# Preface

## Geleitwort des Doktorvaters

Die Dissertation von Dr. Yll Haxhimusa setzt die Forschungen mit Bildpyramiden auf der Basis von Graphenstrukturen fort. Die Grundidee einer Bildpyramide ist die schrittweise Reduktion der Information eines digitalen Bildes um einen konstanten Faktor und die Stapelung dieser Bilder in einer sich logarithmisch zuspitzenden Pyramide. Die dadurch entstehenden regelmässigen Beziehungen der Pixel der verschiedenen Ebenen erlauben anschliessenden Prozessen, auch auf Details zuzugreifen, die durch die Reduktion auf höheren Ebenen verloren gingen. Diese Art der Pyramide hat Eingang in verschiedene Standards und zahlreiche effiziente Bildverarbeitungsmethoden gefunden.

Die Regelmässigkeit der Ebenenbeziehungen bewirkt leider, dass eine kleine Verschiebung oder Bewegung eines Objekts ganz neue Zugriffspfade erfordert ('Shiftvariance'). Daher wurden von Meer, Jolion, Montanvert und Rosenfeld Strukturen erforscht, die die Reduktion von der Lage eines Objektes im Bild abhängig machen. Daraus resultierte die zu einem Forschungsschwerpunkt am PRIP-Labor gewordene unregelmässige Graphenpyramide. Durch Einsatz der Operationen 'Kantenkontraktion' und 'Kantenelimination' kann das Bild hierarchisch mit in der Bildebene eingebettete Graphen beschrieben werden. Dies hat den Vorteil, dass die Entscheidung über Struktur und Zugriffspfade in der Pyramide von den in Attributen der Knoten und Kanten gespeicherten Bildinhalte sauber getrennt werden kann.

Die gewonnene Flexibilität in der Struktur hatte zur Folge, dass die logarithmische Länge der Zugriffspfade nicht immer gewährleistet werden konnte, manche Pyramiden wuchsen höher und produzierten zu lange Zugriffspfade. Hier setzt der erste **originale Beitrag von Dr. Haxhimusa** an: Das von Meer ursprünglich verwendete Verfahren des 'maximal independent vertex sets' (kurz MIS) wurde durch ein erweitertes 'maximal independent edge set' (MIES) ersetzt. Für diese Auswahl der 'überlebenden' Knoten und Kanten konnte gezeigt werden, dass die logarithmische Pyramidenhöhe nicht überschritten werden konnte. Egal wie die Kanten und Knoten für die Kontraktion ausgewählt werden, solange die Kanten den Bedingungen eines MIES entsprechen, wird die Anzahl der Knoten bei jedem Reduktionsschritt auf mindestens die Hälfte reduziert. Diese theoretische Erkenntnis wurde experimentell verifiziert und in der Arbeit eindrucksvoll dargestellt. Damit kann die Effizienz der Bildpyramide nun auch auf grundsätzlich beliebig aufgebauten Graphenpyramiden erreicht werden.

Eine zweite Strategie 'maximal independent directed edge set' (MIDES) berücksichtigt ausserdem die Richtung der zu kontrahierenden Kanten durch Einführung einer gerichteten Nachbarschaft. Diese Strategie erlaubt zusätzlich die Festlegung, welcher der Knoten eines Kontraktionskernes überleben soll. Der Beweis der Reduktion der Knoten auf die Hälfte stößt

hier auf das Problem, dass isolierte Knoten nicht ausgeschlossen werden können. Im Fall von isolierten Knoten werden jene Konfigurationen charakterisiert, die diese isolierten Knoten hervorrufen. Diese Konfigurationen treten in der Praxis äusserst selten auf, was sich in den ausgezeichneten experimentellen Ergebnissen ausdrückt.

Der **zweite Schwerpunkt der Arbeit** wendet die unregelmäßige Pyramide zur Segmentation von Bildern an. Auch hier folgt Dr. Haxhimusa nicht den ausgetretenen Pfaden, sondern hat den ersten Algorithmus zur Bestimmung eines Minimal Spanning Trees (MST) von Borůvka den Erfordernissen der dualen Graphenkontraktion in der Pyramide angepaßt. Kombiniert mit den Selektionskriterien von Felzenszwalb (interner und externer Kontrast) entstehen Hierarchien von Segmentationen, die in den Experimenten zumindest gleich gut abschneiden, wie die derzeit am häufigsten zitierte Methode des Normalized Cut von Shi und Malik. Dies wird ausführlich bewertet und mit Segmentationsresultaten von Menschen verglichen. Statistisch schneiden dabei die verglichenen Methoden etwa gleich gut ab. Qualitativ können die Ergebnisse der neuen Methode als sehr gut eingestuft werden, was durch Vergleich der Variation der Regionengrössen nachgewiesen wird.

Die Bedeutung der Resultate geht weit über die konkrete Anwendung auf digitale Bilder hinaus.

- Im Forschungsprojekt TWIST-CV ('Tracking With Structure in Computer Vision', FWF P18716-N13) untersuchen wir derzeit die Erweiterungsmöglichkeiten für das Verfolgen von Objekten (Tracking) auf Bildfolgen.

- In einer Zusammenarbeit mit Prof. Z. Pizlo steht die Plausibilität der Pyramide für die Erklärung der enormen Effizienz, die Menschen bei der Lösung NP-vollständiger Probleme wie dem 'Traveling Salesman Problem' (TSP) erbringen, im Zentrum.

- Die Erweiterung der Pyramide von zwei auf drei Dimensionen unter Einsatz von kombinatorischen Karten wurde in der von der Fakultät für Informatik ausgezeichneten Diplomarbeit von Herrn Dipl.Ing. Thomas Illetschko begonnen. Die Prinzipien sind dabei dieselben, allein die Datenmenge erreicht die Grenze der derzeit sinnvoll zu bearbeitenden Daten.

- Interessante topologische Eigenschaften der unregelmässigen Pyramiden sind Gegenstand einer weiteren Forschungsrichtung. Erste Experimente lassen vermuten, dass die mit den beschriebenen Prinzipien aufgebauten Pyramiden die Homologiegruppen der zugrundeliegenden Daten erhalten. Das bedeutet, dass die riesigen Datenmengen auf eine kleine Menge reduziert werden können, die dieselben topologischen Eigenschaften hat wie das ursprüngliche Bild. Es ist daher möglich, topologische Eigenschaften mit hochkomplexen Algorithmen auf einer kleinen Datenmenge zu bestimmen und sie dann mit Hilfe der logarithmischen Zugriffspfade effizient auf die Originaldaten abzubilden.

Die Arbeit von Herrn Dipl. Ing. Dr. techn. Yll Haxhimusa beinhaltet zwei wichtige neue Beiträge, die neue theoretische Einsichten herleitet, beweist und an zahlreichen praktischen Beispielen austestet. Die vermittelten Inhalte werden methodisch einwandfrei, klar und vollständig dargeboten. Der Stand des Wissens ist nicht nur durch die umfangreiche Bibliographie eindrucksvoll belegt, jedes Kapitel beinhaltet einen umfangreichen Überblick mit entsprechend detailierten Zitaten. In vier sehr nützlichen Anhängen hat Dr. Haxhimusa das

für die Arbeit notwendige Grundlagenwissen zusammengefaßt. Dadurch wird die Dissertation zu einem in sich abgeschlossenen Beitrag mit sehr hohem Innovationsgrad und kann als Ausgangspunkt für viele neue Anwendungen dienen.

Wien im März 2007,
Univ. Prof. Dr. techn. Walter G. Kropatsch
Technische Universität Wien,
Fakultät für Informatik
Institut für computergestützte Automation,
Mustererkennung und Bildverarbeitung (PRIP-Labor)

*Dedicated to my parents*

# Acknowledgments

This work would have not seen the light without the full support of many people and without a stimulating working environment. Special thanks goes to my adviser Prof. Walter G. Kropatsch, who taught me how to work scientifically. His guide and his critical comments improved essentially the quality of this document. I would like to thank Prof. Luc Brun for his comments and review of this document. An extra special thanks go to my colleagues Adrian Ion, Georg Langs and Roland Glantz, with whom I had worked on different topics and problems. I would like to thank all the PRIPlers for making PRIP Labs such a great place to work. Danke!

This work started to have a form (Gestalt) during the three months stay at Institut National des Sciences Appliquées de Lyon in France. I would like to express many thanks to Prof. Jean-Michel Jolion and his collaborators at LIRIS for a pleasant working atmosphere. Merci!

Deep appreciation goes to my parents, my girlfriend and my family for supporting me all these years of my studies and being always whenever I needed them. Many thanks to my many dear friends in Austria and Kosova, that have influenced my work on many ways. Falemnderit!

I am deeply in dept to many many other people all around the world for making things like Emacs, TEX, LATEX, XFig, *x*v, and many other wonderful gadgets that I used to produce this document. Thank you!

I would like to thank the Austrian Science Found (Fonds zur Förderung der wissenschaftlichen Forschung) under grants P1444MAT, P14462INF and P18716-N13 for supporting this work. Many thanks to Sammuel Peltier for translating the abstract into French. Merci! Also I would like to thank Institute of Computer Aided Automation of Vienna University of Technology for their support to print this book. Danke!

# Abstract

The goal of computer vision is to make *machines see*, or at least to perform vision tasks with the same quality, quantity and speed as humans and animals. Humans and animals are able to delineate, detect and recognize objects in complex scenes 'in no time'. One of the most valuable and critical resources in human visual processing is time, therefore a *highly parallel model* is the biological answer dealing satisfactorily with this resource. Hierarchical representation and hierarchical processing in computer vision systems are the credible approach to address space and performance constraints, observed in human and animal visual systems. A widely used hierarchical representation in many areas of computer vision and pattern recognition is the (regular) image pyramid, which employs both coarse to fine and fine to coarse processing strategies. The main advantage of regular pyramids is rapid computation of global information in a recursive manner, due to the logarithmic height with respect to the size of the input, making algorithms running in this data structure having a logarithmic time complexity. However, regular image pyramids lack shift invariance and do not preserve object connectivity as a result of the fixed vertical neighborhood. Thus they should be abandoned as general segmentation methods. In order to cope with shift invariance, among others, new hierarchical structures, the so called *irregular pyramids*, should be used. However, the logarithmic height of irregular pyramids is lost, as well as the computational efficiency. The non-logarithmic height is the main drawback of the irregular structures. Employing graph theoretical formalisms to describe irregular hierarchical structures allows an easy analysis of irregular graph pyramids. In order to be able to encode multiple boundaries between regions we use *dual graphs*, and the build pyramid is called *dual graph pyramid*.

In this thesis we mainly deal with dual graph pyramids and their application in image partitioning. We introduce two new graph concepts and show that the presented methods, *maximal independent edge set* (MIES) and *maximal independent directed edge set* (MIDES) used for the construction of stochastic irregular pyramids, bound logarithmically the height of the pyramid. We show that the two widely used stochastic decimation strategies, the maximal independent vertex set (MIS) and data driven decimation process (D3P) do not lead to logarithmic tapering graph pyramids. After studying different techniques to build the structure of the pyramid, we are motivated to use these structures in a framework for bottom-up processing. Thus into this irregular graph pyramid framework, we introduce a time efficient image partitioning method based on Borůvka's minimum spanning tree principle (MST)(BorůSeg). Although the goal of image segmentation is a single partitioning of the image, and not necessarily an image hierarchy, a hierarchical representation is needed, especially if the image context is not taken into consideration. Even thought this method makes greedy decisions during the merging process it is able to capture important perceptually groupings. We evaluate the quality of the segmentation

results of the MIES, MIS and D3P versions of this MST based method with respect to humans and to other graph-based methods. The evaluation shows that in image segmentation, the different stochastic decimations used in this MST image partitioning method produce comparable results. We summarize the major contribution of this thesis and discuss about the computationally hard problem of graph matching and the applicability of hierarchical approaches in dealing with its complexity.

**Keywords:** Hierarchical representation, image pyramid, regular pyramid, irregular pyramid, dual graph representation, irregular graph pyramid, dual graph pyramid, dual graph contraction, graph decimation strategies, maximal independent vertex set (MIS), maximal independent edge set (MIES), maximal independent directed edge set (MIDES), data driven decimation process (D3P), minimum spanning tree (MST), graph-based segmentation, MST based image partitioning and segmentation, Borůvka's MST based image partitioning, segmentation evaluation.

# Zusammenfassung

Das Ziel im Bildverstehen[1] ist, *Maschinen sehen zu lassen*, oder mindestens den Maschinen die Fähigkeit beizubringen, Sehtätigkeiten mit derselben Qualität, Quantität und Geschwindigkeit wie von Menschen oder Tieren durchzuführen. Menschen und Tiere sind imstande, Objekte in komplexen Szenen sofort abzugrenzen, zu detektieren und zu erkennen. Daher ist Zeit eine sehr kritische Ressource und die biologische Antwort im menschlichen visuellen System ist ein hoch paralleles Model. Hierarchische Repräsentationen und hierarchische Verarbeitung im Bildverstehen sind ein guter Ansatz um die Raum- und Leistungsbeschränkungen zu bewältigen, die im menschlichen visuellen System beobachtet worden sind. Eine häufig verwendete hierarchische Repräsentation in vielen Bereichen des Bildverstehens und der Mustererkennung ist die (reguläre) Bildpyramide. Diese Repräsentation setzt beide Verarbeitungsstrategien ein, grob zu fein und fein zu grob. Der Hauptvorteil der regulären Pyramide ist eine schnelle Berechnung der globalen Information in rekursiven Verfahren, aufgrund der logarithmischen Höhe bezüglich der Eingabegröße. Daher haben die Algorithmen, die auf diese Datenstruktur laufen, eine logarithmische Zeitkomplexität. Dennoch mangelt die reguläre Bildpyramide an Schiebeinvarianz und ist nicht in der Lage die Zusammenhänge der Objekte wegen der festgelegten vertikalen Nachbarschaft zu gewährleisten. Daher sollte die reguläre Pyramide nicht als allgemeine Segmentationsmethode verwendet werden. Zur Bewältigung der Schiebeinvarianz sollten neue hierarchische Strukturen, die so genanten irreguläre Pyramiden, bevorzugt werden. Allerdings, sowohl die logarithmische Höhe der irregulären Pyramide als auch die rechnerbetonte Effizienz gehen verloren. Die Verwendung des Graphentheorieformalismus, um die irregulären Strukturen zu beschreiben, ermöglicht eine einfache Analyse dieser irregulären Pyramide. Um in der Lage zu sein mehrfache Grenzen zwischen Regionen zu repräsentieren, benutzen wir duale Graphen und deshalb heißt die konstruierte Pyramide duale Graphenpyramide.

In dieser Doktorarbeit befassen wir uns hauptsächlich mit dualen Graphenpyramiden und ihrer Anwendung in der Bildpartitionierung. Wir führen zwei neue Graphenkonzepte ein und zeigen, dass die präsentierte Methoden (maximal unabhängige Kantenmenge (MIES) und maximal unabhängige gerichtete Kantenmenge (MIDES)), die für Konstruktion der stochastischen irregulären Pyramide einsetzt, grenzen logarithmisch die Höhe der Pyramide ein. Wir zeigen auf, dass die zwei häufig verwendete stochastische Dezimierungsstrategien (die maximal unabhängige Knotenmenge (MIS) und den datengesteuerte Dezimierungsprozess (D3P)) nicht zur logarithmischen Höhe der Graphenpyramiden führen. Nach der Analyse der verschiedenen Techniken für den Aufbau der Bildpyramidenstrukturen möchten wir diese Strukturen in einem

[1] Computer vision.

Rahmen für bottom-up Verarbeitung einsetzen. Basierend auf den Borůvkas minimalen aufspannenden Baum (MST) präsentieren wir eine zeiteffiziente Bildpartitionierungmethode in dieser irregulären Graphenpyramide (BorůSeg). Wenn der Bildkontext nicht in Betracht gezogen wird, ist die hierarchische Repräsentation erforderlich, obwohl das Ziel der Bildsegmentation eine einzige Bildpartitionierung und nicht unbedingt eine Bildhierarchie ist. Selbst wenn diese Methode nur greedy Entscheidungen während des Prozesses betrifft, ist sie imstande wichtige Gestaltgruppierungen wahrzunehmen. Wir vergleichen die Qualität der Segmentationsergebnisse von MIES, MIS und D3P Versionen dieser MST Methode mit menschlichen Bildsegmentationen und Segmentationen von anderen graphenbasierten Methoden. Diese Bewertung zeigt, dass verschiedene stochastische Dezimierungen in MST basierte Bildsegmentation dieselben vergleichbare Ergebnisse liefert. Wir fassen die wichtigsten Beiträge dieser Doktoratarbeit zusammen, und schließen die Diskussion über das Graph-Matching Problem und die Anwendbarkeit der hierarchischen Ansätze, um die Komplexität dieses schwerrechenbaren Problems zu lösen.

**Schlagwörter:** Hierarchische Repräsentation, Bildpyramide, reguläre Pyramide, irreguläre Pyramide, duale Graphenrepräsentation, irreguläre Graphenpyramide, duale Graphenpyramide, duale Graphenkontraktion, Graphdezimierungsstrategien, maximal unabhängige Knotenmenge (MIS), maximal unabhängige Kantenmenge (MIES), maximal unabhängige gerichtete Kantenmenge (MIDES), datengesteuerter Dezimierungsprozess (D3P), minimaler aufspannender Baum (MST), graphbasierte Segmentation, Borůvka MST basierte Bildsegmentation, Segmentationevaluation.

# Résumé

Le but de la vision par ordinateur est de faire voir les machines ou au moins de leur faire accomplir les tâches de vision avec la même qualité, quantité et rapidité que les hommes ou les animaux. Les hommes et les animaux sont capables de dépeindre, détecter et reconnaître des objets dans des scènes complexes en 'un rien de temp'. Le temps est une ressource cruciale et la réponse biologique à ce problème dans le système de vision humain est un modèle parallèle élevé. L'approche crédible qui prend en compte les contraintes d'espace et de performance observées dans le système de vision humain est la représentation hiérarchique et le processus hiérarchique pour les systèmes de vision par ordinateur. Les pyramides (régulières) sont une représentation hiérarchique largement utilisée dans beaucoup de domaines de la vision par ordinateur et de la reconnaissance de formes. Cette représentation emploie les deux stratégies de traitement: de grossier à fin et de fin à grossier. L'avantage principal de la pyramide régulière est le calcul rapide de l'information globale de façon récursive. En effet elle a une hauteur logarithmique, fonction de la taille de l'entrée, de sorte que l'algorithme exécuté dans cette structure de données a une complexité temporelle logarithmique. Cependant, la pyramide régulière ne présente pas d'invariance de déplacement et ne préserve pas la connectivité à cause du voisinage vertical fixe. C'est à dire que si un objet dans une image est soumise à une translation d'un seul pixel, la description de l'objet dans la pyramide résultante n'est pas la même. Par conséquent la pyramide régulière devrait être abandonnée comme méthode de segmentation générale. Parmi d'autres nouvelles structures hiérarchiques, la pyramide irrégulière devrait être utilisée pour faire face à cette invariance de déplacement. Toutefois la hauteur logarithmique de cette pyramide irrégulière est perdue, ainsi que l'efficacité de calcul. L'utilisation du formalisme théorique des graphes pour décrire des structures hiérarchiques irrégulières permet une analyse facile de cette pyramide graphique irrégulière. Pour être capable d'encoder les frontières multiples entre les régions, nous employons les graphes duaux, ainsi la pyramide construite est apelée pyramide de graphes duaux.

Dans cette thèse nous traiterons principalement des pyramides de graphes duaux et de leurs applications dans la partition d'images. Nous introduirons deux nouveaux concepts de graphes et montrerons que les méthodes présentées ensemble maximal d'arêtes indépendantes (maximal independant edge set MIES) et ensemble maximal d'arêtes orientées indépendantes (MIDES) utilisées pour la construction de pyramides irrégulières stochastiques délimitent de facon logarithmique la hauteur de la pyramide. Nous montrerons que les deux stratégies de décimation couramment employées: la méthode de décimation stochastique (MIS) et le processus de décimation orienté données (D3P) ne conduisent pas à des pyramides graphiques effilées logarithmiques. Après l'étude de différentes techniques pour construire la structure de la pyramide, nous utiliserons ces structures dans un cadre de calcul de bas en haut dans lequel les

primitives sont de même type dans toute la hiérarchie. Ensuite, nous introduirons une méthode de partition d'images dans cette pyramide graphique irrégulière fondée sur le principe d'arbre couvrant minimal de Borůka (MST) qui est donc efficace dans le temps (peu coûteuse en temps) (BorůSeg). Bien que le but de la segmentation d'images soit une partition simple de l'image et pas nécessairement une hiérarchie, on a besoin de la représentation hiérarchique, en particulier si on ne prend pas en considération le contexte de l'image. Bien que cette méthode prenne des décisions radicales pendant le processus de fusion, elle est capable de saisir des groupages importants pour la perception. Nous comparerons la qualité des résultats des segmentations à l'aide des versions MIES, MIS et D3P de cette méthode avec celle des hommes et avec d'autres méthodes basées sur les graphes. Cette évaluation montre que différentes décimations stochastiques utilisées dans la méthode MST de partition d'images fournissent des résultats comparables pour la segmentation d'images. Nous résumerons la contribution majeure de cette thèse et discuterons du problème de l'appariement des graphes et de l'applicabilité des approches hiérarchiques dans le traitement de la complexité de ces problèmes de calcul difficiles.

**Mot Clé:** Représentations hiérarchiques, pyramides d'images, pyramides régulières, pyramides irrégulières, représentations de graphes duaux, pyramides irrégulières de graphes, pyramides de graphes duaux, contraction de graphes duaux, décimation de graphes duaux, MIS, MIES, MIDES, D3P, MST, segmentation d'images basée sur la méthode MST, evaluation des segmentations.

# Përmbledhje

Qëllimi i analizës së imazheve me llogaritës[2] është të *bëjë makinat të shofin*, ose së paku të bëjë makinat të kryejnë detyra me të njejtin kualitet, kuantitet dhe shpejtësi sikurse sistemi vizuel i njeriut apo i shtazëve. Njeriu dhe shtazët janë në gjendje shpejt dhe pa mund të madh, të kufizojnë, detektojnë edhe njofin objektet në skena shumë komplekse. Kjo nënkupton se koha është njëra prej resurseve më të çmuara në sitemim vizuel të njeriut. Përgjegjja biologjike e sistemin vizuel të njeriut ndaj përdormit minimal të këtij resursi është modeli ultra paralel. Reprezentacioni hierarkik dhe procesimi hierarkik janë qasje plauzible në zgjidhen e problemeve të obeservuara në sistemin vizuel të njeriut për shkak të kufizimeve në performancë dhe në hapsirën për ruajtjen e informatave. Një reprezentacion shumë i popullarizuar në shumë fusha të analizës së imazhit është piramida (e regullt) e imazhit. Ky reprezentacion ofron të dy mundesitë e procesimit: poshtë-lartë (bottom-up) dhe lartë-poshtë (top-down). Përparsia kryesore e piramidave të regullta është llogaritja e shpejtë e informatave globale në menyrë rekursive përshkak të lartësisë logaritmike të saj. Kjo kushtëzon që algoritmet që realizohen në këtë reprezentacion kanë kompleksitet kohorë logaritmikë. Megjithatë, pyramidet e rregullta nuk kanë invariance në zhvendosje dhe nuk kanë mundesi të ruajnë tërsinë e objekteve, përshkak të fqinjësisë vertikale fikse. Prandaj piramidet e rregullta nuk duhen të përdoren si metoda gjenerale në segmentimin e imazheve. Në mënyre që të zgjidhen problemet e invariancës në zhvendosje, në mes tjerash, struktura të reja hierarkike duhet të përdoren, të ashtu quajturat *pyramidat të parregullta*. Fatkeqësisht, lartësia logaritmike e piramidave si një veti e dëshiruar, humbet e me këtë edhe eficienca llogaritëse. Përdorimi i teorisë së grafeve për të përshkruar këto struktura hierarkike mundësonë një analizë më të thjeshtë të graf pyramidave të parregullta. Për të mundësuar reprezentimin e më shumë kufinjëve në mes të dy regjioneve, ne përdorim *grafet duale*, si rrjedhim piramida e ndërtuar mbi to quhet *graf pyramida duale*.

Në këtë temë ne kryesisht studjojmë graf piramidat duale dhe aplikimet e saja në particionim e imazhit. Ne paraqesim dy graf koncepte të reja teorike dhe demostrojmë se metodat e prezentuara, *bashkësia maksimale e arqeve të pavarura* (maximal independent edge set - MIES) dhe *bashkësia maksimale e arqeve të pavarurt të drejtuara* (maximal independent directed edge set - MIDES) që përdoren në konstrukcionin e piramdave të parregullta stohastike, kufizojnë lartësinë e pyramidës logaritmikisht. Ne po ashtu demostrojmë se të dy strategjitë e popullarizuara decimuese stohastike, *bashkësia maksimale e nyjeve të pavarura* (maximal independent set - MIS) dhe procesi decimues i mbështetur në të dhëna (data driven decimation - D3P) nuk qojnë në graf piramda të zvogluara logaritmikisht. Pas studimit të teknikave të ndyshme për konstruktimin e strukturës së piramidës, ne jemi të motivuar të përdorim këto

[2]Computer vision.

struktura në kuadër të një sistemi për bottom-up procesim në të cilin të gjithë primitivët visual janë të të njejtit tip në gjithë hierarkinë. Ne prezentojmë një metodë për particionim e imazheve në këtë graf pyramidë të parregullt të bazuar në pemën minimale të bashkruar (minimum spanning tree - MST) sipas Borůvka, prandaj metoda është eficiente në kohë (BorůSeg). Edhe pse qëllimi i segmentimit të imazhit është një particionim i vetëm i imazhit, dhe jo medoemos një hierarki e imazheve, reprezentacioni hierarkik është i nevojshëm, sidomos nëse konteksti i imazhit nuk mirret parasysh. Ideja prapa kësaj është se nëse nuk dimë se cka jemi duke kërkuar në imazhë, atëhere duhet përdorur një reprezentacion hierakik të imazhit. Me gjithë që kjo metodë merr vendime lakmiqare gjatë procesit të shkrirjes së regjioneve, prap se prap është në gjendje të zaptojë grupimet perceptuale të rëndësishme. Në vlerësojmë kualitetin e rezultateve segmentuese të MIES, MIS dhe D3P versioneve të kësaj metode të bazuar në MST në varshmëri me rezultatet segmentuese të bëra nga njerzit dhe nga metodat tjera po ashtu të bazuare në grafe. Evaluacioni dëfton se në segmentimin e imazheve, decimimet e ndyshme stohastike të përdorura në MST metodën e particionimit të imazheve shfaqin rezultate të krahasueshme. Në të vërtete asnjëra nga metodat e krahasuara nuk është dukshëm më e mirë. Ne përmbledhim kontributet kryesore të temës dhe përfundojmë diskutim me përshkrimin e një problemi të rëndë në shkencat kompjuterike, graph matching, dhe përdorim e mundshëm të hierarkisë në zgjidhjen e kompleksitetit kompjuterik të këtij problemi.

**Fjalët kryesore:** Reprezentacioni hierarkik, imazh piramida, piramida e regullt, piramida e paregullt, reprezentacioni i grafeve duale, strategjitë decimuese të grafit, bashkësia maksimale e nyjeve të pavarura (MIS), bashkësia maksimale e arqeve të pavarura (MIES), bashkësia maksimale e arqeve të pavarur të drejtuara (MIDES), procesi decimues i mbështetur në të dhëna (D3P), pema minimale e bashkruar (spanning) (MST), metoda e particionimit të imazhit sipas MST, metoda e particionimit të imazhit sipas MST Borůvkës, evaluacion i segmentimeve.

# Contents

CHAPTER 1

# Introduction

> ” ’So what’s the point of showing me something I can’t see?’
> ’So that you understand that just because you see something, it doesn’t mean to say it’s there. And if you don’t see something, it doesn’t mean to say it’s not there. It’s only what your senses bring to your attention’ ”
>
> **“Mostly Harmless” by Douglas Adams.**

## 1.1 Introduction

Vision in humans is one of the most valuable senses by which *the qualities of an object (as color, luminosity, shape and size) constituting its appearance are perceived and which is mediated by the eye* [Webster, 1913]. Humans and animals rely heavily on this sense to extract important information about the surrounding environment. This information is extracted and processed from vision cues in such **quality**, **quantity** and **speed** necessary for the subject to perform particular actions. All these tasks driven by vision are performed by humans and animals seemingly without effort. The goal of computer vision[1] is to make *machines to see*, or at least make machines to perform vision tasks with the same quality, quantity and speed as humans or animals. This is the ’holy grail’ in computer vision, a goal which is not yet reached. Even though, computer vision scientists are not limited to mimic biological vision, there are techniques applied in computer vision inspired by biological vision. On the other side techniques developed by computer vision are valuable source of computational models for biological vision [Rosenfeld, 1989].

[1] Called also image understanding or machine vision.

## 1. Introduction

Humans and animals are able to delineate, detect and recognize objects in complex scenes 'at a blink of an eye'. One of the most valuable and critical resources in human visual processing is time[2], therefore a *highly parallel model* is the biological answer dealing satisfactorily with this resource, since 'all complex behaviors are carried in less then 100 steps'[3] [Feldman and Ballard, 1982]. That is, since neurons have a computational speed of a few milliseconds and each perceptual phenomenon occurs in a few hundreds of milliseconds yield that biologically motivated algorithms must be carried out in less than 100 steps. [Tsotsos, 1988a, Tsotsos, 1990, Tsotsos, 1992] performed complexity analysis to show that hierarchical internal representation and hierarchical processing are the credible approach to deal with space and performance constrains, observed in human visual systems. Moreover, [Tsotsos, 1988a, Tsotsos, 1988b] concludes that in addition to spatial parallelization, other characteristics are necessary in visual systems, among others:

- **hierarchical organization** through abstraction of prototypical visual knowledge, such that search time is minimized, at least **logarithmically**,
- **receptive fields are localized**, because the physical world is spatio-temporally localized and that events and objects, and their physical characteristics, are not arbitrarily spread over time and space,
- semantic content is maintained by the **hierarchical abstraction** of the input arrays, such that the number of retinotopic elements is reduced,
- higher level maps[4] can be directly accessed through input abstraction hierarchy computations,
- prediction for the overall architecture of the visual system in terms of the size and number of maps,
- a **bottom-up** 'pre-attentive' performance similar to humans, and
- a **top-down** control mechanism.

Hence, **hierarchical structure** might be the answer to the time and space complexity in computer vision systems. It is now accepted that the human visual system has a hierarchical architecture and that the visual mechanisms can be adequately modeled by hierarchical algorithms. Specifically, neuro-physiological and neuro-anatomical data indicate that the visual systems of cats, monkeys and human beings are hierarchical, with neurons on lower layers having smaller receptive fields and neurons on higher layers having larger receptive fields [Zeki, 1993]. Pyramid algorithms are adequate models for the Gestalt rules of perceptual organization such as proximity, good continuation, etc. [Pizlo et al., 1997, Pizlo, 2001]. In the case of size processing, modeling visual processes involves both bottom-up (fine to coarse) and top-down (coarse to fine) analysis. The human visual system takes advantage of the ability to distinguish between highly valuable and less relevant regions in the field of view, employing speed-accuracy trade-off, improving its performance. [Stark and Privitera, 1997, Privitera and Stark, 2000] and [Chernyak and Stark, 2001] simulating human visual system describe two strategies to obtain and apply information about the importance of different regions of an image: the **bottom-up**

[2]Evolution conditioned the usage of this resource sparsely, because of survival necessity.
[3]Called 100 step rule.
[4]Called layers in image pyramid framework.

methods retrieve features only from the input image, and **top-down** methods are driven by available knowledge about the world. More recently, hierarchical (pyramid) algorithms have been used to model the mental mechanisms involved in solving the visual version of the Traveling Salesman Problem [Graham et al., 2000], as well as other types of visual problems [Pizlo and Li, 2003, Pizlo and Li, 2004]. Humans seem to represent states of a problem by clusters (recursively) and determine the sequence of transformations from the start to the goal state by a top-down sequence of approximations. This approach leads to algorithms whose computational complexity is as low as that of the mental processes (i.e. linear), and which produce solution paths that are close to optimal [Pizlo et al., 2006].

Local processing is important in early vision, since operations like convolution, thresholding, mathematical morphology etc. belong to this class of operations. However, this approach is not efficient for high or intermediate level vision, such as symbolic manipulation, feature extraction etc., because these processes need both local and global information. Therefore a data structure must allow the transformation of **local** information (based on sub-images) into **global** information (based on the whole image), and be able to handle both locally distributed and globally centralized information. This data structure is known as **hierarchical architecture** [Jolion and Rosenfeld, 1994]. This structure allows distribution of the global information to be used by local processes. This thesis mainly deals with the hierarchical structure called **pyramid**. Pyramids were first formalized in [Tanimoto and Pavlidis, 1975], as solution of contour detection and delineation in digital images, and in conjunction with the merge and split segmentation algorithm of [Horowitz and Pavlidis, 1976] are used extensively in image segmentation.

The hierarchical processing paradigm, sometimes also called *fine to coarse and coarse to fine processing strategy* are extensively used in many areas of computer vision and pattern recognition, for an overview see [Rosenfeld, 1984, Jolion and Rosenfeld, 1994, Kropatsch, 1991]. Generally, algorithms that need $\mathcal{O}(n^2)$ steps[5] on a array, need $\mathcal{O}(\log n)$ on a pyramid [Bischof, 1995]. The main advantage of the hierarchical structures is rapid computation of a global information in a recursive manner. The change of local over to global information, e.g. from pixels arrays to descriptive data structures, is a point of discontinuity in vision systems [Jolion and Rosenfeld, 1994]. Hierarchical structures offer a way to alleviate this discontinuity, where global structures become local in higher levels of this hierarchy. [Rosenfeld, 1984] writes that "pyramid, in general, are data structures that provide successfully condensed representation of the information in the input image"; e.g. intensity image can be condensed thus producing a stack of images of reduced resolution; or more complex descriptive information inferred from the (sub-)image itself can be condensed as well, producing a hierarchy of coarser version of these features.

The notion pyramid is broadly used to characterize the so called **regular pyramid**: an ordered stack of images defined on a regular square grid, such that resolution of each image toward the top is reduced by a factor of at least $r = 2$. One of the main characteristics of regular pyramid is that it has fixed height (in fact the height is logarithmic, e.g. for a reduction factor of $r = 2$, the height is $h = \log_2 n$), making algorithms running in this data structure having a logarithmic time complexity. However, [Bister et al., 1990] show that regular pyramid lack shift invariance and do not preserve connectivity because of the fixed vertical neighborhood i.e. if an image is translated or rotated by a pixel the resulting pyramid is not unique. Moreover,

[5] $n$ is the image size.

[Bister et al., 1990] concludes that regular pyramids should be rejected as general segmentation methods. In order to cope with this shift invariance, among others, [Meer, 1989, Montanvert et al., 1991, Jolion and Montanvert, 1992, Kropatsch and Montanvert, 1991b] presented new hierarchical structures the so called **irregular pyramid**. These structures have been successfully applied in many image analysis problems. If these hierarchical structures are posed using graph formalism, then they are called **irregular graph pyramids**. The logarithmic height of this pyramid is however lost [Montanvert et al., 1991, Bischof, 1995], as well as the computational efficiency. This non-logarithmic height happens, because the degree of the cell is not bounded, and is the main drawback of the irregular structures [Bischof, 1995, page 36]. Thus there is a need to study this irregular structure, and try to bound their height (Chapter 5).

Graph hierarchies allow to use other spatial orderings of image primitives, not only the regular spatial structures. Image primitives (e.g. pixels, edges, etc) are represented by vertices and their relations by edges of the graph. These vertices and edges are attributed. A classical example of graph representation of a set of primitives is the **region adjacency graph**, where each image region is represented by a vertex, and adjacent regions are connected by an edge. Attributes of vertices can be region area, average gray value, region statistics etc.; and attributes of edges can be the length of the boundary, the curvature, etc. between pair of adjacent regions. The graph hierarchy is then build by aggregating these primitives. The main application area of the region based representation is *image segmentation* and *object recognition*. Note that region adjacency graph representation is capable to encode only the neighborhood relations. In order to be able to encode multiple boundaries between regions[6] we use **dual graphs** [Kropatsch, 1994], i.e. we use the region adjacency graph and its (geometrical) dual graph[7]. Note that the dual graph is always defined for a planar graph. In contrast (simple) graphs are able to encode only one boundary between regions[8]. The dual graph representation may be used to characterize inclusion relationship as well. The hierarchy built on this dual graph representation is called irregular **dual graph pyramid**.

Segmentation is an ill-posed problem and up to now there is no standard approach to segmentation. Generally speaking segmentation is partitioning of the image plane into segments such that they satisfy homogeneity criteria, and such that the union of adjacent segments does not. A homogeneity criteria will not determine a unique segmentation [Nacken, 1994]. Segmentation methods that do not take the context of the image into consideration cannot produce a 'good' segmentation. Moreover, the quality of the segmentation produced by a method is evaluated with segmentation done by humans (as it is done in Chapter 7). For an overview on segmentation techniques see [Pal and Pal, 1993]. [Bister et al., 1990] reject the use of irregular pyramid because "we do not know how many levels we will have, or how many pixels are in each level thus e.g. eliminating the use of pyramid architectures for implementation of this approach". Anyhow, since irregular pyramid approach does not take the image context into consideration it has large potential application areas in computer vision. On the other side bottom-up methods are restricted because they cannot use all types of context knowledge in the image content during the construction of the hierarchy. Problems arise if we require that every salient region is reduced to only one cell and that they all are represented in the same level of the pyramid. For both, irregular and regular segmentation algorithms, one can envision

[6] This is called by [Brun and Kropatsch, 2006] meets each relation.

[7] Sometimes called extended region adjacency graph (RAG+).

[8] This is called by [Brun and Kropatsch, 2006] meets exist relation.

counterexamples. This generally occurs since very small and very large regions cannot be at the same level. Anyway [Nacken, 1994] shows that it is possible to parse the hierarchy in a top-down manner by using a relinking principle to repair for the 'errors' that the bottom-up methods have done. Moreover [Kropatsch and BenYacoub, 1996] shows that any segmentation can be represented in one single layer, by using equivalent contraction kernels.

There are many reasons for using hierarchical paradigm in image partitioning [Nacken, 1994]:

- the scale at which interesting structure is important is not known in advance, therefore a **hierarchical image representation** is needed,
- **efficiency of computation**; the results obtained from the coarse representation are used to constrain the costly computation in finer representations, and
- **bridging the gap** between elementary descriptive elements (e.g. pixels) and more descriptive elements, (e.g. regions).

Although the goal of image segmentation is producing a single partition of the image, and not necessarily a hierarchy, the hierarchical representation is needed, especially if the image context is not taken into consideration. The idea behind this is if you do not know what you are looking for in an image, then use a hierarchical representation of the image, and moreover a data structure that allows the ability to access the finest partitioning (in our case the bottom of the pyramid) or in case of 'bad' partitioning the faculty to repair this 'errors'. Even though the image partitioning methods (ours as well) that do not take the context of the image into consideration cannot give 'good' segmentation, they can be valuable tools in image analysis in the same sense as the efficient edge detectors are. Note that efficient edge detectors do not take the context of the image also, and in spite of this, are widely used tools in early vision. Therefore, the low-level coherence of brightness, color, texture or motion attributes should be used to come up sequentially with hierarchical partitions. Mid and high level knowledge can be used to either confirm these groups or select some further attention. A wide range of computational vision problems could make use of segmented images, just to mention some: object recognition, image indexing, video representation by regions etc, were such segmentation rely on efficient computation.

## 1.2 Objectives

The major goal of this thesis is the in-depth analysis of the irregular graph pyramids. Therefore we studied in details the structure of graph pyramids, and their application in image segmentation. The overall goal is the development of an efficient framework for hierarchical image processing. As mentioned earlier, the logarithmic height of the irregular pyramid is lost [Montanvert et al., 1991, Bischof, 1995], as well as the computational efficiency. Not being able to bound degree of the cell is the main drawback of the irregular structures [Bischof, 1995, page 36]. Therefore we posed a question:

*Whether it is possible to bound the height of the irregular pyramid ?*

and thus improve the computational efficiency of irregular pyramids. In fact we were looking for a possibility to bound the degree of the surviving cells. In the first part of this thesis we will

deal with this question (Chapter 5). The problem is posed in a graph theoretical framework. We introduce two new graph concepts and show that the presented methods (maximal independent edge set (MIES), and maximal independent directed edge set (MIDES)) used for the construction of the stochastic irregular pyramid bounds the height of the pyramid, i.e. the resulting pyramids are logarithmically tapered. These methods are compared with the two widely used methods: maximal independent set (MIS) [Meer, 1989] and data driven decimation process (D3P) [Jolion, 2003] (see Chapter 5). In this comparison only the structure of the irregular pyramid is taken into consideration.

These new methods can be employed instead of the stochastic decimation algorithm, with the advantage of producing less levels in the hierarchy, and thus being more efficient, e.g. in connected component analysis, distance transform, *segmentation*, efficient data reduction i.e. efficient memory consumption, enhanced principal component analysis, visual navigation, monotonic dual graph contraction, and possibly watersheds[9]. Even though these methods are developed in the image analysis framework, they can be used in computer graphics as well especially in mesh simplification with an additional property of preserving the topology.

Humans and animals seem to be very efficient in seeing (segmenting) regions by well defined edges. In Chapter 5 we show how to bound the irregular pyramid structurally. Thus we are motivated to use these structures into a framework for bottom-up processing in which primitives are of the same type in the whole hierarchy. Our goal in the second part of the thesis is defining a method that gathers global information and detects global structures as fast as possible and without effort which is necessary for real-time perception for example (Chapter 6). Therefore the framework should be able to:

- capture perceptually important regions (or groupings), and
- be efficient in time, possibly running in linear time with respect to the number of pixels in the image.

As well as be able to represent images in a hierarchical manner with different levels of granularities. The gap between elementary elements (e.g. pixels) and more abstract representation (e.g. regions) should be closed. Since we use the graph formalism the method presented is general enough and can be applied to any tessellation of image plane. A strong connection of hierarchical image analysis paradigm to perceptual grouping is established in [Rosenfeld, 1986]. We have chosen to use the minimum spanning tree (MST) principle, especially Borůvka's MST algorithm since it is inherently parallel and is naturally integrated into the irregular graph pyramid framework. Another motivation for using the MST principle is the computational time complexity (nearly linear) and that this algorithm is deterministic. Dual graph contraction [Kropatsch, 1994] is employed to reduce the size of graphs and thus produce naturally a hierarchy of image graphs. Therefore different decimation strategies (MIES, MIS and D3P) are used in dual graph contraction to build these hierarchical partitioning of images.

We show that the irregular image partition method based on minimum spanning tree principle (Chapter 6) is capable to cope with indoor and real world images in which at least one region is distinctive, and produce perceptual regions satisfactorily. In fact the method is able to handle 'busy regions' [10] as well as 'smooth regions' [11]. But as expected, not all possible seg-

[9] There are some links between monotonic dual graph contraction and the watersheds algorithms.

[10] Regions with high variability of intensity e.g. bush, hair etc.

[11] Regions with low intensity variability, e.g. grass, house etc.

mentations are included in this hierarchy, and the exhausted search of all possible segmentation is not computationally efficient. The 'errors' done during the merging process can be overcome by relinking process like in [Nacken, 1994]. Also one can produce a single segmentation by finding cells[12] representing a wanted region by employing techniques as in [Lallich et al., 2003]. These two last matters are not studied in this thesis. In order to test the quality performance, we evaluated (Chapter 7) this image partitioning method (all versions: MIES, MIS and D3P) with segmentation made by humans. Thus, a question is posed

*Whether different decimation strategies produce different segmentation results?*

It is shown that different decimation strategies (MIS, MIES and D3P) of this method are comparable to each other i.e. the experimental data shows that the results of segmentations are not changed to much by using different decimation strategies. The comparison of other graph-based segmentation methods (normalized cut [Shi and Malik, 2000] and Kruskal's minimum spanning tree method [Felzenszwalb and Huttenlocher, 2004]) with humans shows that the segmentation results are comparable as well. Moreover all the methods studied show similar segmentation results.

The segmentation method presented in this document is successfully employed to detect fixation point in an eye tracking method, see [Bartelma, 2004, Roy et al., 2004], as well as in segmenting multi-spectral satellite data [Kaiser, 2004]. [Bartelma, 2004, Bartelma and Roy, 2006] use this method for robust object detection. A similar graph based segmentation method is used in [Keselman and Dickinson, 2005] as the first stage in a generic modeling from examples, in real-time robot navigation [Sanfeliu et al., 2002], and in hierarchical graph matching of panoramic images [Glantz et al., 2004]. A different field of application of the result presented in this thesis might be in the filed of psychology trying to simulate the mental mechanism on humans that are evoked by solving the traveling salesmen problem or the minimum spanning tree as shown in [Pizlo and Li, 2003].

## 1.3 Structure of the Thesis

An effort is done to make this document as self contained as possible, so that it can be read without to much browse into the literature. Each chapter begins with a short introduction and ends with a short summary. Throughout this document different aspects of irregular graph pyramids are presented.

After the introductory Chapter 1, the basic concepts from the graph theory used throughout this document are presented in Chapter 2. An introduction into the image pyramid concepts are given in Chapter 3. Dual graph representation, dual graph contraction algorithm and the way how the dual graph pyramid are build are described in Chapter 4.

A detailed discussion of different graph decimation methods for building irregular graph pyramids is discussed in Chapter 5. It is shown that stochastic pyramid in some cases is not logarithmically tapered, i.e. the decimation process does not exponentially reduce the number of cell successively. The main reason for this behavior is that the cell's neighborhood is not bounded, for some cases the degree of the cell increases exponentially. In Chapter 5 we discuss methods that overcome this drawback.

[12] A single cell is preferred if it exist, which is not always the case.

A hierarchical image partitioning method based on dual graph representation and minimum spanning tree is shown in Chapter 6. This method is capable to capture important perceptual groupings satisfactorily based only on local information. The quality of the segmentation results of all version of this method are compared with human segmentation in Chapter 7. It is shown that different flavors (MIS, MIES and D3P) of this method are comparable to each other as well as to other graph-based segmentation methods.

Finally in the last Chapter 8 future research points are discussed, especially the problem of graph matching of large graphs.

This document contains some appendices to clarify in more details some concepts and results mentioned in chapters. Appendix A introduces vector spaces, used to explain the duality concept of graphs; Appendix B discusses a procedure on how to build a dual graph out of the given planar one; Appendix C addresses the memory requirement for different data structures used to represent graphs; and finally in Appendix D names of images used in segmentation evaluation (Chapter 7) are given.

CHAPTER 2

# Basics of Graph Theory

> " For one has only to look around to see 'real-world graphs' in abundance, either in nature (trees, for example) or in the works of man (transportation networks, for example). Surely someone at some time would have passed from some real-world object, situation, or problem to the abstraction we call graphs, and graph theory would have been born."[1]
>
> **by D. R. Fulkerson.**

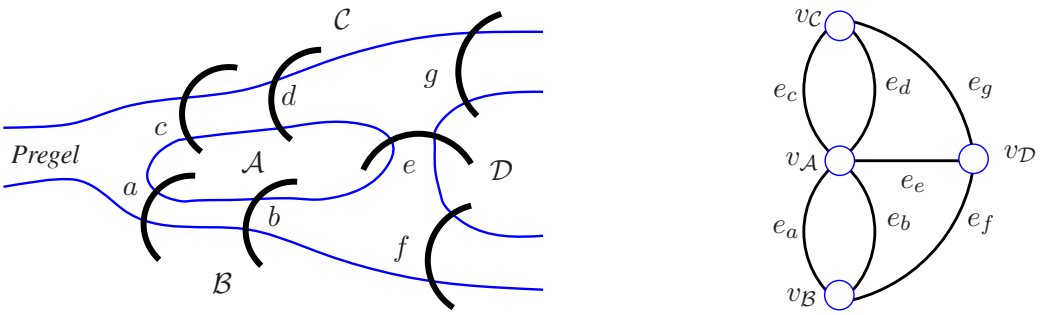
**Summary** In this chapter a short introduction of the basic definitions from graph theory will be given. These definitions will help to follow the discussion given in rest of the document as well as for easy reference to the nomenclature used afterward.

**Keywords:** Graph, multi-graph, vertex neighbor, edge adjacency, vertex degree, subgraphs, walk, paths, cycles, connectivity, forest, tree, vertex removal, edge removal, vertex identifying, edge contraction.

## 2.1 Introduction

In 1736, Leonard Euler was puzzled whether it is possible to walk across all the bridges on the river Pregel in Königsberg[2] only once and return to the starting point (see Figure 2.1a). This is how Euler stated the problem in "Solutio problematis ad geometriam situs pertinentis." [Euler, 1736] (an English translation of this paper can be found in [Biggs et al., 1976]):

[1] From preface to *Studies in Graph Theory*, Part II, The Mathematical Association of America, 1975.
[2] Nowadays Pregoyla in Kaliningrad.




(a) Seven bridges on the river Pregel
$\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ – landmasses
$a$, $b$, $c$, $d$, $e$, $f$, and $g$ – bridges

(b) Abstracted graph
$v_{\mathcal{A}}$, $v_{\mathcal{B}}$, $v_{\mathcal{C}}$, and $v_{\mathcal{D}}$ – vertices
$e_a$, $e_b$, $e_c$, $e_d$, $e_e$, $e_f$, and $e_g$ – edges

**Figure 2.1:** The seven bridges problem and the abstracted graph.

> "In Königsberg in Prussia, there is an island $\mathcal{A}$, called *Kneiphoff*; the river (Pregel) which surrounds it is divided into two branches, as can be seen in Figure 2.1a, and these two branches are crossed by seven bridges, $a$, $b$, $c$, $d$, $e$, $f$ and $g$. Concerning these bridges, it was asked whether anyone could arrange a route in such a way that he would cross each bridge once and only once. I was told that some people asserted that this was impossible, while others were in doubt; but nobody would actually assert that it could be done. On the basis of the above, I have formulated the general problem: Given any configuration of the river and the branches into which it may divide, as well as any number of bridges, to determine whether or not it is possible to cross each bridge exactly once."

In order to solve this problem, Euler in an ingenious way abstracted the bridges and the landmasses. He replaced each landmass by a dot (called vertex) and the each bridge by an arch (called edge or line) (Figure 2.1b). Euler proved that there is no solution to this problem. The Königsberg bridge problem was the first problem studied in what is later called graph theory. This problem was a starting point also for another branch in mathematics, the topology. This example shows a connection between graph theory and topology.

Unfortunately many books on graph theory have different notions for the same thing, or the same term has different meanings. The main purpose of this chapter is to collect basic notions of the graph theory in one place and to be consistent in terminology. This will help to follow the discussion given in rest of the document as well as for easy reference to the nomenclature used afterward. The definitions are compiled from the books [Diestel, 1997], [Thulasiraman and Swamy, 1992], [Harary, 1969], [Christofides, 1975] and [Bondy and Murty, 1976], therefore the citations are not repeated. Interested reader can find all these definitions and more in the above mentioned books. The erudite reader in graph theory can skip reading this chapter.

## 2.2 Basic Definitions

The Webster dictionary [Webster, 1913] defines graphs as having two meanings:

> **Graph**, n. (Math.)
> 1. A curve or surface, the locus of a point whose coordinates are the variables in the equation of the locus.
> 2. A diagram symbolizing a system of interrelations by spots, all distinguishable from one another and some connected by lines of the same kind.

The non-formal definition of the graph given in point 2 is the meaning used in this document. Formally, one can define graph $G$ on sets $V$ and $E$ as:

**Definition 2.1 (Graph)** *A graph $G = (V(G), E(G), \iota_G(\cdot))$ is a pair of sets of $V(G)$ and $E(G)$ and an incidence relation $\iota_G(\cdot)$ that maps pairs of elements of $V(G)$ (not necessarily distinct) to elements of $E(G)$.*

The elements $v_i$ of the set $V(G)$ are called vertices (or nodes, or points) of the graph $G$, and the elements $e_j$ of $E(G)$ are its edges (or lines). Let an example be used to clarify the incidence relations $\iota_G(\cdot)$. Let the set of vertices of the graph $G$ in Figure 2.1b be given by $V(G) = \{v_\mathcal{A}, v_\mathcal{B}, v_\mathcal{C}, v_\mathcal{D}\}$ and the edge set by $E(G) = \{e_a, e_b, e_c, e_d, e_f, e_g\}$. The incidence relation is defined as :

$$\begin{aligned} \iota_G(e_a) = (v_\mathcal{A}, v_\mathcal{B}),\ \iota_G(e_b) = (v_\mathcal{A}, v_\mathcal{B}),\ \iota_G(e_c) = (v_\mathcal{A}, v_\mathcal{C}), \\ \iota_G(e_d) = (v_\mathcal{A}, v_\mathcal{C}),\ \iota_G(e_e) = (v_\mathcal{A}, v_\mathcal{D}),\ \iota_G(e_f) = (v_\mathcal{B}, v_\mathcal{D}), \\ \iota_G(e_g) = (v_\mathcal{C}, v_\mathcal{D}). \end{aligned} \tag{2.1}$$

For the sake of simplicity of the notation, the incidence relation will be omitted, therefore one can write, without the fear of confusion:

$$\begin{aligned} e_a = (v_\mathcal{A}, v_\mathcal{B}),\ e_b = (v_\mathcal{A}, v_\mathcal{B}),\ e_c = (v_\mathcal{A}, v_\mathcal{C}), \\ e_d = (v_\mathcal{A}, v_\mathcal{C}),\ e_e = (v_\mathcal{A}, v_\mathcal{D}),\ e_f = (v_\mathcal{B}, v_\mathcal{D}), \\ e_g = (v_\mathcal{C}, v_\mathcal{D}). \end{aligned} \tag{2.2}$$

i.e. the graph is defined as $G = (V, E)$ without explicit mentioning of the incidence relation, even though it is always understood. The vertex set $V(G)$ and $E(G)$ are simple written as $V$ and $E$. There will be no distinction between a graph and its sets, one may write a vertex $v \in G$ or $v \in V$ instead of $v \in V(G)$, an edge $e \in G$ or $e \in E$, and so on. Vertices and edges are usually represented with symbols like $v_1, v_2, ...$ and $e_1, e_2, ...$, respectively. Note that in Equation 2.2, each edge is identified with a pair of vertices. If the edges are represented with ordered pairs of vertices, than the graph $G$ is called *directed* or *oriented*, otherwise it is called *undirected* or *non-oriented*. Two vertices connected by an edge $e_k = (v_i, v_j)$ are called *end vertices* or *ends* of $e_k$. In the directed graph the vertex $v_i$ is called the *source*, and $v_j$ the *target* vertex of edge $e_k$. Since the elements of edge set $E$ are distinct, more than one edge can join the same vertices. Edges having the same end vertices are called *parallel edges*[3]. If $e_k = (v_i, v_i)$, i.e. the end vertices are the same, then $e_k$ is called *self-loop*.

**Definition 2.2 (Multigraph)** *A graph $G$ containing parallel edges and/or self-loops is a multigraph.*

[3] Also called double edges.

A graph having no parallel edges and self-loops is called *simple graph*.

The number of vertices in $G$ is called the *order*, written as $|V|$; its number of edges is given as $|E|$. A graph of order $0$ is called an *empty graph*[4], and of order $1$ is simply called *trivial graph*[5]. A graph is *finite* or *infinite* based on its order. In this document all the graphs used are finite and not empty, if not otherwise stated.

The usual way to visualize graphs is by drawing a circle (or a dot) for each vertex and a line connecting these dots (as it was done by Euler), and in oriented graphs an arrow depicts the order of vertices (Figure 2.2). Just how these dots and lines are visualized is not important, what is relevant is the information which vertices are paired with which edge. In the graph in Figure 2.2, edges $e_4$ and $e_5$ are parallel edges; edge $e_7$ is the self loop. Note that in non-oriented graphs the order of vertices defining the edge does not matter, for example the edge $e_1 = (v_1, v_2)$ could have been defined also as $e_2 = (v_2, v_1)$ (see Figure 2.2a), whereas in oriented graphs the order of vertices defines the edge as well, i.e. $e_2 = (v_2, v_1)$; the edge $(v_1, v_2)$ does not exist in Figure 2.2b.

**Definition 2.3 (Vertex neighbors)** *Two vertices $v_i$ and $v_j$ are neighbors or adjacent if they are the end vertices of the same edge $e_k = (v_i, v_j)$.*

**Definition 2.4 (Edge adjacency)** *Two edges $e_i$ and $e_j$ are adjacent if they have an end vertex in common, say $v_k$, i.e. $e_i = (v_k, v_l)$ and $e_j = (v_k, v_l)$.*

For example, in the graph in Figure 2.2a $v_1$ and $v_2$ are neighbors, since they are connected by edge $e_1 = (v_1, v_2)$; edge $e_1$ and $e_2$ are adjacent since they have vertex $v_1$ as a common end. If all vertices of $G$ are pairwise neighbors, then $G$ is *complete*. A complete graph on $m$ vertices is written as $K^m$.

An edge is *incident* on its end vertices. The degree of a vertex $v$ is defined as:

**Definition 2.5 (Vertex degree)** *The degree (or valency) $deg(v)$ of a vertex $v$ is the number of edges incident on it.*

The vertex of degree $0$ is called *isolated*; of degree $1$ is called *pendant vertex*. Note that by Definition 2.5 a self-loop at a vertex $v$ contributes twice in the $deg(v)$. For example in the graph of Figure 2.2a $deg(v_1) = 2$; $deg(v_3) = 4$; $deg(v_5) = 3$ and so on.

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs:

**Definition 2.6 (Subgraph)** *$G' = (V', E')$ is a subgraph of $G$ ($G' \subseteq G$) if $V' \subseteq V$ and $E' \subseteq E$.*

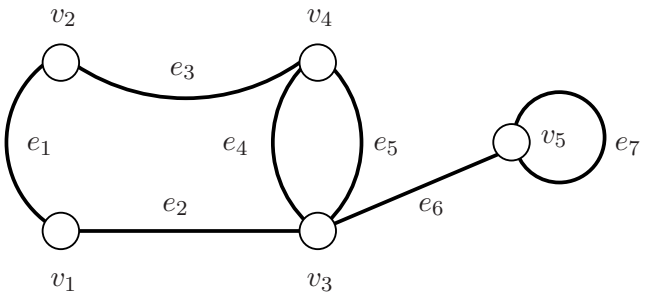
I.e. the graph $G$ contains graph $G'$, graph $G$ is called also a supergraph of $G$ ($G \supseteq G'$). If either $V' \subset V$ or $E' \subset E$, the graph $G'$ is called a proper subgraph of $G$. We say sometimes that $G$ contains $G'$. For example the graphs $G_1$ and $G_2$ in Figure 2.3 represent some of the subgraphs of graph $G$ from Figure 2.2a, graph $G_2$ is a proper subgraph of $G$.

**Definition 2.7 (Induced subgraph)** *If $G' \subseteq G$ and $G'$ contains all the edges of $e = (v_i, v_j) \in E$ such that $v_i, v_j \in V'$, then $G'$ is the (vertex) induced subgraph of $G$ and $V'$ induces (spans) $G'$ in $G$.*

[4] A graph with no vertices and hence no edges.

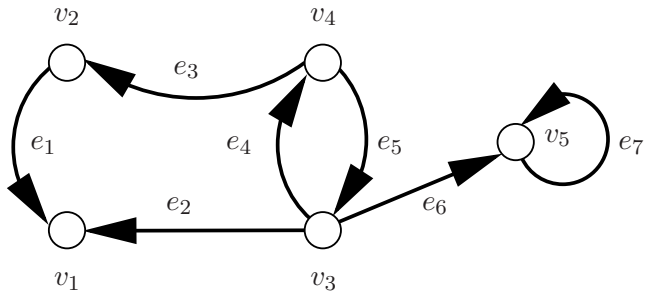
[5] A graph with one vertex and possibly with self-loops.

$$G = (V, E) \,|\, V = \{v_1, v_2, v_3, v_4, v_5\},\ E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$$




and $e_1 = (v_1, v_2), e_2 = (v_1, v_3), e_3 = (v_2, v_4), e_4 = (v_3, v_4),$
$e_5 = (v_3, v_4), e_6 = (v_3, v_5), e_7 = (v_5, v_5)$

(a) Non-oriented multi-graph




and $e_1 = (v_2, v_1), e_2 = (v_3, v_1), e_3 = (v_4, v_2), e_4 = (v_3, v_4),$
$e_5 = (v_4, v_3), e_6 = (v_3, v_5), e_7 = (v_5, v_5)$

(b) Oriented multi-graph

**Figure 2.2:** Non-oriented and oriented multi-graph.

The induced subgraph is usually written as $G' = G[V']$, i.e. since $V' \subset G(V)$, then $G[V']$ denotes the graph on $V'$ whose edges are the edges of $G$ with both ends in $V'$. If not otherwise stated by induced subgraph, the vertex-induced subgraph is meant. If there are no isolated vertices in $G'$, then $G'$ is called the *induced subgraph of $G$ on the edge set $E'$* or simply *edge-induced subgraph of $G$*. An example of vertex-induced subgraph is given in Figure 2.3b. Finally,

**Definition 2.8 (Spanning subgraph)** *If $G' \subseteq G$ and $V'$ spans all of $G$, i.e $V' = V$ then $G'$ is a spanning subgraph of $G$.*

The subgraph in Figure 2.3a $G_1$ is a spanning subgraph of $G$ since it contains all the vertices of $G$.

**Definition 2.9 (Maximal(minimal) subgraph)** *A subgraph $G'$ of a graph $G$ is a maximal (minimal) subgraph of $G$ with respect to some property $\Pi$ if $G'$ has the property $\Pi$ and $G'$ is not a proper subgraph of any other subgraph of $G$ having the property $\Pi$.*

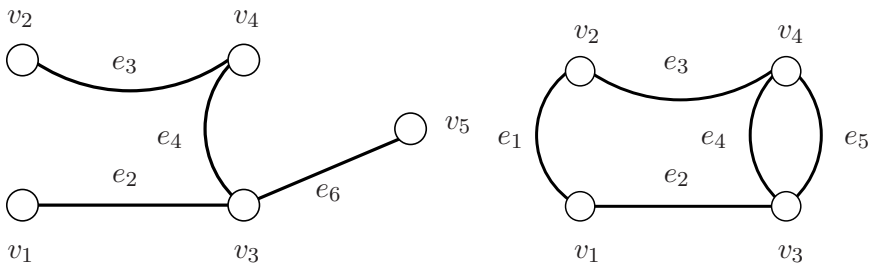



**Figure 2.3:** Subgraphs of graph $G$ from Figure 2.2a.

The minimal and maximal subsets with respect to some property are defined analogously. For example in Figure 2.3b, the edge set $E_2$ of $G_2$, a vertex-induced subgraph of $G$, is the maximal subset of $E$ such that the end vertices of all of its edges are in $V_2$ . This definition will be used later to define a component of $G$ as a maximal connected subgraph of $G$, and a spanning tree of a connected $G$ is a minimal connected spanning subgraph of $G$.

## 2.3 Paths and Cycles

Let $G = (V, E)$ be a graph with sets $V = \{v_1, v_2, \cdots\}$ and $E = \{e_1, e_2, \cdots\}$, then:

**Definition 2.10 (Walk)** *A walk in a graph $G$ is a finite non-empty alternating sequence $v_0, e_1, v_1, \cdots, v_{k-1}, e_k, v_k$ of vertices and edges in $G$ such that $e_i = (v_i, v_{i+1})$ for all $1 \leq i \leq k$*

This walk is called a $v_0 - v_k$ walk with $v_0$ and $v_k$ as the terminal vertices, all other vertices are internal vertices of this walk. In a walk edges and vertices can appear more than once. If $v_0 = v_k$, the walk is *closed*, otherwise it is *open*. For the graph in Figure 2.2a an open walk could be the sequence $v_1, e_2, v_3, e_4, v_4, e_4, v_3, e_5, v_4, e_4, e_6, v_5$ and a closed walk is $v_1, e_1, v_2, e_3, v_4, e_5, v_3, e_4, v_4, e_5, e_2, v_1$.

**Definition 2.11 (Trail)** *A walk is a trail if all its edges are distinct.*

A trail is closed if its end vertices are the same, otherwise it is opened. By the definition the walk can contain the same vertex many times. For example the walk $v_2, e_1, e_2, v_3, e_4, v_4, e_5, v_3, e_6$ is a trail in graph shown in Figure 2.2a, even though the vertex $v_3$ appears twice.

**Definition 2.12 (Path)** *A path $P$ is a trail where all vertices are distinct.*

A path defined thus a sequence of vertices together with a sequence of edges which allow to connect each vertex of the path to its sucessor. A simple path is defined as a sequence of vertices $v_0, v_1, v_2, \cdots, v_k$, each vertex being joint to its successor by some edge. Thus a simple path does not explicitly encode which edge allows to pass from one vertex to the next one. Note that using simple graphs two vertices are connected by at most one edge. The notion of path and simple path are thus equivalent on this graphs. This is obviously not the case with more general graphs. Note that in a multigraph a path is not uniquely defined by this nomenclature, because of possible multiple edges between two vertices. Vertices $v_0$ and $v_k$ are linked by the path $P$, also $P$ is called a path from $v_0$ to $v_k$ (as well as between $v_0$ and $v_k$).

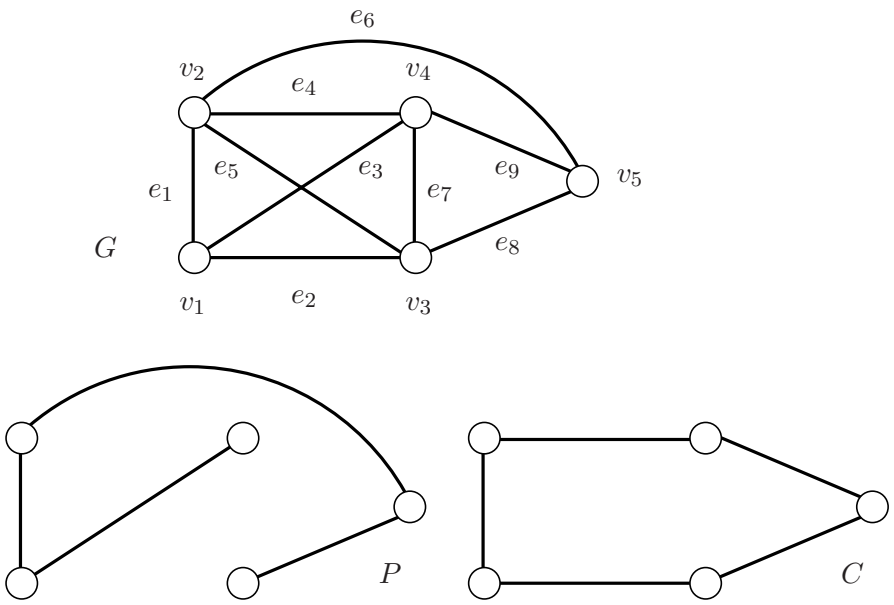



**Figure 2.4:** A path $P = P^4$ and a cycle $C = C^5$ in graph $G$.

**Definition 2.13 (Path length)** *The number of edges in the path is called the path length.*

The path length is denoted with $P^k$, where $k$ is the number of edges in the path. An example of the path is given in Figure 2.4, and it can be written as $P = v_4, v_1, v_2, v_5, v_3$. The length of this path is $4$, i.e. $P = P^4$. Note that by definition it is not necessary that a path contains all the vertices of the graph.

Analogously one defines the cycles as:

**Definition 2.14 (Cycle)** *A closed trail is a cycle $C$ if all its vertices except the end vertices are distinct.*

Cycles, like paths, are denoted by the cyclic sequence of vertices $C = v_0, v_1, \cdots, v_k, v_0$. The length of the cycle is the number of edges and it is called $k$-cycle written as $C^k$. The minimum length of a cycle in a graph $G$ is the girth $g(G)$ of $G$, and the maximum length of a cycle is its circumference. In Figure 2.4a cycle $C^5$ is shown. Note that the *girth* of graph $G$ in Figure 2.4 is $g(G) = 3$. The distance between two vertices $v$ and $w$ in $G$ denoted by $d(u, w)$, is the length of the shortest path between these vertices. The *diameter* of $G$, $diam(G)$ is the maximum distance between any two vertices of $G$.

From the above one can note the following properties of paths and cycles:

- in a path the degree of each vertex is $2$, except for the end vertices for which the degree is $1$,
- in a cycle the vertex degree of each vertex is $2$, and
- in a path the number of edges is one less than the number of vertices; in a cycle the number of edges and of vertices are equal.

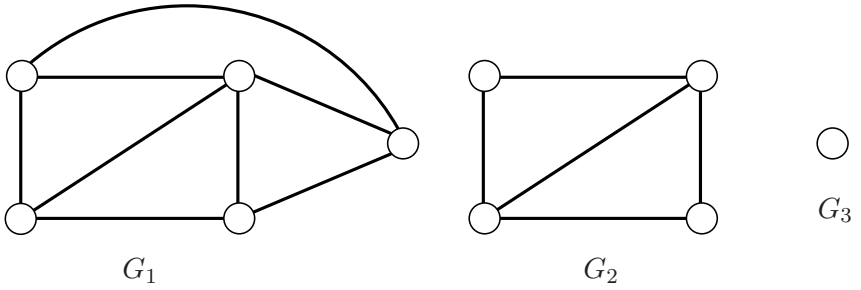



**Figure 2.5:** A non-connected graph $G$ and its components $G_1$, $G_2$ and $G_3$.

## 2.4 Connectivity and Graph Components

The connectivity is an important concept in graph theory and it is one of the basic concept used in this document. Two vertices $v_i$ and $v_j$ are connected in a graph $G = (V, E)$ if there is a path $v_i - v_j$ in $G$. A vertex is connected to itself.

**Definition 2.15 (Connectivity)** *A non-empty graph is connected if any two vertices are joint by a path in $G$.*

In Figure 2.5 graphs $G_1$, $G_2$ and $G_3$ are connected graphs.

Let graph $G = (V, E)$ be a non-connected graph. The set partitioning is defined:

**Definition 2.16 (Set partitioning)** *A set $V$ is partitioned into subsets $V_1$, $V_2$, $\cdots$, $V_p$ if $V_1 \cup V_2 \cup \cdots \cup V_p = V$ and for all $i$ and $j$, $i \neq j$ $V_i \cap V_j = \emptyset$. $\{V_1, V_2, \cdots, V_p\}$ is called a partition of $V$.*

Since the graph $G$ is not connected, the vertex set $V$ can be partitioned into subsets $V_1$, $V_2$, $\cdots$, $V_p$, and each vertex induced subgraph $G[V_i]$ is connected, then there exist no path between a vertex in subset $V_i$ and a vertex in $V_j$, $j \neq i$.

**Definition 2.17 (Component)** *A maximally connected subgraph of $G$ is called a component of graph $G$.*

A component of $G$ is not a proper subgraph of any other connected subgraph of $G$. An isolated vertex is considered to be a component, since it is connected to itself, by definition. Note that a component is always non-empty, and that if a graph $G$ is connected then it has only one component, i.e. itself. Figure 2.5 shows a non-connected graph $G$, with its components $G_1, G_2$ and $G_3$.

The following theorem is used in the Section 4.4 to show that after the edge removal from the cycle the graph stays connected.

**Theorem 2.1** *If a graph $G = (V, E)$ is connected, then the graph remains connected after the removal of an edge $e$ of a cycle $C \in E$, i.e. $G' = (V, E - \{e\})$ is connected.*

*Proof:* Suppose that removing edge $e$ of a cycle $\mathcal{C}$ disconnects graph $G'$ into two graphs[6] say $G^\dagger$ and $G^\ddagger$. This implies that there is no path between the vertices of $G^\dagger$ and of $G^\ddagger$. By

[6] an edge with this properties is called a bridge

definition, a cycle $C$ is a closed trail, therefore there are always two paths joining the vertices of the cycle. Therefore there must be at least another edge $e''$ between vertices of $G^{\dagger}$ and $G^{\ddagger}$ if $e \in C$. This contradicts that graph $G'$ is disconnected. □

From the above theorem one can conclude that edges that disconnect a graph do not lie on any cycle.

The definition of cut and cut-set are as follows. Let $\{V_1, V_2\}$ be partitions of the vertex set $V$ of a graph $G = (V, E)$.

**Definition 2.18 (Cut)** *The set $\mathcal{K}(V_1, V_2)$ of all edges having one end in one vertex partition ($V_1$) and the other end on the second vertex partition ($V_2$) is called a cut.*

**Definition 2.19 (Cut-set)** *A cut-set $\mathcal{K}_S$ of a connected graph $G$ is a minimal set of edges such that its removal from $G$ disconnects $G$, i.e. $G - \mathcal{K}_S$ is disconnected.*

If the induced subgraphs of $G$ on vertex set $V_1$ and $V_2$ are connected then $\mathcal{K} = \mathcal{K}_S$. If the vertex set $V_1 = \{v\}$, the cut is denoted by $\mathcal{K}(v)$. For example the removal of the set of edges $K_1 = \{e_6, e_8, e_9\}$ from the graph shown in Figure 2.4 is a cut-set as well as a cut, since it is minimal and disconnects the graph into two connected components (by definition an isolated vertex is connected). The set of edges $K_2 = \{e_3, e_5, e_6, e_8, e_9\}$ also disconnects the graph into two components but it is not minimal since $K_1$ is the a proper subset of $K_2$ .

## 2.5 Trees and Forests

Trees as simple graph structure, are the most common structure used. Before the definition of the tree is given, a definition of the acyclic graph is required.

