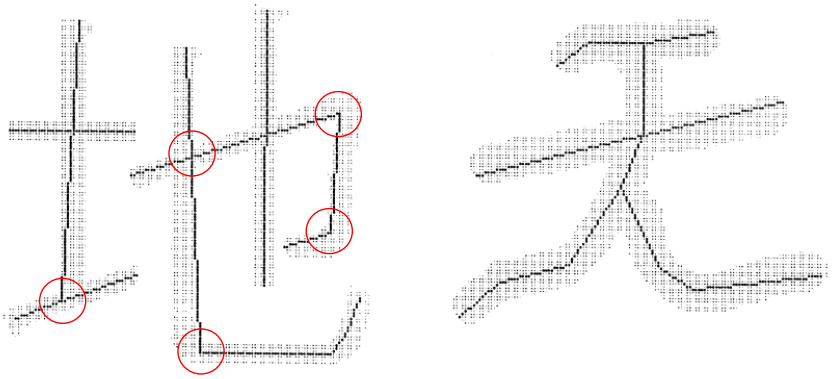


Figure 1.26: The obtained skeletal vectors and junctions in the work of [Fan et al., 1998]. The displacement of skeletal vectors and junctions are marked by the circles; (reproduced from [Fan et al., 1998]).

the largest vector V_1 from the list of non-matched vectors, and then looks for the two closest opposite vectors, V_2 and V_3, based on the distances (e.g., d_1 and d_2 on Figure 1.27) between the terminations of these vectors. A heuristic criterion, using parallelism and foreground density, is then derived to match V_1 versus V_2 and V_3. To illustrate how this criterion is applied, Figure 1.27 gives the match of V_1 and V_2. In this case, the vector V_1 could be decomposed into two sub-vectors by projecting one of two terminations of V_2 into V_1 (e.g., P_1'). The less matched sub-vector (e.g., connecting P_1 and P_1') is then updated as non-matched vector. This procedure is continued until all vectors are processed.

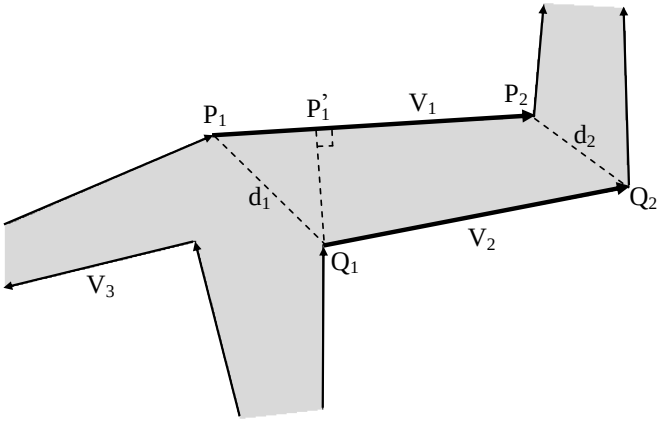


Figure 1.27: Illustration of the contour matching process (see explanation in the text); (reprinted from [Ramel et al., 2000]).

Finally, a structural graph is constructed to represent the structural relationships between the quadrilaterals such as parallel relation (P), T-, X-, and L-junction. Figure 1.28 illustrates such a graph constructed for part of an input image. The main weakness of this work is the loss of connectivity among the adjacent quadrilaterals. It is also sensitive to

parasite and degenerate quadrilaterals. In addition, the structural relation defined between the quadrilaterals is restricted to few simple junction types.

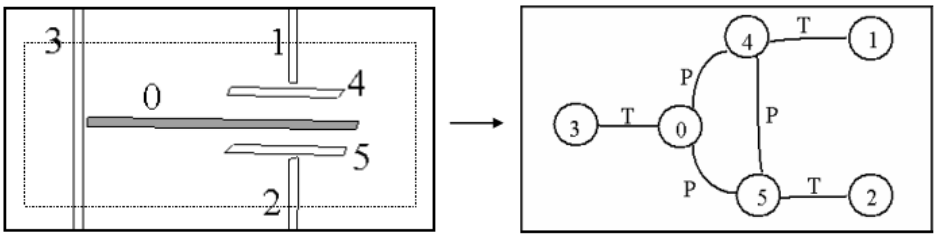


Figure 1.28: Construction of the structural graph (reprinted from [Ramel et al., 2000]).

In short, the contour-based methods avoid the major problem of skeleton distortion, but they give a rise of several other drawbacks. In the first place, image contours must be first extracted and polygonally approximated to obtain a set of contour vectors. The resulting contour vectors are thus highly subjected to contour noise and distortion. On the other hand, while contour vector matching is an obligated stage of these methods, it is well-known that there has been no robust enough technique to accomplish this task. Although existing techniques often employ various heuristic rules to perform contour matching and junction construction, the obtained results are still far from perfection, especially when encountering a crossing zone. Finally, the loss of connectivity of the median lines at the crossing zones is an inherent drawback of these methods.

1.3.4 Tracking-based methods

Apart from the before-mentioned approaches for junction detection, tracking-based methods have been proposed. These methods, known as direct vectorization, are involved in vectorizing line-drawings without using any segmentation steps(i.e., neither skeleton segmentation nor contour segmentation). Therefore, any error-prone introduced by a segmentation process is avoided. Typically, these methods are composed of three main steps: seed initializing, line tracking, and junction handling. Seed initializing is a step to detect a starting element for the subsequent step of line tracking. These elements could correspond to pixels (e.g., Sparse Pixel Vectorization [Dori and Liu, 1999]), oriented bounding boxes [Song et al., 2002], circular regions (e.g., Maximal Inscribing Circle [Chiang et al., 1998]). The second step of line tracking is an iterative process to fit the seed element to the foreground region until some specific terminations are satisfied. Typically, the tracking process terminates where it reaches a junction zone or an entire object is tracked. Then, the last step of junction handling is triggered to determine the next tracking branch or to remove the tracked objects.

A summary of the key works belonging to this approach is presented on Table 1.9 where the two first criteria of multi-detection and junction characterization have the same meaning as previously defined in Table 1.2 (page 34). The last three criteria outline the main ideas of the three steps of a tracking-based algorithm. The details of these methods are presented in the following.

Table 1.9: Tracking-based methods for junction detection

Method	Multi-detection	Junction characterization	Seed initializing	Line tracking	Junction handling
Nieuwenhuizen et al., 1994	No	No	User-interaction	Two opposite contour followers	Circle following and branch construction
Dori and Liu, 1999	Yes	No	Reliable starting median point detection	Tracking the midpoints of successive vertical/horizontal runs	Line merging based on width similarity, collinear, and proximity
Chiang et al, 1998	Yes	No	Maximal inscribing circle fitting	Circular region based tracking	Line labeling and connecting using Bezier curve
Song et al., 2002	Yes	No	Oriented bounding box fitting	Bar segment based tracking	Deletion of the tracked bars using contour directions

In [Van Nieuwenhuizen and Bronsvoort, 1994], seeds are initialized manually. The user points out the line and the direction to be tracked. Next, the indicated line is tracked using two contour followers, called the left and right ones, on two opposite sides of the line. This tracking process terminates until one of the following conditions is met:

- The last point found by the left contour follower equals to the previous point found by the right contour follower and vice versa (i.e., the end of the line is reached).
- The current tracking position returns to the starting position (i.e., the line is closed).
- The distance between two successive median points is smaller than a threshold (i.e., a crossing zone).

While arriving at a junction location, the tracking algorithm detects the relevant branches by constructing a series of n circles centralized at the current interrupted median point with increasing radii as illustrated in Figure 1.29 (a). The radii are determined such that the corresponding circles pass through the previous tracked median points. The intersections between these circles and the contours of the junction area are called *circle-contour* points as shown in Figure 1.29 (b).

Next, starting from a circle-contour point of the current branch B_1 and the i^{th} circle with $i = 2, \ldots, n-1$ (e.g., P_1 in Figure 1.29 (b)), a clockwise contour follower is applied to detect the other circle-contour points. This contour follower terminates when the last circle-contour point P_2 is reached. The detected circle-contour points are then paired to find the pairs of opposite circle-contour points used to identify the relevant branches of the current junction area (e.g., the branches B_2, B_3, and B_4 in Figure 1.29 (b)). Besides, for each pair of two opposite circle-contour points, a new median point is generated. The detected branches are finally provided to the user for the selection of next tracking branch.

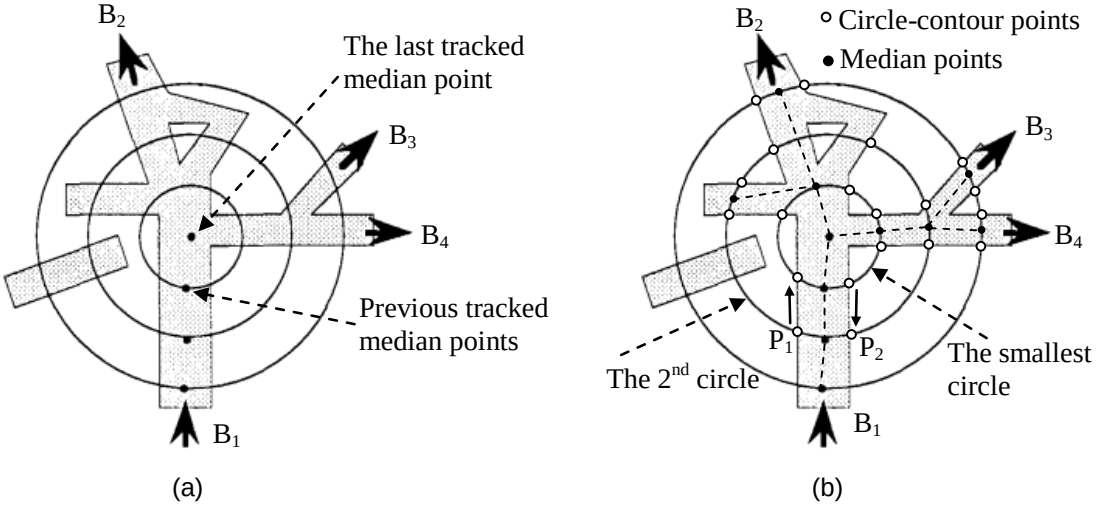


Figure 1.29: The branch construction process: (a) circle following; (b) branch construction; (reproduced from [Van Nieuwenhuizen and Bronsvoort, 1994]).

This method is robust to handle the varying line width and irregular border as well.

The main weakness is that it requires the intervention of the user in both seed initializing and junction handling. Besides, the process of branch construction is sensitive to the fixed setting of the parameter n in the tracking algorithm at the junction location.

[Dori and Liu, 1999] proposed a pixel-tracking-based vectorization system called Sparse Pixel Vectorization (SPV). In the first step, a reliable starting median point is detected as the midpoint of either the vertical run or horizontal run originated from the first encountered black pixel. Next, tracking process is proceeded in either horizontal or vertical direction depending on the corresponding run length. During the tracking, the widths of the tracked lines are estimated and compared to a threshold. If the width run conflicts the width preservation, as shown in Figure 1.30 (a), a junction recovery process is triggered. The junction recovery process works as an iterative search along the junction area, composed of three steps:

- Revert back to the last medial point (Figure 1.30 (b));
- Update the tracking step length with half of the length in the previous step (Figure 1.30 (b, c));
- Explore a new tracking position.

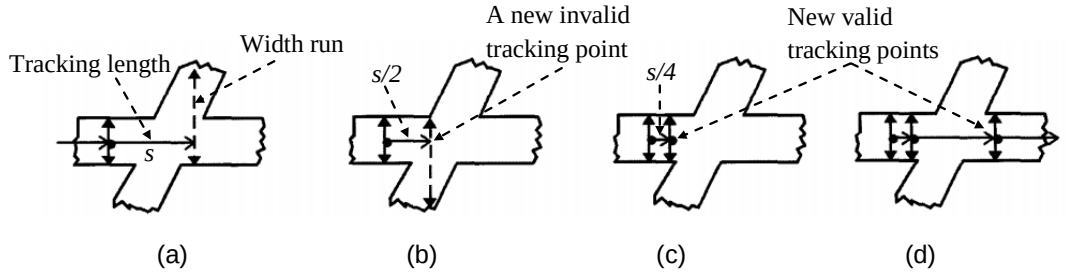


Figure 1.30: Illustration of the junction recovery process: (a) conflict of the width run and the width preservation; (b) a new invalid tracking point; (c) a new valid tracking point on the current branch; (d) a new valid tracking point on a different branch; (reproduced from [Dori and Liu, 1999]).

The junction recovery process leads to two possible cases:

- A new valid tracking point is found, as shown in Figure 1.30 (d), and the tracking process is proceeded on the new line.
- No valid points are found (i.e., the tracking step length becomes zero). In this case, the tracking step continues with a new seed point.

Once the tracking step terminates, the median lines are polygonally approximated following some post-processing steps to link the collinear line segments of the same line width, correct the defects at the junction location, and refine the endpoints. As argued by [Song et al., 2002], the weakness of this junction recovery process is that it fails to pass the junction areas where the length of the intersection zones is greater than the tracking

step length. In addition, the loss of connectivity, when encountering the arcs, is another weakness of the proposed method.

The authors in [Chiang et al., 1998] proposed a region-based vectorization system called *maximal inscribing circle* (MIC). A MIC is defined as a maximal-radius circle fitting inside a line segment in the sense that it has at least two contact points with the border of this line segment (Figure 1.31). The line segments are then tracked using the MIC(s) following a labeling algorithm to analyze different detected lines, identify junction zones, and construct spatial relations among them. After connecting the different lines through common junction zones, the connected lines are vectorized to obtain vector-based representation. Two short line vectors can be merged using the Bezier curve, if necessary. The final junction points are calculated as the intersections of the vectorized lines. Experimental results showed interesting performance of the proposed method, compared to the traditional approaches. In contrast to many existing methods, the proposed system requires no post-processing steps to prune the short segments and unify the spurious junctions. However, the system is highly dependent on the step of line unifying at the common junction areas. In addition, false merging of the two short segments can be happened due to the selection of fixed thresholds.

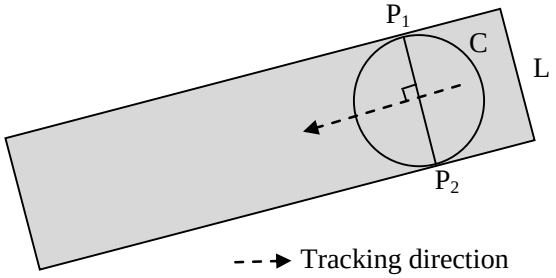


Figure 1.31: Illustration of MIC tracking process; (reproduced from [Chiang et al., 1998]).

In [Song et al., 2002], a different approach based on object-oriented vectorization is proposed. In this approach, the primitives are tracked in a specific order starting from the simple ones (e.g., bar lines) to the complex ones (e.g., arcs, circles, text). This is referred to the progressive simplification process whose main idea is to progressively recognize the objects and then to remove them for reducing their disturbance. The seed initialization step, as illustrated in Figure 1.32 (a), starts from a foreground pixel A and detects the regular runs in both horizontal and vertical directions. This terminates where the number of successive regular runs satisfies a pre-defined threshold, resulting in a successful seed segment detection. The detected seed is then used to track the entire bar in both opposite directions. Tracking process is continued until the length (L) of the last perpendicular run violates the thickness (W) of the seed segment. To be more specific, the termination conditions are as follows: $L < 0.5 \times W$ or $L > 2 \times W$. Once the bar is completely tracked, it will be removed to eliminate the interference with the rest of foreground objects. However, the deletion operator at a junction area is not trivial. At such a junction zone, as illustrated in Figure 1.32 (b), the authors suggested detecting the contours of the branches involved in this zone and then analyzing their trends to calculate the part of the junction zone to be preserved. Followings are the main steps of deleting a recognized bar at a junction area:

- If there are at least two branches on both sides of the tracked bar, use the trends of the incident contours to determine the area to be preserved (e.g., the center area in Figure 1.32 (b)).
- If all the incident branches are on one side of the tracked bar, simply remove half of the tracked bar which corresponds to the side containing no branches (e.g., the top area in Figure 1.32 (b)).

The main challenge with this process is its sensitivity to contour noise. In addition, when two intersection zones are too close to each other, the estimation of contour trends is not reliable. The experimental results, applied to several vectorization test datasets, showed the robustness of their system in the context of vectorization.

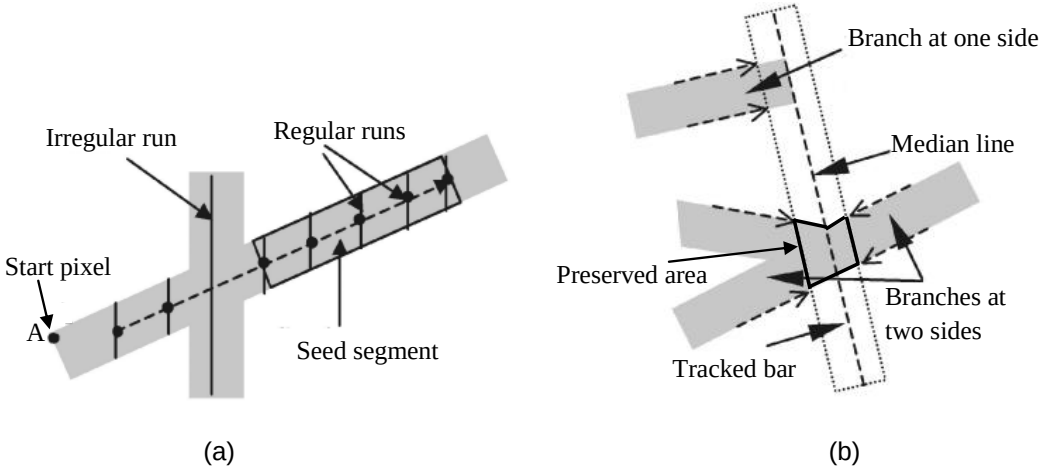


Figure 1.32: Bar tracking process (a) and intersection preservation deletion (b); (reproduced from [Song et al., 2002]).

In summary, one common major problem of the tracking-based methods is the tracking initialization. Depending on the detected seed segment, the subsequent step of line tracking can be impacted since the stop tracking conditions are mainly relied on the seed's information such as its location, direction, and size. Another problem related to the tracking-based methods is the reading order. A tracking method is a recursive process, wherein next tracking steps are initiated from the previous ones. Therefore, a small changes in the location and direction of the previously tracked segment could lead to different results of the newly detected segments. Consequently, the tracked median lines and the obtained junctions could be displaced from time to time. Finally, the junction detection and correction process is triggered by the tracking process. Any mistakes in the tracking progress could lead to the miss detection of the junction point.

1.3.5 Conclusions of junction detection methods in line-drawings

In conclusion of the methods for junction detection in line-drawing images, we can draw here several noticeable remarks. At first, since the junctions are detected as the intersections of the median lines, different methods have been proposed to deal with the problem

of median line extraction and correction. One line of research has been relied on the skeletonization approach. These methods directly extract median lines using some skeletonization techniques and then corrects the problem of junction distortion by incorporating high-level processing steps. To accomplish this, most of these junction correction steps are heuristics-based rules, which require specific parameter tuning to merge the false detections. Besides, scale invariant is partially achieved and junction characterization is rarely discussed. Other methods, which try to avoid the problems resulting from skeletonization step, are based on contour matching and element tracking. However, the contour-based methods have two major problems: contour matching and gap handling. It may be noted that there is no robust technique to handle the one-to-many and many-to-many matches in the process of contour segment pairing. In addition, the loss of connectivity is an inherent drawback of these methods. For the last approach of element tracking, seed initialization and tracking order are two main issues. The junction detection and correction process is highly dependent on the line tracking and detection. The last noticeable point is that no real evaluations of these all methods in terms of junction detection are performed. In fact, all these approaches have evaluated under the context of vectorization contests of which the test images were not severely degraded and the evaluation metrics are not dedicated to keypoint performance evaluation. The common metrics for evaluating a vectorization system are Combined Detection Index (CDI) [Wenyin and Dori, 1997] and Editing Cost Index (ECI) [Phillips and Chhabra, 1999]. The CDI metric computes the positive and negative detection rates at both pixel and vector levels, whereas the ECI metric is computed at vector level measuring the cost of correcting a mistake (e.g., miss detection, false alarm, one-to-many, and many-to-one) of a detection system. Since both the CDI and ECI metrics are calculated at the vectorization results and the detected junctions take very small part of the detection results, the real performance of the detected junctions is thus negligible and not correctly evaluated. More precisely, evaluation of the junction detection process has been embedded in a vectorization evaluation protocol.

1.4 Open discussion

Following the conclusions made at the end of each previous section, it is clear that the techniques in CV for junction detection can not be directly employed to the same problem in line-drawing images. As the line-drawing images are mainly composed of binary features, the matter of extracting the edge map is thus a trivial task. However, junctions in line-drawings are considered as the meeting points of the median lines, the CV techniques based on edge grouping are likely to produce much of false detection rate. On the other hand, the parametric-based techniques need to construct the junction models which are also relying on the behavior of the edge ridges. These techniques therefore can not deal with the correct detection of junction points in line-drawing images. To be more specific, the detected points using these CV methods would be the corner points rather than the junctions points when applying to line-drawing images. At last, due to the lack of standard datasets for junction detectors, performance evaluation of the CV methods was rarely concerned. Although a semantic benchmark was provided by [Maire et al., 2008], few methods for junction detection were evaluated on this dataset.

In line-drawing images, the problem of junction detection has been mainly embedded in a vectorization system. The obtained results are therefore subjected to the error-prone caused by the raster-to-vector conversion process. Such a vectorization system has been known to be sensitive to setting parameters, and present difficulties where heterogeneous primitives (e.g., straight lines, arcs, curves, and circles) appear within a single document. Knowledge about the document content (objects, entities, document layout) must be included and derived using an appropriate strategy to make the system more robust at the cost of adaptability. Besides, few of them (e.g., [Hilaire and Tombre, 2001] and [Hilaire and Tombre, 2006]) are able to handle the multiple junction detection at a given crossing zone, and junction characterization was not thoroughly discussed. Furthermore, the real performance of these junction detection approaches have not been thoroughly evaluated due to the lack of extensive and comparative experiments. Indeed, most of these methods were evaluated under the context of vectorization. Other methods just give few illustrative examples of the detected junctions. Therefore, the real performance of these methods for junction detection is still an open question.

Apart from the before-mentioned challenges, we are also aware of favourable features of the existing methods. First, it is desired to use the median lines as a good presentation of the line-like primitives but not the crossing zones. Therefore, if the crossing zones can be segmented out, an off-the-shelf skeleton-based method would be useful to represent the rest of line-drawings [di Baja, 1994]. Second, once the major part of line-drawings are represented by the median lines, the 2-junctions (e.g., L-, V-, M-junctions) can be detected by partitioning the median curves into straight line segments. To do so, a polygonal approximation method can be applied. However, such a method shall produce many false alarms for perfect smooth primitives taking a circle for example. A good alternative solution is the use of a high curvature point detection. For this concern, we can reuse many robust methods for dominant point detection in the literature [Teh and Chin, 1989]. Finally, once the crossing zones have been segmented out, the incident median lines of a given crossing zone are unlikely to be impacted by the problem of junction distortion or skeleton distortion. The n-junctions ($n > 2$) contained in each crossing zone could be reconstructed using an advanced junction optimization algorithm, and interesting approaches on this problem have investigated in the literature by [Hilaire and Tombre, 2006, Maire et al., 2008].

In this dissertation, we attempt to create such an approach for junction detection in line-drawings that exploits the advantages of several existing techniques. The proposed approach has the following major features: junction distortion avoidance, accurate junction detection, multiple junction detection, efficiency and robustness. We will justify thoroughly all these features in the next chapter.

Chapter 2

Accurate junction detection and characterization in line-drawings

---

Following the conclusions made in the previous chapter, it is highly agreed that junction features are crucial for many applications. This chapter presents a novel approach for junction detection and characterization in line-drawing images. The proposed approach has been deeply evaluated through extensive experiments in comparison with two other baseline methods. Experimental results showed that the proposed approach is robust to common geometry transformations and can resist a satisfactory level of noise/degradation. Furthermore, it works very efficiently in terms of time complexity and requires no prior knowledge of the document content. At last, the proposed method is independent on any vectorization systems.

2.1 Introduction

In line-drawing images, junction features are of crucial importance to support different applications such as symbol localization and spotting, vectorization, storing and retrieval of engineering documents. Despite a number of works for junction detection have been proposed in the literature, each of these approaches has some specific limitations as thoroughly mentioned earlier. As we are interested in detecting and characterizing the junctions for the analysis of engineering document images, this chapter attempts to bring a novel, complete, and accurate approach for that purpose.

In the proposed approach, we directly formulate the problem of junction detection as searching for optimal meeting points of line-like primitives from input images. However, as

it is impossible to obtain ideal line primitives (i.e., 1-pixel-thick lines) from a digitization process, the intersection areas of the line primitives can not converge or contract to one pixel as expected. Therefore, to achieve exact junction localization, the line primitives must be represented in suitable forms that facilitate the step of finding their intersections. Apart from the major drawback of junction distortion, median axis lines have been known to be very good representations of such line primitives. At first glance, it seems that our approach would directly encounter the well-known problem of junction distortion. However, it is important to note that if we can identify and remove all distorted zones, the remaining line segments could be well represented as the mean of the median lines with little (or even without) disturbance concerning the issue of junction distortion. We therefore develop our approach based on this idea.

It is noted that similar idea has been explored in the literature [Hilaire and Tombre, 2001, Hilaire and Tombre, 2006]. However, the approach taken in these works is quite different from ours. Particularly, these two methods partition the skeleton into reliable and unreliable line segments relying solely on a single line-thickness criterion. If the length of one segment is less than the line thickness, it is considered to be a short segment and *vice versa*. The short segments are treated as unreliable segments and thus removed. Using such a threshold is too vulnerable and would lead to many mis-classifications among unreliable/reliable segments.

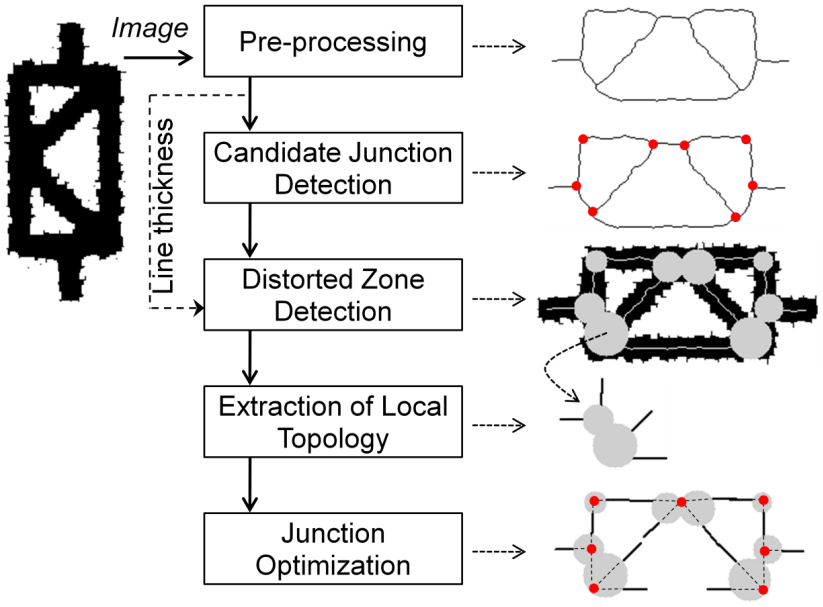


Figure 2.1: Overview of the proposed approach.

In our approach, a distorted zone is identified by addressing two probing questions: where such a zone is likely to appear and how large this distortion zone would be. Naturally, the distorted zones occur at crossing locations and these zones would be restricted to small areas fitting inside the crossing structures. The former fact indicates that we could make use of *candidate* junctions to determine the locations of distorted zones, whereas the latter observation suggests that a maximal inscribing circle [Chiang et al., 1998] would be useful

to determine the areas of the distorted zones. We have investigated these two main lines as the core strategy of our approach to detect the distorted zones.

Once the distorted zones are segmented out, the major part of line-drawings shall be well represented by median lines. The 2-junctions are then detected as the meeting points of two straight line segments. To do so, one can employ a polygonal approximation method. However, such a method is sensitive to smooth objects. Taking the circles and arcs for examples, a polygonal approximation method shall produce many small straight line segments, resulting in many false detections of junction points. Interestingly, an alternative approach is the use of a high curvature point detection method as it detects the points in which the curvature is significantly changed in a local neighborhood. We have thus employed the technique of [Teh and Chin, 1989] with a major change in the step of region of support determination, making the method more robust to scaling change and digitalization effect.

The n-junctions ($n > 2$) are reconstructed from the distorted zones. For this concern, several techniques have been introduced using a specific junction optimization process [Hilaire and Tombre, 2006, Maire et al., 2008]. Both these works share the common idea of using distance error minimization as an objective function. The work in [Maire et al., 2008] seems to be robust to distorted edge lines as its main idea relies on an iterative process of junction-branch optimization (i.e., an EM-based algorithm). However, this work is limited to single junction reconstruction and is not scale invariant. To detect the n-junctions, we have extended this work to make it well-adapted to multiple junction detection and more robust to noise.

The global system we propose is presented in Figure 2.1. It exploits five main stages, each of which is briefly described hereafter:

- Pre-processing: Image enhancement and skeletonization.

- Candidate junction detection: This stage detects 2-junction candidates and n-junction candidates ($n > 2$). The 2-junction candidates are detected as the high curvature points of skeleton segments, and the n-junction candidates are detected as the crossing points.

- Distorted zone detection: This stage detects and conceptually removes the distorted zones using the information of the detected junction candidates and the line thickness.

- Extraction of local topology: The local topology around each distorted zone is extracted and then described using a specific structure to support the subsequent stage of junction optimization.

- Junction optimization: This stages clusters the local skeleton segments of a given topology structure into different groups, each of which shall construct a final junction point.

In the following sections, we shall discuss in detail all these stages of our system.

2.1.1 Pre-processing

Our method is applied to binary images. These images could be obtained following some enhancement processes, such as noise filtering and binarization, depending on the specific application. Next, the median lines are pre-extracted using the technique presented in [di Baja, 1994] because it is probably argued as one of the most robust techniques for skeleton extraction in the literature. In addition, this method is time-efficient, supporting all our processes at a low time cost. At the end of this step, the median lines and crossing points (i.e., the points with at least three 8-connected neighbors) are extracted to be used in the subsequent stage (Figure 2.1).

2.1.2 Detection of candidate junctions

In the second stage, candidate junction points are detected from the median lines extracted previously. These candidate junctions, in combination with the line thickness information, are used to detect distorted zones and drive our junction optimization process in the next stages of our system (see Figure 1). In our case, the candidate junctions are classified into 2-junctions and n-junctions (i.e., the junctions formed by n arms with $n \geq 3$). The n-junction candidates are easily extracted by detecting the crossing-points obtained from the skeletonization step.

The 2-junction candidates are detected as the dominant points of the median lines. Generally, the approaches for dominant point detection can be categorized into two classes: multi-scale and single-scale. Although the former approaches [Awrangjeb and Lu, 2008, Mokhtarian and Suomela, 1998] have been known to be robust to noise, they often produce high false positive rates. This matter is realized based on the fact that curve smoothing and curvature estimation are two of the most critical stages of a dominant point detector [Mohammad Awrangjeb and Fraser, 2012, Teh and Chin, 1989]. However, the choice of an appropriate smoothing scale is not trivial and the use of a multi-scale framework to smooth the curve does not solve the problem of scale selection. For the single-scale-based approaches [Carmona-Poyato et al., 2005, Reisfeld et al., 1995, Teh and Chin, 1989], the major challenge is the determination of the *region of support* (ROS) or *local scale*. In our work, the candidate 2-junctions are detected by exploiting Teh-Chin's method [Teh and Chin, 1989] with a major change in the step of determination of ROS. The key idea in Teh-Chin's work relies on the observation that ROS could be determined by measuring the sudden change of the chord length. Given a point p_i of a digital curve, let l_k is the length of the chord connecting two points p_{i-k} and p_{i+k} and h_k is the Euclidean distance of p_i to the chord l_k. The region of support is determined by starting with $k = 1$ and gradually increasing this value until one of two following conditions is satisfied:

$$l_k \geq l_{k+1} \tag{2.1}$$

$$|\frac{h_k}{l_k}| \geq |\frac{h_{k+1}}{l_{k+1}}| \tag{2.2}$$

The condition (2.1) measures the sudden change of the length of two chords between two consecutive iterations while the condition (2.2) limits how far around the curve that

condition (2.1) should be applied. These two criteria could be useful for continuous curves but are fragile in the case of digital curves. Figure 2.2 presents few examples that the Teh-Chin's technique fails to correctly determine the region of support. In the case of a circle in Figure 2.2(a), the condition (2.2) turns out to measure the sudden change of the angle (*i.e.* $\tan(\theta_k) = \frac{2*h_k}{l_k}$) formed by the chord and the line segment $p_i p_{i-k}$ between two consecutive iterations. Therefore, by applying the condition (2.2) and/or (2.1) we conclude that every point on the circle has the same local scale of r which is also the diameter of the circle. This local scale is clearly not an expected one. Naturally, as a circle is a perfectly smooth curve, the local scale at each point on the circle should be selected as one presented in Figure 2.2(d) such that the included angle estimated at that point is close to 180 degrees. In addition, the condition (2.2) is used only when the $h_k \neq 0$ and thus resulting in some cases that we can not determine true scales as presented in Figure 2.2(b) where the point p_i is the middle point on the line segment $q_1 q_2$. In that case, the desirable local scale of p_i should be one as presented in Figure 2.2(e). Most frequently, there exists many configurations of digital curves as presented in Figure 2.2(c) that the Teh-Chin's technique could not determine the correct scale due to digitization effect and noise. Taking Figure 2.2(c) as an example, the region of support of 1 is selected for the point p_i (*i.e.* the point u in the zoomed in version) resulting in an included angle of 90 degrees at that point. This region of support is again not an expected one while the correct local scale for this case should be selected as those in the Figure 2.2(f).

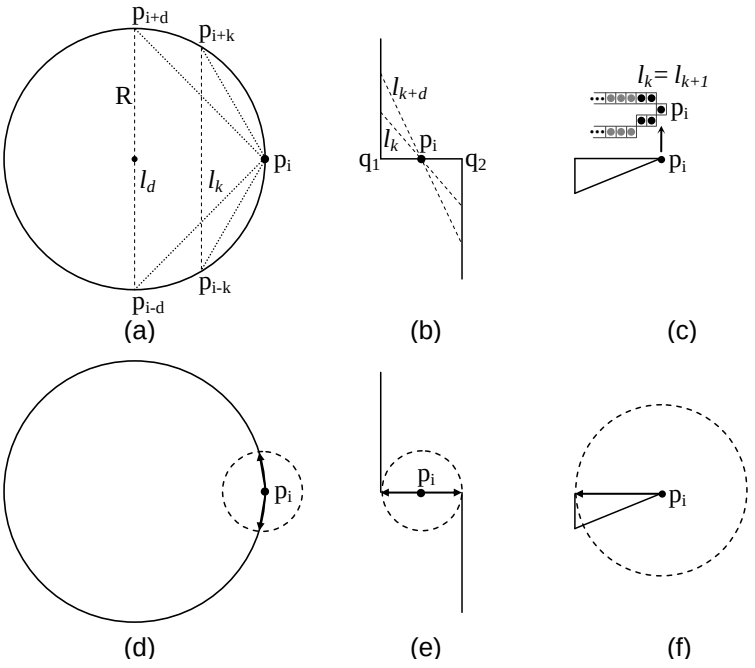


Figure 2.2: Some examples in which Teh-Chin's technique fails to correctly determine the ROS. Top row: (a) $ROS(p_i) = d$ for any point p_i on a circle with radius R, where $l_d = 2R$; (b) $ROS(p_i) = +\infty$ (i.e., p_i is the middle point of segment $q_1 q_2$ and thus $l_{k+d} > l_k$ for any $k, d > 0$); (c) $ROS(p_i) = 1$ (i.e., $l_{k+1} = l_k$). Bottom row: expected ROS for each case in the top row.

Our solution to the problem of ROS determination relies on the observation that for every point p_i of a curve, there exists a trailing line segment (i.e., the segment composing of the points $\{p_i, p_{i-1}, \ldots, p_{i-k_t}\}$) and a leading line segment (i.e., the segment composing of the points $\{p_i, p_{i+1}, \ldots, p_{i+k_l}\}$), where $k_l > 0$ and $k_t > 0$, such that both line segments together constitute a meaningful view of that point regardless of how smooth the curve is. This observation is especially true at dominant points on the curve, where a dominant point is usually treated as the point at which two edges meet and form a vertex. This fact suggests that the ROS of a point could be determined by finding the straight lines fitted to the leading and trailing segments of that point. It turns out that this task could be efficiently accomplished using linear least squares (LLS) line fitting technique. In the proposed approach, given a curve consisting of N ordered points $p_1, p_2, \ldots, p_N$, the ROS at a point p_i is determined as follows:

- Step 1: Start with $k_l = 1$ and gradually increase k_l in increments of one to estimate the straight line d_f of the form $y = \alpha + \beta x$, which provides the best fit for the points $\{p_i, p_{i+1}, \ldots, p_{i+k_l}\}$. The parameters α and β are derived by minimizing the following objective function:

$$Q(\alpha, \beta) = \sum_{j=i}^{i+k_l} (y_j - \alpha - \beta x_j)^2 \tag{2.3}$$

  Next, we define the distance error $h(p_j, d_f)$ computed as the Euclidean distance from a point $p_j(x_j, y_j)$ to the straight line d_f as follows:

$$h(p_j, d_f) = \frac{|\beta x_j - y_j + \alpha|}{\sqrt{\beta^2 + 1}} \tag{2.4}$$

  The step of searching the local scale on the leading segment of p_i will be terminated at some point p_{i+k_l} if either of the two following conditions is satisfied:

$$\frac{1}{k_l} \sum_{j=i}^{i+k_l} h(p_j, d_f) \geq E_{min} \tag{2.5}$$

$$h(p_{i+k_l}, d_f) \geq E_{max} \tag{2.6}$$

  The condition (2.5) requires that the average distance error associated to the fitting line is less than E_{min} pixels. The condition (2.6) is designed to limit the maximum distance error from a point p_j to d_f: no point is E_{max} pixels away from d_f ($E_{max} > E_{min}$). The value $s_l = k_l - 1$ is then treated as the local scale on the leading segment of p_i.

