Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/183-2
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: {yll,krw}@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-81 July 16, 2003

Hierarchical Image Partitioning with Dual Graph Contraction [1]

Yll Haxhimusa and Walter Kropatsch

Abstract

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.

[1]This report has been supported by the Austrian Science Fund under grants P14445-MAT and P14662-INF

Contents

1 Introduction

Wertheimer [54] 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" [1], and introduced the importance of perceptual grouping and organization in visual perception.

Low-level cue image segmentation cannot and should not produce a complete final "good" segmentation. The low-level coherence of brightness, color, texture or motion attributes should be used to come up sequentially with hierarchical partitions [50]. 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 [11]:

- capture perceptually important groupings or regions, which reflect global aspects of the image,
- be highly efficient, running in time linear in the number of image pixels,
- creates hierarchical partitions [50].

In a regular image pyramid (for an overview see [32]) the number of pixels at any level l, is r times higher than the number of pixels at the next reduced level $l+1$. The so called reduction factor r is greater than one and it is the same for all levels l. If s denotes the number of pixels in an image I, the number of new levels on top of I amounts to $log_r(s)$. 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, regular image pyramids are confined to globally defined sampling grids and lack shift invariance [2]. Bister et.al. [2] concludes that regular image pyramids have to be rejected as general-purpose segmentation algorithms. In [41, 23] 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. Moreover in [37, 39, 3, 7, 18] was shown that irregular pyramid can be used for segmentation and feature detection.

[1] "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" [55]

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 [38, 22, 1, 20]:

- the cells have no information about their global position.
- the cells are connected only to (direct) neighbors.
- 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. 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 (8)-connectivity of the pixels. A cell on level $l+1$ (parent) is a union of neighboring cells on level l (children). This union is controlled by so called contraction kernels (decimation parameters, see Section 3). Every parent computes its values independently of other cells on the same level. This implies that an image pyramid is built in $O[log(image\_diameter)]$ time. For more in depth on the subject see the book of Jolion [24] and of Rosenfeld [47]. Neighborhoods on level $l+1$, are derived from neighborhoods on level l. 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, Figure 1(a). We assume that on each level $l+1$ ($l \geq 0$) there exists at least one cell not contained in level l. In particular, there exists a highest level h . In general the top of the pyramid can have one vertex, i.e. an apex.

In this technical report we represent the levels as dual pairs $(G_l, \overline{G_l})$ of plane graphs G_l and its dual (plane) graph $\overline{G_l}$, Figure 1(b). That is why we use the 4-connectivity. The vertices of G_l represent the cells and the edges of G_l represent the neighborhood relations of the cells on level l, depicted with square vertices and dashed edges in Figure 1(b). The edges of $\overline{G_l}$ represent the borders of the cells on level l, depicted with solid lines in Figure 1(b), possibly including so called pseudo edges needed to represent the neighborhood relation to a cell completely surrounded by another cell. Finally, the vertices of $\overline{G_l}$, the circles in Figure 1(b), represent meeting points of at least three edges from G_l, solid lines in Figure 1(b). 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 $attr_v : V \to \mathbb{R}^+$ and $attr_e : E \to \mathbb{R}^+$. We use a weight for $attr_e$ measuring the difference between the two end points.

The aim of this technical report is to built in 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 for contraction kernels proposed in [20] to achieve logarithmic tapering, local construction and shift invariance. Borůvka's algorithm [4] with dual graph contraction algorithm [30] is used for building in a hierarchical way a minimum weight spanning tree (of the region) and at the same time to preserve topology. 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

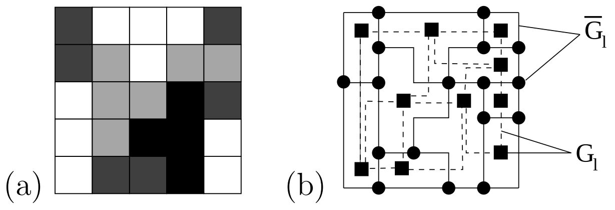


Figure 1: a) Partition of pixel set into cells. b) Representation of the cells and their neighborhood relations by a dual pair $(G, \overline{G_l})$ of plane graphs.

in a bottom-up 'stimulus-driven' based on local differences in a specific feature like in an preattentive vision. See the book of Jolion [24] for an extensive overview of the pyramid framework for early vision.