**Definition 2.20 (Acyclic graph)** *A graph $G$ is acyclic if it has no cycles.*

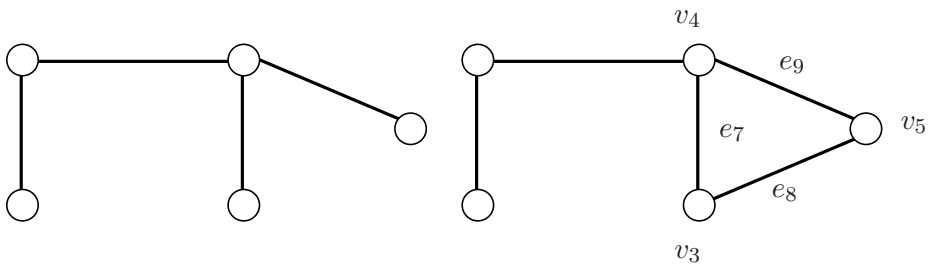
A simple example is shown in Figure 2.6a, whereas the graph under (b) in the same figure is a cyclic graph since it contains a cycle $(v_3, e_7, v_4, e_9, v_5, e_8)$.

**Definition 2.21 (Tree)** *A tree of graph $G$ is a connected acyclic subgraph of $G$.*

The vertices of degree $1$ in a tree are called *leaves*, and all edges are called branches. A non-trivial tree has at least two leaves and a branch, for example the simplest tree with two vertices joined by an edge. Note that an isolated vertex is by the definition an acyclic connected graph, therefore a tree. In Figure 2.6a, (c) and (d) examples of a trees are shown.

**Definition 2.22 (Spanning tree)** *Spanning tree of graph $G$ is a tree of $G$ containing all the vertices of $G$.*

Edges of the spanning tree are called *branches*. The subgraph $G'$ containing all vertices of $G$ and only those edges not in the spanning tree, is called *cospanning tree*, these edges are called *cords*. Note that a cospanning tree may not be connected. In Figure 2.6a and (c) are depicted two spanning trees of the graph $G$ from Figure 2.4. An acyclic graph with $k$ connected components is called a $k$-tree [Thulasiraman and Swamy, 1992]. Each connected component of a $k$-tree is a tree by itself. If the $k$-tree is a spanning subgraph of $G$, then it is called a spanning $k$-tree of $G$.




(a) An acyclic graph (b) A cyclic graph

(c) A spanning tree (d) A tree

(a) and (c) Trees of $G$ (b) and (d) Spanning trees $G$

**Figure 2.6:** Trees and spanning trees of the graph $G$ from Figure 2.4.

**Definition 2.23 (Forest)** *A forest $F$ of a graph $G$ is a spanning $k$-tree of G, where $k$ is the number of component of $F$.*

In other words a forest is a set of trees. In Figure 2.7, two examples of forest are shown, (a) with two and in (b) three components, i.e. trees, and span all the vertices of graph $G$. Note that the trees shown in Figure 2.6a and (c) are also forest containing only one component, the tree shown in (d) is not a forest since it does not span all the vertices of $G$. A forest is simply a set of trees, spanning all the vertices of $G$.

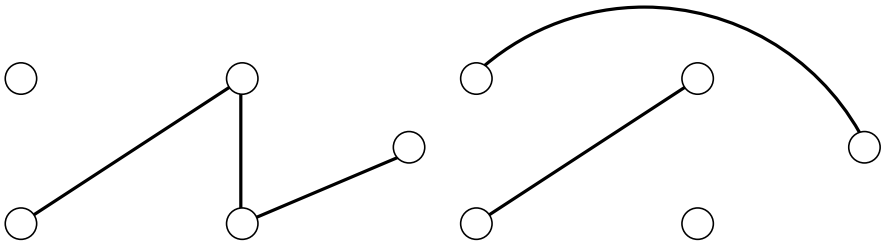
A connected subgraph of a tree $T$ is called a subtree of $T$. If $T$ is a tree then there is exactly one unique path between any two vertices of $T$. For a tree $T$ one can also say that it is

- minimally connected, i.e. $T$ is connected but $T - e$ is disconnected for every $e \in T$; and
- maximally acyclic, i.e. $T$ is acyclic but $T + e$ is cyclic, for any two non-adjacent vertices $v_i, v_j \in T$ such that $e = (v_i, v_j)$.

The proof of these assertion is found in [Thulasiraman and Swamy, 1992]. Note that spanning tree and forest are synonymous if the graph has only one component.

## 2.6 Operations on Graphs

In this section shortly some basic binary and unary operations on graphs are described. Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. Three basic binary operation on two graphs are:



(a) Two components (b) Three components

**Figure 2.7:** Examples of forest of $G$ from Figure 2.4.

## Union and Intersection

The *union* of $G$ and $G'$ is the graph $G'' = G \cup G' = (V \cup V', E \cup E')$, i.e. the vertex set of $G''$ is the union of $V$ and $V'$, and the edge set is the union of $E$ and $E'$, respectively. The *intersection* of $G$ and $G'$ is the graph $G'' = G \cap G' = (V \cap V', E \cap E')$, i.e. the vertex set of $G''$ has only those vertices present in both $V$ and $V'$, and the edge set contains only those edges present in both $E$ and $E''$, respectively. An example in Figure 2.8a of union and (b) of intersection of two graphs is given.

## Symmetric Difference

The *symmetric difference*[7] between two graphs $G$ and $G'$, written as $G \oplus G'$, is the induced graph $G''$ on the edge set $E \boxplus E' = (E \setminus E') \cup (E' \setminus E)$[8], i.e. this graph has no isolated vertices and contains edges present either in $G$ or in $G'$ but not in both. In Figure 2.8 an example of the ring sum between two graphs is given.

The four unary operations on a graph are:

## Vertex Removal

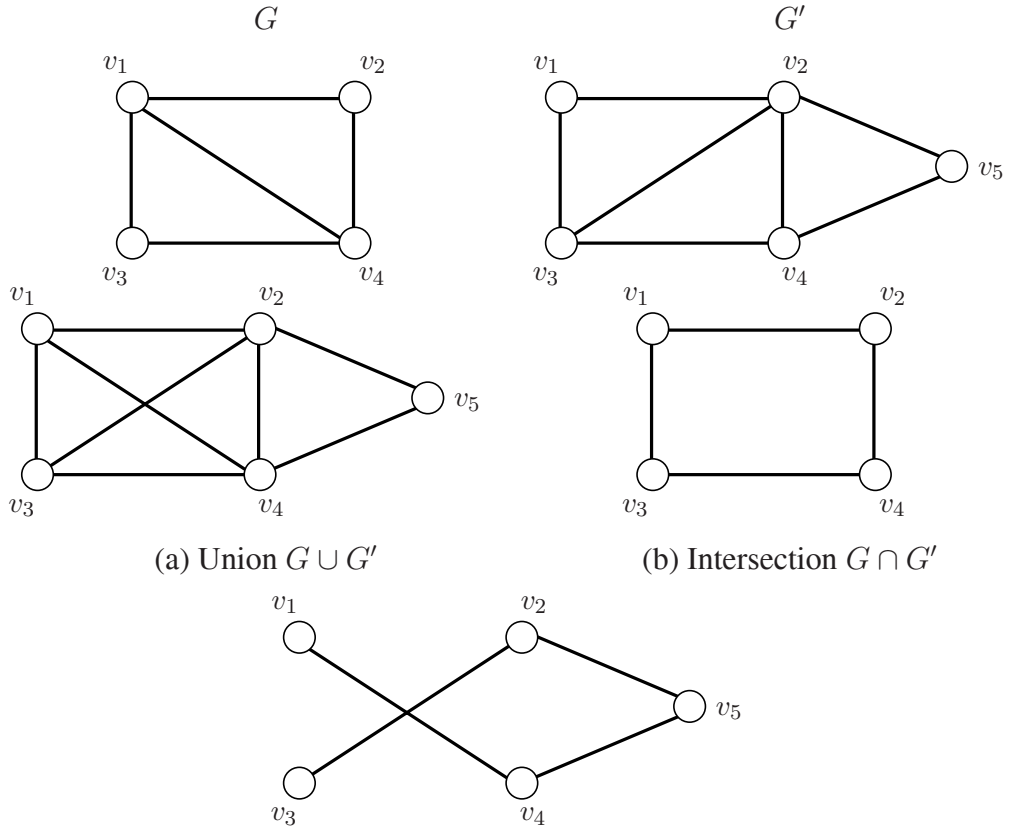
Let $v_i \in G$, then $G - v_i$ is the induced subgraph of $G$ on the vertex set $V - v_i$; i.e. $G - v_i$ is the graph obtained after removing the vertex $v_i$ and all the edges $e_j = (v_i, v_j)$ incident on $v_i$. The removal of a set of vertices from a graph is done as the removal of single vertex in succession. An example of vertex removal is shown in Figure 2.9a.

## Edge Removal

Let $e \in G$, then $G - e$ is the subgraph of $G$ after removing the edge $e$ from $E$. The end vertices of the edge $e = (v_i, v_j)$ are not removed. The removal of a set of edges from a graph is done as the removal of single edge in succession. An example of edge removal is shown in Figure 2.9b.

[7] Called also ring sum.

[8] Where $\setminus$ is the set minus operation and is interpreted as removing elements from $X$ that are in $Y$.




(a) Union $G \cup G'$

(b) Intersection $G \cap G'$

(c) Symmetric difference $G \oplus G'$

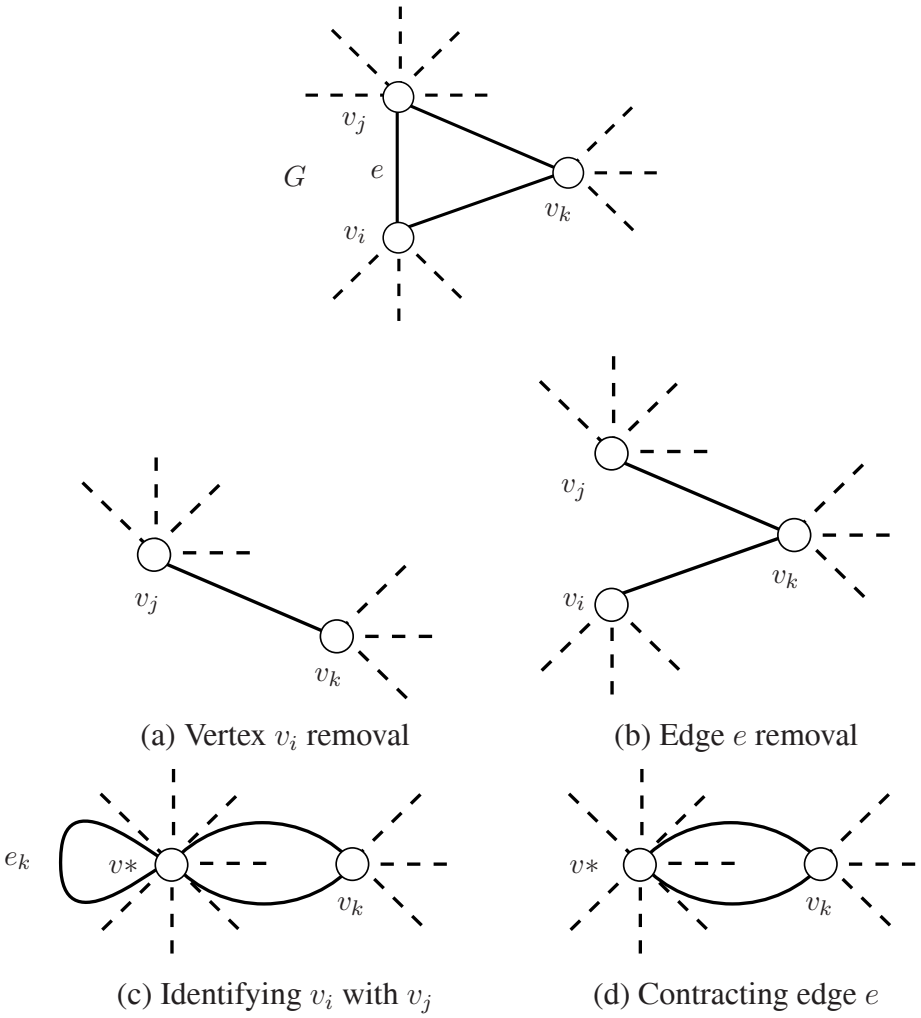
**Figure 2.8:** Binary graphs operations.

## Vertex Identifying

Let $v_i$ and $v_j$ be two distinct vertices of graph $G$ joined by the edge $e = (v_i, v_j)$. Two vertices $v_i$ and $v_j$ are identified if they are replaced by a new vertex $v*$ such that all the edges incident on $v_i$ and $v_j$ are now incident on the new vertex $v*$. An example of vertex identifying is given in Figure 2.9c.

## Edge Contraction

Let $e = (v_i, v_j) \in G$ be the edge with distinct end points $v_i \neq v_j$ to be contracted. The operation of an edge contraction denotes the removing of the edge $e$ and identifying its end vertices $v_i$ and $v_j$ into a new vertex $v*$. If the graph $G'$ results from $G$ after contracting a sequence of edges, than $G$ is said to be *contractible* to a graph $G'$. Note the difference between vertex identifying and edge contraction, in Figure 2.9c and (d). Vertex identifying preserves the edge $e_k$, whereas edge contraction first removes this edge. In Chapter 4, Section 4.4 a detailed treatment of edge contraction and edge removal in the dual graphs context is given.




(a) Vertex $v_i$ removal (b) Edge $e$ removal

(c) Identifying $v_i$ with $v_j$ (d) Contracting edge $e$

**Figure 2.9:** Operations on graph.

### 2.6.1 Homeomorphism

Let $e = (u, v)$ and $e' = (v, u')$ be the only edges incident on a vertex $v$. Removal of the vertex $v$ and replacing $e$ and $e'$ by the edge $(u, u')$ is called *series merger*. Adding a new vertex $v$ on an edge $(u, u')$ creating edges $(u, v)$ and $(v, u')$ is called *series insertion*.

**Definition 2.24 (Isomorphic graphs)** *Two graphs $G = (V, E)$ and $G' = (V', E')$ are isomorphic if there exist a bijection $\beta : V \to V'$ with $(u, u') \in E \Leftrightarrow (\beta(u), \beta(u')) \in E'$ for all $u, u' \in V$, and it is written as $G \simeq G'$.*

The $\beta$ is called an isomorphism. If the graphs are identical, i.e. $G = G'$, $\beta$ is called automorphism.

**Definition 2.25 (Homeomorphic graphs)** *Two graphs are homeomorphic if they are isomorphic and can be made isomorphic by repeated series of insertion and/or mergers.*

Note that if the graph $G$ is planar then any homeomorphic graph to $G$ is also planar.

## 2.7 Vector Spaces on Graphs

The duality concept of planar graphs is easily defined and explained using the vector spaces on graphs. The vector space on graphs is used to give a definition of dual graphs in Chapter 4, Section 4.2, Definition 4.3, which is on the other hand, used to prove an important property of dual graphs with respect to the edge contraction and removal operation in Chapter 4, Section 4.2, Theorem 4.1, which states that graphs during the process of dual graph contraction stay planar and are duals.

In this section the necessary definitions are given for building a vector space over a graph. Let $G = (V, E)$ be a graph with at least one edge, and let the set of all subsets of $E$ (called also the power set of $E$) including $\emptyset$ be denoted by $\mathcal{E}_G$. We use $2^E$ as nomenclature for the power set[9]. Therefore we write $\mathcal{E}_G = 2^E \cup \emptyset = \{E_i | \forall i = 0, \ldots, 2^{|E|}\, E_i \in 2^E, i \neq j \Rightarrow E_i \neq E_j\}$. It follows the prove that $\mathcal{E}_G$ under the operation of addition $\boxplus$ and of multiplication $\boxdot$ is a vector space over the field $\mathbb{F}_2 = \{0, 1\}$ (see Appendix A for more details on how to build a vector space in general). Let the operation of addition $\boxplus$ be defined as the symmetric difference between two sets (see Section 2.6), and the scalar multiplication operation as:

$$1 \boxdot E_i = E_i, \tag{2.3}$$

and

$$0 \boxdot E_i = \emptyset. \tag{2.4}$$

for any $E_i \in \mathcal{E}_G$. The symmetric difference ($\boxplus$) of any two elements $E_i$ and $E_j$ of $\mathcal{E}_G$ is $E_k = E_i \boxplus E_j = (E_i - E_j) \cup (E_j - E_i)$ and it must be an element of the collection of all subsets of $E$, i.e. $E_k \in \mathcal{E}_G$, therefore $\mathcal{E}_G$ is closed under the addition operation. The associativity also holds for all $E_i$, $E_j$ and $E_k$ in $\mathcal{E}_G$ since $(E_i \boxplus E_j) \boxplus E_k = E_i \boxplus (E_j \boxplus E_k)$. Can be easily proved using some set algebra or Venn diagrams and using $(E_i - E_k) \cup (E_k - E_i) = (E_i \cup E_k) \cap (E_i^T \cup E_k^T)$, where $E^T$ consist of all the edges not in $E$ and it is called the complement of $E$. The commutative property $E_i \boxplus E_j = E_j \boxplus E_i$ is proved in the same manner. For any element in $\mathcal{E}_G$ there is an identity element

$$E_i \boxplus \emptyset = E_i, \tag{2.5}$$

and an inverse one

$$E_i \boxplus E_i = \emptyset. \tag{2.6}$$

Therefore it is proved that $\mathcal{E}_G$ is an abelian group under the addition operation $\boxplus$. Let $a,\ b$ be elements in the field $\mathbb{F}_2 = \{0, 1\}$ with the additive identity element $e_{\mathbb{F}_{2\oplus}} = 0$ and multiplicative identity element $e_{\mathbb{F}_{2\odot}} = 1$. Let the addition and multiplication in field $\mathbb{F}$ be defined as modulo 2 addition ($\oplus$) and modulo 2 multiplication ($\odot$) (see Table A.1 in Appendix A for details). $E_i$ and $E_j$ are elements in $\mathcal{E}_G$, and $a$ and $b$ are scalars from $\mathbb{F}_2$. From the definition of the additive and multiplicative operation one can prove that the other axioms needed for a vector space are satisfied:

[9] Sometimes $\mathcal{P}(E)$ is used.

1. $(a \oplus b) \boxdot E_i = (a \boxdot E_i) \boxplus (b \boxdot E_i)$ ,
2. $a \oplus (E_i \boxplus E_j) = (a \boxdot E_i) \boxplus (a \boxdot E_j)$ ,
3. $(a \odot b) \boxdot E_i = a \boxdot (b \boxdot E_i)$, and
4. $e_{\mathbb{F}_{2\odot}} \boxdot E_i = 1 \boxdot E_i = E_i$

All necessary axioms needed for a vector space are proved, hence $\mathcal{E}_G$ is a vector space over the field $\mathbb{F}_2$, more precisely it is an *edge space*. If the set $E = \{e_1, e_2, \ldots, e_n\}$ than the vector set $\mathcal{B} = \{\{e_1\}, \{e_2\}, \ldots, \{e_n\}\}$ constitutes the basis of $\mathcal{E}_G$ and this space has the dimension $n = |E|$. Every element of $\mathcal{E}_G$ can be expressed as the linear combination of elements of $\mathcal{B}$ with scalars from $\mathbb{F}_2$. The element of the edge space can be interpreted as functions of the form $E \to \mathbb{F}_2$. In an analogous way a vertex space $\mathcal{V}_G$ over the field $\mathbb{F}_2$ can be build as a vector space of all function $V \to \mathbb{F}_2$, $\mathcal{V}_G$ can be considered as the power set over $V$, the sum is defined as the (vertex) symmetric difference. The zero vector in $\mathcal{V}_G$ is the empty (vertex) set, and inverse of $V_i \in \mathcal{V}_G$ is the $V_i$ itself. If the set $V = \{v_1, v_2, \ldots, v_m\}$, then set $\{\{v_1\}, \{v_2\}, \ldots, \{v_m\}\}$ is the basis of vertex space $\mathcal{V}_G$, hence its dimension is $m = |V|$. The discussion continues only on the edge vector space afterward and the terms vector space and edge space are interchanging.

From the definition of the symmetric difference operators in Section 2.6, the symmetric difference between two edge-induced subgraphs is the same as the symmetric difference of their edge sets, it follows that the set of all edge-induced subgraphs of $G$ is a vector space over $\mathbb{F}_2$ if the operations of scalar multiplication is defined as:

$$1 \boxdot G_i = G_i, \tag{2.7}$$

and

$$0 \boxdot G_i = \emptyset. \tag{2.8}$$

where $\emptyset$ represents the empty graph.

The duals of the plane graph are easily defined using concept of cycle and cut subspaces of $\mathcal{E}_G$. Let the cycle subspace $\mathcal{C}_G$ represent the set of all cycles (including the empty graph) in $G$; and the cut subspace $\mathcal{K}_G$ represent the set of all cuts (including the empty graph) in $G = (V, E)$. We show, that $\mathcal{C}_G$ and $\mathcal{K}_G$ are subspaces in $\mathcal{E}_G$.

**Proposition 2.1** *The set of all cycles $\mathcal{C}_G$, is a subspace of the vector space $\mathcal{E}_G$ of $G$.*

*Proof:* The proof is due to [Thulasiraman and Swamy, 1992]. The idea of the proof is as follows. $\mathcal{C}_G$ is subspace of $\mathcal{E}_G$, if one prove that for all $C, C' \in \mathcal{C}$ also $C \boxplus C'$ is in $\mathcal{C}_G$. From the definition of the cycles, every vertex is of degree 2, therefore $\mathcal{C}_G$ may be considered as the edge-induced subgraphs of $G$, in which all vertices are of degree even. Let $C$ and $C'$ be in $\mathcal{C}_G$, be edge-induced subgraphs with the degree of all their vertices even. Let $C'' = C \boxplus C'$. In other words we should prove that every vertex in $C''$ is of even degree. Consider a vertex $v \in C''$. This vertex must be present in at least one of subgraphs $C$ and $C'$. Let $X$, $X'$ and $X''$ denote the set of edges incident to $v$ in $C$, $C'$ and $C''$, respectively, and $deg(v_X)$, $deg(v_{X'})$ and $deg(v_{X''})$ the degree of $v$ in $C$, $C'$ and $C''$. From above $deg(v_X)$ and $deg(v_{X'})$ are even (and one of them may be zero), and hence $deg(v_{X''})$ is nonzero. Since $C'' = C \boxplus C''$ follows that $X'' = X \boxplus X'$. Hence $deg(v_{X''}) = deg(v_X) + deg(v_{X'}) - 2deg(X \cap X')$, where $deg(X \cap X')$ is the contribution

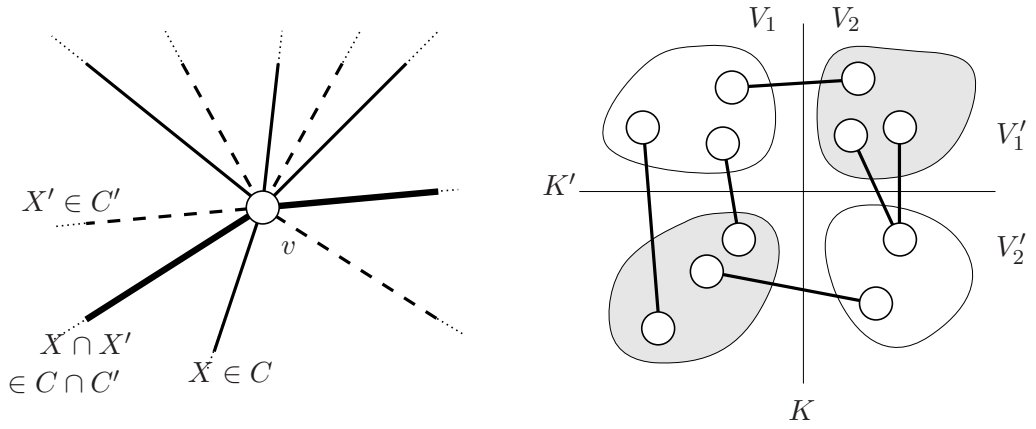



**Figure 2.10:** Cycle vertex in $C \boxplus C'$ and cut edges in $K \boxplus K'$.

to the degree of edges in $X \cap X'$ (see Figure 2.10a) and note that these edges are counted twice first in $deg(v_X)$ and then $deg(v_{X'})$). $deg(v_{X''})$ is even since $deg(v_X)$ and $deg(v_{X'})$ are even i.e. the degree of $v$ in $C''$ is even. Since this is $\forall v \in C''$, it follows that $C''$ is in $\mathcal{C}_G$. □

**Proposition 2.2** *The set of all cuts $\mathcal{K}_G$, is a subspace of the vector space $\mathcal{E}_G$ of $G$.*

*Proof:* The proof is due to [Diestel, 1997]. To prove that $\mathcal{K}_G$ is subspace of $\mathcal{E}_G$, one must prove that for all $K, K' \in \mathcal{K}_G$ also $K \boxplus K'$ is in $\mathcal{K}_G$. Since $K \boxplus K = \emptyset \in \mathcal{C}$, and $K \boxplus \emptyset = K \in \mathcal{C}$ (by the definition of symmetric difference), it is assumed that $K$ and $K'$ are non-empty and distinct. Let $\{V_1, V_2\}$ and $\{V_1', V_2'\}$ be the partition of the set $V$. Note that $V_1 \cup V_2 = V_1' \cup V_2' = V$. Then $K \boxplus K'$ consists of all the edge that cross one of these partitions but not the other (see Figure 2.10). These are precisely the edges between $(V_1 \cap V_1') \cup (V_2 \cap V_2')$ and $(V_1 \cap V_2') \cup (V_2 \cap V_1')$, and by $K \neq K'$ these two sets form another partition of $V$. Hence $K \boxplus K' \in \mathcal{K}_G$ and $\mathcal{K}_G$ is a subspace of $\mathcal{E}_G$. □

**Corollary 2.1** *The space $\mathcal{K}_G$ is created by cuts of form $\mathcal{K}(v)$.*

*Proof:* Due to [Diestel, 1997]. Every edge $e = (v_1, v_2) \in E$ lies in two cuts $\mathcal{K}(v_1)$ and $\mathcal{K}(v_2)$, thus every partition $\{V_1, V_2\}$ satisfies $\mathcal{K}_G = \sum_{v \in V_1} \mathcal{K}(v)$, where the sum is over the operator $\boxplus$. □

Let the concept of (vector) edge space be clarified with a simple example of the graph $G = (V, e)$ given in Figure 2.11. For the given edge set $E$, the set power is:

$$\begin{aligned}\mathcal{E}_G = \{&\emptyset, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_5\},\\ &\{e_1, e_2\}, \{e_1, e_3\}, \{e_1, e_4\}, \{e_1, e_5\}, \{e_2, e_3\}, \{e_2, e_4\}, \{e_2, e_5\}, \{e_3, e_4\}, \{e_3, e_5\}, \{e_4, e_5\},\\ &\{e_1, e_2, e_3\}, \{e_1, e_2, e_4\}, \{e_1, e_2, e_5\}, \{e_1, e_3, e_4\}, \{e_1, e_3, e_5\}, \{e_1, e_4, e_5\}, \{e_2, e_3, e_4\},\\ &\{e_2, e_3, e_5\}, \{e_2, e_4, e_5\}, \{e_3, e_4, e_5\},\\ &\{e_1, e_2, e_3, e_4\}, \{e_1, e_2, e_3, e_5\}, \{e_1, e_2, e_4, e_5\}, \{e_1, e_3, e_4, e_5\}, \{e_2, e_3, e_4, e_5\},\\ &\{e_1, e_2, e_3, e_4, e_5\}\}.\end{aligned}$$

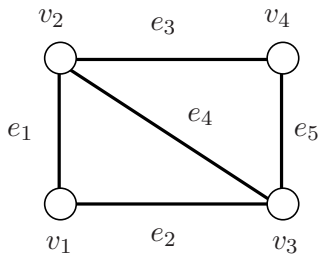



**Figure 2.11:** Edge space on $G$ is $\mathcal{E}_G : E \to \mathbb{F}_2 = \{0, 1\}$.

Altogether $\mathcal{E}_G$ contains $2^{|E|} = 2^5 = 32$ subsets. Let the vector addition $\boxplus$ and scalar multiplication $\boxdot$, as well as the identity and inverse elements for the $\boxplus$ and $\boxdot$ respectively, be defined as described above in this section. It is easily shown that $\mathcal{E}_G$ is an edge space over the field $\mathbb{F}_2$. The set $\mathcal{B} = \{\{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_5\}\}$ is the basis of the edge space, understood this vectors are linearly independent. This space has the dimension $5$. Every element of the set $\mathcal{E}_G$ can be represented as the linear combination of the elements of $\mathcal{B}$, e.g.

$$\{e_1, e_2, e_5\} = (1 \boxdot \{e_1\}) \boxplus (1 \boxdot \{e_2\}) \boxplus (0 \boxdot \{e_2\}) \boxplus (0 \boxdot \{e_3\}) \boxplus (0 \boxdot \{e_4\}) \boxplus (1 \boxdot \{e_5\}) =$$

based on the definition of the scalar multiplication $\boxdot$, it follows:

$$= \{e_1\} \boxplus \{e_2\} \boxplus \emptyset \boxplus \emptyset \boxplus \emptyset \boxplus \{e_5\} = \{e_1, e_2, e_3\}.$$

The set of all cycles for the graph $G$ given is

$$\mathcal{C}_G = \{\emptyset, \{e_1, e_2, e_4\}, \{e_3, e_4, e_5\}, \{e_1, e_2, e_3, e_5\}\},$$

with the base $\mathcal{B}_\mathcal{C} = \{\{e_1, e_2, e_4\}, \{e_3, e_4, e_5\}\}$, and the set of all cuts in $G$ is

$$\mathcal{K}_G = \{\emptyset, \{e_3, e_5\}, \{e_1, e_2\}, \{e_1, e_4, e_5\}, \{e_1, e_3, e_4\}, \{e_3, e_4, e_2\}, \{e_2, e_4, e_5\}, \{e_1, e_2, e_3, e_5\}\}.$$

with the base $\mathcal{B}_\mathcal{K} = \{\{e_3, e_5\}, \{e_1, e_2\}, \{e_1, e_4, e_5\}\}$. One can easily prove that all elements of $\mathcal{C}_G$ can be described as a linear combination of vectors $\mathcal{B}_\mathcal{C}$ with scalar from $\mathbb{F}_2$, the same hold for $\mathcal{K}_G$ and $\mathcal{B}_\mathcal{K}$, respectively. E.g. a cycle $C_1 = \{e_1, e_2, e_3, e_5\} = 1 \boxdot \{e_1, e_2, e_4\} \boxplus 1 \boxdot \{e_3, e_5, e_4\} = \{e_1, e_2\} \boxplus \{e_3, e_5\}$, and a cut $K_1 = \{e_1, e_3, e_4\} = 1 \boxdot \{e_3, e_5\} \boxplus 1 \boxdot \{e_1, e_4, e_5\} = \{e_3, e_5\} \boxplus \{e_1, e_4, e_5\}$. It is shown in Appendix A that the set of $n$-vectors over $\mathbb{F}_2$, where such that each vector's element in $\mathbb{F}_2$ form also a vector space. For the enumeration of the edges $E$ as in Figure 2.11, one writes e.g. instead of $\{e_1, e_2, e_3\}$ an $5$-vector of the form $(1, 1, 1, 0, 0)$. $1$ is put if the edge is present in the set and $0$ otherwise. For example the set of all cycles can be written as

$$\mathcal{C}_G = \{(0, 0, 0, 0, 0), (1, 1, 0, 1, 0), (0, 0, 1, 1, 1), (1, 1, 1, 0, 1)\},$$

This form of presentation of graphs is easier to manipulate in computers, even though for humans it is hard to follow if the number of edges is large, since the vectors of the form $(\cdot, \cdot, \ldots, \cdot)$ are cumbersome.

# CHAPTER 3

# Image Pyramid

> ” We think too small, like the frog at the bottom of the well. He thinks the sky is only as big as the top of the well. If he surfaced, he would have an entirely different view. ”
>
> **by Mao Tse-Tung.**

**Summary** Pyramids are hierarchical structures that are able to transform local information into global one. Two processing paradigms are used in pyramids: fine-to-coarse and coarse-to-fine information processing. The pyramid is a trade off between parallel architecture and the need for a hierarchical representation of an image at several resolutions. In this chapter an overview of basic concepts of image pyramid are presented. Their structure, content and information processing are discussed in more detail.

**Keywords:** Image pyramids, regular pyramid, irregular pyramid, structure of pyramid.

## 3.1 Introduction

The visual data are characterized with a large amount of data and high redundancy, relevant information are clustered in space and time, all this indicates for a need of organization and aggregation principles, in order to cope with computational complexity and to bridge the gap between raw data and symbolic description. Local processing is important in early vision, since operation like convolution, thresholding, mathematical morphology etc. belong to this class of operations. However, this approach is not efficient for high or intermediate level vision, such as symbolic manipulation, feature extraction etc., because these processes need both local and global information. Therefore a data structure must allow the transformation of **local** information (based on sub-images) into **global** information (based on the whole image), and be able to

handle both local distributed and global centralized information. This data structure is known as *hierarchical architecture* [Jolion and Rosenfeld, 1994], and allows distribution of the global information to be used by local processes.

The earliest uses of hierarchical methods in image analysis are the work of [Rosenfeld and Thurston, 1971], finding edges by using differences of average gray levels in neighborhoods of many sizes, and selecting the best size at each pixel. The work of [Kelly, 1970] uses edgel detection in reduced resolution to guide the search for edges in a full-resolution image. This approach was one of the first to use a two level ($5 \times 5/25$) multi-resolution hierarchy for edge detection. First, edges and lines are found in the reduced resolution and then used as a plan to constrain the search in the higher resolution. The name *pyramids* were first coined in [Tanimoto and Pavlidis, 1975] as solution of contour detection and delineation in digital images, and in conjunction with the merge and split segmentation algorithm of [Horowitz and Pavlidis, 1976] are used extensively in image segmentation.

The hierarchy is a stack of levels of reduced abstraction. There are two types of hierarchies [Jolion and Rosenfeld, 1994]:

- visual hierarchy, and
- conceptual hierarchy.

The conceptual hierarchy represent object relationship, like class inclusion or neighborhood relations. An example of a possible hierarchy of abstraction is given in Figure 3.1.

Usually, the bottom layer of the pyramid, is the image and the top layer or *apex* is related to more abstraction levels. Information flows up, down and laterally in the hierarchy and is transformed between layers. In hierarchies there are two kind of processes:

- bottom-up, and
- top-down.

In bottom-up or *fine to coarse* processes, the information is transported from the bottom to the top of the hierarchy. Information of the local data set is transformed into global one recursively. Bottom-up or data driven analysis process data in the uniform matter and aggregate data into more abstract higher level of representation. This allows extraction and detection of important features in image. During this bottom-up processes the data volume is reduced, meaning that levels toward the top will contain less data.

In the top-down process or *fine to coarse* feature values are propagated from the top to the bottom of the hierarchy. The top-down processed use knowledge (in form of a model) to search for the image data in an nonuniform manner to find support or against the presence of particular structure in the image. This process allows to delineate features extracted by bottom-up processes by propagating the global information to lower levels or in the coarse to fine strategy the higher (coarse) levels propose hypothesis that are used by lower (finer) levels to verify these hypothesis.

The pyramid is a trade off between parallel architecture and the need for a hierarchical representation of an image at several resolutions [Jolion and Rosenfeld, 1994]. There are other multi-resolution approaches, just to mention some without trying to be complete. [Witkin, 1986] introduced the continuous scale-space theory and [Lindeberg, 1990] proposed a discrete version of this theory. [Mallat, 1989] introduced the wavelet theory, another hierarchical processing

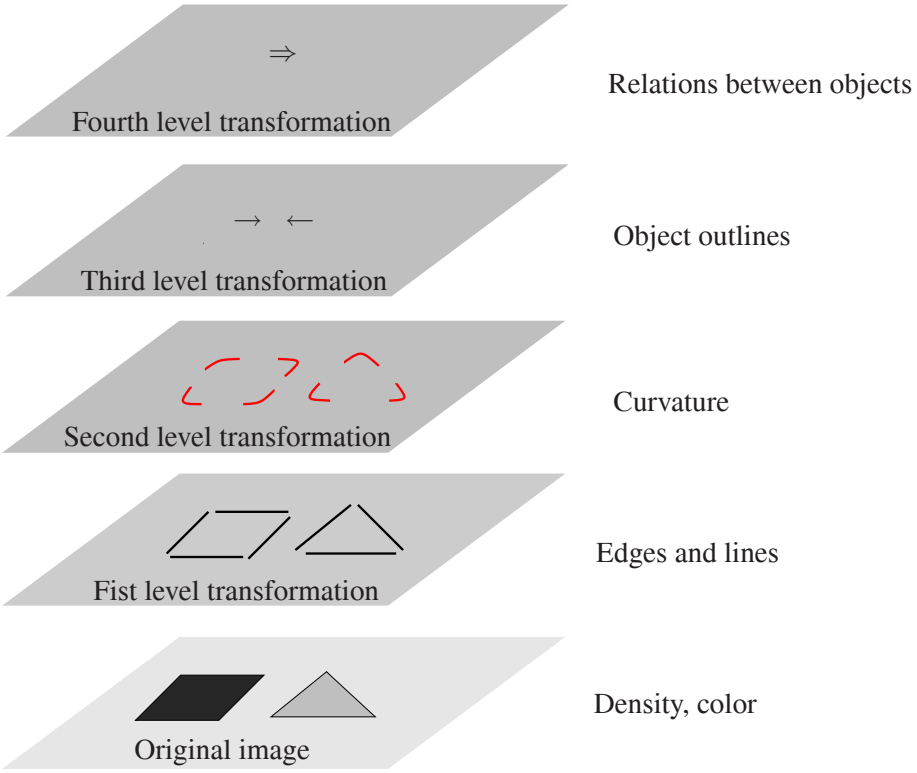



**Figure 3.1:** Conceptual representation of an image as an abstract hierarchy [Granlund, 1999].

paradigm. For an overview and the relation between different hierarchical approaches consult [Kropatsch, 1991].

In this chapter we summarize the concepts and definition developed for building and using multi-resolution pyramids [Rosenfeld, 1984], [Jolion and Rosenfeld, 1994], [Kropatsch, 1999]. First a definition of the image is presented in Section 3.2. Section 3.3 presents the architecture consideration of image pyramids. This chapter closes with a short summary in Section 3.4.

## 3.2 Discrete 2D Images

Almost every image processing technology processes the discrete spatial data by a computer. The function of mapping continuous space $\mathbb{R}^2$ into a discrete space $\mathbb{Z}^2$ is called *spatial quantization* or *digitisation* (in general one can think of a mapping from $\mathbb{R}^n$ into $\mathbb{Z}^n$). Seldom the sampling scheme is adapted to the local variability of the signal. Usually a regular sampling grid is considered, either a triangular or a square. In practice almost all capturing devices have a square regular sampling scheme, since the image sensors are arranged in 2D arrays located in the nodes of the regular grid. So the digital image is the result of sampling the continuous signal at location of the *sampling points*. A discrete representation of an image associates a numerical countable value with each point $(x, y)$ in the digitization grid. Since it is practically impossible to measure the signal in the infinite small surface area and in an infinite small time step, each value is in fact an average over a sampling window and over a time. Thus sampling

points can be considered as the centers of convex polygons [Soille, 1994], which tessellate the signal space. For non-regular grids these polygons are referred as Voronoi polygons, whereas for triangular grid this polygons have all hexagonal shape. In the case of square grids these polygons are called meshes. Usually these polygons are called in the digital image processing *pixels* (picture elements). The set of all pixels cover the entire image [Sonka et al., 1999].

The most primitive discrete representation assigns to each pixel an average measurement, which is in general a continuous value, but in image processing this value is *quantized*. The number of quantization levels should be high enough to allow humans a satisfactory perception of the image. Based on how many quantization levels are used to express the values of the pixel brightness (**g**), images are said to be binary or gray. In (multi-spectral) color images, vectors of scalar values are associated for each pixel. Hence a digital image is a finite set of triples:

$$(x, y, \mathbf{g}) \in \mathbb{Z}^3, \tag{3.1}$$

where $(x, y)$ is the location of the pixel in the digitization grid and **g** is the quantized brightness function. $x, y$ are usually called coordinated and are represented by using integer values. If **g** is a scalar of two values ($0$ and $1$) the image is called a binary images; if **g** is an integer from the interval $[0, 255]$, the image is called a gray value image; if **g** is a vector of scalars from the interval $[0, 255]$ representing red, green and blue color, the image is called a color image and so on. Detailed definition of the image types are found in [Soille, 1994, Sonka et al., 1999].

To summarize, discretization process maps any object of the continuous image into a discrete version if it is sufficiently large to be captured by the sensors at the sampling points. Resolution relates the unit distance of the sampling grid with a distance in reality. The properties of the continuous object, i.e. color, texture, shape, as well as its relations to other (nearby) objects are mapped into the discrete space, too.

In this document only binary, gray and color images are analyzed, it is supposed that images are quantized with enough brightness levels such that problem with false contour do not occur. It is also assumed that during the spatial digitization the proper topology of the image object is also captured. For techniques of spatial digitization that preserves the topology consult [Stelldinger and Ullrich, 2005]. Note that the framework presented in this document is general enough and is not limited in using only images as inputs.

## 3.3 Pyramid Architecture

Image pyramid have been defined as a stack of images of decreasing resolutions [Burt et al., 1981], [Rosenfeld, 1982]. This framework is used for efficient data processing in may areas of computer vision [Rosenfeld, 1984, Jolion and Rosenfeld, 1994]. Usually, higher levels of the pyramid are computed successively by a filtering operation followed by a re-sampling operator [Rosenfeld, 1984].

Image pyramid have these advantages [Bister et al., 1990, Kropatsch et al., 1999]:

- details are removed in lower resolutions, thus reducing the influence of noise,
- resolution independent processing of regions of interest,
- transformation of local information to global one,

**Table 3.1:** Image qualities at different resolutions [Kropatsch et al., 1999]

| Characteristics | Resolution | |
|---|---|---|
| | high | low |
| data amount | huge | smaller |
| computing times | long | short |
| details | rich and many | very few |
| overview | bad | good |
| precision | hight | low |

- divide-and-conquer principle applied to reduce the computational complexity, and
- finding models at lower resolution, thus ignoring details, and employing these models in a low cost top-down model based analysis.

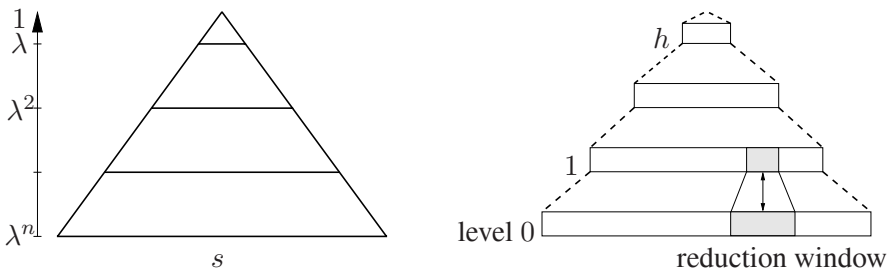
In Table 3.3 some qualities of different resolutions are shown (adapted from [Kropatsch et al., 1999]).

Image pyramids are characterized by these important properties [Jolion and Rosenfeld, 1994, Bischof, 1995]:

- structure[1], e.g. vertical and horizontal neighborhood relations
- content of the cell, e.g. pixel, region, edge, curve or more, and
- processing of the cells, e.g. filtering, symbolic processing.

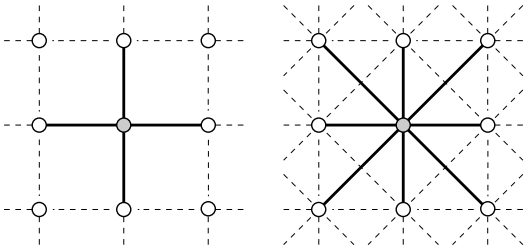
A pyramid (Fig. 3.2a,b) describes the contents of an image at multiple levels of resolution. High resolution input image is at the base level. Successive levels reduce the size of the data by *reduction factor* $\lambda > 1.0$. *Reduction windows* relate one cell at the reduced level with a set of cells in the level directly below. Thus local independent (and parallel) processes propagate information up and down and laterally in the pyramid. The contents of a lower resolution cell is computed by means of a *reduction function* the input of which are the descriptions of the

[1] also called communication network.




(a) Pyramid concept (b) Discrete levels

**Figure 3.2:** Multiresolution pyramid.



(a) 4-neighborhood (b) 8-neighborhood

**Figure 3.3:** Neighborhood in a square grid graph.

cells in the reduction window. Two successive level of a pyramid are related by the reduction window and the reduction factor. Higher level description should be related to the original input data in the base of the pyramid. This is done by the *receptive field* (RF) of a given pyramidal cell $c_i$. The $RF(c_i)$ aggregates all cells (pixels) in the base level of which $c_i$ is the ancestor. In the sections below these notions are explained in more detail following the work of [Jolion and Rosenfeld, 1994], [Bischof, 1995] and [Kropatsch et al., 1999].

### 3.3.1 Structure

The structure of the image pyramid is made of two types of neighborhood: the *horizontal*, i.e. intra-level neighborhood, and the *vertical*, i.e. inter-level neighborhood are defined in the image pyramid. Each cell of the pyramid is related with its neighbors in the same level and with other cells in the level above and below, except cells on the base and cell(s) on top of the pyramid. Cells on the base have only relation with the level above, and the cell(s) on the top has(ve) only relations with the level below. Let each cell (e.g. pixel, edge, region etc.) of the pyramid be represented by a vertex of the graph, and each level $k$ of the pyramid by represented by a graph. Thus the horizontal and vertical network can be defined using graphs.

#### Horizontal neighborhood

Let level $k$ of the image pyramid be represented by $G_k = (V_k, E_k)$, where $V_k$ represent the set of cells, and $E_k$ the set of edges joining cells Every vertex $v \in V_k$ on level, say $k$, is related to its neighbors on the same level by edge(s) $e \in E_k$. Two vertices, $v, w \in V_k$ are neighbors if they are joint by an edge $e = (v, w) \in E_k$ (Definition 2.3). This defines the horizontal neighborhood. If the image plane is regularly tessellated, say by a regular grid mesh, than on the base level one can define $4$ or $8$ connectivity of cells, as shown in Figure 3.3, the cell (vertex) in gray in (a) has 4-neighborhood and under (b) 8-neighborhood. Note that $8$-neighborhood grid graph is not planar. Other neighborhoods are possible, and are discussed in more detail in Chapter 4.

#### Vertical neighborhood

Every vertex in level $k$ is linked with vertices on level directly below $k - 1$, its *children(s)*, and vertices on level directly above $k + 1$, its *parent(s)*. Vertices on the base level have no children(s), and those on the top have no parent(s). The number and form of these links define

the vertical neighborhood, and this neighborhood be defined by an undirected (vertical) graph $G_k^V = (R_k, L_k)$, where $R_k = (V_k \cup V_{k+1})$, and $L_k \subseteq (V_k \times V_{k+1})$. The children relation of vertex $v \in V_{k+1}$ is defined as:

$$Ch(v, w) = \{\exists w | e = (v, w) \in L_k\}; \tag{3.2}$$

we say vertex $w$ is a children of $v$. Analogously, the parent relation of a vertex $w \in V_k$ as:

$$P(w, v) = \{\exists v | e = (w, v) \in L_k\}; \tag{3.3}$$

we say vertex $v$ is a parent of $w$. Note that these relations, are non-reflexive, anti-symmetric, and non-transitive. A parent has one or more children and in general a child can have many parents. If a child has many parent, the pyramid is called an *overlapping pyramid*. Based on these definition, through the transitive closure of these graphs, the *ancestor(s)* and *descendant(s)* of a given vertex can be defined. Let $h$ be the height of the pyramid. The ancestor of a vertex $v \in V_k$ is vertex $w \in V_l$, $h \geq l > k$ if and only if:

$$A(v, w) = \{\exists z_n \in V_n \,|\, P(z_n, z_{n+1}) \forall n = k, ..., l-1\}, \text{ where } v = z_k \text{ and } w = z_l \tag{3.4}$$

Analogously, the descendant of a vertex $w \in V_l$ is vertex $v \in V_k$, $l > k \geq 0$ if and only if:

$$D(w, v) = \{\exists z_n \in V_n \,|\, Ch(z_n, z_{n-1}) \forall n = l, ..., k-1\}, \text{ where } w = z_l \text{ and } v = z_k \tag{3.5}$$

These definition can be used to define the *receptive field* of a vertex, as the set of all its descendants on the base level of the pyramid. Formally, the receptive field is the set of vertices on the base level $G_0 = (V_0, E_0)$ which influence the cell $v \in V_k$:

$$RF(v) = \{ \text{ all vertices } v' \in V_0 \,|\, D(v, v')\} \tag{3.6}$$

One can define also the *projective field* [Bischof, 1995] of vertex $v \in V_m$ on a given level $V_n$, $(n > m)$ as:

$$PF(v) = \{ \text{ all vertices } v' \in V_n \,|\, A(v, v')\} \tag{3.7}$$

In Figure 3.4, pictorially the concepts of parent-children, and ancestor-descendant relation are given with solid line. If we assume that $G_{k-1} = G_0$ is the bottom of the pyramid, then in this figure all the cells $w'$ at the bottom enclosed in the gray area are the receptive field of $v : RF(v)$.

Two types of pyramid exist, based on how the vertical neighborhood is defined:

- regular, and
- irregular pyramids.

These concepts are strongly related to ability of the pyramid to represent the regular and irregular tessellation of the image plane.

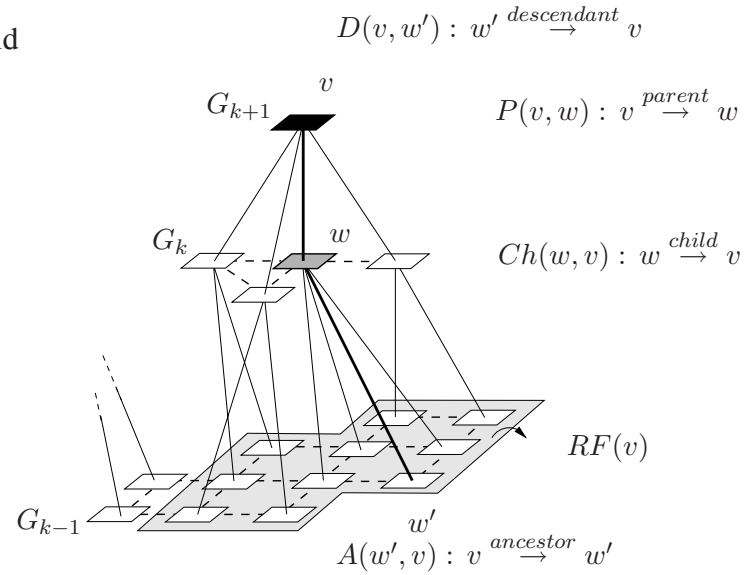
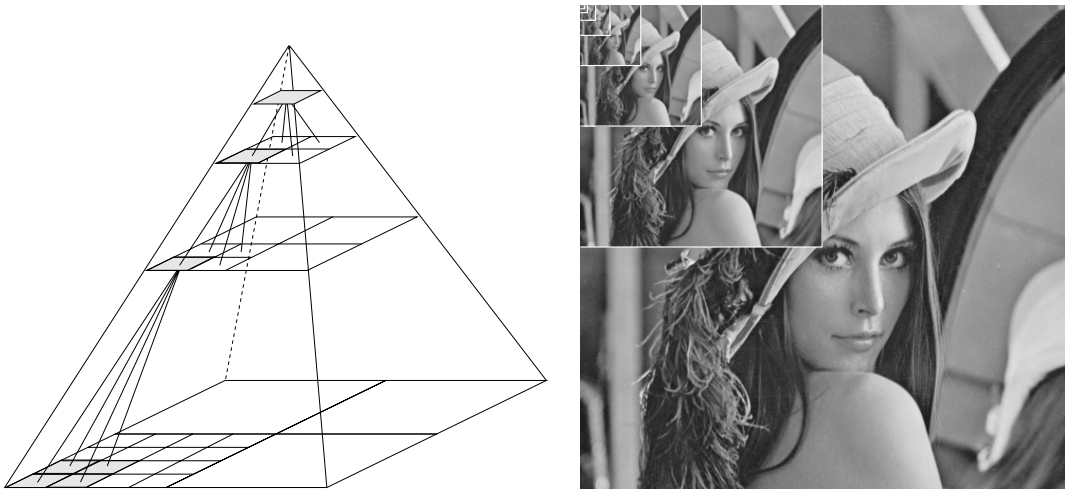



**Figure 3.4:** Vertical neighborhood.

### Regular Pyramids

The *constant reduction factor* and *constant size reduction window* completely define the structure of the regular pyramid. The decrease rate of cells from level to level is determined by the reduction factor. The number of levels $h$ is limited by the reduction factor $\lambda > 1$: $h \leq \log(image\_size)/\log(\lambda)$. The *main computational advantage* of regular image pyramids is due to this *logarithmic complexity*. Usually regular pyramids are employed in a regular grid tessellated image plane, therefore the reduction window is usually a square of $n \times n$, i.e. the $n \times n$ cells are associated by a cell on a higher level directly above. Regular pyramid are denoted using notation $n \times n/\lambda$. The vertical structure of a classical $2 \times 2/4$ is given in Figure 3.5a. In this regular pyramid $2 \times 2 = 4$ cells are related to only one cell in the higher level directly above. Since the children have only one father this class of pyramids is also called non-overlapping regular pyramids. Therefore the reduction factor is $\lambda = 4$. An example of $2 \times 2/4$ regular image pyramid is given in Figure 3.5b. The image size is $512 \times 512 = 2^9 \times 2^9$ therefore the image pyramid consist of $1 + 2 \cdot 2 + 4 \cdot 4 + ... + 2^8 \times 2^8 + 2^9 \times 2^9$ cells, and the height of this pyramid is $9$. The pyramid levels are shown by a white border on the left upper corner of image. The $2 \times 2/4$ regular pyramid is called also the quad pyramid, because of the similarity with the quad tree representation [Samet, 1990]. See [Kropatsch, 1991] for extensive overview of other pyramid structures with overlapping reduction windows, e.g. $3 \times 3/2$, $5 \times 5/4$. It is possible to define pyramids on other plane tessellation, e.g. triangular tessellation [Jolion and Rosenfeld, 1994]

Thus, the regular image pyramid are efficient structure for fast grouping and access to image objects across the input image, because of the rigid vertical structure. globally defined sampling grids and lack shift invariance [Bister et al., 1990]. The regular pyramid representation of a shifted, rotated and/or scaled image is not unique, and moreover it does not preserve the connectivity. Thus, [Bister et al., 1990] concludes that regular image pyramids have to be rejected as general-purpose segmentation algorithms. This major drawback of the regular pyramid motivated a search for a structure that is able to adapt on the image data. It means, that the regularity of the structure is to be abandon.



(a) Vertical structure (b) Image pyramid

**Figure 3.5:** $2 \times 2/4$ regular pyramid.

### Irregular Pyramids

Abandoning the regularity of the structure means that the horizontal and vertical neighborhood have to be explicitly represented, usually by using graph formalism. These not-regular structures are usually called *irregular pyramids*. One of the main goals of irregular pyramids is to achieve the shift invariance, and to overcome this major drawback of regular counterparts. [Kropatsch et al., 1999] list other motivations why one has to use irregular structures:

- arrangement of biological vision sensors is not completely regular,
- the CCD cameras cannot be produced without failure, resulting into an irregular sensor geometry,
- perturbation may destroy the regularity of regular pyramid, and
- image processing to arbitrary pixels arrangement (e.g. log-polar geometries [Bederson, 1992])

Two main processing characteristics of the regular pyramids should be preserved by building irregular ones [Bischof, 1995]:

- operation are local, i.e. the result is computed independently of the order, this allows parallelization, and
- bottom-up building of the irregular pyramid, with an exponentially decimation of the number of cells.

The structure of the regular pyramid as well as the reduction process is determined by the type of the pyramid (e.g. $2 \times 2/4$). Removing this regularity constraint one has to define a procedure to derive the structure of the reduced graph $G_{k+1}$ from $G_k$, i.e. a graph contraction

method has to be defined. Irregular pyramid can be build by parallel graph contraction [Rosenfeld, 1985], or graph decimation [Meer, 1989]. Parallel graph contraction has been developed only for special graph structures, like trees, and will not be analyzed in this thesis.

[Meer, 1989] introduced an efficient random decimation algorithm for building regular pyramid, called *stochastic pyramid*. A detailed discussion of this and similar methods is postponed until Chapter 5. It is shown that stochastic pyramid in some cases is not logarithmically tapered, i.e. the decimation process does not exponentially reduce the number of cell successively. The main reason for this behavior is that the cell's neighborhood is not bounded, for some cases the degree of the cell increases exponentially. In Chapter 5 we discuss methods that overcome this drawback.

An overview of properties of regular and irregular pyramid are found in [Kropatsch and Montanvert, 1991b]. In irregular pyramids the flexibility is paid by less efficient data access.

### 3.3.2 Contents

The type of information stored into cells defines also the the content of the pyramid. Numeric and/or symbolic information can be stored into the cells. A cell store the information that it is gathered (e.g. by a bottom-up process) from its receptive field. [Kropatsch, 1991] defines two conditions to be fulfilled by the representation of the cell:

- the receptive field of the cell covers the pictorial entity (e.g. primitive object parts, objects) completely, and
- no smaller cells (in lower levels) fulfill the above property

A unique cell in the pyramid is allocated for each pictorial entity. Usually, a cell stores only one numeric value, e.g. a gray value. This pyramid is called *numeric pyramid*. More complex numeric values can be stored as well, e.g. parameters of a model. Symbolic information can be stored into cells as well, e.g. curve information [Kropatsch, 1986, Kropatsch and Burge, 1998]. This class of pyramid is called *symbolic pyramid*. It is also possible to mix numeric and symbolic information into a cell as well.

### 3.3.3 Processing

The type of the information stored into cells of the pyramid conditioned also the processing that can be carried out by the pyramid. Therefore, two types of information processing are performed:

- numeric, and
- symbolic processing

Note that one of the major properties of the pyramid is the capability of local processing, i.e. the information in the cells is computed using only other cells in their horizontal and vertical neighborhood, thus these processing can be done in parallel. For an overview of pyramid hardware implementations see [Jolion and Rosenfeld, 1994].

**Numeric Processing**

Filtering is usually used as a reduction function in numerical pyramids. Different types of filters are used: 1) linear filters, 2) non-linear filters, and 3) morphological filters. Most commonly used filters are the low-pass ones. Gaussian filter is one of the most used low-pass linear filters in regular pyramid architecture, since it is the only filter that preserves the zero crossings across the scales [Yuille and Poggio, 1986]. An example of the Guassian pyramid is given in Figure 3.5b. The resulted pyramid is called *Gaussian pyramid* [Burt and Adelson, 1983]. Minimum and the maximum filter are most commonly used non-liner filters.These filters compute the minimum, respectively maximum, of the receptive field of the cell. One can also use other non-linear filters, e.g. median. The morphological operators [Serra, 1982] are introduced into pyramids as well. A large body of the literature exist of using different filters integrated into a pyramid.

**Symbolic Processing**

In symbolic pyramid one has to define also symbolic reduction functions. A finite state machine can be used to compute the symbolic reduction. E.g. [Kropatsch, 1986, Kropatsch and Burge, 1998] introduces a set of rules as reduction function.

## 3.4 Summary

All pyramids are characterized by three properties: its structure (horizontal and vertical neighborhood), its cell content and the way it processes the information content of the cells. We distinguish two main types of pyramids based on the structure: if the structure is beforehand defined the pyramid is called regular; if the structure is adapted on the image data it is called an irregular pyramid. Numeric or/and symbolic information can be stored into the pyramid, therefore one can differentiate between numeric pyramids and symbolic pyramids. Different filters can be used in the pyramid framework, which allows also symbolic computations. In the rest of the chapters we will intensify the discussion on irregular graph pyramids. We will discuss how to optimize the structure of the irregular pyramid, trying to archive a logarithmic height, and how to apply efficiently these structures in image segmentation.

CHAPTER 4

# Irregular Dual Graph Pyramids

> ” Beauty is the first test: there is no permanent place in the world of mathematics for ugly mathematics. ”
>
> **by G.H. Hardy.**

**Summary** The duality concept in planar graphs is presented. It is shown how the dual graphs can encode a topology properly. The transformation of image plane into a dual graph is described. The dual graph contraction, as a topology preserving graph contraction is presented in depth. This graph contraction method contains basic operations to build a stack of hierarchical graphs, called dual graph pyramid. The construction of the dual graph pyramid is shown by an example.

**Keywords:** Topology, dual graph, planar graph, dual graph contraction, dual graph pyramids, topology preserving contraction.

## 4.1 Introduction

Most information in vision today is in the form of array representation. This is advantageous and easily manageable for situations having the same resolution, size, and other typical properties equivalent. Various demands are appearing upon more flexibility and performance, which makes the use of array representation less attractive [Granlund, 1999]. The increasing use of actively controlled and multiple sensors requires a more flexible processing and representation structure [Kropatsch et al., 1999, Kropatsch, 2002]. Cheaper CCD sensor could be produced if defective pixels would be allowed, which yields in the resulting irregular sensor geometry [Bederson, 1992], [Wallace et al., 1994]. Image processing functions should be generalized to arbitrary pixel geometries [Rojer and Schwartz, 1990], [Bederson, 1992]. The conventional

array form of image is impractical as it has to be searched and processed every time if some action is to be performed and that

- features of interest my be very sparse over parts of an array, leaving a large number of unused positions in the array;
- a description of additional detail can not be easily added to a particular part of an array.

It is desirable to have information in some partly interpreted form to fulfill its purpose to rapidly evoke actions. Information in interpreted form, implies that it should be represented in terms of content or *semantic* information, rather than in terms of array values. Content and semantics implies *relations* between units of information or symbols. [Granlund, 1999] calls the relations between object as *linked objects*. These object could be represented as a *graph* [Haralick and Shapiro, 1993].

In order to express the connectivity or other geometric or topological properties the image representation must be enhanced by a neighborhood relation. In the regular square grid arrangement of sampling points it is implicitly encoded as $4$- or $8$-neighborhood with the well known paradox in conjunction with Jordan's curve theorem. The neighborhood of sampling points can be represented explicitly, too: in this case the sampling grid is represented by a *graph* consisting of vertices corresponding to the sampling points and of edges connecting neighboring vertices. Although this data structure consumes more memory space it has several advantages, as follows [Kropatsch et al., 1999]:

- the sampling points need not be arranged in a regular grid,
- the edges can receive additional attributes too, and
- the edges may be determined either automatically or depending on the data.

Planarity and duality of graph are two closely related concepts. Planar graph separates the plane into regions called faces. This idea of separating the plane into regions is helpful in defining the dual graphs (Section 4.2). Duality of a graph brings together two important concepts in graph theory: cycles and cut-sets. Kirkoff's laws of voltage and current in electrical engineering are the real wold problem of this duality concept. The law of voltage is in term of cycles and the law of current is in term of cut-sets. This concept of duality is also encountered in graph-theoretical approach of image region and edge extraction. The definition of dual graphs representing the partitioning of the plane, allows one to apply transformation on these graphs, like edge contraction and/or removal to simplify graphs in the sense of less vertices and edges. Edge contraction and removal introduces naturally a hierarchy of dual graphs, the so called *dual graph pyramid*.

Hence the dual graph representation presented in this chapter addresses primarily the structure, on which a dual graph hierarchy is built, by reducing the number of descriptive elements by applying the dual graph contraction successively. In this chapter, the graph-based image representation and the operation on these graphs are given. In Section 4.3 the transformation of image plane into a dual graph is shown. The analogy of dual graphs with abstract cellular complexes, and the equivalence with combinatorial maps is given in Section 4.3.1, and Section 4.3.2 respectively. The dual edge contraction algorithm is described in Section 4.4 and the hierarchy of graphs build by this algorithm in Section 4.5.

## 4.2 Planar and Dual Graphs

A graph $\widetilde{G}$ of finite sets of vertices $V$ and edges $E$ is called a *plane graph* if it is drawn in a plane in $\mathbb{R}^2$ such that [Diestel, 1997]:

- all $V \subset \mathbb{R}^2$
- every edge is an arc[1] between two vertices,
- no two edges are crossed.

Note that $\mathbb{R} \setminus \widetilde{G}$ is an open set and its connected regions are faces $f$ of $\widetilde{G}$. It is said that the plane graph divides the plane into regions. Since $\widetilde{G}$ is finite, one of its faces in an unbounded one (infinite area). This face is called the *background face*[2]. Other faces enclose finite areas, and are called interior faces. Edges and vertices incident with the face are called the boundary elements.

**Definition 4.1** *An embedding of a graph $G$ on a plane is an isomorphism between $G$ and a plane graph $\widetilde{G}$.*

$\widetilde{G}$ is called a drawing of $G$. When $G$ is drawn, we do not care whether its edges intersect on vertices only or each other. This is in contrast to drawing $\widetilde{G}$ where its edges intersect only on vertices. The graph $G$ can be considered as an abstraction of $\widetilde{G}$. An example of a planar graph and its planar embedding is shown in Figure 4.1. The edge $e_5$ in Figure 4.1b is drawn such that it does not intersect with other edges, whereas in Figure 4.1b we do not ask strictly for edges not to intersect. In this figure $f$ represent faces, and a background face is denoted by $b$. For example the edges $e_1, e_2$ and $e_5$ form the face $f_1$ and they make the cycle $C_{f_1} = \{e_1, e_2, e_5\}$, which enclose the region $f_1$. The cycle that enclose the background face $b$ in this figure is $C_b = \{e_5, e_8, e_6\}$.

**Definition 4.2 (Planar graphs)** *A graph $G$ is planar if it can be embedded on the plane.*

The concept of embeddings can be extended to any surface. A graph $G$ is embeddable in surface $S$ if it can be drawn in $S$ so that its edges intersect only on their end vertices. A graph embeddable on the plane is embeddable on the sphere too. It can be shown by using the stereoscopic projection of the sphere onto a plane [Thulasiraman and Swamy, 1992]. Note that the concept of faces is also applicable to spherical embeddings.

A planar graph $G = (V, E)$ with $m$ vertices, $n$ edges and $l$ faces satisfies the condition:

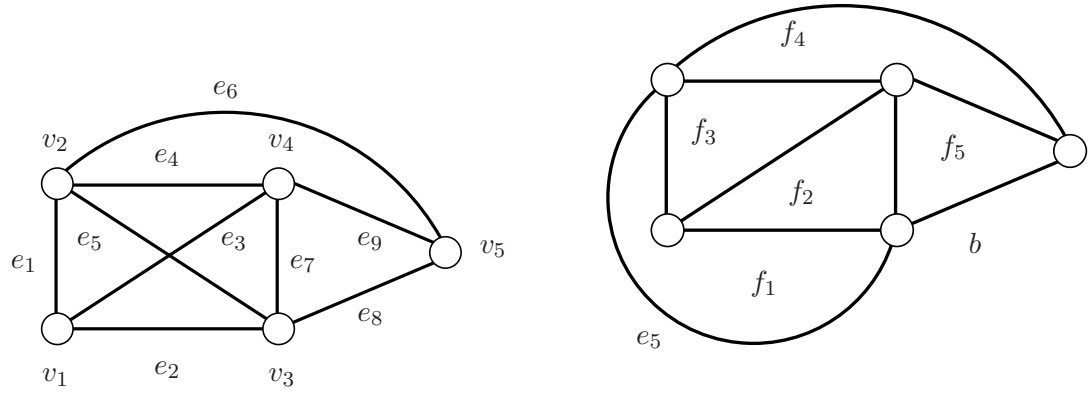
$$m - n + l = 2. \tag{4.1}$$

This is called Euler's formula[3]. Another nice property of simple planar graph is that if $G$ has $m \geq 3$ vertices and $n$ edges than $n \leq 3m - 6$ edges. The so called Kuratowski graphs $K_5$

[1] An arc is a finite union of straight line segments, and a straight line segment in the Euclidean plane is a subset of $\mathbb{R}^2$ of the form $\{x + \lambda(y - x) | 0 \leq \lambda \leq 1\}, \forall\, x \neq y \in \mathbb{R}^2$.

[2] Called also exterior face.

[3] Called also Euler characteristic.




(a) $G = (V, E)$ (b) $\widetilde{G}$

**Figure 4.1:** A planar graph $G$ and its embedding in a plane, the plane graph $\widetilde{G}$.

(complete graph on 5 vertices) and $K_{3,3}$ are not planar. The planar graph can be characterized also using the Kuratowski graphs [Harary, 1969].

Let $G$ in Figure 4.1a represent a plane graph, in general with parallel edges and self-loops. Since the graph is embedded onto a plane, it divides the plane into faces. Let each of these faces be denoted by a new vertex say $f$s, and let these vertices be put inside the faces, as shown in Figure 4.1b. From this point on the notion of face vertices and face are synonymous. Let the faces that are neighbors, i.e. that share the same edge $e_2$ (incident on the same edge), be connected by the edge, say $\overline{e}_2$, so that edge $e$ and $\overline{e}$ are crossed. At the end, for each edge $e_2 \in G$ there is an edge $\overline{e}_2$ of the newly created graph $\overline{G}$, which is called the dual graph of $G$. If the $e$ is incident only with one face a self-loop edge $\overline{e}_2$ is attached to the vertex on the face in which the edge $e_2$ lays, of course $e_2$ and the self-loop edge $\overline{e}_2$ have to cross each other (see Appendix B for a procedure for building dual graph out of plane ones). The adjacency of faces is expressed by the graph $\overline{G}$.

More formally one can define dual graphs for a given plane graph $G = (V, E)$ in this form [Thulasiraman and Swamy, 1992]:

**Definition 4.3 (Dual graphs)** *A graph $\overline{G} = (\overline{V}, \overline{E})$ is a dual of $G = (V, E)$ if there is bijection between the edges of $G$ and $\overline{G}$, such that a set of edges in $\overline{G}$ is a cycle vector of $\overline{G}$ if and only if the corresponding set of edges in $G$ is a cut vector.*

There is a one-to-one correspondence between the vertex set $\overline{V}$ of $\overline{G}$ and the face set $F$ of $G$, therefore sometimes graph $\overline{G} = (\overline{V}, \overline{E})$ is written as $\overline{G} = (F, \overline{E})$ instead, without fear of confusion. In order to show that $\overline{G}$ is a dual of $G$, one has to prove that vectors forming a basis of the cycle subspace of $\overline{G}$ correspond to the vectors forming a basis of the cut subspace of $G$. The edges $e_i$ of graph $G$ in Figure 4.2 correspond to edges $\overline{e}_i$ in graph $\overline{G}$. The cycles $\{e_1, e_3, e_4\}$, $\{e_2, e_3, e_6\}$, $\{e_4, e_5, e_8\}$, and $\{e_6, e_7, e_8\}$ form a basis of the cycle subspace of $G$ (see Chapter 2, Section 2.7). These cycles correspond to the set of edges $\{\overline{e}_1, \overline{e}_3, \overline{e}_4\}$, $\{\overline{e}_2, \overline{e}_3, \overline{e}_6\}$, $\{\overline{e}_4, \overline{e}_5, \overline{e}_8\}$, and $\{\overline{e}_6, \overline{e}_7, \overline{e}_8\}$, which form a basis of cut subspace of $\overline{G}$. It follows according to the definition of the duality, that graph $\overline{G}$ is a dual of $G$. By convention, the graph $G$ is called the *primal graph* and $\overline{G}$ *dual graph*. If a planar graph $G'$ is the dual of $G$,

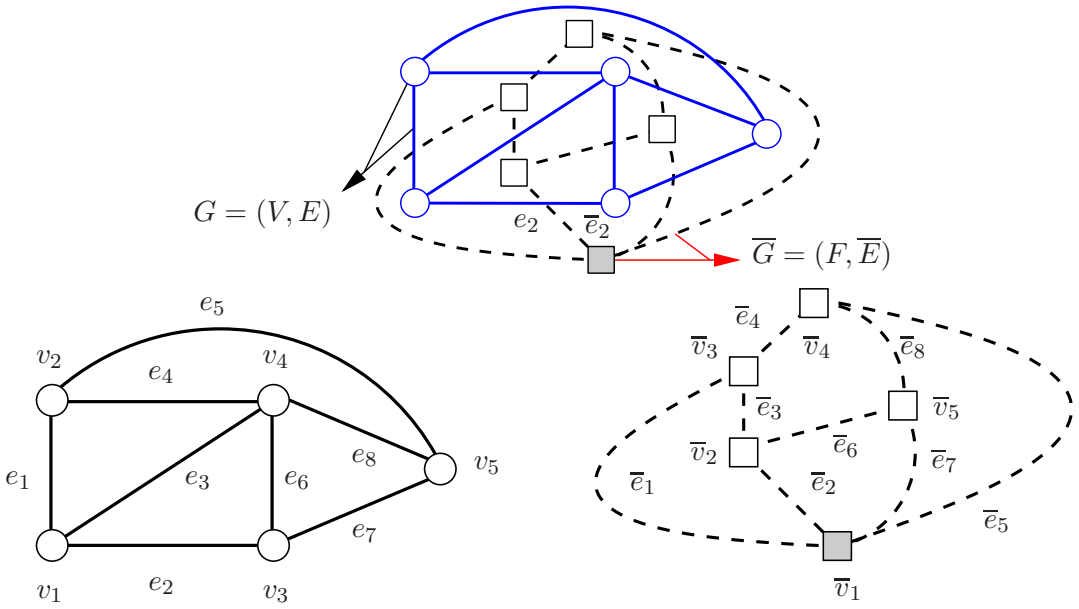



**Figure 4.2:** A plane graph $G$ and it dual $\overline{G}$.

then a planar $G$ is a dual of $G'$ as well, and every planar graph has a dual [Diestel, 1997, Harary, 1969]. Therefore the dual of a dual graph is the primal graph. Dual graphs are denoted by a bar above the capital letter.