- Step 2: Repeat Step 1 to find the optimal scale $s_t = k_t - 1$ on the trailing segment $\{p_i, p_{i-1}, \ldots, p_{i-k_t}\}$.

- Step 3: The ROS of p_i is finally computed as: $ROS(p_i) = Min(s_t, s_l)$.

Our empirical investigation showed that the values of E_{min} and E_{max} produce a negligible impact on the detection rate provided that $E_{min} \in [1.2, 2.0]$ and $E_{max} \in [1.5, 3.0]$. In our implementation, we fixed the following setting for all the experiments: $E_{min} = 1.3$ and $E_{max} = 1.8$. Once the ROS is determined, we then apply Teh-Chin's algorithm to detect the dominant points from skeleton branches. Figure 2.3 shows the dominant points and the corresponding ROS(s) detected from an image. The detected points, in combination with crossing-points, are treated as the candidate junctions and will be used to detect distorted zones in the next stage.

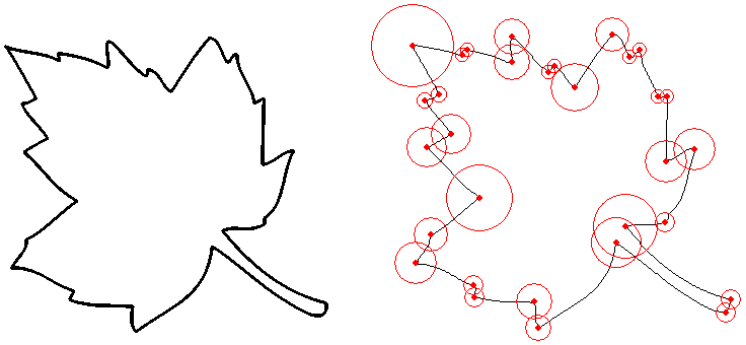


Figure 2.3: (a) An original image; (b) the detected dominant points (small dots) and the corresponding local scales (small circles).

2.1.3 Distorted zone detection

The candidate junction points we have detected previously are used in conjunction with the line thickness information to first detect distorted zones and then conceptually remove these distorted zones to eliminate their interference in terms of the distortion of median lines. In our approach, a distorted zone is identified by tackling two probing questions: where is such a zone likely to occur and how large this distortion zone would be. Naturally, the distorted zones occur at junction locations, and these zones would be restricted to small areas fitting inside the crossing structures. Furthermore, line thickness is also one of main causes of skeleton/junction distortion (i.e., *thin* objects are not or weakly subjected to skeleton distortion). Relying on these observations, the distorted zones could be easily identified by using the information of the line thickness at the candidate junction points detected in the previous steps. More precisely, we define a distorted zone Z_J for a given candidate junction point J as the area constructed by a circle centered at J whose diameter equal to the local line thickness computed at J. This definition is actually a variation of the maximal inscribing circle, as presented in [Chiang et al., 1998]. By making use of line thickness information, these maximal inscribing circles are easily determined with a high degree of accuracy. We call several distorted zones that intersect together a connected component distorted zone ($CCDZ$). Once the CCDZ(s) have been detected, the skeleton segments lying inside these zones are treated as distorted segments and thus removed. From this point, our subsequent stage of junction reconstruction proceeds based on the reliable line segments only. Figure 2.4 (a) shows the reliable segments remaining after removing

all distorted zones (marked as gray connected components).

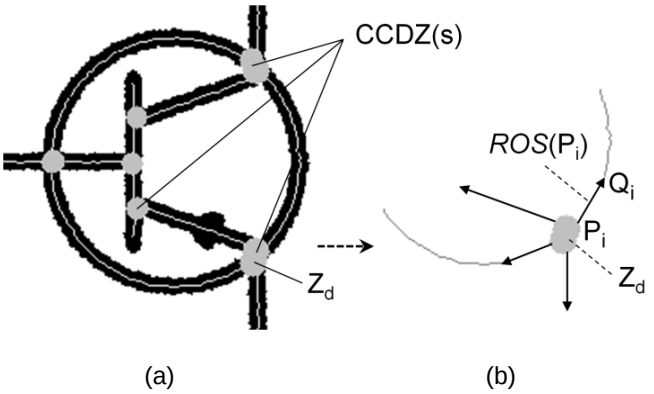


Figure 2.4: (a) An input image with the detected *CCDZs* (gray connected components) and reliable line segments (thin white lines); (b) the local topology defined for a *CCDZ*.

2.1.4 Junction reconstruction

The junction reconstruction exploits candidate junction points to remove possible false alarms, merge candidate junction points, and correct final junction locations. This reconstruction is initiated in a first step by extracting local topologies, corresponding to sets of segments belonging to the same distorted zone or set of intersecting distorted zones. These local topologies will drive a second step in our junction optimization process. We will present these two steps in the following subsections.

2.1.4.1 Extraction of local topology

This step defines and constructs the local topology at each *CCDZ*. In particularly, given a *CCDZ*, its local topology is defined as the set of local lines segments, $\{P_iQ_i\}_{i=1,...,n}$, stemming from this *CCDZ*. That is, for each reliable skeleton segment stemming from a *CCDZ*, we characterize the first part of this segment by a local line segment starting from the extremity linked to the *CCDZ*. By defining and analyzing these local geometry topologies, we have significantly simplified the complexity of the objects of input images; thus, the proposed approach is able to work on any type of shape rather than straight lines and/or arc primitives exclusively. Moreover, this step can be performed efficiently by reusing the results of the ROS determination stage applied at each extremity of each reliable skeleton segment stemming from the *CCDZ*. As a result, for each *CCDZ*, we obtain a list of local line segments describing its local geometry topology. In addition to these local lines, the foreground pixels lying inside the *CCDZ* are also recorded for use as a local search neighborhood for the subsequent step of junction optimization. In summary, the local topology associated with a *CCDZ* is now represented by a list of n local lines, $\{P_iQ_i\}_{i=1,...,n}$, and a set, Z_d, containing the foreground pixels located inside the *CCDZ*. Figure 2.4 (b) illustrates a local topology extracted for a *CCDZ*.

2.1.4.2 Junction optimization

The goal of this step is to reconstruct the junction points for a specific *CCDZ* represented by n line segments $\{P_iQ_i\}_{i=1,\ldots,n}$ and a set, Z_d, of foreground pixels lying inside the *CCDZ*. We accomplish this goal by clustering the line segments into different groups such that the clustered lines in each group will be used to form a junction point. Concerning this problem of clustering segments, the authors in [Hilaire and Tombre, 2001], as discussed above, calculated the intersection zones from the uncertainty domains of the long primitives. This approach is subjected to the constraint that each primitive is allowed to be clustered in one group only, increasing the difficulty of the subsequent junction linking step. Another approach to segment clustering was presented in [Maire et al., 2008] based on the idea that if we know the position of the junction, the associated line segments passing through this junction could be easily identified and *vice versa*. However, this work assumed that each neighborhood (see Figure 2.5) contains one junction only, and this approach is subjected to high computation load because the optimization step must include sufficiently large neighborhoods likely containing junctions to reduce the error introduced by the previous step of contour detection. In addition, the reweighting step does not consider the weights accumulated during the previous iterations. This omission may lead to incorrect convergence, as shown in Figure 2.5.

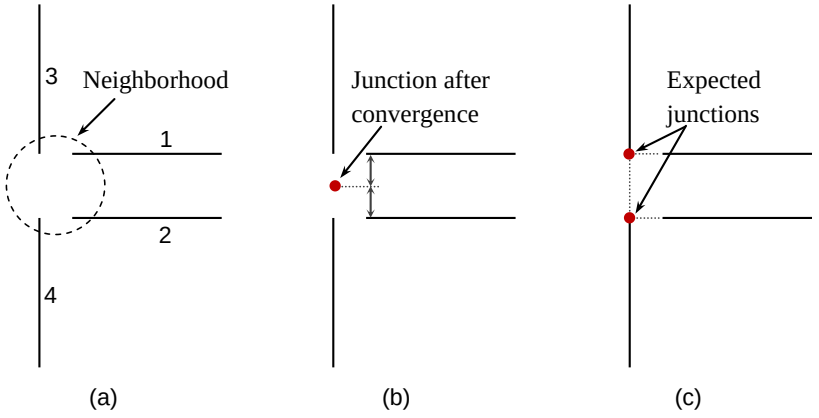


Figure 2.5: Incorrect convergence of the junction optimization step in [Maire et al., 2008]: (a) four line segments with the same length; (b) the detected junction, which is the same distance from the lines 1 and 2; (c) the expected junctions.

We therefore develop an integrated solution for both clustering and optimizing tasks to address the aforementioned weaknesses, described below. In particularly, the proposed algorithm is able to handle the following issues simultaneously: (1) each neighborhood can contain multiple junctions, (2) each line segment can be clustered into more than one group, and (3) junction linking and characterization are automatically derived.

The key spirit behind our algorithm is as follows. Starting from a *CCDZ* (e.g., Figure 2.7(c)), a new junction point is constructed by iteratively searching for a foreground pixel of the *CCDZ* such that the distance error, computed as the sum of weighted Euclidean distances from this pixel to the line segments of the *CCDZ*, is minimized (e.g., Figure 2.7(d)). To achieve this goal, each line segment has to be assigned with a proper weight in

the sense that the lines, which are close to the junction, would have higher weights than the ones away from the junction. This implies that the weights are set mainly relying on the distances from the junction to the lines. In practice, a smooth function (e.g., Gaussian function) should be used to update the weights. As the algorithm evolves, the junction tends to converge towards the lines with higher weights and move away from the lines with lower weights. Hence, the junction would converge to a fixed point after several iterations (e.g., Figure 2.7(d)). Once a newly optimized junction is derived, the weights are re-assigned in such a way that higher weights are given to the lines which have not been involved in constructing the junctions previously. The optimization process is then repeated to find a new junction (e.g., Figure 2.7(e, f)). This continues until every line segment has been participated in constructing at least one junction. A final post-processing step is then applied to make the topology of the obtained junctions be consistent (e.g., Figure 2.7(g)). The proposed algorithm works as follows.

Let w_i be a weight assigned to the line segment P_iQ_i with $1 \leq i \leq n$, and Ω_J be a set of optimal junctions found during the optimization process. At the beginning, $\Omega_J \leftarrow \emptyset$ and all line segments $\{P_iQ_i\}$ are marked as *unvisited*. Basically, the weights could be initiated uniformly (e.g., $w_i = 1.0$), but we can make faster the process of junction convergence by incorporating some priority. One common way is to assign the weights with respect to the strength of the line segments [Hilaire and Tombre, 2001, Maire et al., 2008]. Followings are the main steps of the proposed algorithm with the outline in Figure 2.6.

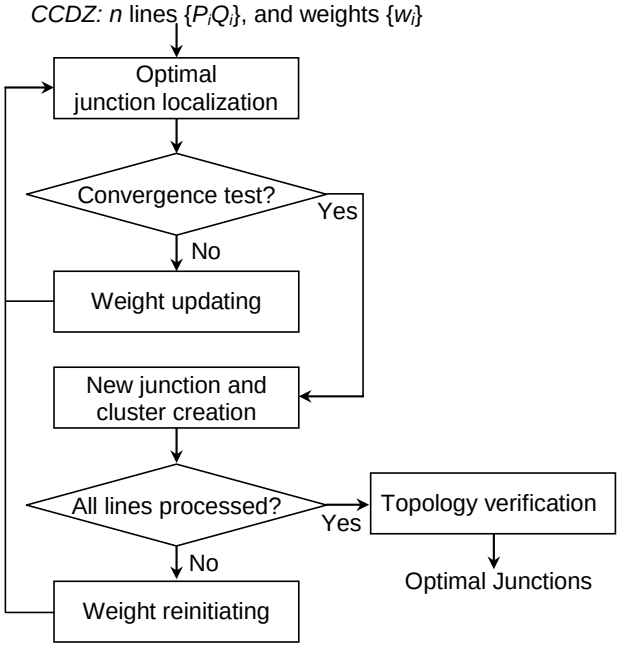


Figure 2.6: Outline of our junction optimization algorithm.

- Step 1: Search for an optimal junction J^*:

$$J^* = \arg\min_{J \in Z_d} \{\sum_{i=1}^{n} w_i \cdot d(J, P_iQ_i)\} \tag{2.7}$$

where $d(J, P_iQ_i)$ is the Euclidean distance from J to P_iQ_i.

- Step 2: Update the weights $\{w_i\}$:

$$w_i = w_i \cdot \exp(\frac{-\pi \cdot d(J^*, P_iQ_i)^2}{S_{CCDZ}}) \tag{2.8}$$

where S_{CCDZ} is the area of the *CCDZ*.

- Step 3: Enact a penalty (i.e., smaller weight) for the line segment farthest from J^*:

$$w_{imax} = \frac{w_{imax}}{\tau} \tag{2.9}$$

where $imax = \arg\max_i\{d(J^*, P_iQ_i)\}$ and $\tau > 1$. If there are several line segments with the same greatest distance from J^*, one is randomly selected to assign a penalty.

- Step 4: Repeat steps $\{1, 2, 3\}$ until J^* converges to a fixed point or a given number of iterations has been reached. Then, insert the newly obtained junction to Ω_J: $\Omega_J \leftarrow \Omega_J \cup \{J^*\}$, and go to Step 5.

- Step 5: Determine the line segments that pass through the junction J^* and mark them as *visited*. A new cluster is constructed corresponding to these line segments. If all line segments have been marked as visited, go to Step 7. Otherwise, go to Step 6 to look for other junctions.

- Step 6: Reinitiate the weights: $w_i = 1$ if the label of P_iQ_i is *visited*; otherwise:

$$w_i = \prod_{k=1}^{L} \exp(\frac{\pi \cdot d(J_k^*, P_iQ_i)^2}{S_{CCDZ}}) \tag{2.10}$$

where L is the number of times that steps $\{1, 2, 3, 4, 5\}$ have been fulfilled, and J_k^* is the optimal junction found during the corresponding cycle. Return to Step 1.

- Step 7: Topology consistency verification by resetting the weights: $w_i = 1$ if the line P_iQ_i is involved in only one cluster; otherwise $w_i = w_\infty = K \cdot \sqrt{H^2 + W^2}$, where W and H are the width and height of input image respectively, and $K = |\Omega_J|$. Then, apply Step 1 to the line segments in each cluster to obtain the final junctions. In this way, the lines assigned with the weight w_∞ are fixed in one place.

The idea of using distance error minimization in Step 1 has been employed in several works. [Hilaire and Tombre, 2001] employed least squares error minimization to find the optimal position of the junction, but this process is performed separately from segment clustering. [Maire et al., 2008] developed this idea by incorporating a reweighting step like that in Step 2 but differing in that it does not incorporate the weights accumulated during the previous iterations and requires a training step to empirically derive a parameter controlling the decay of distance tolerance.

Our investigation has shown that Step 1 will quickly converge to the optimal junction if the weights are updated taking into account the weights derived during the previous

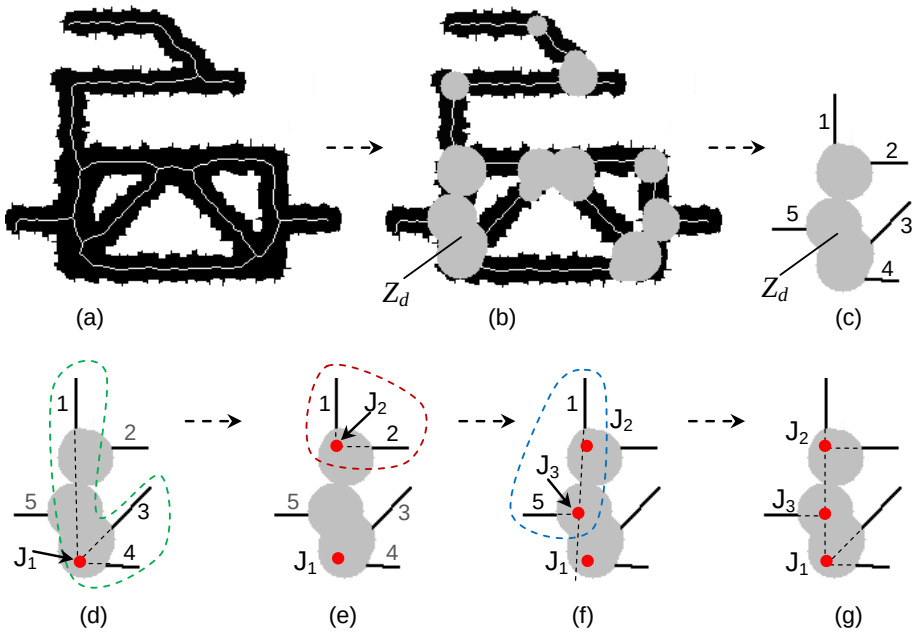


Figure 2.7: (a) An image with its skeleton; (b) the *CCDZ*(s) and the reliable line segments; (c) the local topology configuration extracted for one *CCDZ* (marked by Z_d); (d) the first cycle of steps $\{1, 2, 3, 4, 5\}$: the junction J_1 is found corresponding to the cluster containing line segments $\{1, 3, 4\}$; (e) the second cycle: the junction J_2 is found corresponding to the second cluster comprised of lines $\{1, 2\}$; (f) the third cycle: the junction J_3 is found corresponding to the third cluster of $\{1, 5\}$; (g) topology correction for three junctions.

iterations. In addition, we avoid the training step by normalizing the distances, $d(J, P_iQ_i)$, based on the area of the *CCDZ* (i.e., the factor $\pi \cdot d(J, P_iQ_i)^2 / S_{CCDZ}$ is equal to the ratio of the area constructed by a circle with radius of $d(J, P_iQ_i)$ centralized at J and the area of *CCDZ*). Step 3 enacts a penalty (the parameter $\tau = 2$ in our implementation) for the line segment farthest from the optimal point J^*. If there are several line segments with the same greatest distance from J^*, one is randomly selected to assign a penalty. This step is used to allow the optimization process to quickly converge to a correct junction location. More importantly, it acts as a *trigger* to break the balance state or incorrect convergence, if any, as discussed in Figure 2.5. Note that if one line segment is penalized, it does not imply that this line segment will not pass through the *latest* optimal junction.

Step 4 is used to repeat the three steps above until the optimal junction is found. The obtained junction is then added to the set Ω_J (e.g., the junction J_1 in Figure 2.7(d)). Next, the line segments that actually form this junction are determined by looking for the lines whose distances from the detected junction tend to form a monotonically decreasing order (e.g., the lines $\{1, 3, 4\}$ in Figure 2.7(d)). This is accomplished because at each iteration, the optimal junction would converge towards the lines with higher weights and move away from the lines with lower weights. Therefore, the distances from the optimal junction in each iteration to the line segments are recorded and then used to determine the real lines passing the most recently found junction. The obtained lines are then associated to a new cluster, and marked as visited (i.e., already involved in at least one junction). If all

lines have participated in constructing at least one junction, the algorithm terminates after checking topology consistency in Step 7. Otherwise, Step 6 is invoked to initiate a new cycle to find other junctions (i.e., a cycle is composed of the first five steps $\{1, 2, 3, 4, 5\}$ to completely find a new optimal junction).

In Step 6, the weights are reinitiated such that more priority or higher weights are given to the lines that have not yet been involved in junction construction (e.g., the lines $\{2, 5\}$ in Figure 2.7(d)). To this end, the recorded distances that violate the monotonic decrease are used to accumulate the weights for the lines. In this way, these lines increasingly gain weight, and at the end, when the weights are large enough, the optimization process (i.e., steps $\{1, 2, 3, 4\}$) will be driven by these weights, leading to a new junction convergence at the corresponding lines (e.g., J_2 in Figure 2.7(e) and J_3 in Figure 2.7(f)).

Step 7 is aimed at verifying the topology consistency of all line segments in the obtained clusters. At this time, we obtain K clusters, each containing one optimal junction. As one line segment, say P_iQ_i, can be clustered in several groups (e.g., the line 1 in Figure 2.7(f)) and there is no warranty that all the optimal junctions in these groups will form a straight line that fully contains P_iQ_i, such situations must therefore be identified and corrected. This step could be easily processed by setting a large enough weight for the line P_iQ_i and then performing Step 1 once for each cluster. In this way, a small change in the distance error computed from each point $J \in Z_d$ to the line P_iQ_i will cause a large change in the objective function in Step 1. The line P_iQ_i is thus fixed in one place, and the new optimal junctions found in the clusters, in which P_iQ_i is involved, become consistent (Figure 2.7(g)). Figure 2.7 demonstrates the steps of our junction optimization algorithm, and Figure 2.8 shows all the detected junctions and the corresponding local scales.

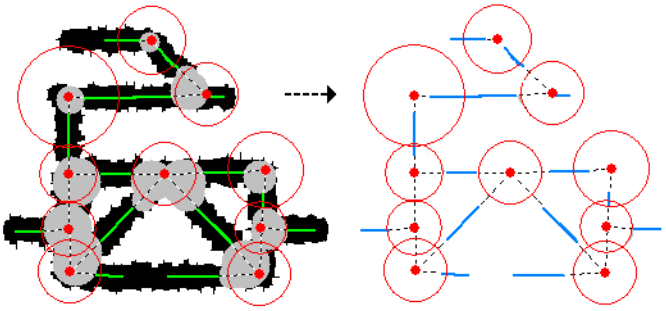


Figure 2.8: Detected junctions (red dots) and local scales (circles).

2.2 Junction characterization

One of the main advantages of our junction reconstruction process is that the detected junctions could be automatically characterized and classified into different types, such as T-, L-, and X-junctions. More generally, we wish to characterize any complicated junctions in the same manner based on the arms forming the junction. In our case, as each junction point is constructed from the local line segments of one group, we can consider these line segments as the arms of the junction point. However, as each *CCDZ* can contain multiple

junctions and each local line segment of the *CCDZ* can participate in several groups, the exact arms of a junction could therefore be greater than the line segments forming this junction (Figure 2.9). Given a local topology represented by n straight line segments $\{P_iQ_i\}$ with $1 \leq i \leq n$, the process of determination of the exact arms of each junction is as follows:

- Let O_J be a set of arms of junction J where $O_J \leftarrow \emptyset$ at the beginning for every junction.
- If the line segment P_iQ_i is clustered in a group whose an optimal junction J is then constructed, the line P_iQ_i is considered as one of the arms of the junction J: $O_J \leftarrow O_J \cup \{P_iQ_i\}$.
- For each line segment P_iQ_i that is clustered in several groups, the corresponding junctions involved in P_iQ_i are sorted in the order of increasing distance to P_i. Then, for each junction J *except the last one* in the list, the corresponding set O_J is updated as: $O_J \leftarrow O_J \cup \{JG\}$, where JG is a straight line segment constructed at J with the same length as P_iQ_i but the point G lies in the opposite direction of vector $P_i\vec{Q}_i$.

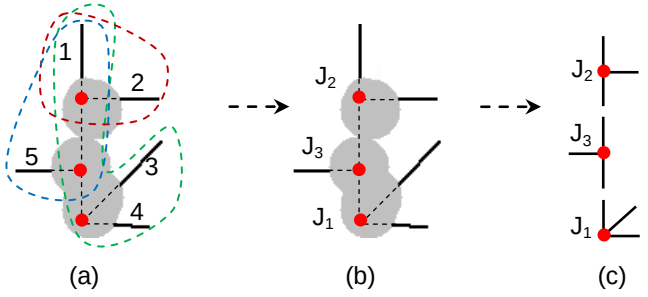


Figure 2.9: (a) A *CCDZ* cropped from Figure 2.7 with the superposition of three clusters; (b) detected junctions $\{J_1, J_2, J_3\}$; (c) the detected junctions are classified as a 3-junction even though J_2 and J_3 are constructed from two clusters, $\{1, 2\}$ and $\{1, 5\}$, respectively, each of which only contains two local line segments.

Once we have correctly determined the arms of each junction, the junction characterization could easily be accomplished as follows. Given a junction J associated with a set of m arms $\{U_iV_i\}_{i=0,\ldots,m-1}$, the characterization of this junction is described as $\{p, s_p, \{\theta_i^p\}_{i=0}^{m-1}\}$, where:

- p is the location of J;
- s_p is the local scale computed as the mean length of the arms of J:

$$s_p = \frac{1}{m}\sum_{i=0}^{m-1} |U_iV_i|$$

- θ_i^p is the difference in degrees between two consecutive arms U_iV_i and $U_{i+1}V_{i+1}$. These parameters $\{\theta_i^p\}_{i=0}^{m-1}$ are tracked in the counterclockwise direction and the θ_{m-1}^p is the difference in degrees between the arms $U_{m-1}V_{m-1}$ and U_0V_0.

It is noted that a similar way of junction characterization has been also exploited in the CV field, taking the work of [Xia, 2011] for example. In their work, the junction arms $\{\theta_i\}_{i=0}^{m-1}$ are described using the absolute angles in a 2D plane. Here, we use the relative difference in degrees between two successive arms to make the junction characterization invariant to the image plane and make it easier for further processes of junction matching.

The description of each junction point derived in this way is rather compact, distinctive, and general. The dimension of this descriptor is variable but limited to the number of arms of each junction point, and in practice, this value is quite small (e.g., 3 for a T-junction, 4 for an X-junction). This point constitutes a great advantage of the detected junctions that provides a very efficient approach to the subsequent task of junction matching. In addition, the junction descriptor is distinctive and general, such that we can describe any junction points appearing in a variety of complex and heterogeneous documents. After this step, junction matching can be performed by simply comparing the descriptors of two junctions. Figure 3.5 shows the corresponding matches of the junctions detected in a query symbol (left) and those of an image cropped from a large document (right). For simplicity, the matches are shown after performing geometry checking using the Generalized Hough Transform [Ballard, 1981].

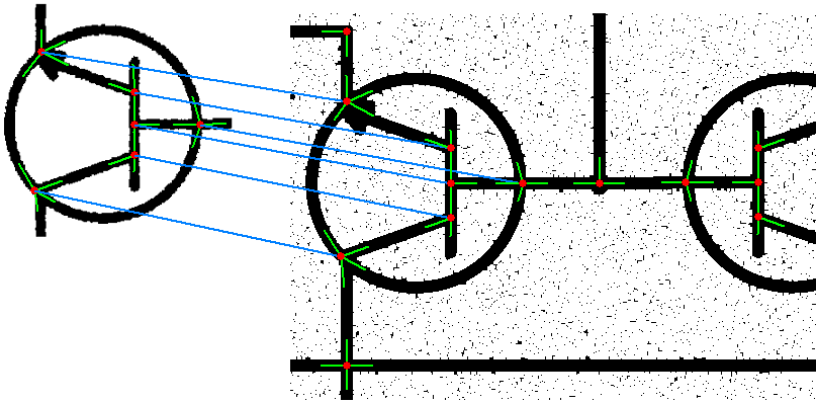


Figure 2.10: Corresponding junction matches between a query symbol (left) and a cropped document (right).

2.3 Complexity evaluation

In this section, we provide a detailed analysis of the complexity of the proposed method given an image I of the size $M \times N$. In the pre-processing stage, before applying the (3,4)-distance transform skeletonization algorithm, several basic pre-processing steps, such as hole filling, small contour removing, and image dilation, are performed, as discussed in the original work of Di Baja [di Baja, 1994]. Such steps can be processed in parallel using one scan over the image. The skeletonization step is then applied, which requires two scans of the image to calculate the (3,4)-chamfer distance. In summary, the computation complexity for the pre-processing stage is basically linear (i.e., $O(MN)$).

In the next stage of scale selection and dominant point detection, the ROS determination step is applied for each skeleton point, thus using a single loop of length S, where S is the number of skeleton points. The technique of least squares line fitting is a second-order linear computation of the length that it traverses. In practice, it is not necessary to traverse a full skeleton branch; rather, a short path of the branch with a length $k_\rho = 50$ (pixels) may be sufficient, for example. As the technique of least squares line fitting is performed in both directions at each point, it is equal to a complexity of $O(2Sk_\rho^2)$ in total for this step. The 2-junctions are then detected as dominant points by applying Teh-Chin's algorithm, which is a sequential 4-pass process, where the first pass is performed on the full length of the median lines to detect a list of H candidate dominant points and the other passes are conducted on one of these candidate points, where H is much smaller than S. Furthermore, the crossing-points could be detected in parallel with a cost of first-order linear polynomial time $O(S)$. The overall computation complexity for these processes is essentially linear to the length of the median points (i.e., $O(Sk_\rho^2)$).

For the last stage of junction reconstruction, let K be the number of candidate junctions comprised of 2-junction points and crossing-points. The distorted zone Z_i defined at each candidate junction p_i ($1 \leq i \leq K$) has an area of $\pi r_i^2/4$ where r_i is the line thickness at p_i. Given a distorted zone Z_i, the maximum complexity of the computation to find an optimal junction in Z_i is $O(T\pi r_i^2/4)$ where the first factor, T, is the number of times that steps $\{1, 2, 3\}$ are repeated and the second factor, $\pi r_i^2/4$, is the number of foreground pixels in Z_i (i.e., the local searching neighborhood). Our investigation has shown that the number of iterations, T, is very small and is typically less than 10. As Z_i can contain multiple junctions, say L junctions, it implies that the junction optimization process applied to Z_i will be terminated after running L iterations of the steps $\{1, 2, 3, 4, 5\}$. The value of L is also very small in practice, often 2; thus, for a wide range of situations, we have set $L = 5$ in our implementation. Overall, the maximum complexity of computation for this stage, applied to K distorted zones, is $O(KLTr^2)$ where r is the average line thickness of the image I. In other words, this stage is linear time complexity for the areas of the distorted zones. Note that the distorted zones, in practice, could intersect, resulting in connected component distorted zones and making the searching areas much smaller.

2.4 Experimental results

2.4.1 Evaluation metric and protocol

We use *repeatability* criterion to evaluate the performance of our junction detector because this criterion is standard for the performance characterization of local keypoint detectors in CV [Tuytelaars and Mikolajczyk, 2008]. This criterion works as follows. Given a reference image I_{ref} and a test image I_{test} taken under different transformations (e.g., noise, rotation, scaling) from I_{ref}, the repeatability criterion signifies that the local features detected in I_{ref} should be repeated in I_{test} with some small error ϵ in location. We denote $D(I_{ref}, I_{test}, \epsilon)$ as the set of points in I_{ref} that are successfully detected in I_{test} in the sense that for each point $p \in D(I_{ref}, I_{test}, \epsilon)$, there exists at least one corresponding point $q \in I_{test}$ such that $distance(p, q) \leq \epsilon$. Let n_r and n_t be the number of keypoints detected by one detector from I_{ref} and I_{test}, respectively. The repeatability score of this detector

applied to the pair (I_{ref}, I_{test}) is computed as follows:

$$r(I_{ref}, I_{test}, \epsilon) = \frac{|D(I_{ref}, I_{test}, \epsilon)|}{Mean(n_r, n_t)} \tag{2.11}$$

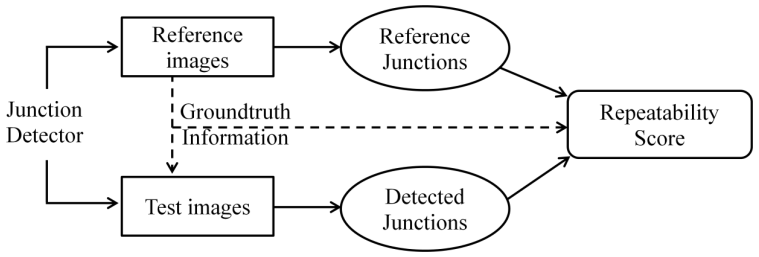


Figure 2.11: The evaluation strategy applied to each detector.

Our strategy to perform evaluation in the experiments is described in Figure 2.11. This strategy follows the general characterization protocol for keypoint detection in CV. Particularly, we first apply each detector to the reference images and the test images in each dataset to obtain the reference junctions (S_r) and the detected junctions (S_t), respectively. Then, we use groundtruth information to compute repeatability scores of this detector from two sets of junctions: S_r and S_t. The overall repeatability score of each detector in each experiment is computed as an average score from the repeatability scores of the detector applied for all model symbols and test symbols in each dataset. We vary the value of parameter ϵ in the range of $[1, 8]$ to obtain a *ROC-like* curve of the repeatability score.

2.4.2 Baseline methods

To compare the proposed system with other methods, we have selected two baseline systems [Liu et al., 1999, Hilaire and Tombre, 2006] dedicated to junction and fork-point detection. The work of [Liu et al., 1999] is dedicated to fork-point detection in handwritten Chinese characters, whereas the work of [Hilaire and Tombre, 2006] is a vectorization-based system for line-drawings. We wish to highlight that although the later work is designed for vectorization, the major contribution in this work is the process of skeleton optimization to correct skeletons and reconstruct junctions. As the implementations of these works are not publicly available, we have developed our own implementation for these two systems[1]. For each system, several running trials have been performed to select the best parameter settings, and only junctions or fork-points are compared with our detected junctions. It is worth noting that we have applied the same pre-processing steps for all three systems and used the same parameter settings in all the experiments.

2.4.3 Datasets

The datasets used in the experiments are summarized in Table 2.1, including the final datasets from Symbol Recognition Contest in GREC2011 (SymRecGREC11)[2], the UMD

[1]The source codes for the three systems and the demonstration of our junction detector and symbol spotting are publicly available at https://sites.google.com/site/ourjunctiondemo/

[2]http://iapr-tc10.univ-lr.fr/index.php/symbol-contest-2011

Table 2.1: Datasets used in our experiments

No.	Dataset	Type	Noise	#Images	
				#References	#Tests
#1	GREC11	Line-drawing	Rotation	150	1339
#2	GREC11	Line-drawing	Scaling	150	1200
#3	GREC11	Line-drawing	Kanungo+Rotation+Scaling	150	15000
#4	GREC11	Line-drawing	Context	18	1800
#5	SESYD	Line-drawing	Low Resolution	100	936
#6	UMD Logos	Filled-shape	Kanungo+Rotation+Scaling	104	1272

Logo Database of University of Maryland, Laboratory for Language and Media Processing (LAMP)[3], and low resolution diagram dataset from SESYD[4]. The SymRecGREC11 dataset is composed of 4 folders, namely *setA*, *setB*, *setC*, and *setD* with respect to 2500, 5000, 7500, and 1800 test images, respectively. The first three folders are distorted by a mixture of Kanungo noise and geometric transformations (i.e., scaling and rotation), whereas the last dataset is disturbed by context noise (i.e., symbols cropped from full line-drawing images). The UMD Logo Database consists of 104 model logos, which have been used to generate 1272 test images by applying a combination of Kanungo noise and geometric transformations. The low resolution diagram SESYD dataset contains 100 reference images and 936 test images by applying 4 levels of low resolution, corresponding to the scaling factors $\{1/2, 1/4, 1/8, 1/16\}$. Consequently, the image resolution is varied from 1700×1700 to 100×100, and the line thickness is also varied in the range of $[2, 18]$. These test images are then exported in PNG format in which some blurring effect will be automatically incorporated to these images. In addition, for the evaluation of single parameter changes (i.e., rotation and scaling), we have used 150 model symbols from GREC2011 to generate 1339 test images taken under different levels of rotation (e.g., from 10^0 to 90^0) and 1200 test images taken under different scaling factors (i.e., $\{1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0\}$).

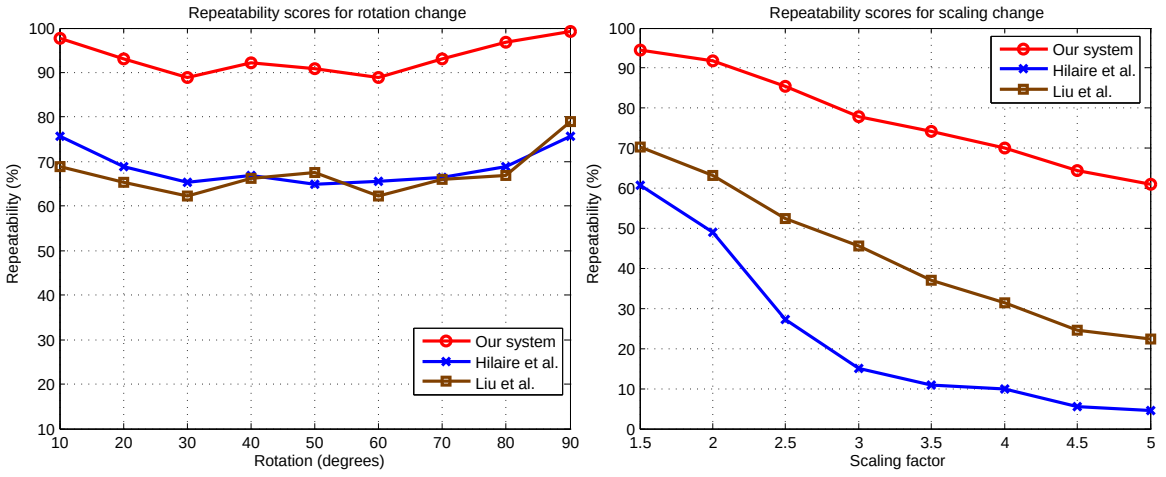
2.4.4 Comparative results

2.4.4.1 Evaluation of rotation and scaling change

In this experiment, the repeatability scores are computed over single parameter changes while the location error is fixed at 4 (pixels). Figure 2.12(a) presents the effect of rotation change for three systems. It can be noticed that the proposed approach far outperforms (almost 25%) the two other systems and that our repeatability scores tend to be quite stable and almost over 88% when varying the rotation parameter. These results confirm that the proposed system is very robust to rotation change. The systems of Liu et al. and Hilaire et al. are theoretically rotation invariant; however, the results reported here show that these systems are less adaptive to rotation change under real-world conditions.