The plan of the technical report is as follows. In order to make the reading of this technical report easy we recall some of the basic notion as follows. In Section 2 we recall the main idea of the minimum weight spanning tree and in Subsection 2.1 we recall Borůvka's MST algorithm. The dual graph contraction algorithm is explained in Section 3 and then Borůvka's algorithm is re-defined in Section 4, so that we can construct an image graph pyramid, and, at the same time the minimum spanning tree. In Section 5 we give the definition of internal and external contrast and the merging decision criteria based on these definitions. The algorithm for building the hierarchy of partitions is introduced in this section. Section 6 reports on experimental results.

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 [11, 13, 18], a minimum cut [56, 50, 16] and, among these methods a classic approach to clustering (the *complete linkage clustering* algorithm [34]) reduces to a search for a complete subgraph i.e. the maximal clique [42].

Hierarchical structures for description of the data for clustering purposes have been studied very early in [34], or for image segmentation in [21]. Image pyramids are a powerful tool for early vision, see the books of Jolion [24] and of Rosenfeld [47] for an extensive overview of the topic. However, the regular image pyramids are confined to globally defined sampling grids and lack shift invariance. Bister et.al. [2] concludes that regular image pyramids have to be rejected as general-purpose segmentation algorithms. In [41, 23] 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 [39] in his "consensus vison" used the concept of irregular pyramid to produce an image segmentation. Moreover in [37, 7, 3, 7, 18] was shown that irregular pyramid can be used for segmentation and feature detection.

The clustering community [27] has produced agglomerative and divisive algorithms; in image segmentation the region-based merging and spliting algorithms exist. Early graph-based methods [59] 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 [52] 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 [11, 13, 18] uses an adaptive criterion that depend on local properties rather than global ones. Recently the works [11, 40, 13, 18] have the minimum spanning tree as the base algorithm.

Gestalt grouping factors, such as proximity, similarly, continuity and symmetry, are encoded and combined in pairwise feature similarity measures [56, 50, 46, 44, 57, 53, 49, 58, 13]. Other method of segmentation is that of splitting and merging region based on how well the regions fulfill some criterion. Such methods [9, 43] uses a measure of uniformity of a region. Authors in [11, 13, 18] 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 [19, 35].

The use of Markov Random Field (MRF) has been used for image restoration and segmentation [17]. However the use of MRF to image segmentation usually leads to NP-hard problems. The graph-based approximation method for MRF problems [5] 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 [56, 50, 16]. Wu and Leahy in [56] define a cut criterion, but it was biased toward finding small components. Shi and Malik [50] 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 [11]. Shi and Malik [50] 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 [12]) 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 [27]. Pavan and Pelillo [42] 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 [48].

Our method is related to the work in [11, 18] in the sense of pairwise comparison of region similarity. 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 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.

2 Minimum Weight Spanning Tree

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 $v_i, v_j \in V$. Let each edge $e \in E$ be associated with **nonnegative unique real** weights $w(e) = w(v_i, v_j)$. Note that parallel edges, for ex. $e_1 = (v_1, v_2)$ and $e_2 = (v_1, v_2)$ $e_1 \neq e_2$, have different weights. First let us give some basic graph theoretical notions taken from [51].

Definition 1 [**walk**]. *A walk in a graph G is a finite alternating sequence of vertices and edges $v_0, e_1, v_1, ..., v_{k-1}, e_k, v_k$ beginning and ending with vertices such that v_{i-1} and v_i are the end vertices of the edge e_i, $1 \leq i \leq k$.*

Definition 2 [**trail**]. *A walk is a trail if all its edges are distinct. A trail is closed if its end vertices are the same.*

Definition 3 [**circuit**]. *A closed trail is a circuit if all its vertices except the end vertices are distinct.*

Definition 4 [**acyclic graph**]. *A graph G is acyclic if it has no circuits.*

Definition 5 [**tree**]. *A tree of graph G is a connected acyclic subgraph of G.*

Definition 6 [**spanning tree**]. *Spanning tree of graph G is a tree of G containing all the vertices of G.*

The problem is formulated as construction of a minimum weight spanning tree of G. A deterministic solution to this problem was proposed by Borůvka [4], Kruskal [33], and Prim [45] as the widely known greedy algorithms in the literature. A better solution to this problem was proposed (depending on the assumptions made) by Karger et.al. in [28] with a randomized algorithm, which solves the problem in linear expected time; Fredman et.al. in [14] with the linear worst case time solution under the assumption that weights are small integers. The solution proposed by Gabow et.al. in [15] solves the problem with the solution's exact bound $O(m \log \beta(m,n))$, on a graph with n vertices and m edges. Here $\beta(m,n) = min\{i| \log^{(i)}(n) \leq m/n\}$, where $\log^{(i)} n$ is $\underbrace{\log(\log(..(\log(n))..))}_{i-times}$.