In the following two important properties of dual graphs with respect to the edge contraction and removal operation are given, the proofs are due to [Thulasiraman and Swamy, 1992]. These properties are used in Section 4.4 to prove that graphs during the process of dual graph contraction stay planar and are duals. Let $G$ and $\overline{G}$ be two graphs. Let edge $\overline{e} \in \overline{G}$ correspond to edge $e \in G$. Note that a cycle in $G$ corresponds to a cut in $\overline{G}$ and vice versa [Thulasiraman and Swamy, 1992]. Let $\overline{G'}$ denote the graph $\overline{G}$ after the *contraction* of the edge $\overline{e}$, and $G'$ the graph after the *removal* of the corresponding edge $e$ from $G$.

**Theorem 4.1** *A graph and its dual are duals also after the removal of an edge $e$ in the primal graph $G$ and the contraction of the corresponding edge $\overline{e}$ in the dual graph $\overline{G}$.*

*Proof:* Let $C$ and $\overline{C}$ be the corresponding set of edges in $G$ and $\overline{G}$, respectively. Assume that $C$ is a cycle in $G'$. Since it does not contain $e$, it is also a cycle in $G$. Hence $\overline{C}$ is a cut, say $\{\overline{V_1}, \overline{V_2}\}$ in $\overline{G}$. Since cut $\overline{C}$ does not contain $\overline{e} = (\overline{v_1}, \overline{v_2})$, the vertices $\overline{v_1}$ and $\overline{v_2}$ are both either in $\overline{V_1}$ or in $\overline{V_2}$, implying $\overline{C}$ is a cut in $\overline{G'}$. Therefore every cycle in $G'$ is a cut in $\overline{G'}$.

Suppose that $\overline{C}$ is a cut in $\overline{G'}$. Since $\overline{C}$ does not contain $\overline{e}$, it is also a cut in $\overline{G}$. Hence $C$ is a cycle in $G'$. Since it does not contain $e$, it is also a cycle in $G$. Thus every cut in $\overline{G'}$ is a cycle in $G'$. □

**Corollary 4.1** *If a graph $G$ has a dual, then every edge-induced subgraph of $G$ has also a dual.*

*Proof:* Every edge-induced subgraph $G'$ of $G$ can be obtained by removing from $G$ the edges not in $G'$ and using the Theorem 4.1, the proof follows. □

This section is concluded by the a topological characterization of graphs that have duals. The duality property of a graph is as important as planarity, and these properties are symbiotics.

**Theorem 4.2 (Whitney 1933)** *A graph is planar if and only if it has a dual.*

*Proof:* The proof can be found in [Diestel, 1997]. □

A detailed discussion of the data structures for the dual graph are given in Appendix C.

## 4.3 Dual Image Graphs

An image is transformed into a graph such that, for each pixel a vertex is associated, and pixels that are neighbors in the sampling grid are joint by an edge . Note that no restriction in the sampling grid is made, therefore an image of regular as well as non-regular sampling grid can be transformed into a graph. The gray value or any other feature is simply considered as an attribute of a vertex (and/or an edge). Since the image is finite and connected, the graph is finite and connected as well. The graph which represent the pixels is denoted by $G = (V, E)$ and is called *primal graph*[4]. Note that pixels represent finite regions, and the graph $G$ is representing in fact a graph with faces as vertices. The dual of a face graph (see Section 4.2) is the graph representing borders of the faces, which in fact are inter-pixel edges and inter-pixel vertices [Braquelaire and Brun, 1998, Kropatsch, 1994]. This graph is denoted by $\overline{G}$ and is called simply *dual graph*. This discussion, which is done on purpose here to show the duality concept, is not in contradiction with the presentation of dual graphs given in Section 4.2, because of the property of dual graphs to be dual of each other. The above discussion could have been started by defining the primary graph as the graph denoting borders of faces, and its dual would have been the graph representing the faces. An example of image with square grid sampling transformed into a graph is given in Figure 4.3, where square vertices represent faces, the bold square vertex represents the background face, circle vertices represent meeting points of at least three boundary segments.

Based on the Theorem 4.2 dual graphs are planar, therefore images with square grid are transformed into $4-$ connected square grid graphs, since $8-$ connected square grid graphs are in general not planar [5].

The same formalism as is done for the pixels can be used at intermediate levels in image analysis i.e. for region adjacency graphs (RAGs). RAGs are the results of image segmentation processes. Region are connected sets of pixels, and are separated by region borders. Its geometric dual causes problems [Kropatsch, 1995a]. Let a simple example, adapted from [Kropatsch, 1994], clarifies this claim and motivate the usage of pair of graph (duals of each other) to represent an image or in general adjacency of regions. Note that in this presentation the graph and its duals are depicted as in Section 4.2, and is different with the above presentation of dual graphs. In the example of the house in Figure 4.4a the graph $\overline{G'}$ representing regions of the house, like door, windows etc., is depicted by square vertices and dashed edges. To reconstruct the boundary graph $G'$ out of the face graph $\overline{G'}$, circle vertices are put where at least three boundary segments intersect. The edges of $G$ are drawn between these vertices following the boundary

[4] Called also neighborhood graph.

[5] This hold for square grid graphs of grid size $\geq 4 \times 4$.

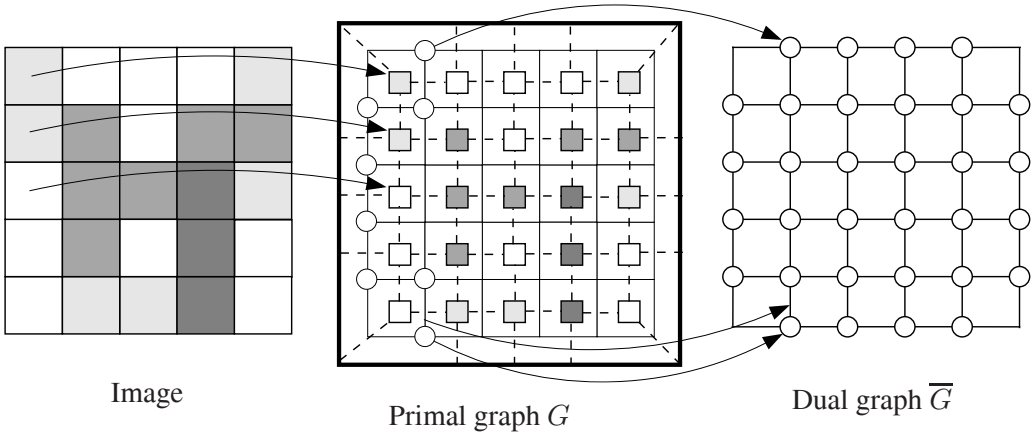


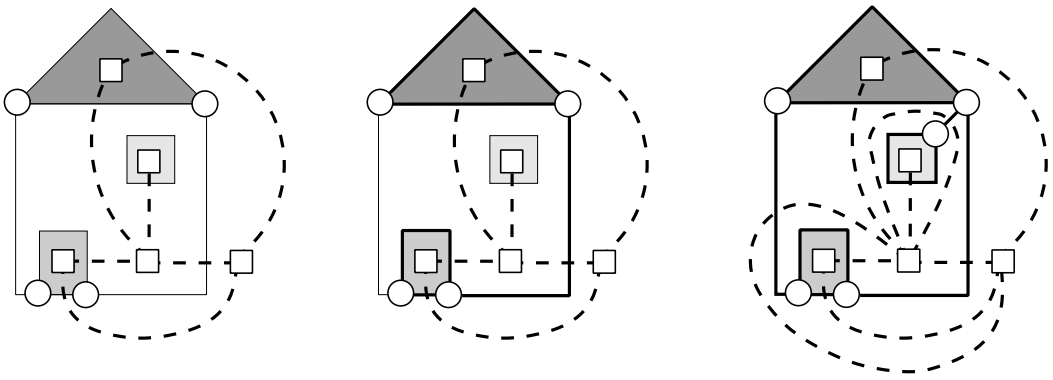

**Figure 4.3:** Image to dual graphs.

of the house regions such that these edges cross the edges of the graph $\overline{G'}$ (as shown with bold lines in Figure 4.4b. As can be seen from the Figure 4.4b there are two problems:

- one boundary of the front side of the house is not crossed by any edge in $\overline{G'}$, and
- the above described procedure will not produce an edge for the window since there is no crossing of the border of the window with the surrounding region of the front side, and therefore no clear answer where to put a vertex of $G'$.

The problems are encountered due to the fact that the border of the front side consists of two parts not connected to each other: the inner part of the window, and the outer part which is also fragmented into pieces separating the front from the roof, from the background and from the door. A resolution to these problems is given in Figure 4.4c: a self loop is put in $\overline{G'}$ surrounding the window, a circle vertex is put in $G'$ arbitrarily in the window border and this vertex is connected by a fictive edge [6] such that it connects the boundary of the window with the front

[6] Called a bridge in [Kropatsch, 1994].



(a) RAG $\overline{G'}$ (b) the dual of $\overline{G'}$ : $G' = \overline{\overline{G'}}$ (c) dual image graphs $(G, \overline{G})$

**Figure 4.4:** A house example and the dual image graph [Kropatsch, 1994].

side of the house, and front side and background are connected by parallel edges in $\overline{G}$. The resulting pair of graph $(G, \overline{G})$ are plane, dual of each other, and in general not simple, it contains self-loops and parallel edges. This section is concluded by a formal definition of the dual image graphs:

**Definition 4.4 (Dual image graphs [Kropatsch, 1994])** *The pair of graphs $(G, \overline{G})$, where $G = (V, E)$ and $\overline{G} = (\overline{V}, \overline{E})$ are called dual image graphs if both of the graphs are finite, planar, connected, not simple in general and duals of each other.*

Dual graphs can be seen as an extension of the well know region adjacency graphs (RAG). Note that this representation is capable to encode multiple boundaries between neighboring regions. See the outside border of the house in Figure 4.4c connected multiple times (roof, left wall, part of the door and right wall) with the background.

### 4.3.1 Dual Image Graph and Cellular Complexes

In this section a relation between dual graphs and 2D-abstract cellular cells (ACC) is shown. [Kovalevsky, 1989] showed that on the abstract cellular cells one can define a topological space. Topology is formally defined as:

**Definition 4.5 (Topology)** *Let $\mathcal{X}$ be a non-empty set, the universe. A topology on $\mathcal{X}$ is a family $\mathcal{T}$ of subsets of $\mathcal{X}$ such that:*

*(T1) $\mathcal{X}$ and $\emptyset$ belong to $\mathcal{T}$,*

*(T2) the union of any number of sets of $\mathcal{T}$ belongs to $\mathcal{T}$,*

*(T3) the intersection of any two sets of $\mathcal{T}$ belongs to $\mathcal{T}$.*

A set $\mathcal{X}$ for which a topology $\mathcal{T}$ has been specified is called a topological space. The members of $\mathcal{T}$ are called open sets. A subset $C$ of $\mathcal{X}$ is called closed set if its compliment $C^c$ is in $\mathcal{T}$. Analogously to the previous definition one can define a topological space using only closed sets. If the set $\mathcal{X}$ contains finite number of elements then it is called finite topology. A set $\mathcal{X} = \{1, 2, 3\}$ with $\mathcal{T} = \{\{\emptyset\}, \{1, 2, 3\}\}$ is a trivial topology and $\mathcal{X}$ a topological space. The trivial topology is the smallest topology on the set $\mathcal{X}$. The largest topology on $\mathcal{X}$ is called *discrete topology*. For the set example above the discrete topology is $\mathcal{T} = \{\{\emptyset\}, \{1, 2, 3\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$. A detailed treatment of topological spaces can be found in [Munkres, 1993].

**Definition 4.6 (Abstract cell complex [Kovalevsky, 1989])** *An abstract cell complex $\mathcal{C} = (O, B, dim)$ is a set $O$ of abstract elements, with a binary relation $B \subset O \times O$ called bounding relation, which is antisymmetric, irreflexive and transitive; and with dimension mapping $dim : O \to \mathbb{I}$, from $O$ into a set of non-negative integers $\mathbb{I}$ such that $\dim(o) < dim(o')$ for all $(o, o') \in B$.*

If the dimension of cell $o$ is $d$ then $o$ is called d-dimensional cell or simply d-cell, and ACC is $n$ dimensional if one of the cells is $n$-cell. An example of the 2-dimensional ACC is shown in Figure 4.5, where 0-cells are vertices, 1-cells are edges, and 2-cells are faces. Formally,

$$O = V \cup E \cup F, \tag{4.2}$$

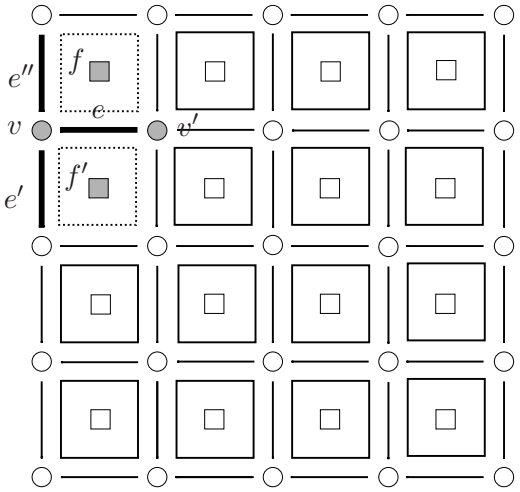



**Figure 4.5:** A 2-dimensional cellular model.

and

$$\dim(o) = \begin{cases} 0 & \text{if } o \in V, \\ 1 & \text{if } o \in E, \\ 2 & \text{if } o \in F. \end{cases} \tag{4.3}$$

A simple example of a 2D finite topological space is given in Figure 4.5; it consists of three types of elements: faces ($\square$), edges ($-$) and vertices ($\bigcirc$). As can be seen from the figure, an edge bounds two faces, say $f$ and $f'$. An edge $e$ is bounded by two vertices, say $v$ and $v'$; two faces $f$ and $f'$ are bounded by these two vertices as well. Let any subset $O$ of faces, edges and vertices be called *open set*, such that every element $o \in O$ of the surface bounded by $o$ is also in $O$. From this definition a face is not bounded by any element, therefore it is an open subset; an edge $e$ with the two faces $f$ and $f'$ that it bounds is an open subset; a vertex $v$ with all the edges ($e$, $e'$, $e''$) that it bounds and all the couple of faces ($f$, $f'$) bounded by these edges is an open set. A set of cells containing $o$ and all higher dimensional cells bounded by $o$ is called the open star of cell $o$ [Ahronovitz et al., 1995]. The open sets defined like this satisfy the axioms (T1), (T2) and (T3) and define a topological space [Kovalevsky, 1993, Ahronovitz et al., 1995].

Note that abstract cells should not be regarded only as Euclidean point set. For a square grid image, the pixels could be considered as square faces, inter-pixels boundaries as edges, and the intersection of two or more inter-pixel boundaries could be represented with vertices. Note that this is exactly the same representation for the 2D space as the dual image graphs [Kropatsch, 1994]. The cell model is more general and can be used to represent space element of higher dimension, whereas the dual graph representation for higher dimension is not defined yet. In 2D space a dual graphs representation is able to encode any subdivision of the 2D topological space. Encoding higher dimension with (dual) graphs is a difficult problem.

### 4.3.2 Dual Image Graph and Combinatorial Map

The usage of dual graph framework for 3 or higher dimension is cumbersome and not well defined. These problems are alleviated by using the combinatorial maps or generalized maps. N-

dimensional combinatorial maps [Lienhardt, 1989] may be seen as a graph with an embedding in an $N$-dimensional space i.e in the case of 2D [Brun and Kropatsch, 2001b], combinatorial maps are planar graphs encoding the orientation of edges around vertices. The base elements of a N-dimensional combinatorial map are the darts, also called half edges, which are connected together (sewed) by the orbits of 1 permutation and $N-1$ involutions. In the case of 2D [Brun and Kropatsch, 2001b] the permutation is called $\sigma$ and forms vertices, and the involution is called $\alpha$ and specifies edges. One of the advantages of combinatorial maps is that in the 2D case, unlike dual-graphs, they explicitly encode the orientation of the plane, correctly handling all the complicated cases with self-loops and parallel edges.

Like combinatorial maps, $n$-dimensional Generalized maps [Lienhardt, 1991] are defined in any dimension and correctly represent all topological configurations of the $n$-dimensional space (including 2D). Their base elements are darts and use only involutions to represent the connections between them, that describe cells in any dimension.

Combinatorial maps and generalized combinatorial maps define a general framework which allows to encode any subdivision on $n$D topological spaces orientable or non-orientable with or without boundaries. Using 2D images, combinatorial maps my be understood as a particular encoding of a planar graph, where each edge is split into two half-edges called darts. Since each edge connects two vertices, each dart belongs to only one vertex. A 2D combinatorial map is formally defined by the triplet $G = (\mathcal{D}, \sigma, \alpha)$ [Brun and Kropatsch, 2001b] where $\mathcal{D}$ represent the set of darts and $\sigma(d)$ is a permutation on $\mathcal{D}$ encountered when turning clockwise around each vertex. Finally $\alpha(d)$ is an involution on $\mathcal{D}$ which maps each of the two darts of one edge to the other one. Given a combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$, its dual is defined by $\overline{G} = (\mathcal{D}, \varphi, \alpha)$, with $\varphi = \sigma \circ \alpha$. The cycles of permutation $\varphi$ encode the faces of the combinatorial map. In what it follows, the cycles of $\alpha$, $\sigma(d)$ and $\varphi$ containing a dart $d$ will be respectively denoted by $\alpha^*(d)$, $\sigma^*(d)$ and $\varphi^*(d)$. An example of the combinatorial map is given in Figure 4.6a and (b).

Thus all graph definitions used in irregular graph pyramids [Kropatsch, 1994] are analogously defined with combinatorial maps [Brun and Kropatsch, 2003b].

### 4.3.3 Dual Graphs versus Combinatorial Maps

Advantages of combinatorial maps over dual graphs come form the embedding, that is inherently present at the former ones. Let us analyze the *'flower'* example given in Figure 4.6b, and (c) with respect to uniqueness of topological representation. The combinatorial map of this *'flower'* is shown and defined in Figure 4.6b by $G = (\mathcal{D}, \sigma, \alpha)$[7]. If the leafs of the 'flower' exchange position for e.g. leafs 1 and 3, a different $\sigma = (3, -3, 2, -2, 1, -1, 4, -4)$ will be defined, hence uniquely encoding the topology. The dual graphs are encoded by a pair of graphs, the (planar) primal graph vertices) and its dual. For each edge in the primal graph there is a corresponding one in the dual, that crosses it (Figure 4.6c). Since there is no ordering of the edges around the vertices, the dual graph representation does not uniquely encode the topology of the *'flower'*, as can be easily seen if we exchange the position, for e.g. of leafs 1 and 3, the dual graph describing this configuration is identical to the previous one (the one without exchanging the position of leafs.) The presence of similar cases happens very rare in 2D, thus the dual graph representation is used as the representation in the rest of discussion.

[7] $\sigma$ is encoded clockwise, shown with the arrow in Figure 4.6b.

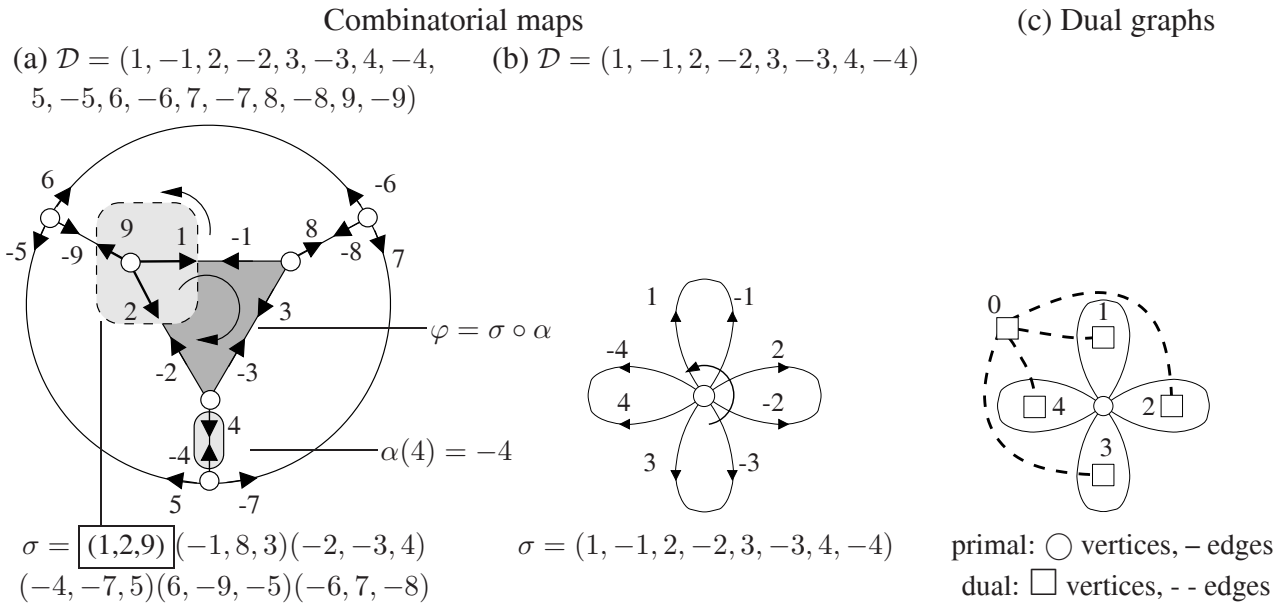



**Figure 4.6:** Combinatorial map and correctly handling topology.

## 4.4 Dual Graph Contraction

The irregular (dual graph) pyramids are constructed in bottom-up way such that subsequent level (say $k+1$) results by (dually) contracting precedent level (say $k$). In this section a short exposition of dual graph contraction algorithm is given, following the work of [Kropatsch, 1995a], and building the dual graph pyramid using this algorithm is presented in the next section. Dual graph contraction (DGC) [Kropatsch, 1994], [Kropatsch, 1995a] proceeds in two steps:

I. primal-edge contraction and removal of its dual, and

II. dual-edge contraction and removal of its primal.

In Figure 4.7 examples of these two steps are shown in three possible cases. Note that these two steps correspond in [Kropatsch, 1995a] to the steps (I) dual edge contraction, and (II) dual face contraction.

The base of the pyramid consists of the pair of dual image graphs $(G_0, \overline{G_0})$. In order to proceed with the dual graph contraction a set of so called contraction kernels (decimation parameters) must be defined. The formal definition is postponed until the Section 4.4.1. Let the set of contraction kernels be $\langle S_k, N_{k,k+1} \rangle$. This set consists of a subset of surviving vertices $S_k = V_{k+1} \subset V_k$, and a subset of non-surviving primal-edges $N_{k,k+1} \subset E_k$ (where index $k, k+1$ refer to contraction from level $k$ to $k+1$). Surviving vertices in $v \in S_k$ are vertices not to be touched by the contraction, i.e. after contraction these vertices make up the set $V_{k+1}$ of the graph $G_{k+1}$; and every non-surviving vertex $v \in V_k \backslash S_k$ must be paired to one surviving vertex in a unique way, by non-surviving primal-edges (Figure 4.8). In this figure, the shadowed vertex $s$ is the survivor and this vertex is connected with arrow edges ($n$s) with non-surviving

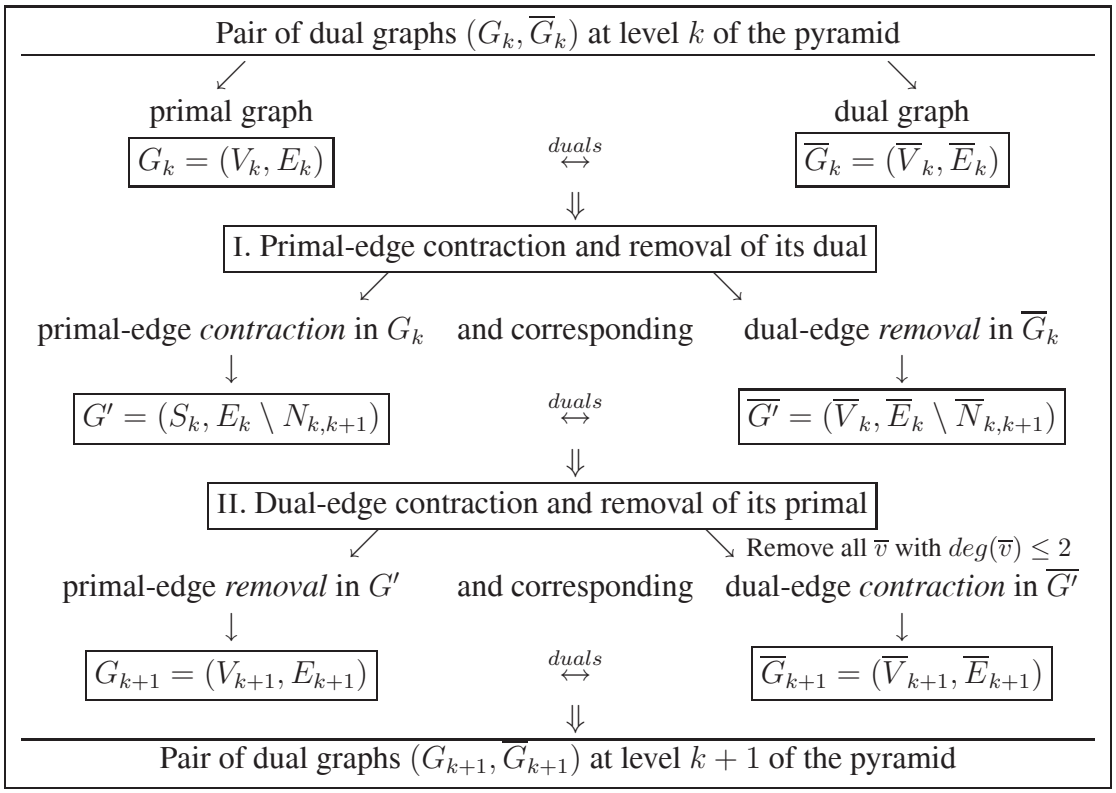



**Figure 4.7:** Dual graph contraction procedure (DGC).

vertices. Note that a contraction kernel is a tree of depth one, i.e. there is only one edge between a survivor and a non-survivor, or analogously one can say that the diameter of this tree is two.

The contraction of a non-surviving primal-edge consists in the identification of its endpoints (vertices) and the removal of both the contracted primal-edge and its dual edge (see Chapter 2, Section 2.6 for more details on these operations). Figure 4.9a shows the normal situation, Figure 4.9b the situation where the primal-edge contraction creates multiple edges, Figure 4.9c and self-loops. Redundancies (lower part) in case Figure 4.9c are decided through the corresponding dual graphs and removed by dual graph contraction. In Figure 4.9, the primal graph is shown with square (□) vertices and broken lines (- -) and its dual with circle vertices (❍) and full lines (-).

[Kropatsch, 1995a] shows that $\langle S_k, N_{k,k+1} \rangle$ determine the structure of an irregular pyramid. The relation between two pairs of dual graphs,$(G_k, \overline{G_k})$ and $(G_{k+1}, \overline{G_{k+1}})$, is established by dual graph contraction with the set of contraction kernels $\langle S_k, N_{k,k+1} \rangle$ as:

$$(G_{k+1}, \overline{G}_{k+1}) = C[(G_k, \overline{G}_k), \langle S_k, N_{k,k+1} \rangle]. \tag{4.4}$$

Dual-edge contraction and removal of its primal (second step) has a role of cleaning the primal graph by simplifying most of the multiple edges and self-loops[8], but not those enclosing

[8]Called also redundant edges.

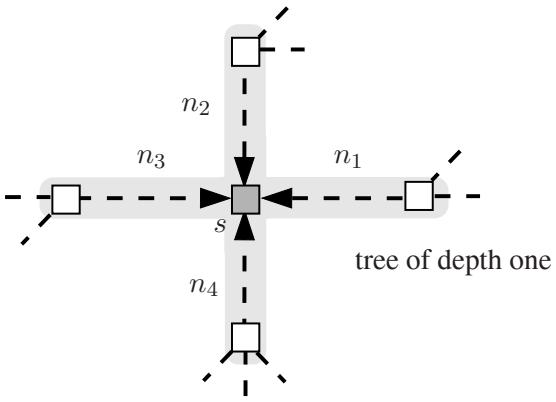



**Figure 4.8:** Contraction kernel with $s$ as a survivor and arrow edges ($n$-s) to be contracted.

any surviving parts of the graph. They are necessary to preserve correct structure [Kropatsch, 1995a]. So, the dual graph contraction reduces the number of vertices and edges of a pair of dual graphs, while preserving the topological relations among surviving parts of the graph. In [Kropatsch, 1994, Willersinn, 1995, Kropatsch, 1995b], a detailed presentation of dual graph contraction with some computational aspects (time and space complexity) are given. Moreover, in [Willersinn, 1995] different implementation details are presented.

### 4.4.1 Contraction Kernels

Let $S$ be the set of surviving vertices, and $N$ the set of non-surviving primal-edges. The connected components [9] $CC(s), s \in S$, of subgraph $(S, N)$ form a set of tree structures $T(s)$ that if contracted would collapse into vertex $s$ of the contracted graph. The number of this trees is $|S|$. The union of trees $T(s)$ contain the non-surviving primal-edges $N$. $T(s)$ is a spanning tree of the connected component $CC(s)$, or equivalently, $(V, N)$ is a spanning forest of graph $G = (V, E)$.

In order to decimate the graph $G = (V, E)$ a set of *surviving* vertices $S \subset V$ and a set of *non-surviving primal-edges* $N \subset E$ must be selected, such that the following conditions are satisfied:

- graph $(V, N)$ is a spanning forest of graph $G = (V, E)$, and
- the surviving vertices $s \in S \subset V$ are the roots of the forest $(V, N)$.

**Definition 4.7 (Contraction kernels)** *The set of disjoint rooted trees with length two of path going through the root is called a set of contraction kernels.*

Analogously, the trees $T(v)$ of the forest $(V, N)$ with root $v \in V$ are *contraction kernels*.

[9] Neglected level index refer to contraction from level $k$ to level $k + 1$.

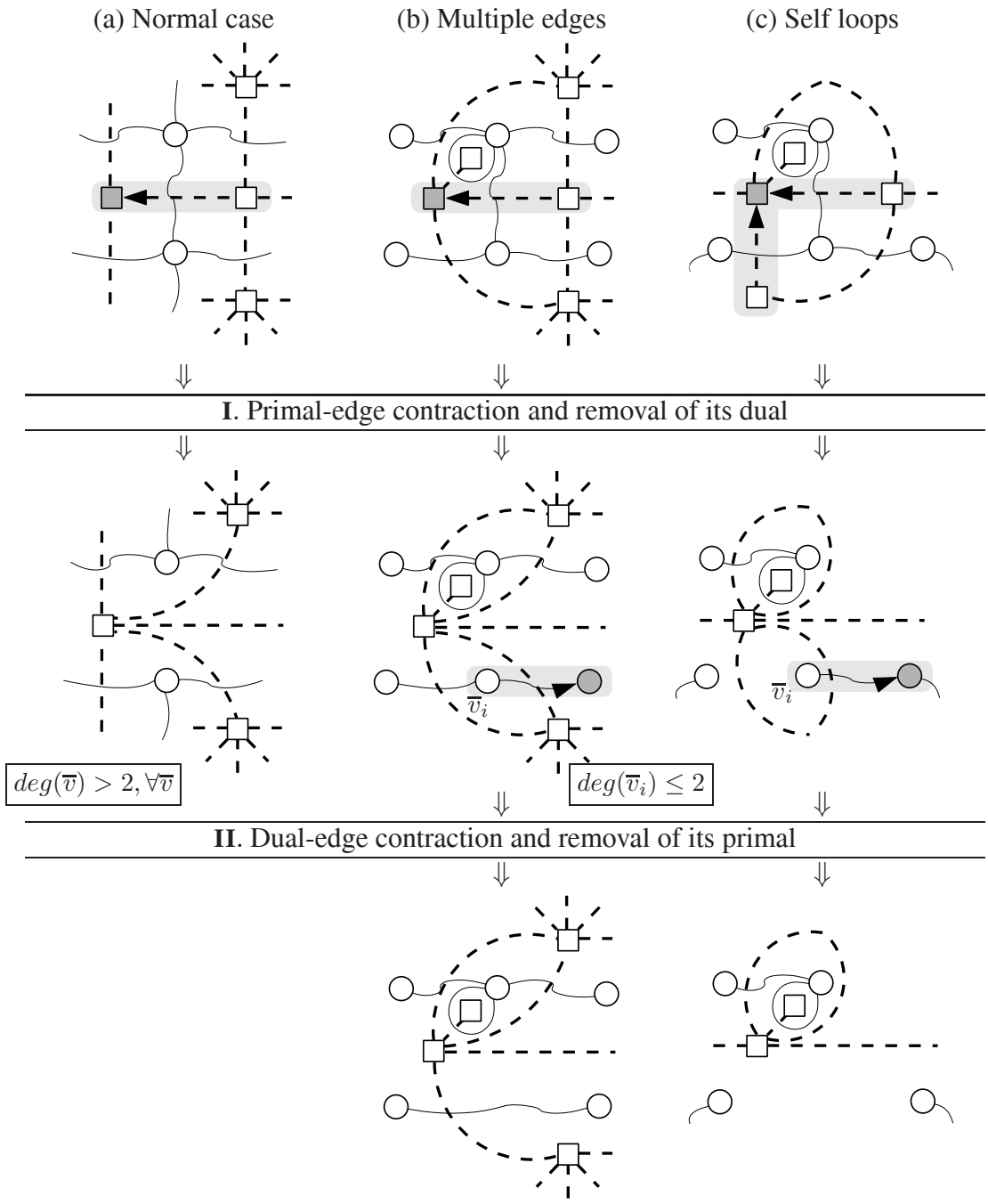



**Figure 4.9:** Dual graph contraction of a part of a graph.

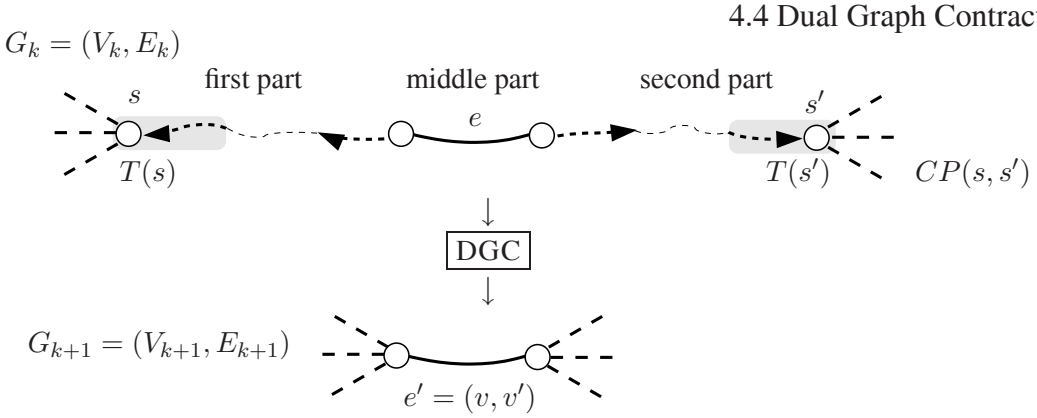



**Figure 4.10:** Connecting path $CP(v, v')$, $e$ is the bridge of this path.

After deploying the dual graph contraction algorithm on a graph one has to establish a path connecting between two surviving vertices on the resulted new graph. Let $G = (V, E)$ be a graph with decimation parameters $(S, N)$.

**Definition 4.8 (Connecting path [Kropatsch, 1994])** *A path in $G = (V, E)$ is called a connecting path between two surviving vertices $s, s' \in S$ if it consists of three subsets of edges:*

- *the first part is a possibly empty branch of contraction kernel $T(s)$.*
- *the middle part is an edge $e \in E \setminus N$ that bridges the gap between the two contraction kernels $T(s)$ and $T(s')$.*
- *the third part is a possibly empty branch of contraction kernel $T(s')$.*

See Figure 4.10 for explanation. Connecting path is denoted by $CP(s, s')$. Edge $e$ is called the *bridge* of the connecting path $CP(s, s')$. Each edge $e' = (v, v') \in E_{k+1}$ has a corresponding connecting path $CP_k(s, s')$, where $s, s' \in S \subset V_k$ are survivors in graph $G_k = (V_k, E_k)$. This means that two surviving vertices $s$, and $s'$, $s \neq s'$ that can be connected by a path[10] $CP_k(s, s')$ in $G_k$ are connected by an edge in $E_{k+1}$. If the graph $G_k$ is connected, than after the dual graph contraction the connectivity of the graph $G_{k+1}$ is preserved [Kropatsch, 1994].

Dual edge contraction can be implemented by (1) simply renaming all the non-surviving vertices to their surviving parent vertex (e.g. by using a find union set algorithm [Cormen et al., 2001]), (2) deleting all non-surviving edges $N$ and (3) their duals $\overline{N}$. The question on how to build contraction kernels is given in Chapter 5, where different methods are presented and their properties are analyzed.

### 4.4.2 Equivalent contraction kernels

[Burt and Adelson, 1983] combines two or more successive reductions in one equivalent weighting function in order to compute any level of any regular pyramid directly from the base level.

[10] By definition of connectivity of graph, there exist always a path between any two vertices of graph, see Chapter 2.

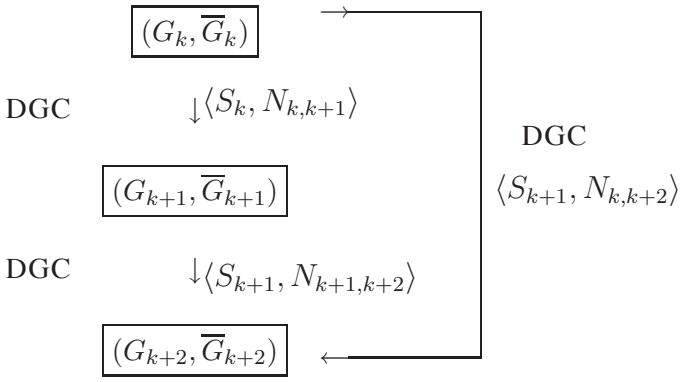



**Figure 4.11:** Equivalent contraction kernel.

Similarly, [Kropatsch, 1995b] combines two (or more) dual graph contractions (as shown in Figure 4.11) of graph $G_k = (V_k, E_k)$ with decimation parameters $\langle S_k, N_{k,k+1}\rangle$ and $\langle S_{k+1}, N_{k+1,k+2}\rangle$ into one single equivalent contraction kernel (ECK) $N_{k,k+2} = N_{k,k+1} \circ N_{k+1,k+2}$[11]:

$$\begin{aligned} C[C[G_k, \langle S_k, N_{k,k+1}\rangle], \langle S_{k+1}, N_{k+1,k+2}\rangle] &= C[G_k, \langle S_{k+1}, N_{k,k+2}\rangle] \\ &= G_{k+2} \end{aligned} \tag{4.5}$$

The structure of $G_{k+1}$ is determined by $G_k$ and the decimation parameters $\langle S_k, N_{k,k+1}\rangle$. Simple overlaying the two sets of contraction kernels, $\langle S_k, N_{k,k+1}\rangle$ (the one from level $k$ to $k+1$) and $\langle S_{k+1}, N_{k+1,k+2}\rangle$ (the one from level $k+1$ to $k+2$) will not yield a proper equivalent contraction kernel $\langle S_{k+1}, N_{k,k+2}\rangle$. The surviving vertices from $G_k$ to $G_{k+2}$ are $S_{k+1} = V_{k+2}$. The edges of the searched contraction kernels must be formed by edges $N_{k,k+2} \subset E_k$. An edge $e_{k+1} = (v_{k+1}, v'_{k+1}) \in N_{k+1,k+2}$ corresponds to a connecting path $CP_k(v_{k+1}, v'_{k+1})$ in $G_k$[12]. By Definition 4.8, $CP_k(v_{k+1}, v'_{k+1})$ consists of one branch of $T_k(v_{k+1})$, one branch of $T_k(v'_{k+1})$, and one surviving edge $e_k \in E_k$ connecting the two contraction kernels $T_k(v_{k+1})$, and $T_k(v'_{k+1})$.

**Definition 4.9 (Bridge [Kropatsch, 1994])** *Function bridge*: $E_{k+1} \mapsto E_k$ *assigns to each edge* $e_{k+1} = (v_{k+1}, w_{k+1}) \in E_{k+1}$ *one of the bridges* $e_k \in E_k$ *of the connecting paths* $CP_k(v_{k+1}, w_{k+1})$*:*

$$\textit{bridge}(e_{k+1}) = e_k. \tag{4.6}$$

Connecting two disjoint tree structures by a single edge results in a new tree structure. Now, $N_{k,k+2}$ can be defined as the result of connecting all contraction kernels $T_k$ by bridges as:

$$N_{k,k+2} = N_{k,k+1} \quad \cup \bigcup_{e_{k+1} \in N_{k+1,k+2}} \text{bridge}(e_{k+1}) \tag{4.7}$$

This definition satisfies the requirements of a contraction kernel [Kropatsch, 1994]. Analogously, the above process can be repeated for any pair of levels $k$ and $k'$ such that $k < k'$. If $k = 0$ and $k' = h$, where $h$ is the level index of the top of the pyramid, than with the resulted equivalent contraction kernel $(N_{0,h})$, the base level $(0)$ is contracted in one step into an apex $V_h = \{v_h\}$. ECKs are able to compute any level of the pyramid directly from the base.

[11] Only for $G_k$ is shown instead of $(G_k, \overline{G_k})$) for simplicity.

[12] If there are more than one connecting paths, one is selected.

### 4.4.3 Homotopy Preserving Transformations

As already discussed in Section 4.2, the plane graph partitions the plane into faces. Faces can have holes which are represented by connected components, and have to be connected by the fictive edges with the connected component which surrounds it (see the Figure 4.4c where the window is connected by a fictive edge with the front of the house). The notion of homotopy on (dual) graphs, i.e. for the 2D case, is derived from [Serra, 1982, page 187]. Let $\mathcal{G}$ be the set of all graphs.

**Definition 4.10 (Homotopy graphs)** *A mapping $\Psi$ from $\mathcal{G}$ into itself is said to be homotopic (or preserves the homotopy) if it transforms a graph $G$ into a graph $\Psi(G)$ such that there is a one-to-one and onto correspondence between connected components and the holes of $G$ and $\Psi(G)$.*

Two finite graphs $G$ and $G'$ are said to be homotopic when there exist a homotopic transformation $\Psi$ such that $G' = \Psi(G)$. [Marchadier et al., 2003] proved that the set of transformation that preserves the homotopy of (dual) graphs are:

- contraction of an edge, which is not a self loop,
- removal of a pendant edge, and
- contraction of redundant edges.

All these transformation do not change the number of connected components and holes of the dual graphs, therefore they preserve the graph homotopy. In the case of the 2D, the homotopy can be defined also using the homotopy tree (or adjacency tree) [Soille, 1994, page 56]. Two set are homotopic if their homotopy trees are identical.

### 4.4.4 Graph Minors with Dual Graph Contraction

Two containment relations between graphs: the subgraph relation and the induced subgraph relation are exposed in Chapter 2, Section 2.2. The graph contraction operation is described in details in this section and in Chapter 2, Section 2.6. In this section the minor relation is presented. Minor relation is closely related with the graph contraction operation, as will be shown.

Let $G' = (V', E')$ be a graph and $\{V_v \,|\, v \in V'\}$ a set partition of $V'$ into connected subset such that for any two vertices $v, v' \in V'$ there is an edge between two connected subsets $V_v$ and $V_{v'}$ in $G$ if and only if $(v, v') \in E'$. Then $G$ is called a $\mu G'$ and it is written[13] as $G = \mu G'$. The sets $V_v$ are the contraction kernels (called also branch sets [Diestel, 1997]) of $\mu G'$. One can think of obtaining $G'$ from $G$ by contracting every contraction kernel to a single vertex.

**Proposition 4.1 ([Diestel, 1997])** *$G$ is an $\mu G'$ if and only if $G$ is contractible to $G'$, i.e. if there exist graphs $G_0, G_1, ..., G_h$ and edges $e_i \in G_i$ such that $G_0 = G$, $G_h \simeq G'$ and $G_{i+1} = G_i/e_i$ for all $i < h$.*

Intuitively, $G$ is created by repeated contraction and deletion (or vice versa) of edges in the graph $G'$. If $G = \mu G'$ is a subgraph of another graph $G''$, then $G''$ is a *minor* of $G'$ and it is

[13] $\mu G'$ denotes a whole class of graphs and $G = \mu G'$ means that $G$ belongs to this class.

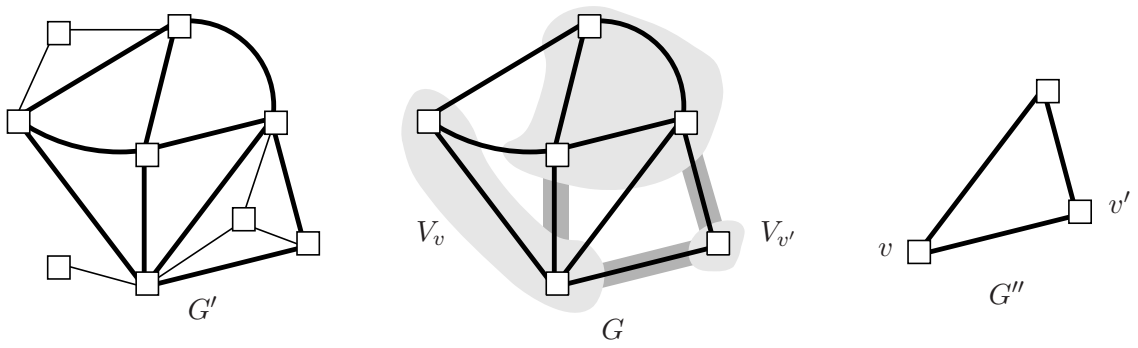



**Figure 4.12:** $G = \mu G'$ and $G \subseteq G'$, then $G''$ is minor of $G'$.

written as $G'' \preccurlyeq G'$. Every subgraph of a graph is its minor, moreover every graph is its own minor. In general any graph derived from another by repeated deletion and contraction (in any order) is its minor [Diestel, 1997]. The Figure 4.12 clarifies the minor relation.

**Proposition 4.2 (Partial orderings of minors [Diestel, 1997])** *The minor relation $\preccurlyeq$ is a partial orderings on the class of finite graphs, i.e. they are reflexive, antisymmetric and transitive.*

*Proof:* Can be found in [Diestel, 1997] □

Note that the minor relation in Proposition 4.1 is defined only for an edge contraction. Moreover this relation is defined for many edge contractions as well, because of the transitivity of the minor relation (Proposition 4.2).

In the previous section it is shown that dual graph contraction consists in repeated contraction and removal operations. The graph, say $G_{k+1}$ at level $k+1$ is obtained by contracting and removing edges in $G_k$ at level $k$, thus based on Proposition 4.1 and 4.2 graph $G_{k+1}$ is a minor of $G_k$, i.e. $G_{k+1} \preccurlyeq G_k$ for all $0 \leq k \leq h$, all higher levels $G_k$ are minors of the base graph $G_0$.

## 4.5 Dual Graph Pyramid

A graph pyramid is a pyramid where each level is a graph $G = (V, E)$ consisting of vertices $V$ and of edges $E$ relating two vertices. In order to correctly represent the embedding of the graph in the image plane [Glantz and Kropatsch, 2000b] additionally store the dual graph $\overline{G} = (\overline{V}, \overline{E})$ at each level.

In irregular pyramids, each level represents a partition of the pixel set into cells, i.e. connected subsets of pixels. The construction of an irregular image pyramid is iteratively local [Meer, 1989], [Bischof and Kropatsch, 1993], [Jolion, 2003], [Haxhimusa et al., 2002]:

- the cells have no information about their global position,
- the cells are connected only to (direct) neighbors, and
- the cells cannot distinguish the spatial positions of the neighbors.

This means that we use only local properties to build the hierarchy of the pyramid. Usually, on the base level (level $0$) of an irregular image pyramid the cells represent single pixels and the neighborhood of the cells is defined by the $4$ -connectivity of the pixels. A cell on level $k+1$ (parent) is a union of neighboring cells on level $k$ (children). As shown in Section 4.4 this union is controlled by contraction kernels (decimation parameters). Every parent computes its values independently of other cells on the same level. This implies that an image pyramid is built in $O[log(image\_diameter)]$ parallel steps. Neighborhoods on level $k+1$ are derived from neighborhoods on level $k$. Two cells $c_1$ and $c_2$ are neighbors if there exist pixels $p_1$ in $c_1$ and $p_2$ in $c_2$ such that $p_1$ and $p_2$ are $4$-neighbors.

The levels are represented as *dual pairs* $(G_k, \overline{G_k})$ of plane graphs $G_k$ and $\overline{G_k}$. See Section 4.3 for more details on this representation. The sequence $(G_k, \overline{G_k})$, $0 \leq k \leq h$ is called dual *graph pyramid*, where $0$ is the base level index and $h$ is the top level index, also called the height of the pyramid. Moreover the graph is attributed, $G = (V, E, attr_v, attr_e)$, where $attr_v : V \rightarrow \mathbb{R}^+$ and $attr_e : E \rightarrow \mathbb{R}^+$ are functions, i.e. content of the graph is stored in attributes attached to both vertices and edges. In general a graph pyramid can be generated bottom-up as follows:

**Algorithm 1 –** Constructing Dual Graph Pyramid

*Input*: Graphs $(G_0, \overline{G_0})$

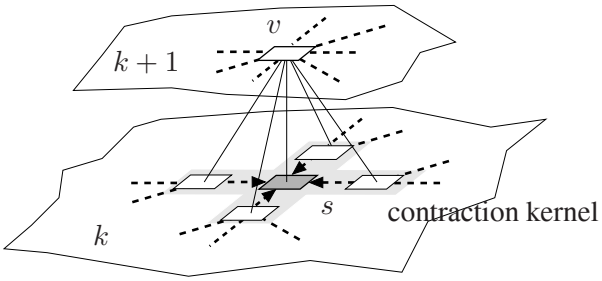
1: **while** further abstraction is possible **do**
2: select contraction kernels
3: perform dual graph contraction and simplification of dual graph
4: apply reduction functions to compute content of new reduced level
5: **end while**

*Output*: Graph pyramid – $(G_k, \overline{G_k})$, $0 \leq k \leq h$.

In the previous section it is shown that $G_{k+1}$ is minor of $G_k$, i.e. $G_{k+1} \preccurlyeq G_k$ for all $0 \leq k \leq h$, therefore graphs in a pyramid belong to a class of graphs related by a minor relation. One can say that a dual graph pyramid is a set of partial order graphs.

Let the building of the dual graph pyramid be explained by using the simple $5 \times 5$ gray value image in Figure 4.3. For the sake of simplicity of the presentation in the figures afterward the dual graphs as well as inter-level relations are not shown explicitly. An example of this inter-level relation is shown in Figure 4.13 with solid lines. In this figure a contraction kernel (shadowed) on level $k$ is shown, with its surviving vertex $s$ (dark shadowed) and its non-surviving vertices (white). After the dual graph contraction a new vertex $v$ on the level $k$ is created as well as the child-parent relations. In this example initially the attributes of the vertices receive the gray values of the pixels, and the cell in the new level (the parents) becomes the gray value of its children.

The first step determines what information in the current top level is important and what can be dropped. A contraction kernel is a (small) sub-tree of the top level the root of which is chosen to survive (black circles in Figure 4.14b. Figure 4.14a shows the window and the selected contraction kernels each shown with gray. Selection criteria in this case contract only



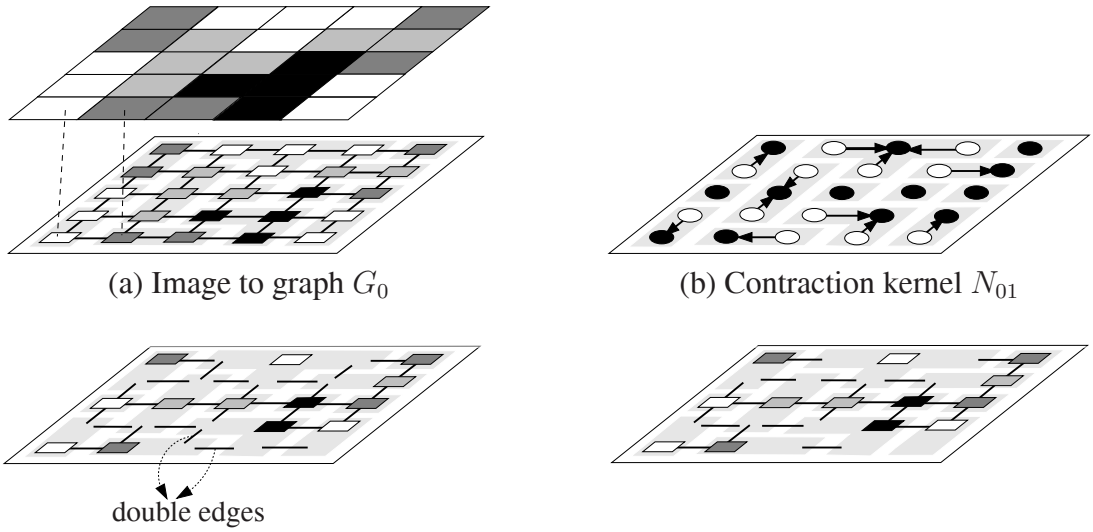

$s \in V_k$ and $v \in V_{k+1}$
Creation of a new vertex $v$ in the level $k+1$ of the pyramid.

**Figure 4.13:** Parent-child relation.

edges inside connected components having the same gray value.

All the edges of the contraction trees are dually contracted during step 3. Dual contraction of an edge $e$ (formally denoted by $G/\{e\}$) consists of contracting $e$ and removing the corresponding dual edge $\overline{e}$ from the dual graph (formally denoted by $\overline{G} \setminus \{\overline{e}\}$). This preserves duality and the dual graph needs not be constructed from the contracted primal graph $G'$ at the next level.

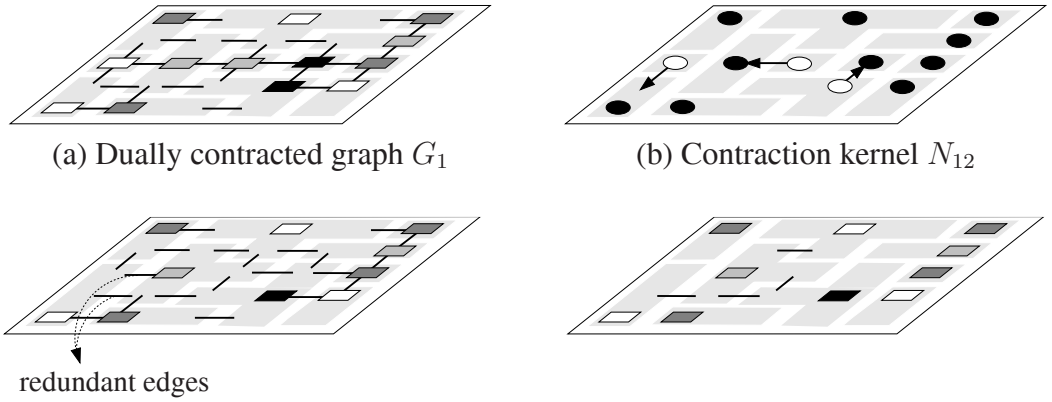
Since the contraction of an edge may yield multi-edges (an example shown with arrow in Figure 4.14c and self-loops there is a second simplification phase of step 3 which removes all redundant multi-edges and self-loops. Note that not all such edges can be removed without destroying the topology of the graph: if the cycle formed by the multi-edge or the self-loop surrounds another part of the data its removal would corrupt the connectivity! Fortunately this




(a) Image to graph $G_0$ (b) Contraction kernel $N_{01}$

(c) $G'$ after dual-edge contraction of $G_0$ (d) $G_1$ after the removal of redundant edges in $G'$

**Figure 4.14:** Dual graph contraction in $G_0$ and the creation of the $G_1$ of the pyramid.




(a) Dually contracted graph $G_1$ (b) Contraction kernel $N_{12}$

(c) Edge contraction of graph $G_1$ (d) Removal of the shown redundant edges

**Figure 4.15:** Dual graph contraction in $G_1$ and the resulting $G_2$.

can be decided locally by the dual graph since *faces of degree two* (having the double-edge as boundary) and *faces of degree one* (boundary = self-loop) cannot contain any connected elements in its interior. Since removal and contraction are dual operations, the removal of a self-loop or of one of the double edges can be done by contracting the corresponding dual edges in the dual graph, which are not depicted in our example for the sake of the simplicity. The dual contraction of our example remains a simple graph $G_1$ without self-loops and multi-edges (Figure 4.15a).

Step $3$ generates a reduced pair of dual graphs. Their contents is derived in step $4$ from the level below using the reduction function. In our example reduction is very simple: the surviving vertex inherits the color of its son.

The resulted graph $G_2$ of another dual contraction step is shown in Figure 4.16. The selection rules and the reduction function are the same as in the first iteration. The result shows that the regions with the same color are brought together. This fact could be used in a top-down verification step which checks the reliability of merging criterion in the more general context. Figure 4.17 depicts an overview of the results produced by the algorithm, by using a simple merging criterion, in which vertices having the same attributes (gray value) are merged. Moreover in this figure, contraction kernels and the equivalent contraction kernel are shown. By contracting the edges of the equivalent contraction kernel $N_{0,2}$ one can reach the top of the pyramid $G_2$ directly from the base $G_0$. A more complex merge criterion is shown in Chapter 6. The following Table 4.1 summarizes dual graph contraction in terms of the control parameters used for abstraction and the conditions to preserve topology (from [Kropatsch, 1994]).

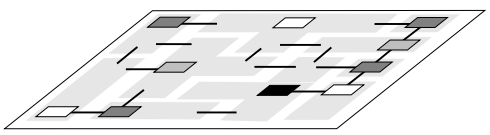


**Figure 4.16:** Graph $G_2$ after two steps of dual graph contraction.

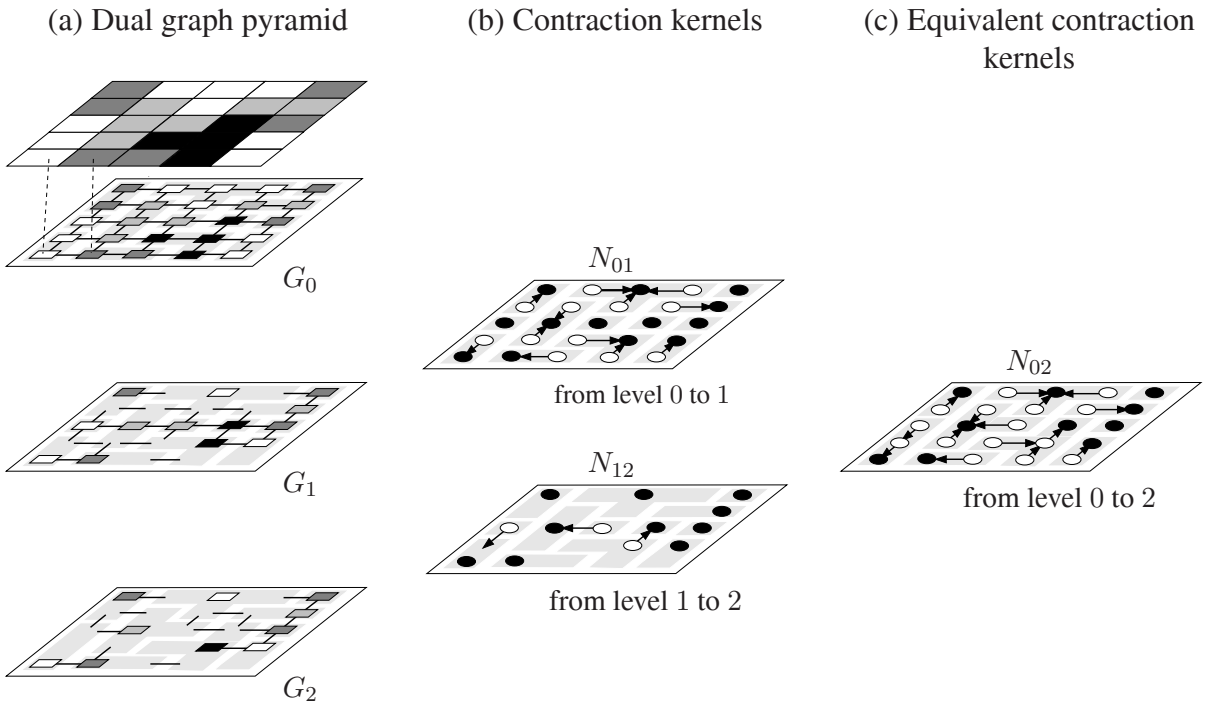



**Figure 4.17:** Dual graph pyramid with three levels: $G_0$, $G_1$, and $G_2$.

**Table 4.1:** Graph contraction parameters [Kropatsch, 1994]

| Level | representation | contract / remove | conditions |
|---|---|---|---|
| 0 | $(G_0, \overline{G}_0)$ | | |
| | $\downarrow$ | contraction kernel $N_{0,1}$ | forest, depth 1 |
| | $(G_0/N_{0,1}, \overline{G}_0 \setminus \overline{N}_{0,1})$ | | |
| | $\downarrow$ | redundant multi-edges, self-loops | $\deg \overline{v} \leq 2$ |
| 1 | $(G_1, \overline{G}_1)$ | | |
| | $\downarrow$ | contraction kernel $N_{1,2}$ | forest, depth 1 |
| | $\cdots$ | $\cdots$ | $\cdots$ |

In this section every level of the dual irregular pyramid is presented explicitly. An implicit representation can be done if vertices and edges of ECK of the apex $(V_0, N_{0,h})$ are labeled, such that vertices and edges are labeled with the highest level in which they survive. This labeled spanning tree uniquely defines the structure of the dual irregular pyramid [Kropatsch, 1995b]. If the plane graph is transformed into a combinatorial map the transcribed operations form the combinatorial pyramid [Brun and Kropatsch, 2001a, Brun and Kropatsch, 2001b]. This framework allows to link dual graph pyramids with topological maps (especially combinatorial maps) which enable to extend the scope to three (or more) dimensions.

## 4.6 Summary

In order to express the connectivity or other geometric or topological properties the image representation must be enhanced by a neighborhood relation. The neighborhood relation can be represented explicitly by a *graph* consisting of vertices corresponding to the sampling points and of edges connecting neighboring vertices. To represent not only neighborhood relations but also the inclusion relations, the planarity and the duality concepts come very good in hand. Since every planar graph has a dual, one can use dual graphs to represent the partitioning of the (image) plane, encoding the neighborhood and inclusion relations. One can define transformation of these graphs, like the dual graph contraction to simplify graphs in the sense of less vertices and edges. Edge contraction and removal introduces naturally a hierarchy of dual graphs, the dual graph pyramid.

CHAPTER 5

# Optimizing the Pyramid Structure

” Entia non sunt multiplicande praeter necessitatem. ” [1]

**by William of Occam.**

**Summary** We present two new methods to determine contraction kernels for the construction of graph pyramids MIES and MIDES. The first method is restricted to undirected graphs and yields a reduction factor of at least 2.0. This means with our method the number of vertices in the subgraph induced by any set of contractible edges reduces to half or less by a single parallel contraction. Our second method also works for directed graphs. Our methods yield better reduction factors than stochastic decimation algorithm, in all tests. The lower bound of the reduction factor becomes crucial with large images. We also give a method to compare the structure of the image pyramid.

**Keywords:** Graph based image analysis, image graph pyramids, logarithmic complexity maximal independent vertex set (MIS), maximal independent edge set (MIES), maximal independent directed edge set (MIDES), vertical path lengths.

## 5.1 Introduction

In a regular image pyramid (for an overview see [Kropatsch et al., 1999]) the number of pixels at any level $k$, is $\lambda$ times higher than the number of pixels at the next reduced level $k+1$. The so called reduction factor $\lambda$ is greater than one and it is the same for all levels $k$. If $s$ denotes the number of pixels in an image $I$, the number of new levels on top of $I$ amounts to $log_\lambda(s)$. Thus, the regular image pyramid may be an efficient structure to access image objects in a top-down process.

[1] Entities are not to be multiplied without necessity.

However, regular image pyramids are confined to globally defined sampling grids and lack shift invariance [Bister et al., 1990]. In [Montanvert et al., 1991], [Jolion and Montanvert, 1992] it was shown how these drawbacks can be avoided by irregular image pyramids, the so called adaptive pyramids. Irregular pyramids can perform most of the operations their regular counterparts are employed for [Rosenfeld, 1987]. Each level represents a partition of the pixel set into cells, i.e. connected subsets of pixels. The construction of an irregular image pyramid is iteratively local [Meer, 1989], [Bischof and Kropatsch, 1993], [Jolion, 2003]. Although adaptive pyramids overcome the drawbacks of their regular ancestors and although they grow to a reasonable height as long as the base is small, they grow higher than the base diameter with a larger input size because the progressive deviation from the regular base favors configurations that slow down the contraction process. As a consequence of the greater height the efficiency of pyramids degrades. [Montanvert et al., 1991],[Kropatsch, 1995] and [Bertin et al., 1996] encountered the problem that the height of irregular pyramid is not bounded. In this chapter we show that this problem can be resolved by a new selection mechanism which guarantees logarithmic heights.

Each level of the irregular pyramid consists of dual pair $(G_k, \overline{G_k})$ of plane graphs $G_k$ and $\overline{G_k}$ (see Chapter 4 for more details). The sequence $(G_k, \overline{G_k})$, $0 \leq k \leq h$ is called (dual) graph pyramid. The building of dual graph pyramid is presented previously in Chapter 4, Section 4.5, but without going into details on how to determine contraction kernels. Here we repeat this algorithm by expanding the second step with the iterative local methods presented in the rest of the chapter.

**Algorithm 2 –** Constructing Dual Graph Pyramid

*Input*: Graphs $(G_0, \overline{G_0})$

1: **while** further abstraction is possible **do**
2: select contraction kernels by an iterative local method
{*use one of the algorithms presented in Sections 5.2.2, 5.3.1, 5.4.1 or 5.5.1 to determine contraction kernels*}
3: perform dual graph contraction and simplification of dual graph (DGC)
4: apply reduction functions to compute content of new reduced level
5: **end while**

*Output*: Graph pyramid – $(G_k, \overline{G_k})$, $0 \leq k \leq h$.

**Definition 5.1 (Reduction factor)** *The reduction factor $\lambda$ is the ratio of the number of vertices of graphs $G_{k+1} = (V_{k+1}, E_{k+1})$ and $G_k = (V_k, E_k)$, $\forall k = 0, ..., h$ such that:*

$$|V_{k+1}| \leq \frac{|V_k|}{\lambda}. \tag{5.1}$$

The aim of this chapter is to combine the advantage of regular pyramids (logarithmic tapering) with the advantages of irregular pyramids (their purely local construction, universal segmentation, topology preservation, preservation of face degree etc.). The aim is reached by replacing the selection method for contraction kernels proposed in [Meer, 1989] by two new

iteratively local methods. The method in Section 5.3 that now guarantees a reduction factor of $2.0$ and the method in Section 5.4. Experiments with selection methods show that:

- the stochastic decimation method ([Meer, 1989]) does not lead to logarithmic tapering graph pyramids, as opposed to our methods, i.e. the reduction factors of graph pyramids build by the old method can get arbitrarily close to $1.0$ This problem was encountered also in [Montanvert et al., 1991].
- the sizes of the cells from the new method are much more uniform, i.e. the cell degree is not exponentially rising.

Not only stochastic decimation [Meer, 1989] but also connected component analysis [Kropatsch and Macho, 1995] gains from the new methods. The method in Section 5.4 turned out to produce logarithmic tapering graph pyramids also in case of monotonic dual graph contraction [Glantz and Kropatsch, 2000b].

Irregular pyramids can be used to enhance appearance based object recognition. Their structure gives the opportunity to apply top-down processing on data and introduce additional a priori knowledge similar to visual attention guidance. Thereby efficiency of succeeding algorithms can be improved considerably [Langs and Bischof, 2002, Langs et al., 2002, Langs et al., 2001].

The plan of the chapter is as follows. These hierarchies must be 'shallow' to be efficient in both bottom-up information aggregation and in top-down processes for verification and focus of attention. In Section 5.2 we recall the main selection algorithm used in the stochastic pyramid construction and demostrate in Section 5.6.1 graph pyramids from maximal independent vertex sets may have a very poor reduction factor. Moreover, experiments show that small reduction factors are likely, especially when the images are large. We propose two modifications. The one in Section 5.3 guarantees a reduction factor of $2.0$, but is applicable only if the edges may be contracted in both directions. The modification proposed in Section 5.4 also works in case of constraints on the directions. This modification yields the highest reduction factors in the case of stochastic graph pyramids, as our experiments in Section 5.6 confirms. In Section 5.5 we give a short introduction of an idea proposed by Jolion in [Jolion, 2003]. A detailed experimental comparison of all introduced methos is given in Section 5.6. In Section 5.7 we give a method to compare the structure of the graph pyramids build by different strategies.

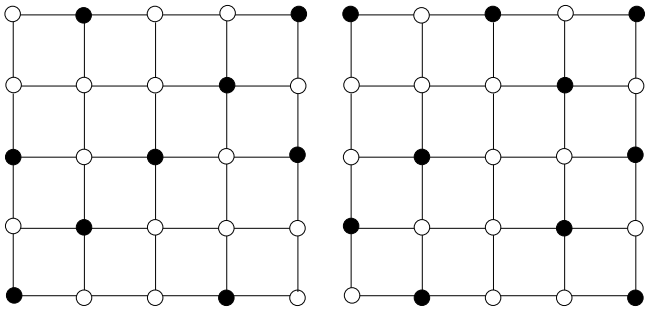
## 5.2 Maximal Independent Vertex Set (MIS)

In the following the iterated local construction of the (stochastic) irregular image pyramid in [Meer, 1989] is described in the language of graph pyramids. The main idea is to first calculate a so called *maximal independent vertex set* [Christofides, 1975], [Thulasiraman and Swamy, 1992]. Let the vertex set and edge set of $G$ be denoted by $V$ and $E$, respectively. The incidence relation of $V$, denoted by $\iota(\cdot)$ maps each edge from $E$ to its set of end vertices.

The neighborhood $\mathcal{N}(v)$ of a vertex $v \in V$ is defined by

$$\mathcal{N}(v) = \{\{v\} \cup \{w\} \in V \mid \exists e \in E \text{ such that } v, w \in \iota(e)\}. \tag{5.2}$$

**Definition 5.2 (Independent set)** *A set $I$ of vertices $V$ ($I \subseteq V$) is called independent if no two vertices $u, v \in I$ are neighbors, i.e. $u \notin \mathcal{N}(v)$ and $v \notin \mathcal{N}(u)$ .*



(a) $W$: with 9 vertices (b) $W'$: with 10 vertices

**Figure 5.1:** Maximal independent vertex set.

**Definition 5.3 (Maximal independent set)** *A subset $I_{max}$ of $V$ is called maximal independent vertex set if there is no independent set properly containing $I_{max}$.*

For each vertices $u, v$ (called members) of $I_{max}$ this means that:

1. $u \notin \mathcal{N}(v)$ for all $u, v \in I_{max}$,
2. for all $v \in V_k$ there exists a vertex $u \in I_{max}$ such that $v \in \mathcal{N}(u)$.

The $1^{st}$ condition requires that two surviving vertices cannot be in the neighborhood of each other (black vertices in Figure 5.2d). The $2^{nd}$ condition says that every non-surviving vertex has in its neighborhood a surviving vertex (white vertices in Figure 5.2d). An example of a maximal independent vertex set is shown with black vertices in Figure 5.2d, the arrows indicate a corresponding collection of contraction kernels. A maximal independent vertex set $W$ is not necessarily maximum, there may be another set $W'$ that contains more vertices than $W$ (Figure 5.1).

### 5.2.1 Related Works

In general finding *maximum* cardinality independent (vertex) set is an NP hard problem. The maximum independent set problem is among the first problems proved NP-hard, and can not be solved in polynomial time unless NP=P [Garey and Johnson, 1979]. Note that, a maximum independent set of a graph $G$ is equivalent to the maximum clique on the graph complement $G'$ [Skiena, 1990]. However, here we will not mention works done in this area.

In this chapter the algorithms presented solve the *maximal* independent vertex set[2], and NOT the maximum independent set. There is a $\mathcal{O}(|E|)$ time complexity sequential algorithm for solving the MIS, by just simple putting any vertex in the maximal independent set, and deleting its neighbors, repeat these steps until there are no vertices left in the graph. The most common technique for generation a maximal independent vertex set uses a greedy approach [Gavril, 1972]. This technique selects a set of vertices for the MIS with probability reciprocal to the degree of the vertex.

[2]Also called maximal stable set; we distinguish maximal from maximum independent set.