In the context of scaling change, as shown in Figure 2.12(b), the same situation is repeated for the baseline methods, whereas the proposed system still strongly outperforms

[3]http://lampsrv02.umiacs.umd.edu/projdb/project.php?id=47
[4]http://mathieu.delalandre.free.fr/projects/sesyd/



(a) (b)

Figure 2.12: Repeatability scores of three systems on rotation change (a), and scaling change (b), where location error is set at 4 pixels.

the others. It is noticed, in particularly, that the results obtained by the system of Hilaire et al. are significantly degraded when increasing the scaling factor. This degradation could be due to the two baseline systems being quite sensitive to the digitization effect caused by rotating and scaling the input images. In fact, the results reported in the work of [Hilaire and Tombre, 2006] are applied to several line-drawing images that are typically used in the context of vectorization contests, whereas the results reported by [Liu et al., 1999] are applied to handwritten Chinese characters, which are not taken under extreme rotation/scaling changes.

2.4.4.2 Evaluation of a mixture of Kanungo noise and rotation/scaling change

We have selected the first three sets, *setA*, *setB*, and *setC*, from the final recognition datasets in GREC2011 to evaluate the performance of three systems under different combination of binary noise and geometric transformations. Some examples of such degradation are shown on Figure 2.14 (a, b). The purpose of this experiment is to justify how well each system can work under different levels of degradation. The results are presented on Figure 2.13 where the proposed system achieves much better results on all three *setA*, *setB*, and *setC*, compared to the systems of Hilaire et al. and Liu et al. On average, the repeatability scores obtained by the proposed system are 15% higher than those of the other systems, especially for the first small range of location errors (e.g., see the first part of the score curve of the proposed system). If we have a more detailed analysis at our results by fixing the location error at 4 (pixels), it can be noticed that the repeatability scores of the proposed system are almost 80% on all three *setA*, *setB*, and *setC*. These obtained results are quite interesting considering a severe degree of degradation applied on these datasets. Under the same conditions, we can see that the scores of two baseline systems are approximately 60%, which is much less than that of the proposed system. These results suggest that the

proposed system can resist to a satisfactory level of degradation composing of common binary noise and geometric transformations.

Two main factors explain these results. First, the polygonization process in the system of Liu et al. and the skeleton segmentation step in the system of Hilaire et al. are rather sensitive to the distortion of contours. Second, the post-process of junction merging using *Criterion A* is quite sensitive to the variation and distortion of the line thickness of foreground objects. The results of our system suggest that the proposed system can satisfactorily resist degradation including common binary noise and geometric transformations. A last noticeable point exhibiting on Figure 2.13 is that that there is a little difference of performance obtained on the *setA*, *setB*, and *setC* for all three systems because there is, in fact, no significant difference of degradation among the images of these three test sets.

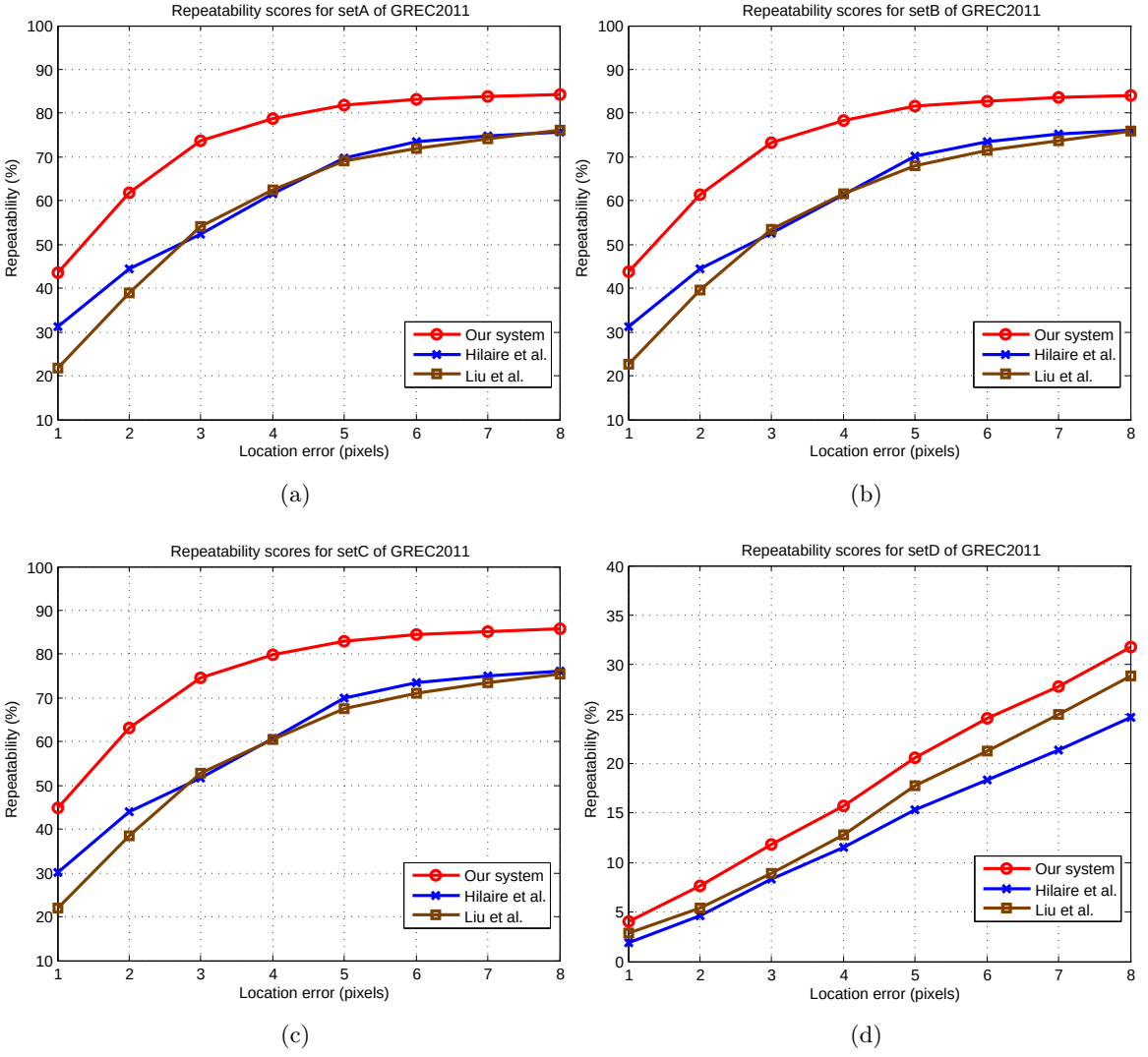


Figure 2.13: Repeatability scores of three systems for *setA*, *setB*, *setC*, and *setD* of GREC2011.

2.4.4.3 Evaluation of context noise

Although Kanungo noise and geometric transformations are very common degradation models in DIA, a more realistic type of degradation is known as *context* noise. By definition, context noise is concerned with a type of disturbance caused by background or context information. For this purpose, we have selected *setD* from the final recognition dataset in GREC2011. The test images in this set have been cropped from full line-drawing documents where each reference image could be touched with other context information. The repeatability scores of three systems are reported in Figure 2.13(d). Although the proposed system still outperforms the others, the repeatability scores of all systems are quite low. This finding is attributed to the fact that the images in *setD* are embedded into other context information, resulting in many false positives being detected in this set, as shown in Figure 2.14 (c, d). However, without any prior knowledge about groundtruth information, these false alarms correspond to the mismatches of correct detected junctions missing in the reference images.

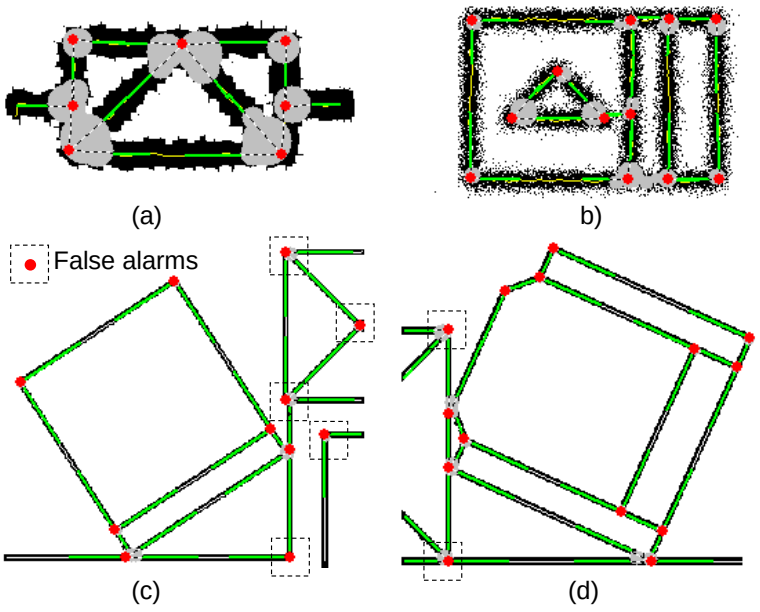
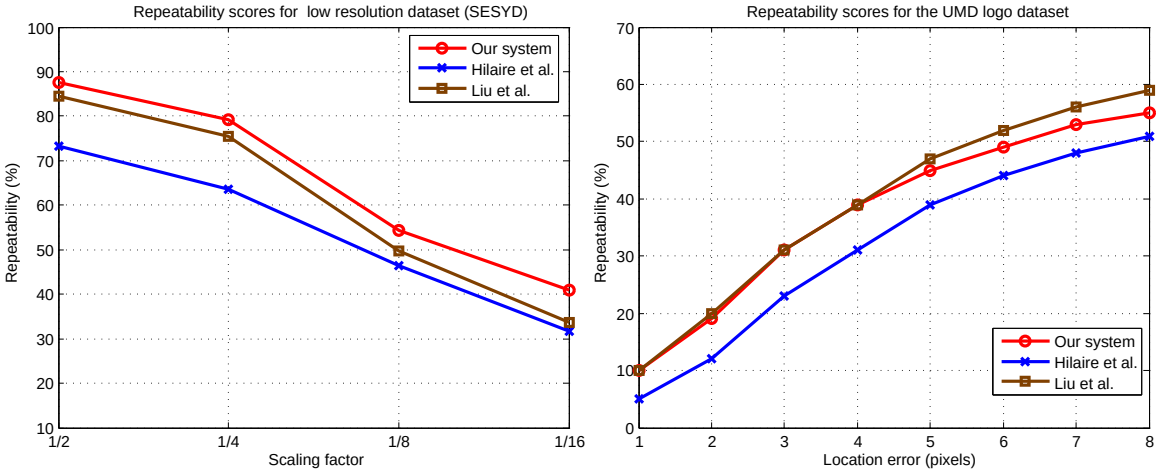


Figure 2.14: Junctions detected (red dots) by the proposed system for *setC* ((a) and (b)) and *setD* ((c) and (d)) of GREC11. False positives caused by context noise in *setD* are marked by dashed-line boxes.

2.4.4.4 Evaluation of the low resolution dataset

In this experiment, we wish to assess the performance of the three systems for very severely low resolution images. We have selected the low resolution diagram SESYD dataset, in which the test images have been generated from the reference images by applying four exponential levels of low resolution corresponding to the scaling factors $\{1/2, 1/4, 1/8, 1/16\}$ and incorporating a compression scheme (JPG) from gray-scale images. The results of the three systems are shown in Figure 2.15(a), where the proposed

system again performs better than the baseline systems. On average, the proposed system provides 10% and 5% better results than those obtained from the systems of Hilaire et al. and Liu et al., respectively. All three systems perform quite well on the first levels of low resolution, but their performance rapidly degrades for the later levels of low resolution. This behavior is mainly due to the loss of much of the original information, especially finer features, when reducing the resolution.



(a) (b)

Figure 2.15: Repeatability scores of three systems for SESYD low resolution dataset (a), and the UMD logo dataset (b).

2.4.4.5 Evaluation of the filled-shape and non-uniform stroke dataset

In this experiment, we want to justify how different kind of images, such as filled-shape and non-uniform stroke images, can impact the performance of the three systems. For this purpose, we have selected the UMD logo dataset, which typically composes of filled-shape and non-uniform stroke objects at a hard level. The repeatability scores are presented in Figure 2.15(b). Even though the skeleton-based representation for such kind of images is not perfect, the obtained results are encouraging. The system of Hilaire et al. achieves the lowest scores because of a lower number of outputted junctions. As the line thickness of the filled shapes is greatly varied compared to the typical line-drawings, many short skeleton segments are produced. Consequently, few long skeleton segments are retained, and thus the number of detected junctions is rather limited in the system of Hilaire et al. Our system also produces a limited number of junctions, even less than that of the Hilaire's system, but it still noticeably outperforms the system of Hilaire et al. and almost gives the same results as those of the system of Liu et al. These results confirm the accuracy of the detected junctions of the proposed system. Some visual results of the proposed system, applied to the logo images and Chinese characters, are shown in Figure 2.16.

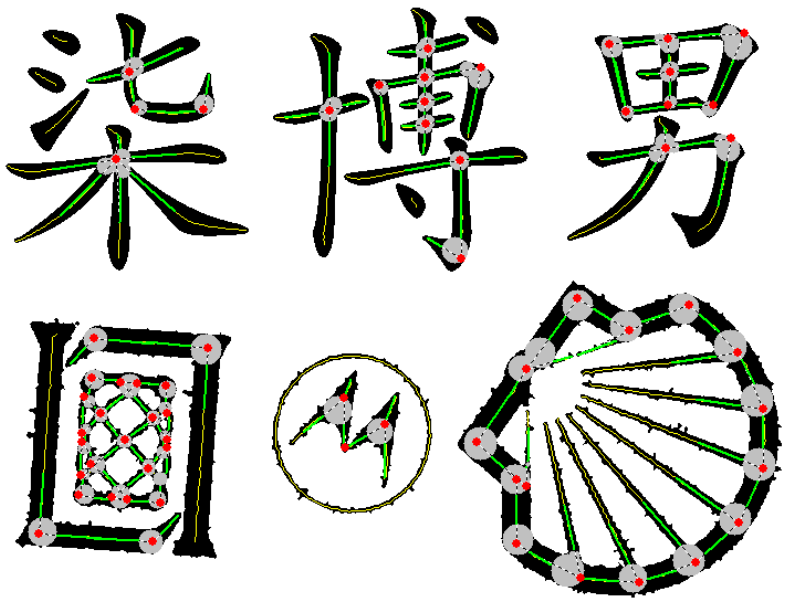


Figure 2.16: Junctions detected (red dots) by the proposed system for few Chinese characters and logo images.

2.4.4.6 Evaluation of the built-in aspects

In addition to the evaluations discussed above, we have investigated several additional trials to understand the behavior of the proposed approach at the system level. In particularly, we wish to present a detailed analysis of the impact of the stage of determination of ROS and the computation time of our junction detector. Regarding the first aspect, we have computed the repeatability scores for the $setC$ up to the stage of dominant point detection over three different scenerios: the use of ROS based on local line thickness (i.e., the ROS at given a point p is set as the local line thickness at p), the use of ROS proposed by [Teh and Chin, 1989], and the use of ROS proposed by our approach. The results are presented in Table 2.2. It can be seen that our method achieves much better results than the others (by almost 23%). The results linked to the ROS proposed by Teh-Chin are quite low because, as we have discussed in Section 2.1.2, the Teh-Chin's ROS determination step is sensitive to digitization effects, whereas in this dataset, the noise applied to these images is quite severe, distorting their shapes. The results shown in Table 2.2 also reveal that line thickness could be a good feature to estimate local scales.

Table 2.2: Comparison of the dominant point detection rates for three scenarios.

Dominant point detection mode	Repeatability Score
With ROS of the proposed method	**67.5** %
With ROS based on line thickness	44.3 %
With ROS proposed by Teh-Chin	21.3 %

We also performed an additional experiment to study the impact of the two parameters

E_{min} and E_{max} in the stage of ROS determination. For this purpose, we vary the values of E_{min} and E_{max}, and compute the repeatability score of the proposed system for the *setC* up to the stage of dominant point detection. The obtained results are presented in Figure 2.17. These results show that the detection rate is quite stable (e.g., varying in the range of $[63, 68]$) given various settings of E_{min} and E_{max}. This is expected as It is noted that the proposed system achieves the repeatability score of 67.5% in Table 2.2 with respect to the following setting: $E_{min} = 1.3$ and $E_{max} = 1.8$.

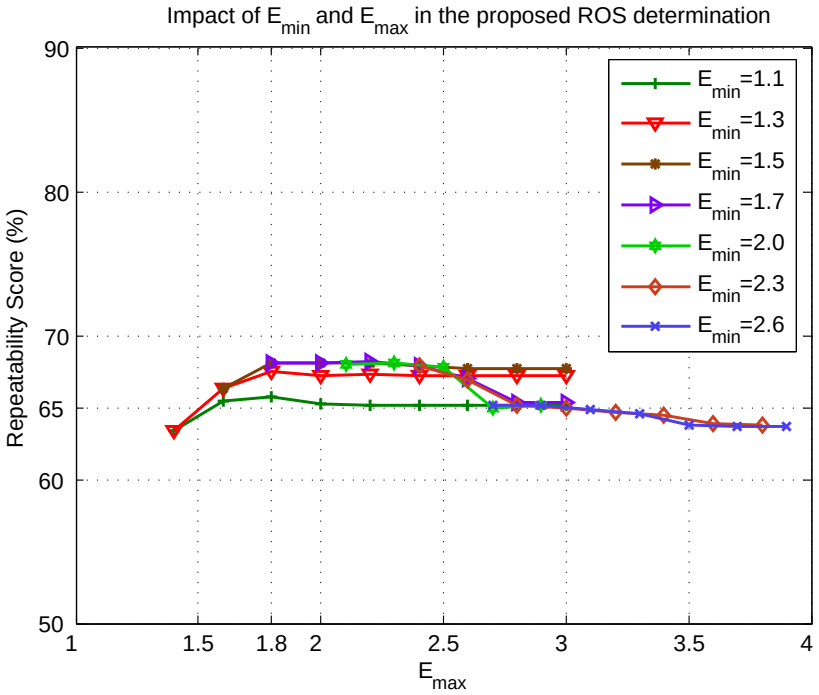


Figure 2.17: Impact of the parameters E_{min} and E_{max}: the repeatability score is computed on the *setC*.

For time complexity evaluation, we provide in Table 2.3 some information about the processing time (excluding the pre-process step) of three systems applied on several images with different sizes. The processing time has been recorded on our specific computer configuration: Intel(R) Core(TM) i5 CPU 2.4 GHz, RAM 2.4 GB, Windows 8.

Table 2.3: Report of the processing time (ms) and the number of detected junctions (in brackets).

System	Image size (Width × Height)		
	900 × 984	3600 × 3938	2100 × 4433
Our system	**16.0 (67)**	**187.0 (79)**	**140.0 (105)**
Hilaire et al.	110.0 (89)	297.0 (147)	265.0 (146)
Liu et al.	563.0 (82)	14953.0 (157)	4078.0 (174)

In general, the system of Liu et al. is subjected to a high computation load because distorted skeleton correction using *Criterion A* is very time-consuming. The system of Hilaire et al. seems to provide a reasonable level of processing time because the criterion

to merge two discrete primitives is somewhat similar in spirit to *Criterion A* but with the elimination of much of the redundant computation. Our system works most efficiently, not only for the cases of the several images reported in Table 2.3 but also throughout the extensive experiments we have performed. It is also noted that the number of detected junctions (in brackets) provided by our system is much smaller than those outputted by the other systems. We provided few illustrative examples of the detected junctions of our approach applied to different kinds of images as shown in Figures {2.18, 2.19, 2.20, 2.21}.

2.5 Discussion

This chapter presents a new approach for junction detection and characterization in line-drawing images. The main contribution of this work is three-fold. First, a new algorithm for the determination of the region of support is presented using the linear least squares technique. The crossing-points, in combination with the dominant points detected from median lines, are treated as candidate junctions. Next, using these candidate junctions, an efficient algorithm is proposed to detect and conceptually remove all distorted zones, retaining reliable median line segments only. These line segments are then locally characterized to construct the topological representations of the crossing zones. Finally, a novel junction optimization algorithm is presented, yielding accurate junction localization and characterization. The proposed approach is extremely robust to common geometry transformations and can resist a satisfactory level of noise/degradation. Furthermore, it works very efficiently in terms of time complexity and requires no prior knowledge of the document content. The proposed method is also independent on any vectorization systems. All of these prominent features of the proposed approach have been validated relative to other baseline methods by our extensive experiments.

In addition to these advantages, the proposed approach has several shortcomings. First, as this approach is dedicated to working with line-like primitives, its performance would be degraded if applied to filled-shape objects, such as logo images. In addition, the junction optimization process could lead to some difficulties in correctly interpreting the junction position as originally produced by craftsmen. However, although this point is valid for some specific domains of exact line-drawing representation, such as vectorization, we are interested in detecting local features that would be useful to addressing the problem of large-scale document indexing and retrieval. In this sense, a low rate of false positives in the final results is not problematic. A last noticeable point is that the detected junctions could be used in combination with additional features (e.g., end-points, *isolated* straight lines, arcs, and circles) to obtain the complete representation of a graphical document image. The detected junctions can be also used to address the problem of vectorizing the line-drawings.

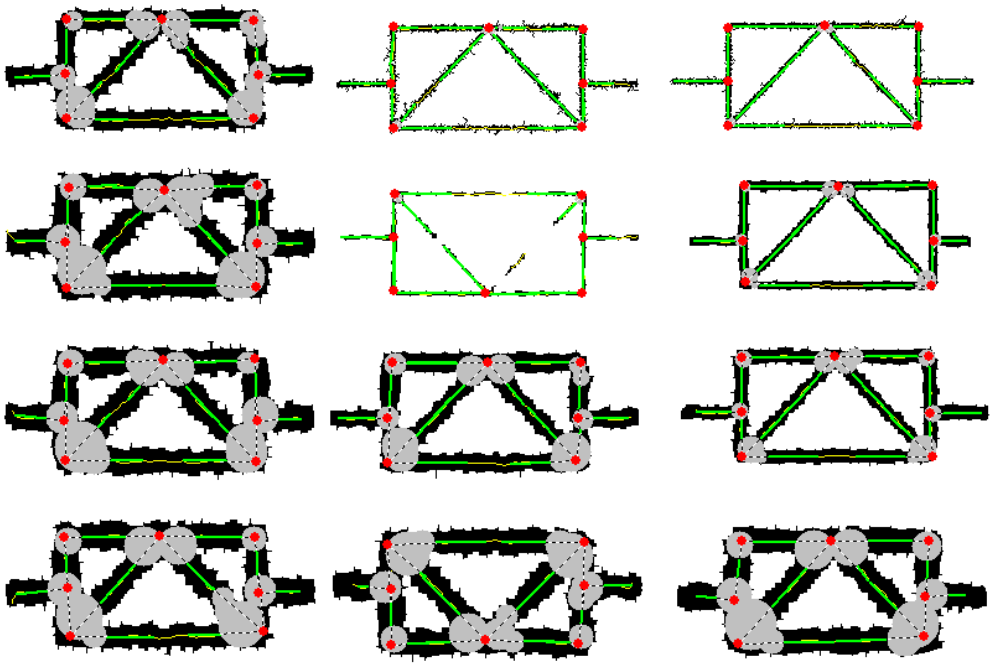


Figure 2.18: Detected junctions for a synthetic symbol with different levels of noise.

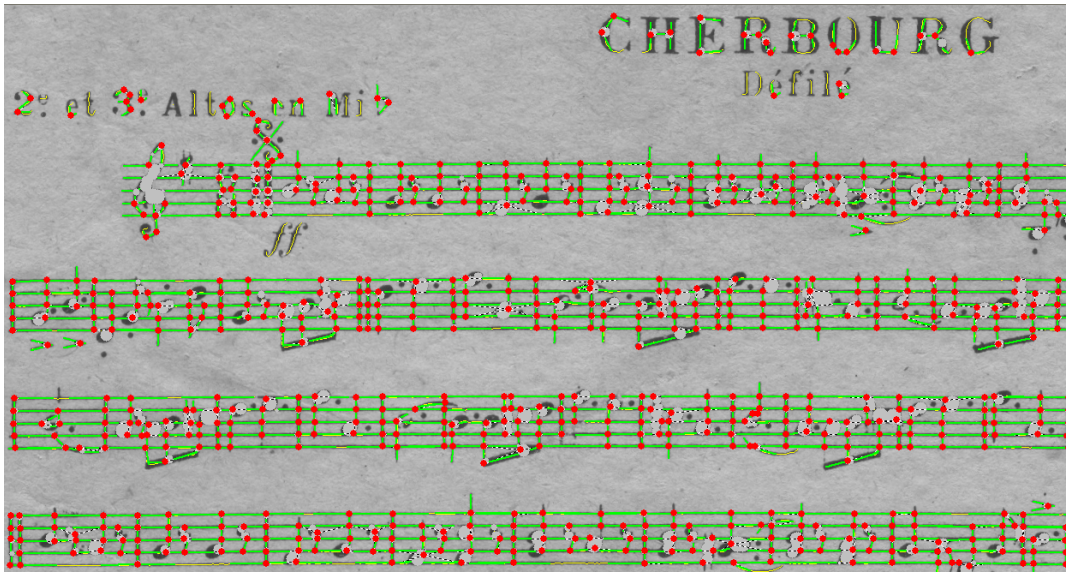


Figure 2.19: Detected junctions for a real musical score image.

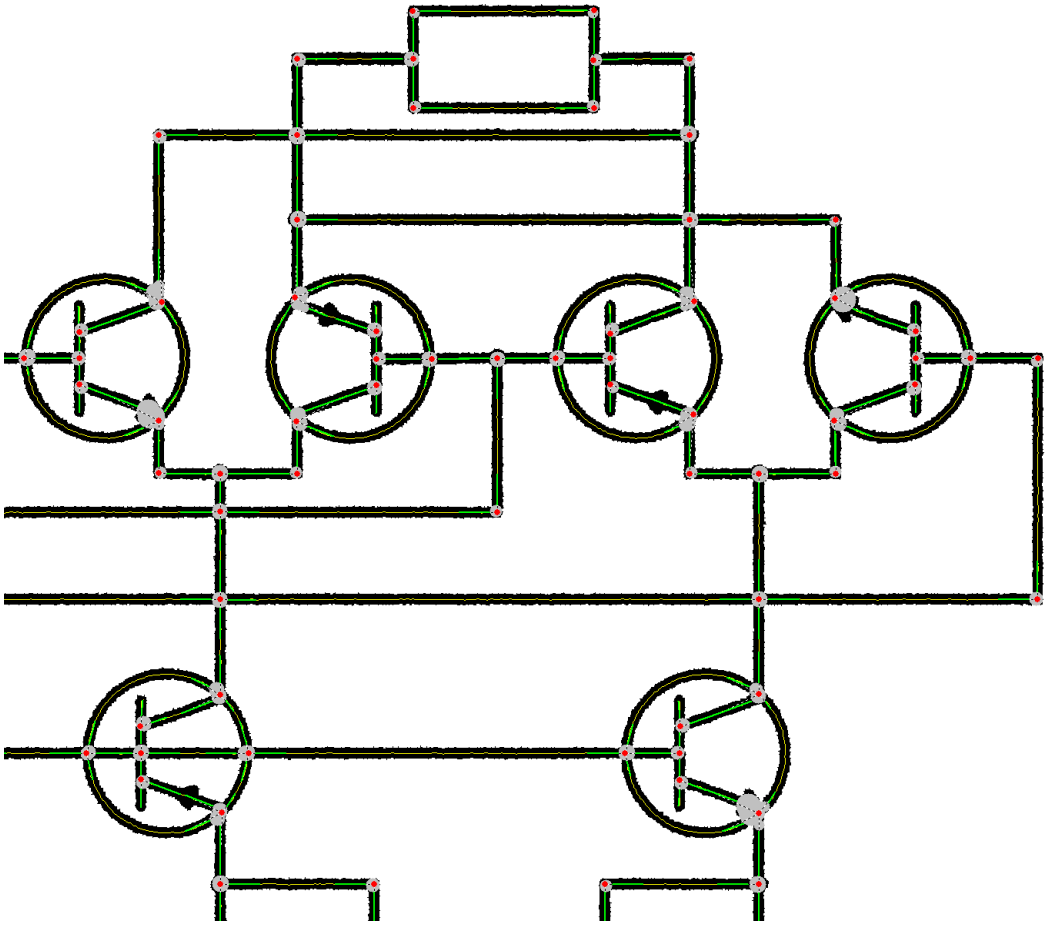


Figure 2.20: Detected junctions for part of an electronical image.

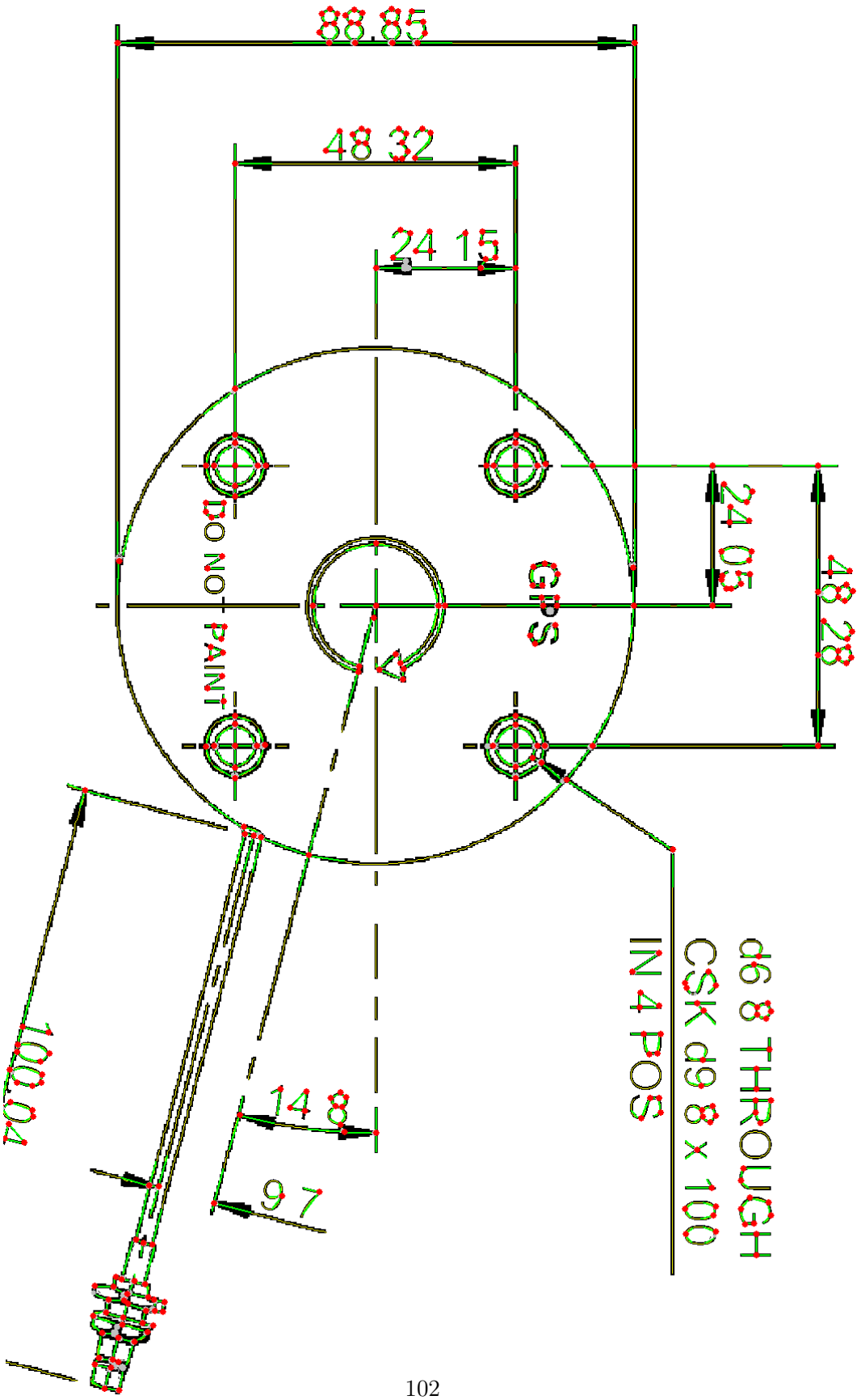


Figure 2.21: Detected junctions for a mechanical-text image.

Chapter 3

Application to symbol localization

In this chapter, an application of symbol localization in line-drawing images is developed to demonstrate that our junction detector is robust and discriminated enough to be used in the context of object localization. The experimental results, applied to several public datasets, show that our system is very time- and memory-efficient. Our precision and recall results in terms of symbol localization highlight that we outperform other methods in the literature on this problem.

3.1 Introduction

A common problem of any symbol processing systems, recognition or spotting, is localization or detection of the symbols. Symbol localization can be defined as the ability of a system to localize the symbol entities in the complete documents. It could be embedded in the recognition/spotting method or works as a separated stage in a two-step system [Qureshi et al., 2008]. The approaches used for localization are similar for recognition and spotting. All systems rely first on a primitive extraction step (e.g., connected components, loops, key-points, lines, etc.). These systems differ mainly in the way that the detected primitives are processed, using machine learning or retrieval and indexing techniques. Different approaches have been investigated in the literature to deal with the localization problem.

One of the earliest approach employed in many systems is subgraph matching. Graph is a very effective tool to represent line drawings. Attributed Relational Graphs (ARGs) can be used to describe the primitives, their associated attributes and interconnections. However, subgraph isomorphism is known to be a NP-hard problem, making it difficult

to use the graph for large images and document collections, despite the approximate solutions of subgraph isomorphism developed in the literature [Messmer and Bunke, 1996, Bodic et al., 2009]. In addition, subgraph isomorphism remains very sensitive to the robustness of the feature extraction step, as any wrong detection can result in strong distortions in the ARGs.

An alternative approach to subgraph matching is the use of a "triggering" mechanism. Such a system looks for some specific primitives in line drawing images and triggers a matching process at the symbol level within the Regions of Interest (ROIs) around these primitives. The system in [Nguyen et al., 2009] is a typical example. In this work, given a query symbol, the keypoints (i.e., Difference of Gaussian features) and its corresponding vocabularies are computed and used to find the matched keypoints from the database documents. For each pair of two matched keypoints, the local scale and orientation extracted at the keypoint in the query symbol are used to generate the ROI in the document that probably contains the instance of the symbol. Because the number of detected keypoints would be very large and the local scale computed at each keypoint could be far from satisfaction, the ROI extraction step is thus fragile and time-consuming. Triggering mechanisms have been also developed from graph-based representations, as in [Rusinol and LLados, 2006, Qureshi et al., 2008]. These proposed systems work from the ARGs, where the structures and attributes of the graphs are exploited to identify the ROIs without recognizing the symbols. Triggering-based localization is very sensitive to robustness of the mechanism in that any missed detection at the triggering level will result in the failure of symbol localization.

In the other place, a different approach to deal with object localization is framing [Dosch and LLados, 2004, Kong et al., 2011, Dutta et al., 2011]. These techniques involve the decomposition of the image into frames (i.e., tiles, buckets, windows) in which the frames could be overlapped [Kong et al., 2011] or disjointed [Dosch and LLados, 2004]. Local signatures are computed from the primitives contained in the frames and matched to identify the candidate symbols. The size of the frames can be determined based on the symbol models [Dosch and LLados, 2004, Kong et al., 2011] or set at different resolutions [Dutta et al., 2011]. In this way, framing is not scale invariant as the size of the frames cannot be dynamically adapted. The position of the frames can be set with a grid [Dosch and LLados, 2004, Dutta et al., 2011] or by sliding [Kong et al., 2011]. Sliding could be performed by steps to reduce the entire processing time [Kong et al., 2011], as any computations with overlapping would be subjected to a polynomial complexity.

A part from the before-mentioned approaches, a recent framework for symbol localization is the use of geometry consistency checking as presented in [Nayef and Breuel, 2011, Jain and Doermann, 2012, Rusinol et al., 2013]. Such a method concerns a pipeline of decomposing a graphical document into a set of primitives, matching the primitives of the model against the test image, and checking the geometry constraints among the matches. Different techniques for geometric verification have been applied in these works. [Nayef and Breuel, 2011] proposed to use a branch and bound algorithm to search for a transformation that maps a maximal subset of the matches between the primitives of the model and the test images. [Rusinol et al., 2013] employed a classical technique RANSAC [Fischler and Bolles, 1981] to achieve this goal. [Jain and Doermann, 2012] incorporated the use of orientation information of the detected features to prune the matches, following

the verification step of pair-wise angles between every two triangles in the model and test images.