Here we will give two lemmas that provide the basis of the minimum weight spanning tree algorithms, taken from [51], which will help us in proving the correctness of MST algorithm. 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 :

$$w(T) = \sum_{e \in T} w(e). \tag{1}$$

Theorem 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. $\Box$

Theorem 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.*

Proof: Let T_{min} be a minimum weight spanning tree of G containing T. If $T_{min} = T \cup T'$ (Figure 2(a)), then clearly T' is a spanning tree of G' (Figure 2(b)). Therefore,

$$w(T') \geq w(T'_{min}). \tag{2}$$

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,

$$\begin{aligned} w(T'_{min} \cup T) &\geq w(T_{min}) \\ &= w(T) + w(T'). \end{aligned} \tag{3}$$

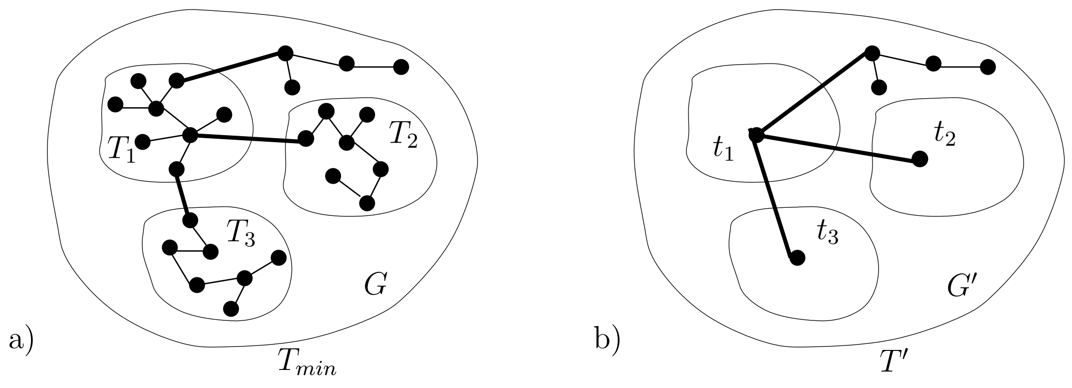


Figure 2: (a) $T_{min} = T \cup T'$, where $T = T_1 \cup T_2 \cup T_3$ and (b) T' after contraction of all edges in T.

From this we get

$$w({T'}_{min}) \geq w(T'). \tag{4}$$

Combining Equations (2) and (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. $\square$

We intend to solve the problem of MST in parallel by using Borůvka's algorithm in conjunction with dual graph contraction [30]. We use Borůvka's algorithm since it can be used to built MST in parallel [8]. In the next Section 2.1 we give the parallel version of this algorithm.

2.1 Borůvka's Algorithm

The idea of the Borůvka [4] is to do steps like in Prim's algorithm, in parallel over the graph at the same time. This algorithm constructs a spanning tree in iterations composed of the following steps:

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 3; (a) the simple attributed graph as input, where each vertex is a tree in L; (b) the edges with the smallest weight (solid lines) incidented 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 3(c) is the MST of the input graph.

Algorithm 1 – **Borůvka's Algorithm**

```
Input: Graph $G(V, E)$.
1: $MST$ := 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: end while
Output: Minimum weight spanning tree - $MST$.
```

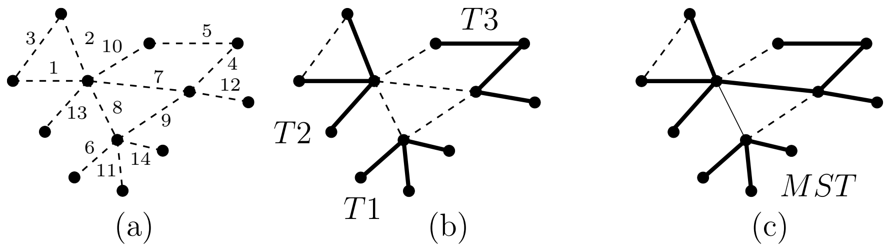


Figure 3: The steps of building MST with Borůvka's algorithm. a) Starting graph; b) Edges with minimum weight (solid lines); c) MST (solid line).

The proof that this algorithm builds the minimum weight spanning tree is based on the proof of the Kruskal's MST algorithms given in [51].

Theorem 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 1 the edge e_{i+1} is contained in a minimum weight spanning tree T'_{min} of G'. By Theorem 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 1 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.