The maximal independent vertex set has a trivial sequential algorithm, but appears much harder to find a parallel version. There is a large body of literature on parallel algorithms finding MIS [Alon et al., 1986, Goldberg and Spencer, 1989, Luby, 1986, Karp and Wigderson, 1985]. The problem arises since there are interdependences between vertices to be added in the set. Because of adjacency relations not all of these vertices can be added at the same time. The symmetry-breaking techniques enable the algorithms to choose subset of independent operations. The MIS problem is very well suited for applying symmetry breaking techniques. A symmetry breaking technique is necessary to find large sets of vertices to add as it is done in [Goldberg et al., 1988, Karp and Wigderson, 1985, Luby, 1986]. [Karp and Wigderson, 1985] proved that MIS is in NC[3]. In [Luby, 1986] and [Alon et al., 1986] probabilistic algorithms are shown running in $\mathcal{O}(\log^2(|V|)$ time with a linear number of processor on a PRAM[4]. These algorithms have a local property: each vertex is randomly chosen to be put in the independent set based only on information about the adjacent vertices in the graph (the valency of the vertex). [Luby, 1986] showed also how to convert a probabilistic algorithm into a deterministic one. This technique preserves the running time but needs a larger number of processors $\mathcal{O}(|V|^2|E|)$. In [Goldberg and Spencer, 1989] a deterministic algorithm is presented that runs in $\mathcal{O}(\log^3(|V|)$ time with $\mathcal{O}((|V|+|E|)/\log|V|)$ number of processors. Problems like the maximal set packing problem or the matching problem are NC thought the reductions to maximal independent set problem [Karp and Wigderson, 1985].

A natural symmetry breaking technique is randomization. Since in our application of MIS, one cannot say whether the vertex with a large neighborhood is more important than the others, the degree of the vertex is not taken into consideration in the decision whether this vertex should be part of MIS or not. Therefore, in [Meer, 1989] a probabilistic symmetry breaking iterative parallel technique similar to that in [Luby, 1986], which does not explicitly take into account the degree of the vertices is shown. This technique is the base of the presentation in the rest of the chapter.

### 5.2.2 Algorithm MIS

Maximal independent vertex set (MIS) problem was solved using the heuristic in [Meer, 1989]. The number of iterations to complete MIS converges in most of the cases very fast, so called iterations for correction [Meer, 1989]. MIS [Meer, 1989], [Jolion, 2003] may be generated as in Algorithm 3.We assume that no two random numbers are equal.

The $2^{nd}$ step of the algorithm (the iteration for correction) is shown in more detail in a simple graph in Figure 5.2a, (b), and (c). The random numbers assigned to the vertices of graph are shown in Figure 5.2a. In Figure 5.2b is shown the parallel process of finding the local maximum vertices and marking them as *members* (depicted with black) and as *non-candidates*, also marking as *non-candidate* all the vertices that are neighbors of the *members*, vertices shown with white. Since after this step there are some *candidates* (5, 6, 15, 10 and 12) status of which is undefined, we repeat the $2^{nd}$ step in order to find a status for these *candidates*.

[3]A problem is NC (Nick Class) if for constants $k$ and $c$ the problem can be solved in time $\mathcal{O}((\log|V|)^c)$ using $\mathcal{O}(n^k)$ parallel processors. In other words a problem is solvable in poly-logarithmic time on a parallel computer needing polynomial number of processors. See [JáJá, 1992] for more details.

[4]Parallel random access machines, an abstract machine used to study parallel algorithms.

**Algorithm 3 – MIS Algorithm**

*Input*: graph $G = (V, E)$

1: mark every element of $V$ as *candidate*
2: **while** there are candidates **do**
3: assign random numbers to the candidates of $V$
4: determine the candidates whose random numbers are greater than the random numbers of all neighboring candidates and mark them as *member* (of the maximal independent set) and as *non-candidate*
5: mark every neighbor of every new member as *non-candidate*
6: **end while**
7: in each neighborhood of a vertex that is not a member there will now be a member. Let each non-member choose the edge to be contracted to its neighboring member, say the one with the maximal random number.

*Output*: set $\mathcal{C}$ of contraction kernels based on MIS.

The complete maximal independent set of vertices for this simple example is found in two iterations, Figure 5.2c.

**Definition 5.4 (Contraction Kernel)** *A contraction kernel is a vertex disjoint rooted tree. The depth of a kernel is the longest distance of a leaf from the root.*

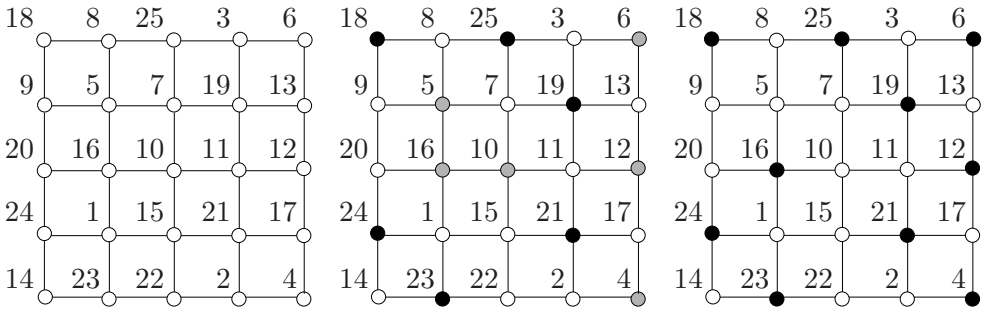
With this definition we can redefine the reduction factor as follows:

**Proposition 5.1** *The reduction factor can be determined from the sizes of vertices and the number of contraction kernels $|\mathcal{C}|$:*

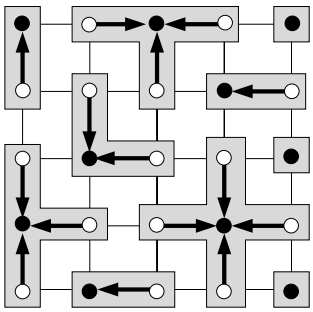
$$\lambda \geq \frac{|V_k|}{|\mathcal{C}|}. \tag{5.3}$$

*Proof:* The roots of each contraction kernel from $\mathcal{C}$ are the survivors in the next level of the pyramid. Hence the number of vertices in the next level is the same as the number of contraction kernels, i.e. $|V_{k+1}| = |\mathcal{C}|$. □

The assignment of the non-members to their members determine a collection of *contraction kernels* $\mathcal{C}$: each non-member is contracted toward its member and all contractions can be done in a single parallel step, $3^{rd}$ step of the algorithm. In Figure 5.2d the contractions are indicated by arrows. A graph pyramid from maximal independent vertex sets can be seen in Figure 5.2e, where $G_i$ are graphs in levels $i = 0, ..., 3$. Note that we remove parallel edges and self-loops that emerge from the contractions, if they are not needed to encode inclusion of regions by other regions. In the example of Figure 5.2e we do not need loops nor parallel edges. This can be done by the dual graph contraction algorithm [Kropatsch, 1995a] (See Chapter 4). Trivial contraction kernels do occur very often in MIS as can be seen in Figure 5.2d, isolated black vertices. In Section 5.3 we introduce a method that finds contraction kernels that are non trivial in this sense.

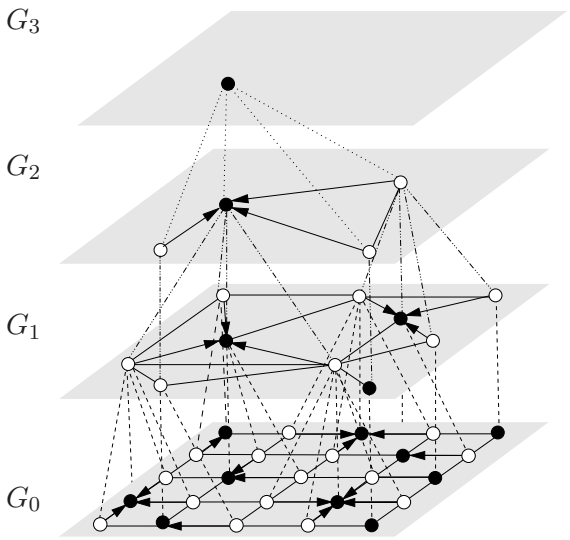



(a), (b), (c) Second step of MIS



(d) The black vertices form a MIS.

The frames indicate a corresponding collection of contraction kernels




(e) A graph pyramid from MIS.

**Figure 5.2:** A graph pyramid from maximal independent vertex sets.

## 5.3 Maximal Independent Edge Set (MIES)

In the following we aim at a collection $\mathcal{C}$ of contraction kernels in a plane graph $G$ such that

- each vertex of $G$ is contained in exactly one kernel of $\mathcal{C}$, and
- each kernel $\mathcal{C}$ contains at least two vertices.

**Definition 5.5 (Maximal Matching)** *A set $M$ of independent edges in a graph $G = (V, E)$ is a maximal matching if every vertex in $U \subset V$ incidents with an edge in $M$, and $M$ cannot be enlarged by an edge without loosing independence. The vertices in $U$ are then called matched vertices (by $M$); all other vertices are called unmatched vertices.*

Two edges are independent if they do not incident in the same vertex. We assume that $G$ is connected and planar, and that the data do not impose constraint on the selection, i.e. in large homogeneous regions. Clearly, the contraction of all kernels in $\mathcal{C}$ will reduce the number of vertices to half or less. Note that $M$ is only required to be maximal, i.e. the edge set $M$ cannot be enlarged by another edge from $\overline{G}$ without loosing independence. A maximal matching $M$ is not necessarily maximum: there may be a matching $M'$ (Figure 5.3b) that contains more edges than $M$ (Figure 5.3a).

### 5.3.1 Algorithm MIES

We start with independent *edge* sets or *matchings*, i.e. edge sets in which no pair of edges has a common end vertex. The *maximal independent edge set* (MIES), $\mathcal{C}$ is done in three steps (Algorithm 4).

**Algorithm 4 – MIES Algorithm**

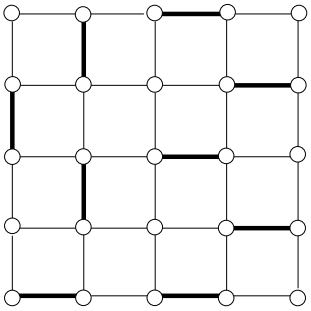
*Input*: graph $G = (V, E)$

1: find a maximal matching $M$ of edges from $G$ (e.g. using MIS algorithm on the edge graph of $G$)
2: $M$ is enlarged to a set $M^+$ that edge-induces a spanning subgraph of $G$
3: $M^+$ is reduced to a subset $\mathcal{C}$ such that each vertex of $G$ is contained in exactly one contraction kernel and each contraction kernel contains at least two vertices.

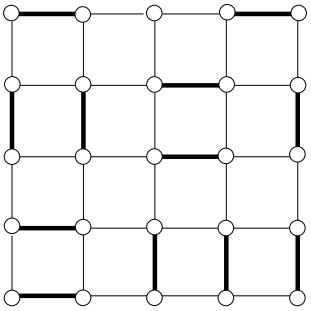
*Output*: set $\mathcal{C}$ of contraction kernels based on MIES

A maximal matching of $G$ is equivalent to a maximal independent vertex set on the edge graph of $G$ [Diestel, 1997], [Christofides, 1975]. Thus, a maximal matching may be determined by the iterated process as used in MIS algorithm (Section 5.2). Note that $M$ is only required to be maximal, i.e. the edge set $M$ cannot be enlarged by another edge from $E \setminus M$ without loosing independence.

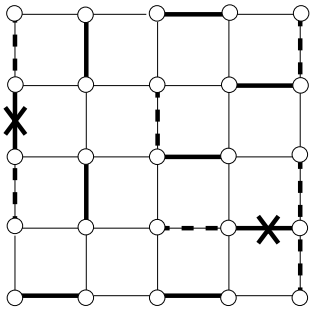
The collection of contraction kernels defined by a maximal matching $M$ may include kernels with a single vertex. Let $v$ denote such an isolated vertex, isolated from $M$, and choose a non-self-loop $e$ that has $v$ as an end vertex. Since $M$ is maximal, the other end vertex $w \neq v$ of $e$



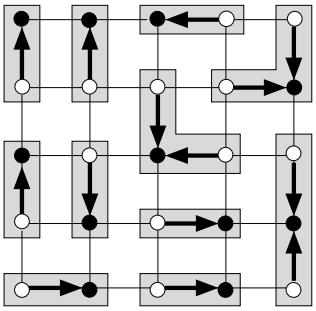
(a) $M$: a maximal matching with 9 edges



(b) $M'$: a matching with 12 edges



(c) $M^+$: the matching from a) enlarged by connecting formerly isolated vertices to the maximal matching

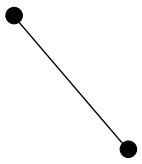


(d) (c) after breaking up trees of depth two into trees of depth one

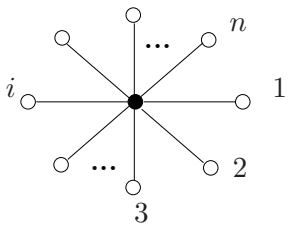
**Figure 5.3:** Maximal independent edge set.

belongs to an edge that is contained in the matching. Let $M^+$ denote the set of edges that are in $M$ or that are chosen to connect isolated vertices to $M$, $2^{nd}$ step of MIES.

The subgraph of $G$ that is induced by $M^+$ spans $G$ and its connected components are trees of depth one, two or three, Figure 5.3c. A tree of depth three can be separated into two trees of depth one each by removing the unique edge, both end vertices of which belong to other



(a) Without restriction




(b) Only the center vertex (depicted with black) can be chosen as the root of the tree

**Figure 5.4:** Restriction in choosing the surviving vertex.

edges of the tree (indicated by the crosses in Figure 5.3c); $3^{rd}$ step of MIES. Still, each vertex of $G$ belongs to a tree (of depth one). The arrows in Figure 5.3d indicate possible directions of contractions. Since each vertex of $G$ is now contained in a non-trivial contraction kernel, we prove the following proposition:

**Proposition 5.2 (Reduction factor of MIES at least** $2.0$**)** *The number of contraction kernels produced by algorithm* MIES *is at most* $|V|/2.0$.

*Proof:* Let $G = (V, E)$ be a planar connected graph with $|V|$ vertices. Let $M \subset E$ be a maximal matching of $G$ and let $U = \{i(e)|e \in M\}$ be the set of vertices matched by $M$ and $U' = V \setminus U$ be the set of unmatched vertices (isolated vertices). A vertex $v \in U$ is called matched by $M$ if it is incident with an edge in $M$. Since the matching is maximal, no two adjacent vertices $v, w$ may be unmatched. i.e. in the neighborhood of the unmatched vertex all vertices must be matched. This means that in the neighborhood of an unmatched vertex $w \in U'$ there is at least one *edge* connecting $w$ to a matched vertex $v \in U$. If there are more than one connecting edge, we select arbitrarily only one of them.

The edge set $M$ is enlarged to $M^+$ by adding all edges connecting unmatched vertices. Thus all vertices of $V$ are incident to edges in $M^+$. The subgraph of $G$ that is induced by $M^+$ spans $G$ and its connected components are trees of diameter one (the matched edge), two (isolated vertices connected to one endpoint of a matched edge only) or three (isolated vertices connected to both ends of the matched edge). Trees of diameter three are split into two trees of diameter one by removing the *unique central edge* $ue \in U$. The two endpoints of the matched edge of a tree of diameter three have a degree of at least two (they cannot be leaves). Splitting of the tree removes this matched edge which reduces the degree of both end vertices by one, but they are still greater than zero. Hence this edge deletion does not create a new isolated vertex. This ensures that all vertices $V_k$ are also incident edges of the new set $M^{++} = M^+ \setminus \{ue\}$ and there are no isolated vertices left.

We conclude that the subgraph of $G$ that is induced by $M^{++}$ spans $G$ and its connected components are trees of diameter one or two i.e. with more than one vertex. If all the components are trees of diameter one, the number of contraction kernels $\mathcal{C}$ is $|V|/2.0$. If there is at least one tree of diameter two then $|\mathcal{C}| < |V|/2.0$. We proved that in general case $|\mathcal{C}| \leq |V|/2.0$.

□

The surviving vertices can be in the neighborhood of each other, relaxing the $1^{st}$ condition of the MIS, which were proposed in [Kropatsch and Montanvert, 1991a]. Non-surviving vertices are in the neighborhood of at least one surviving vertex, fulfilling the $2^{nd}$ condition of MIS (Section 5.2). Note that in case of kernels with more than one edge, Figure 5.4b and Figure 5.5a, the roots (surviving vertex) within the kernel cannot be chosen arbitrarily. In Figure 5.4a any of the vertices can be chosen to be the root of the tree. But for the case in Figure 5.4b the black vertex can only be chosen as root of the tree, otherwise the (rooted) trees have depth greater than one. This would contradict our requirement about the depth of the contraction kernel. This is why the proposed method cannot be extended to applications in which there are a priori constraints on the directions of the contractions. However, the proposed method works for the stochastic case (no preconditions on edges to be contracted) and for connected component analysis [Kropatsch and Macho, 1995], where the attributes of the end vertices are required to be identical. In the case of Figure 5.5 the MIES algorithm chooses the only possible vertex $v$


(a) Input (b) $M$: maximal matching (c) $M^+$: enlarged $M$ (d) Possible contraction kernel

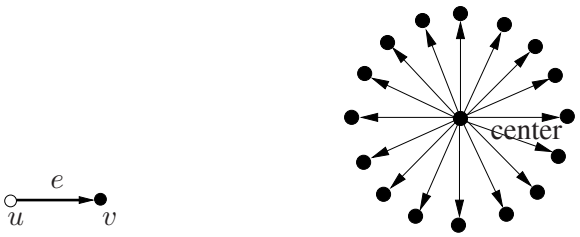
**Figure 5.5:** Restriction in choosing the surviving vertex.

for the root of tree (Figure 5.5d) i.e. as a survivor and there is only one possible contraction kernel, implying also the direction of contraction. For other cases the survivor can be chosen arbitrarily. The arrows in Figure 5.3d indicate possible directions of contractions, i.e. possible choice of the survivors. But some applications [Burge and Kropatsch, 1999], [Kropatsch and Burge, 1998], [Glantz et al., 1999], [Kammerer and Glantz, 2001], [Glantz and Kropatsch, 2000b] could restrict the way of choosing the surviving vertices, i.e. choosing vertex $u$ as a survivor, or could constrain the direction of contraction. This is why the proposed method cannot be extended to applications in which there are a priori constraints on the directions of the contractions. However, the proposed method works for the stochastic case (no preconditions on the direction of edges to be contracted) and for connected component analysis [Kropatsch and Macho, 1995], where the attributes of the end vertices are required to be identical.

## 5.4 Maximal Independent Directed Edge Set (MIDES)

In the previous section the direction of edge contraction is not of importance, i.e. edge $e = (u, v) = (v, u)$ can be contracted in arbitrary direction, from $u$ to $v$ or vice versa. In many graph pyramid applications such as line image analysis [Burge and Kropatsch, 1999], [Kropatsch and Burge, 1998] and the description of image structure [Glantz et al., 1999], [Kammerer and Glantz, 2001], [Glantz and Kropatsch, 2000b] a directed edge $e$ with source $u$ and target $v \neq u$ must be contracted (from $u$ to $v$), only if the attributes of $e$, $u$, and $v$ fulfill a certain condition, i.e. the edge $e = (u, v)$ is to be contracted but not $e' = (v, u) \neq (u, v)$. In particular, the condition depends on $u$ being the source and $v$ being the target, making the direction of contraction an important issue (Figure 5.6a). In line drawings, end point of lines or intersections must be preserved for geometric accuracy reasons. The edges that fulfill the condition are called *preselected* edges.

From now on, the plane graphs in the pyramid have (bi)directed edges. Typically, the edges in the base level of the pyramid form pairs of reverse edges, i.e. for each edge $e$ with source $u$ and target $v$ there exists an edge $e'$ with source $v$ and target $u$. Undirected graphs can be converted into directed graphs by substituting the edges into pairs of reverse edges, i.e. for each edge $e$ with source $u$ and target $v$ there exists an edge $e'$ with source $v$ and target $u$. However, the set of preselected edges may contain $e$ without containing $e'$. The goal is to build contraction kernels with a *high* reduction factor from the set of preselected edges. The reduction will always be determined according to the directed graph induced by the preselected edges. For example, if the number of vertices in the induced subgraph is reduced to half, the reduction factor will be $2.0$. From the example in Figure 5.6b it is clear that, in general, the reduction factor can be




(a) direction of contraction (b) reduction factor of a star

**Figure 5.6:** Direction of contraction.

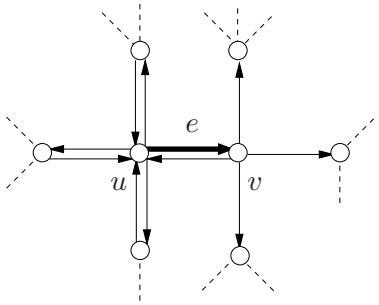



**Figure 5.7:** The neighborhood $\mathcal{N}(e)$.

arbitrarily close to $1.0$. To perform the contractions in parallel, we need a vertex disjoint union of contraction kernels. The plan is to define such a union in terms of independent directed edges, where *independent* means that no pair of directed edges belongs to the same neighborhood $\mathcal{N}(e)$ (see Definition 5.6). Then, dealing with edges instead of vertices, we may find the contraction kernels as in MIS.

**Definition 5.6** *Let $e = (u, v)$ be a directed edge of $G$ and $u \neq v$. Then the directed neighborhood $\mathcal{N}(e)$ is given by all directed edges with the same source $u$, targeting the source $u$ or emanating from target $v$:*

$$\begin{aligned} \mathcal{N}(e) &= \mathcal{N}((u,v)) \\ &= \{(u,v') \in E\} \cup \{(u',u) \in E\} \\ &\qquad\qquad \cup \{(v,u') \in E\}. \end{aligned}$$

The neighborhood $\mathcal{N}(e)$ of $e$ is given by all edges which point toward the source of $e$, edges with the same source $u$ as $e$ and the edges the source of which is the target of $e$. Note that edges pointing toward the target of $e$ are not part of the directed neighborhood. Figure 5.7 depicts $\mathcal{N}(e)$ in case of $u$ and $v$ both having $4$ neighbors.

**Definition 5.7** *A contraction kernel is a vertex disjoint rooted tree of depth one or zero (single vertex), each edge of which is directed toward the root (survivor).*

Note that the direction of edges uniquely determines which vertex survives on the next level of the pyramid, i.e determines the contraction kernel (decimation parameter [Kropatsch, 1995a]). A set $\mathcal{C}$ of directed edges forms such a collection of contraction kernels if and only if $\mathcal{C}$ contains none of the edge pairs depicted in Figure 5.8a. Seen from a directed edge $e$ with source $u$ and target $v \neq u$ that one wants to contract (from $u$ to $v$), no edge $e' \neq e$ with end vertex (source or target) equal to $u$ or source equal to $v$ may be contracted. An edge $e$ to be contracted together with those edges that one may not contract form a directed edge neighborhood $\mathcal{N}(e)$ of $e$.

**Definition 5.8** *Let $v$ be a vertex of directed graph $G$. We define the out-degree, the source $s$ as*

$$s(v) := \#\{e \in G \,|v \textit{ is source of } e\},$$

*and in-degree, the target $t$ as*

$$t(v) := \#\{e \in G \,|v \textit{ is target of } e\}.$$

In Figure 5.8b the number of edges with target in $root$ is $t(root) = 5$; and for the center vertex with $n$ edges pointing away $s(center) = n$, Figure 5.6b.

**Proposition 5.3** *Let $D$ denote a set of directed edges from a graph $G$ and $G_D$ denote the subgraph induced by $D$ with out-degrees $s_D(\cdot)$ and in-degrees $t_D(\cdot)$. Then the following statements are equivalent:*

(a) $s_D(v) < 2 \ \wedge \ s_D(v) \cdot t_D(v) = 0, \quad \forall v \in G_D$.

(b) *$G_D$ is a vertex disjoint union of contraction kernels.*

*Proof:*

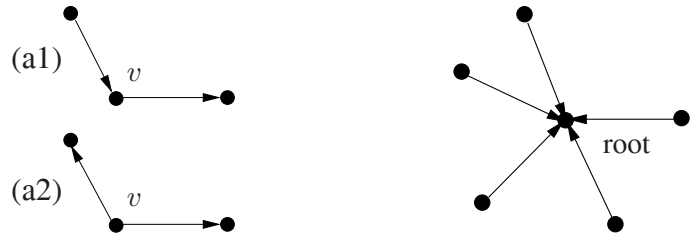
**i)** (a) $\Rightarrow$ (b): Let $R := \{r|\ r$ is target of some $e \in D \wedge s(r) = 0\}$ be the set of roots. Furthermore, each set $E_r := \{e \in D|\ r$ is target of e$\}$ is a tree with root $r \in R$, hence $E_r$ is a contraction kernel $\mathcal{C}_r$. Let us suppose that $E_r$ contains a cycle. Then it exists a vertex $u$ and two edges $e$ and $e'$ in $D$ such that $e = (u, r)$ and $e' = (u, r)$. But in this case $s_D(u) = 2$ contradicting $s_D(u) < 2$.

Let us suppose that it exists $u \in C_r \cap C_{r'}$, $r \neq r' \in R$. There are two cases:

1. if $s_D(u) = 0$ then either $u = r$ or $\exists e = (r, u) \in D$ but it contradicts $s_D(r) \cdot t_D(r) = 0$, and
2. $s_D(u) \neq 0$ then $t_D(u) = 0 \Rightarrow \exists (u, r) \in E_r$ and $(u, r') \in E_{r'}$ but it contradicts $s_D(u) < 2$.

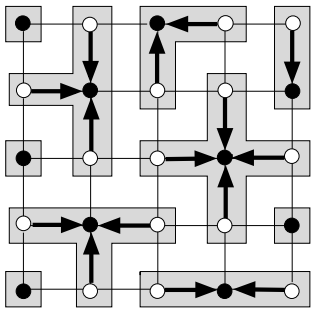
**ii)** (b) $\Rightarrow$ (a): Let $T$ be the set of roots of the vertex disjoint contraction kernels and let $C_t$ denote the unique contraction kernel with root $t \in T$. Furthermore, let $v \in G_D$. Since $C_t$ , $\forall t \in T$ are vertex disjoint, exactly one of the following holds:

1. $v \in T$ and $s_D(v) = 0$.
2. $v \notin T$ and $s_D(v) = 1, t_D(v) = 0$.




(a) Forbidden contraction kernels (b) Legal contraction kernel

**Figure 5.8:** Forbidden and legal configuration of directed edges.



MIDES with respect to $\mathcal{N}(e)$

**Figure 5.9:** Maximal independent edge set.

□

Examples of kernels which do not fulfill the Definition 5.7 are shown in Figure 5.8a1 where $s(v) = 1$ and $t(v) = 1$; and a2) where $s(v) = 0$ and $t(v) = 2$. From the example in Figure 5.6b it is clear that only one edge can be contracted (otherwise one ends with forbidden contraction kernels), which means in general, vertex reduction factor can get arbitrarily close to $1.0$.

Note that, in contrast to MIS, the roots of two contraction kernels may be neighbors. Condition (a) in Proposition 5.3 is fulfilled if and only if no pair of directed edges from $D$ belongs to the same $\mathcal{N}(e)$, $e \in D$, where $\mathcal{N}(e)$ is defined in Definition 5.6. Hence, a maximal vertex disjoint union of contraction kernels may be found via a maximal set of directed edges that are independent with respect to $\mathcal{N}(e)$. A parallel algorithm to find a maximal independent set with respect to $\mathcal{N}(e)$ is specified in the next section.

### 5.4.1 Algorithm MIDES

To find a *maximal independent set of directed edges* (MIDES) forming vertex disjoint rooted trees of depth zero or one, we proceed analogously to the generation of maximal independent vertex sets (MIS), as explained in the Section 5.2. Let $E$ denote the set of directed edges in the graph $G$ of the graph pyramid. We proceed as shown in Algorithm 5. Since the direction of edges uniquely determines the roots of the contraction kernels (the survivors), all the vertices

**Algorithm 5 – MIDES Algorithm**

*Input*: graph $G = (V, E)$

1: mark every directed edge of $E$ as *candidate*
2: **while** there are candidates **do**
3: assign random numbers to the candidates of $E$
4: determine the edge candidates $e$ whose random numbers are higher (larger) than the random numbers in $N(e) \setminus \{e\}$ and mark them as *member* (of a contraction kernel), also mark every $e' \in N(e)$ of every new member $e$ as *non-candidate*
5: **end while**
6: the targets of the directed edge candidates are marked as survivors, all the vertices which are the sources of directed edges are marked as non-survivor and the remaining unmarked vertices are marked to be survivors as well

*Output*: set $\mathcal{C}$ of contraction kernels based on MIDES

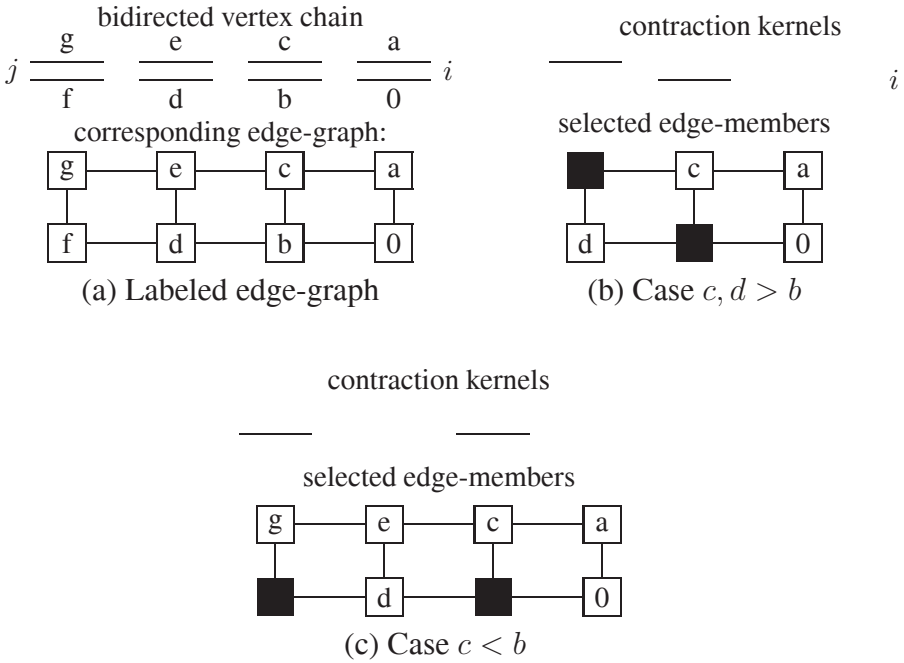



**Figure 5.10:** The edge-graph of a sequence of vertices (from [Kropatsch et al., 2005]).

which are the sources of directed edges are marked as non-survivor. Remaining isolated vertices are marked to be survivors. An example of a set of contraction kernels $C$ found by MIDES is given in Figure 5.9 (the survivors are depicted with black and non-survivors with white).

In order to show why MIDES method shows good reduction we follow the work in [Kropatsch et al., 2005]. Isolated vertices $i$ can occur after MIDES, too. Because all contraction kernels are disjoint and because each edge covers two vertices it is clear that following proposition holds:

**Proposition 5.4** *If there are no isolated vertices left after algorithm* MIDES *the number of contraction kernels is at most* $|V|/2$ *(allowing a reduction factor of at least* $2.0$*).*

In order to understand the difference to MIES let us study what happens in the while-loop after the first iteration. After local maxima are marked as members and all neighbors are removed as candidates, all remaining edge candidates are non-maxima.

The particular neighborhood $\mathcal{N}(e)$ used in the (bi-)directed graph creates the edge adjacencies shown in the edge-graph (Figure 5.10a). Each directed edge corresponds to a vertex ($\square$) in the edge-graph and the attributes (random numbers) are denoted by the letters $a, b, c, d, e, f$ and $g$. All edges in the neighborhood $\mathcal{N}(e)$ create edges in the edge-graph. Two edges pointing to the same vertex are NOT connected in the edge-graph.

Isolated vertices $i$ appear at the end of the correction phase when the neighborhood of edge-members cover all edges incident to $i$. Furthermore all edges incident to $i$ are edge-neighbors of a member whose orientation points away from $i$. Otherwise $\mathcal{N}(e)$ would allow the reverse edge to be a member. In Figure 5.10a edges $a, 0$ are incident to the isolated vertex $i$, both neighbors of edge $b, N(b) = \{0, a, c, d\}$. If $c$ would be the member instead of $b$, then 0 would become a member, too, since $0 \notin N(c) = \{e, d, b, a\}$ and $i$ would not be isolated.

From each isolated vertex $i$ emanate paths with monotonically increasing random numbers leading to a local maximum. We therefore study such a path which is a linear sequence of vertices (Figure 5.10a).

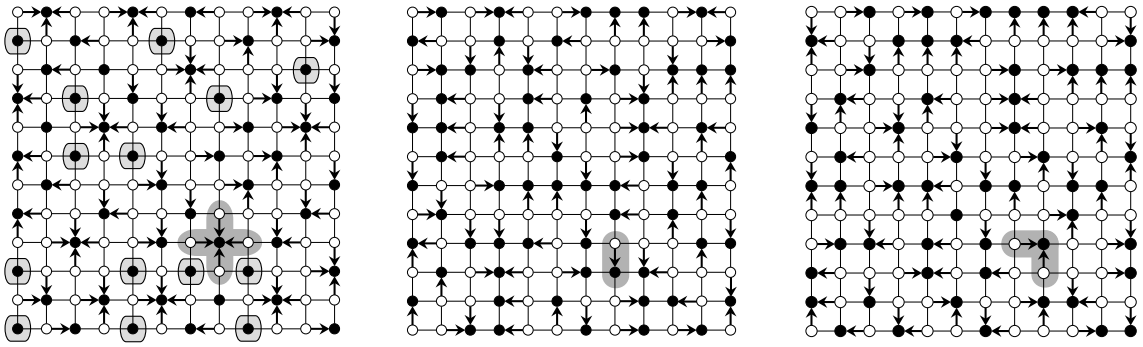
Assume $g$ is the only local maximum, all the other edges have at least one edge-neighbor which has a higher value. In this case $g$ would become a member and $e, f$ would be removed as candidates. The next highest edges could be $c$ or $d$. If $c$ is higher then it would be selected next and $a, b, d$ removed leaving 0 which would be the last edge selected. In this case no isolated vertex would appear. Assume $d$ is selected instead of $c$. Then $b, c$ would be removed as neighbors, $a$ would be selected next and 0 removed as neighbor of $a$. We see that many possibilities lead to solutions without an isolated vertex.

Let us reason backwards: In order for vertex $i$ to be an isolated vertex, both edges 0 and $a$ must be neighbors of a member. The only choice is $b$!

**Case** $c, d > b$**:** If both $c$ and $d$ are higher than $b$ then $e$ is the last member before $b$ was chosen. Since $b$ and $e$ share the same root vertex they are part of the same contraction kernel covering 3 vertices, thus compensating for the isolated vertex $i$.

**Case** $c < b$**:** If $c$ is less than $b$ then $f$ could be the last edge-member before $b$ (see Figure 5.10c). We note that the selected edges for contraction point in the same direction.

We can continue the construction until either the orientation of the selected edge changes or we reach the local maximum (vertex $j$ in Figure 5.10a) In the first case the two reversed edges form a contraction kernel covering three vertices and, hence compensating the isolated vertex. If the input graph has $5$ vertices then the vertex $j$ is isolated too and is not compensated, thus the reduction factor is less then $2$. This is the only example where the reduction factor is less then $2$, for input graph with $6$ or more vertices in a sequence, it is equal or larger than $2$. In the second case the search for a larger contraction kernel covering more than just two vertices would continue for all other paths emanating from the same isolated vertex. If the input graph has $3$ vertices in a sequence, the isolated vertex $i$ is not compensated, thus the reduction factor is less then $2$, only in this example. In general, the only case where the isolated vertex $i$ is not compensated by a larger contraction kernel consists of a set of edge-members filling the region of the correction phase and ALL pointing away from $i$ and toward the local maximum. As we



(a) MIS (b) MIES (c) MIDES

**Figure 5.11:** Examples of contraction kernels for a $12 \times 12$ grid graph.

will see in our experiments this is extremely unlikely and could, in addition, be compensated by any other larger contraction kernel.

MIDES has another feature different from MIES. It can create star-like kernels due to the special neighborhood chosen. Edges $(x_1, r), (x_2, r), \ldots$ having the same target $r$ can ALL be local maxima and selected as members in the first iteration of algorithm MIDES since $N((x_1, r)) \cap N((x_2, r)) = \emptyset$. Therefore the resulting set of edge-members is not necessarily a matching and may well contain larger kernels with one root and several edges attached to it.

We conjecture that larger kernels occur more frequently than isolated vertices. Each isolated vertex needs only one kernel of more edges to keep the balance for a reduction factor of two. This may explain the experimental results with MIDES.

In Figure 5.11 examples of contraction kernels for a grid graph of size $12 \times 12$ for MIS, MIES, and MIDES. Note that MIS algorithm has isolated surviving vertices (encircled isolated vertices in light gray), whereas MIES has none (as shown theoretically previously), and MIDES in this example has none as well. Even thought theoretically MIDES can have isolated vertices, it is unlikely as shown in this section. In this figure, the non-surviving vertices are shown with white, and together with arrowed edges compose the contraction kernels (some of which are shown by dark gray surroundings).

## 5.5 Data Driven Decimation Process (D3P)

In this section we recall an idea proposed in [Jolion, 2003, Jolion, 2001] on how to build stochastic irregular image pyramid, the *data driven decimation process* (D3P). The main difference from the MIS algorithm presented in Section 5.2 is that the $2^{nd}$ step of MIS is relaxed, i.e. no iteration will be performed. Skipping the iteration is motivated by the fact that the iteration is used only for completing the maximal independent set. It is assumed that being a local maximum is of importance [Jolion, 2003], [Jolion, 2001].

### 5.5.1 Data Driven Decimation Process Algorithm (D3P)

Let the vertex set of $G$ be denoted by $V$. To determine contraction kernels [Jolion, 2003], [Jolion, 2001] proceed as shown in Algorithm 6.

**Algorithm 6 – D3P Algorithm**

*Input*: graph $G = (V, E)$

1: mark every element of $V$ as *candidate*
2: assign random numbers to the candidates of $V$
3: determine the candidates whose random numbers are greater than the random numbers of all neighboring candidates and mark them as *member* (of the maximal independent set), also mark every neighbor of every new member as *non-candidate*
4: mark as *member* all the remaining candidates that do not have a *member* in the neighborhood
5: in each neighborhood of a vertex that is not a member there will now be a member. Let each non-member choose one of its neighboring members ( say the one with the maximal random number; we assume that no two random numbers are equal) and add the corresponding edge to the contraction kernel of the member

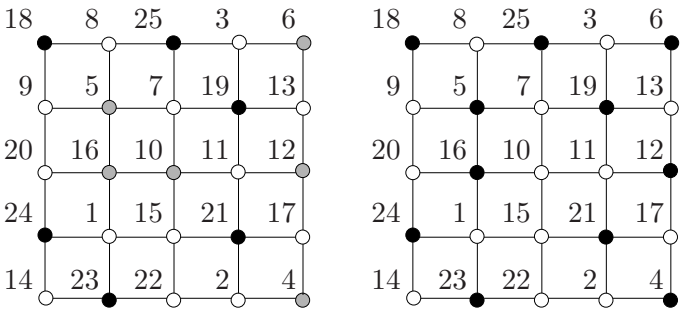
*Output*: set $\mathcal{C}$ of contraction kernels based on D3P.

Note that the output of the algorithm is not a maximal independent vertex set. We will use again the example shown in Figure 5.2 to explain the difference between MIS and D3P. All the *candidates* that have an undefined status in the $2^{rd}$ step of the MIS algorithm are simply marked as *members*. Note the differences shown in Figure 5.2c and Figure 5.12b. The *candidates* that have non-defined status ($6$, $5$, $16$, $10$, $12$, and $4$ ) after the step 3, are marked as *members* in step 4, shown with black in Figure 5.12b. In Figure 5.12c is given a set of contraction kernels build by the $3^{rd}$ step of the algorithm.

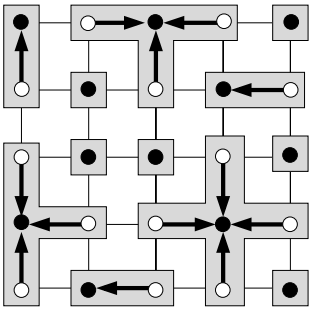
## 5.6 Comparing the Speed of Reduction

The experiment in this section compares the height of the pyramid (the solution quality), i.e. the reduction factors of vertices, edges and faces of stochastic graph pyramids using algorithms MIS, MIES, MIDES, and D3P. These experiments were done using guides in [Johnson, 2002], [McGeoch, 1992], [Hooker, 1995] to study the average-case behavior.

Uniformly distributed random numbers are assigned as attributes to the vertices or edges in the base level grid graphs. We do not change the structure of the base graph by these randomization. Then the pyramids are built using the stochastic strategy. Finally we calculate statistics of all pyramids built by each specific selection (MIS, MIES, MIDES, D3P) to compare the properties of different strategies. The same observation resulted from the two different random number generators [Matsumoto and Nishimura, 1998], [Mehlhorn and Näher, 1999]. By changing the seed of the uniformly distributed random generator we generated $1000$ attributed grid graphs, on top of which we built stochastic graph pyramids by each selection MIS, MIES, MIDES, and




(a) and (b) Third and fourth step of D3P



(c) Contraction kernels

**Figure 5.12:** Data driven decimation process.

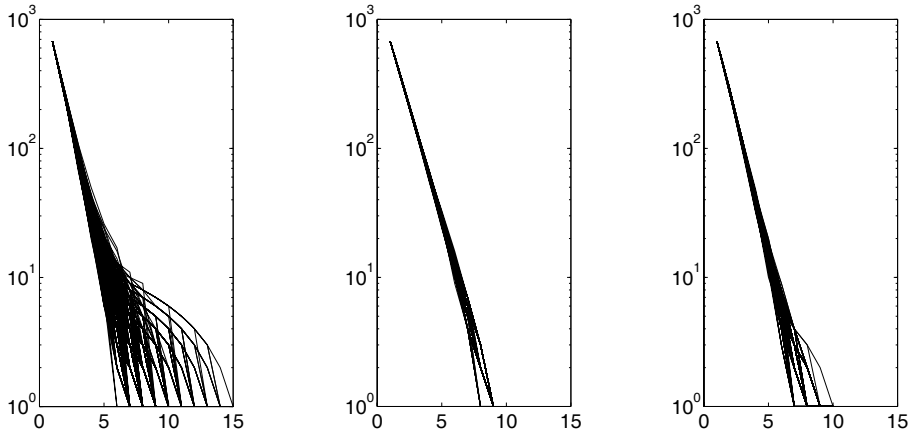
D3P. We reduce these graphs until we reach the top of the pyramid, the apex, which consist of one vertex and one face and no self loops.

In our experiments, Section 5.6.1, Section 5.6.2, Section 5.6.4, and Section 5.6.3 we used bi-directed grid graphs of size $10000$, $22500$ and $40000$ vertices respectively, which correspond to image sizes of $100 \times 100$, $150 \times 150$ and $200 \times 200$ pixels, respectively. Even thought the input grid graphs in which vertices have bounded degree are used as inputs, the results are general since the graphs in the second level are non-grid graphs with vertices of different degree. Therefore it is not necessary to test graphs with of large neighborhoods, i.e. large vertex degree.

The properties of different strategies are compared by using these following parameters:

- the height of the pyramid - *height*,
- the degree of vertices - *vertex degree*,
- the reduction factor for vertices - $\frac{|V_k|}{|V_{k+1}|}$,
- the reduction factor for edges $\frac{|E_k|}{|E_{k+1}|}$, and
- the number of iteration for correction [Meer, 1989] - *correction*,

The number of levels needed to reduce the graph at the base level (level $0$) to an apex (top of the pyramid) consisting of one vertex is the *height* of the pyramid. The number of edges incident to a vertex, i.e the number of non-survivors identified to the survivor represent the *vertex degree*



(a) MIS (b) MIES (c) MIDES

*L*egend: Number of vertices $v$ ($y$-axis) in levels ($x$-axis ) of the first 100 pyramids.

**Figure 5.13:** Reduction factor of graph pyramids for $26 \times 26$ input grid graphs.

complexity. The number of iterations to complete maximal independent set for any graph in the contraction process is the *iteration for correction*. The *ratio* of the number of vertices of two consecutive levels is the vertex reduction ratio ($|V_k|/|V_{k+1}|$). We average these parameters (sample mean $\widehat{\mu}_{pyr}$ and sample standard deviation $\widehat{\sigma}_{pyr}$) for every experiment (i.e pyramid) as shown in the Table 5.1 and 5.2 over all experimental sets ($\widehat{\mu}_{data}$, $\widehat{\sigma}_{data}$). Figures 5.15, 5.16, 5.17, and 5.18 summarize the results of the first $100$ of $1000$ tests for images of sizes $100 \times 100$, $150 \times 150$ and $200 \times 200$ pixels; solid lines connect the observed number of vertices/edges/faces of one particular pyramid, where the number of levels of the graph pyramid constitute the horizontal axis and the vertical axis shows the number of vertices $v$, edges $e$ and faces $f$ in logarithmic scale at the respective pyramid level. In this choice of coordinate axes a constant reduction factor becomes a straight line. Since we reduce the pyramid to a single vertex in the apex all the solid lines in all $v-$ and $f-$ diagrams end at the $10^0(=1)$. The solid lines of the $e$-diagrams end at the last but one level since the top does not contain any edge.

Statistical results using $1000$ grid graphs are discussed in Subsection 5.6.5, Tables 5.1, and 5.2 The statistics of vertex degree[5] is given in Table 5.1 and 5.2. These parameters are of importance because they are directly related to the memory costs of the representation of graph [Jolion, 2003, Jolion, 2001]. In Table 5.1 and 5.2 the mean and standard deviation for height of the pyramid, and for the number of iteration for corrections are given. The mean and standard deviation of the decimation ratio for vertices and edges is given in Table 5.1 and 5.2. The software tool $dgc\_tool$ [6] [Saib et al., 2002] was used to make these results.

### 5.6.1 Experiments with MIS

The graph size $26 \times 26$ is used in MIS of [Meer, 1989] to test the method. This graph size is too small to recognize the inefficiency of MIS with respect to the height of the pyramid. The

[5]The number of edges incident to a vertex.

[6]Can be found at ftp://ftp.prip.tuwien.ac.at/pub/dgc_tool/.

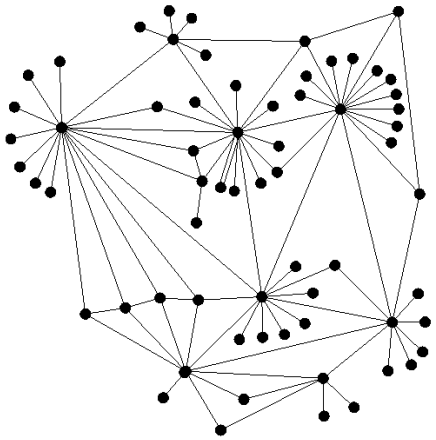


**Figure 5.14:** Evolution of vertices with large neighborhood using MIS.

reduction factor of different methods is shown in Figure 5.13. There is not a 'big' difference in the height of the pyramid between MIS and MIES (or MIDES, respectively) in this figure. The height of pyramid is not an issue when graphs are of small sizes and therefore is not studied in [Meer, 1989], but it is encountered as a major drawback of irregular graph pyramids in [Montanvert et al., 1991, Bischof, 1995].

The number of levels needed to reduce the graph at the base level (level $0$) to a graph consisting of a single vertex, with no self loops and a single face (top of the pyramid) are given in Figure 5.15, ($v$) for vertices, ($e$) for edges and ($f$) for faces. From Figure 5.15 we see that the height of the pyramid cannot be guaranteed to be logarithmic, except for some good cases. In the worst case the pyramid had $22$ levels for $100 \times 100$ vertices and $41$ levels for the graph with $200 \times 200$ vertices, respectively. Poor reduction factors are likely, as can be seen in Figure 5.15, especially when the images are large. This is due to the evolution of larger and larger variations between the vertex degrees in the contracted graphs (Table 5.2, $\widehat{\mu}_{data}(\max) = 70.69$ and Figure 5.14). The absolute maximum in-degree was $148$. The a priori probability of a vertex being the local maximum depends on the size of its neighborhood. The larger the neighborhood the smaller is the probability that a vertex will survive [Jolion, 2003]. MIS tends to have vertices with a large neighborhood (stars), as shown exemplary in Figure 5.14. This causes the reduction factor to be very poor at highest levels, also noted in [Montanvert et al., 1991], which is mainly due to the good performance of the decimation at the first few levels where the graphs have large sizes and a small neighborhood size (e.g. each vertex has $4$ neighbors in the base level). The mean reduction factor of vertices is $1.94$ (Table 5.2).The collapse of the hight-order-star-like configuration into the apex causes the sudden break of the solid lines ('star-contraction') in the $v$ and $f$ -diagrams. The number of iterations necessary to complete the maximum independent set per level (iterations for correction) are $3$ as reported by [Meer, 1989]. The mean value of the height of the pyramid is $20.78$ (Table 5.2).

To summarize, a constant reduction factor higher then $1.0$ cannot be guaranteed and bad cases have a high probability (almost horizontal solid lines in Figure 5.15).

### 5.6.2 Experiments with MIES

The number of levels needed to reduce the graph at the base level by the MIES strategy to an apex are shown in Figure 5.16. The experiments show that the reduction factor, even in the worst case, is always higher than the theoretical bound $2.0$, as indicated by the dashed line in Figure 5.16$v$). The MIES is more stable than MIS, as can be seen in Figure 5.16, where the slope of the solid lines have smaller variations. The mean and the variance of reduction factor for MIES is smaller than in case of MIS or D3P, which implies that the height of the pyramid ($\widehat{\mu}_{data}(\text{height}) = 14.01$) is also smaller than that for MIS ( $\widehat{\mu}_{data}(\text{height}) = 20.78$) or D3P ($\widehat{\mu}_{data}(\text{height}) = 88.25$).

The mean numbers of iteration for correction per level was higher for MIES (Table 5.1 and 5.2). To summarize the reduction factor was always below the theoretical upper bound of $2.0$.

### 5.6.3 Experiments with MIDES

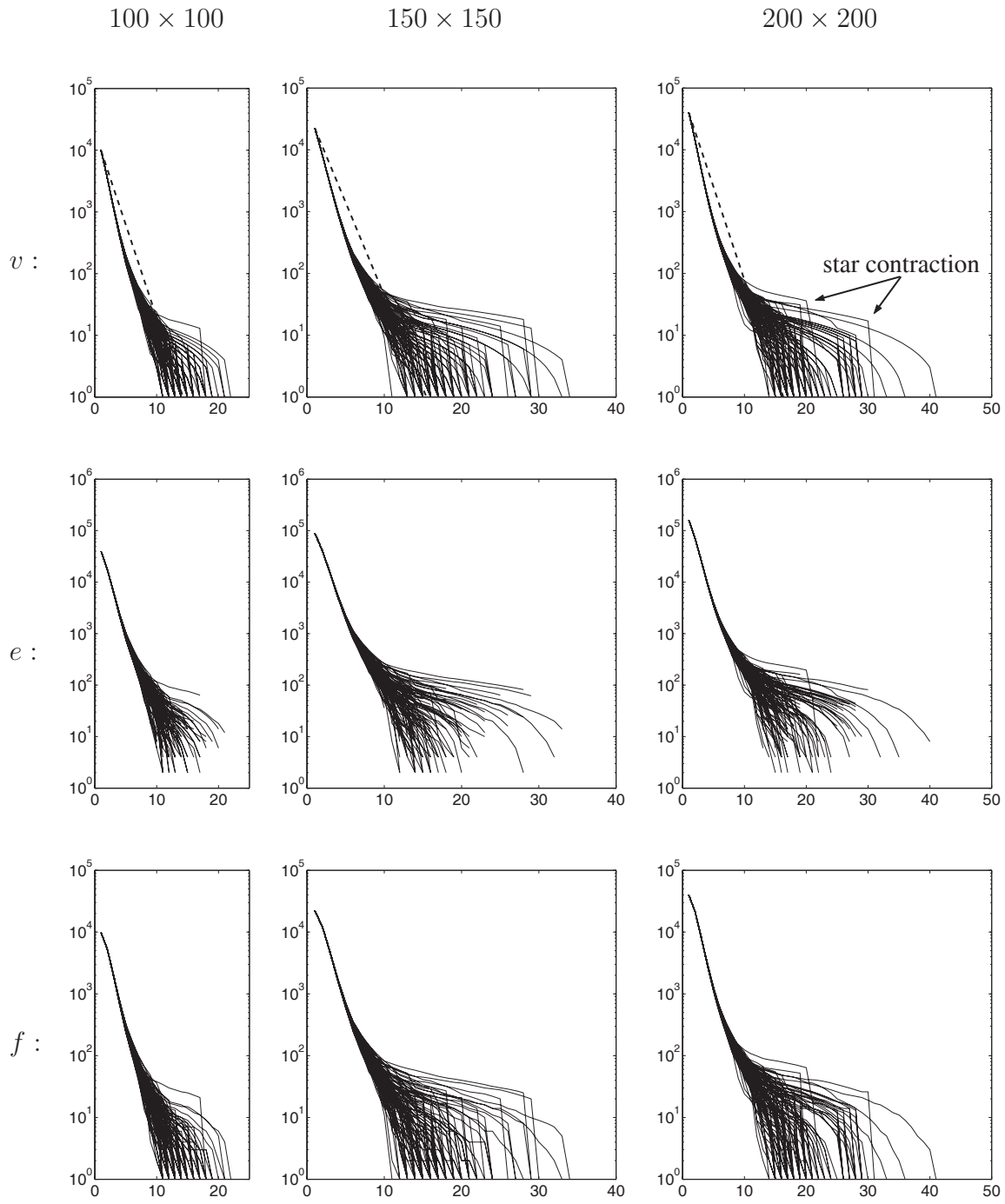
The Figure 5.17 depicts the number of levels required to get on top of the pyramid. We see that the reduction factor is higher than $2.0$ (dashed line) even in the worst case. Also the maximum indegree of the vertices is much smaller ($\widehat{\mu}_{data}(\max) = 13.29$) than for MIS ($\widehat{\mu}_{data}(\max) = 70.69$, Table 5.2). For MIES and MIDES we have not encountered nodes with large neighborhood as for MIS. For the case of the graph with size $200 \times 200$ vertices, MIDES needed $13$ levels in comparison to $15$ levels in the worst case of MIES. The number of iterations needed to complete the maximum independent set was comparable with the one of MIS (Table 5.1 and 5.2). The MIDES algorithm shows a better reduction factor than MIES, as can be seen in Figure 5.17 and Table 5.2 ($\widehat{\mu}_{data}(\widehat{\mu}_{pyr}(\frac{|V_k|}{|V_{k+1}|})) = 2.63$).

To summarize the reduction factor was always better than the theoretical upper bound of $2.0$ in all our tests. Bounded reduction factor cannot be guaranteed.

### 5.6.4 Experiments with D3P

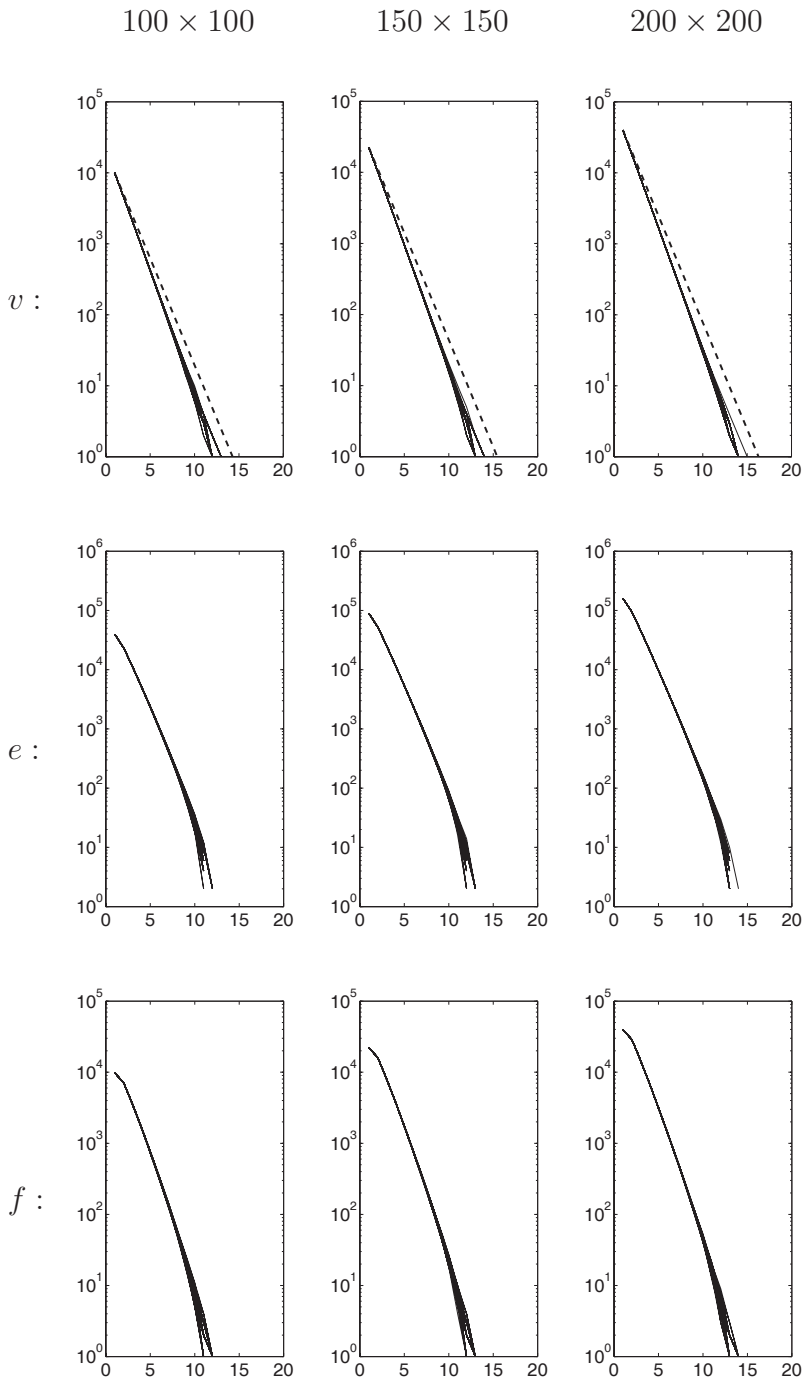
The Figure 5.18 gives the reduction factors ($v$) for vertices, ($e$) for edges and ($f$) for faces. The experiments show that poor reduction factors are likely, because of the large degree of vertices ($\widehat{\mu}_{data}(\max) = 433.92$, Table 5.2). Also the height of the pyramid is related to the complexity of the vertices, which is why the D3P gives the highest pyramids ($\widehat{\mu}_{data}(\text{height}) = 88.25$). The number of iterations for correction is $1$ because we do not iterate at all. As expected we have large neighborhoods for D3P. The high variation of the height of pyramid produced by D3P shows that this method fit to the distribution of values associated with the vertices [Jolion, 2003].

To summarize, a constant reduction factor, like for the MIS, cannot be guaranteed and bad cases have a high probability, as can be seen in Figure 5.18. Note that this method is developed having in mind importance of the data itself, therefore the results of only the structural simplification taken into account imply that the method should be used always in conjunction with the data.



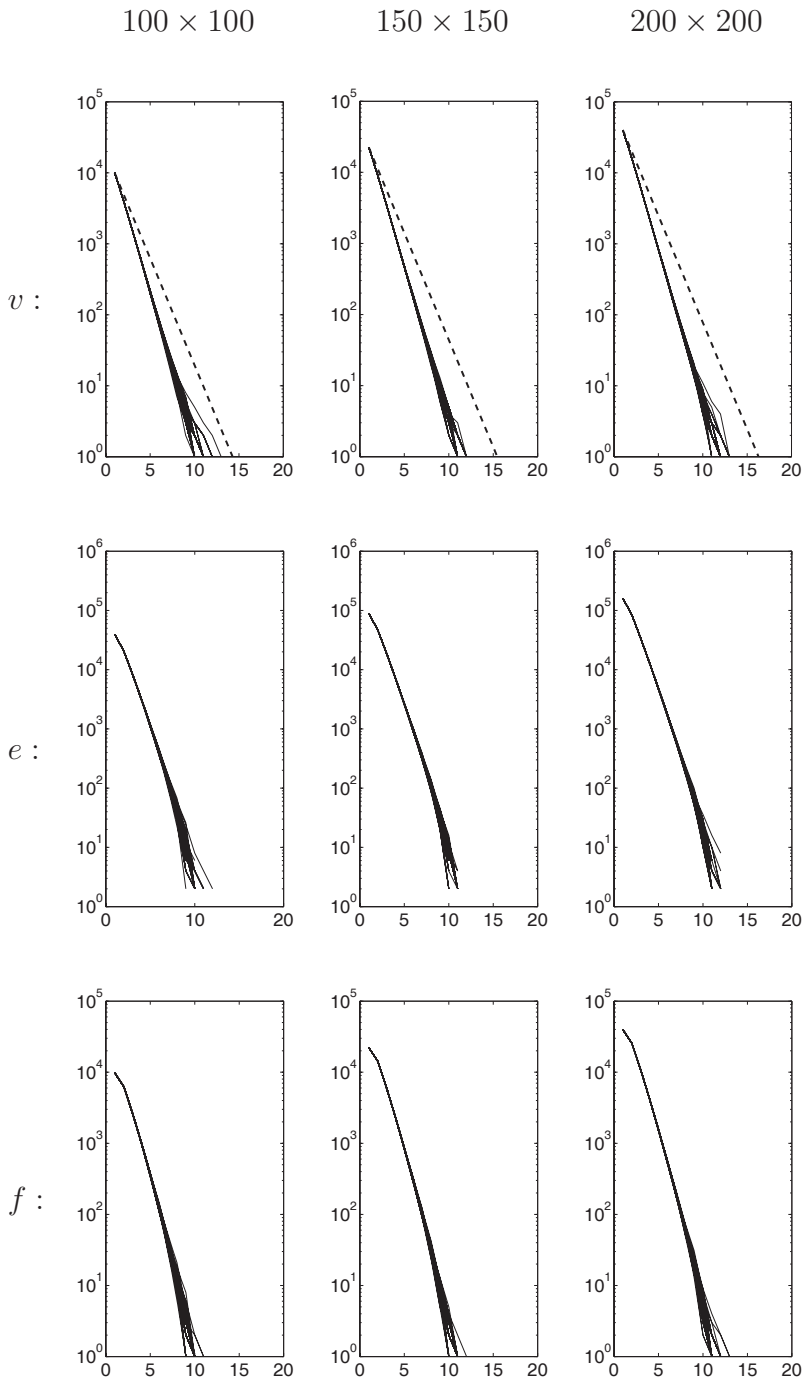

*L*egend: Number of vertices $v$, edges $e$, and faces $f$ ($y$-axis) in levels ($x$-axis ) of the first 100 MIS pyramids. The base levels are rectangular grid containing $100 \times 100$, $150 \times 150$ and $200 \times 200$ vertices. Dashed lines represent the theoretical reduction factor of 2.0.

**Figure 5.15:** MIS Algorithm.



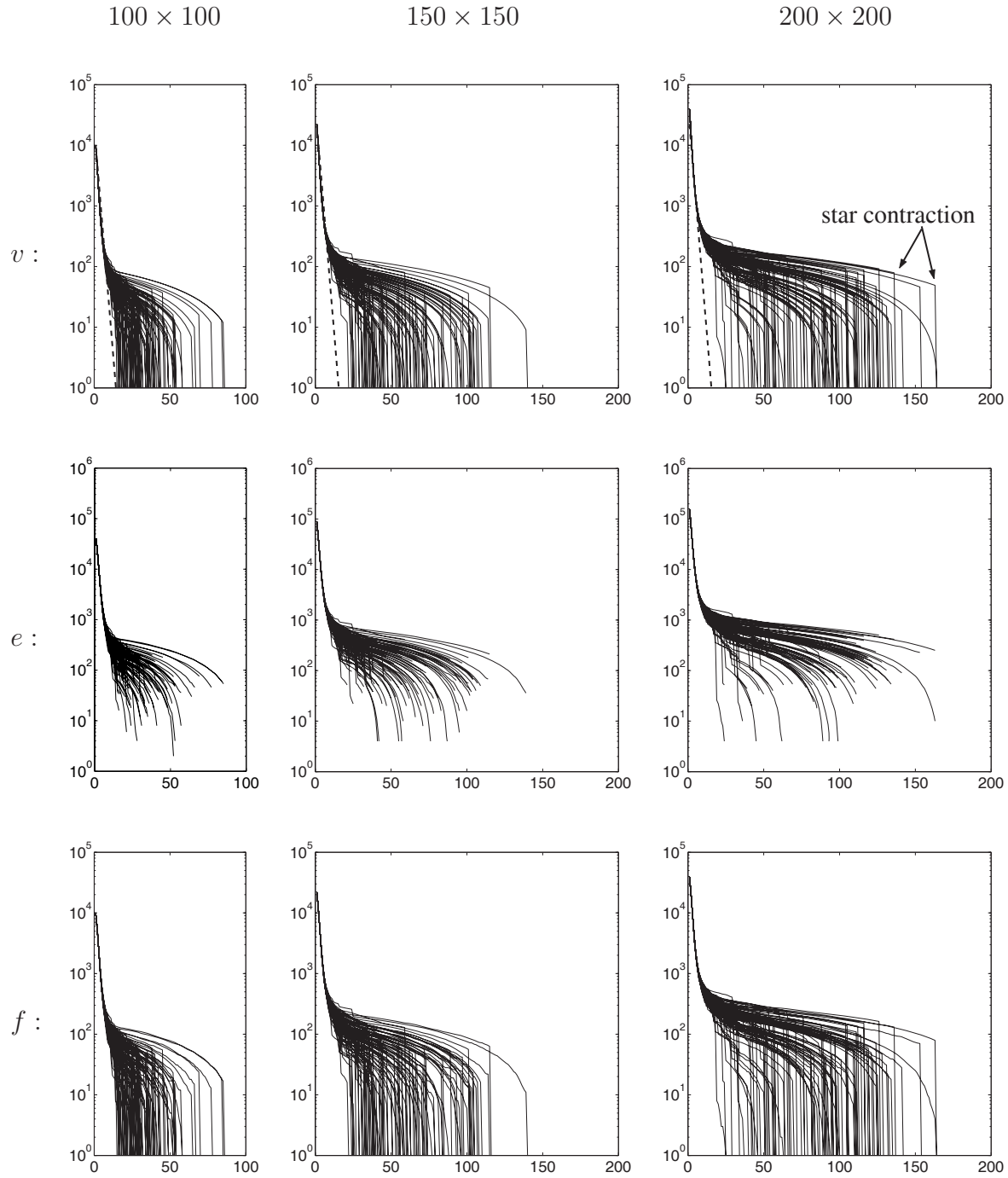

*L*egend: Number of vertices $v$, edges $e$, and faces $f$ ($y$-axis) in levels ($x$-axis ) of the first 100 MIES pyramids. The base levels are rectangular grid containing $100 \times 100$, $150 \times 150$ and $200 \times 200$ vertices. Dashed lines represent the theoretical reduction factor of 2.0.

**Figure 5.16:** MIES Algorithm.



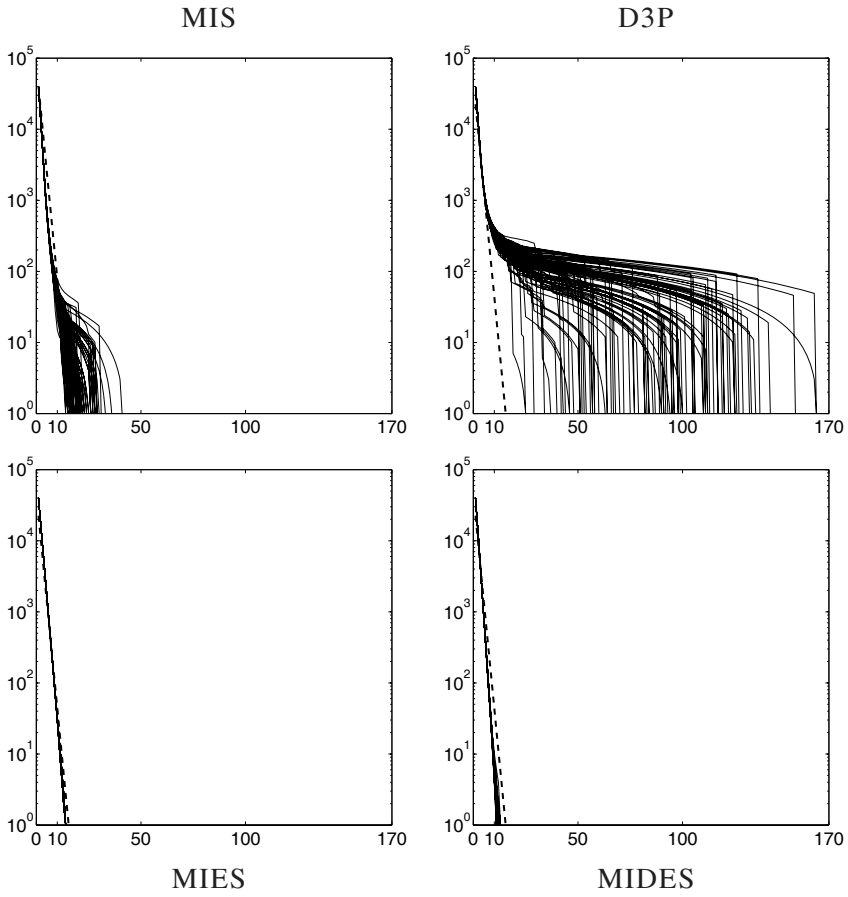

*L*egend: Number of vertices $v$, edges $e$, and faces $f$ ($y$-axis) in levels ($x$-axis ) of the first 100 MIDES pyramids. The base levels are rectangular grid containing $100 \times 100$, $150 \times 150$ and $200 \times 200$ vertices. Dashed lines represent the theoretical reduction factor of 2.0.

**Figure 5.17:** MIDES Algorithm.




*L*egend: Number of vertices $v$, edges $e$, and faces $f$ ($y$-axis) in levels ($x$-axis ) of the first 100 D3P pyramids. The base levels are rectangular grid containing $100 \times 100$, $150 \times 150$ and $200 \times 200$ vertices. Dashed lines represent the theoretical reduction factor of 2.0.

**Figure 5.18:** D3P Algorithm.




The base levels are rectangular grid containing $200 \times 200$ vertices. Dashed lines represent the theoretical reduction factor of $2.0$.

**Figure 5.19:** Direct comparison of vertex reduction ratio.

### 5.6.5 Discussion of Results

In Figure 5.19 the vertex reduction ratio of MIS, MIES, MIDES and D3P methods for $200 \times 200$ grid graphs is given for a direct comparison and easy following the presentation in this section. In Table 5.2 a more extensive comparison is made using $1000$ graphs of size $200 \times 200$. We extract three parameters, the maximum ($\max$), the mean ($\mu$) and the standard deviation ($\sigma$) of the vertex degree for any graph in the contraction process. We average these parameters (sample mean $\widehat{\mu}_{pyr}$ and sample standard deviation $\widehat{\sigma}_{pyr}$) for every experiment (i.e pyramid) as shown in the Table 5.1 and 5.2 over all experimental sets ($\widehat{\mu}_{data}$, $\widehat{\sigma}_{data}$). We made experiments using small grid graphs $26 \times 26$ ([Meer, 1989, Jolion, 2003]) and encountered no differences in the reduction factor between MIS, D3P, MIES and MIDES. The reason for this behavior is the small size of input instances. By using graphs of larger size ($100 \times 100$ and $200 \times 200$) different properties of these selection methods did occur.

The maximum degree was encountered using D3P and MIS which is why this method have problems during contractions. Thus memory costs for D3P and MIS will be higher than for

MIES and MIDES. Notice however that the mean degrees are similar for all algorithms. The height stability (in the sense of smaller variation) in the first case shows that MIS, MIES and MIDES do not depend on the data. The degrees for D3P exhibits more variations. This means that D3P better fit to the distribution of values associated with the vertices [Jolion, 2003], [Jolion, 2001].