The common drawback of all these methods is the computation complexity of the geometric verification process. In this application, we mainly aim at demonstrating that our contribution on junction detection can be used for symbol localization by incorporating some addition processing steps. In that concern, we show that (1) the detected junctions are useful to deal with the problem of symbol localization, and (2) these junctions support the process of geometry verification in a very efficient way. Particularly, our system is composed of four main stages, each of which is briefly described below with respect to the Figure 3.1.

- In the first stage, the junction points are detected and characterized into different types such as T-, L-, and X-junctions.
- The second stage decomposes a document image into a set of smooth primitives, composing of isolated shapes (e.g., isolated circles and straight lines) and curve segments bounded between either two junctions or a junction and an end-point. These primitives are then associated with a new set of keypoints including Line-, Arc-, and Circle-keypoints. The obtained keypoints, in combination with the junction points and end-points, form a complete and compact representation of graphical documents.
- In the third stage, keypoint matching is performed to find the correspondences among the keypoints of the query and those of database documents.
- Finally, geometry consistency checking is applied to the obtained matches using a new and efficient algorithm, which is designed to work on our specific keypoints.

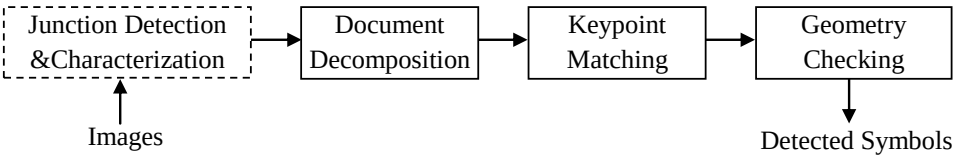


Figure 3.1: Overview of our symbol localization system.

As the first stage is accomplished by simply applying our contribution as presented in chapter 2, it will not be detailed in the following sections. Instead, we directly describe hereafter the last three stages.

3.2 Document decomposition

We use the detected junctions to decompose a document image into a set of *smooth primitives*. Here, we define the smooth primitives as those composing of isolated shapes (e.g., isolated circles and straight lines) and curve segments bounded between either two junctions or a junction and an end-point. This definition is derived based on the fact that after the process of junction detection, every median line segment bounded between two

junctions are sufficiently smooth. Otherwise, some new junction points are likely to be detected on this segment. In this work, we restrict the smooth primitives to three kinds of segment: straight line segment, arc segment, and circle. These basic-shape primitives could be derived using the linear least squares (LLS) fitting technique as follows (see Figure 3.2):

- For each smooth primitive P, we try first to fit P to a straight line segment by comparing the average distance error to a fixed threshold (e.g., 1.5 pixels in our implementation). The distance error is simply computed as the Euclidean distance from each point of P to the fitted straight line.
- If P is not fitted to a straight line, circle fitting will be performed next. This is also accomplished by comparing the average distance error to a threshold but the distance error is now computed as the Euclidean distance from each point of P to the fitted circle.
- If P is fitted to a circle, it will be further classified as an arc primitive (e.g., opened curve) or a circle (e.g., closed curve).

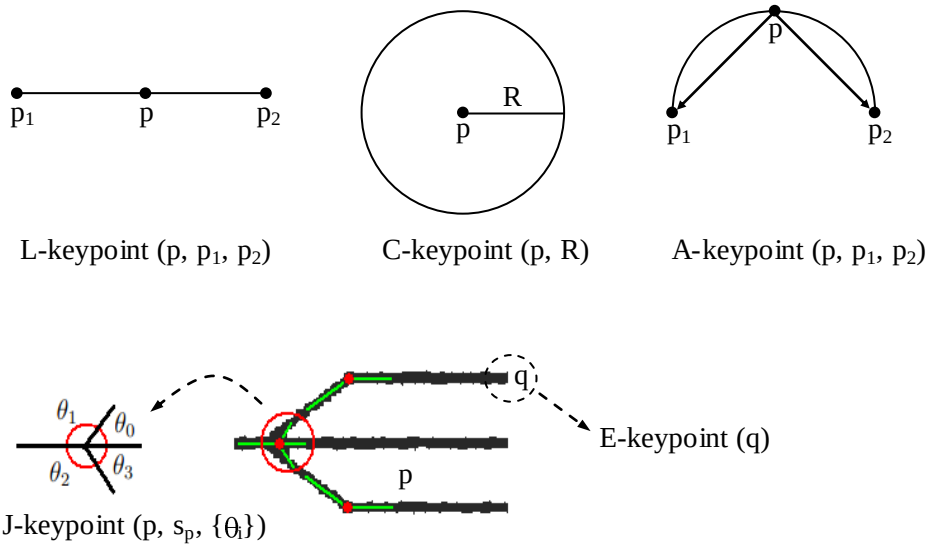


Figure 3.2: Basic primitive decomposition and description.

Next, each type of these primitives is characterized as a specific structural keypoint as follows:

- A straight line primitive is represented by a triple $\{p_L, p_{L_1}, p_{L_2}\}$ corresponding to the middle point and two extremity points, respectively. A triple $\{p_L, p_{L_1}, p_{L_2}\}$ is regarded as a Line-type keypoint or *L-keypoint*.
- An arc primitive is represented by $\{p_A, p_{A_1}, p_{A_2}\}$ with the same meaning as that of a straight line primitive. A triple $\{p_A, p_{A_1}, p_{A_2}\}$ is regarded as an Arc-type keypoint or *A-keypoint*. It is noted that the characterization of an A-keypoint is proceeded in the same spirit as that of a junction whose two arms are $p_A p_{A_1}$ and $p_A p_{A_2}$.

- A circle primitive is represented by $\{p_C, r_C\}$ corresponding to its centroid and radius. A couple $\{p_C, r_C\}$ is regarded as a Circle-type keypoint or *C-keypoint*.

For completeness, we call the junction points as J-keypoints and end-points as E-keypoints. The description of a J-keypoint is performed using the same process of junction characterization, while the description of the E-keypoints is no needed. To this end, a document image is completely represented by a set of structural keypoints, composing of L-keypoints, A-keypoints, C-keypoints, J-keypoints, and E-keypoints. Figure 3.3 shows a decomposition of a document image into a set of structural keypoints.

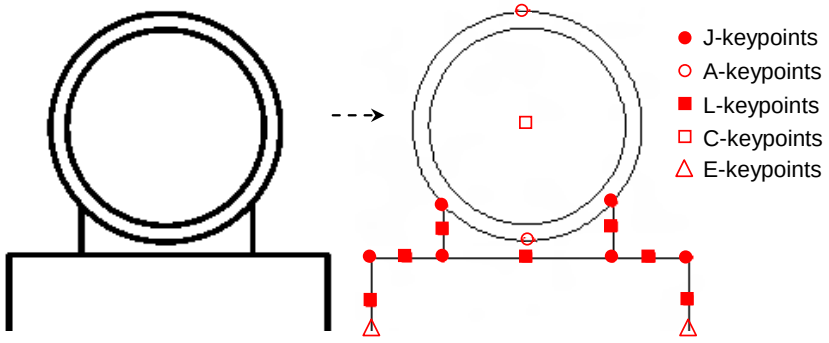


Figure 3.3: Keypoint-based representation of a simple document image.

3.3 Keypoint matching

In the previous stage of document decomposition, a document image is completely represented by a set of structural keypoints. In this stage, keypoint matching will be performed to establish the correspondences between the keypoints of the query Q and those of the database document D. Keypoint matching is independently processed for each type of keypoint. Particularly, matching of the L-, A-, C-, and E-keypoints is proceeded very simply as outlined below:

- An E-keypoint (*resp.* C-keypoint) is always matched with any other E-keypoints (*resp.* C-keypoints).

- A L-keypoint is matched with another L-keypoint if the two extremity points of one L-keypoint are matched with those of the other L-keypoint (Figure 3.4 (a)). It is noted that the extremity points of one L-keypoint could be the E-keypoints and J-keypoints. Therefore, matching of two extremity points is also performed in the same manner as keypoint matching.

- An A-keypoint is matched with another A-keypoint if the difference in direction of the two keypoints is not significant and the two extremity points of one A-keypoint are matched with those of the other A-keypoint (Figure 3.4 (b)).

The matching process of the J-keypoints is often performed by pair-wise matching between the angles of the two corresponding junction points [Xia, 2011]. Particularly, given two J-keypoints characterized as $\{p, s_p, \{\theta_i^p\}_{i=0}^{m_p-1}\}$ and $\{q, s_q, \{\theta_j^q\}_{j=0}^{m_q-1}\}$, the information of junction location and junction scale is used to quickly refine the matches to be described later, and the rest is used to compute a similarity score, $C(p, q)$, of matching two junctions p and q as follows:

$$C(p,q) = \max_{i,j}\{\frac{1}{H}\sum_{k=0}^{h-1} D(\theta^p_{(i+k) \bmod m_p}, \theta^q_{(j+k) \bmod m_q})\} \tag{3.1}$$

where $h = min(m_p, m_q)$, $H = Max(m_p, m_q)$, and

$$D(\theta_i^p, \theta_j^q) = \begin{cases} 1, & \text{if } |\theta_i^p - \theta_j^q| \le \theta_{thres} \quad (3.2) \\ 0, & \text{for otherwise.} \quad (3.3) \end{cases}$$

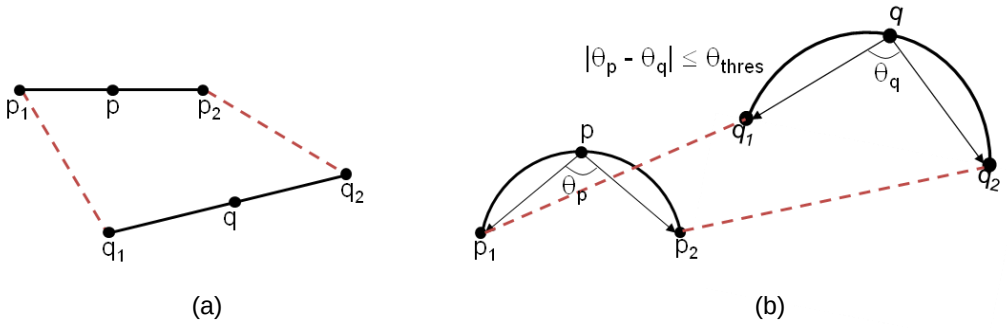


Figure 3.4: The matching process of L-keypoints (a) and A-keypoints (b).

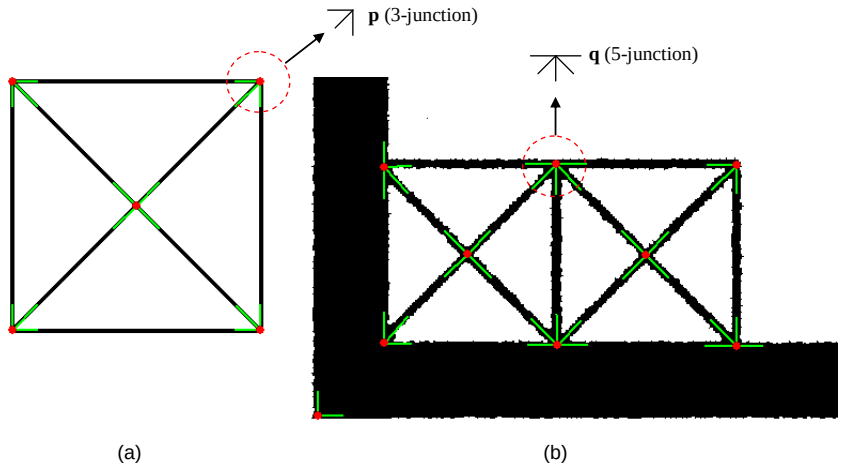


Figure 3.5: An example of context distortion of the detected junctions: a 3-junction **p** in (a) is distorted as a 5-junction **q** in (b).

The similarity score $C(p, q)$ is in the range $[0, 1]$ and θ_{thres} is an angle difference tolerance. Most of the time, two J-keypoints are matched if their similar score is higher than

a threshold. However, in some specific domains taking object localization for example, a query object or symbol is often connected to different context information appearing in a document. Figure 3.5 provides an example where the two instances of the query symbol are touched resulting in some context distortion of the detected junctions (e.g., a 3-junction **p** is distorted as a 5-junction **q**). In such cases, using the similarity score $C(p, q)$ could be too restricted to find corresponding junctions.

We therefore relax the junction matching step by introducing a new constraint as follows. Two J-keypoints p and q are matched if an *inclusion test* is hold for these two junctions. Here, we consider that p is included in q if there are exact $m_p - 1$ angle matches between the angles of p and q. This implies: $C(p, q) * Max(m_p, m_q) = m_p - 1$.

3.4 Geometry consistency checking

Given a query symbol Q and a database document D, their keypoints are first detected and matched as described in the previous sections. The obtained matches are finally verified by checking geometry consistency. This step will remove false matches and cluster the remaining matches into different clusters, each of which indicates an instance of the query symbol. Concerning this problem of geometry consistency checking, two main strategies are often exploited in the literature. A brief review of these two strategies is given in the following.

The first strategy treats data (i.e., the matches) in a top-down way. One typical technique belonging to this strategy is known as RANSAC (RANdom Sample Consensus) [Fischler and Bolles, 1981]. The key idea of RANSAC is to randomly select k matches for estimating a transformation model (typically an affine transformation and thus $k = 2$ or 3). The model is then assigned with a confidence factor, which is calculated as the number of matches fitting well to this model. Next, these steps are repeated a number of times to find the model with the highest confidence. RANSAC is often used to find *a single transformation model* between two images with a high degree of accuracy of the derived parameters, provided that the ratio of inliers and outliers occurring in the data is sufficiently high ($\geq 50\%$). However, when this is not the case, it is difficult to use RANSAC. In addition, RANSAC would be time-consuming because the number of iterations is often large to ensure that an optimal solution could be found. Precisely, the computation complexity is $O(MN)$ where M is the number of iterations and N is the size of the data. A family of advanced algorithms based on the RANSAC technique is reported in [Choi et al., 2009] where the accuracy and robustness are thoroughly investigated. The computation complexity of these algorithms is also evaluated as the trade-off between accuracy and robustness.

The second strategy treats data in a bottom-up manner by performing a voting process starting from all data points, and then finding the parameters (typically composing of 4 parameters: orientation, scaling, and x, y-translation) corresponding to the dense density areas of support. One typical technique falling this strategy is known as Generalized Hough Transform (GHT) [Ballard, 1981]. GHT is most commonly used for the cases where *multiple transformation models* are presence in the data. It is less accurate than RANSAC in terms of parameter estimation but very robust to noise even if a large number of outliers are presence. However, because GHT requires a process of parameter quantization, it is

subjected to very high cost of memory space $O(M^4)$ and processing time $O(N^2)$[1], and is sensitive to the quantization of the parameters. A comparative evaluation of the GHT techniques can be found in [Kassim et al., 1999]. In this work, different extensions of the GHT method are presented to reduce the memory requirement and computational complexity.

In our case, as each keypoint of the query symbol is often occurred in a database document with a high frequency, the outliers are thus significantly higher than the inlier matches. In addition, as multiple instances of a query symbol can be appeared in a database document, it is therefore not a good idea to use some techniques like RANSAC or GHT because of the aforementioned weaknesses. We therefore present, below, an efficient algorithm to deal with the problem of geometry consistency checking. The proposed algorithm incorporates the advantages of both RANSAC and GHT while avoiding their weaknesses. It exploits the information extracted from the matches of L- and A-keypoints to speed up the process of estimating the affine transformation models.

Algorithm 1 Estimation of the affine transformation models

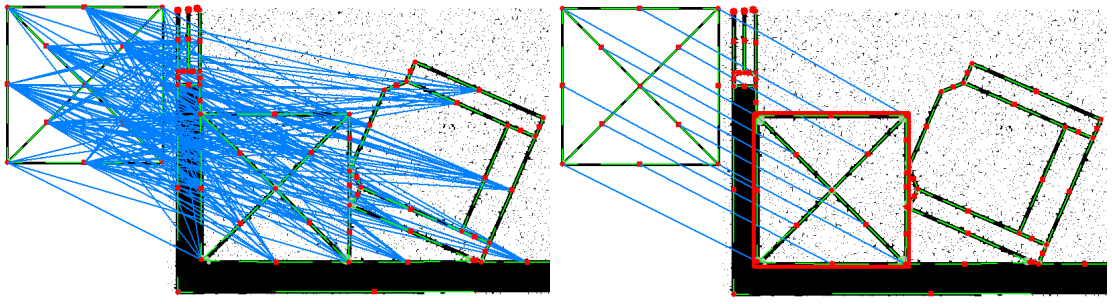
Input: Two set of structural keypoints of a query Q and a document image D, and a match list T between the keypoints of Q and D
Output: A set of affine transformation models (F_{out})
$K \leftarrow$ sort the L- and A-keypoints of Q in descending order of the primitive's length
$m \leftarrow |K|$
$i \leftarrow 1$
$F_{out} \leftarrow \emptyset$
while $i \leq m/2$ **do**
  $p \leftarrow K_i$
  for each match in T bwteen $p \in Q$ and $q \in D$ **do**
    $F \leftarrow$ solve the two linear equations formed by the extremity points of p and q
    $n_F \leftarrow$ count the matches fitting to F
    if $n_F > n_{thres}$ **and** $F \notin F_{out}$ **then**
      $F_{out} \leftarrow F_{out} + \{F\}$
    end if
  end for
  $i \leftarrow i + 1$
end while

Our method of geometry consistency checking is outlined in Algorithm 1. The basic idea is to directly estimate a geometry model F (i.e., an affine transformation) based on every match formed by a pair of either two L-keypoints or two A-keypoints. This idea is inspired by the fact that a pair of two matched lines (or arcs) provides us with 4 parameters (i.e., orientation, scaling, and x, y-translation) of an affine transformation F. As a result, we need only one match to estimate a model F other than two matches as the cases of GHT and RANSAC. Next, we apply the transformation F to all the keypoints detected on the query Q, resulting in a new set of projected points on the database document D. If there is a sufficiently large overlap among the projected points and the keypoints of D matched

[1] N and M, are the number of matches and sampling bins, respectively.

with those of Q, the transformation F is accepted and used to localize the position of the corresponding instance of Q on D. Here, we consider two points are overlapped if the distance between them is less than a threshold. Alternatively, we can consider two points are overlapped if they are positioned in a local window of the size $\epsilon_{dist} \times \epsilon_{dist}$. This can be efficiently done by using a 2D lookup table. The memory complexity is thus linear to the image size. It was found empirically that the parameter $\epsilon_{dist} \in [10, 20]$ is a common setting. It is also noted that we can set ϵ_{dist} to the minimum distance between two keypoints of the query document Q projected on the database document D to ensure no mismatch in our geometry consistency checking step.

Regard to the computation complexity, as the number of the L- and A-keypoints of Q is quite small, and few real computations are needed, the proposed method is very time-efficient. Particularly, the computation complexity of the proposed method is limited up to a linear order $O(\epsilon_{dist}^2 N_1 N_2)$, where N_1 is the number of keypoints of Q, and N_2 is the number of matches corresponding to the L- and A-keypoints of Q. In addition, with a bit prior knowledge of the dataset, we can quickly prune a large number of matches by setting the lower and upper scales for the query symbol. In this way, the local scales associated to the keypoints are used to prune the matches. It is also noted that there is no need to process all the L- and A-keypoints of Q. In our experiments, we first sort m L- and A-keypoints of Q in a decreasing order of their length and then choose the first $m/2$ keypoints to be processed. Since the primitives having higher length would be less distorted by the transformation, the estimation of the parameters is thus more robust. For each accepted model F, we obtain an instance of the query. Therefore, multiple instances of the query are thus successfully detected with respect to the number of accepted models. Figure 3.6 demonstrates the result of applying this step of geometry consistency checking.



(a) (b)

Figure 3.6: Geometry consistency checking: (a) the matches before checking; (b) the matches after checking.

3.5 Experimental Results

For performance evaluation of the proposed approach, we selected the latest dataset for symbol spotting in GREC2011[2] [Valveny et al., 2011]. The detail of this dataset is

[2]http://iapr-tc10.univ-lr.fr/index.php/final-test-description

described on Table 3.1. For performance evaluation, we selected the evaluation metric in [Rusinol and Llados, 2009b] in order to make the comparison with other methods [Dutta et al., 2013a, Dutta et al., 2013b]. This metric includes precision (P), recall (R) and Fscore computed as follows.

$$P = \frac{S_{Int}}{S_{Ret}},\ R = \frac{S_{Int}}{S_{GT}},\ Fscore = 2 \cdot \frac{P \cdot R}{P + R}$$

Where S_{Int} is the sum of intersection areas between the bounding boxes retrieved by the spotting system and ground-truth, S_{Ret} is the sum of areas of the bounding boxes retrieved by the spotting system, and S_{GT} is the sum of areas of the bounding boxes in ground-truth. It is worth mentioning that each bounding box (B_{gt}) in ground-truth is counted at most once. Typically, B_{gt} will be marked as already considered if there exists a bounding box (B_{ret}) retrieved by the spotting system such that the ratio of the intersection of their areas to the union of their areas exceeds a given threshold (e.g., 75% in our experiments). This is needed to avoid biased scores caused by multiple detections of a same symbol at a same location. Furthermore, we have set a strict constraint for our system in that the ratio of overlapping area to union area of any two retrieved bounding boxes is always less than a threshold (10% in our experiments).

Table 3.1: Dataset used for symbol spotting in GREC2011.

Test Set	Models	Images	Queries	Symbols	Noise	Image Size
Elec1.	21	20	118	246	Ideal	Min:1700x1600
Elec2.	21	20	127	274	Level 1	
Elec3.	21	20	114	237	Level 2	Max:4400x2100
Elec4.	21	20	156	322	Level 3	
Archi1.	16	20	247	633	Ideal	Min:2300x2500
Archi2.	16	20	245	597	Level 1	
Archi3.	16	20	245	561	Level 2	Max:5400x2900
Archi4.	16	20	249	593	Level 3	

Table 3.2: Experimental results of our system (%).

Test Set	Precision	Recall	F-Score	Max$\frac{Overlap}{Union}$	Mean Time (ms)
Elec1.	0.85	0.84	0.84	0.00 %	723.38
Elec2.	0.86	0.79	0.82	0.00 %	677.83
Elec3.	0.92	0.81	0.86	5.76 %	655.39
Elec4.	0.80	0.81	0.80	5.76 %	1097.05
Archi1.	0.91	0.96	0.93	6.69 %	1521.47
Archi2.	0.91	0.92	0.92	9.92 %	1341.90
Archi3.	0.94	0.89	0.92	9.92 %	1605.80
Archi4.	0.91	0.90	0.90	10.63 %	1409.88

The detailed results of our system are reported on Table 3.2 including precision, recall, Fscore, maximum ratio of overlapping area to union area of any two bounding boxes re-

trieved by our system, and mean processing time of a complete query. General speaking, the proposed system achieves quite good results for both detection and accuracy scores. On average, the Fscore(s) of the proposed system are 0.83 and 0.92 for electrical and architectural datasets, respectively. In addition, these scores are obtained under a very small overlap of the detected bounding boxes. It is noted that the results obtained on the architectural dataset are much better than those on the electrical dataset. The reason lies in the fact that in the electrical dataset, more query symbols are used and many query symbols looks very similar making them difficult to be correctly distinguished. The processing time is calculated as the mean time of the whole process (i.e., both the online and offline phases), subjected to our specific computer configuration: Intel(R) Core(TM) i5 CPU 2.4GHz, RAM 2.4 GB, Windows XP.

The last experiment is performed using the SESYD database[3] [Delalandre et al., 2010]. The detail of this dataset is summarized on Table 3.3. This time, we measure the processing time for the online phase only. That is, we compute the query time for returning the full list of the detected symbol entities given a query symbol. Table 3.4 reports the obtained results of our system. It can be seen that we obtain very good results in terms of both detection rate and recall. On average, the proposed system achieves the precision of 0.92 and the recall of 0.95, resulting in the F-score of 0.93. The mean query time is just 0.3 (s).

In order to make some comparative results, we provide on Table 3.5 the results of some recent symbol localization systems. It is clear that our system much outperforms all these baseline methods. It is worth mentioning that the system of [Dutta et al., 2013b] gives the lowest processing time (i.e., 0.7 (s)) because it was integrated with an indexing hashing-based scheme. Our system requires 0.3 (s), on average, to perform a query without using any indexing methods.

Table 3.3: The detail of the SESYD (floorplans) dataset.

Test Set	Images	Models	Symbols	Noise	Image Size
floorplans16-01	100	16	2671	None	6775×2858
floorplans16-02	100	16	2488	None	3059×3341
floorplans16-03	100	16	2661	None	2218×2475
floorplans16-04	100	16	3251	None	2056×1837
floorplans16-05	100	16	2148	None	2596×2313
floorplans16-06	100	16	2068	None	2352×2507
floorplans16-07	100	16	3898	None	5498×2961
floorplans16-08	100	16	2260	None	3026×2967
floorplans16-09	100	16	3948	None	4307×1893
floorplans16-10	100	16	2653	None	4349×2227

Figure 3.7 shows an example of our symbol localization system where the query is perfectly localized even though it is embedded into a complicated database document. There are, however, some queries as shown in Figure 3.8 that the proposed system fails to detect the symbol "outlet" because of a very limited number of keypoints detected on the instance of the symbol on the database document. Besides, we showed in Figure 3.9

[3]http://mathieu.delalandre.free.fr/projects/sesyd/symbols/floorplans.html

Table 3.4: The results of our system for the SESYD (floorplans) dataset.

Test Set	Precision	Recall	F-score	Mean time (ms)
floorplans16-01	0.97	0.95	0.96	339.5
floorplans16-02	0.89	0.98	0.93	284.2
floorplans16-03	0.87	0.91	0.89	242.3
floorplans16-04	0.93	0.95	0.94	418.6
floorplans16-05	0.92	0.98	0.95	112.3
floorplans16-06	0.90	0.97	0.93	161.1
floorplans16-07	0.96	0.97	0.96	621.2
floorplans16-08	0.94	0.95	0.94	226.8
floorplans16-09	0.90	0.98	0.94	406.8
floorplans16-10	0.88	0.81	0.84	252.3
Average	**0.92**	**0.95**	**0.93**	**306.5**

Table 3.5: Comparision of recent methods for symbol localization on the SESYD (floorplans-01) dataset.

System	Precision	Recall	F-score	Mean time (s)
Our system	0.97	0.95	0.96	0.34
[Dutta et al., 2013a]	0.62	0.95	0.74	0.57
[Dutta et al., 2013b]	0.41	0.82	0.52	0.07
[Nguyen et al., 2009]	NA	NA	0.82	NA

another example that the system correctly detects multiple instances of the symbol "sofa1" even these detections take part of a different symbol (i.e., "table2"). This further confirms the interesting results of our system.

3.6 Discussion

We have presented an application to symbol localization in line-drawing images using junction feature and geometry consistency checking. This system proves that our contribution on junction detection can be used in the context of object detection and localization by incorporating some feature extraction steps at primitive level. As the junction detectors is robust and accurate, the obtained primitives are stable under the different contexts of the documents. These primitives are used to support object matching, geometry consistency checking, and object localization. In that sense, this highlights that our detector can support and to be combined in an efficient way within a vectorization process. The experimental results in terms of symbol localization confirm the advantages of the system for both efficiency and accuracy.

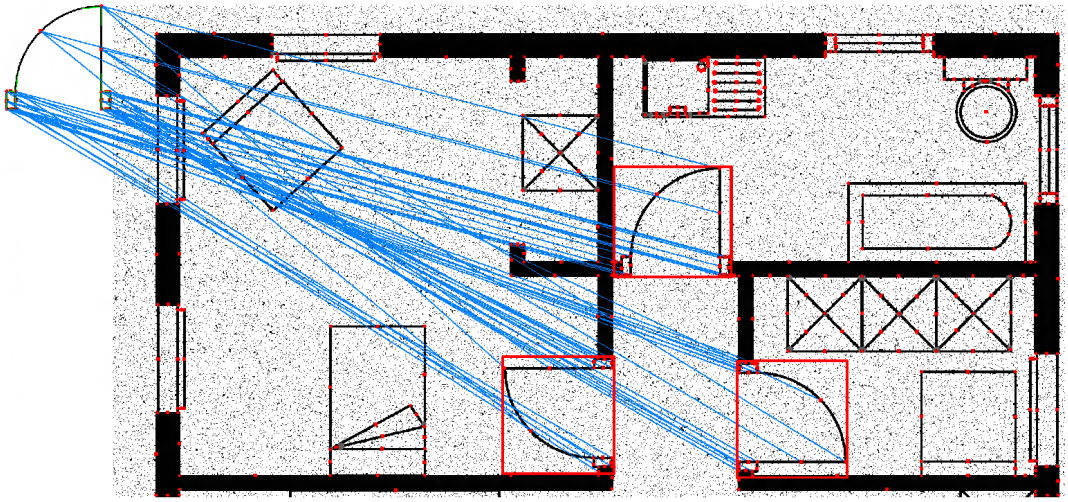


Figure 3.7: An example of symbol localization: a query symbol (left) and the detected instances of the query (red bounding boxes).

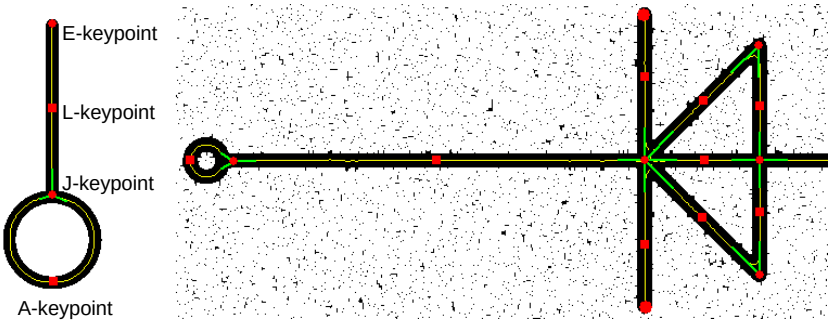


Figure 3.8: The system fails to detect the symbol "outlet" (left) due to the omission of the E-keypoint and the displacement of the L-keypoint on the instance of the symbol (right).

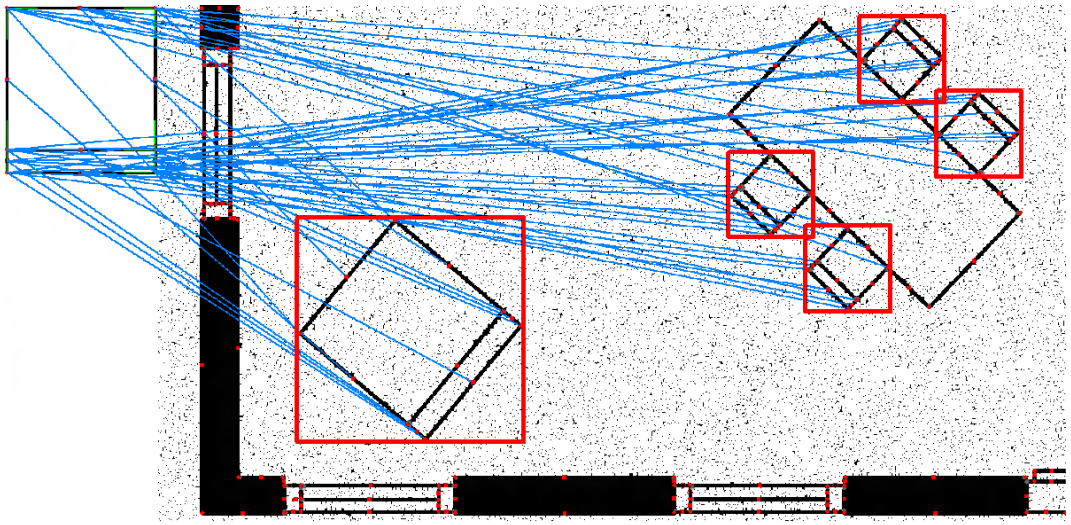


Figure 3.9: Few examples of the detected symbols of our system.

Part II

Feature indexing in high-dimensional vector space

Chapter 4

State-of-the-art in feature indexing

In the previous chapter, we have provided a local detector of junction points. As it is often the case that the detected keypoints shall be described by local descriptors followed by a further step of descriptor indexing. Hence, the next two chapters are attributed to the problem of feature indexing. This chapter reviews the state-of-the-art in feature indexing in high-dimensional feature vector space. The main ideas, favourable features and shortcomings of each method are carefully discussed. We also provide our subjective remarks for these methods and highlight the need of an advanced contribution for an efficient indexing technique.

4.1 Introduction

As we have discussed earlier, robust feature extraction is of central importance for an image processing system. Furthermore, feature indexing is of a crucial need for all the real-time image processing applications. For these reasons, the two next chapters attempt to deal with the problem of feature indexing. At first, an overview of the existing techniques is provided. Next, an attempt is made to give a new contribution in feature indexing for quickly answering the queries of proximity search. Let us first describe the general context of the fast proximity search problem. Considering a scenery where the objects are represented by real feature vectors in a feature vector space S, the problem of finding the nearest neighbor of a given query object q over a dataset X has been well-established in the literature. Usually, two problematic factors make the difficulty of this problem. First, the dataset X is often composed of very large data points (e.g., millions of feature vectors). Second, each data point is of a high-dimensional feature space (e.g., > 100). A

conventional solution is to sequentially search for each object $p \in X$, to find the closest one of q, based on some similarity distance function $d : S \times S \rightarrow \Re$. While this brute-force search is practicable in a low-dimensional and small-scale space, it is subjected to the problem of *curse of dimensionality* when working on high-dimensional and large-scale space [Yamamoto et al., 1999].

To deal with this problem, many approaches have been proposed in the literature supporting the tasks of *exact/approximate* nearest neighbor (ENN/ANN) search. These approaches are mainly concerned with designing an efficient indexing scheme. Basically, an indexing algorithm is formulated as the task of reorganizing the data so that it is able to answer quickly the query of proximity search [Beis and Lowe, 1997, Chávez et al., 2001]. For completeness, we introduce hereafter popular formulations of the ENN and ANN search in the literature.

- Exact nearest neighbor search [Gionis et al., 1999]: given a query object q and a database X, return an object $p^* \in X$ such that $\forall p \in X : d(p^*, q) \leq d(p, q)$.
- Exact K-nearest neighbor search [Gionis et al., 1999]: given a query object q and a database X, return at most K objects $\{p_k^*\}_{k=1}^K$ in X such that $\forall p \in X : d(p_1^*, q) \leq d(p_2^*, q) \leq \ldots \leq d(p_K^*, q) \leq d(p, q)$.
- Range search [Böhm et al., 2001]: given a query object q and a database X, and a distance $\epsilon > 0$, return all objects $p^* \in X$ such that $d(p^*, q) \leq \epsilon$.
- *c-approximate* nearest neighbor search (c-ANN) [Gionis et al., 1999]: given a query object q and a database X, and a parameter $c \geq 1$ and let p_{exact} be the closest object to the query q, return an object $p^* \in X$ such that $d(p^*, q) \leq c \cdot d(p_{exact}, q)$. The parameter c is treated as *approximate factor* or approximate tolerance. If $c = 1$, we obtain the exact nearest neighbor of q.

Several surveys of indexing algorithms in vector space are presented in [Böhm et al., 2001] and [Liu et al., 2004]. Such methods are often categorized into four classes as shown in Figure 4.1. These approaches are detailed in the following.

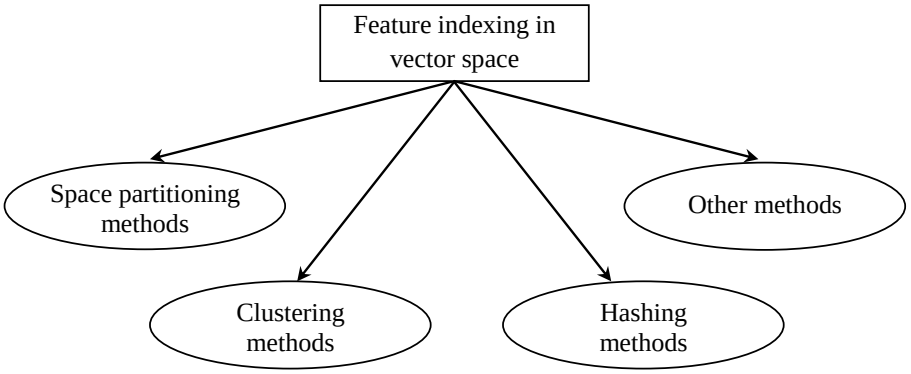


Figure 4.1: Different approaches for feature indexing in vector space.

Table 4.1: Indexing methods based on space partitioning in $\Re^k$ vector space

Method	Worst-case search complexity	Memory cost	Dataset & Feature dimension (k)	Main features
Friedman et al., 1977 (KD-tree)	$O(kn^{1-1/k})$	$O(n)$	Uniform dataset, $k \leq 20$	Static binary tree, fast tree building, ENN search, good for low-dimensional data space
Silpa-Anan and Hartley, 2008 (NKD-trees and PKD-trees)	$O(n)$	$O(nm)$	SIFT dataset, $5 \cdot 10^5$ records, $k = 128$	Static binary trees, building m randomized KD-trees, ANN search
McNames, 2001 (PA-tree)	$O(kn^{1-1/k})$	$O(n)$	Uniform dataset, 10^5 records, $k \leq 20$	Static m-ary tree, data alignment with PCA analysis at each partitioning level, high computation cost of lower bounds and tree building, ENN search
Guttman, 1984 (R-tree) Sellis et al., 1987 (R^+-tree) Beckmann et al., 1990 (R^*-tree)	$O(n)$ $O(n)$ $O(n)$	$O(n)$ $O(n \log n)$ $O(n)$	Rectangle dataset, 5000 records, $k = 2$ Point dataset, 10^5 records, $k = 2$ Rectangle dataset, 10^5 records, $k = 2$	*Dynamic* and balanced m-ary tree, supporting both data points and extended spatial data, difficulties of maintaining minimized overlap and empty space, range and ANN search
White and Jain, 1996 (SS-tree)	$O(n)$	$O(n)$	Uniform dataset, 10^5 records, $k \leq 11$ EigenFace dataset, 10^4 records, $k \leq 100$	Dynamic m-ary tree, high empty space, less overlap-free, range and ANN search
Katayama and Satoh, 1997 (SR-tree)	$O(n)$	$O(n)$	Uniform dataset, 10^5 records, $k \leq 64$	Dynamic m-ary tree, high memory space for storing both the sets of hyper-spheres and the hyper-rectangles, range and ANN search
Berchtold et al., 1996 (X-tree)	$O(n)$	$O(n)$	Uniform dataset and Fourier points, $k \leq 16$	Dynamic m-ary tree, overlap-free for point data with the use of super-nodes, more parameters involved, high memory space, range and ANN search

4.2 Space-partitioning-based methods

The main spirit of the space-partitioning-based indexing methods is that they hierarchically divide the underlying data space into sub-spaces such that the cardinality of every sub-space at the same level is roughly equivalent. The partitioning process is repeated until the cardinality of every new sub-space at the highest level is small enough. As the resulting sub-spaces form a hierarchical representation of the original data space, a tree-based structure is often chosen to represent these sub-spaces. Fast proximity search is then accomplished using various tree traversing strategies including back tracking and priority search. Table 4.1 briefly outlines the main characteristics of these methods. For each method, the main ideas, favourable features, and shortcomings are discussed. The detail of each method is given in the following.

KD-tree: KD-tree [Friedman et al., 1977] is probably argued as one of the most popular techniques for feature indexing. Its basic idea is to iteratively perform a partition of X into two roughly equal-sized subsets X_l and X_r by using a hyperplane perpendicular to one of the axes in R^k. Particularly, a partition is performed at the i^{th} dimension ($1 \leq i \leq k$) using a pivot value s_{med} to split X into two subsets X_l and X_r. The X_l contains the points whose values at the i^{th} dimension are smaller than s_{med}, while the X_r contains the rest of X. A new node is then constructed to record the split information such as the split axis and the pivot value. The pivot value s_{med} is often chosen as the median value at the i^{th} dimension of all the points contained in the underlying dataset. This process is then repeated for the two new subsets X_l and X_r until the size of each subset falls below a pre-defined threshold. The resulting tree is known as a balanced KD-tree whose leaf nodes form a full splitting space of the original data set X (Figure 4.2).

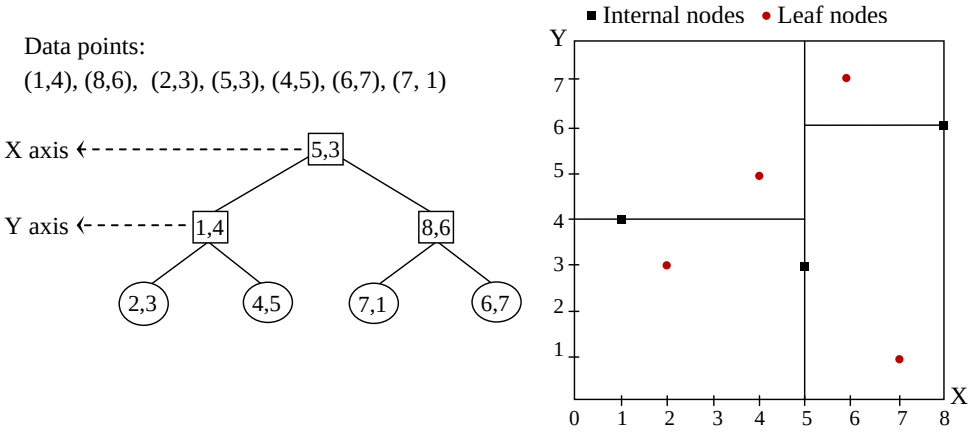


Figure 4.2: Illustration of data partitioning of the KD-tree in an 2D space: the resulting KD-tree (left) and the corresponding partitioned space (right).

Searching for a nearest neighbor of a given query point q in the KD-tree is proceeded using a branch-and-bound technique whose pruning rule works as follows: a node u is selected to explore if its hyper-rectangle *does* intersect the hyper-sphere centered at q with a radius equal to the distance of q to the nearest neighbor found so far (Figure 4.3). The most efficient technique for constructing such a KD-tree has been reported by

[Wald and Havran, 2006] with the time complexity $O(Nlog(N))$ and the worst case search time was reported as $O(kN^{1-1/k})$ in [Lee and Wong, 1977], where N is the number of nodes in the tree. The KD-tree has been shown to work efficiently when the dimensionality is low (e.g., most often $k \leq 20$), otherwise its performance is almost close to brute-force search.

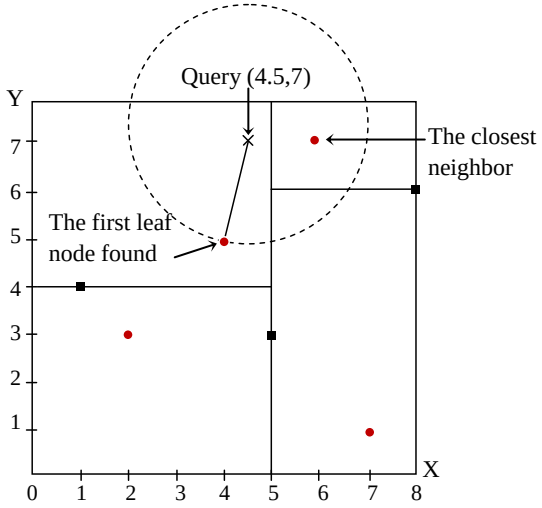


Figure 4.3: Illustration of the searching process in the KD-tree: only the regions whose bounding boxes do intersect the circle are inspected.

The KD-tree has been extended to deal with the problem of approximate nearest neighbor search by several researchers. [Beis and Lowe, 1997] introduced an extension of the KD-tree, known as *Best-Bin-First* (BBF) or *priority* search, to quickly find approximate nearest neighbors of a given query point. The improvement of the BBF technique is twofold: (1) it limits the maximum number of leaf nodes to be visited; and (2) it visits the points in the order of increasing distance from the query point to the bins of splitting space. The first improvement indicates that the search procedure will be terminated early once the number of visited nodes in the tree is greater than a specific value. The second improvement guides the search procedure to the branches (or bins) which are "closer" to the query. This is accomplished by defining the distance from the query to a bin as the minimum distance to the points in the bin boundary. Their experimental results showed the efficiency of the BBF algorithm for a synthetic dataset with the dimensionality $k \in \{8, 15\}$.

NKD-trees and PKD-trees: The use of priority search was further improved in the work of [Silpa-Anan and Hartley, 2008], where the author constructed multiple KD-trees, called *NKD*-trees, with different settings of the orientation parameter. That is, the data are rotated by different angles before constructing the KD-tree. The obtained results are quite interesting. Based on that, the idea of using multiple KD-trees has been developed in two new different ways. In the first place, multiple *randomized* KD-trees (*RKD*-trees) are constructed, where each tree is built by selecting randomly, at each node, a split axis from few dimensions having the highest variance. In the second place, multiply randomized KD-trees are constructed in the same manner but the data are initially aligned to the principal axes obtained from PCA analysis. The later indexing scheme is thus called principal component KD-trees (*PKD*-trees). Experimental results showed significantly out-

standing performance compared to the use of a single KD-tree because the use of multiple randomized trees gives a better chance of reaching the true answers than using a single tree.

PA-trees: The PA-tree (*Principal Axis* tree) [McNames, 2001] extends the KD-tree at twofold. First, it constructs a bigger fanout tree by partitioning dataset at each step into n_c subsets ($n_c \geq 2$). Second, PCA is applied to the underlying dataset at each partition to select the split axis having the highest variance. Therefore, the obtained regions are treated as hyper-polygons rather than hyper-rectangles as in the KD-tree. Consequently, this approach complicates the process of computing the lower bound of the distance from the query to a node of the tree. To facilitate this matter, the author proposed a convenient solution for computing the lower bound using the cosine rule as illustrated in Figure 4.4. Let x be any point in some node of the tree, say the Region 2, and b_2 be the projection of a query point q on the common split hyper-plane between the Region 2 and the Region 1. By applying the cosine rule, the distance $d(q, x)$ is computed as follows:

$$d(q,x)^2 = d(q,b_2)^2 + d(b_2,x)^2 - 2d(q,b_2)d(b_2,x)\cos\angle qb_2x \tag{4.1}$$

As the angle $\angle qb_2x \geq 90^0$, Equation 4.1 implies: $d(q,x)^2 \geq d(q,b_2)^2 + d(b_2,x)^2$. Hence, the lower bound $d(q,x)^2$ can be recursively computed via $d(b_2,x)^2$ as follows:

$$d(q,x)^2 \leftarrow d(q,b_2)^2 + d(b_2,x)^2 \tag{4.2}$$

It is noted that each time of computing an intermediate distance, it requires $O(k)$ real computations to compute $d(q,b_2)^2$ as the PCA is applied to every level of partitioning. Despite of the heavy computations on the lower bounds, the PA-tree was showed to outperform many other indexing schemes for many datasets.

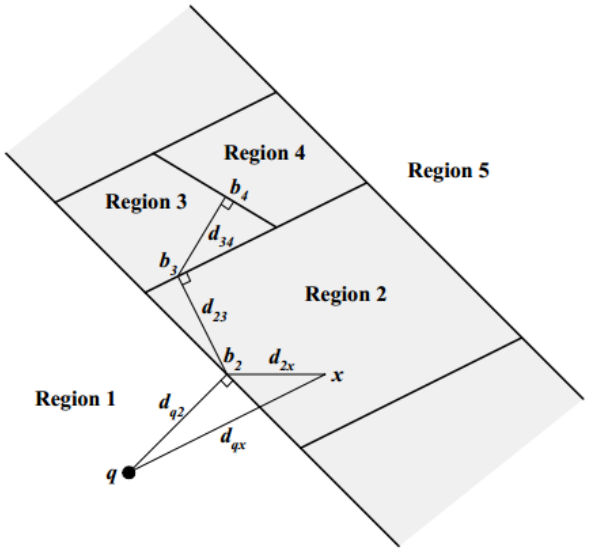


Figure 4.4: Illustration of computing the lower bound using the cosine rule of the PA-tree (reprinted from [McNames, 2001]).

R-tree: The R-tree (*Rectangle* tree) [Guttman, 1984] is *dynamically* built based on an idea that nearby points are grouped together to form an internal node represented by a minimum bounding hyper-rectangle (Figure 4.5). Each internal node of the R-tree has at most M children and the root contains at least two children. Each leaf node contains also at most M entries corresponding to data points. Inserting a new data object, p_n, into the R-tree is performed by two processes: target node searching and node splitting. The former process traverses down the tree and considers all the subtrees whose hyper-rectangles contain p_n. Tie[1] resolving is treated by choosing the child node whose bounding hyper-rectangle needs least area enlargement to include p_n, and further selecting the one with the smallest volume if necessary. This continues when reaching a leaf node. If the number of entries at this leaf node does not exceed M, the new point p_n is simply inserted to the leaf node. Otherwise (e.g., this case is called *overflow*), node splitting is performed to split $(M+1)$ points into two new nodes, subjected to an optimal criterion that the total volume of two new bounding hyper-rectangles is minimized. Then, upward change propagation is performed to update the bounding hyper-rectangles, and to split higher-level nodes, if necessary. Once the R-tree has been constructed, the *range* search algorithm starts from the root and visits every subtree whose the bounding hyper-rectangle does intersect the range query. This continues when reaching the leaf nodes, where only the entries lying inside the query range are returned as final answers. In this way, many branches of the tree are pruned, providing rapid answers of proximity queries. The main challenge of this tree is realized on the process of constructing a balanced tree, while maintaining at the same time the minimal overlap among the bounding hyper-rectangles.

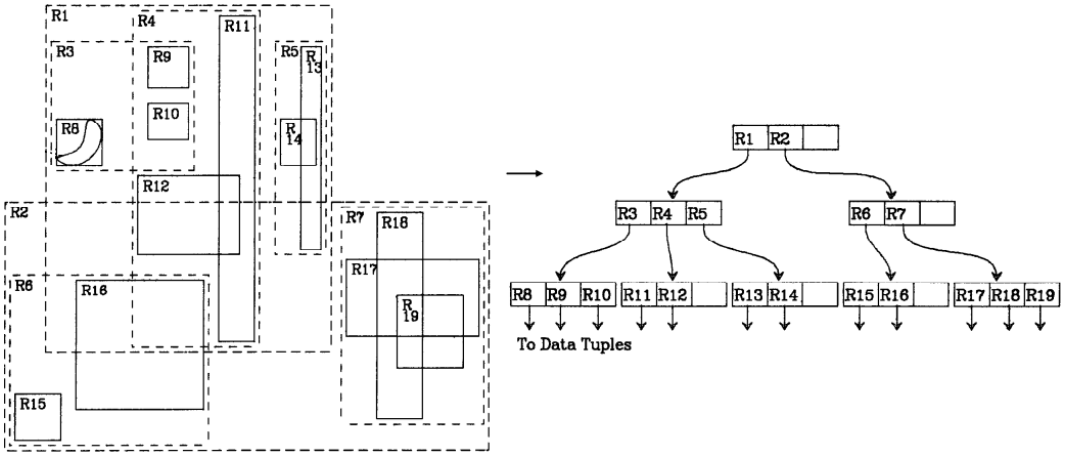


Figure 4.5: The construction of the R-tree (right) for a dataset (left); (reprinted from [Guttman, 1984]).

R^+-tree: The R^+-tree [Sellis et al., 1987] improves the R-tree in a different way. It optimizes the overlapping of internal nodes by allowing a new data object p_n to be inserted into several leaf nodes. To insert p_n into the tree, the target leaf node is first detected in a similar manner as in the R-tree. If overflow happens at the leaf node, node splitting will be performed. In this case, the leaf node is partitioned into two subsets by finding a

[1]This means that more than a single solution is happened.

cutting hyper-plane perpendicular to one of the axes, subjected to several optimal criteria, including minimal coverage of *dead* space (i.e., empty space of each new subset) and minimal number of hyper-rectangle splits. In contrast to the splitting process of the R-tree, node splitting in R^+-tree may be propagated to both upward and downward nodes. Range searching in the R^+-tree is more efficient than in the R-tree as the overlap space is taken place at the leaf nodes only. However, R^+-tree requires a logarithmically increase of memory space as the points are duplicated at the leaf nodes. The overlap is therefore shifted to the leaves, making the inspection of multiple paths at the leaf nodes necessary.

R^*-tree: The R^*-tree [Beckmann et al., 1990] is another development of the R-tree, where the main improvements are attributed to two process of searching for a target leaf node and inserting a new data object. Instead of using the sole criterion of area enlargement to determine the best path, the R^*-tree makes use of both area enlargement and overlap enlargement when searching for the best path. Particularly, given a new data object p_n to be inserted, the process of searching for the target leaf node is basically relying on that in the R-tree, but with the following modifications at each step:

- If the children of the current node, p_{curr}, point to the leaves, then choose the entry whose hyper-rectangle needs least overlap enlargement to include p_n. Tie resolving is handled by selecting the one, which needs least area enlargement, followed by the selection of the entry with the smallest hyper-rectangle volume, if needed.
- If the children of p_{curr} do not point to the leaves, then choose the entry whose hyper-rectangle needs least area enlargement to include p_n, followed by selecting the one having the smallest hyper-rectangle volume.

Then, p_n is added as a new entry of the obtained target node if its capacity is not full. If it is not the case, overflow is handled by either reinsertion or node splitting. Reinsertion is applied if the children of the target node have not been reinserted previously, and is accomplished by reinserting the k entries farthest away from the current node. If node splitting is applied, several optimal criteria are employed, including minimization of total volume of two new nodes, minimization of overlap volume of two new nodes, minimization of total surface of two new nodes, minimization of storage utilization. Upward propagation for handling overflow might be needed. Experiments showed a superiority search performance, compared to the original R-tree. However, as mentioned in [Berchtold et al., 1996], the R^*-tree performs poorly on high-dimensional space (> 5) because the overlap at the internal nodes increases rapidly with respect to the increase of data dimensionality.

SS-tree: The SS-tree (*similarity search* tree) [White and Jain, 1996] is a new variation of the R^*-tree where its nodes are represented by hyper-spheres rather than hyper-rectangles. A hyper-sphere in the SS-tree is defined by a centroid and a radius computed as the biggest distance from the centroid to all elements contained in the hyper-sphere. To insert a new object p_n into the tree, a target leaf node is first detected to contain p_n by descending the tree and choosing at each step the subtree whose centroid is closest to p_n. Next, the process of node insertion and reinsertion is quite similar to that in the R^*-tree. The only difference is in the process of node splitting, where the dimension with the highest variance is selected as the split axis, and the split location is selected such that it minimizes the sum of variances on each side of the split plane. Exper-

iments shows better results of the SS-tree in comparison with the R^*-tree. However, the problem with the SS-tree is the difficulty in yielding overlap-free of the splitting process [Böhm et al., 2001]. In addition, the use of the hyper-spheres, in general, occupies higher space than the hyper-rectangles in high-dimensional space, making the similarity search less efficient [Katayama and Satoh, 1997, Böhm et al., 2001].

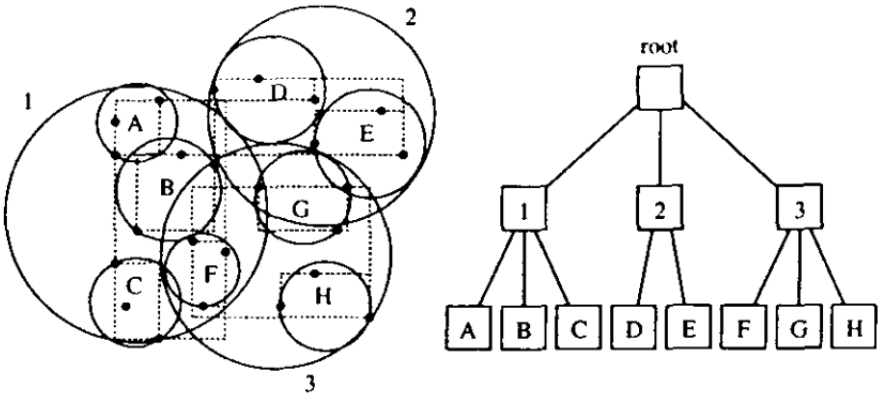


Figure 4.6: The construction of the SR-tree (reprinted from [Katayama and Satoh, 1997]).

SR-tree: The SR-tree (*Sphere/Rectangle*-tree) [Katayama and Satoh, 1997] overcomes the weakness of the SS-tree by assembling the spirit of the R^*-tree and SS-tree into an unified scheme. A node of the SR-tree is represented by the common space (R) of the hyper-sphere and the hyper-rectangle as shown in Figure 4.6. Unfortunately, the space R is not explicitly computed due to the complicated computation of the intersection between the hyper-rectangle and hyper-sphere. Instead, each node of the SR-tree records the information of both the hyper-sphere and hyper-rectangle. Given a data object q, an estimation of the distance from q to R is provided as the biggest value of the minimum distances from q to the corresponding hyper-sphere and hyper-rectangle. Next, the insertion algorithm is performed essentially similar to the SS-tree. Experimental results reported better performance of the SR-tree, compared to the SS-tree and R^*-tree.

X-tree: One common weakness of the R-tree, R^+-tree, and R^*-tree is to maintain a minimized overlap volume of the bounding hyper-rectangles when proceeding on high-dimensional space. All the heuristic solutions introduced in these works address this problem to some extent but not completely resolved [Berchtold et al., 1996, Böhm et al., 2001]. The X-tree (eXtended node tree) [Berchtold et al., 1996] gives more investigation for the optimization of overlap in the nodes of R-tree-based structures. The new investigation of the X-tree is two-fold. First, it introduces a new kind of internal node, so-called super-node. A super-node is similar to an internal node used in the R-tree-based structures except with a big capacity for containing its entries. Second, it introduces a new split procedure for optimizing the overlap of the hype-rectangles. The split procedure first tries to find an optimal split of the overflow node by using the same heuristic rules as presented in the R+-tree and R^*-tree. If the obtained overlap is still high enough, the split procedure tries to find an overlap-free split relying on split history obtained previously. If the obtained split results in unbalance nodes (*i.e.* the difference in cardinality of each new node is high enough), the

split procedure terminates without any available splits. In this case, the current node is extended to a super-node. The super-nodes again can be extended by one addition block if no available splits are found. Experiments showed the efficiency of the X-tree compared to R^*-tree and TV-tree by up to two orders of magnitude in higher-dimensional space.

4.3 Clustering-based methods

The clustering-based indexing methods differ from the space-partitioning-based methods mainly in the step of tree construction. Instead of dividing the data using a hyper-plane, these methods employ a clustering method such as K-means and K-medoids to iteratively partition the underlying data into sub-clusters. The partitioning process is repeated until the size of every sub-cluster falls below a threshold. A tree-based structure is constructed to hierarchically represent the resulting sub-clusters at all levels of decomposition. Proximity search is often handled using a branch-and-bound algorithm. Table 4.2 briefly outlines the main characteristics of these methods.

K-means clustering tree: One of the first clustering-based trees was reported by [Fukunaga and Narendra, 1975]. The proposed algorithm recursively divides all points in the dataset into smaller regions using the K-means clustering technique, and constructs a corresponding clustering tree. Each node p of the tree has the following parameters: $\{S_p, M_p, N_p, r_p\}$ corresponding to the set of data points contained in p, the cluster center, the number of data points, and the farthest distance from M_p to an $X_i \in S_p$, respectively. The iterative clustering process terminates when the size of each obtained region falls below a threshold. Searching for k-nearest neighbors of a given query q is then proceeded by a branch-and-bound algorithm. Let Y be the current nearest neighbor of q, two following pruning rules are used to eliminate the branches too far from the query:

- Rule 1: A node p will be not searched if $d(q, Y) + r_p < d(q, M_p)$ as illustrated in Figure 4.7 (a).
- Rule 2: A point $X_i \in S_p$ will not be the nearest neighbor of q if $d(q, Y) + d(X_i, M_p) < d(q, M_p)$ as illustrated in Figure 4.7 (b).

Experiment results demonstrate the efficiency of the proposed algorithm for a small dataset (1000 data points).

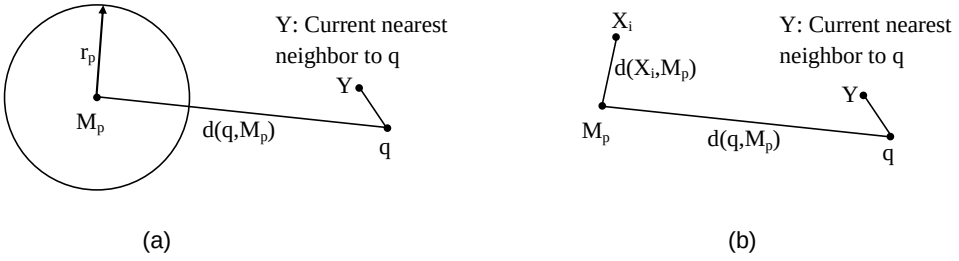


Figure 4.7: Illustration of the rule 1 (a) and rule 2 (b) for tree pruning; (reproduced from [Fukunaga and Narendra, 1975])

Table 4.2: Indexing methods based on clustering in $\Re^k$ vector space.

Method	Worst-case search complexity	Memory cost	Dataset & Feature dimension (k)	Main features
Fukunaga and Narendra, 1975	$O(n)$	$O(n)$	Uniform distribution data, three thousand records, $k = 8$	Single K-means-based clustering tree, high computation cost of tree building, ENN search
Nister and Stewenius, 2006	$O(n)$	$O(n)$	SIFT features, one million images, $k = 128$	Single hierarchical vocabulary tree based on K-means, high computation cost of tree construction, ANN search
Leibe et al., 2006	$O(n)$	$O(n)$	SIFT features, one million records, $k = 128$	Combined partitional-agglomerative clustering algorithm, fast matching with the ball tree, fast clustering and tree building, range and ANN search
Muja and Lowe, 2009	$O(n)$	$O(n)$	SIFT features, 31 billions records, $k = 128$	Single K-means-based clustering tree, priority search, ANN search
Muja and Lowe, 2012	$O(n)$	$O(n+m)$	SIFT and SURF features, $k \in \{128, 64\}$	Multiple randomized K-medoids-based clustering trees, random selection of centers, modest computation cost of tree building, priority search, ANN search

Vocabulary K-means tree: [Nister and Stewenius, 2006] proposed a hierarchical vocabulary tree for representation of feature vectors. At first, the K-means algorithm is employed to partition the feature vectors into K groups, where the parameter K is treated as the branching factor of the tree rather than the number of cluster centers as usual. Each group contains the feature vectors closest to a particular center. This process is recursively repeated for each of the obtained groups until the height of the tree is higher than a predefined level L. The nodes at the highest level are considered as leaf nodes, containing the actual feature vectors. In this way, the tree is hierarchically constructed and defines the quantized cells of feature vectors treated as the visual vocabularies. Figure 4.8 illustrates this process. Next, each node i of the tree is assigned with a weight w_i computed as follows:

$$w_i = \ln \frac{N}{N_i} \tag{4.3}$$

where N is the number of images in the database and N_i is the number of images in the database such that at least one feature vector contained in each image passes through node i. Given a query feature vector, a path is defined by traversing down the tree until a leaf node is reached. At each intermediate level, the node with the center closest to the feature vector is selected to be further explored. Image retrieval is performed by computing a relevant score between a query image and each database image. This relevant score is computed as follows. A global descriptor is created for each image by accumulating the weighted frequencies of the feature vectors of the image whose corresponding paths pass through the nodes of the tree. The relevance score of two images is then defined as the normalized difference between the global descriptors.

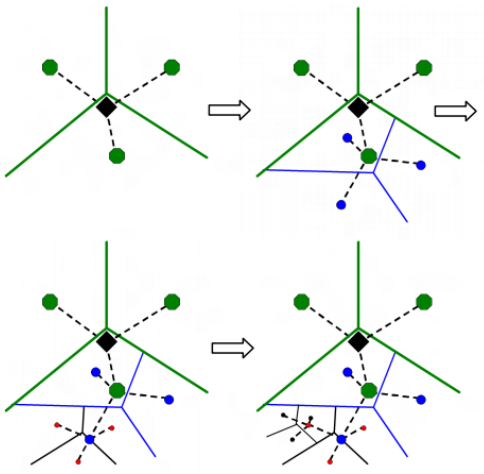


Figure 4.8: Construction of the vocabulary tree with the branching factor $K = 3$ (reprinted from [Nister and Stewenius, 2006]).

Agglomerative clustering tree: [Leibe et al., 2006] proposed an efficient clustering and fast matching methods by combining the advantages of the K-means and agglomerative techniques. In contrast to the traditional K-means technique, the agglomerative algorithm builds a hierarchical clustering tree in a bottom-up manner. Particularly, starting from the data points, it iteratively selects and merges the pairs of two clusters up to the root node

of the tree. At the beginning, each data point is a cluster. The criterion for selecting two clusters to be merged is based on the average similarity which is computed as the average pairwise similarity between the points of the two clusters [Guha et al., 1998]. Therefore, the proposed algorithm can be considered as a fusion of both top-down and bottom-up clustering strategies. First, an initial partitioning of the data points is performed by dividing them into two subsets corresponding to Laplacian maxima and minima. Each subset is then further partitioned using the K-means algorithm where K is the number of clusters in each subset and is set to be small for gaining efficiency. Next, the agglomerative clustering technique is independently applied to each resulting partition. Finally, the agglomerative algorithm is run once more on all the clusters obtained previously. For fast matching and searching, a binary ball tree is constructed in which two children of a node correspond to two clusters merged during the agglomerative clustering process. Fast searching is accomplished with the use of the triangle inequality rule. Experimental results showed the efficiency of the proposed clustering algorithm and significant speedup is achieved for feature matching and searching.

Priority K-means clustering tree: [Muja and Lowe, 2009] extended the work of [Fukunaga and Narendra, 1975] by incorporating the use of priority queue or best-bin-first [Beis and Lowe, 1997] to the hierarchical clustering tree. In their work, approximate nearest search is proceeded by traversing down the tree and always choosing the node whose cluster center is closest to the query point. Each time when a node is selected for further exploration, the other sibling nodes are inserted into a priority queue, which contains a sequence of nodes stored in the increasing order of the distances to the query. This continues when reaching a leaf node, followed by a sequence search for the points contained in this node. Backtracking is then invoked starting from the top node stored in the priority queue. During the search process, the algorithm maintains adding new candidate nodes to the priority queue along the search path. The search process terminates early when a given number of leaf nodes have been visited. The authors reported in their experiments that the use of priority search along with the hierarchical clustering tree gives significant improvements, compared to the locality sensitive hashing algorithm and the KD-tree for many datasets.

Multiple K-medoids clustering trees: The hierarchical clustering tree has been further extended by [Muja and Lowe, 2012] to build up multiple hierarchical clustering trees. Single tree construction is essentially similar to that in [Muja and Lowe, 2009] except the use of a new K-medoids clustering algorithm. Particularly, the cluster centers are randomly selected from the input points. The clusters are then built by assigning each input point to the closest center. To achieve fast construction of the tree, the step of cluster center optimization by using squared error minimization is discarded. If the size of a cluster is sufficiently small, a leaf node is constructed containing all points in this cluster. Otherwise, an internal node is constructed associated to the cluster center. These steps are recursively applied to the new clusters corresponding to the internal nodes. Following the advanced work of [Silpa-Anan and Hartley, 2008] concerned with the advantage of using multiple randomized trees for nearest neighbor search, multiple hierarchical clustering trees are thus constructed in this work. Approximate nearest search is proceeded in parallel from multiple hierarchical clustering trees. Single search for each tree is accomplished in a same manner to that in [Muja and Lowe, 2009]. Once the single search is finished for

every tree, backtracking is applied starting from the top node stored in the priority queue. Experimental results showed that significant improvements are achieved by using parallel search in multiple hierarchical clustering trees.

4.4 Hashing-based methods

Apart from the space-partitioning-based and clustering-based approaches, there is also a wide range of hashing-based techniques dealing with the problem of ANN search. Such methods are often applied to binary features employing the Hamming distance as the similarity score of two feature vectors. The basic idea is to hash data points into buckets in which similar points might likely be hashed in the same or adjacent buckets, and dissimilar points are likely to be hashed into different buckets. Multiple hash tables are often used to ensure a good chance of collision among the similar points. ANN search can be efficiently accomplished with a sub-linear time complexity. Table 4.2 summaries the representative methods in this domain, and the details are discussed in the following paragraphs.

Locality Sensitive Hashing (LHS): The LSH-based indexing scheme has been known as one of the most popular methods, which can perform proximity search with a sub-linear time for high-dimensional data [Gionis et al., 1999, Indyk and Motwani, 1998]. The key idea of LHS is to design the hash functions that the similar points may be hashed with a high probability of collision, while the dissimilar points may likely be hashed with different keys. Particularly, let $D(p, q) \in [0, 1]$ be a similarity function of two given points p and q in a dataset X, a LSH function family $H = \{h : X \to U^1\}$ is defined as follows:

$$Pr_{h \in H}[h(p) = h(q)] = D(p, q) \tag{4.4}$$

To design such a kind of hash functions, the authors proposed to conceptually transform the d-dimensional feature vectors into d'-dimensional unary representations, where the L_1 distance is preserved ($d' > d$). Next, the hash functions are constructed by selecting l subsets, $\{g_i\}_{i=1}^{l}$. Each is composed of k elements uniformly sampled with replacement from the set $R = \{1, 2, \ldots, d'\}$ (i.e., the axes in the unary space). Each subset g_i can be regarded as a hash function composing of k random lines:

$$g_i : X \to U^k \tag{4.5}$$

Equivalently, a hash function g_i is composed of k LSH functions:

$$g_i(p) = \{h_{i1}(p), h_{i2}(p), \ldots, h_{ik}(p)\} \tag{4.6}$$

where $h_{it} \in H$. As there are l hash functions, l hash tables are created to store all the projected feature vectors. More precisely, given any data point $x \in X'$ in d'-dimensional space, the hash table T_i ($1 \leq i \leq l$) is constructed as follows:

$$T_i = g_i(x) = \{h_{i1}(x), h_{i2}(x), \ldots, h_{ik}(x)\} \tag{4.7}$$

Table 4.3: Hashing-based indexing methods in $\Re^k$ vector space.

Method	Worst-case search complexity	Memory cost	Dataset & Feature dimension (k)	Main features
Indyk and Motwani, 1998	$O(n^{1/c})$	$O(nl)$	Histograms of color, 20000 records, $k = 64$	Locality Sensitive Hashing (LHS), randomized projection, using l hash tables, (c, r)-ANN search, expensive space cost
Kulis and Grauman, 2009	$O(n^{1/c})$	$O(nl)$	80 million tiny images, GIST features, $k = 384$	Kernelized LSH, good for kernelized data, (c, r)-ANN search, expensive space cost
Panigrahy, 2006	$O(n^{2.09/c})$	$O(n)$	None	Entropy LSH, less cost of memory space, (c, r)-ANN search
Lv et al., 2007	$O(n^{1/c})$	$O(\frac{nl}{m})$	Audio dataset, 2.7 million records, $k = 192$	Multi-probe LSH, search for multiple adjacent buckets, less cost of memory space ($m \in [14, 18]$), (c, r)-ANN search
Auclair, 2009	$O(n)$	$O(nl)$	512000 images, 256 SIFT descriptors/image, $K = 128$	Making use of the highest distinctive dimensions as projection axes, poor collision of similar data points, ANN search
Jain and Doermann, 2012	$O(n)$	$O(nl)$	Logo dataset, SURF features, 664 million records, $k = 32$	Making use of the highest and lowest distinctive dimensions as projection axes, poor collision of similar data points, ANN search

For ANN search, a query q is first projected using the designed hash functions resulting in l indices $\{g_1(q), g_2(q), \ldots, g_l(q)\}$. Each is a k-dimensional vector. Using these indices, the search algorithm looks up in the hash tables to obtain a set of candidate points. Next, sequence search is applied to return the K-nearest neighbor points. This method has been shown to provide a sub-linear complexity of computation, $O(n^{1/c})$, for answering the (c, r)-ANN query. However, the main drawback is the utilization of a huge amount of memory, $O(nl)$, to store l hash tables, raising another problem known as bottleneck as argued in [Auclair, 2009].

Kernelized LSH: [Kulis and Grauman, 2009] extended the LSH to the case, when the similarity function is an arbitrary kernel function κ: $D(p, q) = \kappa(p, q) = \phi(p)^T \phi(q)$. Given an input feature vector x, the problem is then to design a specific LSH function over the feature vector $\phi(x)$ rather than x itself, where $\phi(x)$ is some unknown embedding function. In their work, the LSH hash function is constructed as: $h(\phi(x)) = sign(r^T \phi(x))$, where r is a random hyperplane drawn from $N(0, I)$ and is computed as a weighted sum of a subset of the database feature vectors. Since the newly derived $h(\phi(x))$ satisfies the LSH property (i.e., $Pr_{h \in H}[h(p) = h(q)] = D(p, q)$), the new indexing scheme is thus capable of performing similarity searching in a sub-linear time complexity, while being useful to the cases of kernelized data. However, the problem of expensive cost of memory space is still unsolved.

Entropy-based LSH: To reduce the hash tables used in the original LSH indexing scheme, [Panigrahy, 2006] introduced an entropy-based LSH indexing technique. The basic idea of the entropy-based LSH method is as follows: given a query q and a parameter r of the distance from q to its nearest neighbor, the *synthetic* neighbors of q within a distance r are randomly generated to be hashed using the LSH functions. The obtained hash keys are then used to access the bucket candidates in which we expect that the true nearest neighbor of q may be present. The rest is then shifted to determine a sufficient large number of the synthetic neighbors of q to ensure that the *c-approximate* nearest neighbor search (c-ANN) can be resolved under a sub-linear time. For this purpose, the author proposed to compute the entropy $I(h(p)|q, h)$, given a query q and the LSH function h. This entropy is then used to estimate the required search time of $O(n^{(1+o(1))1.47/c})$. In other words, it is required to generate $n^{(1+o(1))1.47/c}$ synthetic nearest neighbors of q, each of which is used to access the corresponding bucket. Sequence search is applied to the data points contained in the obtained buckets. A detailed analysis of the entropy-based LSH algorithm was presented in [Lv et al., 2007] showing that this method is subjected to several weaknesses. First, it is not trivial to determine a good setting for the parameter r since this value is highly dependent on the dataset. Second, without prior knowledge about the data, the estimation of r is not reliable. Third, the process of generating the synthetic neighbors of q by sampling the feature space from a normal distribution is inefficient and sensitive to the parameter quantization. Finally, experiment results performed by [Lv et al., 2007] show that the entropy-based LSH algorithm does not give noticeable search improvement, compared to the original LSH method.