**Table 5.1:** Summary statistics for $100 \times 100$ graphs.

| Algorithm | | height | vertex degree max | vertex degree $\widehat{\mu}_{pyr}$ | vertex degree $\widehat{\sigma}_{pyr}$ | $\lvert V_k/V_{k+1}\rvert$ $\widehat{\mu}_{pyr}$ | $\lvert V_k/V_{k+1}\rvert$ $\widehat{\sigma}_{pyr}$ | $\lvert E_k/E_{k+1}\rvert$ $\widehat{\mu}_{pyr}$ | $\lvert E_k/E_{k+1}\rvert$ $\widehat{\sigma}_{pyr}$ | correction $\widehat{\mu}_{pyr}$ | correction $\widehat{\sigma}_{pyr}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | max | 22.00 | 73.00 | | | | | | | | |
| **MIS** | $\widehat{\mu}_{data}$ | **14.39** | **32.24** | 4.39 | 2.13 | **2.18** | 0.86 | **2.17** | 0.75 | **3.01** | 0.81 |
| | $\widehat{\sigma}_{data}$ | 2.44 | 8.12 | 0.23 | 0.62 | 0.25 | 0.46 | 0.41 | 0.83 | 0.15 | 0.09 |
| | max | 13.00 | 13.00 | | | | | | | | |
| **MIES** | $\widehat{\mu}_{data}$ | **12.26** | **10.91** | 4.73 | 0.48 | **2.28** | 0.23 | **2.54** | 0.85 | **3.80** | 1.13 |
| | $\widehat{\sigma}_{data}$ | 0.44 | 0.74 | 0.11 | 0.03 | 0.07 | 0.10 | 0.14 | 0.41 | 0.14 | 0.11 |
| | max | 13.00 | 15.00 | | | | | | | | |
| **MIDES** | $\widehat{\mu}_{data}$ | **10.73** | **12.03** | 4.58 | 0.57 | **2.62** | 0.36 | **3.11** | 1.21 | **2.60** | 1.01 |
| | $\widehat{\sigma}_{data}$ | 0.62 | 0.82 | 0.20 | 0.03 | 0.15 | 0.13 | 0.24 | 0.62 | 0.17 | 0.13 |
| | max | 86.00 | 303.00 | | | | | | | | |
| **D3P** | $\widehat{\mu}_{data}$ | **34.72** | **148.76** | 4.68 | 5.84 | **1.90** | 2.92 | **1.33** | 0.45 | **1.00** | 0.00 |
| | $\widehat{\sigma}_{data}$ | 15.00 | 47.67 | 0.20 | 1.23 | 0.61 | 2.28 | 0.16 | 0.21 | 0.00 | 0.00 |

**Table 5.2:** Summary statistics for $200 \times 200$ graphs.

| Algorithm | | height | vertex degree max | vertex degree $\widehat{\mu}_{pyr}$ | vertex degree $\widehat{\sigma}_{pyr}$ | $\lvert V_k/V_{k+1}\rvert$ $\widehat{\mu}_{pyr}$ | $\lvert V_k/V_{k+1}\rvert$ $\widehat{\sigma}_{pyr}$ | $\lvert E_k/E_{k+1}\rvert$ $\widehat{\mu}_{pyr}$ | $\lvert E_k/E_{k+1}\rvert$ $\widehat{\sigma}_{pyr}$ | correction $\widehat{\mu}_{pyr}$ | correction $\widehat{\sigma}_{pyr}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | max | 41.00 | 148.00 | | | | | | | | |
| **MIS** | $\widehat{\mu}_{data}$ | **20.80** | **70.69** | 4.72 | 3.67 | **2.01** | 1.30 | **1.82** | 0.69 | **3.01** | 0.82 |
| | $\widehat{\sigma}_{data}$ | 5.24 | 23.89 | 0.22 | 1.16 | 0.35 | 1.08 | 0.26 | 0.34 | 0.17 | 0.11 |
| | max | 15.00 | 13.00 | | | | | | | | |
| **MIES** | $\widehat{\mu}_{data}$ | **14.02** | **11.78** | 4.90 | 0.47 | **2.27** | 0.22 | **2.55** | 1.03 | **4.04** | 1.20 |
| | $\widehat{\sigma}_{data}$ | 0.14 | 0.68 | 0.05 | 0.03 | 0.01 | 0.05 | 0.27 | 0.72 | 0.11 | 0.12 |
| | max | 13.00 | 18.00 | | | | | | | | |
| **MIDES** | $\widehat{\mu}_{data}$ | **12.05** | **13.29** | 4.78 | 0.58 | **2.63** | 0.32 | **3.13** | 1.31 | **2.83** | 1.10 |
| | $\widehat{\sigma}_{data}$ | 0.40 | 1.07 | 0.13 | 0.04 | 0.10 | 0.16 | 0.31 | 0.83 | 0.15 | 0.10 |
| | max | 164.00 | 689.00 | | | | | | | | |
| **D3P** | $\widehat{\mu}_{data}$ | **89.25** | **433.92** | 5.10 | 9.18 | **1.63** | 4.16 | **1.15** | 0.37 | **1.00** | 0.00 |
| | $\widehat{\sigma}_{data}$ | 31.72 | 123.41 | 0.15 | 1.89 | 0.58 | 3.96 | 0.17 | 0.55 | 0.00 | 0.00 |

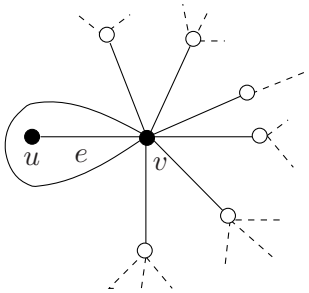



**Figure 5.20:** Vertex $u$ is favored by MIS and D3P.

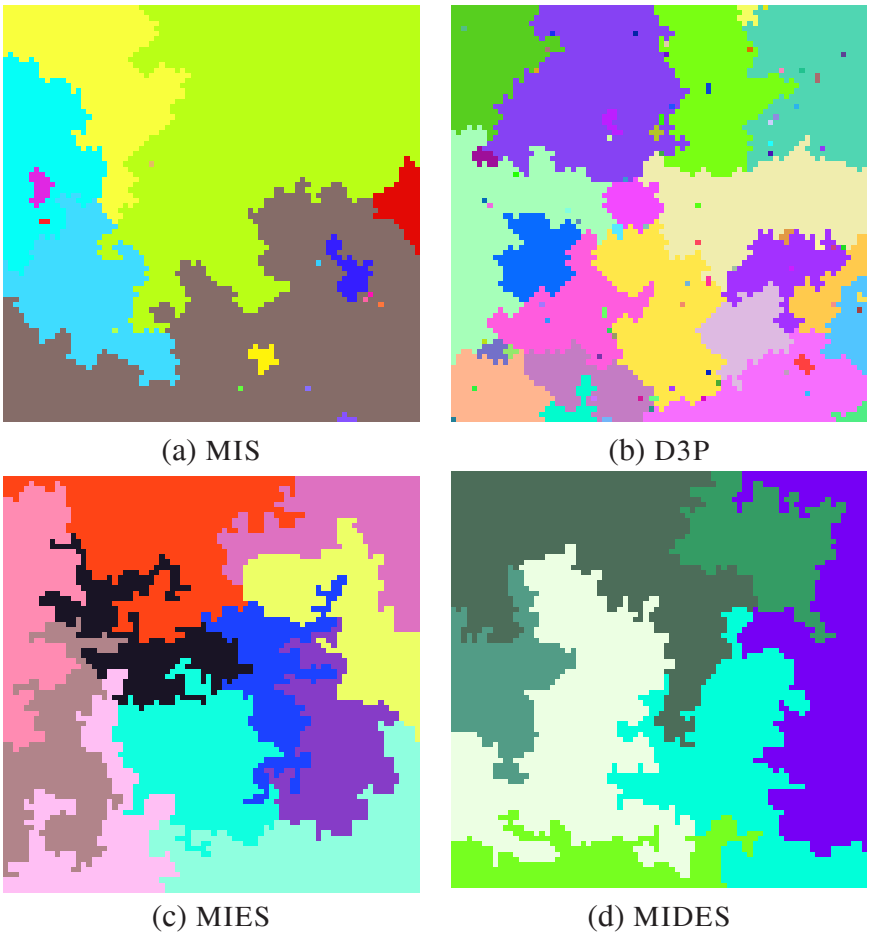
Table 5.1, and 5.2 gives statistics on *height* of the image pyramid, *vertex degree*, *decimation ratios*, and iteration for *corrections* to complete the maximal independent set. Results in both tables are computed using $1000$ graphs of size $100 \times 100$, and $200 \times 200$, respectively. Number of iterations for correction was almost the same for all methods except for the D3P, where it was $1$ because we do not iterate at all. MIDES gave the best reduction factors ($\widehat{\mu}_{data}(v) = 2.63$) for all tests (Table 5.2). MIS, MIDES and D3P have the same algorithmic complexity for the worst case. The worst case happens when the neighboring vertices have increasing random numbers. We have not encountered the worst case in our test, since it is highly unlikely. The *a priori* probability that a vertex survives depends on the size of its neighborhood. In Figure 5.20 vertex $u$ will be favored by MIS and D3P. Vertex $u$ survives with the probability of $1/2$, since it has only the vertex $v$ in the neighborhood. Vertex $v$ survives with the probability of $1/n$, where $n$ is the size of its neighborhood. The center of the star will have smaller probability to survive than its leafs, which causes the poor reduction factor, since only one edge will be contracted. Since two surviving vertices cannot be neighbors, the center of the star will be pulled toward one of it leafs. But still there will be a vertex with large neighborhood. There are cases where the center of the star is the largest in its neighborhood. In these cases all its leafs will be contracted toward the star, yielding a very good reduction factor. The contraction of stars can be seen in Figures 5.15, and 5.18, where the solid lines descent rapidly. An arrow in Figure 5.15($v$) and Figure 5.18($v$) depict an example of star contraction. The probability that edge $e$ in Figure 5.20 will be contracted i.e. one of the end vertices will survive, is the same for all neighboring edges of $e$ using MIES or MIDES. The existence of vertices with large neighborhood was not encountered in MIES and MIDES, which can be seen also in Figures 5.16, and 5.17, where there are no cases of rapidly descent lines.

In Table 5.3 statistics of edge neighborhood for MIES and MIDES for $100 \times 100$ graphs are given. The edge neighborhood for both of the methods are of the same order as their vertex neighborhood i.e. compare in MIES edge neighborhood: $\widehat{\mu}_{data}(\widehat{\mu}_{pyr}) = 7.89$ with vertex neighborhood: $\widehat{\mu}_{data}(\widehat{\mu}_{pyr}) = 4.73$; and in MIDES edge neighborhood: $\widehat{\mu}_{data}(\widehat{\mu}_{pyr}) = 3.41$ with vertex neighborhood: $\widehat{\mu}_{data}(\widehat{\mu}_{pyr}) = 4.58$ respectively).

Since no vertex is favored the size of receptive fields using MIES or MIDES will be better distributed, in the intermediate levels of the pyramid. There is no occurrence of very small or very big receptive fields as for MIS or D3P, where there are receptive fields as small as one pixel in higher levels of the image pyramid. Figure 5.21 shows receptive fields of comparable pyramid

**Table 5.3:** Statistics of edge degree for $100 \times 100$.

| Algorithm | | Edge degree max | $\widehat{\mu}_{pyr}$ | $\widehat{\sigma}_{pyr}$ |
|---|---|---|---|---|
| **MIES** | $\widehat{\mu}_{data}$ | **17.27** | **7.89** | 0.71 |
| | $\widehat{\sigma}_{data}$ | 0.91 | 0.24 | 0.06 |
| **MIDES** | $\widehat{\mu}_{data}$ | **9.01** | **3.41** | 0.41 |
| | $\widehat{\sigma}_{data}$ | 0.01 | 0.12 | 0.08 |



(a) MIS (b) D3P

(c) MIES (d) MIDES

**Figure 5.21:** Receptive fields.

levels of MIS, D3P, MIES and MIDES. Each vertex of the same level received arbitrary but distinguishable gray values which was propagated down to the base of the pyramid. Hence the regions with the same gray value in Figure 5.21 correspond to the receptive field of one vertex of the high level. Without any constraint from the data there is no need to require big variations in the sizes of receptive fields. This kind of behavior was encountered also in Section 5.7.1.

## 5.7 Comparing the Path Lengths

Path length are computed as steps to bring an attribute from every vertex in the base (bottom-up) to the top of the pyramid, and from a top to a vertex in the base (top-down). We call this vertical path lengths. The experiments in this section compare vertical path lengths (costs) in stochastic graph pyramids. This will allow to measure the structure of the irregular pyramids built with different algorithms. In some applications some parts of image are of special interest, so we need to access data in a top-down process very often [Glantz and Kropatsch, 2000a]. We are interested not only in comparing the complexity of vertex degree and the height of the pyramid, as done in Section 5.6, but also to be able to compare the internal structure of image pyramids of the same height. Our goal is to build stochastic pyramids locally that are optimal in the sense of bottom-up and top-down processes.

First we built a graph pyramid in a bottom-up process using one of the algorithms MIS, MIES, MIDES or D3P to find contraction kernels (decimation parameters) as presented previously. Graphs are reduced using the dual graph contraction [Kropatsch, 1995a]. Since we build stochastic graph pyramids we end having always a single vertex at the top of a pyramid. Figure 5.22a shows a rooted tree on level $G_k$ (the children) and their relation (dashed lines) with the vertex on $G_{k+1}$ (the parent); white vertices on $G_k$ are the non-surviving children $ns$, and they are contracted (arrows) toward the surviving children $ss$, depicted with black. Note that, the costs for inheritance from parents to surviving child is kept zero since it is a simple copy of the attributes. Costs for contracting an edge are set to 1 since reduction involves the merging operation for the attributes of the two end vertices. This means that the surviving child will have the path length of the parent and non-surviving children the incremented path length of the parent by one.

Vertical paths connect the apex with the base of the pyramid following *parent-children* relations from level to level. Path lengths $p \in \mathbb{N}$ of vertices at $G_0$ (level 0) can be found in a top-down process as in Algorithm 7. An example of vertical path is shown in Figure 5.22b. We

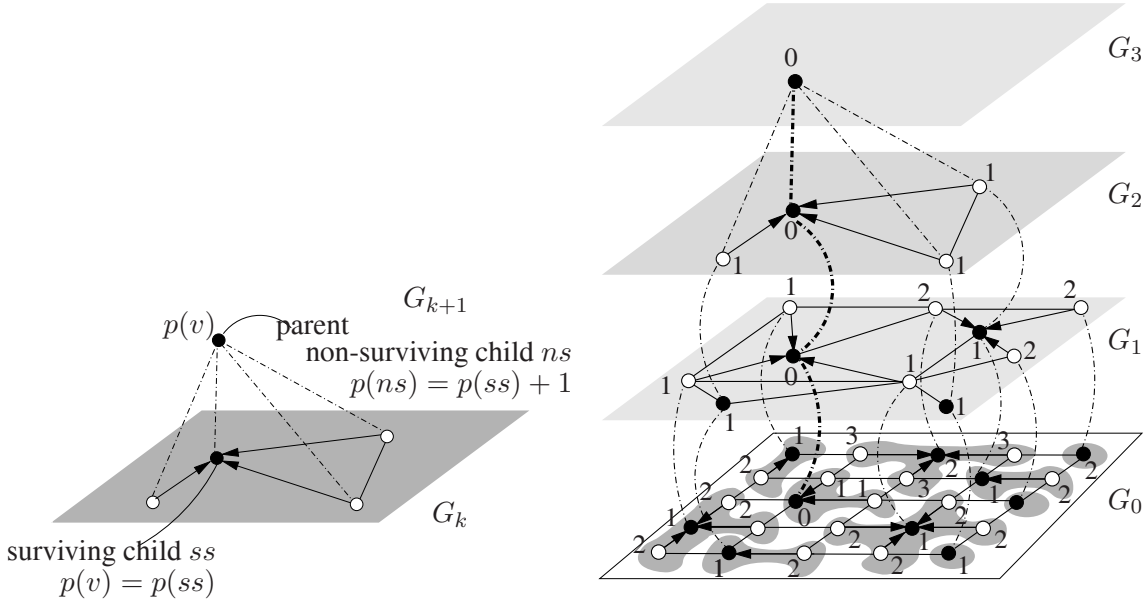
**Algorithm 7 – Path Length Algorithm**

*Input*: A graph pyramid $G_k = (V_k, E_k)$, $\forall k = 0, ..., h$

1: Let the vertices $v \in G_h$ at a top level of the pyramid have path length 0, $p(v) \leftarrow 0$
2: **for** $k = k-1, k-2, ..., 0$ **do**
3: $\forall v \in G_{k+1}$: down propagate the path length $p(v)$ of the vertex $v$ at level $k+1$ to its surviving child $ss$ at level $k$ below, so the path length of $ss$ is $p(ss) \leftarrow p(v)$
4: all non-surviving children $ns \in V_k$ of $v$ at level $k$ have path length $p(ns) \leftarrow p(v) + 1$
5: **end for**

*Output*: path lengths of the vertices.

start at the top of the pyramid. For a stochastic image pyramid there is only one vertex at the top of the pyramid, vertex $h$ at the top has path length 0 and the surviving child $p(ss) = 0$. An one-to-one relationship between children (vertices at $k$) and parents (vertices at $k+1$) is created during construction of the image pyramid [Kropatsch, 1995a]. The number of vertices $|V_{k+1}|$ in level $k+1$ is the same as the number of surviving vertices in $|V_k|$ (surviving children) in level




(a) Parent-child relations (b) Path lengths

**Figure 5.22:** Graph pyramid built using DGC.

$k$. In Figure 5.22b at $G_2$ a vertex has the same length as its parent (black vertex with path length $0$), and all non-surviving children (white vertices) have the path length $0 + 1 = 1$. To arrive to the top of the pyramid from the edge with path length $3$, three edges must be contracted.

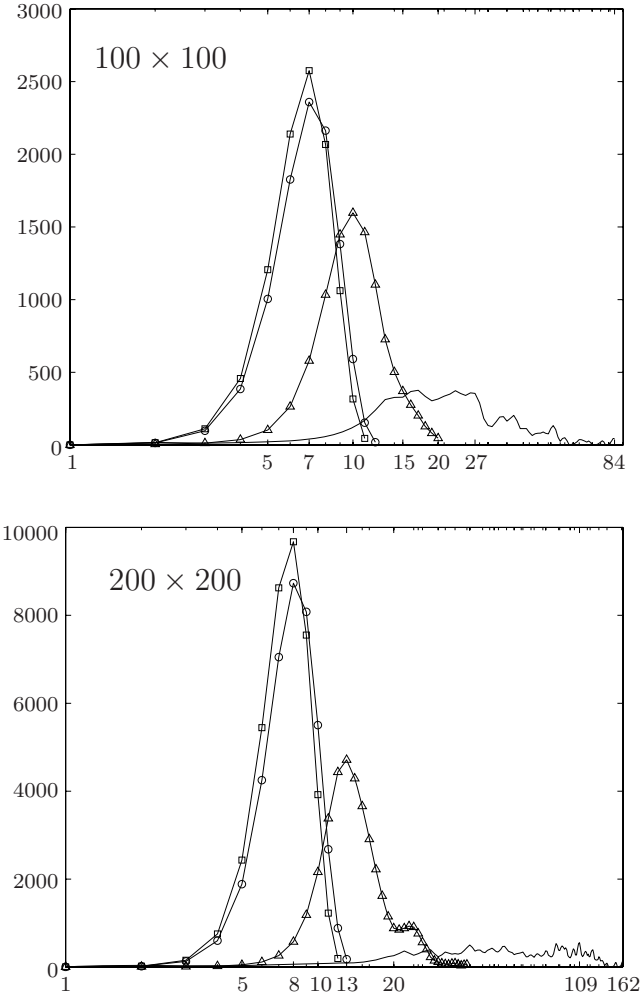
The path length shows costs to arrive to the top of the pyramid (vertical path lengths) from every vertex in the base. This is closely related to the height of the pyramid, since higher pyramid would imply longer paths.

### 5.7.1 Experimental Results and Discussion

We use one of the algorithms MIS, MIES, MIDES or D3P to build stochastic pyramids i.e to compute the path lengths on the same setup as in the Section 5.6 In these experiments we used grid graphs of size $10000$ and $40000$ vertices respectively, which correspond to image sizes of $100 \times 100$ and $200 \times 200$ pixels.

The result of the mean value of number of vertices per path length over $100$ pyramids are given in Figure 5.23, (a) for $100 \times 100$ and (b) $200 \times 200$ image size. The two diagrams of Figure 5.23 show on the $x$-axis in a logarithmic scale the length of the vertical paths and on the $y$-axis the number of vertices of the base level. Each base vertex has a certain 'vertical distance' to the apex, the number of vertices having the same vertical distance can be accumulated in the histogram of vertical path lengths. Every pyramid generates such a histogram using the *path length algorithm*. Histograms generated by a particular selection strategy can be averaged and are shown for the MIS, MIES, MIDES and D3P strategies in Figure 5.23.

Path lengths i.e. costs for MIES and MIDES are smaller, even when the image size were $4$ times larger. MIS and D3P have tendency to have longer path lengths. Table 5.4 shows the




*L*egend: △ MIS, ◯ MIES, □ MIDES, and − D3P.
$x$ axis the path lengths, and $y$-axis number of vertices.

**Figure 5.23:** $\widehat{\mu}$ of path lengths.

maximal path length ($\max$) and in brackets the most frequent path length ($\widehat{\mu}$). For MIES and MIDES almost $50\%$ of vertices have path length of $6$, $7$, and $8$ for $100 \times 100$, and $7$, $8$, and $9$ for $200 \times 200$ image, respectively. These values are comparable with the $log(diameter)$, which would be the height of the regular pyramid, a property we are trying to achieve. The MIS, and D3P path lengths are longer in both cases. Also the maximum height found by using different decimation strategies is of the same order as the maximum path length used for the same strategy (compare maximum height of Table 5.1, and 5.2 with maximal path length of Table 5.4 respectively, e.g. for $100 \times 100$ using MIS:$\max$ height$= 22$ and $\max$ path length $= 20$; MIES:$\max$ height$= 13$ and $\max$ path length $= 12$, MIDES:$\max$ height$= 13$ and $\max$ path length $= 11$; and D3P:$\max$ height$= 86$ and $\max$ path length $= 84$.). This is valid also for the mean value of the height and the path length.

Vertical path lengths cannot be explained as the path lengths of equivalent contraction ker-

**Table 5.4:** The maximum path length.

| Algorithm | Image size | | | |
|---|---|---|---|---|
| | $100 \times 100$ | | $200 \times 200$ | |
| | max | $\widehat{\mu}$ | max | $\widehat{\mu}$ |
| MIS | 20 | 10 | 39 | 13 |
| MIES | 12 | 7 | 13 | 8 |
| MIDES | 11 | 7 | 12 | 8 |
| D3P | 84 | 27 | 160 | 109 |
| diameter | 200 | | 400 | |
| log(diameter) | 8 | | 9 | |

nels (ECK) toward the root (the apex) of the pyramid. Figure 5.24 shows the vertical path lengths (numbers inside the vertices) and the ECK at the top of the pyramid (the tree). As can be seen vertical path length is not the same as the number of steps (edges) needed to arrive from a vertex (any of them) to the apex (shown with black). In Figure 5.24 the corner vertex $(2)$ shown in gray will need $59$ steps to arrive to the apex, which is quite large in comparison with $2$ which is the vertical path length. So using ECK will imply larger costs to access the pixel at the base level. To compare or compute all pixel pairs of an image of size $|s|$ in an array the time complexity is $\mathcal{O}(s^2)$. For the pyramid of height $h$ is twice the sum of path lengths $p$ i.e. $\mathcal{O}(s \cdot p)$ and if the cost of vertical neighborhood are taken into account then it is $\mathcal{O}(s \cdot p \cdot \log s)$. Note that $p$ in experiments with MIS, MIES and MIDES is a constant (slowly changing function) and thus $\mathcal{O}(s \cdot p \cdot \log s) = \mathcal{O}(s \cdot \log s)$.

To summarize, MIS and MIDES tend to find shorter paths. Longer path lengths imply bigger costs to access a vertex in the base from the apex of the pyramid.

## 5.8 Top-down Optimization

In images often large homogeneous regions need to be shrinked into a single vertex of the region adjacency graph. In this case it is important to summarize the properties of the large region in a small number of steps [Kropatsch et al., 2006]. The methods presented above in this chapter build contraction kernels stochastically, i.e. they summarize data of regions in local bottom-up stochastic way. However other approaches are also possible. In this section briefly, one such method is presented due to [Kropatsch et al., 2006]. In this method, first a spanning tree of the regions (e.g by using a minimum spanning tree) is build, afterward this tree is decomposed recursively into small subtrees (contraction kernels) of depth one that if contracted would produce the optimal height of the pyramid. The method is global [Kropatsch et al., 2006] since in a top-down manner decomposes the spanning tree. Moreover this method is deterministic. In the Table 5.5 the comparison of the bottom-up methods (MIS, MIES and MIDES) with the top-down method (recursive decomposition of a tree (RDT)) are shown. Random graphs of $4000$ vertices and $27700$ edges are used as the basis of pyramid, and simple statistic is done on $100$ pyramids, by finding the minimal and maximal value of the height of pyramid ($h$), as well as

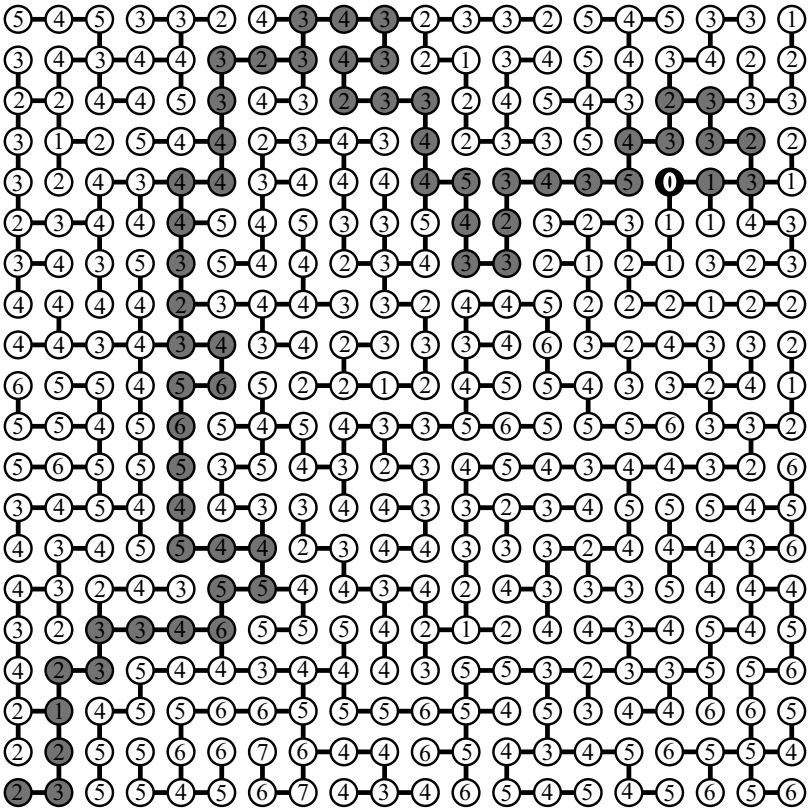


**Figure 5.24:** Path length and ECK.

**Table 5.5:** Comparison of the height of the pyramid $h$ (adapted from [Kropatsch et al., 2006]).

| Algorithm | min(h) | max(h) | $\widehat{\mu}_h$ | $\widehat{\sigma}_h$ | Approach |
|---|---|---|---|---|---|
| Stochastic with MIS | 9 | 20 | 12.39 | 2.43 | bottom-up, local |
| Stochastic with MIES | 10 | 11 | 10.26 | 0.44 | bottom-up, local |
| Stochastic with MIDES | 8 | 11 | 8.73 | 0.61 | bottom-up, local |
| Deterministic with RDT | 7 | 8 | 9.94 | 0.01 | top-down, global |

the mean ($\widehat{\mu}_h$) and standard deviation ($\widehat{\sigma}_h$). As can be seen the deterministic top-down method build a shallower pyramid than the other method, which is expected. Note that the deterministic method in order to decompose the tree needs a tree as input. This tree can be also produced by using one of the methods: MIS, MIES or MIDES.

This idea of top-down tree decomposition is used in [Kropatsch et al., 2004, Kropatsch et al., 2004] to compute a boundary distance of a binary shape, the depth of all subtrees, and the diameter of all outer subtrees. These features capture the properties of complete substructures.

## 5.9 Conclusion

The efficiency of pyramids is tightly coupled with their ability to propagate information from any cell to any other cell in at most twice the height's steps. The gained freedom in choosing structures adapting the data may affect the height if the reduction proceeds too slowly.

Experiments with stochastic decimation using maximal independent vertex sets (MIS) showed a problematic behavior on large images. After an initial phase of strong reduction, the reduction decreases dramatically. This is due to the evolution of larger and larger variations between the vertex degrees in the contracted graphs. To overcome this problem we proposed a method, MIES, based on matchings which guarantees a reduction factor of $2.0$. As in the case of independent vertex sets, the method based on matchings does not allow the control of the directions of the contractions. The second method, MIDES, that we proposed and tested is based on directed edges and allows the control of the directions of the contractions. The experiments showed a non-decreasing reduction that was even stronger than the one obtained MIES. We have shown that $2.0$ is also a bound for the reduction with MIDES if no isolated vertices are encountered. Furthermore we were able to characterize the configuration creating such isolated vertices which explains the good experimental results. The properties of these configurations are important when the random sampling is replaced by data-adaptive importance values where no prediction about statistical distribution can be made.

CHAPTER 6

# Irregular Graph Image Partitioning

> ” Ich stehe vor einen Fenster und sehe ein Haus, Bäume, Himmel. Theoretisch koennte ich sagen, daß es 327 Farbnuancen und Helligkeiten gibt. Habe ich 327? Nein. Ich habe Himmel, Haus und Bäume. ” [1]
>
> **by Max Wertheimer.**

**Summary** We present a hierarchical partitioning of images using a pairwise similarity function on a graph-based representation of an image. This function measures the difference along the boundary of two components relative to a measure of differences of the components' internal differences. This definition tries to encapsulate the intuitive notion of contrast. Two components are merged if there is a low-cost connection between them. Each component's internal difference is represented by the maximum edge weight of its minimum spanning tree. External differences are the smallest weight of edges connecting components. We use this idea for building a minimum spanning tree to find region borders quickly and effortlessly in a bottom-up way, based on local differences in a specific feature.

**Keywords:** Hierarchical graph-based image partitioning, irregular graph pyramids, minimum weight spanning tree, topology preserving contraction.

## 6.1 Introduction

The authors in [Keselman and Dickinson, 2001] asked: ”How do we bridge the representational gap between image features and coarse model features?” They identify the 1-to-1 correspondence between salient image features (pixels, edges, corners,...) and salient model features (generalized cylinders, polyhedrons,...) as limiting assumption that makes prototypical or generic

[1] I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have 327? No. I have sky, house, and trees.

object recognition impossible. They suggested to bridge and not to eliminate the representational gap, and to focus efforts on:

- **region segmentation**,
- **perceptual grouping**, and
- **image abstraction**.

In this chapter a method is given that focuses on image partitioning and perceptual grouping. The multi-resolution is considered under viewpoint of the abstraction in [Kropatsch, 2002].

Wertheimer [Wertheimer, 1925] has formulated the importance of wholes (Ganzen) and not of its individual elements as: "Es gibt Zusammenhänge, bei denen nicht, was im Ganzen geschieht, sich daraus herleitet, wie die einzelnen Stücke sind und sich zusammensetzen, sondern umgekehrt, wo - im prägnanten Fall - sich das, was an einem Teil dieses Ganzen geschieht, bestimmt von inneren Strukturgesetzen dieses seines Ganzen" [2], and introduced the importance of perceptual grouping and organization in visual perception.

The union of regions forming the group is again a region with both internal and external properties and relations. Low-level cue image segmentation cannot and should not produce a complete final 'good' segmentation, because there is *an intrinsic ambiguity* in the exact location of region boundaries in digital images [Chen and Pavlidis, 1980] as well as the problems in defining the context of an image. Problems emerge because:

- homogeneity of low-level cues will not map to the semantics [Keselman and Dickinson, 2001], and
- the degree of homogeneity of a region is in general quantified by threshold(s) for a given measure [Fuh et al., 2000]

Even though the segmentation methods (ours as well) that do not take the context of the image into consideration cannot produce a 'good' segmentation, they can be valuable tools in image analysis in the same sense as efficient edge detectors are. Note that efficient edge detectors do not consider the larger context of the image, too. Therefore, the low-level coherence of brightness, color, texture or motion attributes should be used to come up sequentially with hierarchical partitions [Shi and Malik, 1997]. Mid and high level knowledge can be used to either confirm these groups or select some further attention. A wide range of computational vision problems could make use of segmented images, were such segmentation rely on efficient computation. For instance motion estimation requires an appropriate region of support for finding correspondence. Higher-level problems such as recognition and image indexing can also make use of segmentation results in the problem of matching.

It is important that a grouping method has following properties [Felzenszwalb and Huttenlocher, 2004]; it should

- capture **perceptually important groupings** or regions, which reflect global aspects of the image,

[2] "There are wholes (Ganzen), the behaviour of which is not determined by that of their individual elements, but where the part-processes are themselves determined by the intrisinic nature of the whole" [Willis, 1997]

- be **highly efficient**, running in time linear in the number of image pixels,
- create **hierarchical partitions** [Shi and Malik, 1997].

It is already shown that in regular image pyramids the number of pixels at any level, is $\lambda$ times higher than the number of pixels at the next reduced level. This is called the reduction factor $\lambda$ and it is greater than one and constant for all levels. Let $s$ denote the number of pixels in an image $I$, the number of new levels on top of $I$ amounts to $log_\lambda(s)$. Thus, the regular image pyramid may be an efficient structure for fast grouping and access to image objects in top-down and bottom-up processes.

However, it is shown that regular image pyramids are confined to globally defined sampling grids and lack shift invariance [Bister et al., 1990]. [Bister et al., 1990] conclude that regular image pyramids have to be rejected as general-purpose segmentation algorithms. [Montanvert et al., 1991] and [Jolion and Montanvert, 1992] describe how these drawbacks can be avoided by irregular image pyramids, the so called *adaptive pyramids*, where the hierarchical structure (the vertical network) of the pyramid is not *a priori* known but recursively built based on the data. Moreover, [Cho and Meer, 1997], [Guigues et al., 2003], [Nacken, 1995], [Shen et al., 1998], [Mathieu et al., 1992], [Borowy and Jolion, 1995], [Meer et al., 1990] show that irregular pyramids can be used for segmentation and feature detection. All these pyramids use only local information to build the hierarchy. Usually on the base level the cells represent single pixels and the neighborhood of the cells is defined by the connectivity of the pixels. Every parent computes its values independently of other cells on the same level. This implies that an image pyramid is built in $\mathcal{O}[log(image\_diameter)]$ time [Jolion and Rosenfeld, 1994].

In this chapter we represent the levels of the pyramid as the dual pair of graphs $(G_l, \overline{G_l})$ of plane graphs $G_l$ and its dual (plane) graph $\overline{G_l}$. In graph-theoretical terms image region extraction and border extraction are dual problems and as an immediate consequence of this is that the region boundaries extracted are always closed curves [Sanfeliu et al., 2002]. The sequence $(G_l, \overline{G_l})$, $0 \leq l \leq h$ is called (dual) graph pyramid. Moreover the graph is attributed, $G = (V, E, attr_v, attr_e)$, where $attr_v : V \rightarrow \mathbb{R}^+$ is a weighted function defined on vertices and $attr_e : E \rightarrow \mathbb{R}^+$ is a weighted function defined on edges. We use a weight for $attr_e$ measuring the difference between the two end points.

The aim of this chapter is to build a minimum weight spanning tree (MST) of an image by combining the advantage of regular pyramids (logarithmic tapering) with the advantages of irregular graph pyramids (their purely local construction and shift invariance). The aim is reached by using the selection method (MIES) for contraction kernels proposed in [Haxhimusa et al., 2002] to achieve logarithmic tapering, local construction and shift invariance. Borůvka's algorithm [Borůvka, 1926a], [Borůvka, 1926b], [Neštřil et al., 2001] is combined with dual graph contraction [Kropatsch, 1995a] for building in a hierarchical way a minimum weight spanning tree (of the region) and preserving topology at the same time. The topological relation seems to play an even more important role for vision tasks in natural systems than precise geometrical position. We use the idea of building a minimum weight spanning tree to find region borders quickly and effortlessly in a bottom-up 'stimulus-driven' based on local differences in a specific feature like in an pre-attentive vision.

The plan of the chapter is as follows. In order to make the reading of this chapter easy we recall some of the basic notion as follows. In Section 6.2 we recall the main idea of the minimum weight spanning tree and in Subsection 6.2.1 we recall Borůvka's MST algorithm,

as well as Kruskal's Algorithm and Prim-Jarniks's Algorithm (Section 6.2.2 and Section 6.2.3, respectively) . Using the dual graph contraction algorithm of Section 4.4, Borůvka's algorithm is re-defined in Section 6.3, so that we can construct an image graph pyramid, and, at the same time the minimum spanning tree. In Section 6.4 we give the definition of internal and external contrast and the merge decision criteria based on these definitions. The algorithm for building the hierarchy of partitions is introduced in this section. Section 6.5 reports on experimental results.

### 6.1.1 Related Work

A graph-theoretical clustering algorithm consists in searching for a certain combinatorial structure in the edge weighted graph, such as a minimum spanning tree [Felzenszwalb and Huttenlocher, 2004], [Fischer and Buhmann, 2002], [Guigues et al., 2003], a minimum cut [Wu and Leahy, 1993], [Shi and Malik, 1997], [Gdalyahu et al., 2001] and, among these methods a classic approach to clustering (the *c*omplete linkage clustering algorithm [Lance and Williams, 1967]) reduces to a search for a complete subgraph i.e. the maximal clique [Pavan and Pelillo, 2003], [Pavan and Pelillo, 2003].

Hierarchical structures for description of the data for clustering purposes have been studied very early in [Lance and Williams, 1967], or for image segmentation in [Horowitz and Pavlidis, 1976]. Image pyramids are a powerful tool for early vision. See [Jolion and Rosenfeld, 1994] and [Rosenfeld, 1984] for an extensive overview of the topic. [Hird and Willson, 1989] demonstrates that hierarchical segmentation on a pyramid yields better results than conventional 2D techniques. However, the regular image pyramids are confined to globally defined sampling grids and lack shift invariance. [Bister et al., 1990] concludes that regular image pyramids have to be rejected as general-purpose segmentation algorithms. In [Montanvert et al., 1991], [Jolion and Montanvert, 1992] it was shown how these drawbacks can be avoided by irregular image pyramids, the so called adaptive pyramids, where the hierarchical structure (vertical network) of the pyramid was not 'a priori' known but recursively built based on the data. [Meer et al., 1990] in his 'consensus vision' used the concept of irregular pyramid to produce an image segmentation. Moreover in [Cho and Meer, 1997], [Guigues et al., 2003], [Marfil et al., 2004], [Nacken, 1995], [Shen et al., 1998], [Borowy and Jolion, 1995], [Mathieu et al., 1992] was shown that irregular pyramid can be used for segmentation and feature detection.

The clustering community [Jain and Dubes, 1988] has produced agglomerative and divisive algorithms; in image segmentation the region-based merging and splitting algorithms exist. Early graph-based methods [Zahn, 1971] use fixed thresholds and local measures in computing a segmentation, i.e the minimum spanning tree (MST) is computed. The segmentation criterion is to break the MST edges with the largest weight. The idea behind is that edges in the MST reflect the low-cost connection between two elements. The work of Urquhart [Urquhart, 1982] attempts to overcome the problem of fixed threshold by normalizing the weight of an edge using the smallest weight incident on the vertices touching that edge. The methods in [Felzenszwalb and Huttenlocher, 2004], [Fischer and Buhmann, 2002], [Guigues et al., 2003] uses an adaptive criterion that depend on local properties rather than global ones. Recently the works [Felzenszwalb and Huttenlocher, 2004], [Guigues et al., 2003], [Sanfeliu et al., 2002], [Vergés-Llahi et al., 2000], [Fischer and Buhmann, 2002], [Meyer, 1999] have the minimum spanning tree as the base algorithm. It is shown in [Chowdhury and Murhty, 1997] that minimum spanning

tree clustering technique, although unsupervised one, approaches the performance of 'Bayes classifier', as the number of sample points from each class increases.

Gestalt grouping factors, such as proximity, similarly, continuity and symmetry, are encoded and combined in pairwise feature similarity measures [Wu and Leahy, 1993], [Shi and Malik, 1997], [Hofmann et al., 1998], [Perona and Freeman, 1998], [Gdalyahu et al., 1998], [Weiss, 1999], [Sharon et al., 2000], [Yu and Shi, 2001], [Fischer and Buhmann, 2002]. Other method of segmentation is that of splitting and merging region based on how well the regions fulfill some criterion. Such methods [Cooper, 1998], [Pavlidis, 1977] uses a measure of uniformity of a region. Authors in [Felzenszwalb and Huttenlocher, 1998], [Fischer and Buhmann, 2002], [Guigues et al., 2003] use, in contrast, in a graph-based method a pairwise region comparison rather than applying a uniformity criterion to each individual region. It has been demonstrated that complex grouping phenomena can emerge from simple computation on these local cues [Guy and Medioni, 1996], [Malik et al., 2001].

The use of Markov Random Field (MRF) has been used for image restoration and segmentation [Geman and Geman, 1984]. However the use of MRF to image segmentation usually leads to NP-hard problems. The graph-based approximation method for MRF problems [Boykov et al., 1998] yields practical solution, if the number of labels for the pixel is small, which limits these methods for use in segmentation.

The methods based on minimum cuts in graph are designed to minimize the similarity between pixels that are being split [Wu and Leahy, 1993], [Shi and Malik, 1997], [Gdalyahu et al., 2001]. [Wu and Leahy, 1993] define a cut criterion, but it was biased toward finding small components. [Shi and Malik, 1997] developed the normalized cut criterion to address this bias, which takes into consideration self-similarity of regions. These cut-criterion methods capture the non-local properties of the image, in contrast with the simple graph-based methods such as breaking edges in the MST. It also produce divisive hierarchical tree, the dendogram. However they provide only a characterization of such cut rather than of final segmentation as is provided by [Felzenszwalb and Huttenlocher, 2004]. [Shi and Malik, 1997] developed an approximation method for computing the minimum normalized cut, the error in these approximation in not well understood (it is closely related to spectral graph methods, e.g [Fiedler, 1975]) and this method is computational expensive.

The minimal spanning tree and the minimum cut are explicitly defined on edge weighted graph, whereas the concept of a maximal clique is defined on unweighted edge graphs. As a consequence, maximal clique based clustering algorithms work on unweighted graphs derived from the edge weighted graphs by means of some thresholding [Jain and Dubes, 1988]. [Pavan and Pelillo, 2003] and [Pavan and Pelillo, 2003] generalized the concept of maximal clique to weighted graphs. The maximal cliques are found using the discrete replicator dynamics, which turns out to be an instance of relaxation labeling algorithm [Rosenfeld et al., 1976].

A disadvantage of graph-theoretical approaches to image segmentation, as can be seen in [Wu and Leahy, 1993], [Vlachos and Constantinides, 1993], [Xu and Uberbacher, 1997], [Shi and Malik, 1997], is that these algorithms are very time consuming, which prohibits their implementation for real-time applications.

Our method is related to the work in [Felzenszwalb and Huttenlocher, 2004], [Vergés-Llahi et al., 2000], [Guigues et al., 2003] in the sense of pairwise comparison of region similarity. It also similar to [Mathieu et al., 1992] in the sense that the MST is built. Our method differs from this method, because it builds the MST during the process, and not as the first step of the

algorithm as it is done in [Mathieu et al., 1992]. We create a hierarchy of attributed graphs. At each level of the pyramid a region adjacency graph (RAG) is created, in an agglomerative way (by contraction) with the proper topology. A vertex of the RAG is a representative of the region in the base level (receptive field), and it is created by taking into consideration the self-similarity of the region.

## 6.2 Minimum Weight Spanning Tree

The minimum spanning tree, called afterward MST, is the simplest and best-studied optimization problem in computer science. According to [Neštřil, 1997] the

> *Minimum spanning tree is a cornerstone problem of combinatorial optimization and in a sense its cradle.*

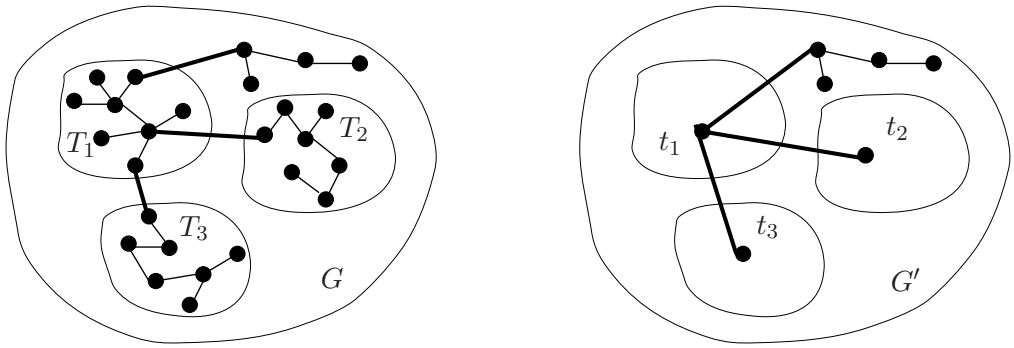
The problem is defined as follows. Let $G = (V, E)$ be the undirected connected plane graph consisting of the finite set of vertices $V$ and the finite set of edges $E$. Each edge $e \in E$ is identified with a pair of vertices $v_i, v_j \in V$ such that $v_i \neq v_j$. Let each edge $e \in E$ be associated with *unique* weight $w(e) = w(v_i, v_j)$, from the totally ordered universe (assumed that weight are distinct, if not, ties can be broken arbitrarily). Note that parallel edges, for e.g. $e_1 = (v_1, v_2)$ and $e_2 = (v_1, v_2)$ $e_1 \neq e_2$, have different weights. Self-loops are not encountered, since this would produce a cycle contradicting the definition of the tree. The problem is formulated as construction of a minimum weight spanning tree of $G$. Here we will give two lemmas that provide the basis of the minimum weight spanning tree algorithms, taken from [Thulasiraman and Swamy, 1992], which will help us in proving the correctness of our MST algorithm, afterward. The *weight of the subgraph* of $G$ is the sum of edge weights of subgraph, i.e. for $T \subseteq G$, the weight of a subgraph is :

$$w(T) = \sum_{e \in T} w(e). \tag{6.1}$$

**Theorem 6.1** *Consider a vertex $v$ in a weighted connected graph $G$. Among all the edges incident on $v$, let $e$ be one of minimum weight. Then, $G$ has a minimum weight spanning tree that contains e.*

*Proof:* Let $T_{min}$ be a minimum weighted spanning tree of $G$. If $T_{min}$ does not contain $e$, then consider the fundamental circuit $C$ of $T_{min}$ with respect to $e$, i.e. created by adding $e$ to $T_{min}$. Let $e'$ be the edge of $C$ that is adjacent to $e$ and incident to the same vertex $v$. Clearly $e' \in T_{min}$. Also $T' = T_{min} - e' + e$ is a spanning tree of $G$. Since $e$ and $e'$ are both incident on $v$, we have $w(e) \leq w(e')$, by assumption. But from the fact that $T_{min}$ is MST $w(T') = w(T_{min}) - w(e') + w(e) \geq w(T_{min})$ follows $w(e) \geq w(e')$. So, $w(e) = w(e')$ and $w(T') = w(T_{min})$. Thus $T'$ is also a minimum weighted spanning tree. The theorem follows since $T'$ contains $e$. □

**Theorem 6.2** *Let $T$ be an acyclic subgraph of a weighted connected graph $G$ such that there exists a minimum weight spanning tree containing $T$. If $G'$ denotes the graph obtained by contracting all the edges of $T$, and $T'_{min}$ is a minimum weight spanning tree of $G'$, then $T'_{min} \cup T$ is a minimum weight spanning tree of $G$.*




(a) $T_{min} = T \cup T'$, where $T = T_1 \cup T_2 \cup T_3$ (b) $T'$ after contraction of all edges in $T$

**Figure 6.1:** The contraction of sub-trees.

*Proof:* Let $T_{min}$ be a minimum weight spanning tree of $G$ containing $T$. If $T_{min} = T \cup T'$ (Figure 6.1a, then clearly $T'$ is a spanning tree of $G'$ (Figure 6.1b). Therefore,

$$w(T') \geq w(T'_{min}). \tag{6.2}$$

Since $T$ is an acyclic subgraph of $G$ and $T'_{min}$ is a minimum spanning tree of $G'$, it is easy to see that $T'_{min} \cup T$ is also a spanning tree of $G$. So,

$$\begin{array}{rcl} w(T'_{min} \cup T) & \geq & w(T_{min}) \\ & = & w(T) + w(T'). \end{array} \tag{6.3}$$

From this we get

$$w(T'_{min}) \geq w(T'). \tag{6.4}$$

Combining Equations (6.2) and (6.4) we get $w(T'_{min}) = w(T)$, and so $w(T'_{min} \cup T) = w(T' \cup T) = w(T_{min})$. Thus $T'_{min} \cup T$ is a minimum spanning tree of $G$. □

Note that these theorems are derived from the well known *strong cut property* of the minimum spanning trees, which is formulated as

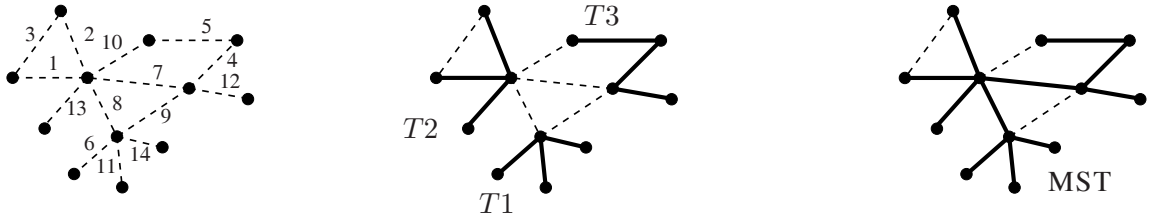
- $e \in$ MST $\Leftrightarrow$ $e$ is the lightest edge across some cut of $G$.

which is complementary to the *strong cycle property*

- $e \notin$ MST $\Leftrightarrow$ $e$ is the heaviest edge across some cycle of $G$.

All the algorithms presented below are based on the *strong cut property* of the graph $G$, since they try to connect the lightest edge to the MST tree.

We intend to solve the problem of MST in by using Borůvka's algorithm in conjunction with dual graph contraction [Kropatsch, 1995a]. We use Borůvka's algorithm since it can be used to build MST in parallel. In the next Section 6.2.1 we give the parallel version of this algorithm.




(a) Input graph (b) Edges with minimum weight (solid lines) (c) MST (solid line)

**Figure 6.2:** The steps of building MST with Borůvka's algorithm.

### 6.2.1 Borůvka's Algorithm

The idea of the Borůvka [Borůvka, 1926a], [Borůvka, 1926b] is to do steps like in Prim-Jarnik's algorithm, in parallel over the graph at the same time. This algorithm constructs a spanning tree in iterations composed of the steps shown in Algorithm 8.

**Algorithm 8 – Borůvka's Algorithm**

*Input*: graph $G = (V, E)$

1: MST$\leftarrow$ empty edge list
2: all vertices $v \in V$ make a list of trees $L$
3: **while** there is more than one tree in $L$ **do**
4: each tree $T \in L$ finds the edge $e$ with the minimum weight which connects $T$ to $G \setminus T$ and add edge $e$ to MST.
5: using edge $e$ merge pairs of trees in $L$
6: clean the graph from self-loops if necessary
7: **end while**

*Output*: minimum weight spanning tree - edge induced subgraph on MST.

First create a list $L$ of trees, each a single vertex $v \in V$. For each tree $T$ of $L$ find the edge $e$ with the *smallest weight*, which connects $T$ to $G \setminus T$. The trees $T$ are then connected to $G \setminus T$ with the edge $e$. In this way the number of trees in $L$ is reduced, until there is only one, the minimum weight spanning tree. A simple example of steps of Borůvka's algorithm is given in Figure 6.2; (a) the simple attributed graph as input, where each vertex is a tree in $L$; (b) the edges with the smallest weight (solid lines) incident on vertices, connect the trees (vertices in this case) to each other creating the trees $T1$, $T2$ and $T3$; (c) each tree in b) finds the edge with the smallest weight to connect to the other tree. The output shown in Figure 6.2c is the MST of the input graph. Steps $4$, $5$, and $6$ are called Borůvka phase.

The proof that this algorithm builds the minimum weight spanning tree is based on the proof of the Kruskal's MST algorithms given in [Thulasiraman and Swamy, 1992].

**Theorem 6.3** *Borůvka's algorithm constructs a minimum weight spanning tree of a weighted connected graph $G$.*

*Proof:* Let $G$ be the given nontrivial weighted connected graph. When Borůvka's algorithm terminates the selected tree $T_{min}$ is a spanning tree. Thus we have to show that $T_{min}$ is indeed a minimum weight spanning tree of $G$ by proving that every $T_i$ constructed in the course of Borůvka's algorithm is contained in a minimum weight spanning tree of $G$. Our proof is by induction on $i$. The subgraph $T_{i+1}$ is constructed from $T_i$ by adding an edge of minimum weight with exactly one end vertex in $T_i$. This construction ensures that all $T_i$'s are connected. As inductive hypothesis assume that $T_i$ is contained in a minimum spanning tree of $G$. If $G'$ denotes the graph obtained by contracting the edges of $T_i$ and $v'$ denotes the vertex of $G'$, which corresponds to the vertex set of $T_i$, then $e_{i+1}$ is in fact a minimum weight edge incident on $v'$ in $G'$. Clearly by Theorem 6.1 the edge $e_{i+1}$ is contained in a minimum weight spanning tree $T'_{min}$ of $G'$. By Theorem 6.2, $T_i \cup T'_{min}$ is a minimum weight spanning tree of $G$. More specifically $T_{i+1} = T_i \cup \{e_{i+1}\}$ is contained in a minimum weight spanning tree of $G$ and the correctness of Borůvka's algorithm follows. □

It can be easily shown that Algorithm 8 may fail to build MST if the edge weights are not distinct, since the set of selected edges may contain cycles. This problem can be solved as follows. If there are several edges of minimal weight that touch a vertex $v$, then choose among them the edge with the smallest random number. This random numbers should be all distinct. If the merging step is implemented in $\mathcal{O}(|E|)$ time, the overall algorithmic complexity is $\mathcal{O}(|E| \log |V|)$. In practice the algorithm converges faster. In the case of planar graphs, i.e. sparse graphs or dense graphs the algorithms runs in $\mathcal{O}(|V|)$ [Atallah, 1999, Chapter 6].

**Observation 6.1** *In $3^{rd}$ step of the Algorithm 8, each tree $T \in L$ finds the edge with the minimal weight, and as trees become larger, the process of finding the edge with the minimal weight for each tree $T$ takes longer.*

For completion of the presentation the other two classical algorithms to build a minimum spanning tree are given, Kruskal's and Prim-Jarnik's algorithm. Kruskal's algorithm is used as a basis of the segmentation method presented in Chapter 7. The proofs that these algorithms build minimum spanning trees are found in [Thulasiraman and Swamy, 1992].

### 6.2.2 Kruskal's Algorithm

The Kruskal's [Kruskal, 1956] approach is proceeded by a sorting of the weight of edges in a non decreasing order. Every edge from this sorting is included in the tree as long as it does not create a cycle in the already created tree. More formally the steps are shown in Algorithm 9. This algorithm has computational complexity of $\mathcal{O}(|E| \log |E|)$, because of the sorting step, and can be reduced into $\mathcal{O}(|E| \log |V|)$, if one uses more efficient sorting algorithm like heap sort [Atallah, 1999].

### 6.2.3 Prim-Jarniks's Algorithm

The Prim-Jarnik's algorithm [Prim, 1957] [Jarník, 1930] consist in construction of a sequence of trees say $T_i$ such that $T_{i+1}$ is constructed from $T_i$ by adding an edge $e$ with the minimum

**Algorithm 9 – Kruskal's Algorithm**

*Input*: graph $G = (V, E)$

1: make MST$\leftarrow$ empty edge list
2: sort edges of $G$, by nondecreasing order
3: **while** for all edges $e \in E$ in sorted order **do**
4: select $e \in E$ of minimal weight such that $e \notin$ MST
5: put edge $e$ in MST such that it does not create a cycle
6: remove edge $e$ from $E$
7: **end while**

*Output*: Minimum weight spanning tree - edge induced subgraph on MST.

**Algorithm 10 – Prim-Jarniks's Algorithm**

*Input*: graph $G = (V, E)$

1: make MST$\leftarrow$ empty edge list, $W$ an empty vertex list
2: select any vertex $v$ and set $W \leftarrow \{v\}$
3: **while** there is more than one vertex in $V$ **do**
4: select an edge $e = (u, w)$ with minimum weight such that one its of vertices say $u$ is in $W$
5: put edge $e$ in MST, add $u$ in $W$ and remove $e$ from $E$
6: **end while**

*Output*: minimum weight spanning tree - edge induced subgraph on MST.

weight, which has one end vertex in $T_i$. The steps are shown in Algorithm 10. This algorithm has computational complexity of $\mathcal{O}(|E| + |V| \log |V|)$ using Fibonacci heaps, $\mathcal{O}(|V|^2)$ using arrays, and $\mathcal{O}(|E| \log |V|)$ using binary heaps [Atallah, 1999].

### 6.2.4 Related Works

In the sections above the three text book algorithms for solving the minimum spanning tree problem are given, the Borůvka's, Kruskal's, and Prim-Jarnik's approaches. All these algorithms give a deterministic solution to the MST problem and are widely known in the literature as 'generalized greedy algorithms' [Gabow et al., 1986]. These text book algorithms run in $\mathcal{O}(|E| \log |V|)$. A better performance is achieved by [Yao, 1975] and [Cheriton and Tarjan, 1976] to $\mathcal{O}(|E| \log \log |V|)$. With the Fibonacci heaps [Fredman and Robert, 1987] reduced the complexity to $\mathcal{O}(|E|\beta(|E|, |V|))$, where $\beta(|E|, |V|) = \min\{i | \log^{(i)}(|V|) \leq |E|/|V|\}$. [3] In [Gabow et al., 1986] the solution of MST is improved to $\mathcal{O}(|E| \log \beta(|E|, |V|))$. The algorithm proposed in [Chazelle, 2000] based on Borůvka's approach is the fastest known present. In this proposal the graph $G$ is first decomposed into vertex-disjoint contractible subgraphs of suitable size, called contractions kernels. Each of this subgraphs are contracted into a single vertex, and thus a minor of $G$ is formed, the process of forming contractible subgraphs is

[3] Where $\log^{(i)} |V|$ is $\underbrace{\log(\log(..(\log(|V|))..))}_{i-times}$.

continued until $G$ becomes a single vertex. This forms a hierarchy of contractible subgraphs, which can be modeled by perfect balanced tree $\mathcal{T}$, where its leaves represent the vertices of $G$ (level 0), the root correspond to the whole graph $G$ (apex of the tree). An internal vertex $v$ with children $\{v_{v_i}\}$ is associated with the graph $N_v$, whose vertices are the contraction of graphs $\{N_{v_i}\}$, i.e. of graphs in a level below. Each level of $\mathcal{T}$ represents a minor of $G$. Once the $\mathcal{T}$ is found the MST of each contraction kernel is found, recursively. The MST of $G$ is computed by joining together the trees of MST of the contraction kernels. In [Chazelle, 2000] the best possible trade-off between the size of the contraction kernels and the height of the hierarchy is found such that the algorithms has the complexity of $\mathcal{O}(|E|\alpha(|E|,|V|))$, where $\alpha(|E|,|V|) = \min\{i \geq 1 | A(i, 4\lceil |E|/|V| \rceil > \log |V|\}$ is the inverse Ackerman's function[4] defined in [Tarjan, 1975]. In [Fredman and Dan, 1994] an algorithm is given with the linear worst case time solution under the assumption that weights are small integers. The ultimate goal is to find a deterministic algorithm with linear complexity, i.e. $\mathcal{O}(|E|)$, under no restriction.

A non-deterministic solution the MST problem was proposed in [Karger et al., 1995] with a randomized algorithm, which solves the problem in linear expected time. This algorithm need an algorithm to verify that the result is MST. The first linear time verification algorithm is given in [Dixon et al., 1992]. A simpler linear time verification algorithm is proposed in [King, 1997]. Both [King, 1997] and [Karger et al., 1995] make use of the Borůvka's algorithm.

The MST problem has been solved using parallel machines as well. In [Hirschberg et al., 1979] gave a parallel deterministic algorithm on CREW PRAM [5] with time complexity $\mathcal{O}(\log^2 |V|)$ and $|V|^2/\log|V|$ number of processors. In [Chin et al., 1982] the results are improved into $\mathcal{O}(\log^2 |V|)$ time complexity and $|V|^2/\log^2|V|$ needed processors. On the CRCW PRAM model the authors [Shiloach and Vishkin, 1982] gave an algorithm with time complexity $\mathcal{O}(\log|V|)$ and $\mathcal{O}(|E|+|V|)$ processors, and it was further improved by [Cole et al., 1996] needing only $\mathcal{O}((|E|+|V|)\alpha(|E|,|V|)/\log|V|)$ processors. The logarithmic time linear work randomized algorithm is proposed in [Cole et al., 1996], improving the results of [Cole et al., 1994] from $\mathcal{O}(2^{\log^*|V|}\log|V|)$ to $\mathcal{O}(\log|V|)$ time complexity. This works is based on the work of [Karger et al., 1995].

The single-link clustering technique, used in statistics to cluster data points into coherent groups is essentially MST [Duda et al., 2001]. The nearest-neighbor algorithm could be viewed as an algorithm for building an MST of the data [Duda et al., 2001]. In [Chowdhury and Murhty, 1997] it was shown the MST approaches the performance of the Bayes classifier as the number of data point goes to infinity, and an example of applying MST to cluster real world data are shown in [Päivinen, 2005].

[4]The Ackerman function is defined as:

$$A(i,j) = \begin{cases} A(0,j) = 2j, & \text{for any} \quad j \geq 0; \\ A(i,0) = 0 & \text{and} \quad A(i,1) = 2 \quad \text{for any} \quad i \geq 1; \\ A(i,j) = A(i-1, A(i,j-1)), & \text{for any} \quad i \geq 1, j \geq 2. \end{cases}$$

[5]C-concurrent, E-exclusive, R-read, W-write, PRAM-parallel random access machine.

### 6.2.5 Minimum Spanning Tree as Matroid Optimization Problem

In this section the connections of matroids and MST is shown. [Glantz and Kropatsch, 2000a] shows that graph pyramids can be represented as a basis of matroid. The study of matroids[6] arises from the need to capture fundamental properties of dependence in graphs and matrices. One can think of matroids as an abstraction that generalizes both graphs and vector spaces. A matroid $M$ is an ordered pair $(E, \mathcal{I})$ consisting of a finite set $E$ and a collection $\mathcal{I} \subset E$ that satisfies these conditions [Oxley, 1992]:

(I1). $\emptyset \in \mathcal{I}$,

(I2). every subset of every element of $\mathcal{I}$ is also in $\mathcal{I}$, and

(I3). if $I_1$ and $I_2$ are in $\mathcal{I}$ and $|I_1| = |I_2| + 1$, then there is an element $e$ in $I_1 - I_2$ such that $I_2 \cup e \in \mathcal{I}$.

Condition (I2) is called the hereditary axiom, and condition (I3) is called independence augmentation axiom. If $M$ is the matroid $(E, \mathcal{I})$, then $M$ is matroid on $E$. Elements of $\mathcal{I}$ are independent sets of $M$, and $E$ is the ground set of $M$.

Matroids arise in a number of combinatorial optimizations, among others also in minimum spanning tree problem. An overview of many optimization problems that are special cases of matroid problems are found in [Bixby and Cunnigham, 1995]. In this section a tight relation of the greedy algorithm and the matroids is given. Note that the greedy algorithm is used to solve the minimum spanning tree problem in [Borůvka, 1926a, Jarník, 1930, Kruskal, 1956, Prim, 1957]. It is shown that the greedy algorithm is the one that solves a weight matroid optimization problem. A graph $G = (V, E)$ can be characterized as a matroid where the set of edges $E$ and a set of edges $E'$ is independent if and only if it does not contain a cycle, or equivalently the subgraph induced by $E'$ is a forest. Since the spanning trees of this forest are a maximally independent set, this sets are called bases. This kind of matroid is called a cycle matroid or *graphic matroid*. If the edges of the graph are weighted then it is called *weighted matroid*.

The minimum spanning tree is a special case of the weighted matroid optimization problem [Oxley, 1992]. Let $\mathcal{I}$ be the subset of the edge set $E$ which satisfies conditions (I1) and (I2), and edges in $E$ are weighted $w : E \rightarrow \mathbb{R}$. Let the weight of a non-empty set $T$ of $E$ be defined as $w(T) = \sum_{e \in T} w(e)$ and $w(\emptyset) = 0$. The optimization problem for the pair $(\mathcal{I}, w)$ is as follows [Oxley, 1992]:

$$\textit{Find a maximal member } T \textit{ of } \mathcal{I} \textit{ of maximum weight.} \tag{6.5}$$

The set $T$ is the solution of this optimization problem. If the weight function is $w' = W_0 - w$, where $W_0$ is larger then the largest weight $w(e)$, and solve the optimization problem $(\mathcal{I}, w)$, then $T'$ is the maximal member of $\mathcal{I}$ for which $w(T')$ is minimal, thus solving the minimal spanning tree problem, as a special case of Problem 6.5. The simple greedy algorithm which begins with empty independent set $X$, and consider the matroid elements in order of increasing weight, adds each element to $I$ if by adding $I$ is kept independent. The result of this greedy algorithm yields the Jarnik-Kruskal's algorithm.

The set $(E, \mathcal{I})$ is a matroid if the conditions (I1) and (I2) are satisfied together with the condition

[6] Called also combinatorial geometry.

(G) For all weight functions $w : E \rightarrow \mathbb{R}$, the greedy algorithm produces a maximal member of $\mathcal{I}$ of maximum weight.

The prove that $(E, \mathcal{I})$ with conditions (I1), (I2), and (G) is a matroid is given in [Oxley, 1992, page 64], i.e. greedy algorithm works for matroid. In Chapter 2, Section 2.7 was shown that graphs can be represented over the Galois fields $\mathbb{F}_2$, therefore matroids can be represented over the same fields $\mathbb{F}_2$ as well. In this section the relation of matroids with MST is shown, thus graph pyramids build on the MST principle could be characterized as a special case of the weighted matroid optimization problem, as well.

## 6.3 Minimum Spanning Tree with DGC

Taking the Observation 6.1 into consideration, a contraction of the edge $e$, which connects $T$ and $G \setminus T$ in the $4^{th}$ step of Algorithm 8 will speed up the process of searching for minimum weight edges in the Borůvka's algorithm. If the graphs are represented as adjacency list then a vertex with degree $d$ can enumerate its incident edges in its neighborhood in time $\mathcal{O}(d)$. Since each tree (in level $k$) after edge contraction will be represented by a vertex (in the level $k+1$), the search for the edge with the minimum weight would be a local search, and the resulting graph is smaller (in the sense of less vertices and less edges), thus the next pass can run faster. Note that introduction of the edge contraction method introduces naturally the hierarchy of graphs. [Kropatsch, 1995a] and [Dey et al., 1998] introduce a topology preserving edge contraction.

The dual graph contraction algorithm [Kropatsch, 1994], [Kropatsch, 1995a] is used to contract edges and create *super* vertices i.e. to create father-son relation between vertices in subsequent levels (vertical relation) and at the same time to preserve the topology, whereas Borůvka's algorithm is used to create son-son relation between vertices in the same level (horizontal relation). In Section 6.4 we will refine the son-son relation to simulate the pop-out phenomena [Julesz, 1981], [Julesz and Bergen, 1983], and to find region borders quickly and effortlessly in a bottom-up 'stimulus-driven' based on local differences in a specific feature (color).

Here we expand Borůvka's algorithm with the steps that contract edges, remove parallel edges and self loops (if the topology is not changed), see Algorithm 11.

**Theorem 6.4** *The equivalent contraction kernel of the apex of the graph pyramid constructed by Algorithm 11 is the minimum spanning tree of the base level.*

*Proof:* The top $h$ of the pyramid, consists of an isolated vertex (apex). The set of edges at the base level (input graph) which are contracted so that the apex is derived, is the equivalent contraction kernel $N_{0,h}$ (see Section 4.4.2). The proof of this theorem is analogous with Theorem 6.3 Let $G_0$ be the given nontrivial weighted connected graph. When Algorithm 11 terminates the equivalent contraction kernel of the apex $h$ of the pyramid is a spanning tree. Thus we have to show that $N_{0,h}$ is indeed a minimum weight spanning tree of $G_0$ by proving that every equivalent contraction kernels of all vertices $v \in G_k$, $\forall k = h, ..., 1$ constructed in the course of the algorithm is contained in a minimum weight spanning tree of $G_0$. Our proof is by induction on $k$. The contraction kernel of vertex $v' \in G_{k+1}$, $N_{k,k+1}(v')$ is constructed from all its children $Ch(v) \in G_k$ such that for each children an edge of minimum weight is added (Step 3 of Algorithm 11). Note that every contraction kernel in $G_k$ is a tree, therefore connected.

**Algorithm 11 – Borůvka's Algorithm with DGC**

*Input*: attributed graph $G_0 = (V, E)$

1: $k \leftarrow 0$
2: **repeat**
3: for each vertex $v \in G_k$ find the minimum-weight edge $e \in G_k$ incident to the vertex $v$ and mark the edges $e$ to be contracted
4: determine $CC_i^k$ as the connected components of the marked edges $e$
5: contract connected components $CC_i^k$ in a single vertex and eliminate the parallel edges (except the one with the minimum weight) and self-loops and create the graph $G_{k+1} = C[G_k, CC_i^k]$
6: $k \leftarrow k + 1$
7: **until** all connected components of $G$ are contracted into one single vertex

*Output*: a graph pyramid with an apex.

As inductive hypothesis assume that $N_{k,k+1}$ is contained in a minimum spanning tree of $G_k$, i.e. $T_k$. If $G_{k+1}$ denotes the graph obtained by contracting the edges of $N_{k,k+1}$ into the vertices $v' \in G_{k+1}$, then minimum weight edges $e' \in G_{k+1}$ are incident on $v' \in G_{k+1}$. Since the edges of the graph $G_{k+1}$ correspond in fact to paths that connect trees in the level $G_k$ and based on Theorem 6.1 holds that $e'$ are contained in a minimum spanning tree of $G_k$. By Theorem 6.2, $N_{k,k+1} \cup e' = N_{k,k+2}$ (Definition 4.9) are contained in the minimum spanning tree of $G_k$. This holds for all $k$ and thus the proof that the $N_{0,h}$ is the minimum spanning tree of $G_0$. □

Each level of the pyramid represents a set of sub trees of the MST. In the first step we find the edges with the minimum weight incidented on every vertex $v \in G_k$. We contract the marked edges (in general a connected component) into a single vertex (the survivor, 'super vertex) using edge contraction and face contraction [Kropatsch, 1995a] (applying DGC), and produce the graph $G_{k+1}$. If there are parallel paths that connect the same trees they are removed except the one with the minimum-weight middle edge (see Figure 4.10). Self-loops are removed if they do not include a surviving vertex. We repeat this process until we arrive at the apex of the graph pyramid.

An example of building MST using DGC for Figure 6.2 is given in Figure 6.3. The solid lines represent the marked edges (in general connected components) which are contracted by DGC to create the graph of the next level. The equivalent contraction kernel of the apex in level $G_3$, i.e. the spanning tree consisting of the edges which are contracted, is the **minimum spanning tree** of the graph $G_0$ as shown in Figure 6.3.

For a connected attributed graph the time complexity of the algorithm above is as follows

**Theorem 6.5** *Algorithm 11 finds the minimum spanning tree after at most* $\log |V|$ *iterations each of which involves* $\mathcal{O}(|E| + |V|)$ *operations.*

*Proof:* In the $3^{rd}$ step each vertex of $G_k$ chooses one edge that will be contracted in the $5^{th}$ step. Since the contraction of an edge also eliminates one of its incident end vertices the number of vertices decreases by a reduction factor of at least 2 at every iteration. Thus, the number of iterations cannot exceed $\log_2 |V|$. The selection of $CC$ can be performed in $\mathcal{O}(|V|)$

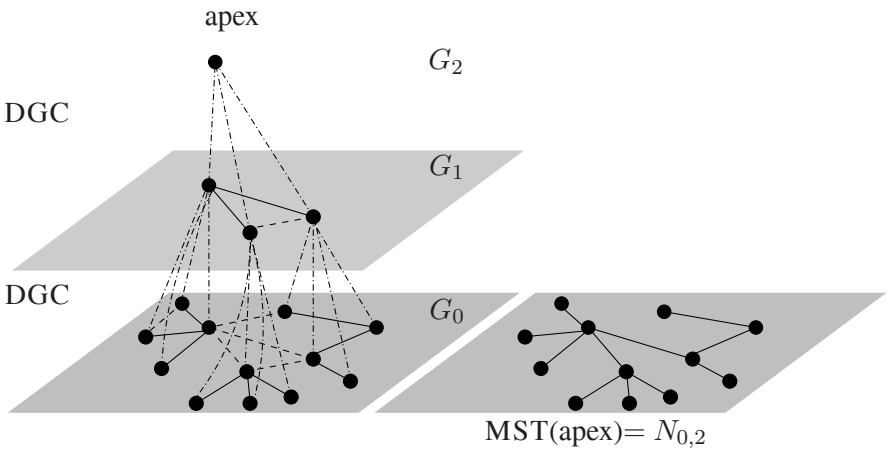



**Figure 6.3:** Building MST with DGC.

and contraction of edges in $\mathcal{O}(|E|)$. Hence, the overall running time cannot exceed $\mathcal{O}((|E| + |V|) \log_2 |V|)$. □

## 6.4 Hierarchy of Partitions

The segmentation problem is supposed to find natural groupings from the pixel set. The first question that comes in mind is how this natural groupings are found. In other words what makes pixels in a partition to be more like one another than pixels in other segments. This observation pours down into two issues [Duda et al., 2001]:

- how to measure the similarity between pixels, and
- how to evaluate a partitioning of the pixels into segments.

In Section 6.5 different similarity (dissimilarity) measures between pixels are given, all of them are metric measures. It is expected that, these measures of dissimilarity captures the expectation that the distance of pixels within segment is less than the distance between pixels in different segments. The second issue is defining the criterion function to be optimized. The goal is to find the groups or segments that have strong internal similarities, which optimize the criterion function.