Multi-probe LSH: [Lv et al., 2007] proposed another approach, known as *multi-probe* LSH, to reduce the hash tables used in the original LSH indexing scheme. This method requires less memory space, while retaining the same search precision as in the original LSH algorithm. The basic idea is to search multiple buckets, which probably contain the

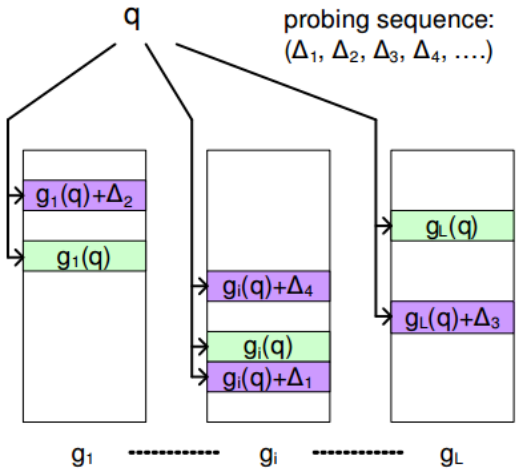


Figure 4.9: Illustration of searching process in the multi-probe LSH algorithm: $g_i(q)$ is the hash values of the query q (green buckets), and $g_i(q) + \Delta_i$ are the adjacent buckets for probing (reprinted from [Lv et al., 2007]).

potential nearest neighbors of the query. The rationale of looking at multiple buckets is realized based on the fact that if a data point p is close to another data point q but they are not hashed into the same bucket, then there is a high chance that they are hashed into two "close" buckets. Therefore, the multi-probe LSH method works by constructing a sequence of hash perturbation vectors $\{\Delta_1, \Delta_2, \ldots, \Delta_l\}$. Next, let $\{g_1(q), g_2(q), \ldots, g_l(q)\}$ be the indices of a given query q obtained from the original LSH algorithm, the new indices $\{g_i(q) + \Delta_i\}$ are used to look up in the hash tables for obtaining the nearest neighbor candidates (Figure 4.9). Finally, sequence search is applied to return the K-nearest neighbor points. In this way, the proposed method reduces the space requirement and increases the chance of finding the true answers. Experimental results showed a significant improvement in space efficiency of the multi-probe LSH method, compared with the original LSH scheme, when working on a wide range of search precision. However, our experiments, using the implementation provided from the open source FLANN library[2], showed that the multi-probe LSH algorithm performs poorly, compared to the randomized KD-trees and hierarchical K-means tree, applied to the SIFT and GIST features.

Distinctive-dimension-based hashing: [Auclair, 2009] proposed a new indexing algorithm for near duplicate images based on local descriptors (typically SIFT descriptor). Once the feature vectors are computed, the value in each dimension of this vector is normalized by using the mean and standard deviation in the same dimension, computed from all the feature vectors. These values are then sorted to select the K strongest scores whose corresponding dimensions are considered as the distinctive positions. The hashing function is then defined by selecting the first K distinctive dimensions of a given feature vector. Given a query feature vector, its hash keys are first computed. Next, the hash keys are used to access the appropriate buckets in the hash tables to obtain a shortlist of nearest candidates. Sequence search is finally applied to return the approximate nearest answers.

[2]http://www.cs.ubc.ca/ mariusm/index.php/FLANN/FLANN

For this approach, it can be noted that the use of distinctive dimensions is too sensitive to discriminate the objects since very similar points could be hashed with different keys (Figure 4.10). In addition, the task of selecting a good setting for K needs to be more thoroughly studied.

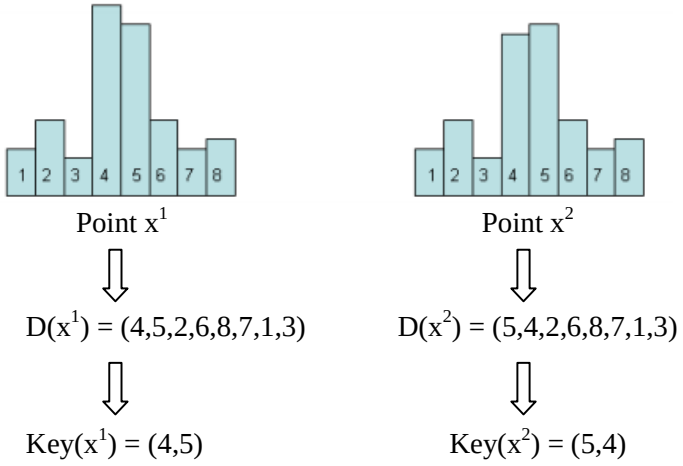


Figure 4.10: Example of the hashing algorithm in [Auclair, 2009]: Two points x^1 and x^2 are quite similar in a 8-dimensional space; $D(x^1)$ and $D(x^2)$ are the sorted vectors with respect to the distinctive dimensions; $Key(x^1)$ and $Key(x^2)$ are the obtained hash keys where $K = 2$ in this example; (reprinted from [Auclair, 2009]).

[Jain and Doermann, 2012] improved the work of [Auclair, 2009] in a different way. In their work, the hash functions are defined by selecting 6 dimensions with the highest distinctive scores and 6 dimensions with the lowest distinctive scores. In the retrieval stage, the hash keys of a given descriptor are used to find out all the keypoints having collision with these hash keys to form the candidate list. All these candidate keypoints are finally filtered using a geometric consistency process. Experimental results showed quite good performance under the context of logo retrieval. The main issue of this method remains the same as in the [Auclair, 2009] which concerns the sensitiveness of the distinctive dimensions.

4.5 Other methods

All the methods discusses so far can be regarded as direct-based indexing algorithms since they directly address the problem of feature indexing by performing an offline phase of reorganization of the data. In addition to these direct-based indexing method, indirect-based techniques for feature indexing have also been studied in the literature. Such techniques often address the problem of fast proximity search by improving the feature-extraction step [Mikolajczyk and Matas, 2007, Roy et al., 2012] and/or the matching step [Cheng et al., 1984, Mori et al., 2001]. Table 4.4 reviews some typical methods in this domain, and the details are discussed in the following.

[Mikolajczyk and Matas, 2007] employed linear projection to improve the descriptors

Table 4.4: Other indexing methods in $\Re^k$ vector space.

Method	New descriptors	New matching	Search performance	Main features
Mikolajczyk and Matas, 2007	×		+++	Feature improvement by de-correlation and dimension reduction, limited to Gaussian distribution of feature variation
Roy et al., 2012	×	×	++	Coarse-to-fine matching, dedicated to historical documents, sensitive to glyphs segmentation
De-Yuan Cheng, 1984		×	+++	Partial distance search (PDS), loop unrolling
Mori et al., 2001	×	×	+++	The use of few representative shape contexts to filter the matches, feature improvement with *shapeme*, and vector quantization of shape contexts

for fast tree matching. In their work, the descriptor space are transformed into a new space by applying a whitening linear transformation (i.e., for feature de-correlation), followed by a PCA transformation (i.e., for dimension reduction). The experiments, applied to the SIFT descriptors, demonstrated that the nearest searching procedure based on the transformed descriptors provides high accuracy and low running time on various tree-based indexing structures. It is noticed that even though the proposed linear transformation can be applied for any descriptors, it is restricted to one constraint that the variation of two descriptors of the same scene or surface has a Gaussian distribution.

[Roy et al., 2012] presented a coarse-to-fine indexing technique for fast text retrieval in historical documents. In their work, the input text lines are indexed at two levels as illustrated in Figure 4.11:

- Coarse level: the text lines are segmented and described with respect to the connected components (CCs).
- Finer level: the text lines are segmented and described with respect to the glyphs.

For each level of features, a training step is applied to generate a codebook. For each codebook, the representative elements are learnt using an unsupervised algorithm. After that, the text lines of the books are indexed at two levels corresponding to two codebooks. Given a query text line, a retrieval step is first applied to the CC-based codebook and then to the glyph-based codebook, if necessary. The approximate string matching algorithm is finally applied to retrieve the answers. The authors demonstrated a good performance (i.e., high accuracy and low cost of processing time) of the proposed system for the historical books. The advantage of this system is the avoidance of the word segmentation step which is very sensitive to noise. However, the robustness of the primitive segmentation step (i.e.,

CCs and glyphs) needs to be further studied as this step is sensitive to the noise and the variation of characters. For example, the primitive segment step is sensitive to the text characters having multiple components, such as 'i', 'à', 'é'.

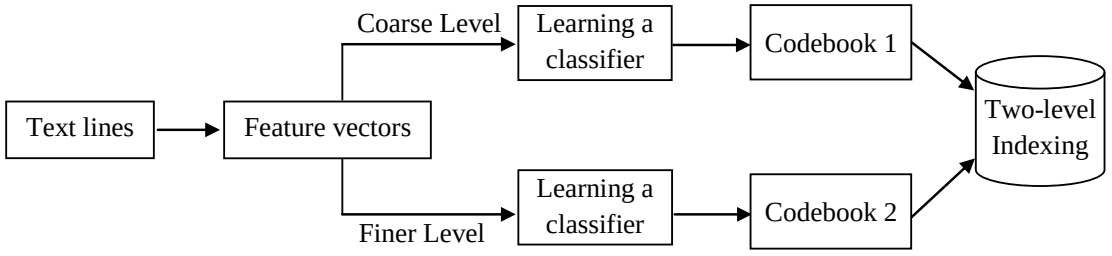


Figure 4.11: Illustration of multi-level indexing in [Roy et al., 2012].

[Cheng et al., 1984] presented a simple and efficient improvement of the matching step for brute-force search, so-called partial distance search (PDS). Given two feature vectors u and v in a k-dimensional space and a reference value D_{ref} computed as the distance from the query point to the nearest neighbor found so far, let $sum_i(u, v)$ be the partial sum of square differences up to the i^{th} coordinate ($1 \leq i \leq k$):

$$sum_i(u, v) = \sum_{t=1}^{i} (u_t - v_t)^2 \tag{4.8}$$

The basic idea of the PDS technique is that the calculation of the distance between u and v is terminated early at the dimension i^{th} ($1 \leq i \leq k$) if $sum_i(u, v) \geq D_{ref}$. This partial distance search has been widely used in the literature [McNames, 2001, Muja and Lowe, 2009, Muja and Lowe, 2012] as an alternative solution for brute-force search.

[Mori et al., 2001] introduced two new matching methods for efficient retrieval of similar shapes based on the shape context descriptor. The first method is so-called *representative shape contexts*, which concerns the use of few shape contexts of a given query shape to match against the full set of shape contexts of every database image. The rationale of this idea is that when trying to match two sufficiently different shapes, few or none of the shape contexts from the first shape have good correspondences on the other. Therefore, if there is a lack of good correspondences between two shapes, we can assume that the two shapes are different. In this method, only five representative shape contexts are randomly computed for the query. Then, a matching step is performed using these representative shape contexts to obtain a shortlist of candidate shapes.

The second matching method is so-called *shapemes*, which concerns the use of vector quantization to construct a discrete set of the shape context labels. Its basic idea is to quantize the shape context features into different classes using an unsupervised algorithm. To be more specific, in the training stage, a clustering technique such as K-means, can be used to cluster a set of shape context feature vectors into discrete groups. Each group is then quantized by an integer number called shape context label. Once all the context labels are derived, each shape is then represented by a global descriptor, as illustrated in Figure 4.12, constructed by counting the frequency of each label from the shape contexts

of the shape. Both these shape context-based matching methods have been verified on different shape datasets, demonstrating quite good results.

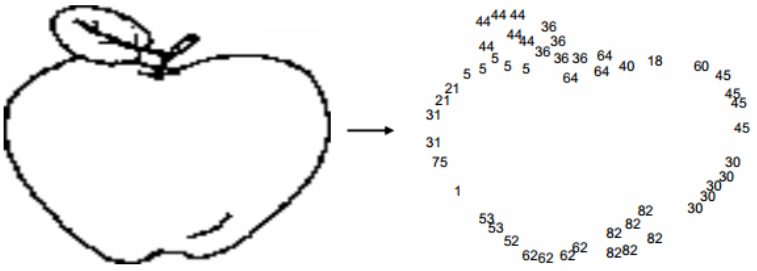


Figure 4.12: Shapeme descriptor: (a) an input image; (b) shapeme representation of (a); (reproduced from [Mori et al., 2001]).

4.6 Discussion

Table 4.5 presents some key conclusions about all the indexing methods discussed previously. The hashing-based approaches, particularly the LHS-based indexing methods, provided a solid basis dealing with the problem of ANN search. These methods have been theoretically shown to perform approximate nearest neighbor search in a sub-linear time, given a sufficient space requirement. The main drawback is the high memory cost to construct the hash tables. Several works [Panigrahy, 2006, Lv et al., 2007] have been investigated to deal with this weakness but the obtained results are not satisfactory. In addition, the search precision could be a problem because the true nearest neighbors could be hashed into many adjacent buckets, making the access to a single hash bin insufficient to recover the true answers. Furthermore, these hashing-based methods are known to work well on binary features, their behaviors on real feature vectors would need further investigation.

The clustering-based approaches have shown quite good performance in a wide range of feature types, and scaled well to the increase of the dataset size. Extensive experiments have been performed in [Muja and Lowe, 2009, 2012] showing that these approaches are able to give a speedup of hundreds of times, compared to the brute-force search, while maintaining a reasonably high search precision (i.e., $> 80\%$). They are also known to work well on both binary features and real feature vectors. The main disadvantage is the expensive time of the process of tree construction. However, this matter would make sense only to several application domains where a regular update of data and tree construction is needed.

The space-partitioning-based approaches, particularly the KD-tree-based algorithms, seem to be the most appropriate solutions for all aspects of search precision, search speedup, and tree construction time. Extensive experiments have been performed by [McNames, 2001, Silpa-Anan and Hartley, 2008, Muja and Lowe, 2009], demonstrating the same search performance as the clustering-based approaches. These experiments also reveal that the space-partitioning- and clustering-based approaches significantly outperform the

Table 4.5: Summary of different approaches for feature indexing in vector space

Approach	Pros.	Cons.	Search performance
Space partitioning	Fast tree building, applicable to both real and binary features, good generality	Poor ENN search, difficulties of dynamic update of the tree, tree-traversing-inefficient	+++
Clustering	Applicable to both real and binary features, good generality, memory space-efficient	Offline computation complexity, poor ENN search, difficulties in dynamic update of the tree	+++
Hashing	Applicable to binary features only, easy to index new data points	Memory space complexity, poor ENN search	+
Others	Good retrieval performance	Poor generality, domain-dependent, poor ENN search, poor support of dynamic update	++

LHS-based indexing schemes. All in all, while the aforementioned approaches are shown to give satisfactory results for approximate nearest neighbor search, their performance for *exact* nearest neighbor search is not significantly improved.

Finally, some other indexing methods, which rely on improving the features and/or matching step, might work well under some particular domains but they face several difficulties. At first, most of these methods are dedicated to specific applications, where prior knowledge about the domain is often known, and thus it is difficult to apply them to other tasks. Furthermore, since these methods rely on feature handling, the real problem of feature indexing still remains. As a result, such methods tend to perform poorly with respect to the increase of dataset size and data dimensionality.

In the next chapter, we attempt to bring a new contribution for feature indexing in feature vector space. We also empirically demonstrate that the proposed indexing algorithm performs much better for both the tasks of ENN and ANN search, compared to the state-of-the-art KD-tree-based and clustering-based methods.

Chapter 5

An efficient indexing scheme based on linked-node m-ary tree (LM-tree)

Following the conclusions given in the previous chapter, it is highly agreed that feature indexing is very important for a real-time image processing system. This chapter presents a new contribution for feature indexing in high-dimensional feature vector space. The proposed algorithm, called linked-node m-ary tree (LM-tree), has many favourable properties that make it different from all the existing methods. Extensive experiments, applied to a wide corpus set of features, have been performed to validate the proposed algorithm in comparison with the state-of-the-art indexing schemes.

5.1 Introduction

As we argued earlier, despite the fact that a large number of indexing algorithms have been proposed in the literature, few of them (e.g., LHS-based schemes [Lv et al., 2007], randomized KD-trees [Silpa-Anan and Hartley, 2008], randomized K-medoids clustering trees [Muja and Lowe, 2012], and hierarchical K-means tree [Muja and Lowe, 2009]) have been well validated on extensive experiments to give satisfactory performance on specific benchmarks. Although the mentioned algorithms can produce quickly the answers of approximate nearest neighbors of a given query, their search performances are still limited in the case where a pretty high search precision is desired (e.g., $> 90\%$). Especially, in some applications where exact search is required, these algorithms give little or even no better search performance compared to the brute-force search. These arguments leave a room for advanced indexing algorithms.

In this chapter, an attempt of bringing a new and efficient indexing algorithm in feature vector space is made. Particularly, a linked-node m-ary tree (LM-tree) indexing algorithm is presented, which works really well for both exact and approximate nearest neighbor search. The proposed indexing algorithm consists of three main parts, each of which is described as follows. First, a new polar-space-based method of data decomposition is presented to construct the LM-tree. The new decomposition method employs randomly two axes from a few axes having the highest variance of the underlying dataset to iteratively partition the data into m ($m > 2$) roughly equally sized subsets. This spirit is in contrast to many existing tree-based indexing algorithms, where only one axis is employed for the same purpose. Second, a novel pruning rule is proposed to efficiently narrow down the search space. Furthermore, the computation of the lower bounds is very simple, avoiding the overhead of complicated computation as often seen in many existing approaches. Finally, a bandwidth search method is introduced to explore the nodes of the LM-tree. Experimental results, applied to one million 128-dimensional SIFT features and 250000 960-dimensional GIST features, show that the proposed algorithm gives a significant improvement of search performance, compared to many state-of-the-art indexing algorithms. An additional application to image retrieval is also investigated to further demonstrate the efficiency of the proposed indexing scheme.

5.2 The proposed algorithm

The proposed indexing scheme is a tree-based structure, composing of three main components: constructing the LM-tree, doing exact nearest search with the LM-tree, and doing approximate nearest search with the LM-trees. The detail of each component is described in the following.

5.2.1 Construction of the LM-tree

Given a dataset X that is composed of N feature vectors in a D-dimensional space R^D, we present, in this section, an indexing structure to index the dataset X while supporting an efficient proximity search. For a better presentation of our approach, we use the notation $\mathbf{p}$ as a point in the R^D feature vector space, and p_i as the i^{th} component of $\mathbf{p}$ ($1 \leq i \leq D$). We also denote $p = (p_{i_1}, p_{i_2})$ as a point in a $2D$ space. We adopted here the conclusion made in [Silpa-Anan and Hartley, 2008] about the use of PCA for aligning the data before constructing the LM-tree. This approach enables us to partition the data via the narrowest directions. In particular, the dataset X is translated to its centroid following a step of data rotation to make the coordinate axes aligned with the principal axes. Note that no dimension reduction is performed in this step. In fact, PCA analysis is used only to align the data. Next, the LM-tree is constructed by recursively partitioning the dataset X into m roughly equal-sized subsets. Next, we present the main steps of the LM-tree's construction process with respect to the outline in Algorithm 2:

- Sort the axes in decreasing order of variance, and choose randomly two axes, i_1 and i_2, from the first L highest variance axes ($L < D$).

- Project every point $\mathbf{p} \in X$ into the plane $i_1\mathbf{c}i_2$, where $\mathbf{c}$ is the centroid of the set X, and then compute the corresponding angle: $\phi = \arctan(p_{i_1} - c_{i_1}, p_{i_2} - c_{i_2})$.
- Sort the angles $\{\phi_t\}_{t=1}^{n}$ in increasing order ($n = |X|$), and then divide the angles into m disjointed sub-partitions: $(0, \phi_{t_1}] \cup (\phi_{t_1}, \phi_{t_2}] \cup \ldots \cup (\phi_{t_m}, 360]$, each of which contains roughly the same number of elements (e.g., the data points projected into the plane $i_1\mathbf{c}i_2$).
- Partition the set X into m subsets $\{X_k\}_{k=1}^{m}$ corresponding to m angle sub-partitions obtained in the previous step.

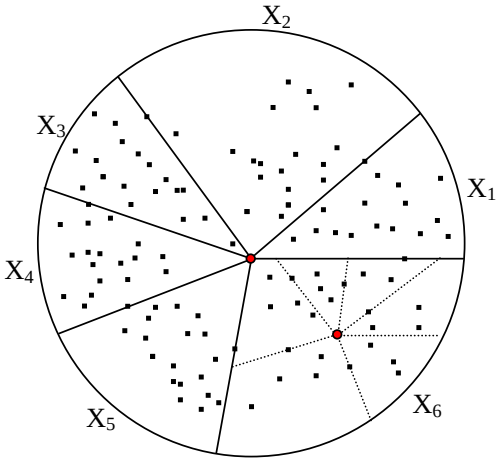


Figure 5.1: Illustration of the iterative process of data partitioning in an 2D space: the 1^{st} partitioning is applied to the dataset X, and the 2^{nd} partitioning is applied to the subset X_6 (the branching factor $m = 6$).

For each subset X_k, a new node T_k is constructed and then attached to its parent node, where we also store the following information: the split axes (i.e., i_1 and i_2), the split centroid (c_{i_1}, c_{i_2}), the split angles $\{\phi_{t_k}\}_{k=1}^{m}$, and the split projected points $\{(p_{i_1}^k, p_{i_2}^k)\}_{k=1}^{m}$, where the point $(p_{i_1}^k, p_{i_2}^k)$ corresponds to the split angle ϕ_{t_k}. For efficient access across these child nodes, a direct link is established between two adjacent nodes T_k and T_{k+1} ($1 \leq k < m$), and the last one T_m is linked to the first one T_1. Next, we repeat this partitioning process for each subset X_k that is associated with the child node T_k until the number of data points in each node falls below a pre-defined threshold L_{max}. Figure 5.1 illustrates the first and second levels of the LM-tree construction with a branching factor of $m = 6$.

It is worthwhile pointing out that each time that a partition proceeds, two axes are employed for dividing the data. This approach is in contrast to many existing tree-based indexing algorithms, where only one axis is employed to partition the data. Consequently, as argued in [Silpa-Anan and Hartley, 2008], considering a high-dimensional feature space, such as 128-dimensional SIFT features, the total number of axes that are involved in the tree construction is rather limited, making any pruning rules less-efficient, and the tree is less discriminative for later usage of searching. Naturally, the number of principal axes involved in partitioning the data is proportional to both the search efficiency and precision.

Algorithm 2 LMTreeBuilding(X, m, L, L_{max})

Input: A dataset $X \in R^D$, the branching factor (m), the number of highest variance axes to be selected (L), and the maximum number of data points in a leaf node (L_{max}).
Output: The LM-tree of representing all the data points.
$A \leftarrow$ sort the axes in the decreasing order of variance
$t_1 \leftarrow random(L)$ {select randomly a number in $[1, L]$}
$t_2 \leftarrow random(L)$
{make sure that $t_1 \neq t_2$}
$i_1 \leftarrow A[t_1]$, $i_2 \leftarrow A[t_2]$ {select two axes i_1 and i_2}
$\mathbf{c} \leftarrow$ compute the centroid of X
$i \leftarrow 1$, $n \leftarrow |X|$
{project every point $\mathbf{p} \in R^D$ into the plane $i_1\mathbf{c}i_2$ to compute ϕ_i}
while $i \leq n$ **do**
  $\mathbf{p} \leftarrow X[i]$ {for each point $\mathbf{p} \in X$}
  $\phi_i = \arctan(p_{i_1} - c_{i_1}, p_{i_2} - c_{i_2})$ {compute the corresponding angle w.r.t the selected axes i_1 and i_2}
  $i \leftarrow i + 1$
end while
$B \leftarrow$ sort the angles ϕ_i in the increasing order
$\Delta \leftarrow n/m$
$X_k \leftarrow \emptyset$ {reset each subset X_k to empty $(1 \leq k \leq m)$}
{start partitioning X into m equal-sized subsets $\{X_k\}$}
for each $\mathbf{p} \in X$ **do**
  $\phi_{\mathbf{p}} = \arctan(p_{i_1} - c_{i_1}, p_{i_2} - c_{i_2})$
  $k \leftarrow 1$
  while $k \leq m$ **do**
    if $\phi_{\mathbf{p}} \leq B[k\Delta]$ **then**
      $X_k \leftarrow X_k \cup \{p\}$ {put $\mathbf{p}$ into a proper subset X_k}
      $k \leftarrow m + 1$ {break the loop}
    end if
    $k \leftarrow k + 1$
  end while
end for
{Recursively building the tree for each subset $\{X_k\}$}
$k \leftarrow 1$
while $k \leq m$ **do**
  $T_k \leftarrow$ construct a new node corresponding to X_k
  if $|X_k| < L_{max}$ **then**
    LMTreeBuilding(X_k, m, L, L_{max}) {recursive tree building for each subset X_k}
  end if
end while

5.2.2 Exact nearest neighbor search in the LM-tree

Exact nearest neighbor search in the LM-tree is proceeded by using a branch-and-bound algorithm. Given a query point $\mathbf{q}$, we first project $\mathbf{q}$ into a new space by using the principal axes, which is similar to how we processed the LM-tree construction. Next, starting from the root, we traverse down the tree and we use the split information stored at each node to choose the best child node for further exploration. Particularly, given an internal node u along with the corresponding split information $\{i_1, i_2, c_{i_1}, c_{i_2}, \{\phi_{t_k}\}_{k=1}^{m}, \{(p_{i_1}^k, p_{i_2}^k)\}_{k=1}^{m}\}$ which is already stored at u, we first compute an angle: $\phi_{q_u} = \arctan(q_{i_1} - c_{i_1}, q_{i_2} - c_{i_2})$. Next, a binary search is applied to the query angle ϕ_{q_u} over the sequence $\{\phi_{t_k}\}_{k=1}^{m}$ to choose the child node of u that is closest to the query $\mathbf{q}$ for further exploration. This process continues until a leaf node is reached, followed by the partial distance search (PDS) [Cheng et al., 1984, McNames, 2001] to the points contained in the leaf. Backtracking is then invoked to explore the rest of the tree. Algorithm 3 outlines the main steps of this process.

Algorithm 3 ENNSearch(u, $\mathbf{q}$, $dist_{lb}$)

Input: A pointer to a current node of the LM-tree (u), a query ($\mathbf{q}$) and the current lower bound distance ($dist_{lb}$)
Output: The exact nearest neighbor of $\mathbf{q}$
if u is a leaf node **then**
  $\{\mathbf{p}_{best}, dist_{best}\} \leftarrow$ apply sequence search for the points contained in u
else
  $\{i_1, i_2, c_1, c_2, \{\phi_{t_k}\}_{k=1}^{m}\} \leftarrow$ access split information stored at u
  $\phi_{q_u} = \arctan(q_{i_1} - c_1, q_{i_2} - c_2)$
  $s_k \leftarrow$ find a son s_k of u where ϕ_{q_u} is contained in $(\phi_{t_{k-1}}, \phi_{t_k}]$
  ENNSearch(s_k, $\mathbf{q}$, $dist_{lb}$) {explore s_k first}
  $m \leftarrow$ the number of sons of u
  $L_{ord} \leftarrow \emptyset$ {construct an ordered list of visiting the nodes}
  $s_{left} \leftarrow s_k$, $s_{right} \leftarrow s_k$, $i \leftarrow 1$
  while $i \leq m/2$ **do**
    $L_{ord} \leftarrow L_{ord} \cup move2left(s_{left})$ {get the adjacent node in the left}
    $L_{ord} \leftarrow L_{ord} \cup move2right(s_{right})$ {get the adjacent node in the right}
    $i \leftarrow i + 1$
  end while
  for each node s in L_{ord} **do**
    $\{dist_{nlb}, \mathbf{q}_n\} \leftarrow$ ComputeLB(s, $\mathbf{q}$, $dist_{lb}$) {update new lower bound and query}
    if $dist_{nlb} < dist_{best}$ **then**
      ENNSearch(s, $\mathbf{q}_n$, $dist_{nlb}$)
    end if
  end for
end if
return $\mathbf{p}_{best}$ {the exact nearest neighbor of $\mathbf{q}$}

Each time when we are positioned at a node u, the lower bound is computed as the distance from the query $\mathbf{q}$ to the node u. If the lower bound is higher than the distance

from **q** to the nearest point found so far, we can safely avoid exploring this node and proceed with other nodes. In this section, we present a novel rule for efficiently computing the lower bound. Our pruning rule was inspired by the work presented in the principal axis tree (PAT) [McNames, 2001]. PAT is a generalization of the KD-tree, where the page regions are hyper-polygons rather than hyper-rectangles, and the pruning rule is recursively computed based on the law of cosines.

The drawbacks of the PAT's pruning rule are the computational cost of the complexity (i.e., $O(D)$) and the inefficiency when working on a high-dimensional space because only one axis is employed at each partition. As our method of data decomposition (i.e., LM-tree construction) is quite different from that of the KD-tree-based structures, we have developed a significant improvement of the pruning rule used in PAT. Particularly, we have incorporated two following major advantages for the proposed pruning rule:

- The lower bound is computed as simple as in a 2D space, regardless of how large the dimensionality D is. Therefore, the time complexity is just $O(2)$ instead of $O(D)$ as in the case of PAT.
- The magnitude of the proposed lower bound is significantly higher than that in PAT. This enables the proposed pruning rule to work efficiently.

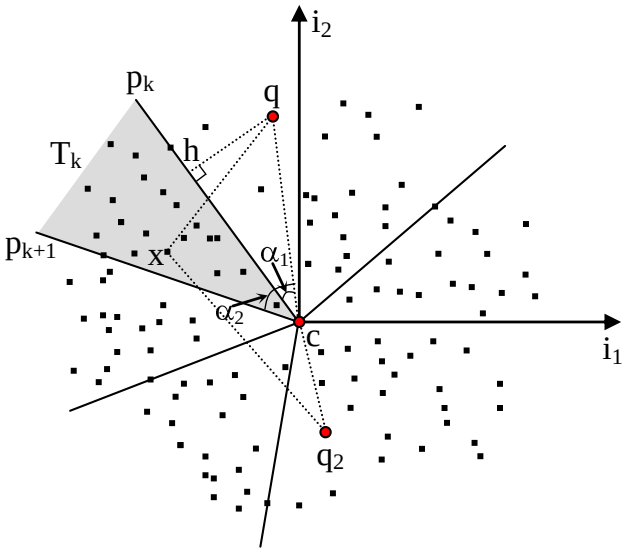


Figure 5.2: Illustration of the lower bound computation.

We now come back to the description of computing the lower bound. Let u be the node in the LM-tree at which we are positioned, let T_k be one of the children of u that is going to be searched, and let $p_k = (p^k_{i_1}, p^k_{i_2})$ be the k^{th} split point, which corresponds to the child node T_k (see Figure 5.2). The lower bound $LB(\mathbf{q}, T_k)$, from **q** to T_k, is recursively computed from $LB(\mathbf{q}, u)$. The main steps of this process are as follows (i.e., Algorithm 4):

- Compute the angles: $\alpha_1 = \angle qcp_k$ and $\alpha_2 = \angle qcp_{k+1}$, where $q = (q_{i_1}, q_{i_2})$ and $c = (c_{i_1}, c_{i_2})$.

- If one of two angles α_1 and α_2 is smaller than 90^0, then we have the following fact due to the rule of cosines [McNames, 2001]:

$$d(q,x)^2 \geq d(q,h)^2 + d(h,x)^2 \tag{5.1}$$

where x is any point in the region of T_k, and $h = (h_{i_1}, h_{i_2})$ is the projection of q on the line cp_k or cp_{k+1}, while accounting for $\alpha_1 \leq \alpha_2$ or $\alpha_1 > \alpha_2$, and $d(h,x)$ is the Euclidean distance between two points. Then, we apply the rule of computing the lower bound in PAT in an 2D space as follows:

$$LB^2(\mathbf{q}, T_k) \leftarrow LB^2(\mathbf{q}, u) + d(q,h)^2 \tag{5.2}$$

Next, we treat the point $\mathbf{h} = (q_1, q_2, \ldots, h_{i_1}, \ldots, h_{i_2}, \ldots, q_{D-1}, q_D)$ in place of $\mathbf{q}$ by the means of lower bound computation for the descendant of T_k.

- If both angles, α_1 and α_2, are higher than 90^0 (e.g., the point q_2 in Figure 5.2), we have a more restricted rule as follows:

$$d(q,x)^2 \geq d(q,c)^2 + d(c,x)^2 \tag{5.3}$$

Therefore, the lower bound is easily computed as:

$$LB^2(\mathbf{q}, T_k) \leftarrow LB^2(\mathbf{q}, u) + d(q,c)^2 \tag{5.4}$$

Again, we treat the point $\mathbf{c} = (q_1, q_2, \ldots, c_{i_1}, \ldots, c_{i_2}, \ldots, q_{D-1}, q_D)$ in place of $\mathbf{q}$ by the means of lower bound computation for the descendant of T_k.

As the lower bound $LB(\mathbf{q}, T_k)$ is recursively computed from $LB(\mathbf{q}, u)$, an initial value must be set for the lower bound at the root node. Obviously, we set $LB(\mathbf{q}, root) = 0$. It is also noted that when the point $\mathbf{q}$ is fully contained in the region of T_k, no computation of the lower bound is required. Therefore: $LB(\mathbf{q}, T_k) \leftarrow LB(\mathbf{q}, u)$.

5.2.3 Approximate nearest neighbor search in the LM-tree

Approximate nearest neighbor search is proceeded by constructing multiple randomized LM-trees to account for different viewpoints of the data. To achieve the optimized utilization of memory space, the data points are stored in a common buffer, while the leaf nodes of the trees contain the indexes to these actual data points. The idea of using multiple randomized trees for ANN search was originally presented in [Silpa-Anan and Hartley, 2008], where the authors proposed to construct multiple randomized KD-trees. This technique was then incorporated with the priority search and successfully used in many other tree-based structures [Muja and Lowe, 2009], [Muja and Lowe, 2012]. Although the priority search was shown to give better search performance, it is certainly subjected to a high computational cost because the process of maintaining a priority queue during the online search is rather expensive.

Here, we exploit the advantages of using multiple randomized LM-trees but without using the priority queue. The basic idea is to restrict the search space to the branches

Algorithm 4 ComputeLB(u, $\mathbf{q}$, $dist_{lb}$)

Input: A pointer to a node of the LM-tree (u), a query ($\mathbf{q}$) and the current lower bound distance ($dist_{lb}$)
Output: New lower bound ($dist_{nlb}$) and new query point ($\mathbf{q}_n$)
$p_u \leftarrow$ get the parent node of u
$\{i_1, i_2, c_1, c_2\} \leftarrow$ access split information stored at p_u
$\alpha_{min} \leftarrow \min\{\alpha_1, \alpha_2\}$ {see explanation of α_1 and α_2 in the text}
$\mathbf{q}_n \leftarrow \mathbf{q}$
if $\alpha_{min} \leq 90^0$ **then**
    $\mathbf{h} \leftarrow$ projection of $\mathbf{q}$ on the split axis {see explanation in the text}
    $\mathbf{q}_n[i_1] \leftarrow h_{i_1}$ {update the new query at the coordinates i_1 and i_2}
    $\mathbf{q}_n[i_2] \leftarrow h_{i_2}$
    $dist_{nlb} \leftarrow dist_{lb} + (q_{i_1} - h_{i_1})^2 + (q_{i_2} - h_{i_2})^2$ {update the new lower bound distance}
else
    $\mathbf{q}_n[i_1] \leftarrow c_{i_1}$ {update the new query based on the centroid}
    $\mathbf{q}_n[i_2] \leftarrow c_{i_2}$
    $dist_{nlb} \leftarrow dist_{lb} + (q_{i_1} - c_{i_1})^2 + (q_{i_2} - c_{i_2})^2$ {update the new lower bound distance}
end if
return $\{dist_{nlb}, \mathbf{q}_n\}$

that are not very far from the considering path. In this way, we introduce a specific search procedure, so-called *bandwidth* search, which proceeds by setting a search bandwidth to every intermediate node of the ongoing path. Particularly, let $P = \{u_1, u_2, \ldots, u_r\}$ be a considering path obtained by traversing down a single LM-tree, where u_1 is the root node and u_r is the current node of the path. The proposed bandwidth search indicates that for each intermediate node u_i of P ($1 \leq i \leq r$), every sibling node of u_i at a distance of $b+1$ nodes ($1 \leq b < m/2$) on both sides from u_i does not need to be searched. The value b is called search bandwidth. Taking one example as shown in Figure 5.3, where X_6 is an intermediate node on the considering path P, then only X_1 and X_5 are candidates for further inspection given a search bandwidth of $b = 1$.

There is a notable point that when the projected query q is too close to the projected centroid c, all of the sibling nodes of u_i should be inspected as in the case of an ENN search. Particularly, this scenario occurs at a certain node u_i if $d(q, c) \leq \epsilon D_{med}$, where D_{med} is the median value of the distances between c and all of the projected data points that are associated with u_i, and ϵ is a tolerance radius parameter. In addition, in order to obtain a varying range of the search precision, we would need a parameter E_{max} for the maximum data points to be searched on a single LM-tree. Algorithm 5 outlines the flowchart of our bandwidth search process.