Let $G = (V, E)$ be a given attributed graph with the vertex set $V$ and edge set $E$. We will discuss later about the attributes. The goal is to find partitions $P = \{CC_1, CC_2, ..., CC_n\}$ such that these elements are disjoint and satisfy certain properties. Moreover $P$ is a partition of $V \in G$, and

$$\forall i \neq j \quad CC_i \cap CC_j = \phi \quad \wedge \quad \bigcup_{i=1,...,n} CC_i = V. \tag{6.6}$$

In [Horowitz and Pavlidis, 1976] define a consistent homogeneity criteria over a set $V$ as a boolean predicate $P$ over its parts $\Phi(V)$ that verifies the consistency property:

$$\forall (x, y) \in \Phi(V) \quad x \subset y \Rightarrow (P(y) \Rightarrow P(x)). \tag{6.7}$$

In image analysis Equation 6.7 states that the subregions of homogeneous region are also homogeneous. It follows that if $\mathcal{P}$ is a hierarchy and $P$ a consistent homogeneity criteria on $V$ then the set of maximal elements of $\mathcal{P}$ that fulfill $P$ defines a unique partition of $V$. Thus the joint use of hierarchy and homogeneity criteria allow to define a partitioning in a natural way.

**Definition 6.1 (Hierarchy)** *Given a set $G$, a set $\mathcal{P}(G)$ is a hierarchy if for any $A, B \in \mathcal{P}$ such that $A \cap B \neq \emptyset$ implies that $A \subset B$ or $B \subset A$.*

The hierarchical partitioning are the best known unsupervised methods. In general the agglomerative (bottom-up, merging) procedure starts with singletons (e.g. pixels) and forms the new partitions by successive merging. The divisive (top-down, splitting) procedure starts with a single partitions and the new partitions are created by successive splitting old partitions.

The algorithm for hierarchical partitioning in general is given in Algorithm 12. The algorithm terminates when the number of partitions reach $c$. If $c = 1$ the partitions are merged until there is only one partition, i.e. a hierarchy of partitions with an apex is created.

**Algorithm 12 – Agglomerative Hierarchical Partitioning Algorithm**

*Input*: graph $G = (V, E)$

1: each vertex is a partition $CC_i \, \forall i = 1, \ldots, |V|$
2: **repeat**
3: find closest partition, say $CC_i$ and $CC_j$
4: agglomerate $CC_i$ and $CC_j$
5: **until** there are $c$ partition

*Output*: a hierarchy of partitions.

The Algorithm 12 in line 3 needs a definition of the 'closness' between two partitions. If the dissimilarity between two partitions is defined by[7]

$$\delta_{\min}(CC_i, CC_j) = \min_{u \in CC_i, u' \in CC_j} \delta(u, u'). \tag{6.8}$$

then the Algorithm 12 will induce a distance function for the given starting partitions. It can be shown that a distance $d(u, u')$ between $u, u'$ defined as the lowest level in the hierarchy for which $u \; u'$ are in the same partitions, form a metric [Duda et al., 2001].

Note that if the definition of dissimilarity is defined as:

$$\delta_{\min}(CC_i, CC_j) = \min_{u \in CC_i, u' \in CC_j} \|u - u'\|. \tag{6.9}$$

the Algorithm 12 is called *nearest-neighbor cluster algorithm*, or *minimum algorithm*. If the algorithm stops when the distance between nearest partitions exceeds an arbitrary threshold, it is called *single-linkage algorithm* [Duda et al., 2001]. The usage of the $\delta_{\min}(\cdot, \cdot)$ as the measure of dissimilarity means that the nearest partitions are found by nearest vertices. The merging of $CC_i$ and $CC_j$ is in fact joining by an edge these partitions, and the resulting graph will not

[7] Or by the $\max$ function.

contain closed loops or circuits. If the algorithm continuous until all the partitions are joint, i.e. all the vertices are connected with a path, it can be shown that the sum of edge lengths of the resulting tree is minimal. Therefore, if $\delta_{\min}(\cdot,\cdot)$ is used as the distance measure, the agglomerative algorithms is an algorithm for creating a minimum spanning tree [Duda et al., 2001], i.e. the Algorithm 12 can be used to solve the optimization problem of the minimum spanning tree. In Section 6.2.5 it was shown that a greedy algorithm is the solution to the minimum spanning tree problem. On contrary, if the distance measure is defined as:

$$\delta_{\max}(CC_i, CC_j) = \max_{u \in CC_i,\, u' \in CC_j} \|u - u'\|. \tag{6.10}$$

the Algorithm 12 is called *farthest-neighbor cluster algorithm* or *maximum algorithm*. If the algorithm stops when the distance between nearest partitions trespass an arbitrary threshold, it is called *complete-linkage algorithm* [Duda et al., 2001]. Posed into graph theory this algorithm produce clusters in which all the vertices are connected to each other, i.e. create a *complete subgraph* or *clique*.

Even thought the definition in Equation 6.8 is very sensitive to noise, the so called 'chaining effect', this definition of the nearness between two partitions is used as basis to define more precisely the dissimilarity measure in the next section.

### 6.4.1 Building a Hierarchy of Partitions

The pairwise comparison of neighboring vertices, i.e. partitions is used to check for similarities [Felzenszwalb and Huttenlocher, 2004], [Vergés-Llahi et al., 2000], [Fischer and Buhmann, 2002], [Guigues et al., 2003]. In [Felzenszwalb and Huttenlocher, 1998], a definition of a pairwise group comparison function $Comp(CC_i, CC_j)$ is given that judges whether or not there is evidence for a boundary between two partitions $CC_i, CC_j \in P$. Definition of $Comp(CC_i, CC_j)$ depends on the application. $Comp(CC_i, CC_j)$ is true, if there is evidence for a boundary between $CC_i$ and $CC_j$, and false when there is no boundary. This function measures the difference along the boundary of two components relative to a measure of differences of components' internal differences. This definition tries to encapsulate the intuitive notion of contrast: a contrasted zone is a region containing two connected components whose inner differences (*internal contrast*) are less than differences within it's context (*external contrast*). We define an external contrast measure between two components and an internal contrast measure of each component.

Let $G = (V, E, attr_v, attr_e)$ be a given attributed graph with the vertex set $V$ and edge set $E$ on the base level (level 0). Vertices $v \in V$ and edges $e \in E$ are attributed, i.e. $attr_v : V \to \mathbb{R}^+$ and $attr_e : E \to \mathbb{R}^+$. One possible way to attribute the edges is given in Section 6.5. The graph on level $k$ of the pyramid is denoted by $G_k$. Every vertex $u \in G_k$ is a representative of a component $CC_i$ of the partition $P_k$. The equivalent contraction kernel of a vertex $u \in G_k$, $N_{0,k}(u)$ is e set of edges of the base level $e \in E$ that are contracted; i.e. applying equivalent contraction kernel on the base level, one contracts the subgraph $G' \subseteq G$ onto the vertex $u$ (see Section 4.4.2 for more details).

The **internal contrast** measure of the $CC_i \in P_k$ is the **largest dissimilarity** measure of component $CC_i$ i.e. the largest edge weight of the $N_{0,k}(u)$ of vertex $u \in G_k$:

$$Int(CC_i) = \max\{attr_e(e), e \in N_{0,k}(u)\}. \tag{6.11}$$

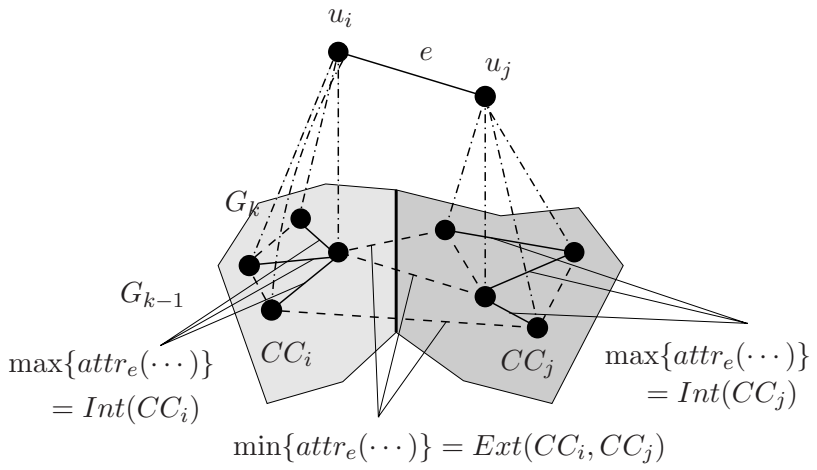



**Figure 6.4:** Internal and External contrast.

Let $u_i, u_j \in V_k$ be the end vertices of an edge $e \in E_k$. The **external contrast** measure between two components $CC_i, CC_j \in P_k$ is the **smallest dissimilarity** measure between component $CC_i$ and $CC_j$ i.e. the smallest edge weight connecting $N_{0,k}(u_i)$ and $N_{0,k}(u_j)$ of vertices $u_i \in CC_i$ and $u_j \in CC_j$:

$$Ext(CC_i, CC_j) = \min\{attr_e(e), e = (v, w) : v \in N_{0,k}(u_i) \wedge w \in N_{0,k}(u_j)\}. \tag{6.12}$$

In Figure 6.4, a simple example of $Int(CC_i)$ and $Ext(CC_i, CC_j)$ is given. The $Int(CC_i)$ ($Int(CC_j)$) of the component $CC_i$ ($CC_j$) is the $maximum$ of weights of the solid line edges, whereas $Ext(CC_i, CC_j)$ is the $minimum$ of weights of the dashed line edges (bridges) connecting component $CC_i$ and $CC_j$ on the base level $G_0$. Vertices $u_i$ and $u_j$ are representative of the components $CC_i$ and $CC_j$. By contracting the edges $N_{0,k}(u_i)$ one arrives to the vertex $u_i$, analogously $N_{0,k}(u_j)$ for the vertex $u_j$ (see Section 4.4.2).

The pairwise comparison function $Comp(\cdot, \cdot)$ between two connected components $CC_i$ and $CC_j$ can now be defined as:

$$Comp(CC_i, CC_j) = \begin{cases} \text{True} & \text{if } Ext(CC_i, CC_j) > PInt(CC_i, CC_j), \\ \text{False} & \text{otherwise,} \end{cases} \tag{6.13}$$

where $PInt(CC_i, CC_j)$ is the minimum internal contrast difference between two components:

$$PInt(CC_i, CC_j) = \min(Int(CC_i) + \tau(CC_i), Int(CC_j) + \tau(CC_j)). \tag{6.14}$$

For the function $Comp(CC_i, CC_j)$ to be true i.e. for the border to exist, the external contrast difference must be greater than the internal contrast differences. Note that $Comp(CC_i, CC_j)$ is a boolean comparison function for pairs of partitions and the resulted segmentation is the so called *crisp* segmentation. The reason for using a threshold function $\tau(CC)$ in Equation (6.14) is that for small components $CC$, $Int(CC)$ is not a good estimate of the local characteristics of the data, in extreme case when $|CC| = 1$, $Int(CC) = 0$. Any non-negative function of a single component $CC$, can be used for $\tau(CC)$ [Felzenszwalb and Huttenlocher, 1998]. One can define $\tau$ to be function of the size of $CC$:

$$\tau(CC) = \alpha/|CC|, \tag{6.15}$$

where $|CC|$ denotes the size of the component $CC$ and $\alpha$ is a constant. More complex definition of $\tau(CC)$, which is large for certain shapes and small otherwise would produce a partitioning which prefers certain shapes, e.g. using ratio of perimeter to area would prefer components that are not long and thin.

### 6.4.2 Constructing a Hierarchy of Partitions

The Algorithm 11 in Section 6.3 in conjunction with the comparison function $Comp(\cdot,\cdot)$ defined in Section 6.4.1 is used as basis to build the hierarchy of partitions. We proved that Algorithm 11 builds a minimum spanning tree of an attributed graph, so the definition of internal and external contrast are now recalled for clarity:

$$\begin{gathered} Int(CC^k) = \max\{attr_e(e), e \in \text{MST}(u_k)\}. \\ Ext(CC_i^k, CC_j^k) = \min\{attr_e(e), e = (v,w) : v \in \text{MST}(u_{k,i}) \wedge w \in \text{MST}(u_{k,j})\}. \\ PInt(CC_i^k, CC_j^k) = \min(Int(CC_i^k) + \tau(CC_i^k), Int(CC_j^k) + \tau(CC_j^k)). \\ \tau(CC^k) = f(CC^k), \end{gathered} \tag{6.16}$$

where $f(CC^k)$ is non-negative function, not defined yet.

Let $P_k = CC_i^k, CC_j^k, ..., CC_n^k$ be the partitions on the level $k$ of the pyramid i.e $P_k$ is the graph $G_k(V_k, E_k)$. The algorithm to build the hierarchy of partitions is as follows:

**Algorithm 13 – Construct Hierarchy of Partitions (BorůSeg)**

*Input*: attributed graph $G_0$.

1: $k \leftarrow 0$
2: **repeat**
3: **for all** vertices $u \in G_k$ **do**
4: $E_{min}(u) \leftarrow argmin\{attr_e(e) \,|\, e = (u,v) \in E_k \text{ or } e = (v,u) \in E_k\}$
5: $E_{min} = E_{min} \cup E_{min}(u)$
6: **end for**
7: **for all** $e = (u_{k,i}, u_{k,j}) \in E_{min}$ with $Ext(CC_i^k, CC_j^k) \leq PInt(CC_i^k, CC_j^k)$ **do**
8: include edge $e$ in contraction edges $N_{k,k+1}$
9: **end for**
10: contract graph $G_k$ with contraction kernels, $N_{k,k+1}$: $G_{k+1} \leftarrow C[G_k, N_{k,k+1}]$.
11: **for all** $e_{k+1} \in G_{k+1}$ **do**
12: set edge attributes $attr_e(e_{k+1}) \leftarrow \min\{attr_e(e_k) \,|\, e_{k+1} = C[e_k, N_{k,k+1}]\}$
13: **end for**
14: $k \leftarrow k + 1$
15: **until** $G_k = G_{k-1}$

*Output*: a region adjacency graph (RAG) at each level of the pyramid.

If we assume that the steps 6 to 8 of the Algorithm 13 are left out, it can be shown, that this algorithm produces a MST (Theorem 6.3). Each vertex $u_k \in G_k$ i.e. $CC^k$ represents a connected region on the base level of the pyramid, and since the presented algorithm is based

on Borůvka's algorithm [Borůvka, 1926a], it builds a MST$(u_k)$ of each region, i.e $N_{0,k}(u_k) =$ MST$(u_k)$ (see Theorem 6.4).

The idea is to collect smallest weighted edges $e$ ($4^{th}$ step) that could be part of MST, and then to check if the edge weight $attr_e(e)$ is smaller than the internal contrast of both of the components (MST of end vertices of $e$) ($6^{th}$ step), if these conditions are fulfilled then these two components will be merged ($7^{th}$ step). Two regions will be merged if the internal contrast, which is represented by its MST, is larger than the external contrast, represented by the weight of the edge, $attr_e(e)$. All the edges to be contracted form the contraction kernels $N_{k,k+1}$, which then are used to create the new graph $G_{k+1} = C[G_k, N_{k,k+1}]$ [Kropatsch et al., 1999]. Note that edges consisting $N_{k,k+1}$ have no cycles, i.e. having cycles will contradict step 4 of Algorithm 13. In general $N_{k,k+1}$ is a forest. We update the attributes of those edges $e_{k+1} \in G_{k+1}$ with the minimum attribute of the edges $e_k \in E_k$ that are contracted into $e_{k+1}$ ($11^{th}$ step). This means that we do not recompute the attributes of the edges but simple inherit it. In oder to make computation faster, one can store in each vertex of the new graph $Int(CC)$ and the size of the receptive field $RF$. Instead of computing $Int(CC^{k+1})$ and $RF$ from the base of the pyramid one can update them successively. This means that $\forall u_{k+1} \in G_{k+1}$ on the new level $k+1$, the $Int(CC(u_{k+1}))$ is computed as the $\max$ of the $Int(u_k),\ u_k \in Ch(u_{k+1}, u_k)$ and as $\max$ of all attributes on edges of the contraction kernel that is contracted into $u_{k+1}$. The receptive field of $u_{k+1}$ is the sum of all vertices $u_k \in G_k$ of its contraction kernel. The $Int()$ and $RF$ are set to zero for all vertices on the base level to start the algorithm.

The output of the algorithm is a pyramid where each level represents a RAG, i.e. a partition. Each vertex of these RAGs is the representative of a MST of a region in the image. In general the top of the pyramid consists of an apex, which represents the whole image. The algorithm is greedy since it collects only the nearest neighbor with the minimal edge weights and merges them if Equation 6.13 is false.

**Proposition 6.1** *For any connected attributed graph $G = (V, E, attr_e, attr_v)$ the set of regions fulfilling Equation 6.13 is a hierarchy. over $V$.*

*Proof:* Vertices $v \in V_0$ on the base level partition the base graph $G_0$. It is only needed to check that partitions are partially ordered by the inclusion relation (see Definition 6.1). Assume that for each pair of neighboring regions this is not the case, i.e. $\exists(CC_i^k, CC_j^k) \in P_k$, such that $CC_i^k \cap CC_j^k \neq \phi$ but neither $C_i^k \subset C_j^k$ nor $C_j^k \subset C_i^k$. There are at least two edges, $e'$ connecting $CC_i^k$ and $CC_j^k \setminus CC_i^k$ and the other edge $e''$ connecting $CC_j^k$ and $CC_i^k \setminus CC_j^k$, from which it follows

Let us consider $e'$ connecting $CC_i^k$ and $CC_j^k \setminus CC_i^k$ and the other edge $e''$ connecting $CC_j^k$ and $CC_i^k \setminus CC_j^k$, from which it follows

$$Ext(CC_i^k, CC_j^k) \leq \min(attr_e(e'), attr_e(e'')) \leq \min(Int(CC_i^k), Int(CC_j^k)) < PInt(CC_i^k, CC_j^k) \tag{6.17}$$

this implies that Equation 6.13 is false, i.e. $CC_i^k$ and $CC_j^k$ are one component.

□

**Proposition 6.2** *For any connected attributed graph $G = (V, E, attr_e, attr_v)$ Algorithm 13 produces partitions on each level which are invariant under any monotone transformation of*

*dissimilarity measure $attr_e$. Moreover the hierarchy over $V$ is invariant under monotone transformation.*

*Proof:* It should be verified that the order by which the edges are contracted is not changed by monotone transformation. The monotone transformation does not change the total order of edges incidented on a vertex. This implies that the edge with the minimum weight is also not changed by this monotone transformation in the $2^{nd}$ step of the Algorithm 13. Moreover this transformation does not change the total order of the edges in a connected component $CC_i^k$ and $CC_j^k$, implying that the minima of maximum weight edge of the $CC_i^k$ and $CC_j^k$ is on the same edge ($3^{rd}$ step). Edges marked in the $2^{nd}$ and $3^{rd}$ step of Algorithm 13 are not changed by the transformation, which results in the invariance of the partitions. This implies also that the whole hierarchy of partitions is also invariant under this transformation. □

The presented algorithm collects only the nearest neighbor partitions with the minimal edge weights and merges if they fulfill the criterion $Ext(CC_i^k, CC_j^k) \leq PInt(CC_i^k, CC_j^k)$. This way of merging is also known as *s*ingle linkage clustering [Lance and Williams, 1967].

## 6.5 Experiments on Image Graphs

We start with the trivial partition, where each pixel is a homogeneous region. The attributes of edges are defined as the difference of its end point vertices. The attributes of edges can be defined as the difference between end point features of end vertices,

$$attr_e(u_i, u_j) = |F(u_i) - F(u_j)|, \tag{6.18}$$

where $F$ is some feature. Other distances could be used as well e.g. [Shi and Malik, 1997],

$$attr_e(u_i, u_j) = e^{\frac{-||F(u_i)-F(u_j)||_2^2}{\sigma_I}}, \tag{6.19}$$

where $F$ is some feature, and $\sigma_I$ is a parameter, which controls the scale of proximity measures of $F$. $F$ could be defined as

$$F(u_i) = I(u_i), \tag{6.20}$$

for gray value intensity images, or

$$F(u_i) = [v_i, v_i \cdot s_i \cdot \sin(h_i), v_i \cdot s_i \cdot \cos(h_i)], \tag{6.21}$$

for color images in HSV color distance [Shi and Malik, 1997]. However the choice of the definition of the weights and the features to be used is in general a hard problem, since the grouping cues could conflict each other [Malik et al., 1999].

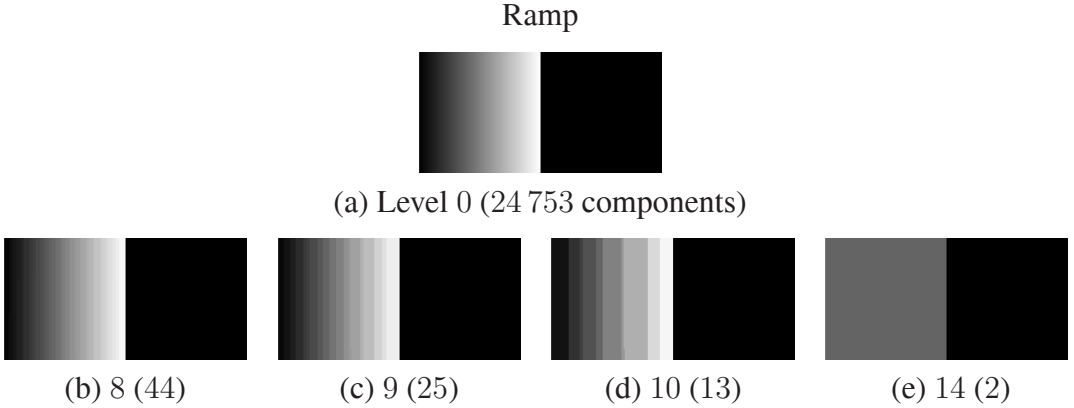
For our experiments we use as attributes of edges the difference between pixel intensities, Equation 6.18, $F(u_i) = I(u_i)$, i.e. a simple distance in $L_1$ norm,

$$attr_e(u_i, u_j) = |I(u_i) - I(u_j)|, \tag{6.22}$$

Using color images, one may think to valuate each edge by the Euclidean distance between the

**Table 6.1:** Test Images.

| Image | Size of image | | Number of vertices |
|---|---|---|---|
| Ramp | $223 \times 111$ | $=$ | 24 753 |
| Lena | $512 \times 512$ | $=$ | 262 144 |
| Monarch | $768 \times 512$ | $=$ | 393 216 |
| Tulips | $400 \times 400$ | $=$ | 160 000 |
| Woman | $116 \times 261$ | $=$ | 25 056 |
| Object 45 | $128 \times 128$ | $=$ | 16 384 |
| Object 18 | $128 \times 128$ | $=$ | 16 384 |




(a) Level 0 (24 753 components)

(b) 8 (44) (c) 9 (25) (d) 10 (13) (e) 14 (2)

*L*egend: level of pyramid (number of components)

**Figure 6.5:** Partitioning of a ramp.

vertice's colors using a perceptual color space such as the CIE-Luv or CIE-Lab[8]. However, the use of CIE color spaces requires the knowledge of the illuminants defining RGB components which is often not available. Therefore, for the sake of simplicity and in order to valuate our method we choose in our experiments a simple Euclidean distance in RGB space. Note that the method is applicable to any color space as well. Using any other more complex norm, have not shown any considerable difference in quality of the result. To compute the hierarchy of partitions we also need to define

$$\tau(CC) = \alpha/|CC|, \tag{6.23}$$

where $\alpha = const$ and $|CC|$ is the number of elements in $CC$, i.e. the size of the region. The algorithm has one running parameter $\alpha$, which is used to compute the function $\tau$. A larger constant $\alpha$ sets the preference for larger components. A more complex definition of $\tau(CC)$, which is large for certain shapes and small otherwise would produce a partitioning which prefers certain shapes, e.g. using ratio of perimeter to area would prefer components that are compact, e.g. not long and thin. Note that as size of $|CC|$ gets larger, which happens as the algorithms

[8]L: luminance, a: red-green color information, and b: yellow-blue color information.


**Figure 6.6:** Partitioning of Lena.

proceeds toward the top of the pyramid, the function $\tau \rightarrow 0$, which means that the influence of the parameter decreases. For computational efficiency we store in vertices the internal contrast $Int()$ and the size of the connected component $|CC|$ (receptive field). We use indoor and outdoor RGB images given in Table 6.1 to test the method. We found that $\alpha = 300$ produces the best hierarchy of partitions of the images shown in 'Lena' and 'Tulips' [9] (Figure 6.6 and 6.8), Monarch [3] (Figure 6.7) , and 'Object45' and 'Object18' [10](Figure 6.10 and Figure 6.11, respectively). For the image in Figure 6.5 and Figure 6.9, $\alpha = 1000$ is chosen.

Images on different level of the pyramid are visualized after the average intensity attribute

[9]Waterloo image database.
[10]Coil 100 image database.

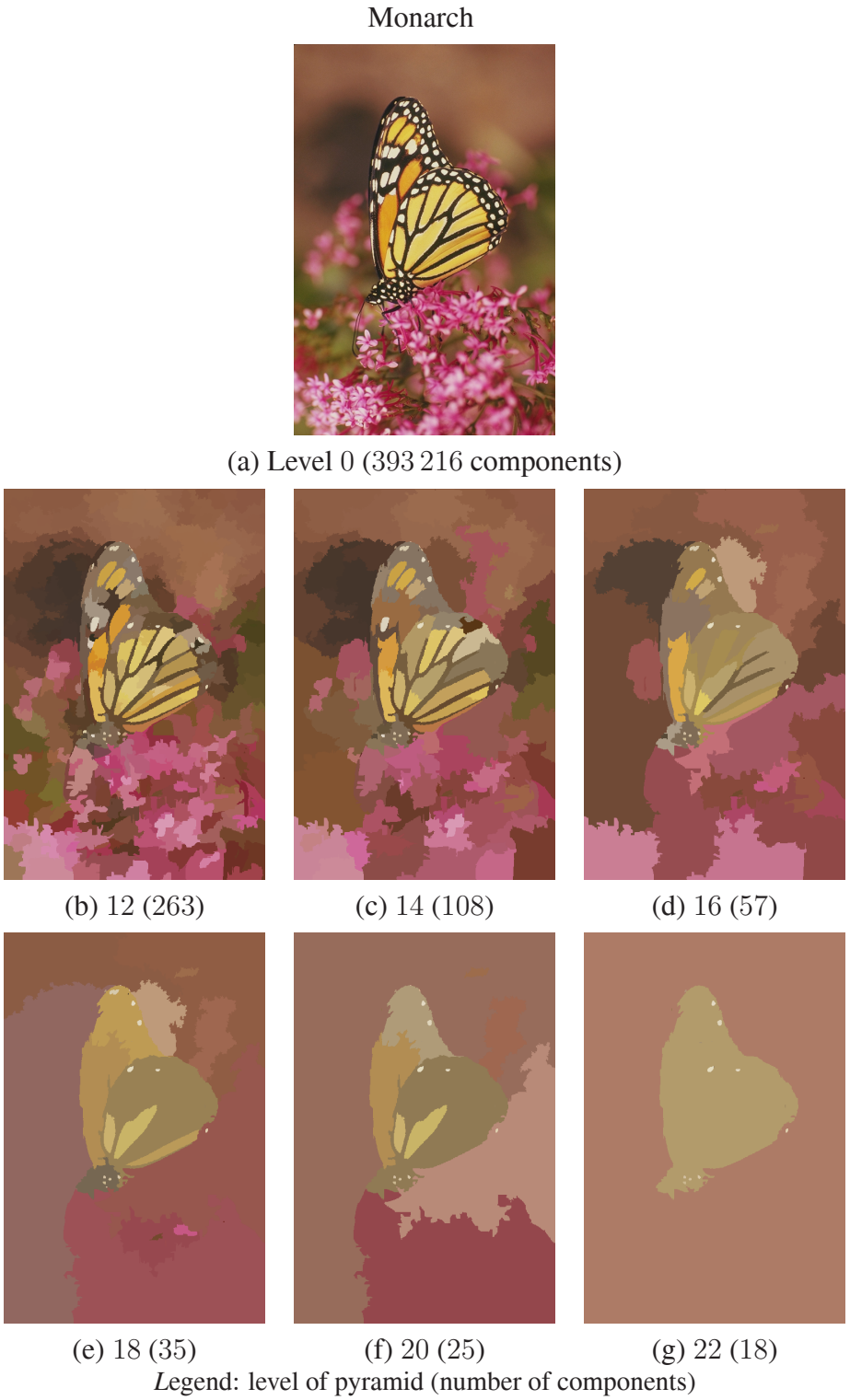



**Figure 6.7:** Partitioning of Monarch.

Tulips

(a) Level 0 (160 000 components)

(b) 21 (2387)

(c) 27 (321)

(d) 37 (41)

(e) 42 (9)

*L*egend: level of pyramid (number of components)

**Figure 6.8:** Partitioning of Tulips.

of vertices is down-projected onto their receptive fields on the base grid. Figures 6.6, 6.7, 6.10, 6.11, 6.8, 6.5 and 6.9 show some of the partitions on different levels of the pyramid and the number of components (number of vertices). More examples of segmentation of gray value images using different decimation strategies (MIS and D3P) are given in Chapter 7. In general the top of the pyramid will consist of one vertex, an apex, which represents the whole image.

### 6.5.1 Discussion of Results

In Chapter 7 a direct comparison of the quality of segmentation results of the method presented here and the ones presented in [Felzenszwalb and Huttenlocher, 2004] and [Shi and Malik, 2000] are given. Moreover in this chapter we study also the segmentation results of different pyramid building strategies: MIES, MIS and D3P. Note that in all images there are regions of large intensity variability and gradient. See for example the hair of Lena or the flower in Monarch, or the gradual change of intensity in the cup of Object45 and Object18. Note also that the background of the Woman image is not homogeneous and contains white spots. This algorithm copes well with this kind of gradient and variability of pixel intensity. The ramp image is taken as a test image to show that the method can cope with the gradient images bordered on 'homogeneous' regions. Even thought the algorithm makes only local decision it is able to capture some certain perceptual groupings. [Felzenszwalb and Huttenlocher, 2004], [Vergés-Llahi et al., 2000] gives final segmentation, in contrast to the result our method which is a hierarchy of partitions with multiple resolutions suitable for further goal driven, domain specific analysis, since the final segmentation is hard to define without knowing the context of the image. Note that a whole class of partitions is created, where a partition is not limited to a certain level of the pyramid, but can be constructed of components from different levels from

Woman

(a) Level 0 (25 056 components)

(b) 8 (120)

(c) 10 (38)

(d) 14 (7)

(e) 15 (3)

*L*egend: level of pyramid (number of components)

**Figure 6.9:** Partitioning of Woman.

the receptive fields of the vertices of a multilevel partition which occupy the whole image, and do not overlap. On the lower level of the pyramid the image is over segmented (partitioned) whereas in upper it is under segmented (partitioned), the help of mid and high level knowledge would select the proper partitioning. Since the algorithm preserves details in low-variability regions, a noisy pixel would survive through the hierarchy. Of course, image smoothing in low variability regions would overcome this problem. We, however do not smooth the images (as is done in [Felzenszwalb and Huttenlocher, 2004]), as this would introduce another parameter into the method. The hierarchy of partitions can also be built from an over-segmented image

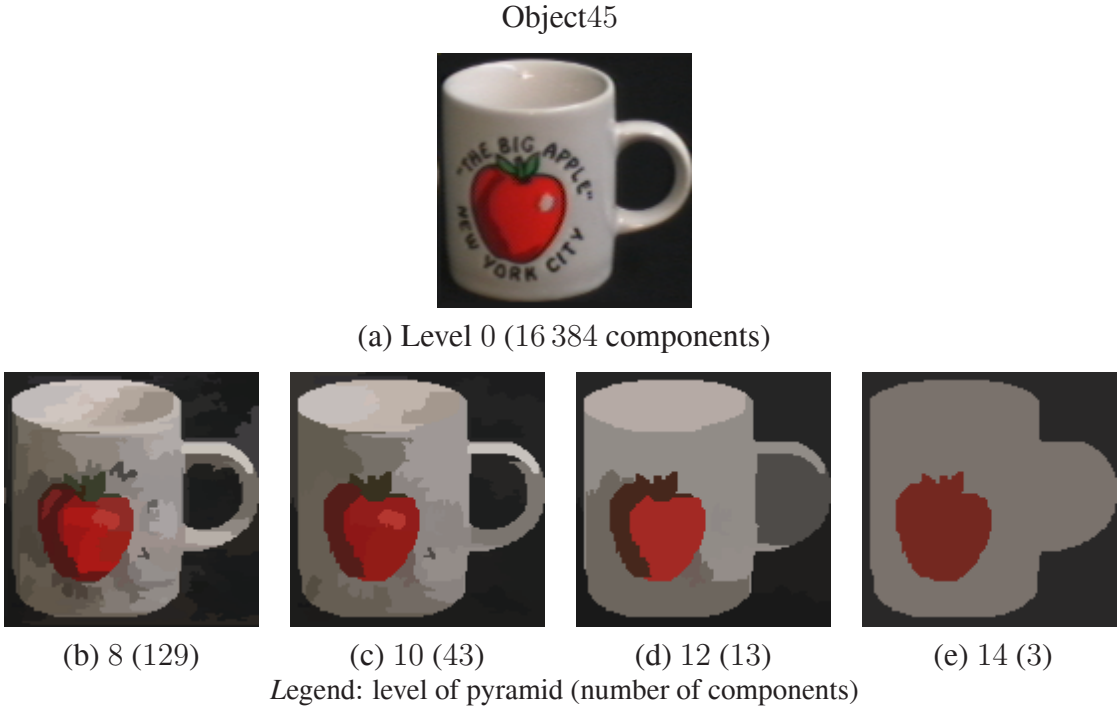



**Figure 6.10:** Partitioning of Object45.

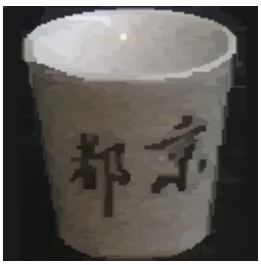
to overcome the problem of noisy pixels. Note that the influence of $\tau$ in decision criterion is smaller as the region gets bigger for a constant $\alpha$. The constant $\alpha$ is used to produce a kind of the over-segmented image and the influence of $\tau$ is smaller after each level of the pyramid. For an over-segmented image, where the size of regions is large, the algorithm becomes parameterless. The method in based on the minimum spanning tree principle therefore it is computationally efficient, running in $\mathcal{O}(|V| \log |V|)$ time for $|V|$ vertices, i.e. pixels if the base of the hierarchy is an image.

## 6.6 Conclusion

In this chapter a method to build a hierarchy of partitions of an image by comparing in a pairwise manner the difference along the boundary of two components relative to the differences of components' internal differences is introduced. Even though the algorithm makes simple greedy decisions locally, it produces perceptually important partitions in a bottom-up 'stimulus-driven' way based only on local differences. It was shown that the algorithm can handle large variation and gradient intensity in images. Since this framework is general enough, RAGs of any over-segmented image can be used to build the hierarchy of partitions. External knowledge can help in a top-down segmentation technique. A drawback is that the maximum and minimum criterion is very sensitive to noise, although in practice it has a small impact. Other criteria like median would lead to an NP-complete algorithm. The algorithm has only one running parameter which controls the sizes of the regions. In order to make this method produce only

Object18

(a) Level 0 (16 384 components)



(b) 23 (512)

(c) 32 (65)

(d) 39 (10)

(e) 42 (4)

*L*egend: level of pyramid (number of components)

**Figure 6.11:** Partitioning of Object18.

one single segmentation a stopping rule or selection of vertices on different levels can be applied as in [Xu et al., 1993, Lallich et al., 2003]. An evaluation of this method to other well known graph-based methods is done in Chapter 7.

CHAPTER 7

# Evaluation of Segmentation Methods

"If your experiment needs statistics, you ought to have done a better experiment." [1]

**by Bertrand Russell.**

**Summary** Different graph-based segmentation methods, one based on normalized cut and the others on the Borůvka's minimum spanning tree principle are evaluated with respect to segmentations produced by humans. Depending on different decimation strategies (MIS, MIES, and D3P), three version of the Borůvka's based method are used for the evaluation. The Borůvka's based method is compared with the method based on the Kruskal's algorithm. The discrepancy measures are chosen as best suited to compute the segmentation error. The evaluation is done using gray value images. This kind of benchmarking will help understanding for which kind of images particular methods are best suited.

**Keywords:** Segmentation evaluations, normalized cut segmentation, Borůvka's based segmentation method, MIS, MIES, D3P, Kruskal's based segmentation method, pyramid segmentation method.

## 7.1 Introduction

The segmentation process results in 'homogeneous' regions with respect to the low-level cues using some similarity measures. Problems emerge because the homogeneity of low levels does not always lead to semantics and the difficulty of defining the degree of homogeneity of a region. Also some of the cues can contradict each other. Thus, low-level cue image segmentation cannot produce a complete final 'good' segmentation [Sudhir and Sarkar, 1997], leading researchers to look the segmentation only in the context of a task, as well as the evaluation of the

[1] From *Mathematical Approach to Biology and Medicine*. N. T. J. Bailey , New York: Wiley, 1967.

segmentation methods. However in [Martin et al., 2001] the segmentation is evaluated purely[2] as segmentation by comparing the segmentation done by humans with those done by a particular method. As can be seen in Figure 7.1 and 7.2 [3], there is a consistency of segmentation done by humans (already demonstrated empirically in [Martin et al., 2001]), even thought humans segment images at different granularity (refinement or coarsening). This refinement or coarsening could be thought as hierarchical structure of the image, i.e. the pyramid. Note that the segmented image #35 Figure 7.1 in (a) can be coarsened to obtain the image in (c), this is called *simple refinement*; whereas to obtain image in (b) from (a) (or vice versa) we must coarsen in one part of the image and refine in the other (notice the chin of the man in (b)), this is called *mutual refinement*. Therefore in [Martin et al., 2001] a segmentation consistency measure that does not penalize this granularity is defined ( Section 7.4).

In this chapter, we evaluate two graph-based segmentation methods, the normalized cut [Shi and Malik, 1997](NCutSeg) and the method based on the Borůvka's minimun spanning tree (MST) [Haxhimusa and Kropatsch, 2003](BorůSeg)(see Chapter 6 for more details). In fact we evaluate three flavors of the BorůSeg depending on the decimation strategy used: MIS, MIES or D3P, and denoted by BorůSeg (MIS), BorůSeg (MIES) and BorůSeg (D3P). See Chapter 6 for details on these decimation strategies. Note that we do not use MIDES in the evaluation since this method is developed to be used where the orientation of contraction is of importance (e.g. in watersheds), which is not the case in the segmentation. The segmentation method based on Kruskal's algorithm [Felzenszwalb and Huttenlocher, 2004](KrusSeg) is compared with the parallel, hierarchical BorůSeg method. We compare these methods following the framework of [Martin et al., 2001] i.e. comparing the segmentation result of the two graph-based methods with the human segmentations and the segmentation of two minimum spanning tree methods to each other. The results of the evaluation are reported in Section 7.4. Also the variation of regions sizes is shown in this section.

Some examples of applying BorůSeg on color image are shown in Chapter 6, Section 6.5, where for visualization purposes each region has the mean color value. In this chapter we use the region borders to highlight the regions. Note that, two pixel wide borders are used only for better visualization purposes, and are not produced by these segmentation methods nor are part of the evaluation process in Section 7.4.

## 7.2 Evaluated Graph-based Segmentation Methods

For an overview of graph based segmentation method please see Chapter 6. In the subsequent subsection only the methods used in the evaluation are given in more details, to highlight the used parameters.

### 7.2.1 Normalized Cuts Based Segmentation Method

For the sake of completion of presentation, in this section a description of the graph partitioning problem based on the normalized graph cut [Shi and Malik, 2000, Shi and Malik, 1997]

[2]The context of the image is not taken into consideration during segmentation.

[3]These images in the original database are in the landscape layout, thus for better visual presentation of images the orderings in this figure is changed with respect to that in Figure 7.1.

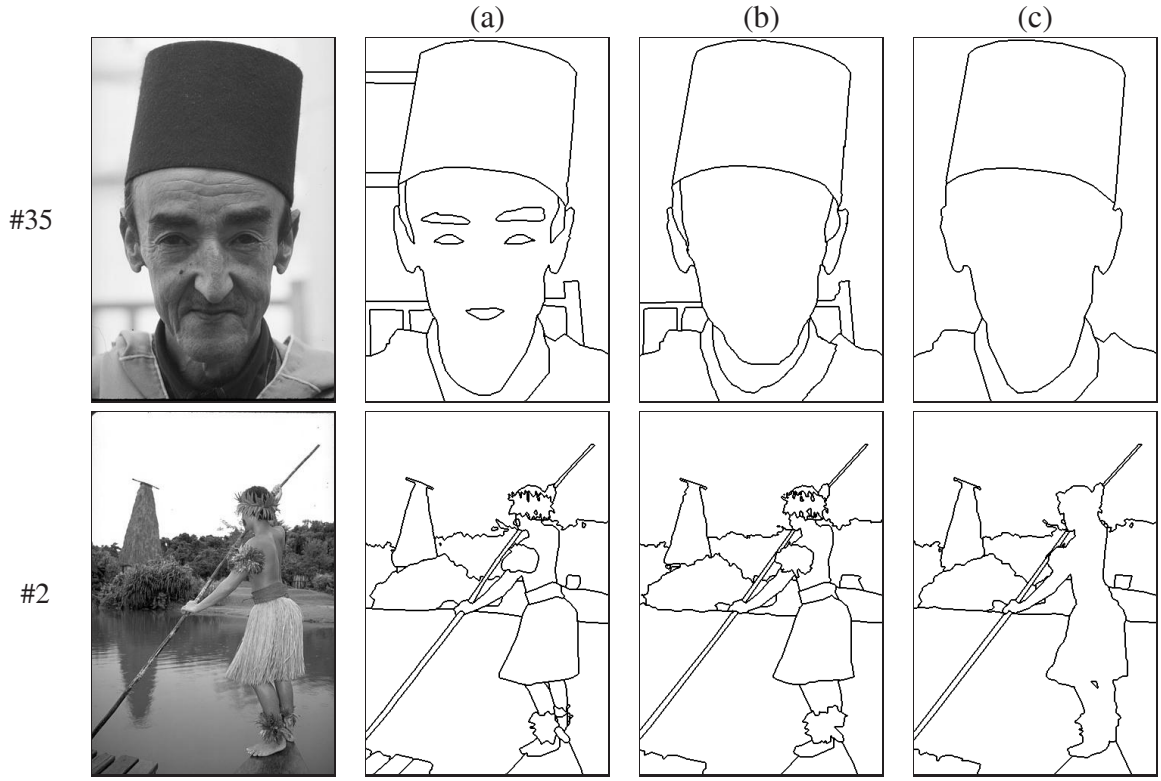



**Figure 7.1:** Sample images with human segmentations [Martin et al., 2001].

(NCutSeg) is presented. The approach starts by representing the problem as a weighted undirected graph $G = (V, E)$, where the vertices represent points in the feature space, and an edge is formed between every pair of vertices. The weight on edges is a function of the similarity between vertices that are joint by edges. The problem is posed as finding the partition of the set of vertices into $V_1, V_2, \ldots, V_m$ such that the similarity is high among the vertices in set $V_i$ and is low across different sets $V_i$, $V_j$. The solution in measuring the goodness of the image partitioning is found as the minimization of the normalized cut, which is formulated as a generalized eigenvalue problem.

The graph $G = (V, E)$ can be partitioned into two disjoint sets, say $A$, $B$ such that $A \cup B = V$ and $A \cap B = \emptyset$, by simply cutting (deleting) edges connecting vertices of these sets. The sum of the weights of deleted edges can be used to measure the dissimilarity between these two sets. In [Wu and Leahy, 1993] this measures is minimized to produce a clustering method. The clustering is done by recursively minimizing the cut criterion in the resulted segments. As it is shown in [Wu and Leahy, 1993], this global optimal criterion can be used to produce 'good' segmentation on images, but the method was biased toward cutting small sets (mostly containing a single vertex). In order to overcome this problem [Shi and Malik, 2000] propose a normalized cut criterion. The weights of the edges for this method in this evaluation are set to:

$$w_{i,j} = e^{-\frac{\|I(i)-I(j)\|}{\sigma_I}} \cdot \begin{cases} e^{-\frac{\|X(i)-X(j)\|}{\sigma_X}} & \text{if } \|X(i)-X(j)\| < r, \\ 0 & \text{otherwise,} \end{cases} \tag{7.1}$$

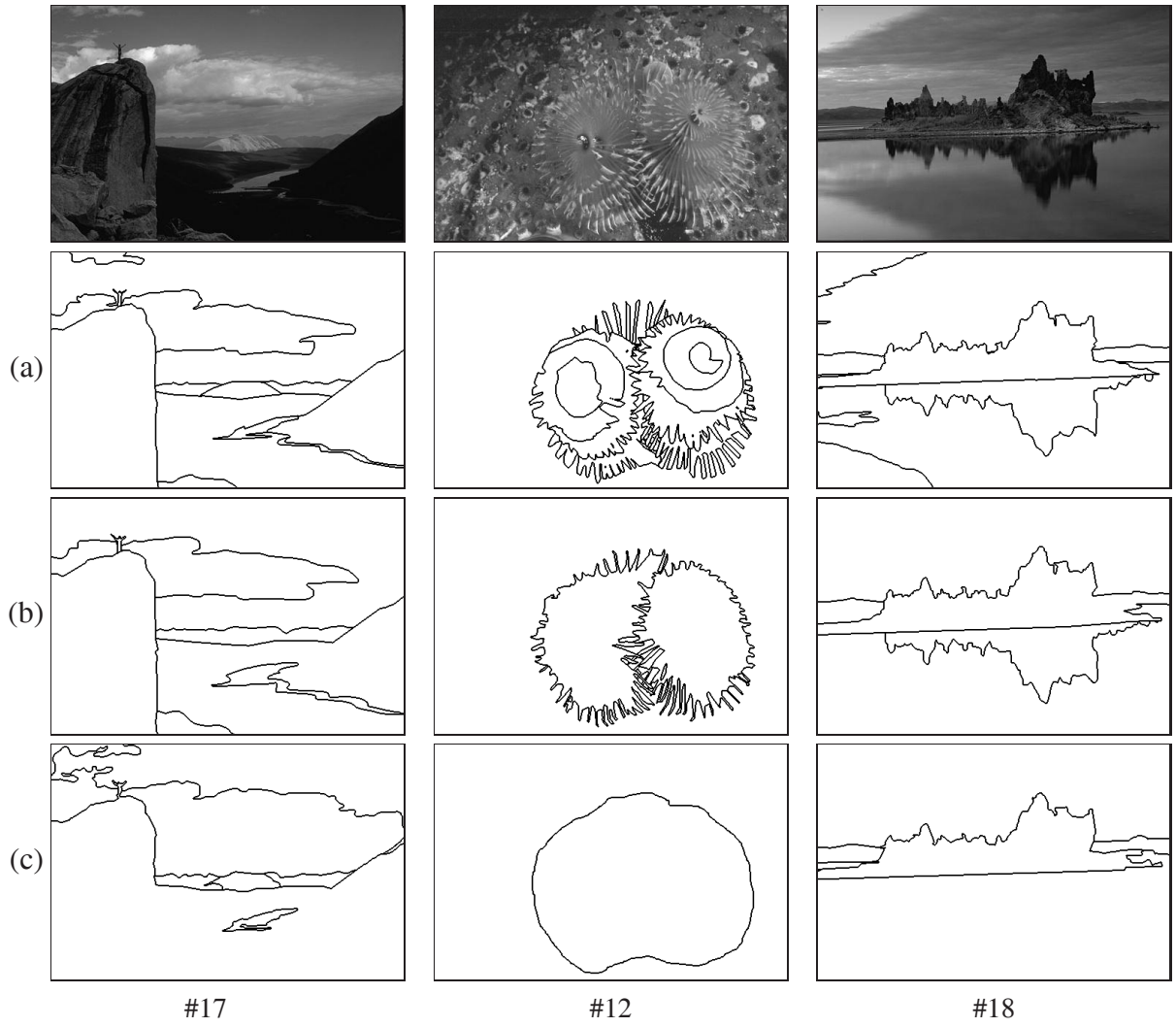



**Figure 7.2:** Sample images with human segmentations [Martin et al., 2001], Figure 7.1 cont.

where $X(i)$ is the spatial location of the vertex $i$, $I(i)$ the intensity value. Note that the first term depends on the difference of intensities. The weight $w_{i,j}$ are set to zero if $i$ and $j$ are $r$ pixels apart. $\sigma_I$ and $\sigma_X$ represent some parameters to control the influence of intensity and/or spatial position on the overall weight. The source code is taken from `www.cis.upenn.edu/~jshi/software/` and the default parameters are not changed ($r = 10$, $\sigma_X = 30$, and $\sigma_I = 0.1$). The method was only instructed to give a particular number of regions. Some segmentation results of NCutSeg method are given in Figure 7.3 and 7.4

### 7.2.2 Kruskal's Minimum Spanning Tree Based Segmentation Method

In this section a description of the graph segmentation method based on the minimum spanning tree principle [Felzenszwalb and Huttenlocher, 2004, Felzenszwalb and Huttenlocher, 1998] (KrusSeg) is shortly presented. The core of this algorithm is the Kruskal's MST algorithm pre-

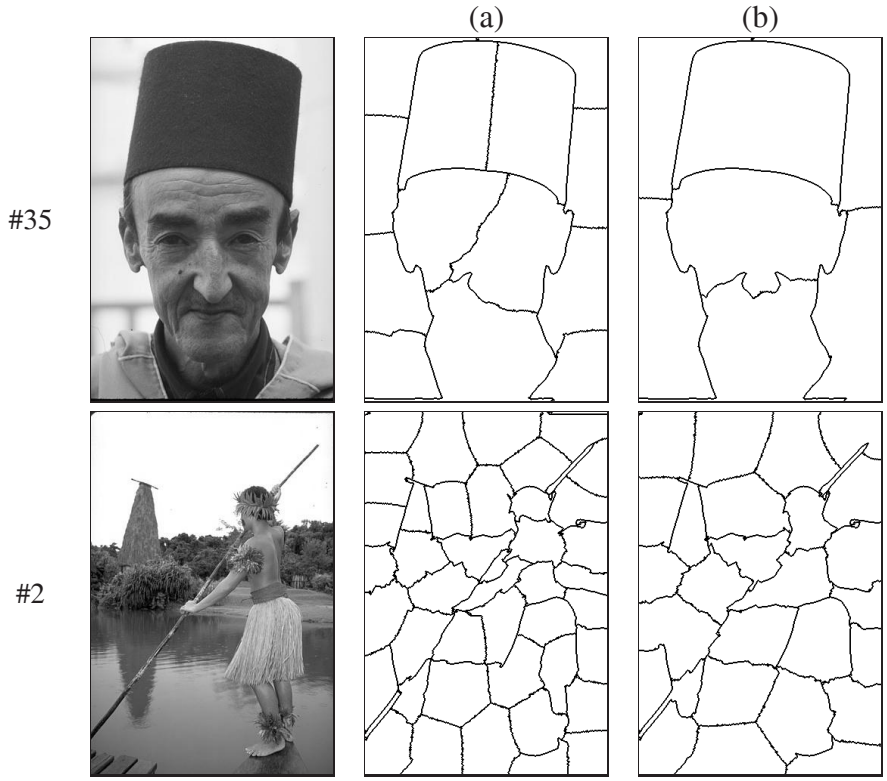



**Figure 7.3:** Segmentation produced by NCutSeg method.

sented in Chapter 6, Section 6.2.2. As in the previous section an undirected weighted graph $G = (V, E)$ is build from the the feature points (e.g. pixels), where $v \in V$ represent the set of vertices to be segmented, and each edge $e \in E$ is weighted by a non-negative measure of dissimilarity between neighboring vertices (segments). In this graph-based segmentation method, a segmentation $S = (CC_1, \ldots, CC_r)$ is a partitions of the vertex set $V$ into components such that each component corresponds to a connected component in the graph $G$. Thus any segmentation is simply induced by a subset of edges in $E$. The problem is specified as finding segments such that elements in a component are similar, and elements in neighboring components be dissimilar, i.e. edges joining vertices in the same component have relatively low weights, and edges between two neighboring components have higher weights.

In order to evaluate whether or not there is a boundary between two components [Felzenszwalb and Huttenlocher, 2004] defines a predicate $Comp(\cdot, \cdot)$ based on the dissimilarity between elements along the border of the two components relative to a measure of the dissimilarity within the components. See Chapter 6, Section 6.4, Equation 6.13 for the definition of this predicate. In order to use this predicate one must define the function $\tau(CC)$:

$$\tau(CC) = \frac{\alpha}{|CC|}, \tag{7.2}$$

where $|CC|$ is the size of the connected component, and $\alpha$ is constant parameter. This parameter is a scale of observation, in that larger $\alpha$ sets preferences for larger components, but it is not

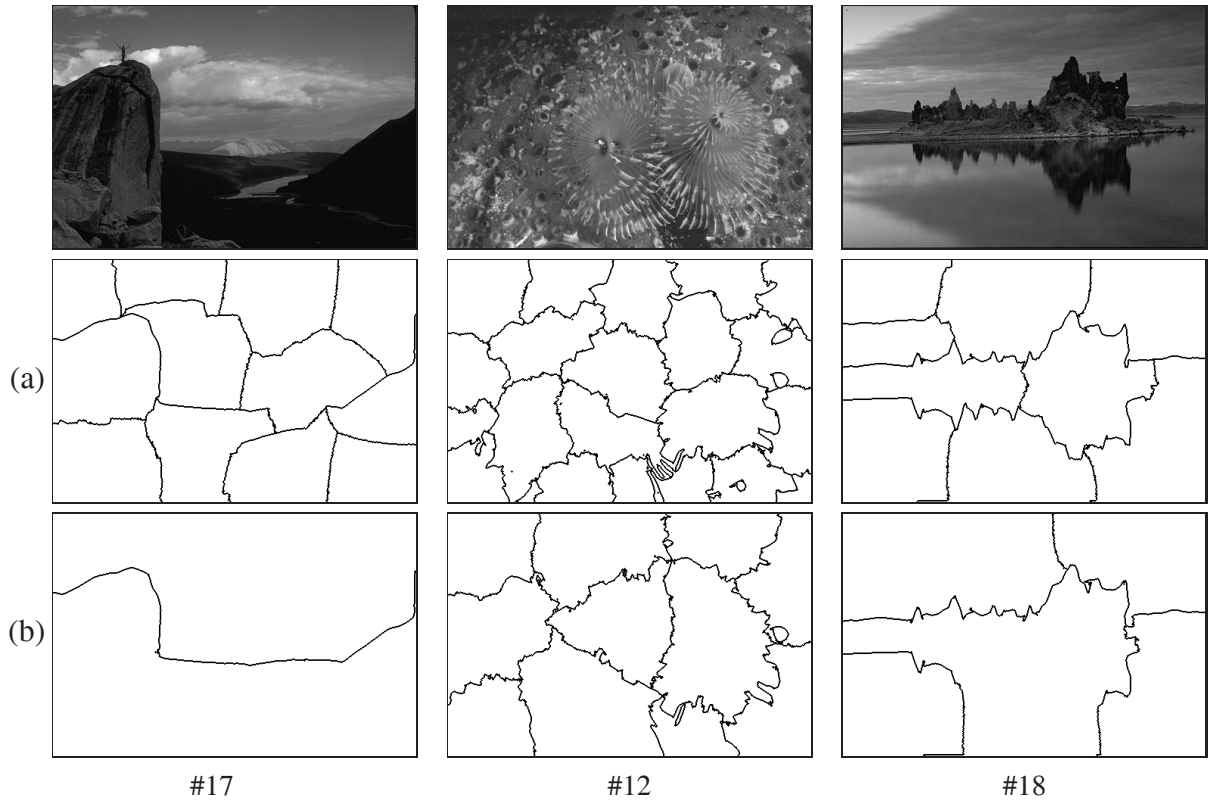



**Figure 7.4:** Segmentation produced by NCutSeg method, Figure 7.3 cont.

the minimum component. Thus, the algorithm segmentation is given in Algorithm 14 [Felzenszwalb and Huttenlocher, 2004].

The weight $w(o)$ between two vertices $v_i$ and $v_j$ is set to the absolute intensity difference between the pixels connected by an edge:

$$w(o) = w(v_i, v_j) = |I(p_i) - I(p_j)| \tag{7.3}$$

where $I(p_i)$ is the intensity of the pixel $p_i$. The images are first smoothed as required by [Felzenszwalb and Huttenlocher, 2004] with a Gaussian filter before the edge weight are computed. As proposed by [Felzenszwalb and Huttenlocher, 2004], the default parameters for the evaluation are not changed, the constant of the function $\tau$ is set to $\alpha = 300$ and the Gaussian filter is used with $\sigma = 1.5$. Some segmentation results of KrusSeg method are given in Figure 7.5 and 7.6 with different parameter settings.

### KruSeg versus BorůSeg

Here we shortly discuss the differences of KrusSeg versus BorůSeg methods. Even thought both of the methods use the same internal/external merging criteria and the same minimum spanning tree principle, one might expect that the final segmentation of KrusSeg and at least one of the levels of the pyramid produced by BorůSeg should at least be the same. This is not the case as

**Algorithm 14 – KrusSeg Algorithm**

*Input*: Weighted graph $G = (V, E)$

1: sort edges of $E$ by non-decreasing order into $\pi = \{o_1, o_2, \ldots, o_m\}$
2: start with a segmentation $S^0$, in which every vertex is a connected component.
3: $q \leftarrow 1$
4: **repeat**
5: **if** $CC_i^{q-1} \neq CC_j^{q-1}$ and $w(o_q) \leq PInt(CC_i^{q-1}, CC_j^{q-1})$ **then**
6: $S^q$ is obtained from $S^{q-1}$ by merging $CC_i^{q-1}$ and $CC_j^{q-1}$ through the edge $o_q$
7: $q \leftarrow q + 1$
8: **else**
9: $S^q = S^{q-1}$
10: **end if**
11: **until** $q = m$ segments

*Output*: $m$ connected segments $S = S^m$.

can be seen by comparing Figures 7.5 and 7.6 with Figures 7.7 and 7.8; 7.9 and 7.10; 7.11 and 7.12 respectively.

In the first step of the Algorithm 14 a sorting of edges based on their weight is done. Let $e$ be an edge that connects two regions $CC_i$ and $CC_j$. Note that the sorting is done outside the repeat loop (steps 4-11). This means, that if the edge $e \in \pi$ that connects two regions $CC_i$ and $CC_j$, is thrown out-side of the if-else loop, because it did not satisfy the internal/external criteria i.e. $w(e) > PInt(CC_i, CC_j)$ ($9^{th}$ step is executed), this edge will not be considered in any further iteration. Thus regions, $CC_i$, and $CC_j$ connected by this edge will never be merged.

In the case of the Algorithm 13, since there is no sorting, the edge $e$ that does not satisfy the internal/external criteria ($6^{th}$ step), it is not thrown from the further processing, but it might be processed in the upper levels of the pyramid. During the generation of new level in the pyramid, the internal contrast of both regions, $CC_i$ and $CC_j$ connected by the edge $e$ increases, hence the $PInt(CC_i, CC_j)$ will increase as well. At the brake point were $w(e) \leq PInt(CC_i, CC_j)$ the two regions will be merged by the BorůSeg. And this behavior has occurred, which explains the differences between results of these two methods. Note that the inclusion trees are different, because of the way the data is processed in these algorithms.

Both methods, KrusSeg and BorůSeg use a threshold dependent on the size of the connected component ($k/|CC|$[4] as shown in the subsection above and in Chapter 6) in the merging criteria. Setting this threshold to zero both of the methods would produce the MST of the image, independent of the way the data is processed.

### 7.2.3 Results of the Graph-based Segmentation Methods

The segmentation results of NCutSeg on gray value images are shown in Figures 7.3 and 7.4; of KrusSeg in Figures 7.5 and 7.6; of BorůSeg (MIS) in Figures 7.7 and 7.8; of BorůSeg (MIES) in Figures 7.9 and 7.10 and of BorůSeg (D3P) in Figures 7.11 and 7.12. Note that NCutSeg

[4] $|CC|$ cardinality of the connected component.


**Figure 7.5:** Segmentation produced by KrusSeg.

and BorůSeg methods are capable of producing a hierarchy of images, whereas KrusSeg gives a final segmentation.

These methods use only local contrast based on pixel intensity values. As it is expected, and can be seen from the Figures 7.3, 7.4, 7.5, 7.6, 7.7, 7.8, 7.9, 7.10, and 7.11, 7.12, segmentation methods which are based only on low-level local cues cannot create segmentation results as good as humans. Even thought it looks like, the NCutSeg method produces more regions, actually the overall number of regions in Figure 7.3a, 7.4a and Figures 7.7b, 7.8b 7.9b, 7.10b 7.11b, 7.12b are almost the same, but the BorůSeg produces a bigger number of small regions. The BorůSeg method is capable of producing a hierarchy of images, the pyramid shown in Figures 7.7, 7.8 7.9, 7.10 7.11, 7.12 under (a) represent lower levels of the pyramid, (b) the middle levels, and (c) the higher levels. We smoothed the images before segmenting them with the KrusSeg[5] method (Gaussian with parameter $\sigma = 1.5$), whereas BorůSeg worked with non smoothed images.

Anyway the methods (see Figures 7.3, 7.5, 7.7 7.9 and 7.11) were capable of segmenting the face of a man satisfactory (image #35). The BorůSeg method did not merge the statue on the top of the mountain with the sky (image #17), compared to humans which do segment this statue as a single region (see Figure 7.2). All methods have problems segmenting the see creatures (image #12). Note that the segmentation done by humans on the image of rocks (image #18), contains

[5]The method is very sensitive to noise [Felzenszwalb and Huttenlocher, 2004].

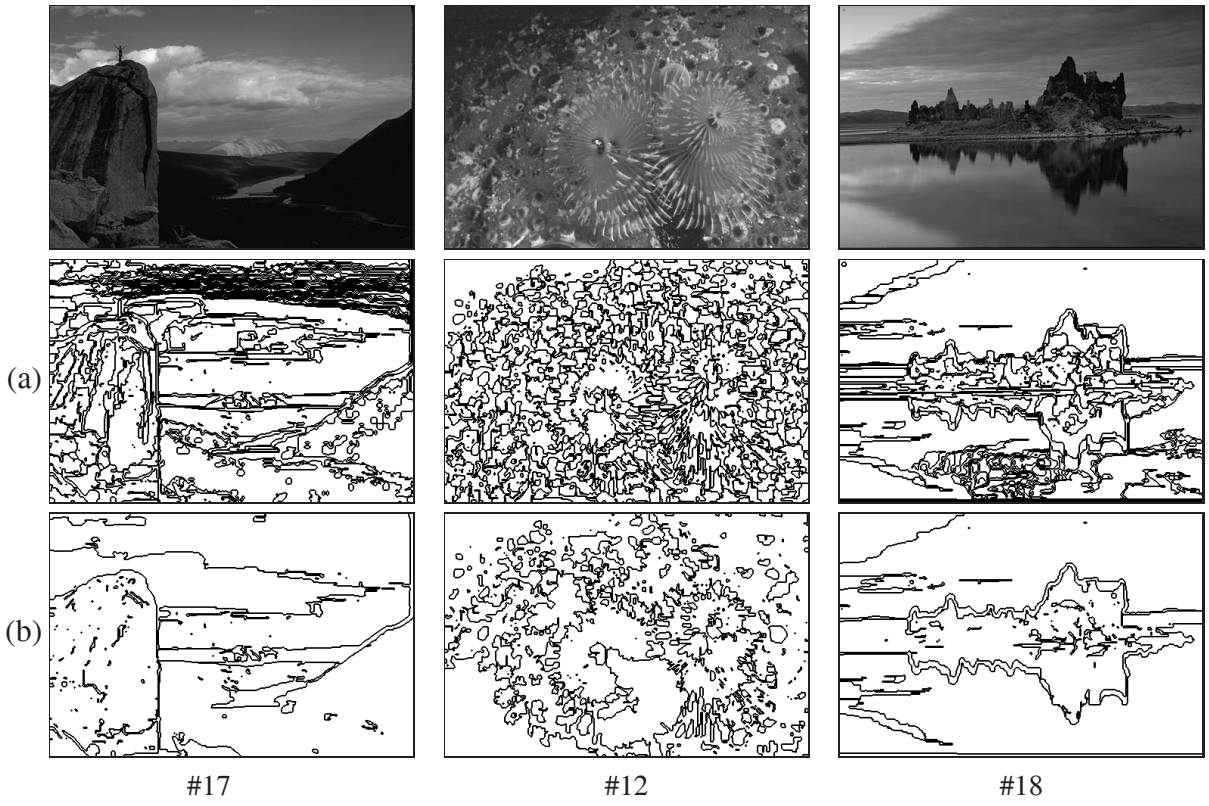



**Figure 7.6:** Segmentation produced by KrusSeg, Figure 7.5 cont.

the symmetry of axis, even thought there is no 'big' change in the local contrast, therefore the NCutSeg and BorůSeg methods fail in this respect, whereas the KrusSeg manged to recover this axis. It must be mentioned that none of the methods is 'looking' for this axis of symmetry.

## 7.3 Evaluating Segmentations

Evaluation of the segmentation algorithms is difficult because it depends on many factors [Heath et al., 1997] among them: the segmentation algorithm; the parameters of the algorithm ; the type(s) of images used in the evaluation; the method for evaluation of the segmentation algorithms, etc. Our evaluation copes with these facts: (i) real world images should be used, because it is difficult to extrapolate conclusion based on synthetic images to real images [Zhou et al., 1989], and (ii)the human should be the the final evaluator [Cinque et al., 1994].

There are two general methods to evaluate segmentations:

- qualitative, and
- quantitative methods.

Qualitative methods are evaluated by humans, meaning that different observers would give different opinions about the segmentations (e.g. already encountered in edge detection evaluation [Heath et al., 1997], or in image segmentation [Martin et al., 2001]). On the other hand

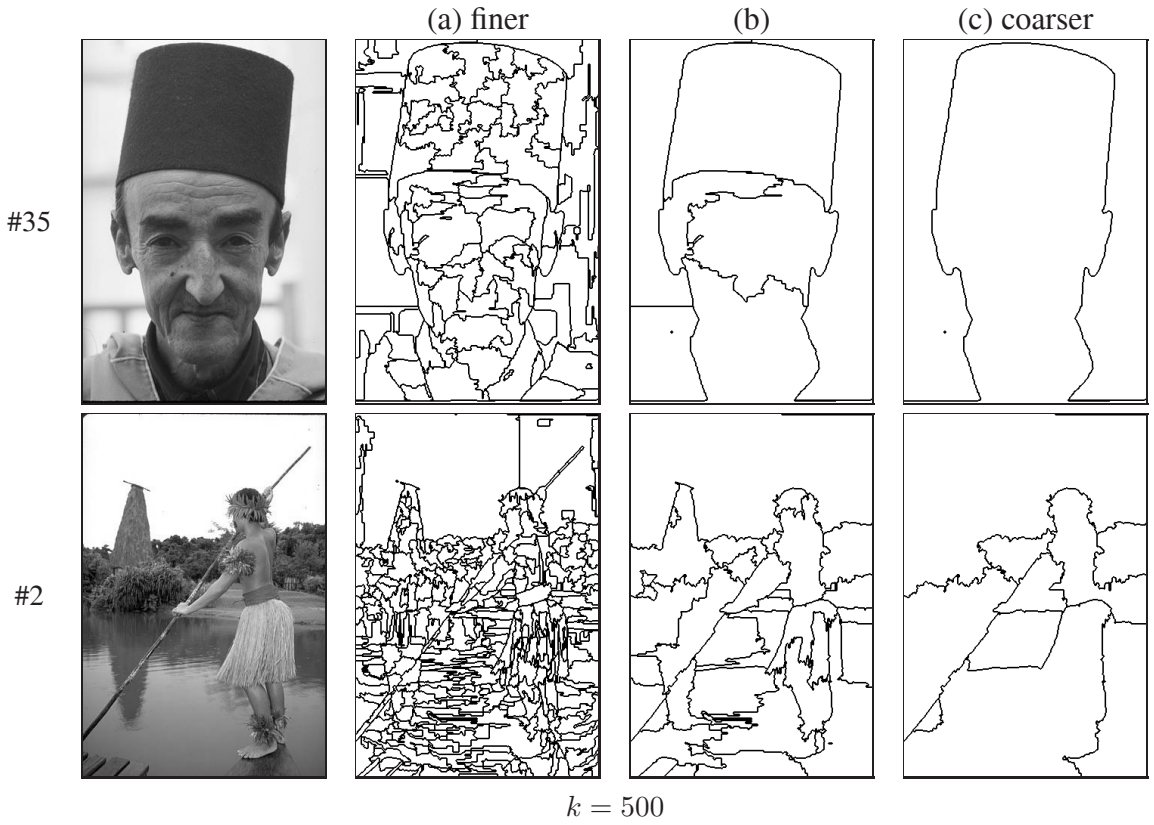



**Figure 7.7:** Hierarchy of segmentation produced by BorůSeg (MIS).

the quantitative methods are classified into analytic methods and empirical methods [Zhang, 1996]. Analytical methods study the principles and properties of the algorithm, like processing complexity, efficiency and so on. For references on the analytic studies of methods based on eigenvalue decomposition and on minimum spanning tree please see Chapter 6, Section 6.1.1. The empirical methods study properties of the segmentations by measuring how 'good' a segmentation is close to an 'ideal' one, by measuring this 'goodness' with some function of parameters. Both of the approaches depend on the subjects, the first one in coming up with the reference (perfect) segmentation[6] and the second one defining the function. The difference between the segmented image and the reference (ideal) one can be used to asses the performance of the algorithm [Zhang, 1996]. The reference image could be a synthetic image or manually segmented by humans. These discrepancy methods measure the difference between the segmented image and reference images. Higher value of the discrepancy means bigger error, signaling poor performance of the segmentation method. In [Zhang, 1996], it is concluded that evaluation methods based on "mis-segmented pixels should be more powerful that other methods using other measures". In [Martin et al., 2001] the error measures used for segmentation evaluation 'count' the mis-segmented pixels.

[6] Also called a gold standard [Graaf et al., 1994].