As we are designing an efficient solution dedicated to an ANN search, it would make sense to use an approximate pruning rule rather than an exact rule. Particularly, we have used only formula (5.4) as an approximate pruning rule. This adaption offers two favourable features. First, it reduces much of the computational cost. Recall that rule (5.4) requires the computation of $d(q, c)$ in a 2D space, where point c has been already computed during the offline tree construction. The computation of this rule is thus very

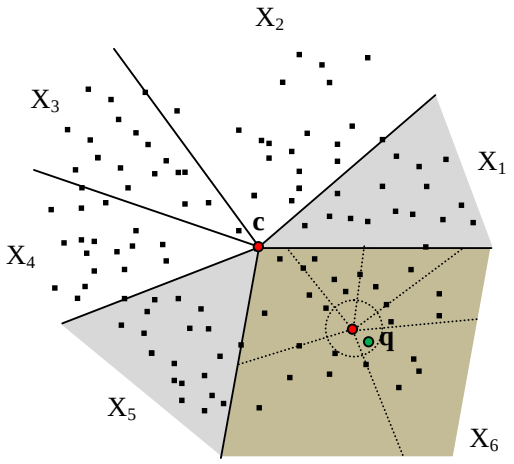


Figure 5.3: Illustration of our bandwidth search with $b = 1$: X_6 is an intermediate node of the considering path, and its adjacent sibling nodes, X_1 and X_5, will also be searched; if $\mathbf{q}$ is too close to the centroid (e.g., inside the circle for the case of X_6), then all of the sibling nodes of X_6 will be searched.

efficient. Second, it also ensures that a larger fraction of nodes will be inspected but few of them would be actually searched after checking the lower bound. In this way, it increases the chance of reaching the true nodes that are closest to the query. More generally, we have adapted formula (5.4) as follows:

$$LB^2(\mathbf{q}, T_k) \leftarrow \kappa \cdot (LB^2(\mathbf{q}, u) + d(q, c)^2) \tag{5.5}$$

where $\kappa \geq 1$ is the pruning factor that controls the rate of pruning the branches in the trees. This factor can be adaptively estimated during the tree construction given a specific precision and a specific dataset. However, we have set this value as $\kappa = 2.5$ and we show, in our experiments, that it is possible to achieve satisfactory search performance on many datasets. Algorithm 6 sketches the basic steps of computing the approximate pruning rule.

5.3 Experimental results

We evaluated our system versus several representative fast proximity search systems in the literature, including randomized KD-trees (RKD-trees) [Silpa-Anan and Hartley, 2008] that use the priority search implementation in [Muja and Lowe, 2009], hierarchical K-means tree (K-means tree) [Muja and Lowe, 2009], randomized K-medoids clustering trees (RC-trees) [Muja and Lowe, 2012], and the multi-probe LSH algorithm [Lv et al., 2007]. These indexing systems were well-implemented and widely used in the literature thanks to the open source FLANN library[1]. The source code of our system is also publicly available at this address[2]. Note that the partial distance search was implemented in these systems

[1]http://www.cs.ubc.ca/ mariusm/index.php/FLANN/FLANN
[2]https://sites.google.com/site/ptalmtree/

Algorithm 5 ANNSearch(u, $\mathbf{q}$, $dist_{lb}$, E_{max}, ϵ, κ, b)

Input: A pointer to a current node of the LM-tree (u), a query ($\mathbf{q}$), the current lower bound distance ($dist_{lb}$), and the 4 parameters E_{max}, ϵ, κ, and b.
Output: The approximate nearest neighbor of $\mathbf{q}$
if u is a leaf node **then**
  $\{\mathbf{p}_{best}, dist_{best}\} \leftarrow$ apply sequence search for the points contained in u
else
  $\{i_1, i_2, c_1, c_2, D_{med}, \{\phi_{t_k}\}_{k=1}^{m}\} \leftarrow$ access split information stored at u
  $\phi_{q_u} = \arctan(q_{i_1} - c_1, q_{i_2} - c_2)$
  $s_k \leftarrow$ find a son s_k of u where ϕ_{q_u} is contained in $(\phi_{t_{k-1}}, \phi_{t_k}]$
  ANNSearch(s_k, $\mathbf{q}$, $dist_{lb}$, E_{max}, ϵ, κ, b) {explore s_k first}
  $m \leftarrow$ the number of sons of u
  $dist \leftarrow (q_{i_1} - c_1)^2 + (q_{i_2} - c_2)^2$
  $m_{bd} \leftarrow b$ {compute the search bandwidth}
  if $dist < (\epsilon D_{med})^2$ **then**
    $m_{bd} \leftarrow m/2$
  end if
  $L_{ord} \leftarrow \emptyset$ {construct an ordered list of visiting the nodes}
  $s_{left} \leftarrow s_k$, $s_{right} \leftarrow s_k$
  $i \leftarrow 1$
  while $i \leq m_{bd}$ **do**
    $L_{ord} \leftarrow L_{ord} \cup move2left(s_{left})$ {get the adjacent node in the left}
    $L_{ord} \leftarrow L_{ord} \cup move2right(s_{right})$ {get the adjacent node in the right}
    $i \leftarrow i + 1$
  end while
  for each node s in L_{ord} **do**
    $\{dist_{nlb}, \mathbf{q}_n\} \leftarrow$ ComputeLB_ANN(s, $\mathbf{q}$, $dist_{lb}$, κ) {compute the lower bound}
    if $dist_{nlb} < dist_{best}$ **then**
      ANNSearch(s, $\mathbf{q}_n$, $dist_{nlb}$, E_{max}, ϵ, κ, b)
    end if
  end for
end if
return $\mathbf{p}_{best}$ {the approximate nearest neighbor of $\mathbf{q}$}

Algorithm 6 ComputeLB_ANN(u, $\mathbf{q}$, $dist_{lb}$, κ)

Input: A pointer to a node of the LM-tree (u), a query ($\mathbf{q}$), the current lower bound distance ($dist_{lb}$) and the pruning factor (κ)
Output: New (approximate) lower bound ($dist_{nlb}$) and new query point ($\mathbf{q}_n$)
$p_u \leftarrow$ get the parent node of u
$\{i_1, i_2, c_1, c_2\} \leftarrow$ access split information stored at p_u
$\mathbf{q}_n \leftarrow \mathbf{q}$
$\mathbf{q}_n[i_1] \leftarrow c_{i_1}$ {update the new query at the coordinates i_1 and i_2}
$\mathbf{q}_n[i_2] \leftarrow c_{i_2}$
$dist_{nlb} \leftarrow dist_{lb} + \kappa \cdot ((q_{i_1} - c_{i_1})^2 + (q_{i_2} - c_{i_2})^2)$ {update the new lower bound distance}
return $\{dist_{nlb}, \mathbf{q}_n\}$

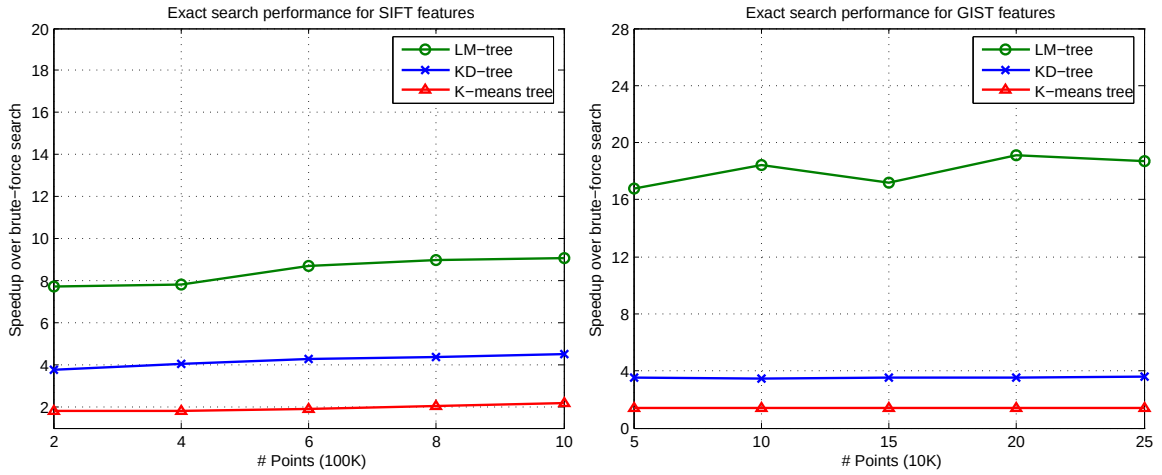
in order to improve the efficiency of the sequence search at the leaf nodes. Two datasets, ANN_SIFT1M and ANN_GIST1M [Jégou et al., 2011], were used for all of the experiments. The ANN_SIFT1M dataset contains a database of one million 128-dimensional SIFT features and a test set of 5000 SIFT features, while the ANN_GIST1M dataset is composed of a database of one million 960-dimensional GIST features and a test set of 1000 GIST features. Because the dimensionality of the GIST feature is very high and our computer configuration is limited (i.e., Windows XP, 2.4G RAM), we were not able to load the full ANN_GIST1M dataset into memory. Consequently, we have used 250000 GIST features for search evaluation. Following the convention of the evaluation protocol used in the literature [Beis and Lowe, 1997], [Muja and Lowe, 2009], [Muja and Lowe, 2012], we computed the search precision and search time as the average measures obtained by running 1000 queries taken from the test sets of the two datasets. To make the results independent on the machine and software configuration, the speedup factor is computed relative to the brute-force search. The details of our experiments are presented in this section.

5.3.1 ENN search evaluation

For ENN search, we use a single LM-tree and set the parameters that are involved in the LM-tree as follows: $L_{max} = 10$, $L = 2$, and $m \in \{6, 7\}$ with respect to the GIST and SIFT datasets (see section 5.2.1). By setting $L = 2$, we choose exactly the two highest variance axes at each level of the tree for data partitioning. We compared the performance of the ENN search of the following systems: the proposed LM-tree, the KD-tree, and the hierarchical K-means tree. Figure 5.4(a) shows the speedup over the brute-force search for the three systems, when applied to the SIFT datasets with different sizes. We can note that the LM-tree outperforms the other two systems on all of the tests. Figure 5.4(b) presents the search performance for the three systems for the GIST features. The proposed LM-tree again outperforms the others and even performs far better than the SIFT features. Taking the test where $\#Points = 150000$ on Figure 5.4(b), for example, the LM-tree gives a speedup of 17.2, the KD-tree gives a speedup of 3.53, and the K-means tree gives a speedup of 1.42 over the brute-force search. These results confirm the efficiency of the LM-tree for the ENN search relative to the two baseline systems.

5.3.2 ANN search evaluation

For the ANN search, we adopted here the benefit of using multiple randomized trees, which were successfully used in the previous works of [Silpa-Anan and Hartley, 2008] and [Muja and Lowe, 2012]. Hence, we set the required parameters as follows: $L_{max} = 10$, $L = 8$, $b = 1$, and $m \in \{6, 7\}$. By setting $L = 8$, we choose randomly two axes from the eight highest variance axes at each level of the tree for data partitioning. In addition, the parameter $b = 1$ indicates that our bandwidth search process visits three adjacent nodes (including the node in question) at each level of the LM-tree. Four systems participated in this evaluation, including the proposed LM-trees, RKD-trees, RC-trees, and K-means tree. Following the conclusion made in [Muja and Lowe, 2012] about the optimal number of parallel trees for ANN search, we used 8 parallel trees in the first three systems as the search precision is expected to be high ($> 70\%$) in our experiments. It is worth explaining

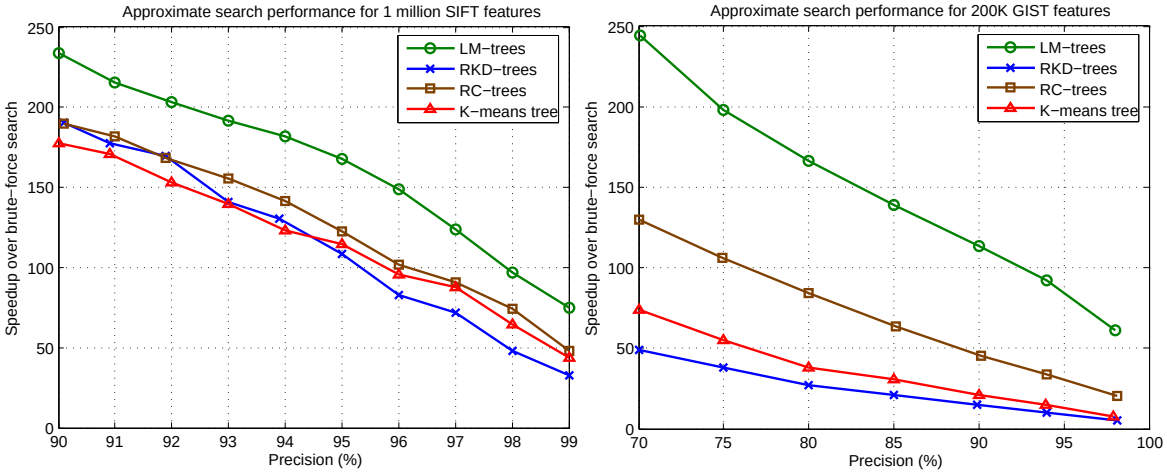


(a) (b)

Figure 5.4: Exact search performance for the SIFT (a) and GIST (b) features.

a bit the meaning of the search precision in our context. For 1-NN ANN search, the search precision is computed as the ratio of the number of exact answers to the number of queries. In our experiments, we performed 1000 queries for each test and if one system achieves a precision of 90% implying that it produces 900 exact answers out of 1000 queries.

It is also noted that we used a single tree for the K-means tree indexing algorithm because it was shown in [Muja and Lowe, 2009] that the use of multiple K-means trees does not give better search performance. For all of the systems, the parameters E_{max} and ϵ (i.e., the LM-tree) are varied to obtain a wide range of search precision.



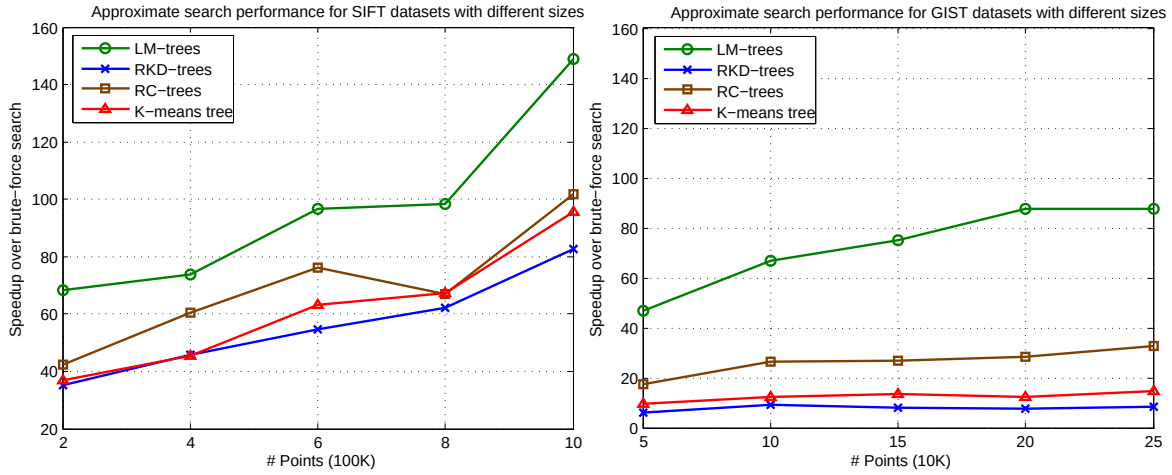
(a) (b)

Figure 5.5: Approximate search performance for the SIFT (a) and GIST (b) features.

Figure 5.5(a) shows the search speedup versus the search precision of the four systems

for 1 million SIFT features. As can be seen, the proposed LM-trees algorithm gives significantly better search performance everywhere compared with the other systems. When considering the search precision of 95%, for example, the speedups over a brute-force search of the LM-trees, RKD-trees, RC-trees, and K-means tree are 167.7, 108.4, 122.4, and 114.5, respectively. To make it comparable with the multi-probe LSH indexing algorithm, we converted the real SIFT features to the binary vectors and tried several parameter settings (i.e., the number of hash tables, the number of multi-probe levels, and the length of the hash key) to obtain the best search performance. However, the result obtained on one million SIFT vectors is rather limited. Taking the search precision of 74.7%, for example, the speedup over a brute-force search (using Hamming distance) is only 1.5.

Figure 5.5(b) shows the search performance of all of the systems for 200000 GIST features. Again, the LM-trees algorithm clearly outperforms the others and tends to perform much better than the SIFT features. The RC-trees algorithm also works reasonably well, while the RKD-trees and K-means tree work poorly for this dataset. Considering the search precision of 90%, for example, the speedups over a brute-force search of the LM-trees, RKD-trees, RC-trees, and K-means tree are 113.5, 15.0, 45.2, and 21.2, respectively.



(a) (b)

Figure 5.6: ANN search performance as a function of the dataset size: (a) the SIFT datasets (search precision = 96%); (b) the GIST datasets (search precision = 95%).

In Figure 5.6, we present the ANN search performance of the four systems as a function of the dataset size. For this purpose, the search precision is set to a rather high degree, especially at 96% and 95% for the SIFT and GIST features, respectively. This time, the LM-trees algorithm still gives a substantially better search performance than the others and tends to perform quite well, considering the increase in the dataset size. For the SIFT features, the RC-trees algorithm works reasonably well, except for the point where $\#Points = 800K$, at which its search performance is noticeably degraded. It is also noted that the speedups of the RKD-trees and K-means tree for the GIST features are quite low, even lower than the speedup of the LM-tree for the ENN search (see Figure 5.4(b)).

Three crucial factors explain these outstanding results of the LM-trees. First, the

use of the two highest variance axes for data partitioning in the LM-tree gives a more discriminative representation of the data in comparison to the common use of the sole highest variance axis as in the literature. Second, by using the approximate pruning rule, a larger fraction of nodes will be inspected, but many of them would be eliminated after checking the lower bound. In this way, the number of data points that will be actually searched, is retained under the pre-defined threshold E_{max}, while covering a larger number of inspected nodes, and thus increasing the chance of reaching the true nodes that are closest to the query. Finally, the use of bandwidth search gives much benefit in terms of the computational cost, compared to the priority search that is used in the baseline indexing systems.

The last experiment is presented on Figure 5.7 to evaluate the distance error (i.e., error ratio) of the *approximate answers* to the exact ones. By approximate nearest neighbor search, it means that not all the answers are the exact ones. More precisely, the quality of approximate nearest neighbor search is evaluated by two metrics: the search precision and the distance error ratio. Taking the search precision of 95% provided by one indexing system, for instance, implies that the system produces 950 exact answers out of 1000 queries (e.g., considering a test of 1000 queries). It also tells us that there are 50 answers which are not the *exact* ones but the *approximate* nearest neighbors. The quality of these approximate answers are evaluated by the distance error ratio to the exact ones. [Gionis et al., 1999] defined the distance error ratio as follows:

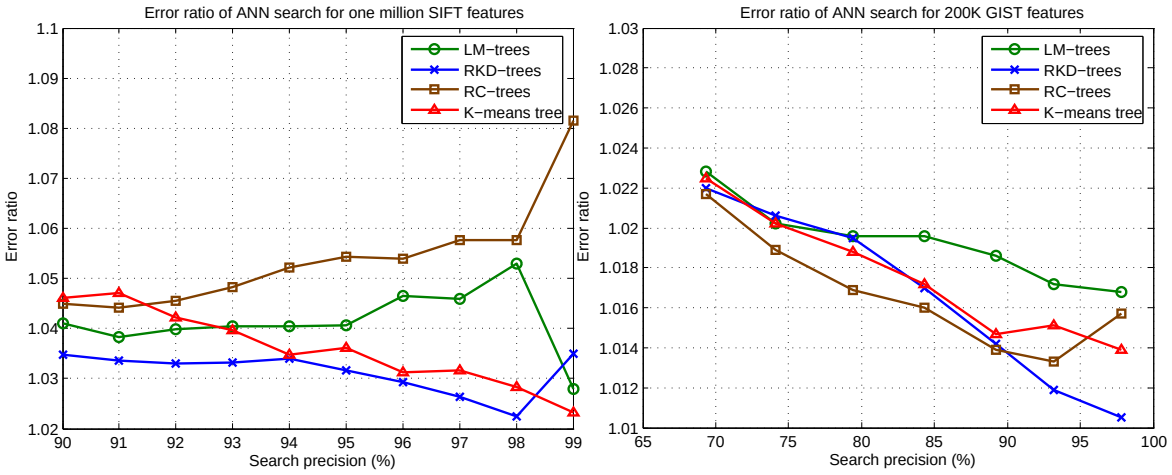
$$dist_{err} = \frac{1}{Q} \sum_{i=1}^{Q} \frac{dist(p_i, q_i)}{dist(p_i^*, q_i)} \tag{5.6}$$

where p_i is the approximate answer of the query q_i, whereas p_i^* is the exact nearest neighbor of q_i, and $dist(,)$ is the Euclidean distance between two points.

Figure 5.7 shows the distance error ratio with respect to the increase of search precision of all the systems for the SIFT and GIST features. For both cases, although the proposed system does not always perform best, the distance error ration is quite low, especially for the GIST features. On average, the distance error ratios of the proposed system are 1.0414 and 1.0193 for the SIFT and GIST features, respectively. In other words, the approximate answers given by the proposed system are, on average, located on the hyper-spheres with the radii of 4.1% and 1.9% higher than that of the exact nearest neighbors for the SIFT and GIST features, respectively.

5.3.3 Parameter tuning

In this section, we study the effects of the parameters that are involved in the LM-tree on the search performance. Two types of parameters are concerned in the LM-tree: static and dynamic parameters. The static parameters are those that are involved in the offline phase of building the LM-tree, including the maximum number of data points at a leaf node (L_{max}), the number of axes having the highest variance (L), and the branching factor (m). The dynamic parameters are used in the online phase of searching, including the search bandwidth (b), the maximum number of data points to be searched in a single LM-tree (E_{max}), the pruning factor ($\kappa = 2.5$), and the tolerance radius ϵ. For the static



(a) (b)

Figure 5.7: Evaluation of distance error ratio for ANN search applied to SIFT (a) and GIST (b) features.

parameters, our investigation, obtained on the experiments performed so far, has shown that they produce a negligible effect on the search performance provided an appropriate setting of the dynamic parameters. Specifically, it was found that the following settings for the static parameters often lead to relatively good search performance: $L_{max} = 10$, $L = 8$, and $m \in \{6, 7\}$. As a rule of thumb, the branching factor m should be high for a large-scale dataset and vice versa. The dynamic parameters (E_{max}, ϵ, b) are used to achieve a given specific search precision. They are thus treated as precision-driven parameters. This implies that given a specific precision P ($0\% \leq P \leq 100\%$), we can design a method to determine automatically the optimized settings for these parameters to achieve the best search time. The basic idea is that given a specific setting of b and ϵ, the parameter E_{max} is estimated by using a binary search. This approach enables the method to work very efficiently. In our case, we have set the parameter b to 1 and designed the following procedure to estimate the optimized setting for E_{max} and ϵ:

- Step 1: Sample the parameter ϵ into discrete values: $\{\epsilon_0, \epsilon_0 + \Delta, \ldots, \epsilon_0 + l\Delta\}$. In our implementation, we set: $\epsilon_0 = 0$, $\Delta = 0.04$, $l = 20$.

- Step 2: For each value $\epsilon_i = \epsilon_0 + i\Delta$ ($0 \leq i \leq l$):

  - Step 2(a): Estimate an initial value for E_{max} by running the approximate search procedure without consideration of the parameter E_{max}. In this way, the search procedure terminates early with respect to the current settings of ϵ_i and b. Assume Q be the number of searched points during this process.

  - Step 2(b): Compute the precision P_i and speedup S_i by using the groundtruth information. If $P_i < P$, then proceed with a new value of ϵ_{i+1} and go to Step 2(a). Otherwise, go to Step 2(c).

  - Step 2(c): If $P_i \geq P$, we apply a binary search to find the optimized value of E_{max} in the range of $[0, Q]$. Particularly, we first set $E_{max} = Q/2$ and run the approximate search procedure. Next, the search range is updated as either $[0, \frac{Q}{2}]$ or $[\frac{Q}{2}, Q]$ taking into account that $P_i > P$ or $P_i \leq P$. This process continues until the search range is unit-length.
  - Step 2(d): Update the best speedup, the parameter ϵ_i, and the optimized parameter E_{max} obtained from Step 2(c). Proceed with a new value of ϵ_{i+1} and go to Step 2(a) to find the better solution.

- Step 3: Return the parameters ϵ_i and E_{max}, corresponding to the best speedup found so far.

The rationale of setting the parameter b to 1 was inspired from the work of multi-probe LSH [Lv et al., 2007]. When using the two highest variance axes for partitioning the data, two close points can be divided into two adjacent bins. It is thus necessary to have a look at the adjacent bins while exploring the tree. Our experiments have revealed that the parameter $b = 1$ is often a good setting for obtaining satisfactory search performance.

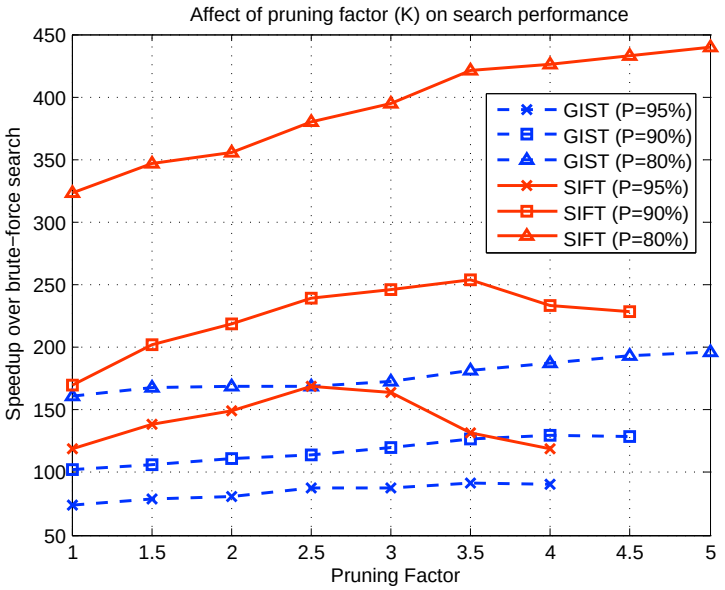


Figure 5.8: Search performance as a function of the pruning factor: precision is set to 95%, 90%, and 80% for both the SIFT and GIST features.

Figure 5.8 shows the search performance as a function of the pruning factor κ. In this experiment, we study the effect of κ for a wide range of search precision. For this purpose, the precision is set to 95%, 90%, and 80% for both the SIFT and GIST features. It can be seen that for both types of features, the speedup increases strongly with respect to the increase in κ to a certain extent. For example, taking the curve that corresponds to the precision $P = 90\%$ for the SIFT features, the speedup starts to decrease for $\kappa \geq 3.5$. Figure 5.8 also reveals that when increasing κ, we achieve much of the speedup for low precision

($P = 80\%$) compared with that for higher precision ($P \geq 90\%$). This relationship is referred to as the problem of over-pruning because a large number of branches has been eliminated as κ increases. Hence, the search process would miss some true answers. Fortunately, this matter can be resolved by using a cross-validation process [Muja and Lowe, 2009] during the tree construction to adaptively select an appropriate value for κ, when provided a specific precision and dataset. However, we have not employed this approach here, and we set $\kappa = 2.5$ for all of our experiments.

5.4 Application to image retrieval

In this section, an application to image retrieval is investigated using the proposed LM-tree indexing scheme. We carry out here a classical approach composing of three main steps: feature extraction, indexing, and retrieval. For this purpose, we selected a wide corpus set[3] of historical books containing ornamental graphics. The dataset, called "Lettrine", is composed of 23062 isolated graphical images that were extracted from old documents. Some examples of the drop caps in this dataset are shown in Figure 5.9.

Figure 5.9: Several example images of ornamental drop caps in the dataset.

Our challenge here is to prove that the proposed LM-tree indexing algorithm can be used in the context of a CBIR system. Hence, for the first step of feature extraction, we selected the GIST feature as it is a commonly used descriptor in the literature for image retrieval [Oliva and Torralba, 2001]. The GIST descriptor comprises a set of perceptual properties that represent the dominant spatial structure of a scene such as naturalness, openness, roughness, expansion, and ruggedness. The GIST descriptor has been widely used in the literature [Kulis and Grauman, 2009, Jégou et al., 2011] on image retrieval, scene recognition and classification because of its distinctiveness and efficiency. We used the original implementation of the 512-dimensional GIST descriptor that was provided by the authors at this address[4].

In order to use the GIST descriptor for image matching and retrieval, it is necessary to normalize the image size. This action was already addressed in the GIST descriptor's implementation by centering, cropping and resizing the image such that the normalized image preserves the size ratio of the original image. In our experiments, the common size

[3] http://www.bvh.univ-tours.fr/

[4] http://people.csail.mit.edu/torralba/code/spatialenvelope/

for normalization is set to 256×256. Among the 23062 ornamental graphics, 500 images were included in the query set, and the remaining served as the database set. Next, the GIST features are computed once for all of the images in an offline phase.

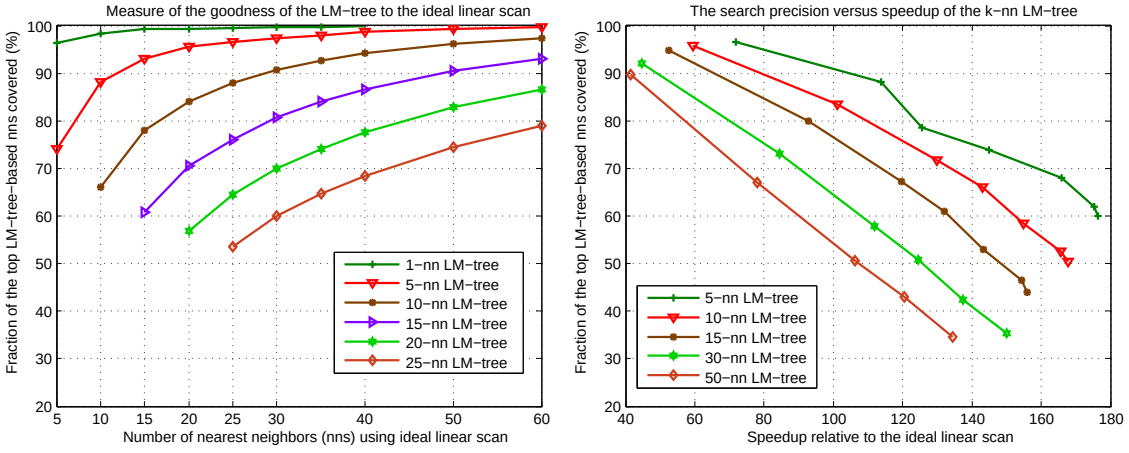
In the second step, all of the database GIST features are indexed by using our LM-tree algorithm. As this application is dedicated to the image retrieval domain, the main goal here is to illustrate that the system can produce sufficiently relevant results with a critical constraint of fast processing time. Therefore, it makes sense to apply an ANN indexing algorithm rather than the ENN method. Here, we keep the same parameter settings for the LM-trees as before. In other words, we used 8 parallel LM-trees, and each of which is associated with the same parameter configuration, as follows: $L_{max} = 10$, $L = 8$, $b = 1$, and $m = 6$.

For performance evaluation, it would be interesting to use standard evaluation metrics such as precision and recall. However, because there is no groundtruth information that is included in this dataset, the use of precision and recall is not possible. Instead, we used an alternative metric to quantify the retrieval performance of our system. This metric was introduced in [Kulis and Grauman, 2009] for the same purpose as ours. Its basic idea is to measure how well a retrieval system can approximate an ideal linear scan (i.e., a brute-force search). Specifically, we computed the fraction of the common answers between our retrieval system and the ideal linear scan to the number of answers of our system. Figure 5.10 (a) quantitatively shows how well our retrieval system approaches the ideal linear scan. These quantitative results are computed using the top 1, 5, 10, 15, 20, and 25 ranked nearest neighbors (NNs) of our system. To have a more detailed analysis of the results, we can derive hereafter several key remarks:

- For the top 5-NNs of the LM-trees, 93.1% of the time, the retrieved answers are covered by that of the top 15-NNs of the ideal linear scan.
- For the top 15-NNs of the LM-trees, 80.6% of the time, the retrieved answers are covered by that of the top 30-NNs of the ideal linear scan.
- For the top 25-NNs of the LM-trees, 74.4% of the time, the retrieved answers are covered by that of the top 50-NNs of the ideal linear scan.

The results presented in Figure 5.10 (a) quantitatively show the quality of our retrieval system, which can be regarded as a function of search precision over recall. Furthermore, we want to measure how fast the system is with respect to these results. For this purpose, Figure 5.10 (b) shows the speedup of our system relative to the ideal linear scan. In this test, both of the systems are evaluated by using the same parameter k for the number of nearest neighbors. For more detail, we extract hereafter several key results of our system:

- The 5-NNs LM-trees achieved a speedup factor of 113.8 relative to the 5-NNs ideal linear scan, given a precision of 88.1%.
- The 15-NNs LM-trees achieved a speedup factor of 92.8 relative to the 15-NNs ideal linear scan, given a precision of 80.1%.
- The 50-NNs LM-trees achieved a speedup factor of 78.2 relative to the 50-NNs ideal linear scan, given a precision of 67.1%.



(a) (b)

Figure 5.10: Search quality of the k-NN LM-trees (a), and search quality versus speedup of the k-NN LM-trees (b).

With regard to the results presented in Figure 5.10 (b), we report in Table 5.1 the absolute search time (ms) and the fraction of searched points for the case of the 5-NNs LM-trees, which were averaged over the 500 queries. Taking a search precision of 73.9% for example, our system must explore 0.78% of the whole database and takes only 0.3 (ms) to return the top 5-NN answers.

Table 5.1: Report of search time and fraction of searched points for the 5-NN LM-trees.

Performance	Search precision (%)						
	59.9	61.8	68.1	73.9	78.6	88.1	96.5
Search fraction (%)	0.27	0.32	0.52	0.78	1.17	2.45	6.21
Mean search time (ms)	0.24	0.25	0.26	0.30	0.36	0.38	0.60

Figure 5.11 shows some examples of our retrieval results in comparison with the ideal linear scan. These retrieval results are obtained using the 5-NN LM-trees that are associated with the precision of 78.6% (see Table 5.1).

5.5 Discussion

In this chapter, a novel and efficient indexing algorithm in feature vector space has been presented. Three main features are attributed to the proposed LM-tree. First, a new polar-space-based method of data decomposition has been presented to construct the LM-tree. This new decomposition method differs from the existing methods in the literature in that the two highest variance axes of the underlying dataset are employed to iteratively partition the data. Second, a novel pruning rule is proposed to quickly eliminate the search paths that are unlikely to contain good candidates of the nearest

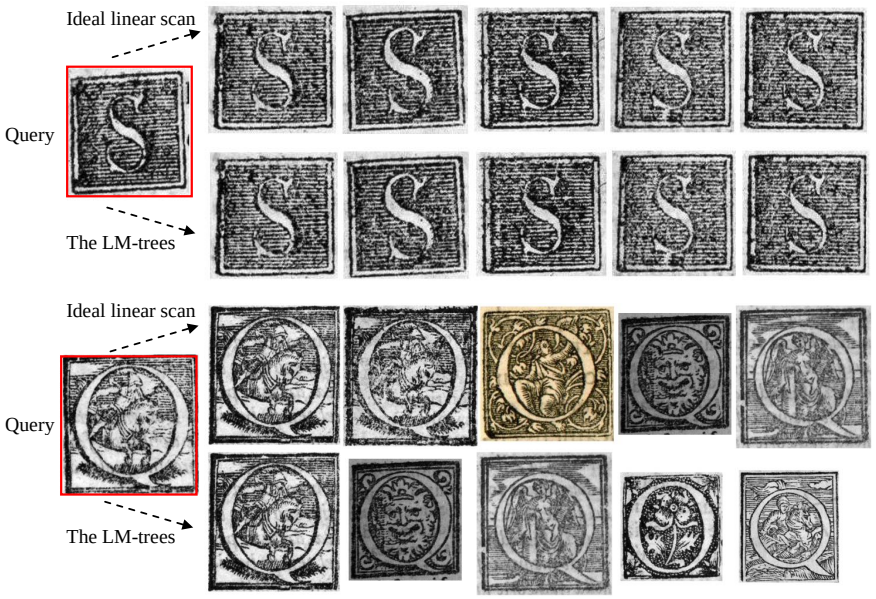


Figure 5.11: Few examples of the retrieval results: for each query in the left, the top ranked 5-NNs are showed for the ideal linear scan (the top row) and our LM-trees (the bottom row).

neighbors. Furthermore, the lower bounds are easily computed, as if the data were in 2D space, regardless of how high the dimensionality is. Finally, a bandwidth search method is presented to explore the nodes of the LM-tree. Its basic idea is inspired by the fact that only a limited number of relevant bins are searched to avoid the overhead of complexity. The proposed LM-tree has been validated on one million SIFT features and 250000 GIST features, demonstrating that it works very well for both ENN and ANN search, compared to the state-of-the-art indexing algorithms, including randomized KD-trees, hierarchical K-means tree, randomized clustering trees, and the multi-probe LHS scheme.

For further improvements to this work, more experiments on binary features (e.g., BRIEF [Calonder et al., 2010], ORB [Rublee et al., 2011]) would be interesting to evaluate the proposed LM-tree. Dynamic insertion and deletion of the data points in the LM-tree would be also investigated to make the system adaptive to dynamic changes in the data. Automatic parameter tuning using cross-validation approach could be also integrated to make the system more robust and well-fitting to specific datasets. The use of many high variance axes (e.g., more than two) could be considered to study the new behaviors of the system. In addition to these open works, the study of designing an indexing system, which can work on the data stored on an external disk, will be investigated to deal with extremely large datasets in which they are often not fully loaded into the main memory.

Conclusions

In this chapter, we summarize the main contributions of this dissertation for junction detection in line-drawing images and feature indexing in high-dimensional feature vector space. The favourable features and the shortcomings of the two contributions are carefully discussed. Possible lines of future research for each of these contributions are also given.