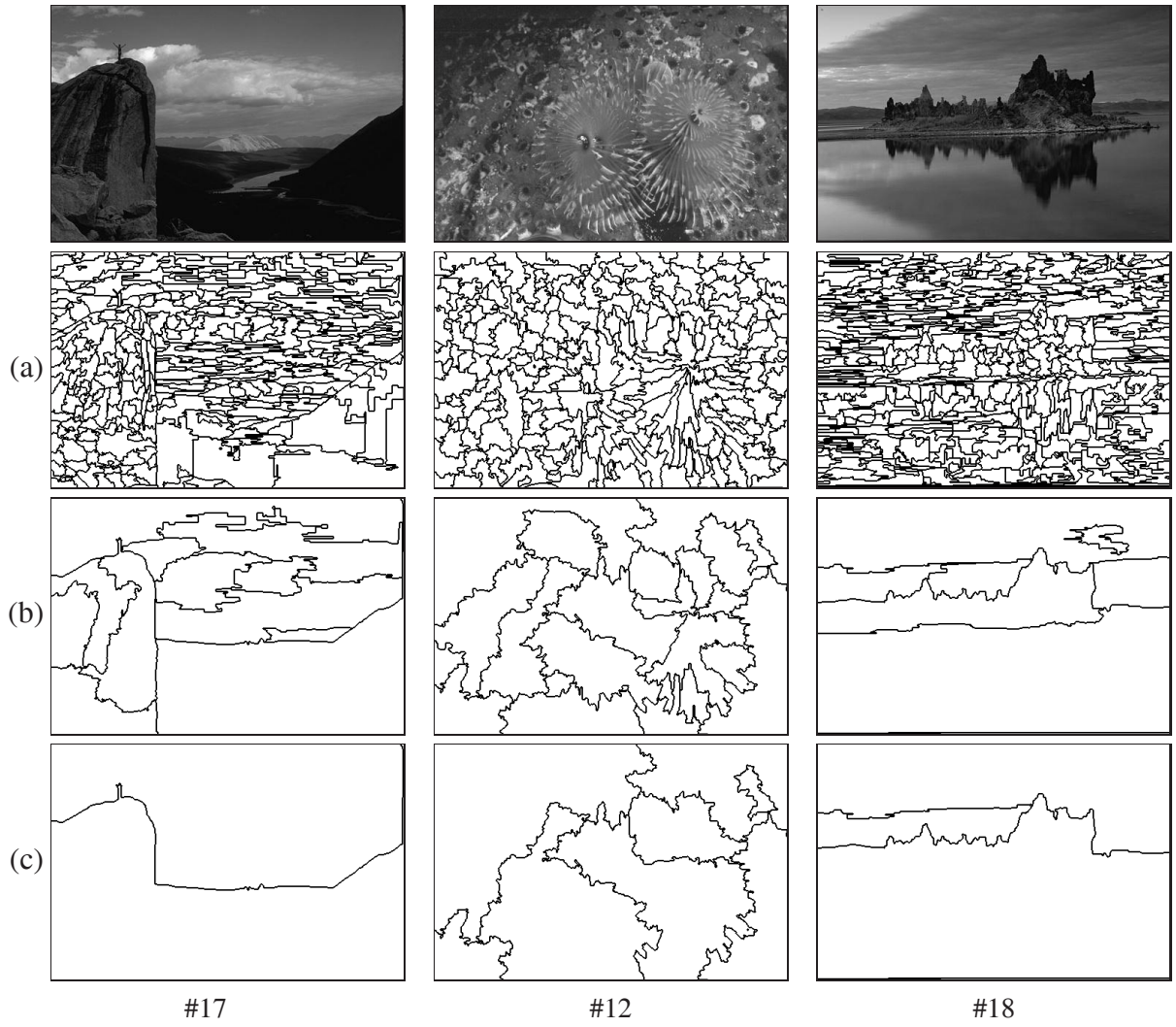



**Figure 7.8:** Hierarchy of segmentation produced by BorůSeg (MIS), Figure 7.7 cont.

## 7.4 Segmentation Benchmarking

In [Martin et al., 2001] segmentations made by humans are used as a reference and basis for benchmarking segmentations produced by different methods. The concept behind this, is the observation, that even though different people produce different segmentations for the same image, the obtained segmentations differ, mostly, only in the local refinement of certain regions. This concept has been studied on the human segmentation database (see Figure 7.1 and 7.2) created and maintained by them [Martin et al., 2001] and used as a basis for defining two error measures, which do not penalize a segmentation if it is coarser or more refined than another. In this sense, a *pixel error measure* $E(S_1, S_2, p)$, called the local refinement error, is defined as:

$$E(S_1, S_2, p) = \frac{|R(S_1, p) \backslash R(S_2, p)|}{|R(S_1, p)|} \tag{7.4}$$

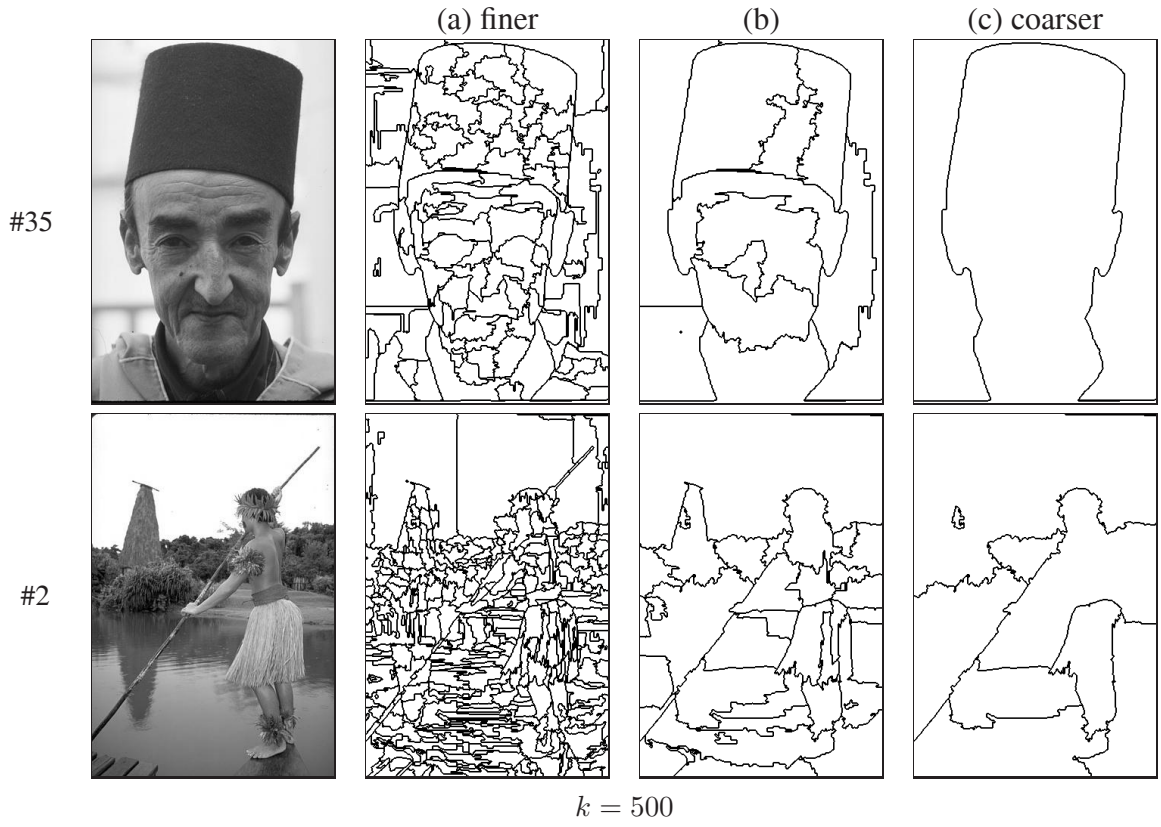



**Figure 7.9:** Hierarchy of segmentation produced by BorůSeg (MIES).

where $\setminus$ denotes set difference, $|x|$ the cardinality of a set $x$, and $R(S, p)$ is the set of pixels corresponding to the region in segmentation $S$ that contains pixel $p$. Using the local refinement error $E(S_1, S_2, p)$ the following error measures are defined [Martin et al., 2001]: the *Global consistency error* (GCE), which forces all local refinements to be in the same direction, and is defined as:

$$GCE(S_1, S_2) = \frac{1}{|I|} \min \left\{ \sum_{p \in I} E(S_1, S_2, p), \sum_{p \in I} E(S_2, S_1, p) \right\} \tag{7.5}$$

and the *Local consistency error* (LCE), which allows refinement in different directions in different parts of the image, and is defined as:

$$LCE(S_1, S_2) = \frac{1}{|I|} \sum_{p \in I} \min \left\{ E(S_1, S_2, p), E(S_2, S_1, p) \right\}, \tag{7.6}$$

where $|I|$ is the number of pixels in the image $I$. Notice that LCE $\leq$ GCE for any two segmentations. GCE is tougher measure than LCE, because GCE tolerates simple refinements, while LCE tolerates mutual refinement as well.

We have used the GCE and LCE measures presented above to do a evaluation of the BorůSeg method using the human segmented images, from the Berkley humans segmented images database [Martin et al., 2001]. The results of comparison of the NCutSeg method versus humans

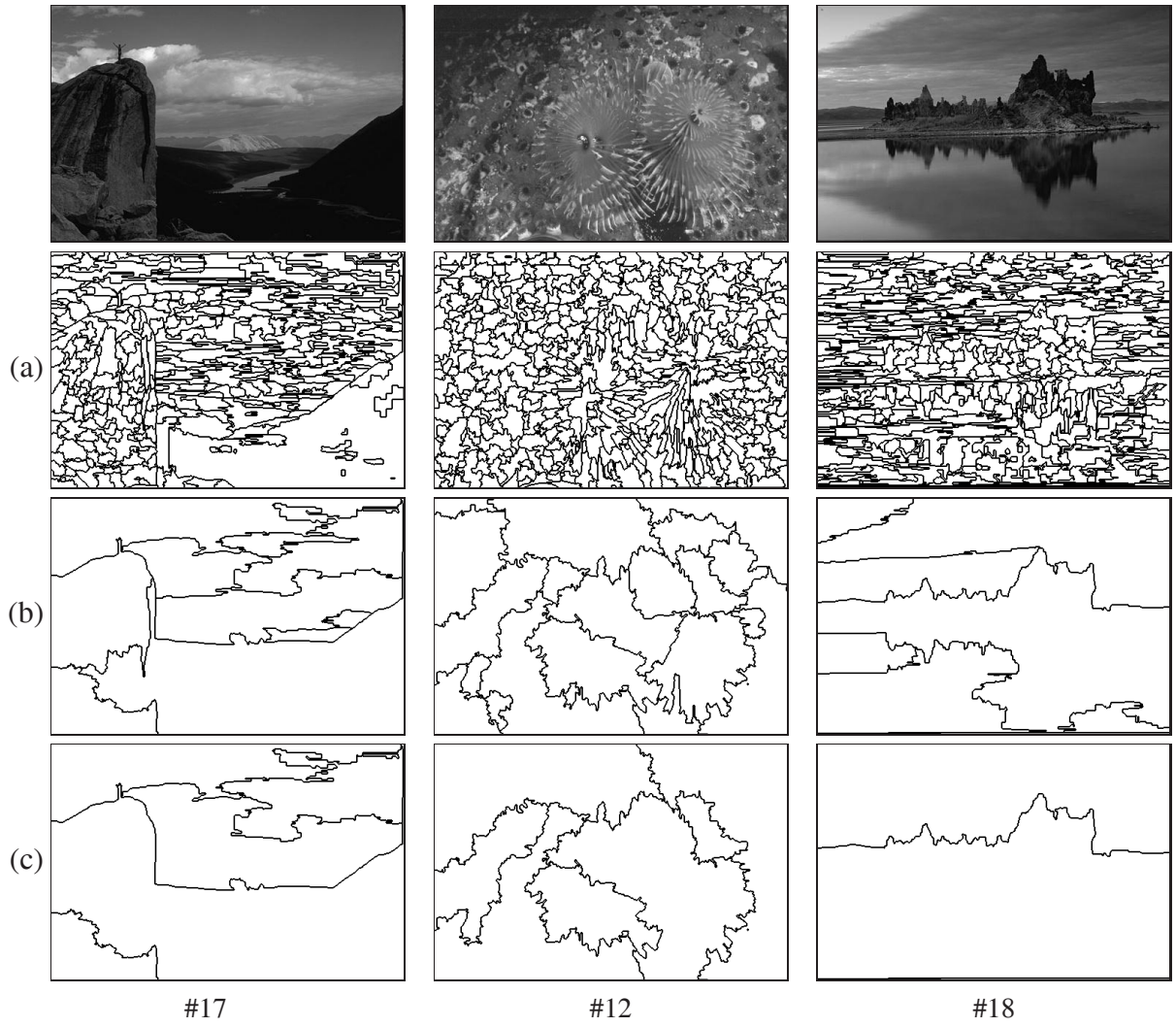



**Figure 7.10:** Hierarchy of segmentation produced by BorůSeg (MIES), Figure 7.9 cont.

and humans versus humans are confirmed [Martin et al., 2001]. The segmentations of BorůSeg method are compared with KrusSeg as well, in order to show that these method do not produce the same segmentation results.

### 7.4.1 Evaluation of Segmentations on Berkley Image Database

As mentioned in [Martin et al., 2001] a segmentation consisting of a single region and a segmentation where each pixel is a region, is the coarsest and finest possible of any segmentation. In this sense, the LCE and GCE measures should not be used when the number of regions in the two segmentation differs a lot. In this line of ideas, taking into consideration that methods can produce segmentations with different number of regions, we have taken for each image as a region count reference number, the average number of regions from the human segmentations available for that image. We instructed the NCutSeg to produce the same number of regions

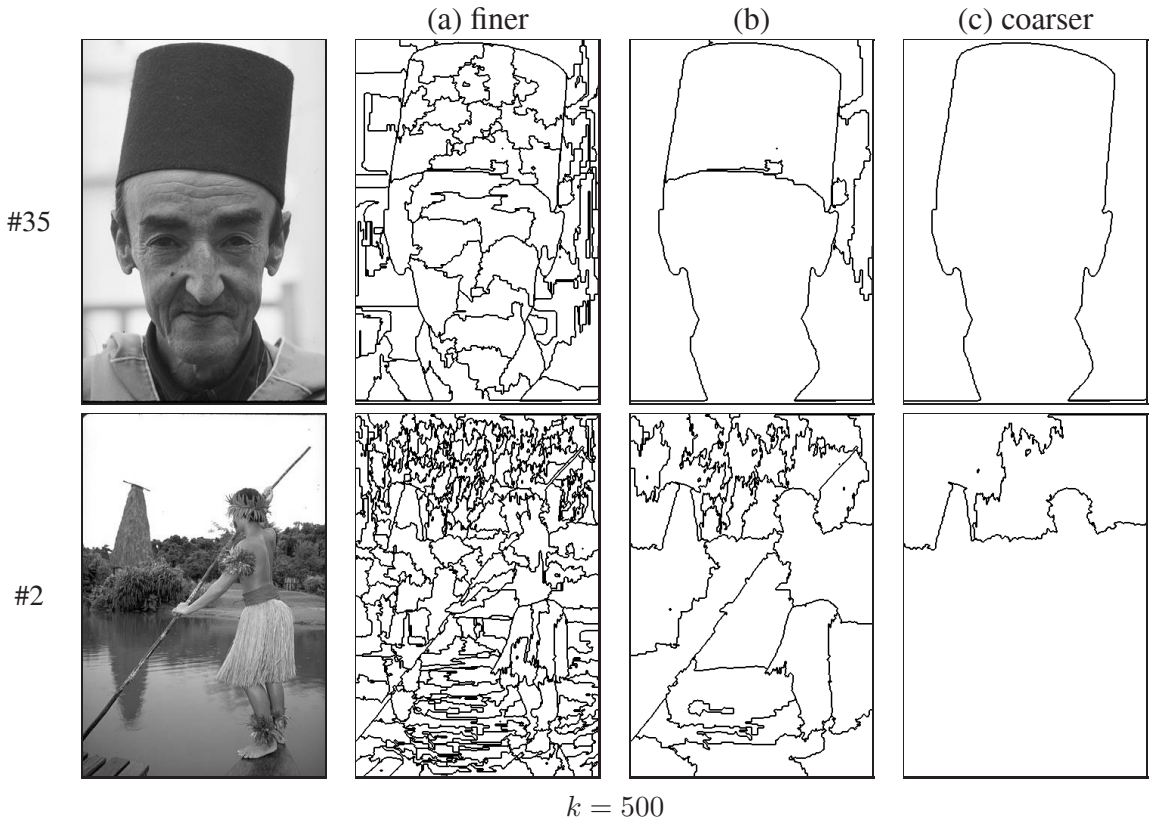



**Figure 7.11:** Hierarchy of segmentation produced by BorůSeg (D3P).

and for the BorůSeg we have taken the level of the pyramid that has the region number closest to the same region count reference number. The input images to the KrusSeg method have been smoothed with a Gaussian filter (e.g. $\sigma = 1.5$), as recommended by [Felzenszwalb and Huttenlocher, 2004]. Because the KrusSeg still produced much more regions than the human segmentations in the database have, a direct evaluation of the KrusSeg versus the humans would have been unfair, because of different number of regions in segmenations. So, taking into consideration that the BorůSeg produces a whole hierarchy of segmentations with different number of regions (from coarser to finer), when evaluating the KrusSeg method, we have selected for the evaluation a levels of the pyramid that has the number of regions closest to the number of regions produced by the KrusSeg method. In all the cases this meant going lower in the pyramid and taking a level which is basically a refinement of the one used when comparing to the humans.

As data for the experiments, we take $100$ gray level images from the Berkley Image Database[7]. The names of these images are given in Appendix D. For segmentation, we have used the normalized cuts Matlab implementation, version 7, available on the Internet[8] and for the BorůSeg and KrusSeg we have implementations based on combinatorial pyramids [Haxhimusa et al., 2005a].

[7] http://www.cs.berkeley.edu/projects/vision/grouping/segbench/.
[8] http://www.cis.upenn.edu/∼jshi/software/.

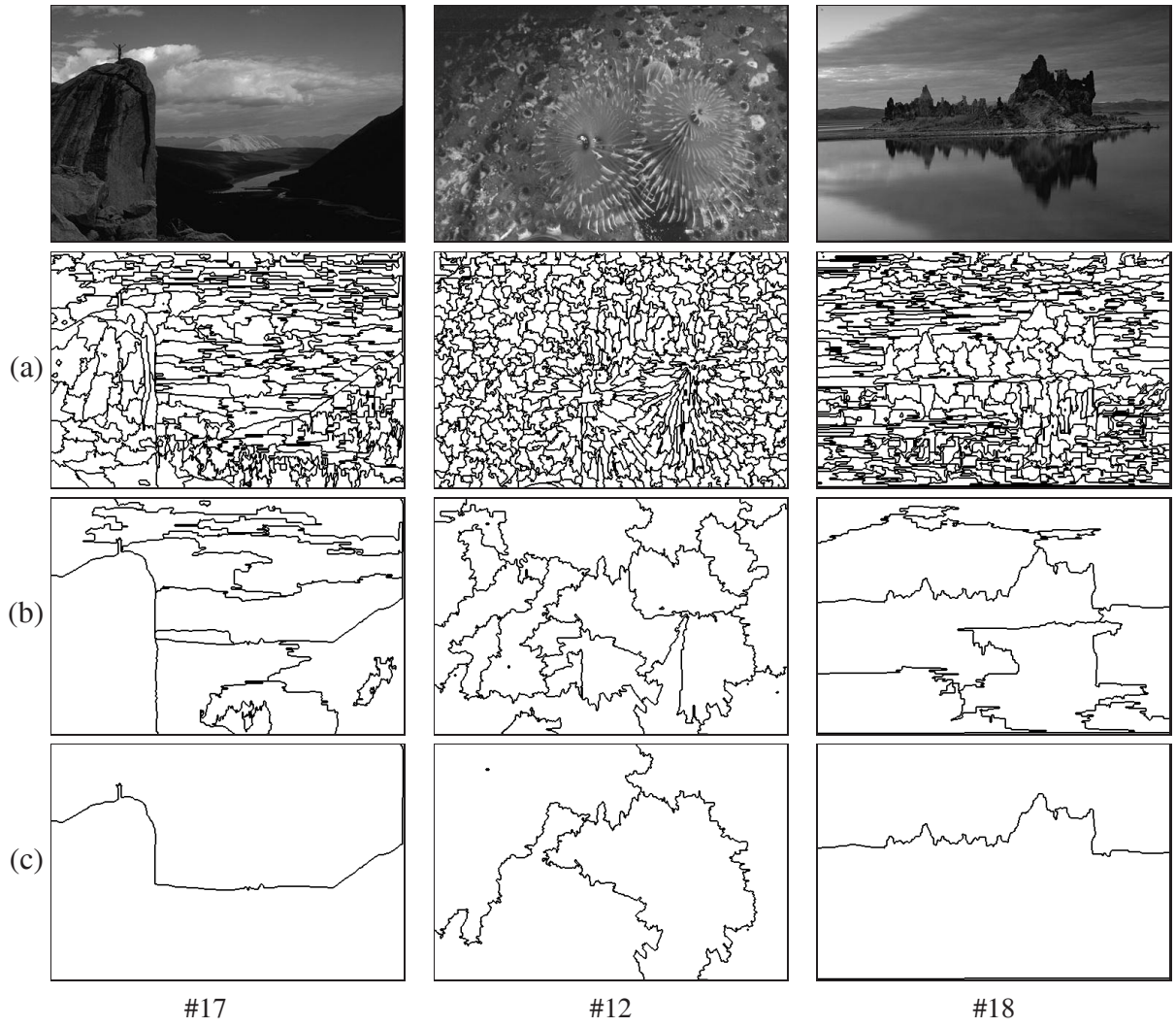



**Figure 7.12:** Hierarchy of segmentation produced by BorůSeg (D3P), Figure 7.11 cont.

For each of the images in the test, we have calculated the GCE and LCE using the results produced by the three methods and all the human segmentations available for that image. Having more then one pair of GCE and LCE for the methods NCutSeg and BorůSeg and each image, we have calculated the mean and the standard deviation. The results can be seen in Figure 7.13, 7.14, 7.15, 7.16 , and 7.17 (where the point stands for the mean, and the interval for the standard deviation). You can notice that there is a big similarity between the values of GCE and LCE for NCutSeg and BorůSeg methods for some images. One can note from the results of the GCE and LCE in Figure 7.13 made by humans, for the same image, that the humans did very good and proved to be consistent when segmenting the same image, and that the NCutSeg and BorůSeg produces segmentations that obtained higher values for the GCE and LCE error measures (Figure 7.14, and 7.15, 7.16, 7.17 respectively). The GCE and LCE using the results produced by the KrusSeg and the corresponding level from the hierarchy produced by BorůSeg, are calculated. The results are summarized in Figure 7.23.

**Table 7.1:** Summary of LCE and GCE discrepancy errors.

| *Method* | | $\widehat{\mu}_{LCE}$ | $\widehat{\mu}_{GCE}$ |
|---|---|---|---|
| Humans | | 0.0592 | 0.0832 |
| NCutSeg | | 0.2041 | 0.2485 |
| BorůSeg | MIES | 0.2038 | 0.2784 |
| | MIS | 0.2000 | 0.2731 |
| | D3P | 0.2150 | 0.3034 |

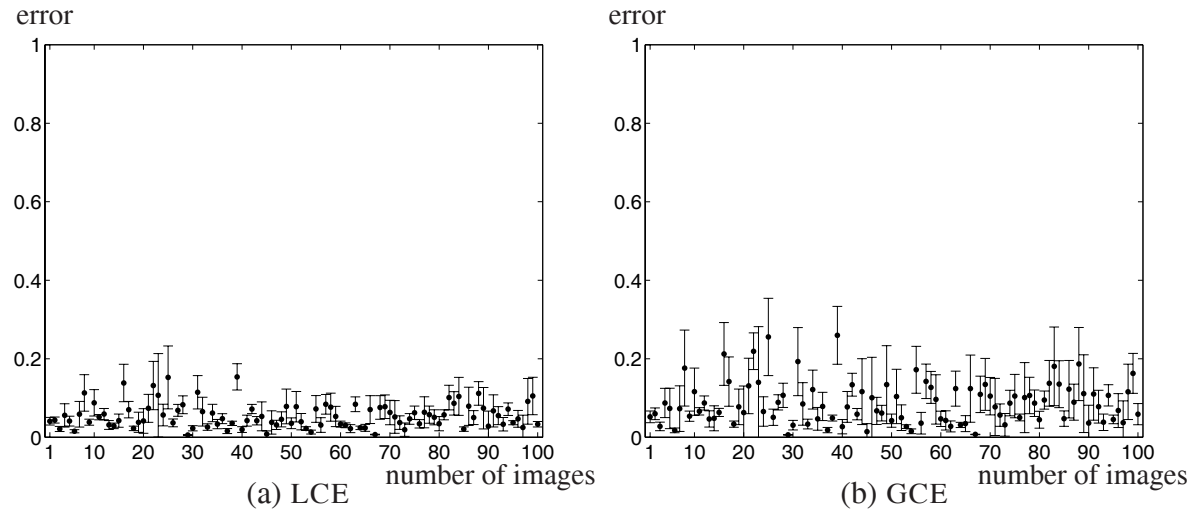



**Figure 7.13:** Error measure results: Human versus Human.

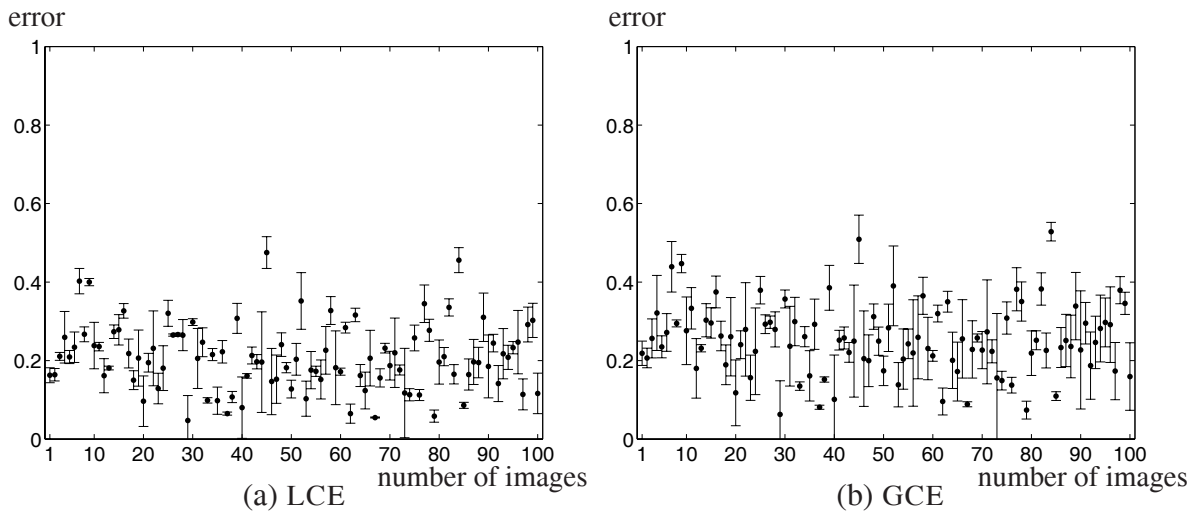



**Figure 7.14:** Error measure results: NCutSeg versus Human.

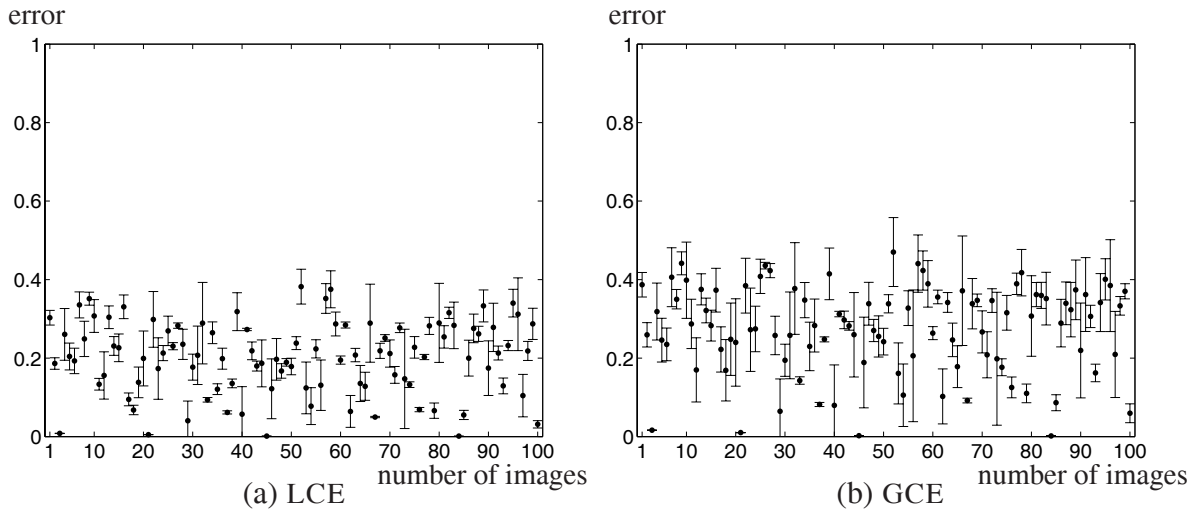



**Figure 7.15:** Error measure results: BorůSeg (MIS) versus Human.

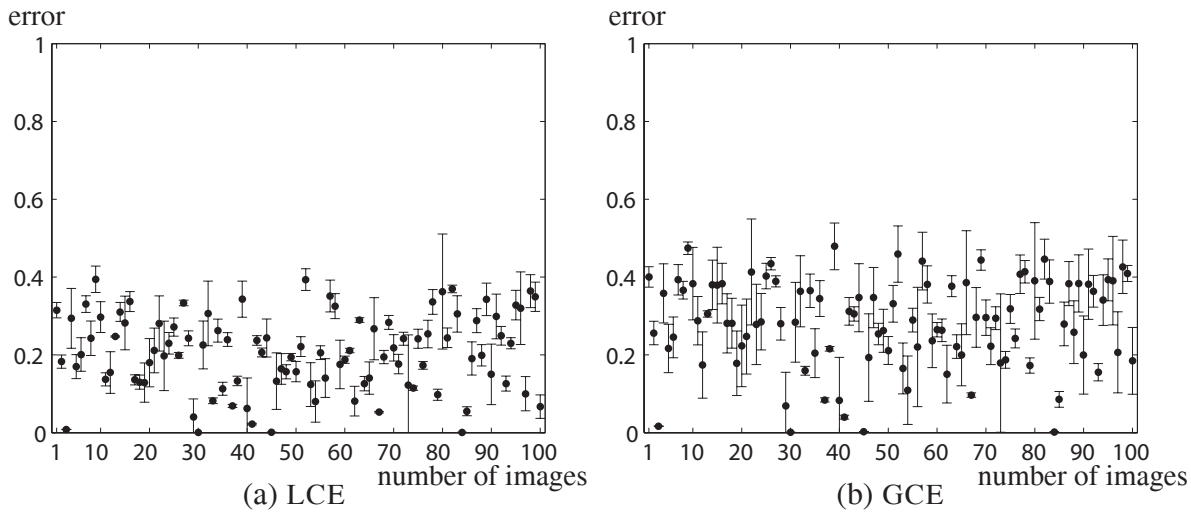



**Figure 7.16:** Error measure results: BorůSeg (MIES) versus Human.

In Fig. 7.18, 7.19, 7.20, 7.21, and 7.22 one can see the histograms of the GCE and LCE values obtained ($[0 \dots 1]$, where zero means no error), Human versus Human, NCutSeg versus Human, BorůSeg (MIES, MIS, D3P) versus Human, and in Figure 7.24 BorůSeg (MIES) versus KrusSeg. In these image $\widehat{\mu}$ represent the mean value of the error. Notice that the humans are consistent in segmenting the images and the Human versus Human histogram shows a peak very close to $0$. i.e. a small $\widehat{\mu} = 0.0592$ for LCE and $\widehat{\mu} = 0.0832$ for GCE. The NCutSeg and BorůSeg there is not a significant difference between the values of LCE and GCE (see the mean values of the respective histograms). One can conclude that the quality of segmentation of these methods seen over the whole database is not different. Also, the results of the histograms in Figure 7.24 show that there is a significant difference (LCE mean $\widehat{\mu} = 0.3619$ and GCE mean $\widehat{\mu} = 0.407$) between the segmentations produced by the BorůSeg and KrusSeg methods. One

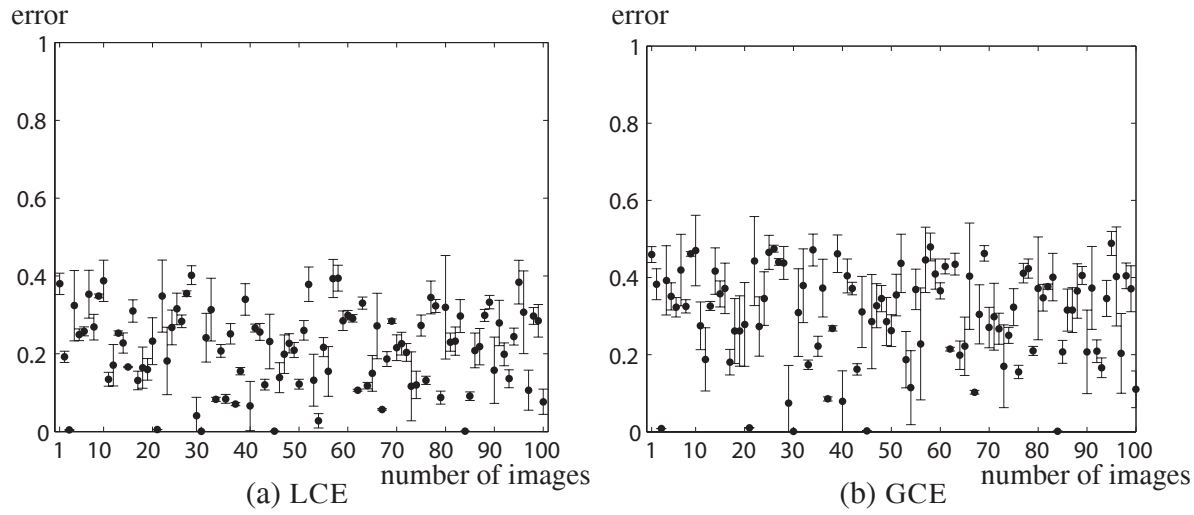



**Figure 7.17:** Error measure results: BorůSeg (D3P) versus Human.

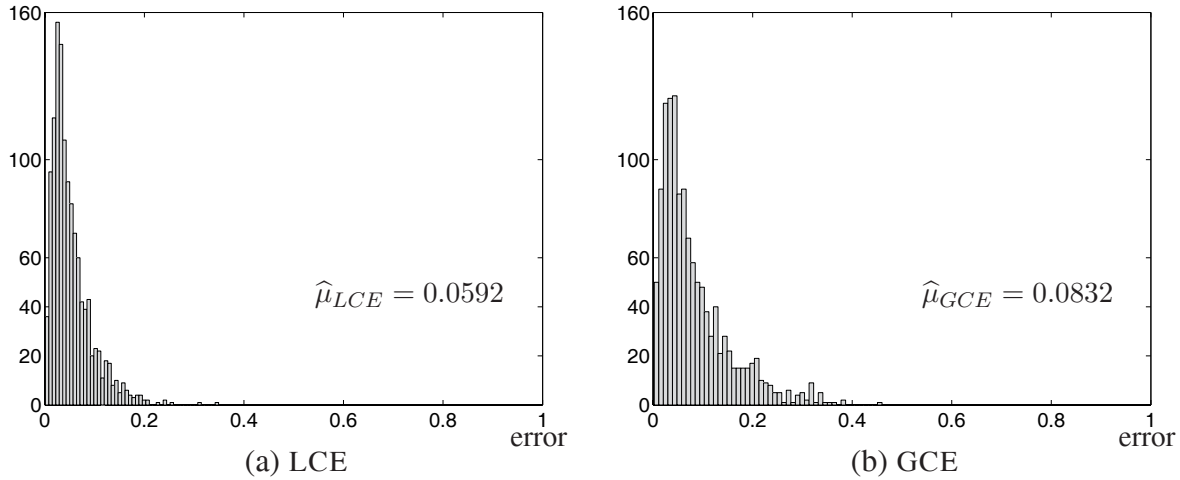
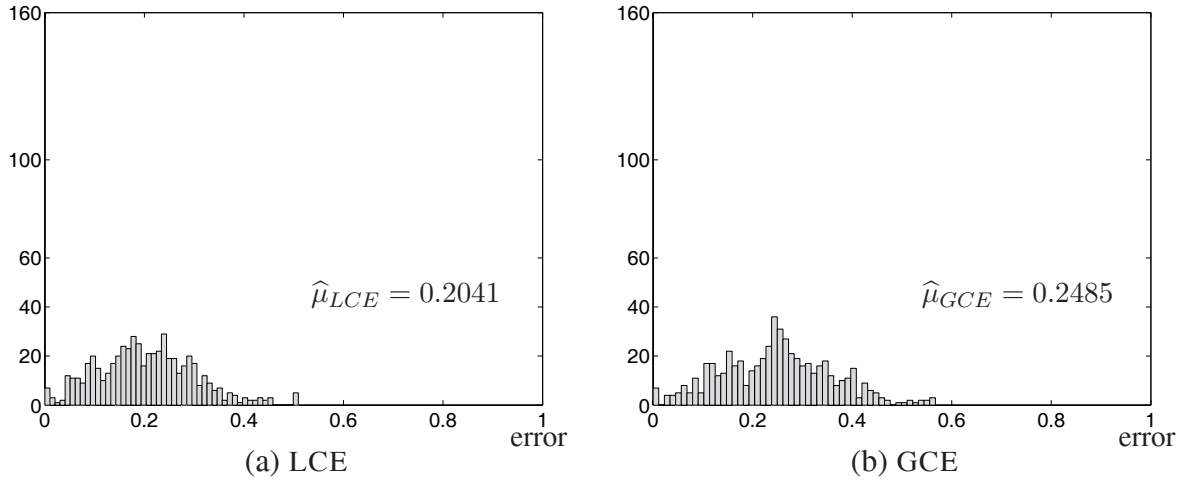



**Figure 7.18:** Histograms of discrepancy measure: Human versus Human.

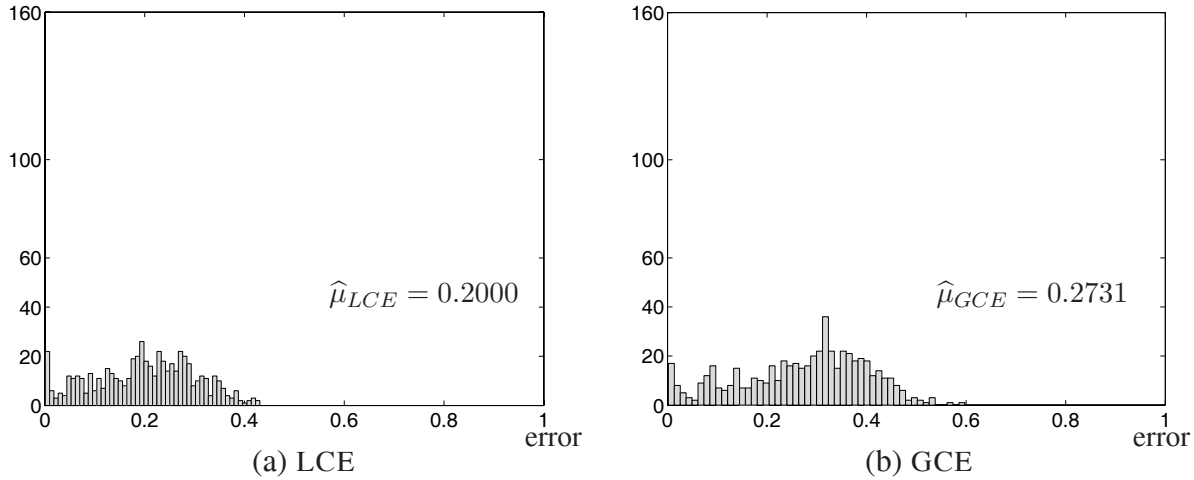
can note that these method produce different segmentation results, as well. In Table 7.1 the histogram mean values of LCE and GCE error are summarized. Note that different decimation strategies have similar error values, indicating that the segmentation results does not depend on the chosen decimation strategy.

In order to analyse how produced region sizes vary from one method to the other and how this variation depends on the content of the segmented images, we have normalized the size of each region by dividing it to the size of the segmented image it belonged to (number of pixels), and for each segmentation, we have calculated the standard deviation ($\sigma_S$) of the normalized region sizes. For the case of human segmented images, we have done separately the calculation for each segmentation and taken the mean of the results for the segmentations of the same image. Figure 7.25a shows the resulting $\sigma_S$ for 70 images (a clear majority for which the $\sigma_S$




(a) LCE (b) GCE

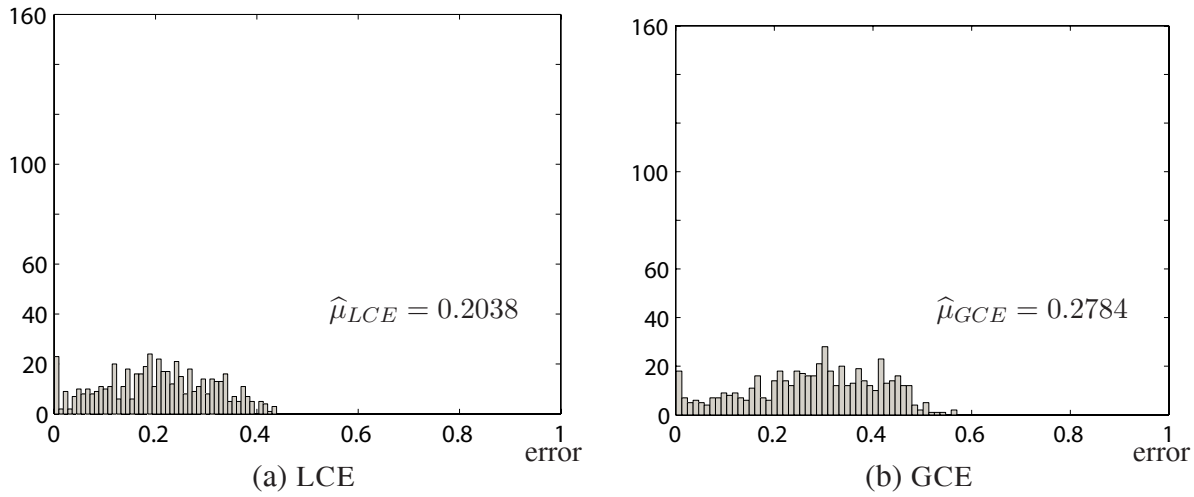
**Figure 7.19:** Histograms of discrepancy measure: NCutSeg versus Human.




(a) LCE (b) GCE

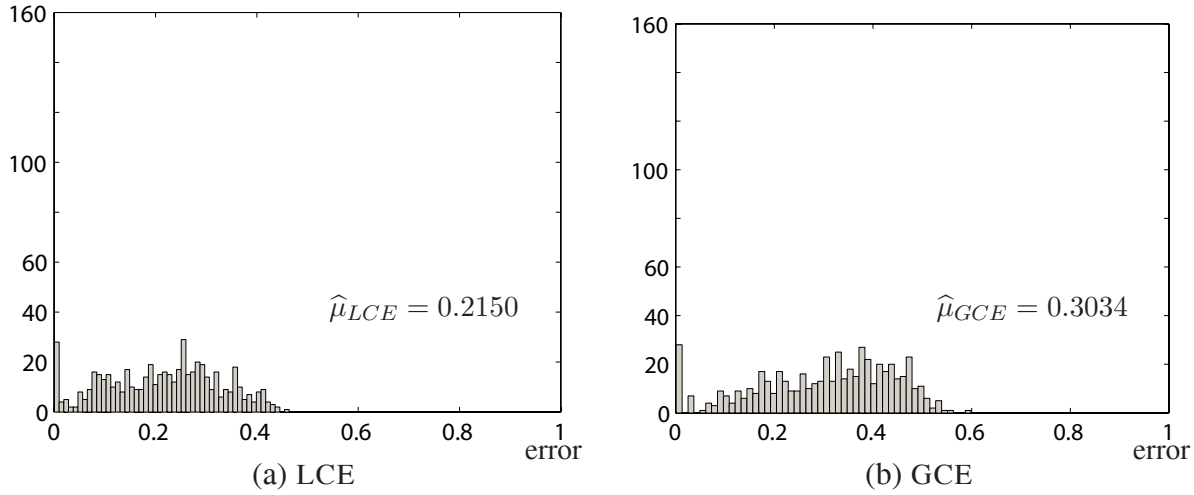
**Figure 7.20:** Histograms of discrepancy measure: BorůSeg (MIS) versus Human.

order Humans>BorůSeg>NCutSeg existed). Results are shown sorted by the sum of the 3 $\sigma_S$ for each image. The average region size variation for the whole dataset is: $0.1537$-Humans, $0.0392$-NCutSeg, and $0.0872$-BorůSeg (MIES). Note, that the size variation is smallest and almost content independent for the NCutSeg and largest for Humans. We also calculated the variation of regions sizes of different decimation strategies MIS and D3P. The average region size variation for the whole data set is $0.0893$ for BorůSeg (MIS) and $0.1037$ for BorůSeg (D3P). One can produce three plot, one for each decimation strategies MIS, MIES, and D3P. In order not to overload the figure with too many plots, we show in Figure 7.25b, a solid line representing the mean region size variation of the three decimation strategies MIES, MIS, and D3P, and the doted line the standard deviation.




(a) LCE (b) GCE

**Figure 7.21:** Histograms of discrepancy measure: BorůSeg (MIES) versus Human.




(a) LCE (b) GCE

**Figure 7.22:** Histograms of discrepancy measure: BorůSeg (D3P) versus Human.

## 7.5 Conclusion

In this chapter we have evaluated segmentation results of three graph-based methods; the well known method based on the normalized cuts (NCutSeg) and two methods based on the minimal spanning tree principle (BorůSeg and KrusSeg). The NCutSeg method and the BorůSeg are compared with human segmentations. The BorůSeg is compared to KrusSeg. The evaluation is done by using discrepancy measures, that do not penalize segmentations that are coarser or more refined in certain regions. We used only gray images to evaluate the quality of results on one feature. In Figure 7.18, 7.19, 7.20, 7.21, and 7.22 you can see the histograms of the GCE and LCE values obtained, Human versus Human, NCutSeg versus Humans, and BorůSeg versus Human. The NCutSeg and BorůSeg segmentation methods have not proved to be as efficient

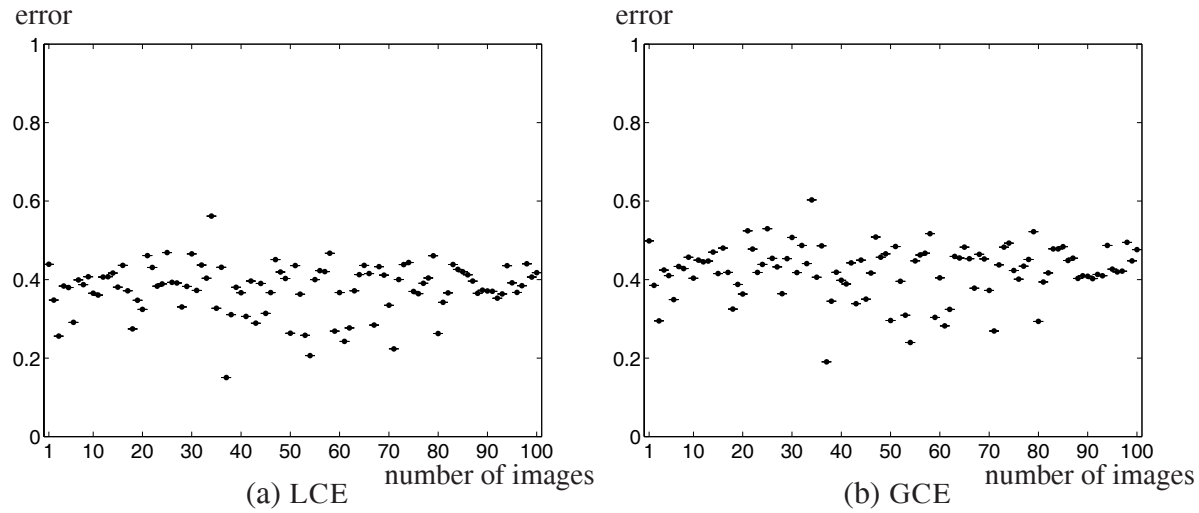



**Figure 7.23:** Error measure results: BorůSeg versus KrusSeg.

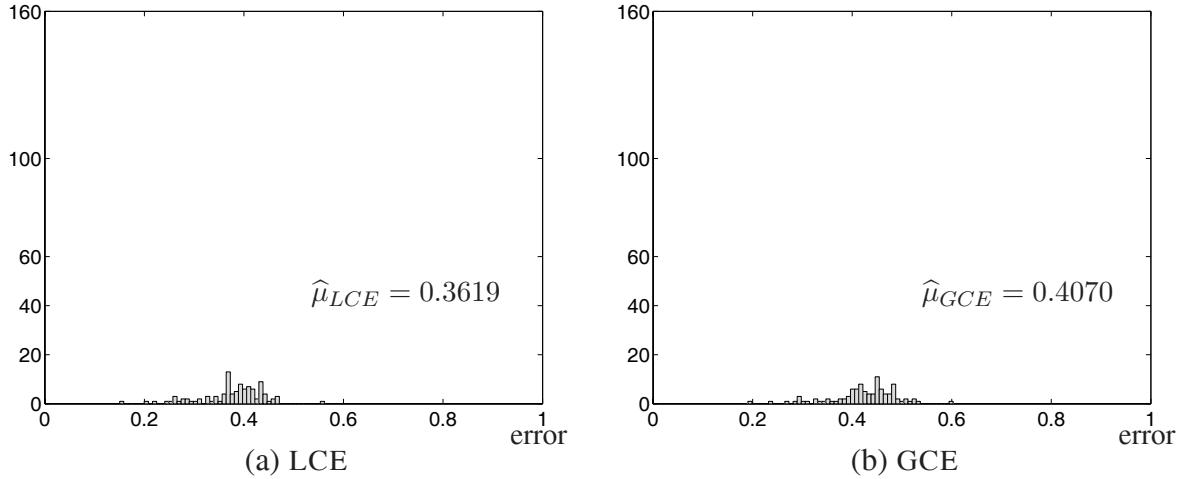
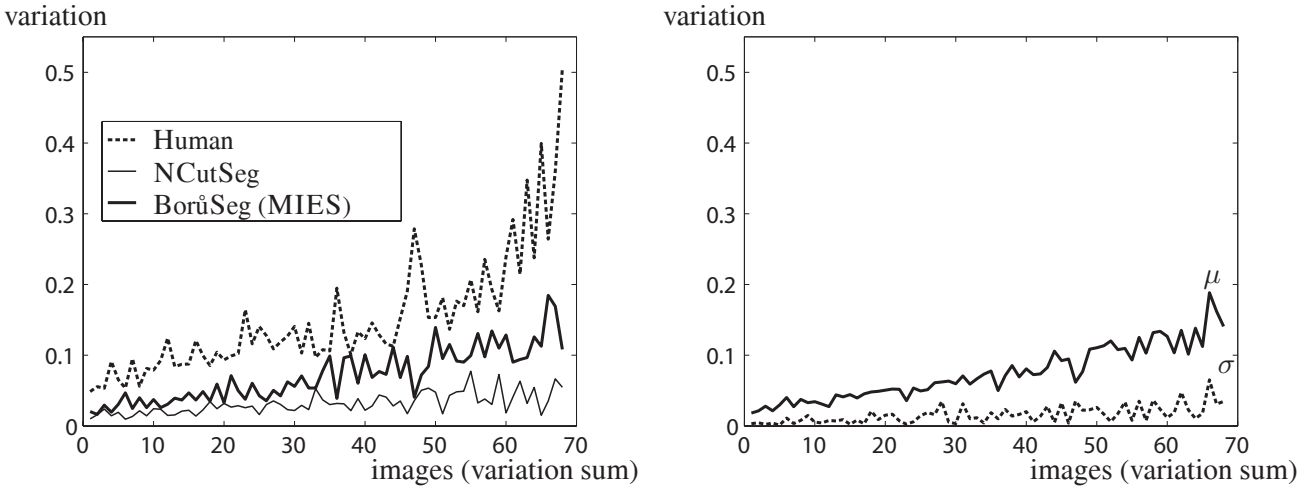



**Figure 7.24:** Histograms of discrepancy measure: BorůSeg versus KrusSeg.

as the humans, but one can see that, for both the error measure results are concentrated in the lower half of the output domain and that the mean of the GCE measure, which is stronger than LCE, is for both around the value of $0.2$. We have observed that the results produced by the BorůSeg versus KrusSeg methods have shown a considerable difference. Moreover different decimation strategies used in BorůSeg have shown same error results. One can say that for image segmentation choosing any of the decimation strategies will produce satisfiable results. This evaluation can be used to find classes of images for which the algorithms have segmentation problems. We plan to use in a larger image database in order to to confirm the quality of the obtained results, and do the evaluation with additional low level cues (color and texture) as well as different statistical measures.




(a) region size variation

(b) $\mu$(MIES+MIS+D3P), and $\sigma$(MIES+MIS+D3P) of region size variation

**Figure 7.25:** Variation of region sizes $\sigma_S$.

CHAPTER 8

# Epilogue

> ” It is actually impossible in theory to determine exactly what the hidden mechanism is without opening the box, since there are many different mechanism with identical behavior. Quite apart from this, *analysis* is more difficult than invention in the sense in which, generally, induction takes more time to perform than deduction: in induction one has to search for the way, whereas in deduction one follows a straightforward path. ”
>
> **“Vehicles - Experiments in Synthetic Psychology” by Valentino Braitenberg.**

## 8.1 Conclusion

The main goal of many approaches in computer vision is to *make machines to see*. Delineating, detecting and recognizing objects in complex scenes using computers is complex and time consuming process. One way to deal with a complex and time consuming processes is to reduce input data without loosing important information, through abstraction. Hierarchical organization are very well suited to cope with complex and time consuming processes, since they facilitate data abstraction. Hierarchical structures allow the transformation of local into a global information. The main advantage of regular hierarchical structures is rapid computation of global information in a recursive way, because of their fixed height, i.e. logarithmic height. The regular pyramid lack shift invariance and do not preserve connectivity of objects, because of the fixed vertical structure. These drawbacks made regular pyramid to be rejected as general segmentation methods. To cope with these drawbacks, the vertical structure of pyramids should be adapted on input data. This implies, that the irregular pyramids loose the good property of rapid computation since they do not have logarithmic height. In this thesis we show

that it is possible to bound irregular pyramid, i.e. the irregular pyramids are logarithmically tapered. We introduce two new iteratively local method for selecting contraction kernels, the maximal independent edge set (MIES) and maximal independent directed edge set (MIDES). We experimentally show that:

- the stochastic decimation method (MIS) and data driven decimation process (D3P) do not lead to logarithmic tapering graph pyramids, as opposed to our methods (MIES, and MIDES).

We introduce also an efficient image partition method into this irregular graph pyramid. This method is based on minimum spanning tree principle and thus is time efficient. Even thought this method makes greedy decision during merging process it is able to capture important perceptually groupings. In fact the method is able to handle regions with high variability of intensity e.g. bush, hair etc. as well as regions with low intensity variability, e.g. grass, house etc. We evaluate the quality of segmentation results of MIES, MIS and D3P versions of this method with respect to humans and to other graph-based methods. The evaluation shows that

- in image segmentation different stochastic decimations used in MST image partitioning method produce comparable results.

The evaluation shows that different flavors (MIES, MIS, and D3P) of this method are comparable to each other as well as to other graph-based methods. It is shown that the quality of the segmentation result of none of the methods is significantly better.

## 8.2 Contribution and Evaluations

The work in this thesis concentrated mostly in hierarchical representation called irregular graph pyramid. We have studied in details many aspects of the irregular graph pyramid. This resulted into new insights and algorithms. Main contributions of this work are:

- Maximal independent vertex set method (MIS) does not guarantee a logarithmic tapered pyramid,
- Two new iteratively local method for selecting contraction kernels lead to logarithmic tapered pyramid:
  - Maximal independent edge set (MIES), and
  - Maximal independent directed edge set (MIDES),
- Comparison of different selection methods (MIS, MIES, MIDES, and D3P),
- Novel efficient hierarchical image partitioning method based on Borůvka's minimum spanning tree,
  - different version of this algorithm by using different selection methods: MIES, MIS, and D3P.
- Evaluation of different graph-based segmentation method with respect to humans
  - Kruskal's and Borůvka's (using MIES, MIS and D3P) minimum spanning tree,

  - different stochastic decimation method produce comparable segmentation results.

Some topics have not been studied in detail, thus some points for further research are:

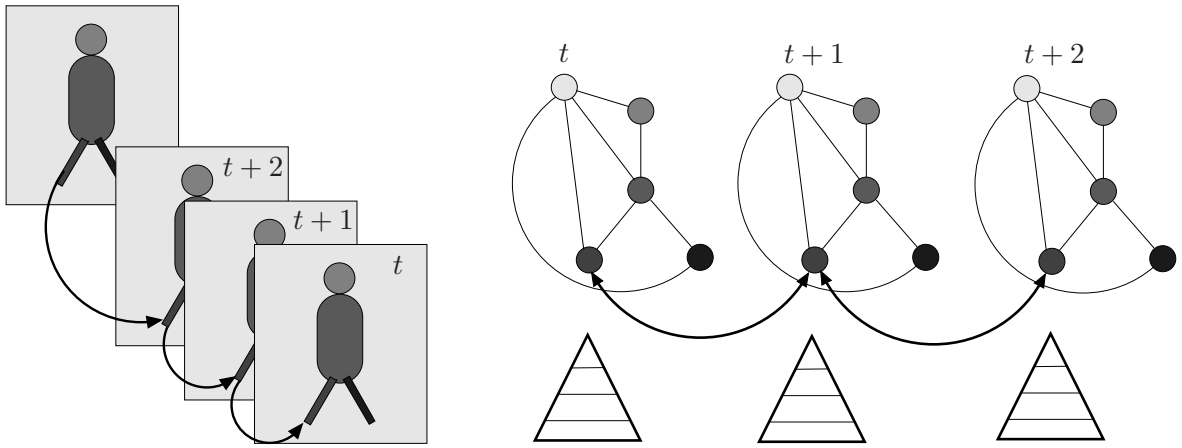
- Comparison of the decimation methods (MIS, MIES, MIDES, and D3P) with the decimation method based on Hopfield neural network [Bischof, 1995],
  - evaluation of the segmentation of Hopfield neural network decimation method.
- Comparison with other pyramid methods [Jolion and Montanvert, 1992, Brun and Kropatsch, 2003a]
- Single segmentation by selecting vertices on different levels of the pyramid like in [Lallich et al., 2003],
- Theoretical investigation of the similarity of the minimum spanning tree segmentation method and watersheds,
- Non-parametric hierarchical image partition method by using as input an over-segmentation (e.g. watershed segmentation),
- Using different minimum spanning tree algorithms for segmentations (e.g. randomized method [Karger et al., 1995] and using *consensus vision* [Cho and Meer, 1997] to produce more robust hierarchical image partitioning),
- Theoretical investigation of graph vector spaces and dual graph contraction on higher dimension.

These are many other things that can be envisioned to enhance the proposed algorithms. One long term topic is however of great interest: *the graph matching problem*. In the section below some ideas on this topic are discussed.

## 8.3 Outlook

Many approaches in computer vision problems like *segmentation*, *tracking* and *stereo vision* do not take structure into consideration. It is to expect that structure would be very useful in these problems. For example in tracking, finding object correspondences in image sequences is not a trivial problem. While in many computer vision applications the tasks segmentation, object detection and tracking are often solved stepwise, the idea is to determine a *structure* within the observed scene that is tracked over time. The structure is represented e.g. in a graph or more generally graph pyramid representation of the segmented image and correspondence between frames (not necessary subsequent) can be found for example by graph matching (see Figure 8.1).

In general a (sub) graph matching problem is NP-complete. A detailed overview on this topic can be found in [Bunke, 2000]. The classical algorithm for graph and subgraph isomorphism detection is the one by Ullman [Ullmann, 1976]. Methods for error-tolerant graph matching based on the $A^*$ search procedure are studied in [Sanfeliu and Fu, 1983]. These methods incorporate various heuristics and look-ahead techniques in order to prune the search space. All of these methods guarantee to find the optimal solution, unfortunately with exponential time and space computational cost, because of the NP-completeness of the problem. It




(a) Image Sequence (b) Graph pyramid representation

**Figure 8.1:** Representation of 2D image structure as graph pyramid in an image sequence.

is possible to find a solution in polynomial time by using sub-optimal (approximate) methods. A wide range of algorithms are used for solving this sub-optimal matching problem, like probabilistic relaxation [Kittler et al., 1993]; or continuous optimization methods: Hopfield neural networks [Feng et al., 1994] and Kohonen maps [Xu and Oja, 1990]; genetic search [Wang et al., 1997]; or Tabu search [Williams et al., 1999]. All of these methods are based on a heuristic optimization function and therefore can easily end in local minima. Alternative approaches are based on eigenvalue decomposition [Umeyama, 1998], linear programming [Almohamed, 1993] or specialized work in matching trees in terms of maximum cliques [Pelillo et al., 1999]. However, due to their complexity, these algorithms cannot be used for graphs having very large sets of vertices ( $> 1000$). An approach to approximate the solution of graph matching is to work on the data, by reducing the number of vertices on graphs, for example by graph reduction strategies. One such strategy is the irregular graph pyramid (as shown in this thesis). The advantage of using a graph pyramid is that it would allow grouping of structures and hence simplify graph matching. Higher levels of the pyramid, containing fewer vertices, can be efficiently matched. This matching can then be used to guide the matching of lower levels of the pyramid. In [Pailloncy et al., 1998, Pelillo et al., 1999, Glantz et al., 2004] graph hierarchies of graphs are used to solve the matching problem in a coarse to fine strategy. Moreover, the graph representation allows the detection and correction of over- and under-segmentation and therefore leads to a new representation of the scene structure.

The first step in structural approach is the representation of the image as a graph (as shown in Chapter 4 or combinatorial map [Brun and Kropatsch, 2001b]). The most common way of doing this is to first segment the image (e.g. by the method presented in Chapter 6 or watersheds [Tremeau and Colantoni, 2000]) and build a region adjacency graph (or graph pyramid). An alternative approach is to extract significant features (like SWIFT [Lowe, 2004] or MSER [Matas et al., 2002]) from the image and represent their characteristics and spatial distribution as a graph. A problem is, however, determining spatial relationships between these features which lead to a useful set of edges connecting the vertices. With a segmentation shown in Chapter 6, these relationships are available based on the neighborhood relationships of the

resulting regions.

The fact that we have motion information can already be used at this early stage to improve the segmentation. Combination of motion measurement with image segmentation can result in better analysis of motion. The knowledge of spatial partition can improve the reliability of motion-based segmentation [Gelgon and Bouthemy, 1999]. Temporal tracking of a spatial partition of an image, from the motion-based segmentation, is easily done than if spatial regions are tracked individually [Gelgon and Bouthemy, 1999]. Sequences can be segmented using spatial information and motion cues.

Using matching of graphs of two successive frames allows objects of interest to be followed along the sequence. Graph matching avoids the costly motion estimation and compensation, and moreover the graph representation permits objects (vertices) which are not visible anymore in the scene to be retained in memory and to be recognized correctly when they reappear. Unfortunately, there is no one-to-one mapping between vertices; if both partitions belong to a sequence, moving objects change their structures or even the global number of vertices due to temporal occlusion. Thus, some question must be answered to overcome drawbacks of already existing methods:

- How can an update of the scene representation (object model) be accomplished?
- Are the existing segmentation and graph matching methods under the assumption of structural search fast enough to allow real time performance?
- What kind of influences have to be considered in case of drastic scene changes (i.e. object changes, illumination changes etc.)?
- How can coarse-to-fine approaches be realized (e.g. using under-segmentations)?

It is possible to extend this idea to track higher level, such as faces and their relations, by creating graphs with vertices representing these features. These graph could be created by lower lying graphs and information from images, yielding to the concept of a ‘pyramid of graphs’.

The information representation embodied by the graphs can be used to find correspondences between subsequent images in a sequence and thereby to track objects. This should allow an integration of the segmentation into the tracking process, a problem which is not solved yet. The use of graphs in this task is a promising approach. This would mean also finding answers to graph matching problem.

# Copyright Acknowledgment

Part of results shown in Chapter 5 are previously published in the German Pattern Recognition Symposium (DAGM) [Haxhimusa et al., 2002] and in the International Workshop on Graph-based Representations in Pattern Recognition [Haxhimusa et al., 2003]. This results are compiled in the journal publication for Pattern Recognition Letters [Kropatsch et al., 2005].

Some of results in Chapter 6 and Chapter 7 are published in the German Pattern Recognition Symposium [Haxhimusa and Kropatsch, 2003], in the International Workshop on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition (SSPR and SPR) [Haxhimusa and Kropatsch, 2004], in the IS&T/SPIE Annual Symposium Computational Imaging [Kropatsch and Haxhimusa, 2004], in the Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition [Haxhimusa et al., 2005a], in the International Conference on Computer Analysis of Images and Patterns [Haxhimusa et al., 2005b], in the International Conference on Pattern Recognition [Haxhimusa et al., 2006a] and in the Iberoamerican Symposium on Pattern Recognition [Haxhimusa et al., 2006b].

# APPENDIX A

# Vector Spaces

**Summary** In this appendix some basics concepts from algebra, namely the group and the filed concept are recalled. These concepts will help in defining a vector space on a graph in Chapter 4. With the help of the vector space on graphs the prove of existence of the dual graph of an planar one is strait forward.

## A.1 Groups and Fields

A set $K$ with a defined binary operation of addition $\oplus$, is a *group* if these conditions are satisfied:

1. $X$ is *closed* under $\oplus$, i.e.
$$\forall x, y \in X,\ x \oplus y \in X,$$
2. the addition operation $\oplus$ is *associative*, i.e
$$\forall x, y, z \in X,\ (x \oplus y) \oplus z = x \oplus (y \oplus z),$$
3. $\exists e_{X_\oplus} \in X$, called *identity*[1] element under addition, if
$$\forall x \in X, x \oplus e_{X_\oplus} = e_{X_\oplus} \oplus x = x,$$
4. $\exists i_{X_\oplus} \in X$, called *inverse* element of $x$ under addition, if
$$\forall x \in X,\ x \oplus i_{X_\oplus} = i_{X_\oplus} \oplus x = e_{X_\oplus},\ i_{X_\oplus}.$$

A group $X$ is called *abelian* if it also fulfills:

5. the addition operation $\oplus$ is *commutative*, i.e.
$$\forall x, y \in X,\ x \oplus y = y \oplus x.$$

An important group when studying graphs is the set $Z_p = \{0, 1, \ldots, p-1\}$ of integers with modulo $p$ addition operation, i.e. $x = y(\text{modulo } p)$ if $x = m * p + y$ for $m \in \mathbb{I}$ and $y \in Z_p$. It can be shown easily that this set is an abelian group.

A set $K$ with defined binary operation of addition $\oplus$ and of multiplication $\odot$, is a *field* if these conditions hold:

[1] Called also neutral element.

**Table A.1:** Modulo 2 operations of addition and multiplication.

| $\oplus$ | | | | |
|---|---|---|---|---|
| $0$ | $\oplus$ | $0$ | $=$ | $0$ |
| $0$ | $\oplus$ | $1$ | $=$ | $1$ |
| $1$ | $\oplus$ | $0$ | $=$ | $1$ |
| $1$ | $\oplus$ | $1$ | $=$ | $0$ |

| $\odot$ | | | | |
|---|---|---|---|---|
| $0$ | $\odot$ | $0$ | $=$ | $0$ |
| $0$ | $\odot$ | $1$ | $=$ | $0$ |
| $1$ | $\odot$ | $0$ | $=$ | $0$ |
| $1$ | $\odot$ | $1$ | $=$ | $1$ |

i. the set $K$ is an abelian group under $\oplus$, with the identity element $e_{K_\oplus}$,

ii. the set $K - \{e_{K_\oplus}\}$ is an abelian group under $\odot$,

iii. the multiplication operation is distributive with respect to addition operation, i.e.

$$\forall x, y, z \in K,\ x \odot (y \oplus z) = (x \oplus y) \odot (x \oplus z).$$

The set $Z_p = \{0, 1, \ldots, p-1\}$ under the addition (modulo $p$) is an abeilan group, with $0$ as identity element. Set $Z_p - \{0\}$ is also an abelian group under multiplication (modulo $p$), if $p$ is a prime. The distributive law is proved simply, therefore $Z_p$ is a field iff $p$ is prime. This kind of fields are called *Galois field* and are denoted by $\mathbb{F}_p$.

A field that is used to create vector space on graph is the $\mathbb{F}_2 = \{0, 1\}$, the set of integers modulo $2$, on which the modulo $2$ addition and modulo $2$ multiplication operation are defined as given in Table A.1[2].

## A.2 Vector Space

Having the definition of the field, the vector space can now be defined. A set $H$ over the field $K$, with (vector) addition $\boxplus$ and a (scalar) multiplicative operation $\boxdot$, such that

$$\forall a \in K \text{ and } \forall \mathbf{x} \in H,\ \text{then}\ a \boxdot \mathbf{x} \in H,\ \text{i.e.}\ K \times H \rightarrow H,$$

is called a *vector space*[3] over the field $K$ if the following axioms are satisfied $\forall a,\ b \in K$ and $\forall \mathbf{x},\ \mathbf{y} \in H$:

I. $H$ is abelian group under $\boxplus$,

II. the scalar multiplication $\boxdot$ is distributive over vector addition $\boxplus$ that is

$$a \boxdot (\mathbf{x} \boxplus \mathbf{y}) = (a \boxdot \mathbf{x}) \boxplus (a \boxdot \mathbf{y}),$$

III. the scalar multiplication $\boxdot$ is distributive over scalar addition $\oplus$ i.e.

$$(a \oplus b) \boxdot \mathbf{x} = (a \boxdot \mathbf{x}) \boxplus (b \boxdot \mathbf{x}),$$

[2]The table resembles the XOR and AND function, respectively.

[3]The elements of $H$ are called vectors, and of $K$ are called scalars, respectively.

IV. scalar multiplication $\boxdot$ is associative, that is

$$(a \odot b) \boxdot \mathbf{x} = a \boxdot (b \boxdot \mathbf{x}), \text{ and}$$

V. $\forall \mathbf{x} \in H$, $e_{K_\odot} \boxdot \mathbf{x} = \mathbf{x}$, where $e_{K_\odot}$ is the multiplicative identity in $K$.

In order to define a vector space $H$ over the field $K$, one has to define vector addition and scalar multiplication in $H$ and to fulfill the five axioms above.

An important case of vector spaces is the set of all $n$-vectors over the field $K$, such that each vector $\mathbf{x} = (a_1, a_2, \ldots, a_n)$ of this vector space is a vector with $n$ elements $a_i$ from the field $K$. Let $\oplus$ and $\odot$ be the addition and multiplication and $e_{K_\oplus}$ and $e_{K_\odot}$ additive and multiplicative identities in $K$. Let this vector space be $R$, and let vector addition $\boxplus$ and scalar multiplication $\boxdot$ be defined as:

1. $\forall\, \mathbf{x} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{y} = (b_1, b_2, \ldots, b_n) \in R$

$$\mathbf{x} \boxplus \mathbf{y} = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n), \tag{A.1}$$

2. $\forall a \in K$

$$a \boxdot \mathbf{x} = (a \odot a_1, a \odot a_2, \ldots, a \odot a_n). \tag{A.2}$$

In order to show that $R$ is vectors space over $K$, one has to prove that $R$ is abelian group under $\boxplus$. Since $K$ is a field then it is closed under $\oplus$ and by the definition of $\boxplus$, $R$ is closed also under $\boxplus$ as well (axiom 1). The $\oplus$ operation is associative, hence for all $\mathbf{x} = (a_1, a_2, \ldots, a_n)$, $\mathbf{y} = (b_1, b_2, \ldots, b_n)$, and $\mathbf{z} = (c_1, c_2, \ldots, c_n) \in R$:

$$\begin{aligned}
(\mathbf{x} \boxplus \mathbf{y}) \boxplus \mathbf{z} &= (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n) \boxplus \mathbf{z} &= \\
&= ((a_1 \oplus b_1) \oplus c_1, (a_2 \oplus b_2) \oplus c_2, \ldots, (a_n \oplus b_n) \oplus c_n) &= \\
&= (a_1 \oplus (b_1 \oplus c_1), a_2 \oplus (b_2 \oplus c_2), \ldots, a_n \oplus (b_n \oplus c_n)) &= \\
&= (a_1 \oplus (b_1 \oplus c_1), a_2 \oplus (b_2 \oplus c_2), \ldots, a_n \oplus (b_n \oplus c_n)) &= \\
&= \mathbf{x} \boxplus (b_1 \oplus c_1, b_2 \oplus c_2, \ldots, b_n \oplus c_n) &= \\
&= \mathbf{x} \boxplus (\mathbf{y} \boxplus \mathbf{z}),
\end{aligned}$$

So the $\boxplus$ is also associative (axiom 2). Let $e_{R_\boxplus}$ be the identity vector under $\boxplus$. Since there exists identity element $e_{K_\oplus} \in K$ for the addition,

$$\begin{aligned}
\mathbf{x} \boxplus \mathbf{e}_{R_\boxplus} &= \mathbf{x} &= \\
&= (a_1, a_2, \ldots, a_n) &= \\
&= (a_1 \oplus e_{K_\oplus}, a_2 \oplus e_{K_\oplus}, \ldots, a_n \oplus e_{K_\oplus}) &= \\
&= (a_1, a_2, \ldots, a_n) \boxplus (e_{K_\oplus}, e_{K_\oplus}, \ldots, e_{K_\oplus})_{1\times n} &= \\
&= \mathbf{x} \boxplus (e_{K_\oplus}, e_{K_\oplus}, \ldots, e_{K_\oplus})_{1\times n}.
\end{aligned}$$

Hence the identity vector $\mathbf{e}_{R_\boxplus} = (e_{K_\oplus}, e_{K_\oplus}, \ldots, e_{K_\oplus})_{1\times n}$. By the same procedure one proves also that $\mathbf{e}_{R_\boxplus} \boxplus \mathbf{x} = \mathbf{x}$. So it is proved that there exists an identity vector for $\boxplus$ (axiom 3). There is an inverse element $i_{K_\oplus}$ of the element $a \in K$ under $\oplus$ then it can be written:

$$\begin{aligned}
(e_{K_\oplus}, e_{K_\oplus}, \ldots, e_{K_\oplus})_{1\times n} &= (a_1 \oplus i_{K_\oplus 1}, a_2 \oplus i_{K_\oplus 2}, \ldots, a_n \oplus i_{K_\oplus n}) &= \\
&= (i_{K_\oplus 1} \oplus a_1, i_{K_\oplus 2} \oplus a_2, \ldots, i_{K_\oplus n} \oplus a_n).
\end{aligned}$$

If inverse vector is written as $\mathbf{i}_{R_{\boxplus}} = (i_{K_{\oplus}1}, i_{K_{\oplus}2}, \ldots, i_{K_{\oplus}n})$ then it holds

$$\mathbf{e}_{R_{\boxplus}} = \mathbf{x} \boxplus \mathbf{i}_{R_{\boxplus}} = \mathbf{i}_{R_{\boxplus}} \boxplus \mathbf{x},$$

with which the axiom 4 is proved. Finally, since $\oplus$ is commutative than

$$\begin{aligned} \mathbf{x} \boxplus \mathbf{y} &= (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n) = \\ &= (b_1 \oplus a_1, b_2 \oplus a_2, \ldots, b_n \oplus a_n) = \\ &= \mathbf{y} \boxplus \mathbf{x}. \end{aligned}$$

All five axioms are proved, therefore $R$ is abelian group, and fulfills the axiom I of being a vector space. Let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in R$ and $a, b \in K$. From the definition of scalar multiplication and vector addition follows:

$$\begin{aligned} a \boxdot (\mathbf{x} \boxplus \mathbf{y}) &= a \boxdot (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n) = \\ &= (a \odot (a_1 \oplus b_1), a \odot (a_2 \oplus b_2), \ldots, a \odot (a_n \oplus b_n)) = \end{aligned}$$

and since $K$ is a field then $\odot$ is distributive over $\oplus$, hence

$$\begin{aligned} &= ((a \odot a_1) \oplus (a \odot b_1), (a \odot a_2) \oplus (a \odot b_2), \ldots, (a \odot a_n) \oplus (a \odot b_n)) = \\ &= (a \odot a_1, a \odot a_2, \ldots, a \odot a_n) \oplus (a \odot b_1, a \odot b_2, \ldots, a \odot b_n) = \\ &= (a \boxdot \mathbf{x}) \boxplus (a \boxdot \mathbf{y}). \end{aligned}$$

Therefore the axiom II of vector space is satisfied. Also based on same conditions the axiom III is proved:

$$\begin{aligned} (a \oplus b) \boxdot \mathbf{x} &= ((a \oplus b) \odot a_1, (a \oplus b) \odot a_2, \ldots, (a \oplus b) \odot a_n) = \\ &= ((a \odot a_1) \oplus (b \odot a_1), (a \odot a_2) \oplus (b \odot a_2), \ldots, (a \odot a_n) \oplus (b \odot a_n)) = \\ &= (a \odot a_1, a \odot a_2, \ldots, a \odot a_n) \boxplus (b \odot a_1, b \odot a_2, \ldots, b \odot a_n) = \\ &= (a \boxdot \mathbf{x}) \boxplus (b \boxdot \mathbf{x}). \end{aligned}$$

It follows the prove for the axiom IV:

$$\begin{aligned} (a \odot b) \boxdot \mathbf{x} &= ((a \odot b) \odot a_1, (a \odot b) \odot a_2, \ldots, (a \odot b) \odot a_n) = \\ &= (a \odot (b \odot a_1), (a \odot (b \odot a_2), \ldots, a \odot (b \odot a_n) = \\ &= a \boxdot (b \odot a_1, b \odot a_2, \ldots b \odot a_n) = \\ &= a \boxdot (b \boxdot \mathbf{x}). \end{aligned}$$

and for a scalar $e_{K_{\odot}}$ as the multiplicative identity in $K$ the prove for the axiom V is:

$$\begin{aligned} e_{K_{\odot}} \boxdot \mathbf{x} &= (e_{K_{\odot}} \odot a_1, e_{K_{\odot}} \odot a_2, \ldots e_{K_{\odot}} \odot a_n) = \\ &= (a_1, a_2, \ldots a_n) = \\ &= \mathbf{x}. \end{aligned}$$

It is proved that all 5 needed axioms for a vector space are satisfied, therefore the set $R$ is vector space over the field $K$.