Beginning with the chapter 1, we have provided the description of the state-of-the-art methods for junction detection. The merits and limitations of each method are discussed in detail. The connections among these methods are studied to establish potential links to our contribution on junction detection.

Based on the discussion of the existing approaches for junction detection, a novel approach is presented in chapter 2 for robust and accurate junction detection and characterization in line-drawing images. The proposed approach has been evaluated through extensive experiments in which we achieved better results compared to other baseline methods. Besides, the computation complexity of the proposed system has been analyzed from the theoretical point of view, showing that its complexity is essentially linear to the image size. An application of symbol localization has been also investigated, confirming very good results in terms of both detection rate and efficiency. In short, the proposed approach has the following major features:

- Junction distortion avoidance: The problem of junction distortion is avoided by completely removing all distorted zones from foreground. For this purpose, an efficient algorithm is presented to detect and remove crossing regions. In this way, the proposed approach works on the remaining line segments only.
- Accurate junction detection: A new algorithm based on linear least square technique is presented to correctly determine local scales (or regions of support) of every median point. Junction localization is achieved by a novel optimization algorithm which analyzes the local structure relations among the characterized line segments to produce precise junction detection.

- Multiple junction detection: The proposed approach can deal with the problem of multiple detection of junction points for a given complicated crossing zone. This was accomplished by iteratively clustering the incident branches into different groups, each of which forms an optimized location of the junction.
- Efficiency: Computation complexity for the whole process of junction detection is $O(MN + k^2S)$ where M, N are the dimensions of input image, S is the number of median points, and k is a small constant value. In addition, the detected junctions are characterized, classified, and matched in very efficient way that makes them suitable to deal with high complexity problems such as image indexing, symbol recognition, and symbol spotting.
- Robustness: The junction detector is stable to common transformations such as rotation, scaling, translation, and can resist a satisfactory level of degradation/noise. Furthermore, the proposed approach requires no prior knowledge about document content and is independent on any vectorization systems.
- Usefulness: The detected junctions can be used to address different applications of symbol localization and spotting, vectorization, and engineering document retrieval. As the junction detector is robust and accurate, the information of the detected junctions can be used to support different tasks including junction matching, geometry consistency checking, and primitive detection. Such favourable features are highly expected to address the time-critical applications.

In addition to these positive characteristics, we are also aware of several shortcomings of the proposed approach. First, as this approach is dedicated to working with line-like primitives, its performance would be degraded if applying to filled-shape objects, such as logo images. The reason is due to the fact that median lines are not an appropriate means for representing these filled shapes. For the filled shapes, as the local line-thickness at each crossing zone is often high, a large part of median lines is eliminated around the crossing zones. For this reason, we have not enough information to reconstruct the junctions. Second, although we have improved the stage of region of support determination to make it more robust to the digitalization effect, the step of dominant point detection is still dependent on the threshold of deciding a low curvature point. Furthermore, the junction optimization process could lead to some difficulties in correctly interpreting the junction position as originally produced by craftsmen. However, although this point is valid for some specific domains of exact line-drawing representation, such as vectorization, we are interested in detecting local features that would be useful to addressing the problem of large-scale document indexing and retrieval. In this sense, a low rate of false positives in the final results is not problematic. To promote the evaluation of this work, the source code and demonstration of the symbol localization application are publicly available at this address[5]. Furthermore, several potential directions of research have planned for improvement of this work:

- It can be noted that the junction optimization algorithm is dependent on the detection of junction candidates. Any false detections of the junction candidates could

[5]https://sites.google.com/site/ourjunctiondemo/

result in the false acceptance of the final junctions. A potential solution for this matter would probably rely on the context information. By incorporating some null hypothesis H_0 of the underlying context information, the powerful *a contrario* detection theory [Desolneux et al., 2000] can be applied to justify the meaningfulness of a given detected junction. That means, a junction is meaningful if it is unlikely to occur at random under the hypothesis H_0. For instance, a successful application of this theory for detecting the junctions in natural images was presented in [Xia, 2011].

- To this end, the evaluation protocol applied to the junction detectors is detector-dependent. Different evaluation metrics and protocols shall be used to study the behavior of the proposed junction detector. It would be also interesting to evaluate the junction detectors using some datasets provided with semantic groundtruth such as the BSDS benchmark presented in [Maire et al., 2008].

- Junction characterization has been addressed in our work as a natural benefit of the junction optimization process. It has several favourable properties of simplicity, distinctiveness, scale invariant, and low dimensionality. However, this characterization is not shift invariant (i.e., it depends on the selection of the start junction arm). Therefore, further works to address these points would make the whole system completely robust. Besides, the idea of employing an off-the-shelf local descriptor in the CV field would be also interesting to characterize the junctions.

Regarding the second contribution of this dissertation about feature indexing, we first provided, in chapter 4, a deep review of the state-of-the-art methods for feature indexing in high-dimensional feature vector space. The main ideas, favourable features, and shortcomings of each method are thoroughly studied. We also give our subjection remarks for these methods and highlight the need of an advanced contribution for efficiently indexing the feature vectors.

Following the conclusions derived from the discussion of the existing indexing methods, a new contribution for feature indexing in high-dimensional feature vector space has been presented in chapter 5. The proposed algorithm, called linked-node m-ary tree (LM-tree), has many desirable features that make it different from all the existing methods. Followings are three main advancements attributed to the proposed LM-tree.

- A new polar-space-based method for data decomposition has been presented to construct the LM-tree. This new decomposition method differs from the existing methods in that two axes are randomly selected from a few axes that have the highest variance of the underlying dataset to iteratively partition the data.

- An efficient pruning rule is proposed to eliminate the search paths that are unlikely to contain the true answers. Furthermore, the lower bounds are easily computed, as if the data were in 2D space, regardless of how high the dimensionality is.

- A bandwidth search method is introduced to explore the nodes of the LM-tree. Its basic idea is inspired by the fact that searching at multiple adjacent bins gives a good chance of reaching the true answers, while reducing the computational overhead. Like this, it makes unnecessary the expensive computations of finding the best bins that is a matter of the priority search technique [Beis and Lowe, 1997].

The proposed LM-tree has been validated on a wide corpus dataset composing of one million SIFT features and 250000 GIST features, demonstrating that the proposed algorithm gives a significant improvement of search performance, compared to the state-of-the-art indexing algorithms, including randomized KD-trees, hierarchical K-means tree, randomized clustering trees, and multi-probe LHS scheme. To further confirm the efficiency the proposed indexing algorithm, an additional application to image retrieval was developed using a wide corpus set of historical books containing ornamental graphics. Performance evaluation has been reported, showing that the LM-tree indexing algorithm is very time-efficient. At last, the source code of this contribution is also made available for the interest of researchers at this address[6].

Notwithstanding the obtained results are interesting, we realize that many different lines of researches for this work are still possible. Followings are several improvements planned in the future work.

- To this end, the proposed LM-tree is a balance but static tree. That said, our indexing algorithm would not work for the cases where the data are dynamically changed over time. Accordingly, dynamic insertion and deletion of the data points in the LM-tree would be also investigated in the future to make the system adaptive to the dynamic change of the data.
- As the study of the binary features is more and more becoming a great interest of researches, different tests on binary features would be interesting to evaluate the proposed LM-tree.
- It is agreed that performance of an indexing algorithm is highly dependent on particular dataset. It is therefore a good idea to train the system on a particular dataset provided some prior knowledge about the desired search precision. As already reported in [Muja and Lowe, 2009], automatic parameter tuning using cross-validation approach would be integrated to make the system more robust and well-fitting to specific datasets.
- The use of many high variance axes for data partitioning (e.g., more than two) would be considered to study the new behavior of the system.
- In addition to these open works, the study of designing an indexing system, which can work on the data stored on external disk, will be investigated to deal with extremely large datasets that are not able to be fully loaded into the main memory.

As the last conclusion, we expected that the two contributions in this dissertation shall draw much of interest and attention from the research community.

[6] https://sites.google.com/site/ptalmtree/

Liste des publications et prix

Revues internationales avec comité de lecture

- The Anh Pham, Mathieu Delalandre, Sabine Barrat and Jean-Yves Ramel, "*Accurate junction detection and characterization in line-drawing images*". Pattern Recognition (2013), In Press. DOI: http://dx.doi.org/10.1016/j.patcog.2013.06.027
- The Anh Pham, Sabine Barrat, Mathieu Delalandre and Jean-Yves Ramel, "*An efficient tree structure for indexing feature vectors*". Invited submission to a special section of Pattern Recognition Letters (2013).

Conférences internationales avec comité de lecture et actes

- The Anh Pham, Sabine Barrat, Mathieu Delalandre and Jean-Yves Ramel, "*An efficient indexing scheme based on linked-node m-ary tree structure*". In proceedings of the 17th International Conference on Image Analysis and Processing (ICIAP 2013), A. Petrosino (Ed.): Part I, LNCS 8156, pp. 752–762, 2013.
- The Anh Pham, Mathieu Delalandre, Sabine Barrat and Jean-Yves Ramel, "*Robust symbol localization based on junction features and efficient geometry consistency checking*". In proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2013), pp. 1083–1087, 2013.
- The Anh Pham, Mathieu Delalandre, Sabine Barrat and Jean-Yves Ramel, "*Accurate Junction Detection and Reconstruction in Line-Drawing Images*", In proceedings of the 21st International Conference on Pattern Recognition (ICPR 2012), pp. 693–696, 2012.
- The Anh Pham, Mathieu Delalandre, Sabine Barrat and Jean-Yves Ramel, "*A robust approach for local interest point detection in line-drawing images*", In proceedings of the 10th IAPR International Workshop on Document Analysis Systems (DAS 2012), pp. 79–84, 2012.
- The Anh Pham, Mathieu Delalandre and Sabine Barrat, "*A contour-based method for logo detection*", In proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011), pp. 718–722, 2011.

Conférence nationale avec comité de lecture et actes

- The Anh Pham, Sabine Barrat, Mathieu Delalandre and Jean-Yves Ramel, "*Une approche robuste pour la détection de points d'intérêts dans les images de documents techniques*", Colloque International Francophone sur l'Écrit et le Document 2012 (CIFED), pp. 445-460, March 21th-23th, 2012, Bordeaux, France.

Prix

- "IAPR Best Student Paper Award" for the paper: The Anh Pham, Sabine Barrat, Mathieu Delalandre and Jean-Yves Ramel, "*An efficient indexing scheme based on linked-node m-ary tree structure*". In proceedings of the 17th International Conference on Image Analysis and Processing (ICIAP 2013), A. Petrosino (Ed.): Part I, LNCS 8156, pp. 752–762, 2013

Bibliography

[Aslan et al., 2008] Aslan, C., Erdem, A., Erdem, E., and Tari, S. (2008). Disconnected skeleton: Shape at its absolute scale. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(12):2188–2203.

[Auclair, 2009] Auclair, A. Vincent, N. C. L. (2009). Hash functions for near duplicate image retrieval. In *Workshop on Applications of Computer Vision (WACV2009)*, pages 1–6.

[Awrangjeb and Lu, 2008] Awrangjeb, M. and Lu, G. (2008). An improved curvature scale-space corner detector and a robust corner matching approach for transformed image identification. *IEEE Trans. Image Process.*, 17(12):2425–2441.

[Bai et al., 2007] Bai, X., Latecki, L. J., and Liu, W.-Y. (2007). Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):449–462.

[Ballard, 1981] Ballard, D. (1981). Generalizing the hough transform to detect arbitrary patterns. *Communications of the ACM*, 13(2):111–122.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359.

[Beckmann et al., 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The r*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2):322–331.

[Beis and Lowe, 1997] Beis, J. S. and Lowe, D. G. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, CVPR'97, pages 1000–1006.

[Belongie et al., 2002] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522.

[Berchtold et al., 1996] Berchtold, S., Keim, D. A., and Kriegel, H.-P. (1996). The x-tree: An index structure for high-dimensional data. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB'96, pages 28–39.

[Bergevin and Bubel, 2004] Bergevin, R. and Bubel, A. (2004). Detection and characterization of junctions in a 2d image. *Comput. Vis. Image Underst.*, 93(3):288–309.

[Biederman, 1986] Biederman, I. (1986). Human image understanding: recent research and a theory. In *The second workshop on Human and Machine Vision II*, volume 13, pages 13–57.

[Bodic et al., 2009] Bodic, P. L., Locteau, H., Adam, S., Heroux, P., Lecourtier, Y., and Knippel, A. (2009). Symbol detection using region adjacency graphs and integer linear programming. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1320–1324.

[Böhm et al., 2001] Böhm, C., Berchtold, S., and Keim, D. A. (2001). Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373.

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 778–792.

[Carmona-Poyato et al., 2005] Carmona-Poyato, A., Fernández-García, N. L., Medina-Carnicer, R., and Madrid-Cuevas, F. J. (2005). Dominant point detection: A new proposal. *Image Vision Comput.*, 23(13):1226–1236.

[Chávez et al., 2001] Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. (2001). Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321.

[Cheng et al., 1984] Cheng, D.-Y., Gersho, A., Ramamurthi, B., and Shoham, Y. (1984). Fast search algorithms for vector quantization and pattern matching. In *The IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'84)*, volume 9, pages 372–375.

[Chiang et al., 1998] Chiang, J. Y., Tue, S., and Leu, Y. C. (1998). A new algorithm for line image vectorization. *Pattern Recognition*, 31(10):1541–1549.

[Choi et al., 2009] Choi, S., Kim, T., and Yu, W. (2009). Performance evaluation of ransac family. In *Proceedings of British Machine Vision Conference*, BMVC'09, pages 1–12.

[Damiand et al., 2009] Damiand, G., Higuera, C., Janodet, J.-C., Samuel, E., and Solnon, C. (2009). A polynomial algorithm for submap isomorphism. In *Graph-Based Representations in Pattern Recognition*, volume 5534 of *Lecture Notes in Computer Science*, pages 102–112.

[Delalandre et al., 2010] Delalandre, M., Ramel, J.-Y., Valveny, E., and Luqman, M. (2010). A performance characterization algorithm for symbol localization. In *Graphics Recognition. Achievements, Challenges, and Evolution*, volume 6020 of *LNCS*, pages 260–271.

[Delalandre et al., 2008] Delalandre, M., Valveny, E., and Llados, J. (2008). Performance evaluation of symbol recognition and spotting systems: An overview. In *Workshop on Document Analysis Systems (DAS)*, pages 497–505.

[Deschênes and Ziou, 2000] Deschênes, F. and Ziou, D. (2000). Detection of line junctions and line terminations using curvilinear features. *Pattern Recogn. Lett.*, 21(6–7):637–649.

[Deseilligny et al., 1998] Deseilligny, M. P., Stamon, G., and Suen, C. Y. (1998). Veinerization: A new shape description for flexible skeletonization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):505–521.

[Desolneux et al., 2000] Desolneux, A., Moisan, L., and Morel, J.-M. (2000). Meaningful alignments. *Int. J. Comput. Vision*, 40(1):7–23.

[di Baja, 1994] di Baja, G. S. (1994). Well-shaped, stable, and reversible skeletons from the (3,4)-distance transform. *J. Vis. Commun. Image R.*, 5(1):107–115.

[Dori and Liu, 1999] Dori, D. and Liu, W. (1999). Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(3):202–215.

[Dosch and LLados, 2004] Dosch, P. and LLados, J. (2004). Vectorial signatures for symbol discrimination. In *Workshop on Graphics Recognition (GREC), LNCS*, volume 3088, pages 154–165.

[Douglas and Peucker, 1973] Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

[Dutta et al., 2013a] Dutta, A., LladóS, J., Bunke, H., and Pal, U. (2013a). Near convex region adjacency graph and approximate neighborhood string matching for symbol spotting in graphical documents. In *Proceedings of 12th International Conference on Document Analysis and Recognition (ICDAR13)*, pages 1078–1082.

[Dutta et al., 2011] Dutta, A., Llados, J., and Pal, U. (2011). Symbol spotting in line drawings through graph paths hashing. In *International Conference on Document Analysis and Recognition (ICDAR)*.

[Dutta et al., 2013b] Dutta, A., LladóS, J., and Pal, U. (2013b). A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recognition*, 46(3):752–768.

[Embley et al., 2011] Embley, D. W., Machado, S., Packer, T., Park, J., Zitzelberger, A., Liddle, S. W., Tate, N., and Lonsdale, D. W. (2011). Enabling search for facts and implied facts in historical documents. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, pages 59–66.

[Faas and van Vliet, 2007] Faas, F. G. A. and van Vliet, L. J. (2007). Junction detection and multi-orientation analysis using streamlines. In *Proceedings of the 12th international conference on Computer analysis of images and patterns*, CAIP'07, pages 718–725.

[Fan et al., 1998] Fan, K.-C., Chen, D.-F., and Wen, M.-G. (1998). Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching. *Pattern Recognition*, 31(7):823–838.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[Förstner, 1994] Förstner, W. (1994). A framework for low level feature extraction. In *Proceedings of the third European conference on Computer Vision (Vol. II)*, ECCV'94, pages 383–394.

[Förstner and Gülch, 1987] Förstner, W. and Gülch, E. (1987). A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Intercomission Conference on Fast Proc. of Photogrammetric Data*, pages 281–305.

[Friedman et al., 1977] Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226.

[Fukunaga and Narendra, 1975] Fukunaga, K. and Narendra, M. (1975). A branch and bound algorithm for computing k-nearest neighbors. *IEEE Trans. Comput.*, 24(7):750–753.

[Gionis et al., 1999] Gionis, A., Indyk, P., and Motwani, R. (1999). Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large DataBases*, VLDB'99, pages 518–529.

[Graves et al., 2009] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868.

[Guha et al., 1998] Guha, S., Rastogi, R., and Shim, K. (1998). Cure: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 73–84.

[Guttman, 1984] Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. *ACM SIGMOD Record*, 14(2):47–57.

[Han and Fan, 1994] Han, C.-C. and Fan, K.-C. (1994). Skeleton generation of engineering drawings via contour matching. *Pattern Recognition*, 27(2):261–275.

[Hansen and Neumann, 2004] Hansen, T. and Neumann, H. (2004). Neural mechanisms for the robust representation of junctions. *Neural Comput.*, 16(5):1013–1037.

[Harris and Stephens., 1988] Harris, C. and Stephens., M. (1988). A combined corner and edge detector. *In Alvey Vision Conference*, pages 147–151.

[Heitger, 1995] Heitger, F. (1995). Feature detection using suppression and enhancement. Technical Report TR-163, Image Science Lab.

[Hilaire and Tombre, 2001] Hilaire, X. and Tombre, K. (2001). Improving the accuracy of skeleton-based vectorization. In *Proceedings of the 4th International Workshop GREC'01, LNCS*, volume 2390, pages 273–288.

[Hilaire and Tombre, 2006] Hilaire, X. and Tombre, K. (2006). Robust and accurate vectorization of line drawings. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(6):890–904.

[Hori and Tanigawa, 1993] Hori, O. and Tanigawa, S. (1993). Raster-to-vector conversion by line fitting based on contours and skeletons. In *Proceedings of International Conference of Document Image Analysis (ICDAR1993)*, pages 353–358.

[Indyk and Motwani, 1998] Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC'98, pages 604–613.

[Jain et al., 2000] Jain, A., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: a review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):4–37.

[Jain and Doermann, 2012] Jain, R. and Doermann, D. (2012). Logo retrieval in document images. In *IAPR International Workshop on Document Analysis Systems*, pages 135–139.

[Janssen and Vossepoel, 1997] Janssen, R. D. T. and Vossepoel, A. M. (1997). Adaptive vectorization of line drawing images. *Comput. Vis. Image Underst.*, 65(1):38–56.

[Jégou et al., 2011] Jégou, H., Douze, M., and Schmid, C. (2011). Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128.

[Jouili et al., 2010] Jouili, S., Coustaty, M., Tabbone, S., and Ogier, J. (2010). Navidomass: Structural-based approaches towards handling historical documents. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 946–949.

[Jouili and Tabbone, 2012] Jouili, S. and Tabbone, S. (2012). Hypergraph-based image retrieval for graph-based representation. *Pattern Recognition*, 45(11):4054–4068.

[Kalkan et al., 2007] Kalkan, S., Yan, S., Pilz, F., and Krüger, N. (2007). Improving junction detection by semantic interpretation. In *Proceedings of the Second International Conference on Computer Vision Theory and Applications*, pages 264–271.

[Kassim et al., 1999] Kassim, A., Tan, T., and Tan, K. (1999). A comparative study of efficient generalised hough transform techniques. *Image and Vision Computing*, 17(10):737–748.

[Katayama and Satoh, 1997] Katayama, N. and Satoh, S. (1997). The sr-tree: an index structure for high-dimensional nearest neighbor queries. *ACM SIGMOD Record*, 26(2):369–380.

[Kim, 2005] Kim, I.-J. (2005). New chances and challenges in camera-based document analysis and recognition. In *Keynote Presentation from First International Workshop on Camera-Based Document Analysis and Recognition (CBDAR2005).*

[K.Mikolajczyk and Schmid, 2001] K.Mikolajczyk and Schmid, C. (2001). Indexing based on scale invariant interest points. *In Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, pages 525–531.

[Kong et al., 2011] Kong, X., Valveny, E., Sanchez, G., and Wenyin, L. (2011). Symbol spotting using deformed blurred shape modeling with component indexing and voting scheme. In *Workshop on Graphic Recognition (GREC)*.

[Köthe, 2003] Köthe, U. (2003). Integrated edge and junction detection with the boundary tensor. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, volume 2, pages 424–431.

[Kulis and Grauman, 2009] Kulis, B. and Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8.

[Kwok, 1988] Kwok, P. (1988). A thinning algorithm by contour generation. *Commun. ACM*, 31(11):1314–1324.

[Laganiere, 2004] Laganiere, R. (2004). The detection of junction features in images. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, volume 3, pages 573–579.

[Laganiere and Elias, 2004] Laganiere, R. and Elias, R. (2004). The detection of junction features in images. In *Proceedings of Acoustics, Speech, and Signal Processing (ICASSP'04)*, pages 573–577.

[Lam et al., 1992] Lam, L., Lee, S.-W., and Suen, C. Y. (1992). Thinning methodologies-a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):869–885.

[Landon, 2013] Landon, G. V. (2013). Automatic photometric restoration of historical photographic negatives. In *5th International Workshop on Camera-Based Document Analysis and Recognition (CBDAR2013)*.

[Lee and Wu, 1998] Lee, C. and Wu, B. (1998). A chinese-character-stroke-extraction algorithm based on contour information. *Pattern Recognition*, 31(6):651–663.

[Lee and Wong, 1977] Lee, D. T. and Wong, C. K. (1977). Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, 9(1):23–29.

[Leibe et al., 2006] Leibe, B., Mikolajczyk, K., and Schiele, B. (2006). Efficient clustering and matching for object class recognition. In *Proceedings of British Machine Vision Conference*, pages 789–798.

[Liang et al., 2005] Liang, J., Doermann, D., and Li, H. (2005). *International Journal of Document Analysis and Recognition (IJDAR)*, 7(2-3):84–104.

[Lin and Tang, 2002] Lin, F. and Tang, X. (2002). Off-line handwritten chinese character stroke extraction. In *16th International Conference on Pattern Recognition*, volume 3, pages 249–252.

[Lins et al., 2011] Lins, R. D., de F. Pereira e Silva, G., and de A. Formiga, A. (2011). Histdoc v. 2.0: enhancing a platform to process historical documents. In *Proceedings*

of the 2011 Workshop on Historical Document Imaging and Processing, HIP '11, pages 169–176.

[Liu et al., 1999] Liu, K., Huang, Y., and Suen, C. Y. (1999). Identification of fork points on the skeletons of handwritten chinese characters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(10):1095–1100.

[Liu et al., 2004] Liu, T., Moore, A. W., Gray, A., and Yang, K. (2004). An investigation of practical approximate nearest neighbor algorithms. In *Proceedings of Neural Information Processing Systems (NIPS 2004)*, pages 825–832. MIT Press.

[Liwicki et al., 2011] Liwicki, M., Malik, M. I., van den Heuvel, C. E., Chen, X., Berger, C., Stoel, R., Blumenstein, M., and Found, B. (2011). Signature verification competition for online and offline skilled forgeries (sigcomp2011). In *Proceedings of International Conference of Document Image Analysis*, pages 1480–1485.

[Llados et al., 2001] Llados, J., Marti, E., and Villanueva, J. (2001). Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(10):1137–1143.

[LLados et al., 2002] LLados, J., Valveny, E., Sanchez, G., and Marti, E. (2002). Symbol recognition : Current advances and perspectives. In *Workshop on Graphics Recognition (GREC), LNCS*, volume 2390, pages 104–127.

[Lowe, 2004] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Luqman, 2012] Luqman, M. M. (2012). *Fuzzy Multilevel Graph Embedding for Recognition, Indexing and Retrieval of Graphic Document Images*. PhD thesis, Laboratory of Computer Science (EA 6300), Francois Rabelais University, Tours city, France.

[Lv et al., 2007] Lv, Q., Josephson, W., Wang, Z., Charikar, M., and Li, K. (2007). Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd International Conference on Very large Databases*, VLDB'07, pages 950–961.

[Maire et al., 2008] Maire, M., Arbelaez, P., Fowlkes, C., and Malik, J. (2008). Using contours to detect and localize junctions in natural images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8.

[McNames, 2001] McNames, J. (2001). A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(9):964–976.

[Messmer and Bunke, 1996] Messmer, B. and Bunke, H. (1996). Automatic learning and recognition of graphical symbols in engineering drawings. In *Workshop on Graphics recognition (GREC), LNCS*, volume 1072, pages 123–134.

[Mikolajczyk and Matas, 2007] Mikolajczyk, K. and Matas, J. (2007). Improving descriptors for fast tree matching by optimal linear projection. In *IEEE 11th International Conference on Computer Vision (ICCV 2007)*, pages 1–8.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630.

[Mohammad Awrangjeb and Fraser, 2012] Mohammad Awrangjeb, G. L. and Fraser, C. S. (2012). Performance comparisons of contour-based corner detectors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 135(9):4167–4179.

[Mokhtarian and Suomela, 1998] Mokhtarian, F. and Suomela, R. (1998). Robust image corner detection through curvature scale space. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1376–1381.

[Mordohai and Medioni, 2004] Mordohai, P. and Medioni, G. (2004). Junction inference and classification for figure completion using tensor voting. In *Proceedings of Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, pages 56–64.

[Mori et al., 2001] Mori, G., Belongie, S., and Malik, J. (2001). Shape contexts enable efficient retrieval of similar shapes. In *Proceedings of the IEEE Computer Society Computer Vision and Pattern Recognition (CVPR'01)*, volume 1, pages 723–730.

[Morris, 2004] Morris, T. (2004). *Computer Vision and Image Processing.* Palgrave Macmillan, ISBN 0-333-99451-5.

[Muja and Lowe, 2009] Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340.

[Muja and Lowe, 2012] Muja, M. and Lowe, D. G. (2012). Fast matching of binary features. In *Proceedings of the Ninth Conference on Computer and Robot Vision (CRV)*, pages 404–410.

[Näf et al., 1997] Näf, M., Székely, G., Kikinis, R., Shenton, M. E., and Kübler, O. (1997). 3d voronoi skeletons and their usage for the characterization and recognition of 3d organ shape. *Comput. Vis. Image Underst.*, 66(2):147–161.

[Nagasamy and Langrana, 1990] Nagasamy, V. and Langrana, N. A. (1990). Engineering drawing processing and vectorization system. *Comput. Vision Graph. Image Process.*, 49(3):379–397.

[Nayef and Breuel, 2011] Nayef, N. and Breuel, T. (2011). On the use of geometric matching for both: Isolated symbol recognition and symbol spotting. In *Workshop on Graphics Recognition (GREC).*

[Nguyen et al., 2009] Nguyen, T., Tabbone, S., and Boucher, A. (2009). A symbol spotting approach based on the vector model and a visual vocabulary. In *International Conference on Document Analysis and Recognition (ICDAR'09)*, pages 708–712.

[Niblack et al., 1990] Niblack, C., Capson, D., and Gibbons, P. (1990). Generating skeletons and centerlines from the medial axis transform. In *Proceedings of the 10th International Conference on Pattern Recognition (ICPR 1990)*, pages 881–885.

[Nister and Stewenius, 2006] Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168.

[Ogniewicz and Ilg, 1992] Ogniewicz, R. and Ilg, M. (1992). Voronoi skeletons: theory and applications. In *Proceedings of 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 1992)*, pages 63–69.

[Oliva and Torralba, 2001] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145–175.

[Panigrahy, 2006] Panigrahy, R. (2006). Entropy based nearest neighbor search in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA'06, pages 1186–1195.

[Parida et al., 1998] Parida, L., Geiger, D., and Hummel, R. (1998). Junctions: detection, classification, and reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(7):687–698.

[Parvez and Mahmoud, 2013] Parvez, M. T. and Mahmoud, S. A. (2013). Offline arabic handwritten text recognition: A survey. *ACM Comput. Surv.*, 45(2):1–35.

[Phillips and Chhabra, 1999] Phillips, I. and Chhabra, A. (1999). Empirical performance evaluation of graphics recognition systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):849–870.

[Qureshi et al., 2008] Qureshi, R., Ramel, J., Barret, D., and Cardot, H. (2008). Spotting symbols in line drawing images using graph representations. In *Worksop on Graphics Recognition (GREC), LNCS*, volume 5406, pages 91–103.

[Ramel et al., 2000] Ramel, J.-Y., Vincent, N., and Emptoz, H. (2000). A structural representation for understanding line-drawing images. *International Journal on Document Analysis and Recognition*, 3(2):58–66.

[Reisfeld et al., 1995] Reisfeld, D., Wolfson, H., and Yeshurun, Y. (1995). Context free attentional operators: the generalized symmetry transform. *Int. J. Comput. Vision*, 14(2):119–130.

[Riesen et al., 2007] Riesen, K., Neuhaus, M., and Bunke, H. (2007). Graph embedding in vector spaces by means of prototype selection. In *Proceedings of the 6th IAPR-TC-15 international conference on Graph-based representations in pattern recognition*, GbRPR'07, pages 383–393.

[Rosenfeld, 1975] Rosenfeld, A. (1975). A characterization of parallel thinning algorithms. *Information and Control*, 29(3):286–291.

[Roy et al., 2012] Roy, P. P., Rayar, F., and Ramel, J.-Y. (2012). An efficient coarse-to-fine indexing technique for fast text retrieval in historical documents. In *Proceedings of the 10th IAPR International Workshop on Document Analysis Systems*, pages 150–154.

[Rubin, 2001] Rubin, N. (2001). The role of junctions in surface completion and contour matching. *Perception*, 30(3):339–366.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Proceedings of 2011 IEEE International Conference on Computer Vision*, ICCV'11, pages 2564–2571.

[Rusinol et al., 2013] Rusinol, M., Karatzas, D., and LLados, J. (2013). Spotting graphical symbols in camera-acquired documents in real time. In *Proceedings of the 10th IAPR International Workshop on Graphics Recognition, GREC 2013*.

[Rusinol and LLados, 2006] Rusinol, M. and LLados, J. (2006). Symbol spotting in technical drawings using vectorial signatures. In *Workshop on Graphics Recognition (GREC), LNCS*, volume 3926, pages 35–46.

[Rusinol and Llados, 2009a] Rusinol, M. and Llados, J. (2009a). Logo spotting by a bag-of-words approach for document categorization. *10th International Conference on Document Analysis and Recognition*, pages 111–115.

[Rusinol and Llados, 2009b] Rusinol, M. and Llados, J. (2009b). A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(2):83–96.

[Rusinol and Llados, 2010] Rusinol, M. and Llados, J. (2010). Efficient logo retrieval through hashing shape context descriptors. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS'10, pages 215–222.

[Rusinol et al., 2010] Rusinol, M., LLados, J., and Sanchez, G. (2010). Symbol spotting in vectorized technical drawings through a lookup table of region strings. *Pattern Anal. Appl.*, 13(3):321–331.

[Sellis et al., 1987] Sellis, T. K., Roussopoulos, N., and Faloutsos, C. (1987). The r+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, VLDB'87, pages 507–518.

[Shen et al., 2011] Shen, W., Bai, X., Hu, R., Wang, H., and Jan Latecki, L. (2011). Skeleton growing and pruning with bending potential ratio. *Pattern Recogn.*, 44(2):196–209.

[Silpa-Anan and Hartley, 2008] Silpa-Anan, C. and Hartley, R. (2008). Optimised kd-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8.

[Sluzek, 2001] Sluzek, A. (2001). A local algorithm for real-time junction detection in contour images. In *Proceedings of the 9th International Conference CAIP'01*, pages 465–472.

[Smith and Brady, 1995] Smith, S. M. and Brady, J. M. (1995). Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–87.

[Song et al., 2002] Song, J., Su, F., Tai, C. L., and Cai, S. (2002). An object-oriented progressive-simplification-based vectorization system for engineering drawings: Model, algorithm, and performance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(8):1048–1060.

[Tabbone et al., 2005] Tabbone, S. A., Alonso, L., and Ziou, D. (2005). Behavior of the laplacian of gaussian extrema. *J. Math. Imaging Vis.*, 23(1):107–128.

[Takeda et al., 2011] Takeda, K., Kise, K., and Iwamura, M. (2011). Real-time document image retrieval for a 10 million pages database with a memory efficient and stability improved llah. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1054–1058.

[Takeda et al., 2012] Takeda, K., Kise, K., and Iwamura, M. (2012). Real-time document image retrieval on a smartphone. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 225–229.

[Tanaka, 1995] Tanaka, E. (1995). Theoretical aspects of syntactic pattern recognition. *Pattern Recognition*, 28(7):1053 – 1061.

[Teh and Chin, 1989] Teh, C.-H. and Chin, R.-T. (1989). On the detection of dominant points on digital curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(8):859–872.

[Tombre, 2006] Tombre, K. (2006). Graphics recognition: The last ten years and the next ten years. In *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 of *Lecture Notes in Computer Science*, pages 422–426.

[Tuytelaars, 2007] Tuytelaars, T. & Mikolajczyk, K. (2007). Local invariant feature detectors: A survey. *Computer Graphics and Vision*, 3:177–280.

[Tuytelaars and Mikolajczyk, 2008] Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280.

[Vajda et al., 2011] Vajda, S., Rothacker, L., and Fink, G. A. (2011). A method for camera-based interactive whiteboard reading. In *4th International Workshop on Camera-Based Document Analysis and Recognition (CBDAR2011)*, pages 112–125.

[Valveny et al., 2011] Valveny, E., Delalandre, M., Raveaux, R., and Lamiroy, B. (2011). Report on the symbol recognition and spotting contest. In *Proceedings of the 9th International Workshop on Graphics Recognition (GREC'11), LNCS*, volume 7423, pages 198–207.

[Van Nieuwenhuizen and Bronsvoort, 1994] Van Nieuwenhuizen, Peter R. Kiewiet, O. and Bronsvoort, W. F. (1994). An integrated line tracking and vectorization algorithm. *Computer Graphics Forum*, 13(3):349–359.

[Wald and Havran, 2006] Wald, I. and Havran, V. (2006). On building fast kd-trees for ray tracing, and on doing that in o(nlogn). In *Proceedings of the 2006 IEEE symposium on interactive ray tracing*, pages 61–70.

[Ward and Hamarneh, 2010] Ward, A. and Hamarneh, G. (2010). The groupwise medial axis transform for fuzzy skeletonization and pruning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(6):1084–1096.

[Wenyin and Dori, 1997] Wenyin, L. and Dori, D. (1997). A protocol for performance evaluation of line detection algorithms. *Mach. Vision Appl.*, 9(5-6):240–250.

[Wenyin and Dori, 1998] Wenyin, L. and Dori, D. (1998). Genericity in graphics recognition algorithms. In *Proceedings of the Graphics Recognition: Algorithms and systems. Lecture notes in computer sciences*, pages 9–20.

[White and Jain, 1996] White, D. A. and Jain, R. (1996). Similarity indexing with the ss-tree. In *Proceedings of the 12th International Conference on Data Engineering*, ICDE'96, pages 516–523.

[Xia, 2011] Xia, G.-S. (2011). *Some Geometric Methods for the Analysis of Images and Textures*. PhD thesis, Télécom ParisTech (ENST).

[Yamamoto et al., 1999] Yamamoto, H., Iwasa, H., Yokoya, N., and Takemura, H. (1999). Content-based similarity retrieval of images based on spatial color distributions. In *10th International Conference on Image Analysis and Processing (ICIAP'99)*, pages 951–956.

[Yan and Wenyin, 2003] Yan, L. and Wenyin, L. (2003). Engineering drawings recognition using a case-based approach. In *International Conference on Document Analysis and Recognition (ICDAR'03)*, pages 1–5.

[Yang et al., 2011] Yang, P., Antonacopoulos, A., Clausner, C., and Pletschacher, S. (2011). Grid-based modelling and correction of arbitrarily warped historical document images for large-scale digitisation. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, pages 106–111.

[Yin et al., 2007] Yin, X.-C., Sun, J., Fujii, Y., Fujimoto, K., and Naoi, S. (2007). Perspective rectification for mobile phone camera-based documents using a hybrid approach to vanishing point detection. In *Second International Workshop on Camera-Based Document Analysis and Recognition (CBDAR2007)*.

[Zhu and Doermann, 2007] Zhu, G. and Doermann, D. (2007). Automatic document logo detection. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, volume 02 of *ICDAR'07*, pages 864–868.

[Zuwala and Tabbone, 2006] Zuwala, D. and Tabbone, S. (2006). A method for symbol spotting in graphical documents. In *Proceedings of the 7th international conference on Document Analysis Systems*, DAS'06, pages 518–528.