Next, some important properties of vector space are listed. Let a set $R$ be a vector space over the field $K$ of scalars, with vector addition $\boxplus$ and scalar multiplication $\boxdot$ defined. If a vector $\mathbf{x} \in R$ can be expressed as

$$\mathbf{x} = (a_1 \boxdot \mathbf{x}_1) \boxplus (a_1 \boxdot \mathbf{x}_1) \boxplus \ldots \boxplus (a_n \boxdot \mathbf{x}_n),$$

where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are vectors in $R$ and $a_1, a_2, \ldots, a_n$ scalars in $K$, than the vector $\mathbf{x}$ is a *linear combination* of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$. A set of vectors are *linearly independent* if no vector from this set can be stated as a linear combination of the others. If the set of vectors say $B = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ from $R$ are linearly independent and every vector $\mathbf{x}_i \in R \wedge \mathbf{x}_i \notin B, \forall i$ can be specified as the linear combination of the vectors from $I$, then $B$ is a *basis of the vector* space and the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ are the *basis vectors*. The number of basis vectors is the *dimension* of the vector space $R$. Finally,

**Proposition A.1** *A set $S$ of vectors from the vector space $R$ over the field $K$ is a vector subspace if $\forall\, \boldsymbol{s}_1, \boldsymbol{s}_2 \in S$ and $\forall\, a \in K$ the following condition are satisfied:*

1. $\boldsymbol{s}_1 \boxplus \boldsymbol{s}_2 \in S$,
2. $a \boxdot \boldsymbol{y} \in S$, *and*
3. $S \neq \emptyset$.

*Proof:* One has to prove that $S$ satisfies the axioms of the vector space. From the condition 1 and 2 the set $S$ is closed under vector addition and scalar multiplication. There exist an additive identity $e_{K_\oplus}$ and additive inverse $i_{K_\oplus}$ in $K$, such that for every $s \in S$, $e_{K_\oplus} \boxdot \mathbf{s} = \mathbf{e}_{S_\boxplus} \in S$ and $i_{K_\oplus} \boxdot \mathbf{s} = \mathbf{i}_{S_\boxplus}$ (using condition 2), where $\mathbf{e}_{S_\boxplus}$ is the vector additive identity and $\mathbf{i}_{S_\boxplus}$ is the vector additive inverse, respectively. Therefore for $S$ to be a subspace it must contain $\mathbf{e}_{S_\boxplus}$ (zero vector) and consequently $S$ is never empty. Since elements of $S$ are also elements of vector space $R$, other axioms of vector space are satisfied as well. Hence, it is proved that $S$ is a vector subspace over the field $K$. $\square$

Finally, the introduction of the *dot product* between two $n$-vectors is given. Let $\mathbf{x}_1 = (a_1, a_2, \ldots, a_m)$ and $\mathbf{x}_2 = (b_1, b_2, \ldots, b_m)$ in $R$ over the field $K$ be the two $n$-vectors, the dot product is specified as:

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = a_1 \odot b_1 \oplus a_2 \odot b_2 + \cdots + a_m \odot b_m.$$

If $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = e_{K_\oplus}$, where $e_{K_\oplus}$ is the additive identity in $K$, it is said that $\mathbf{x}_1$ and $\mathbf{x}_2$ are orthogonal to each other.

Let a simple example clarify the vector space concept. Let $R = \{(0,0), (0,1), (1,1)\}$ be the vector space of all 2-vectors over the field $\mathbb{F}_2 = 0, 1$, and let the addition $\boxplus$ and multiplication $\boxdot$ be defined as given in Equations A.1 and Equations A.2 and scalar addition $\oplus$ and scalar multiplication $\oplus$ as in Table A.1. For example vector $(0,0)$ and $(1,1)$ can be expressed as linear combination of $(0,1)$ and $(1,0)$

$$(0,0) = (0 \boxdot (0,1)) \boxplus (0 \boxdot (0,1)), \text{ and } (1,1) = (1 \boxdot (0,1)) \boxplus (1 \boxdot (0,1).$$

Vectors $(0,1)$ and $(1,0)$ are linearly independent and form the basis of $R$. It was shown above that the rest of the vectors can be stated as the linear combination of these two vectors. Vector space $R$ has 2 dimensions. The set $S = \{(0,0), (0,1), (1,1)\}$ and $S' = \{(0,0), (0,1)\}$ form subspaces of $R$, of dimensionality 2, respectively 1. Vectors $(0,1)$ and $(1,0)$ are orthogonal since $\langle (0,1), (1,0) \rangle = 1 \odot 0 \oplus 0 \odot 1 = 0$ (see Table A.1). Note that basis vectors are orthogonal.

The notions introduced in this appendix are used in Chapter 2, Section 2.7 in order to create a vector space on graphs over the field $\mathbb{F}_2$, which in turn will be helpful in defining a very important notion of dual graphs in Chapter 4, Section 4.2.

# APPENDIX B

# A Procedure for Constructing Dual Graphs

**Summary** In this appendix a procedure to construct a dual graph from a plane graph is described.

## B.1 Constructing the Dual Graph of a Plane Graph

Let a plane graph $G$ with vertex set $V$ and edge set $E$ be given. Let the elements of vertex set be denoted by $v_1, v_2, \ldots, v_m$ and of edge set by $e_1, e_2, \ldots, e_n$. Since the graph $G$ is a plane graph it is embedded in a plane (or a sphere), it separates the plane into regions. Let this regions be depicted with $f_1, f_2, \ldots, f_r$, and called faces. Note that there is one unbounded region (with infinite area), which will be called the background. Construction a dual graph $\overline{G}$ of $G$ is as follows:

**Algorithm 15 – Dual Graph Construction**

*Input*: Plane graph $G(V, E)$.
*Output*: Dual graph $\overline{G} = (\overline{V}, \overline{E})$.

1: Make a vertex $\overline{v}$ in $\overline{G}$ for each face $f_i, i = 1, \ldots, k$.
2: Place vertices $\overline{v}_i$ arbitrarily in each face of $G$.
3: If an edge in $G$ is common to two faces $f_i$ and $f_j$ (not necessarily distinct) then draw a line (or any arbitrary curve) connecting the vertices $\overline{v}_i$ and $\overline{v}_j$, so that it crosses the edge $e = (v_i, v_j)$. This line depicts the edge $\overline{e} = (\overline{v}_i, \overline{v}_j)$ of $\overline{G}$.

The number of edges in the dual graph $\overline{G}$ is the same as in $G$, since edges cross each other in these two graphs. Note also that an edge in $e \in G$ is boundary to at most two faces, and it is possible that an edge lays in one face as well, in this case the dual edge will be a self loop.

Figure B.1 is an illustration of the algorithm above. Not to overload the image by depicting all the elements, only some elements of $G$ and $\overline{G}$ are denoted, just to clarify the steps in the algorithm. The vertices and edges of the graph $G$ are drawn with circles and solid lines, respectively; whereas vertices and edges of the dual graph $\overline{G}$ with square and dotted lines, respectively.

For example in the faces $f_i$, $f_j$ and $f_k$ vertices $\overline{v}_i$, $\overline{v}_j$ and $\overline{v}_k$ of the dual graph $\overline{G}$ is placed arbitrarily (step 2 of the algorithm). The edge $e = (v_i, v_j)$ is a boundary between faces $f_i$ and

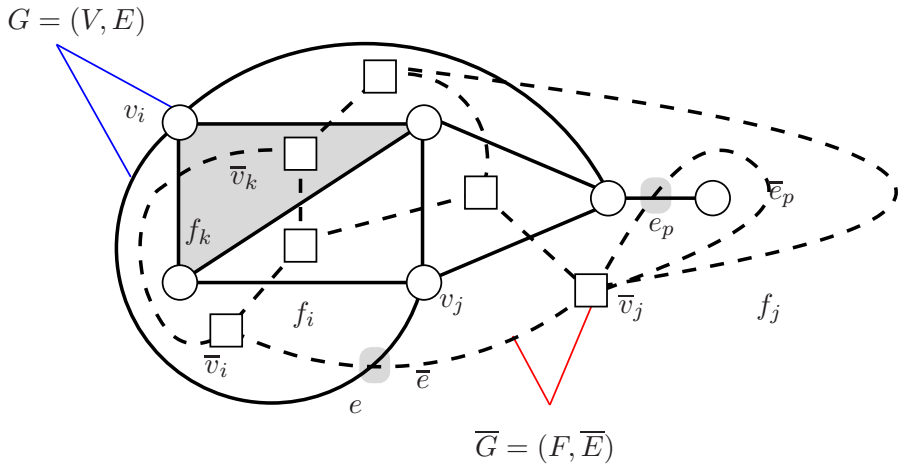



**Figure B.1:** A plane graph $G$ and it dual $\overline{G}$.

$f_j$, therefore an edge $\overline{e}$ is drawn between $\overline{v}_i$ and $\overline{v}_j$, such that it crosses the edge $e$ (the edge cross is shadowed). The edge $e_P$ is a pedant edge and lay in the face $f_j$ therefore its dual edge $\overline{e}_p$ is e self loop. Note that the $deg(\overline{v})$ is equal to the number of edges that bound its face. If a face is bounded by only two edges, i.e. by parallel edges the dual vertex has the degree 2.

Some notes about the notations. Since there is a one to one correspondence between the faces $f$ of the graph $G$ and the vertices in the dual graph $\overline{v} \in G$, these vertices are not explicitly differentiated from the faces $f$, if not otherwise stated, therefore the notation $f$ is used sometimes to denote the vertices $\overline{v}$ of the dual graph of $G$, and the set $F$ of the all the faces $f$ of $G$ represents the set $\overline{V}$. So the notation $\overline{G} = (\overline{V}, \overline{E})$ is sometimes written as $\overline{G} = (F, \overline{E})$.

# APPENDIX C

# Data Structures Representing Graphs

**Summary** Two standard data structures for representing graphs: the adjacency lists and the adjacency matrix are shown. The memory requirements for storing the adjacency lists representing a graph, as well as dual graphs is $\Theta(V + E)$.

## C.1 Representation of Graphs

Generally there are two standard data structures representing a graph $G = (V, E)$: an adjacency matrix and a collection of adjacency lists [Cormen et al., 2001]. The adjacency matrix is usually used for graphs with number of edges $|E| \approx |V|^2$, whereas adjacency list for graphs with number of edges much less than the square of the number of vertices, i.e. $|E| \ll |V|^2$.

### C.1.1 Adjacency Lists Structures

The adjacency list data structure of $G$ consist of an array $A_L$ of $|V|$ lists, that is for each vertex there is a list. A list $A_L[v]$ in this collection contains all the incident vertices $w$ on $v$, i.e. all those vertices which are joint by an edge $e = (v, w) \in E$. One can choose to store pointers to these vertices instead of vertices itself. The order in which these vertices are stored in a list is usually arbitrary.

This data structure is used to represent directed and undirected multigraphs, and can be easily modified to represent other graph types. Note that, in directed graphs the sum of length of all the adjacency list is $|E|$, whereas for undirected graphs is $2|E|$, since for an edge $e = (v, w)$, vertex $v$ appears in $w$'s adjacency list and vice versa. The amount of memory used for storing the adjacency list data structure is $\Theta(V + E)$[1], for both directed and undirected graphs. A disadvantage of adjacency list is that there is no fast way to check if a given edge $e = (v, w)$ is in a graph, other that to search for $w$ in $A_L[v]$. The usage of a hash table for each array in $A_L[v]$ containing the vertices $w$ will softener the problem of fast access to the edges. More advanced data structures for the adjacency list are found in [Mehlhorn and Näher, 1999].

Adjacency list data structures can be used to represent weighted graphs as well. Let $w(e) = w(v, w)$ be a weight of the edge $e = (v, w) \in E$ taken from the weight function $w : E \to \mathbb{R}$.

[1] $\Theta$-notation used for asymptoticly tight bound, see [Cormen et al., 2001] for more details. Note that if $\Theta(f(n))$ implies $\mathcal{O}(f(n))$, the asymptotic upper bound.

This weight is simply stored in $v$'s adjacency list. An example of the adjacency list data structure for an undirected and directed graph is given in Figure C.1a.

There is also another data structures, the *incidence list*, where the edges are represented as an array of pairs of vertices (and the other data if necessary).

### C.1.2 Adjacency Matrix Structures

Let the vertices of a graph $G = (V, E)$ be numbered in some arbitrary way, say $1, 2, \cdots, |V|$. The adjacency matrix is a $|V| \times |V|$ matrix $A_M$ with elements:

$$a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i, j = 1, \ldots, |V|$$

Memory consumption for storing an *adjacency matrix* of a graph is $\Theta(V^2)$, independent of the number of edges. An example of adjacency matrix is shown in Figure C.1b. Note that for the undirected graph edge $(v, w)$ is the same as $(w, v)$, therefore the transpose matrix[2] $A_M^T = A_M$. Hence, the $A_M$ adjacency matrix is symmetric with respect to the main diagonal. So one can store only the upper (or lower) part of the adjacency matrix together with the diagonal, to reduce the memory usage.

Adjacency matrix data structures can be used also to represent weighted graphs. Similar to adjacency lists the weight $w(v, w)$ of the edge $e = (v, w)$ is stored as entry in row $v$ and column $w$ of the adjacency matrix. If an edge does not exist one puts a $0$ or $\infty$ value. If the graph is unweighted, this data structures has an advantage in storage, since instead of using a word of computer memory for each entry, per entries of the matrix one uses only one bit of memory. One major drawback of the adjacency matrix data structures is that they cannot uniquely represent a graph with parallel edges and double (or multiple) self-loops.

There are other matrix structures to represent graphs. The *incidence matrix* $I_M$ of size $|E| \times |V|$, is a matrix with elements:

$$i_{i,j} = \begin{cases} 1 & \text{if edge } i \text{ is incident with vertex } j, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i = 1, \ldots, |E| \text{ and } j = 1, \ldots, |V|$$

The memory amount required is $\Theta(|V||E|)$, and depends on the number of edges and vertices.

The *Laplace matrix* [3], $L_M$ is the matrix of size $|V| \times |V|$ with elements:

$$l_{i,j} = \begin{cases} deg(i) & \text{if vertex } i = j, \\ -1 & \text{if } \exists e = (i,j) \text{ and } i \neq j, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i, j = 1, \ldots, |V|$$

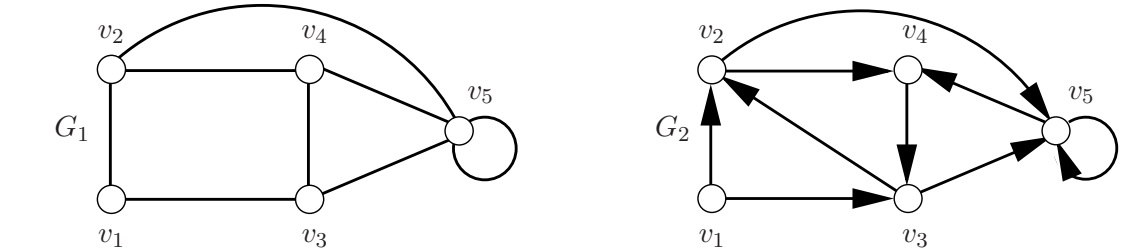
That is $L_M = D_M - A_M$, where $D_M$ is the degree matrix

$$d_{i,j} = \begin{cases} deg(i) & \text{if vertex } i = j, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i, j = 1, \ldots, |V|$$

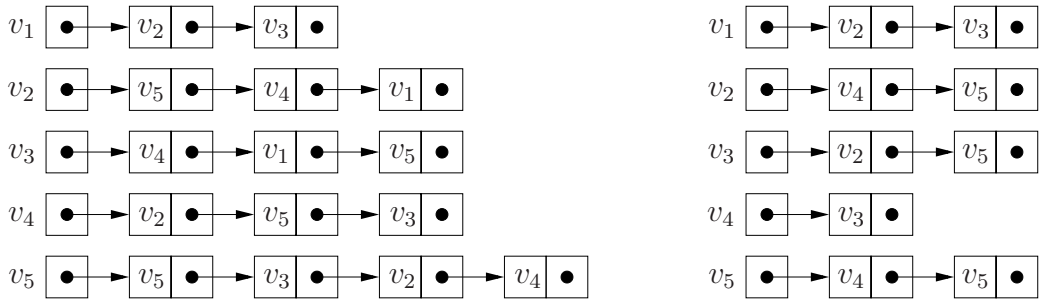
The memory requirement for this matrix structure is $\Theta(V^2)$. A detailed discussion of Laplace matrix for graph representation is found in [Chung, 1997].

[2] Transpose matrix $A^T = (a_{i,j}^T)$ of $A = (a_{i,j})$ is the one with elements $a_{i,j}^T = a_{j,i}$.

[3] Called also Kirchhoff matrix.




Undirected graph | Directed graph




(a) Adjacency lists

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| $v_1$ | 0 | 1 | 1 | 0 | 0 |
| $v_2$ | 1 | 0 | 0 | 1 | 1 |
| $v_3$ | 1 | 0 | 0 | 1 | 1 |
| $v_4$ | 0 | 1 | 1 | 0 | 1 |
| $v_5$ | 0 | 1 | 1 | 1 | 1 |

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| $v_1$ | 0 | 1 | 1 | 0 | 0 |
| $v_2$ | 0 | 0 | 0 | 1 | 1 |
| $v_3$ | 0 | 1 | 0 | 0 | 1 |
| $v_4$ | 0 | 0 | 1 | 0 | 0 |
| $v_5$ | 0 | 0 | 0 | 1 | 1 |

(b) Adjacency matrix

**Figure C.1:** Two usual data structures for representing graphs.

## C.2 Representation of Dual Graphs

In general, the graphs analyzed in this document are planar multi-graphs with multiple parallel edges and self-loops. Since the number of edges in analyzed graphs is $|E| \ll |V|^2$, the adjacency list is used as the data structure to represent graphs[4]. This data structure allows to represent not just graphs with weights on edges but also graphs with weights on vertices as well and it has a good property that in general the memory requirement is $\Theta(V + E)$.

Dual graphs $(G, \overline{G})$, are also represented by the adjacency list, with an additional look-up table. This look-up table relates edges in primary graph $G$ to edges in dual one $\overline{G}$. Therefore the memory consumption for this data structure (taking that $\Theta(V + E)$ is needed to store an adjacency list) is $\Theta(2(V + E))$ for dual graph and $\Theta(E)$ for the look-up table i.e. $\Theta(2(V + E)) + \Theta(E) = \Theta(V + E)$ [5]. In case of simple planar graph $G = (V, E)$ with $|V|$ vertices and

[4]For planar grid graph of size 100 by 100 there are 10000 vertices and approximately 40000 edges, it holds $40000 \ll 10000^2$.

[5]From the properties of $\Theta(k(f(n, n)) = \Theta(f(n, m))$,where $k$ is a constant and $\Theta(f(n, m)) + \Theta(g(n, m)) =$

$|E|$ edges it follows $|E| \leq 3|V| - 6$, thus the overall memory requirement is $\Theta(V)$. Note that the dual of a planar graph is also planar. In Chapter 4 a detailed theoretical treatment of dual graphs is given.

## C.3 Representation of Dual Graph Pyramids

A dual graph is defined as a stack of dual graphs $(G_k, \overline{G_k})$ for $k = 1, \ldots, h$, where $h$ is the height of the pyramid. The adjacency lists are used for representing dual graph and a look-up table to relate vertices of two subsequent levels in the pyramid. The overall memory consumption for the pyramids with the logarithmic height $h = log(|V|)$ (and a reduction factor of at least 2), is $\Theta((V + E)(1 + (1/2) + (1/4) + \cdots + (1/2^h))$ for the pyramid itself and $\Theta(V)$ for the look-up table i.e. the overall memory requirement is $\Theta(V + E)$ i.e. $\Theta((V + E) * (1 + 1/2 + 1/4 + \cdots + 1/\log|V|) + \Theta(V) = \Theta(2(V + E)) + \Theta(V) = \Theta(V + E)$. In case of simple planar graph it holds $|E| \leq 3|V| - 6$ then amount of memory it needs is $\Theta(V)$. Note that for a planar graph, the dual is also planar. In Chapter 4 a detailed theoretical treatment of dual graphs pyramids is given.

$\Theta(max(f(n, m), g(n, m)))$, see [Cormen et al., 2001] for details.

# APPENDIX D

# Names of Images on Berkley Image Database

**Summary** The file names of images from the Berkley Image Database, and their corresponding numbers used in figures in Chapter 7 are given.

## D.1 Corresponding Numbers of Images

In the table below the file names of the images from the Berkley Image Database [Martin et al., 2001] (without file extension), and their corresponding numbers are given. These numbers of images are depicted in Figures 7.13, 7.14, 7.15, 7.16, 7.17 and 7.23 in Chapter 7, Section 7.4. Images are found `www.cs.berkeley.edu/projects/vision/grouping/segbench/`

**Table D.1:** Image names and their corresponding numbers

| Number | Name | Number | Name | Number | Name | Number | Name | Number | Name |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 101085 | 21 | 148026 | 41 | 21077 | 61 | 299086 | 81 | 45096 |
| 2 | 101087 | 22 | 148089 | 42 | 216081 | 62 | 300091 | 82 | 54082 |
| 3 | 102061 | 23 | 156065 | 43 | 219090 | 63 | 302008 | 83 | 55073 |
| 4 | 103070 | 24 | 157055 | 44 | 220075 | 64 | 304034 | 84 | 58060 |
| 5 | 105025 | 25 | 159008 | 45 | 223061 | 65 | 304074 | 85 | 62096 |
| 6 | 106024 | 26 | 160068 | 46 | 227092 | 66 | 306005 | 86 | 65033 |
| 7 | 108005 | 27 | 16077 | 47 | 229036 | 67 | 3096 | 87 | 66053 |
| 8 | 108070 | 28 | 163085 | 48 | 236037 | 68 | 33039 | 88 | 69015 |
| 9 | 108082 | 29 | 167062 | 49 | 24077 | 69 | 351093 | 89 | 69020 |
| 10 | 109053 | 30 | 167083 | 50 | 241004 | 70 | 361010 | 90 | 69040 |
| 11 | 119082 | 31 | 170057 | 51 | 241048 | 71 | 37073 | 91 | 76053 |
| 12 | 12084 | 32 | 175032 | 52 | 253027 | 72 | 376043 | 92 | 78004 |
| 13 | 123074 | 33 | 175043 | 53 | 253055 | 73 | 38082 | 93 | 8023 |
| 14 | 126007 | 34 | 182053 | 54 | 260058 | 74 | 38092 | 94 | 85048 |
| 15 | 130026 | 35 | 189080 | 55 | 271035 | 75 | 385039 | 95 | 86000 |
| 16 | 134035 | 36 | 19021 | 56 | 285079 | 76 | 41033 | 96 | 86016 |
| 17 | 14037 | 37 | 196073 | 57 | 291000 | 77 | 41069 | 97 | 86068 |
| 18 | 143090 | 38 | 197017 | 58 | 295087 | 78 | 42012 | 98 | 87046 |
| 19 | 145086 | 39 | 208001 | 59 | 296007 | 79 | 42049 | 99 | 89072 |
| 20 | 147091 | 40 | 210088 | 60 | 296059 | 80 | 43074 | 100 | 97033 |

# Bibliography


[Ahronovitz et al., 1995] Ahronovitz, E., Aubert, J.-P., and Fiorio., C. (1995). The star-topology: A topology for image analysis. In Ahronovitz, E. and Fiorio, C., editors, $5^{th}$ *Discrete Geometry for Computer Imagery*, pages 107–116.

[Almohamed, 1993] Almohamed, H. (1993). A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:522–525.

[Alon et al., 1986] Alon, N., Babai, L., and Itai, A. (1986). A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583.

[Atallah, 1999] Atallah, M. J., editor (1999). *Algorithms and Theory of Computational Handbook*. CRC Press.

[Bartelma, 2004] Bartelma, J. (2004). Flycatcher: Fusion of gaze with hierarchical image segmentation for robust object detection. Master's thesis, Massachusetts Institute of Technology Media Laboratory, Cambridge, Massachusetts.

[Bartelma and Roy, 2006] Bartelma, J. and Roy, D. (2006). Flycatcher: Fusion of gaze with hierarchical image segmentation for robust object detection. In *submitted draft*.

[Bederson, 1992] Bederson, B. B. (1992). *A Miniature Space-variant Active Vision System.* PhD thesis, New York University, Courant Insitute, New York.

[Bertin et al., 1996] Bertin, E., Bischof, H., and Bertolino, P. (1996). Voronoi pyramids controlled by hopfield networks. *Computer Vision and Image Understanding*, 63(3):462–475.

[Biggs et al., 1976] Biggs, N. L., Lloyd, E. K., and Wilson, R. J. (1976). *Graph Theory 1736-1936.* Clarendon Press, Oxford.

[Bischof, 1995] Bischof, H. (1995). *Pyramidal Neural Networks.* Lawrence Erlbaum Associates.

[Bischof and Kropatsch, 1993] Bischof, H. and Kropatsch, W. G. (1993). Hopfield networks for irregular decimation. In Pölzleitner, W. and Wenger, E., editors, *Image Analysis and Synthesis*, volume 68, pages 317–327. OCG-Schriftenreihe, Österr. Arbeitsgemeinschaft für Mustererkennung, R. Oldenburg.

[Bister et al., 1990] Bister, M., Cornelis, J., and Rosenfeld, A. (1990). A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, 11(9):605–617.

[Bixby and Cunnigham, 1995] Bixby, R. E. and Cunnigham, W. H. (1995). Matroid optimization and algorithms. In Graham, R., Groetschel, M., and Lovazs, L., editors, *Handbook of Combinatroics*, pages 551–609. MIT Press, Elsevier Amsterdam.

[Bondy and Murty, 1976] Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications, 5th Ed.* Elevier Science Publishing Co. Inc.

[Borowy and Jolion, 1995] Borowy, M. and Jolion, J.-M. (1995). A pyramidal framework for fast feature detection. In *Proceedings of International Workshop on Parellel Image Analysis*, pages 193–202.

[Borůvka, 1926a] Borůvka, O. (1926a). O jistém problému minimálnim (about a certain minimal problem). *Práce Moravské Přírodvědecké Společnosti v Brně (Acta Societ. Scienc. Natur. Moravicae)*, 3(3):37–58.

[Borůvka, 1926b] Borůvka, O. (1926b). Příspěvek k řešení otázky ekonomickè stavby elektrovodnčh sìtì (contribution to the solution of a problem of economical construction of electrical networks). *Elekrotechnickỳ Obzor*, 15:153–154.

[Boykov et al., 1998] Boykov, Y., Veksler, O., and Zabih, R. (1998). Markov random fields with efficient approximations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–655, USA. IEEE Computer Society. Also as Cornell CS technical report TR97-1658, December 3, 1997.

[Braquelaire and Brun, 1998] Braquelaire, J.-P. and Brun, L. (1998). Image segmentation with topological maps and interpixel representation. *Journal of Visual Communication and Image Representation*, 9(1):62–79.

[Brun and Kropatsch, 2001a] Brun, L. and Kropatsch, W. G. (2001a). Contraction kernels and combinatorial maps. In Jolion, J.-M., Kropatsch, W. G., and Vento, M., editors, *Proceedings of IAPR Workshop on Graph-based Representations in Pattern Recognition*, pages 12–21. CUEN.

[Brun and Kropatsch, 2001b] Brun, L. and Kropatsch, W. G. (2001b). Introduction to combinatorial pyramids. In Bertrand, G., Imiya, A., and Klette, R., editors, *Digital and Image Geometry*, pages 108–128. Springer, Berlin, Heidelberg.

[Brun and Kropatsch, 2003a] Brun, L. and Kropatsch, W. G. (2003a). Construction of combinatorial pyramid. In Hancock, E. and Vento, M., editors, *4th IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition*, volume 2726 of *Lecture Notes in Computer Science*, pages 1–12, York, UK. Springer, Berlin Heidelberg, New York.

[Brun and Kropatsch, 2003b] Brun, L. and Kropatsch, W. G. (2003b). Contraction kernels and combinatorial maps. *Pattern Recognition Letters*, 24(8):1051–1057.

[Brun and Kropatsch, 2006] Brun, L. and Kropatsch, W. G. (2006). Contains and inside relationship within combinatorial pyramids. *Pattern Recognition*, 39(4):515–526.

[Bunke, 2000] Bunke, H. (2000). Recent developments in graph matching. In Sanfeliu, A., Villanueva, J., Vanrell, M., Alquezar, R., Jain, A., and Kittler, J., editors, *Proceedings of the International Conference on Pattern Recognition ICPR2000*, volume 2, pages 117–124. IEEE Computer Society.

[Burge and Kropatsch, 1999] Burge, M. and Kropatsch, W. G. (1999). A minimal line property preserving representation of line images. *Devoted Issue on Image Processing*, 62(4):355–368.

[Burt and Adelson, 1983] Burt, P. J. and Adelson, E. H. (1983). The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540.

[Burt et al., 1981] Burt, P. J., Hong, T.-H., and Rosenfeld, A. (1981). Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(12):802–809.

[Chazelle, 2000] Chazelle, B. (2000). A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the Association for Computer Machinery*, 47(6):1028 – 1047.

[Chen and Pavlidis, 1980] Chen, P. and Pavlidis, T. (1980). Image segmentation as an estimation problem. *Computer Graphics and Image Processing*, 12(2):153–172. Also in Image Modeling, (A. Rosenfeld, ed.), Academic Press, 1981, pp. 9-28.

[Cheriton and Tarjan, 1976] Cheriton, D. and Tarjan, R. E. (1976). Finding minimum spanning tree. *SIAM Journal on Computing*, 5(4):724–742.

[Chernyak and Stark, 2001] Chernyak, D. and Stark, L. (2001). Top-down guided eye movements. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(4):514–522.

[Chin et al., 1982] Chin, F. Y., Lam, J., and Chen, I.-N. (1982). Efficient parallel algorithms for some graphs problems. *Communications of the Association for Computer Machinery*, 25(9):659–665.

[Cho and Meer, 1997] Cho, K. and Meer, P. (1997). Image segmentation from consensus information. *Computer Vision and Image Understanding*, 68(1):72–89.

[Chowdhury and Murhty, 1997] Chowdhury, N. and Murhty, C. (1997). Minimal spanning tree based clustering technique: Relationship whith bayes classifier. *Pattern Recognition*, 30(11):1919–1929.

[Christofides, 1975] Christofides, N. (1975). *Graph Theory - An Algorithmic Approach.* Academic Press, New York, London, San Francisco.

[Chung, 1997] Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

[Cinque et al., 1994] Cinque, L., Guerra, C., and Levialdi, L. (1994). Reply: On the paper by r. haralick. *CVGIP: Image Understanding*, 60(2):250–252.

[Cole et al., 1994] Cole, R., N., K. P., and E., T. R. (1994). A linear-work parallel algorithm for finding minimum spanning trees. In *Annual Association for Computer Machinery Symposium on Parallel Algorithms and Architectures*, pages 11–15.

[Cole et al., 1996] Cole, R., N., K. P., and E., T. R. (1996). Finding minimum spanning forests in logarithmic time and linear work using random sampling. In *Annual Association for Computer Machinery Symposium on Parallel Algorithms and Architectures*, pages 243–250.

[Cooper, 1998] Cooper, M. (1998). The tractibility of segmentation and scene analysis. In *IJCV*, volume 30:1, pages 27–42.

[Cormen et al., 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, Massachusetts Institute of Technology.

[Dey et al., 1998] Dey, T. K., Edelsbrunner, H., Guha, S., and Nekhayev, D. V. (1998). Topology preserving edge contraction. Technical Report RGI-Tech-98-018, Raindrop Geomagic Inc., Research Triangle Park, North Carolina.

[Diestel, 1997] Diestel, R. (1997). *Graph Theory.* Springer, New York.

[Dixon et al., 1992] Dixon, B., Rauch, M., and Tarjan, R. E. (1992). Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing*, 21(6):1184 – 1192.

[Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. John Wiley & Sons.

[Euler, 1736] Euler, L. (1736). Solutio problematis ad geometriam situs pertinentis (The solution of a problem relating to the geometry of position). *Commentarii academiae scientiarum Petropolitanae*, 8:128–140. reprinted in Opera Omnia: Series 1, Volume 7, pp. 1 - 10, 1766.

[Feldman and Ballard, 1982] Feldman, J. A. and Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science*, (6):205–254.

[Felzenszwalb and Huttenlocher, 1998] Felzenszwalb, P. F. and Huttenlocher, D. P. (1998). Image segmentation using local variation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 98–104. IEEE Computer Society.

[Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.

[Feng et al., 1994] Feng, J., Laumy, M., and Dhome, M. (1994). Inexact matching using neural networks. *Pattern Recognition in practice IV: Multiple paradigm, comparative studies and hybrid systems*, pages 177–184.

[Fiedler, 1975] Fiedler, M. (1975). A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechslovak Mathematical Journal*, 25(100):619–633.

[Fischer and Buhmann, 2002] Fischer, B. and Buhmann, J. M. (2002). Data resampling for path based clustering. In van Gool, L., editor, *Proceedings of German Pattern Recognition Symposium*, volume 2449 of *Lecture Notes in Computer Science*, pages 206–214, Switzerland. Springer.

[Fredman and Dan, 1994] Fredman, M. L. and Dan, W. E. (1994). Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences.*, 48(3):533–551.

[Fredman and Robert, 1987] Fredman, M. L. and Robert, T. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the Association for Computer Machinery*, 34(3):596–615.

[Fuh et al., 2000] Fuh, C.-S., Cho, S, W., and Essig, K. (2000). Hierarchical color image region segmentation for content-based image retrival system. *IEEE Transaction on Image Processing*, 9(1):156–62.

[Gabow et al., 1986] Gabow, H. N., Galil, Z., Spencer, T., and Tarjan, R. E. (1986). Efficient algorithms for finding minimum spanning trees in undirected and directed graps. *Combinatorica*, 6(2):109–122.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability.* Freeman.

[Gavril, 1972] Gavril, F. (1972). Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *Journal of Computing*, 1(2):180–187.

[Gdalyahu et al., 1998] Gdalyahu, Y., Weinshall, D., and Werman, M. (1998). A randomized algorithm for pairwise clustering. In Kearns, M. J., Solla, S. A., and Cohn, D. A., editors, *Advances in Neural Information Processing Systems*, volume 2, pages 424–430. The MIT Press.

[Gdalyahu et al., 2001] Gdalyahu, Y., Weinshall, D., and Werman, M. (2001). Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1053–1074.

[Gelgon and Bouthemy, 1999] Gelgon, M. and Bouthemy, P. (1999). A region-level motion-based graph representation and labeling for tracking a spatial image partition. *Pattern Recognition*, 33(4):725–740.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

[Glantz et al., 1999] Glantz, R., Englert, R., and Kropatsch, W. G. (1999). Representation of image structure by a pair of dual graphs. In Kropatsch, W. G. and Jolion, J.-M., editors, *Proceedings of IAPR Workshop on Graph-based Representations in Pattern Recognition*, pages 155–163. OCG-Schriftenreihe, Österreichische Computer Gesellschaft. Band 126.

[Glantz and Kropatsch, 2000a] Glantz, R. and Kropatsch, W. G. (2000a). Guided relinking of graph pyramids. In Ferri, F. J., Iñesta, J. M., Amin, A., and Pavel, P., editors, *Proceedings of Joint IAPR International Workshops on Syntactic and Structural Pattern Recognition and Statistical Patern Recognition*, volume 1876 of *Lecture Notes in Computer Science*, pages 367–376, Alicante, Spain. Springer Verlag.

[Glantz and Kropatsch, 2000b] Glantz, R. and Kropatsch, W. G. (2000b). Plane embedding of dually contracted graphs. In Borgefors, G., Nyström, I., and Sanniti di Baja, G., editors, *Proceedings of Discrete Geometry for Computer Imagery*, volume 1953 of *Lecture Notes in Computer Science*, pages 348–357, Uppsala, Sweden. Springer, Berlin Heidelberg, New York.

[Glantz et al., 2004] Glantz, R., Pellilo, M., and Kropatsch, W. G. (2004). Matching segmentation hierarchies. *International Journal for Pattern Recognition and Artificial Intelligence*, 18(3):397–424.

[Goldberg et al., 1988] Goldberg, A. V., Plotkin, S. A., and Shannon, G. E. (1988). Parallel symmetry-breaking in sparse graphs. *SIAM Journal of Discrete Mathematics*, 1(4):434–445.

[Goldberg and Spencer, 1989] Goldberg, M. and Spencer, T. (1989). A new parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 18(4):419–427.

[Graaf et al., 1994] Graaf, C. N., Koster, A. S. E., Vincken, K. L., and Viergever, M. A. (1994). Validation of the interleaved pyramid for the segmentation of $3d$ vector images. *Pattern Recognition Letters*, 15(5):469–475.

[Graham et al., 2000] Graham, S. M., Joshi, A., and Pizlo, Z. (2000). The travelling salesman problem: A hierarchical model. *Memory and Cognition*, 28(7):1191–1204.

[Granlund, 1999] Granlund, G. H. (1999). The complexity of vision. *Signal Processing*, 74(1):101–126.

[Guigues et al., 2003] Guigues, L., Herve, L. M., and Cocquerez, J.-P. (2003). The hierarchy of the cocoons of a graph and its application to image segmentation. *Pattern Recognition Letters*, 24(8):1059–1066.

[Guy and Medioni, 1996] Guy, G. and Medioni, G. G. (1996). Inferring global perceptual contours for pairwise clustering. *International Journal of Computer Vision*, 20(1-2):113–133.

[Haralick and Shapiro, 1993] Haralick, R. M. and Shapiro, L. G. (1993). *Computer and Robot Vision.*, volume II. Addison Wesley.

[Harary, 1969] Harary, F. (1969). *Graph Theory*. Addison Wesley.

[Haxhimusa et al., 2003] Haxhimusa, Y., Glantz, R., and Kropatsch, W. G. (2003). Constructing stochastic pyramids by mides - maximal independent directed edge set. In Hancock, E. and Vento, M., editors, *4th IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition*, volume 2726 of *Lecture Notes in Computer Science*, pages 35–46, York, UK. Springer, Berlin Heidelberg, New York.

[Haxhimusa et al., 2002] Haxhimusa, Y., Glantz, R., Saib, M., Langs, G., and Kropatsch, W. G. (2002). Logarithmic tapering graph pyramid. In van Gool, L., editor, *Proceedings of German Pattern Recognition Symposium*, volume 2449 of *Lecture Notes in Computer Science*, pages 117–124, Switzerland. Springer.

[Haxhimusa et al., 2005a] Haxhimusa, Y., Ion, A., Kropatsch, W. G., , and Brun, L. (2005a). Hierarchical image partitioning using combinatorial maps. In *Proceeding of the Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition*, pages 179–186.

[Haxhimusa et al., 2006a] Haxhimusa, Y., Ion, A., and Kropatsch, W. G. (2006a). Evaluating hierarchical graph-based segmentation. In Y. Y Tang, P. Wang, G. L. and Yeung, D. S., editors, *Proceedings of 18th International Conference on Pattern Recognition*, volume 2, pages 195–198, Hong Kong, China. IEEE Society.

[Haxhimusa et al., 2006b] Haxhimusa, Y., Ion, A., and Kropatsch, W. G. (2006b). Irregular pyramid segmentations with stochastic graph decimation strategies. In etal., J. F. M.-T., editor, *Proceedings of 11th Iberoamerical Congress on Pattern Recognition*, volume 4225, pages 277–280, Cancun, Mexico. Springer.

[Haxhimusa et al., 2005b] Haxhimusa, Y., Ion, A., Kropatsch, W. G., and Illetschko, T. (2005b). Evaluating minimum spanning tree based segmentation algorithms. In Gagalowicz, A. and Philips, W., editors, *Proceedings of the 11th International Conference on Computer Analysis of Images and Patterns*, volume 3891 of *Lecture Notes in Computer Science*, pages 579–586, France. Springer.

[Haxhimusa and Kropatsch, 2003] Haxhimusa, Y. and Kropatsch, W. G. (2003). Hierarchy of partitions with dual graph contraction. In Milaelis, B. and Krell, G., editors, *Proceedings of German Pattern Recognition Symposium*, volume 2781 of *Lecture Notes in Computer Science*, pages 338–345, Germany. Springer.

[Haxhimusa and Kropatsch, 2004] Haxhimusa, Y. and Kropatsch, W. G. (2004). Segmentation Graph Hierarchies. In Fred, A., Caelli, T., Duin, R. P., Campilho, A., and de Ridder, D., editors, *Proceedings of Joint International Workshops on Structural, Syntactic, and Statistical Pattern Recognition S+SSPR 2004*, volume 3138 of *Lecture Notes in Computer Science*, pages 343–351, Lisbon, Portugal. Springer, Berlin Heidelberg, New York.

[Heath et al., 1997] Heath, M. D., Sarkar, S., and Sanocki, T. Bowyer, K. W. (1997). A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1338–1359.

[Hird and Willson, 1989] Hird, J. and Willson, D. (1989). A comparison of target detection and segmentation techniques. In Lettington, A., editor, *Optical Systems for Space and Defence*, volume 1191, pages 375–386.

[Hirschberg et al., 1979] Hirschberg, D. S., Chandra, D. H., and Sarwate, D. V. (1979). Computing connected components on parallel computers. *Communications of the Association for Computer Machinery*, 22(8):461–464.

[Hofmann et al., 1998] Hofmann, T., Puzicha, J., and Buhmann, J. M. (1998). Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):803–818.

[Hooker, 1995] Hooker, J. N. (1995). Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:33–42. electronic.

[Horowitz and Pavlidis, 1976] Horowitz, S. and Pavlidis, T. (1976). Picture segmentation by a tree traversal algorithm. *Journal of the Association for Computer and Machinery.*, 2(23):368–388.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. (1988). *Algorithms for Clustering Data.* Prentice Hall, Berlin.

[JáJá, 1992] JáJá, J. (1992). *An Introduction to Parallel Algorithms.* Addison-Wesley, NY.

[Jarník, 1930] Jarník, V. (1930). O jistém problému minimálnim (about a certain minimal problem)). *Práce Moravské Přírodvědecké Společnosti v Brně (Acta Societ. Scienc. Natur. Moravicae)*, 6:57–63.

[Johnson, 2002] Johnson, D. S. (2002). A theoretician's guide to the experimental analysis of algorithms. In Goldwasser, M., Johnson, D. S., and McGeoch, C. C., editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, pages 215–250. American Mathematical Society.

[Jolion, 2001] Jolion, J.-M. (2001). Data driven decimation of graphs. In Jolion, J.-M., Kropatsch, W. G., and Vento, M., editors, *Proceedings of IAPR Workshop on Graph-based Representations in Pattern Recognition*, pages 105–114. CUEN.

[Jolion, 2003] Jolion, J.-M. (2003). Stochastic pyramid revisited. *Pattern Recognition Letters*, 24(8):1035–1042.

[Jolion and Montanvert, 1992] Jolion, J.-M. and Montanvert, A. (1992). The adaptive pyramid, a framework for 2D image analysis. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(3):339–348.

[Jolion and Rosenfeld, 1994] Jolion, J.-M. and Rosenfeld, A. (1994). *A Pyramid Framework for Early Vision.* Kluwer.

[Julesz, 1981] Julesz, B. (1981). Textons, the elements of texture perception and their interactions. *Nature*, 290:91–97.

[Julesz and Bergen, 1983] Julesz, B. and Bergen, J. R. (1983). Textons, the fundamental elements in preattentive vision and perception of textures. *The Bell System Technical Journal*, 62(6):1619–1645.

[Kaiser, 2004] Kaiser, G. (2004). Meta-segmentation of remote sensing images based on combinatorial maps. Master's thesis, University of Natural Resources and Applied Life Sciences., Vienna.

[Kammerer and Glantz, 2001] Kammerer, P. and Glantz, R. (2001). Using graphs for segmenting crosshatched brush strokes. In Jolion, J.-M., Kropatsch, W. G., and Vento, M., editors, *Proceedings of IAPR Workshop on Graph-based Representations in Pattern Recognition*, pages 74–83. CUEN.

[Karger et al., 1995] Karger, D. R., Klein, P. N., and Tarjan, R. E. (1995). A randomized linear-time algorithm to find minimum spanning trees. *Journal of Assoc. Computer and Mathematics*, 42(2):321–328.

[Karp and Wigderson, 1985] Karp, R. and Wigderson, A. (1985). A fast parallel algorithm for the maximal independent set problem. *Journal of Association for Computing Machinery*, 32(4):762–773.

[Kelly, 1970] Kelly, M. D. (1970). Edge detection in pictures by computer using planning. In Meltzer, B. and Michie, D., editors, *Machine Intelligence*, volume 6, pages 397–409, Edinburgh Scotland. Edinburgh University Press.

[Keselman and Dickinson, 2001] Keselman, Y. and Dickinson, S. (2001). Generic model abstraction from examples. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 856–863, Kauai, Hawaii. IEEE Computer Society.

[Keselman and Dickinson, 2005] Keselman, Y. and Dickinson, S. (2005). Generic model abstraction from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1141–1156.

[King, 1997] King, V. (1997). A simpler minimum spanning tree verification algorithm. *Algorithmica*, 18(2):263–270.

[Kittler et al., 1993] Kittler, J., Christmas, W., and Petrou, M. (1993). Probabilistic relaxation for matching problems in machine vision. In *Proceedings 4th International Conference on Computer Vision*, pages 666–674.

[Kovalevsky, 1989] Kovalevsky, V. A. (1989). Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing*, 46:141–161.

[Kovalevsky, 1993] Kovalevsky, V. A. (1993). Digital Geometry Based on the Topology of Abstract Cellular Complexes. In Chassery, J.-M., Francon, J., Montanvert, A., and Réveillès, J.-P., editors, *Géometrie Discrète en Imagery, Fondements et Applications*, pages 259–284, Strasbourg, France.

[Kropatsch, 1986] Kropatsch, W. G. (1986). Curve representations in multiple resolutions. In *Proceedings 8th International Conference on Pattern Recognition*, pages 1283–1285. IEEE Comp.Soc.

[Kropatsch, 1991] Kropatsch, W. G. (1991). Image Pyramids and Curves - An Overview. Technical Report PRIP-TR-2, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/.

[Kropatsch, 1994] Kropatsch, W. G. (1994). Building irregular pyramids by dual graph contraction. Technical Report PRIP-TR-35, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/.

[Kropatsch, 1995a] Kropatsch, W. G. (1995a). Building irregular pyramids by dual graph contraction. *IEE-Proc. Vision, Image and Signal Processing*, 142(6):366–374.

[Kropatsch, 1995b] Kropatsch, W. G. (1995b). Equivalent contraction kernels and the domain of dual irregular pyramids. Technical Report PRIP-TR-42, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/.

[Kropatsch, 1995] Kropatsch, W. G. (1995). Towards higher decimation ratios. In Hlavác, V. and Sára, R., editors, *Proceedings of Computer Analysis of Images and Patterns*, volume 970 of *Lecture Notes in Computer Science*, pages 747–752. Springer.

[Kropatsch, 1999] Kropatsch, W. G. (1999). How useful is structure in motion? In D. Chetverikov and T. Szirányi, editor, *Fundamental Structural Properties in Image and Pattern Analysis*. OCG-Schriftenreihe, Österr. Arbeitsgemeinschaft für Mustererkennung, R. Oldenburg.

[Kropatsch, 2002] Kropatsch, W. G. (2002). Abstraction pyramids on discrete representations. In Braquelaire, A. J.-P., Lachaud, J.-O., and Vialard, A., editors, *Proceedings of Discrete Geometry for Computer Imagery*, pages 1–21, Bordeaux, France, April 3-5, 2002. Springer.

[Kropatsch and BenYacoub, 1996] Kropatsch, W. G. and BenYacoub, S. (1996). A general pyramid segmentation algorithm. In Melter, R., Wu, A. Y., and Latecki, L., editors, *Vision Geometry V*, volume 2826 of *Lecture Notes in Computer Science*, pages 216–224. SPIE.

[Kropatsch and Burge, 1998] Kropatsch, W. G. and Burge, M. (1998). Minimizing the topological structure of line images. In Amin, A., Dori, D., Pudil, P., and Freeman, H., editors, *Proceedings of Joint IAPR International Workshops on Syntactic and Structural Pattern Recognition and Statistical Patern Recognition*, volume 1451 of *Lecture Notes in Computer Science*, pages 149–158, Sydney, NSW, Australia, August 11-13, 1998. Springer.

[Kropatsch and Haxhimusa, 2004] Kropatsch, W. G. and Haxhimusa, Y. (2004). Grouping and Segmentation in a Hierarchy of Graphs. In Bouman, C. A. and Miller, E. L., editors, *Proceedings of 16th Annual IS&T/SPIE Symposium Electronic Imaging, Computational Imaging II*, volume SPIE 5299, pages 193–204, USA. IS&T/SPIE.

[Kropatsch et al., 2006] Kropatsch, W. G., Haxhimusa, Y., and Lienhardt, P. (2006). Hierarchies relating topology and geometry. In Christensen, H. I. and Nagel, H.-H., editors, *Cognitive Vision Systems*, volume 3948 of *Lecture Notes in Computer Science*, pages 199–220. Springer Verlag.

[Kropatsch et al., 2004] Kropatsch, W. G., Haxhimusa, Y., and Pizlo, Z. (2004). Integral trees: Subtree depth and diameter. In Klette, R. and Žunic, J., editors, *International Workshop on Combinatorial Image Analysis 2004*, volume 3322 of *Lecture Notes in Computer Science*, pages 77–87, Berlin Heidelberg. Springer-Verlag.

[Kropatsch et al., 2004] Kropatsch, W. G., Haxhimusa, Y., and Pizlo, Z. (2004). Integral trees: Subtree depth and diameter. Technical Report PRIP-TR-092, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group.

[Kropatsch et al., 2005] Kropatsch, W. G., Haxhimusa, Y., Pizlo, Z., and Langs, G. (2005). Vision pyramids that do not grow too high. *Pattern Recognition Letters*, 26(3):319–337.

[Kropatsch et al., 1999] Kropatsch, W. G., Leonardis, A., and Bischof, H. (1999). Hierarchical, adaptive and robust methods for image understanding. *Surveys on Mathematics for Industry*, 9:1–47.

[Kropatsch and Macho, 1995] Kropatsch, W. G. and Macho, H. (1995). Finding the structure of connected components using dual irregular pyramids. In *Proceedings of Discrete Geometry for Computer Imagery*, pages 147–158. LLAIC1, Université d'Auvergne.

[Kropatsch and Montanvert, 1991a] Kropatsch, W. G. and Montanvert, A. (1991a). Irregular pyramids. Technical Report PRIP-TR-5, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/.

[Kropatsch and Montanvert, 1991b] Kropatsch, W. G. and Montanvert, A. (1991b). Irregular versus regular pyramid structures. In Eckhardt, U., Hbler, A., Nagel, W., and Werner, G., editors, *Geometrical Problems of Image Processing*, pages 11–22. Springer.

[Kruskal, 1956] Kruskal, J. B. J. (1956). On the shortest spanning subtree of a graph and the travelling salesman problem. In *Proc. Am. Math. Soc.*, volume 7, pages 48–50.

[Lallich et al., 2003] Lallich, S., Muhlenbach, F., and Jolion, J.-M. (2003). A test to controll a region growing process within a hiearchical graph. *Pattern Recognition*, 36(10):2201–2211.

[Lance and Williams, 1967] Lance, J. and Williams, W. (1967). A general theory of classificatory sorting strategies: I Hierarchical systems. *Computer Journal*, 9:373–380.

[Langs and Bischof, 2002] Langs, G. and Bischof, H. (2002). Focusing visual attention in mobile robot navigation. In *Proceedings of the Workshop of the Austrian Association of Pattern Recognition*, pages 95–102.

[Langs et al., 2001] Langs, G., Bischof, H., and Kropatsch, W. G. (2001). Irregular image pyramids and robust appearance-based object recognition. Technical Report PRIP-TR-67, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/.

[Langs et al., 2002] Langs, G., Bischof, H., and Kropatsch, W. G. (2002). Hierarchical top-down enhancement of robust pca. In *Proceedings of Joint IAPR International Workshops on Syntactic and Structural Pattern Recognition and Statistical Patern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 234–242. Springer.

[Lienhardt, 1989] Lienhardt, P. (1989). Subdivisions of *n*-dimensional spaces and *n*-dimensional generalized maps. In Mehlhorn, K., editor, *Proceedings of the 5th Annual Symposium on Computational Geometry*, pages 228–236, Saarbrücken. ACM Press.

[Lienhardt, 1991] Lienhardt, P. (1991). Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82.

[Lindeberg, 1990] Lindeberg, T. (1990). Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):234–254.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Luby, 1986] Luby, M. (1986). A simple parallel algorithm for the maximal independent set problem. *SIAM Journal of Computing*, 15(4):1036–1053.

[Malik et al., 2001] Malik, J., Belongie, S., Leung, T., and Shi, J. (2001). Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27.

[Malik et al., 1999] Malik, J., Belongie, S., Shi, J., and Leung, T. (1999). Textons, contours and regions: Cue integration in image segmentation. In *Proceedings of International Conference on Computer Vision*, volume 2, pages 918–925.

[Mallat, 1989] Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693.

[Marchadier et al., 2003] Marchadier, J., Kropatsch, W. G., and Hanbury, A. (2003). Homotopic transformations of combinatorial maps. In Nystrm, I., di Baja, G. S., and Svensson, S., editors, *Proceedings of Discrete Geometry for Computer Imagery*, volume 2886 of *Lecture Notes in Computer Science*, pages 134–143, Naples, Italy. Springer.

[Marfil et al., 2004] Marfil, R., Rodrgíguez, J. A., Bandera, A., and Sandoval, F. (2004). Bounded irregular pyramid: A new structure for color image segmentation. *Pattern Recognition*, 37(3):623–626.

[Martin et al., 2001] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of International Conference on Computer Vision*, volume 2, pages 416–423.

[Matas et al., 2002] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In Rosin, P. L. and Marshall, D., editors, *Proceedings of the British Machine Vision Conference*, volume 1, pages 384–393, London, UK. BMVA.

[Mathieu et al., 1992] Mathieu, C., Magnin, I. E., and Baldy-Porcher, C. (1992). Optimal stochastic pyramid: Segmentation of MRI data. *Proceedings of SPIE-IS&T Medical Imaging VI: Image Processing*, 1652:14–22.

[Matsumoto and Nishimura, 1998] Matsumoto, M. and Nishimura, T. (January, 1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *Transactions on Modeling and Computer Simulation*, 8(1):3–30.

[McGeoch, 1992] McGeoch, C. C. (1992). Analyzing algorithms by simulation: Variance reduction techniques and simulation speedups. *Computing Surveys*, 24(2):195–212.

[Meer, 1989] Meer, P. (1989). Stochastic image pyramids. *Computer Vision, Graphics, and Image Processing*, 45(3):269–294. Also as UM CS TR-1871, June, 1987.

[Meer et al., 1990] Meer, P., Mintz, D., Montanvert, A., and Rosenfeld, A. (1990). Consensus vision. In *AAAI-90 Workshop on Qualitative Vision*, pages 111–115, Boston, Massachusetts, USA.

[Mehlhorn and Näher, 1999] Mehlhorn, K. and Näher, S. (1999). *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, U.K.

[Meyer, 1999] Meyer, F. (1999). Graph based morphological segmentation. In Kropatsch, W. G. and Jolion, J.-M., editors, *Proceedings of IAPR Workshop on Graph-based Representations in Pattern Recognition*, pages 51–60. OCG-Schriftenreihe, Band 126, Österreichische Computer Gesellschaft.

[Montanvert et al., 1991] Montanvert, A., Meer, P., and Rosenfeld, A. (1991). Hierarchical image analysis using irregular tesselations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):307–316.

[Munkres, 1993] Munkres, J. R. (1993). *Elements of Algebraic Topology*. Addison-Wesley.

[Nacken, 1994] Nacken, P. F. (1994). *Image Analysis Methods Based on Hierarchies of Graphs and Multi-scale Mathematical Mophology.* PhD thesis, University of Amsterdam, Amsterdam.

[Nacken, 1995] Nacken, P. F. (1995). Image segmentation by connectivity preserving relinking in hierarchical graph structures. *Pattern Recognition*, 28(6):907–920.

[Neštřil, 1997] Neštřil, J. (1997). A few remarks on the history of mst-problem. *Archivum Mathematicum Brno*, 33:15–22.

[Neštřil et al., 2001] Neštřil, J., Miklovà, E., and Neštřilova, H. (2001). Otakar Borůvka on minimal spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233:3–36.

[Oxley, 1992] Oxley, J. (1992). *Matroid theory*. Oxford University Press, New York, USA.

[Pailloncy et al., 1998] Pailloncy, J.-G., Kropatsch, W. G., and Jolion, J.-M. (1998). Object matching on irregular pyramid. In Jain, A. K., Venkatesh, S., and Lovell, B. C., editors, *14th International Conference on Pattern Recognition*, volume II, pages 1721–1723. IEEE Comp.Soc.

[Päivinen, 2005] Päivinen, N. (2005). Clustering with a minimum spanning tree of a scale-free-like structure. *Pattern Recognition Letters*, 26(7):921–930.

[Pal and Pal, 1993] Pal, N. R. and Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition*, 26(3):1277–1294.

[Pavan and Pelillo, 2003] Pavan, M. and Pelillo, M. (2003). Dominant sets and hierarchical clustering. In *Proceedings of International Conference on Computer Vision*, volume 1, pages 362–369. IEEE Computer Society.

[Pavan and Pelillo, 2003] Pavan, M. and Pelillo, M. (2003). Graph-theoretic approach to clustring and segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 145–152. IEEE Computer Society.

[Pavlidis, 1977] Pavlidis, T. (1977). *Structural Pattern Recognition.* Springer Verlag.

[Pelillo et al., 1999] Pelillo, M., Siddiqi, K., and Zucker, S. W. (1999). Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120.

[Perona and Freeman, 1998] Perona, P. and Freeman, W. (1998). A factorization approach to grouping. In Burkhardt, H. and Neumann, B., editors, *Proceeding of European Conference in Computer Vision*, volume 1406 of *Lecture Notes in Computer Science*, pages 655–670. Springer Verlag.

[Pizlo, 2001] Pizlo, Z. (2001). Perception viewed as an inverse problem. *Vision Research*, 41(24):3145–3161.

[Pizlo and Li, 2003] Pizlo, Z. and Li, Z. (2003). Pyramid algorithms as models of human cognition. In *Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging*, pages 1–12. SPIE.

[Pizlo and Li, 2004] Pizlo, Z. and Li, Z. (2004). Graph pyramids as models of human problem solving. In *Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging*, pages 205–215. SPIE.

[Pizlo et al., 1997] Pizlo, Z., Salach-Golyska, M., and Rosenfeld, A. (1997). Curve detection in a noisy image. *Vision Research*, 37(9):1217–1241.

[Pizlo et al., 2006] Pizlo, Z., Stefanov, E., Saalweachter, J., Li, Z., Haxhimusa, Y., and Kropatsch, W. G. (2006). Traveling salesman problem: a foveating model. *in press*.

[Prim, 1957] Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36:1389–1401.

[Privitera and Stark, 2000] Privitera, C. M. and Stark, L. W. (2000). Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):970–982.

[Rojer and Schwartz, 1990] Rojer, A. and Schwartz, E. L. (1990). Design considerations for a space variant visual sensor complex-logarithmic geometry. In *Proceeding of Internation Conference in Pattern Recognition*, volume 2, pages 278–285, Los Alamitos, California. IEEE Computer Society Press.

[Rosenfeld, 1982] Rosenfeld, A. (1982). Quadtrees and pyramids: Hierarchical representation of images. Technical Report TR-1171, University of Maryland, Computer Science Center.

[Rosenfeld, 1984] Rosenfeld, A., editor (1984). *Multiresolution Image Processing and Analysis.* Springer Verlag.

[Rosenfeld, 1985] Rosenfeld, A. (1985). Arc colorings, partial path groups, and parallel graph contractions. Technical Report TR-1524, University of Maryland, Computer Science Center.

[Rosenfeld, 1986] Rosenfeld, A. (1986). Pyramid algorithms for perceptual organization. *Behavior Research Methods, Instruments, and Computers*, 18:595–600.

[Rosenfeld, 1987] Rosenfeld, A. (1987). Pyramid algorithm for efficient vision. Technical Report CAR-TR-299, University of Maryland, Computer Science Center.

[Rosenfeld, 1989] Rosenfeld, A. (1989). Computer vision: A source of models for biological visual processes?. *IEEE Transaction on Biomedical Engineering*, 36(1):93–96.

[Rosenfeld et al., 1976] Rosenfeld, A., Hummel, R., and Zucker, S. (1976). Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):420 – 433.

[Rosenfeld and Thurston, 1971] Rosenfeld, A. and Thurston, M. (1971). Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, 20(12):562–569.

[Roy et al., 2004] Roy, D., Ghitza, Y., Bartelma, J., and Kehoe, C. (2004). Visual memory augmentation: Using eye gaze as an attention filter. In *Proceedings of the 8th IEEE International Symposium on Wearable Computers*, pages 128–131.

[Saib et al., 2002] Saib, M., Haxhimusa, Y., and Glantz, R. (2002). *Dgc_tool*: Building irregular graph pyramid using dual graph contraction. Technical Report PRIP-TR-69, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/.

[Samet, 1990] Samet, H. (1990). *Applications of Spatial Data Structures*. Addison Wesley Publishers.

[Sanfeliu et al., 2002] Sanfeliu, A., Alquèrez, R., Andrade, J., Climent, J., Serratosa, F., and Vergès, J. (2002). Graph-based representation and techniques for image processing and image analysis. *Pattern Recognition*, 35(3):639–650.

[Sanfeliu and Fu, 1983] Sanfeliu, A. and Fu, K. (1983). Adistance measure between attributed relational graphs for pattern recognition. *IEEE Transaction on Sytems Man and Cybernetics*, 13:353–362.

[Serra, 1982] Serra, J. (1982). *Image Analysis and Mathematical Morphology*, volume I. Academic Press.

[Sharon et al., 2000] Sharon, E., Brandt, A., and Basri, R. (2000). Fast multiscale image segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 70–77.

[Shen et al., 1998] Shen, X., Spann, M., and Nacken, P. F. M. (1998). Segmentation of 2d and 3d images through a hierarchical clustering based on region modelling. *Pattern Recognition*, 31(9):1295–1309.

[Shi and Malik, 1997] Shi, J. and Malik, J. (1997). Normalized cuts and image segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–737. IEEE Computer Society.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

[Shiloach and Vishkin, 1982] Shiloach, Y. and Vishkin, U. (1982). An $o(\log(n))$ parallel connectivity algorithm. *Journal of Algorithms*, 3(1):57–67.

[Skiena, 1990] Skiena, S. (1990). *Implementing Discrete Mathematics.*, chapter 5.6.3, pages 218–219. Addison-Wesley.

[Soille, 1994] Soille, P. (1994). *Morphological Image Analysis*. Springer Verlag.

[Sonka et al., 1999] Sonka, M., Hlavac, V., and Boyle, R. (1999). *Image Processing, Analysis and Machine Vision*. Brooks/Cole Publishing Company.

[Stark and Privitera, 1997] Stark, L. W. and Privitera, C. M. (1997). Top-down and bottom-up image processing. In *IEEE Proceeding of International Conference on Neural Networks*, volume 4, pages 2294 –2299.

[Stelldinger and Ullrich, 2005] Stelldinger, P. and Ullrich, K. (2005). Towards a general sampling theory for shape preservation. *Image and Vision Computing*, 23(10):237–248.

[Sudhir and Sarkar, 1997] Sudhir, B. and Sarkar, S. (1997). A framework for performance characterization of intermediate-level grouping modules. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1306–1312.

[Tanimoto and Pavlidis, 1975] Tanimoto, S. and Pavlidis, T. (1975). A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, (4):104–119.

[Tarjan, 1975] Tarjan, R. E. (1975). Efficiency of a good but not linear set-union algorithm. *Journal of the Association for Computer Machinery*, 22(2):215–225.

[Thulasiraman and Swamy, 1992] Thulasiraman, K. and Swamy, M. N. S. (1992). *Graphs: Theory and Algorithms.* Wiley-Interscience.

[Tremeau and Colantoni, 2000] Tremeau, A. and Colantoni, P. (2000). Regions adjacency graph applied to color image segmentation. *IEEE Transaction on Image Processing*, 9(4):735–744.

[Tsotsos, 1988a] Tsotsos, J. K. (1988a). A 'complexity level' analysis of immediate vision. *International Journal of Computer Vision*, 2(1):303–320.

[Tsotsos, 1988b] Tsotsos, J. K. (1988b). How does human vision beat the computational complexity of visual perception? In Pylyshyn, Z., editor, *Computational Processes in Human Vision: An Interdisciplinary Perspective*, pages 286–338. Ablex Press, Notwood, NJ.

[Tsotsos, 1990] Tsotsos, J. K. (1990). Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, 13(3):423–469.

[Tsotsos, 1992] Tsotsos, J. K. (1992). On the relative complexity of passive vs active visual search. *International Journal of Computer Vision*, 7(2):127–141.

[Ullmann, 1976] Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42.

[Umeyama, 1998] Umeyama, S. (1998). An eigendecomposition approach to weghted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:695–703.

[Urquhart, 1982] Urquhart, R. (1982). Graph theoretical clustering based on limited neighborhood sets. *Pattern Recognition*, 15(3):173–187.

[Vergés-Llahi et al., 2000] Vergés-Llahi, J., Climent, J., and Sanfeliu, A. (2000). Color segmentation solving hard-constraint on graph partitioning greedy algorithms. In Sanfeliu, A., Villanueva, J., Vanrell, M., Alquezar, R., Jain, A., and Kittler, J., editors, *Proceeding of Internation Conference in Pattern Recognition*, volume 3, pages 625–628. IEEE Computer Society.

[Vlachos and Constantinides, 1993] Vlachos, T. and Constantinides, A. (1993). Graph-theoretical approach to color picture segmentation and contour classification. *IEE Proceedings I: Communications, Speech and Vision*, 140(1):36–45.

[Wallace et al., 1994] Wallace, R. S., Ong, P.-W., Bederson, B. B., and Schwatz, E. L. (1994). Space variant image processing. *International Journal of Computer Vision*, 13(1):71–90.

[Wang et al., 1997] Wang, Y., Fan, K., and Horng, J. (1997). Genetic-based search for error-correcting graph isomorphism. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:588–597.

[Webster, 1913] Webster, M. (1913). *Webster's Revised Unabridged Dictionary.* Merriam Webster Inc.

[Weiss, 1999] Weiss, Y. (1999). Segmentation using eigenvectors: A unifying view. In *Proceedings of International Conference on Computer Vision*, volume 2, pages 975–982.

[Wertheimer, 1925] Wertheimer, M. (1925). über Gestaltheorie. *Philosophische Zeitschrift für Forschung und Aussprache*, 1:30–60.

[Willersinn, 1995] Willersinn, D. (1995). *Dual Irregular Pyramid.* PhD thesis, Vienna University of Technology, Faculty of Informatics, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, Vienna.

[Williams et al., 1999] Williams, M. L., Wilson, R. C., and Hancock, E. R. (1999). Deterministic search for relational graph matching. *Pattern Recognition*, 32:1255–1271.

[Willis, 1997] Willis, D. E. (1938, reprinted by Gestalt Journal Press, New York 1997.). *Source Book of Gestalt Psychology*. Harcourt, Brace and Co. New York.

[Witkin, 1986] Witkin, A. P. (1986). Scale space filtering. In Pentland, A. P., editor, *From Pixels to Predicates: Recent Advances in Computational and Robot Vision*, pages 5–19. Ablex, Notwood, NJ.

[Wu and Leahy, 1993] Wu, Z. and Leahy, R. M. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113.

[Xu and Oja, 1990] Xu, L. and Oja, E. (1990). Improved simmulated annealing, boltzman machines and attributed graph matching. In Almeida, L., editor, *Lecture Notes in Computer Science*, volume 412, pages 151–161. Springer Velag.

[Xu et al., 1993] Xu, S., Kamath, M. V., and Capson, D. W. (1993). Selection of partitions from a hierarchy. *Pattern Recognition Letters*, 14(1):7–15.

[Xu and Uberbacher, 1997] Xu, Y. and Uberbacher, E. C. (1997). 2d image segmentation using minimum spanning tree. *Image and Vision Computing*, 15(1):47–57.

[Yao, 1975] Yao, A. C.-C. (1975). An $o(|e| \log\log |v|)$ algorithm for finding minimum spanning tree. *Information Processing Letters*, 4(1):21–23.

[Yu and Shi, 2001] Yu, S. and Shi, J. (2001). Segmentation with pairwise attraction and repulsion. In *Proceedings of International Conference on Computer Vision*, volume 1, pages 52–58. IEEE Computer Society.

[Yuille and Poggio, 1986] Yuille, A. L. and Poggio, T. A. (1986). Scaling theorems for zero crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):15–25.

[Zahn, 1971] Zahn, C. (1971). Graph-theoretical methods for detecting and describing gestal clusters. In *IEEE Transaction on Computing*, volume 20, pages 68–86.

[Zeki, 1993] Zeki, S. (1993). *A Vision of the Brain*. Oxford: Blackwell.

[Zhang, 1996] Zhang, Y. (1996). A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346.

[Zhou et al., 1989] Zhou, Y., Venkateswar, V., and Chellappa, R. (1989). Edge detection and linear feature extraction using a 2-d random field model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):84–95.

# List of Symbols and Abbreviations

If not stated otherwise in the text the symbol and abbreviations applies throughout the document.

**Symbols**

| | |
|---|---|
| $N_{k,k+1}$ | contraction kernel from level $k$ to $k+1$ |
| $\overline{G}$ | dual graph |
| $\overline{E}$ | dual edge set |
| $\overline{e}$ | dual edge |
| $attr_e$ | edge attribute |
| $E$ | edge set |
| $e$ | edge |
| $Ext(\cdot)$ | external contrast |
| $F$ | face set |
| $f$ | face |
| $(G_k, \overline{G}_k)$ | pair of dual graphs at level $k$ |
| $CC$ | connected component |
| $G = (V, E)$ | graph |
| $G_k$ | graph at level $k$ |
| $h$ | height of pyramid |
| $Int(\cdot)$ | internal contrast |
| $k$ | level $k$ of the pyramid |
| $\mathbb{R}$, $\mathbb{Z}$ | number sets |
| $\mathcal{N}(\cdot)$ | neighborhood of a vertex or edge |
| $\widetilde{G}$ | plane graph |
| $\lambda$ | reduction factor |
| $T$ | tree |
| $\overline{V}$ | dual vertex set |
| $\overline{v}$ | dual vertex |
| $attr_v$ | vertex attribute |
| $V$ | vertex set |
| $v$ | vertex |
| $\lvert \cdot \rvert$ | set cardinality |

**Abbreviations**

BorůSeg Segmenation based on Borůvka's MST
BorůSeg (D3P) Segmentation based on Borůka's MST using D3P
BorůSeg (MIES) Segmentation based on Borůka's MST using MIES
BorůSeg (MIS) Segmentation based on Borůka's MST using MIS
D3P Data driven decimation process
DGC Dual Graph Contraction
ECK Equivalent contraction kernel
GCE Global consistency error
KrusSeg Segmenation based on Kruskal's MST
LCE Local consistency error
MIDES Maximal (independent) set of directed edges
MIES Maximal independent edge set
MIS Maximal independent vertex set
MST Minimum spanning tree
NCutSeg Normalized cuts segmentation

# Index

The first page number is usually, but not always, the primary reference to the indexed topic.

**v**