Observation 1 *In 3^{rd} step of the Algorithm 1, each tree $T \in L$ finds the edge with the minimal weight, and as trees becomes larger, the process of finding the edge with the minimal weight for each tree T takes longer.*

3 Dual Graph Contraction (DGC)

Taking the Observation 1 into consideration, a contraction of the edge e, which connects T and $G \setminus T$ in the 4^{th} step of Algorithm 1 will speed up the process of searching for minimum weight edges in the Borůvka's algorithm. 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. Note that introduction of the edge contraction method introduces the hierarchy of graphs. Kropatsch in [30] and Dey et.al. in [10] introduce a topology preserving edge contraction.

First, shortly the dual graph contraction algorithm is given, as in Kropatsch [30], and then re-defined Borůvka's algorithm is described in Section 4 so that we can construct minimum spanning tree i.e. a graph pyramid.

The irregular (or graph) pyramids are constructed bottom-up level by level by repeatedly contracting the image graph in the base. Dual graph contraction [29, 30] proceeds in two basic steps (Figure 4):

- dual edge contraction, and
- dual face contraction.

The base of the pyramid consist of the pair of dual image graphs $(G_0, \overline{G_0})$. The following decimation parameter $\langle S_k, N_{k,k+1} \rangle$ determine the structure of an irregular pyramid [30]: a subset of surviving vertices $S_k = V_{k+1} \subset V_k$, and a subset of primary non-surviving edges $N_{k,k+1} \subset E_k$. (Secondary non-surviving edges are removed during dual face contraction). Every non-surviving vertex $v \in V_k \setminus S_k$ must be connected to one surviving vertex in a unique way. 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 decimation parameters $\langle S_k, N_{k,k+1} \rangle$, as expressed by function $C[.,.]$:

$$(G_{k+1}, \overline{G_{k+1}}) = C[(G_k, \overline{G_k}), \langle S_k, N_{k,k+1} \rangle]. \tag{5}$$

$$\begin{tikzcd}
\text{neighborhood graph} & & \text{face graph} \\
\boxed{G_k(V_k, E_k)} \arrow[rr, leftrightarrow, "\text{dual}"] \arrow[d, Rightarrow, "\text{edge contracted}"'] & \textbf{dual edge contraction} \arrow[r] & \boxed{\overline{G_k}(\overline{V_k}, \overline{E_k})} \arrow[d] \\
\boxed{G^*(S_k, E_k \setminus N_{k,k+1})} \arrow[rr, leftrightarrow, "\text{dual}"] \arrow[d, "\text{dually contracted}"'] & \textbf{dual face contraction} \arrow[l] & \boxed{\overline{G^*}(\overline{V_k}, \overline{E_k} \setminus \overline{N_{k,k+1}})} \arrow[d, Rightarrow, "\text{eliminates } \deg(\overline{v}) < 3"] \\
\boxed{G_{k+1}(V_{k+1}, E_{k+1})} \arrow[rr, leftrightarrow, "\text{dual}"] & & \boxed{\overline{G_{k+1}}(\overline{V_{k+1}}, \overline{E_{k+1}})}
\end{tikzcd}$$

Figure 4: Dual Graph Contraction: $(G_{k+1}, \overline{G_{k+1}}) = C[(G_k, \overline{G_k}), (S_k, N_{k,k+1})]$

The contraction of a primary non-surviving edge consists in the identification of its endpoints (vertices) and the removal of both the contracted edge and its dual edge. Figure 5(a) shows the normal situation, Figure 5(b) the situation where the dual edge contraction creates multiple edges, Figure 5(c) and self-loops. Redundancies (lower part) in case Figure 5(c) are decided through the corresponding dual graphs and removed by dual graph contraction.

Dual face contraction simplifies most of the multiple edges and self-loops, but not those enclosing any surviving parts of the graph. They are necessary to preserve correct structure [30]. See Kropatsch [29, 30, 31] for more details on graph contraction.

3.1 Contraction Kernels

The connected components [2] $CC(s), s \in S$, of subgraph (S, N) form small tree structures $T(s)$ that collapse into vertex s of the contracted graph: $T(s) := (CC(s), N \cap (CC(s) \times CC(s)))$. Their union contain the primary non-surviving edges N. $T(s)$ is a **spanning tree** of the connected component $CC(s)$, or equivalently, (V, N) is a spanning forest of graph $G(V, E)$.

Definition 7 *A decimation of a graph $G(V, E)$ is specified by a selection of* **surviving** *vertices $S \subset V$ and a selection of* **primary non-surviving edges** *$N \subset E$ such that following two conditions are fulfilled:*

1. *Graph (V, N) is a spanning forest of graph $G(V, E)$.*

2. *The surviving vertices $S \subset V$ are the roots of the forest (V, N).*

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

[2]Neglected level indices refer to contraction from level k to level $k + 1$.

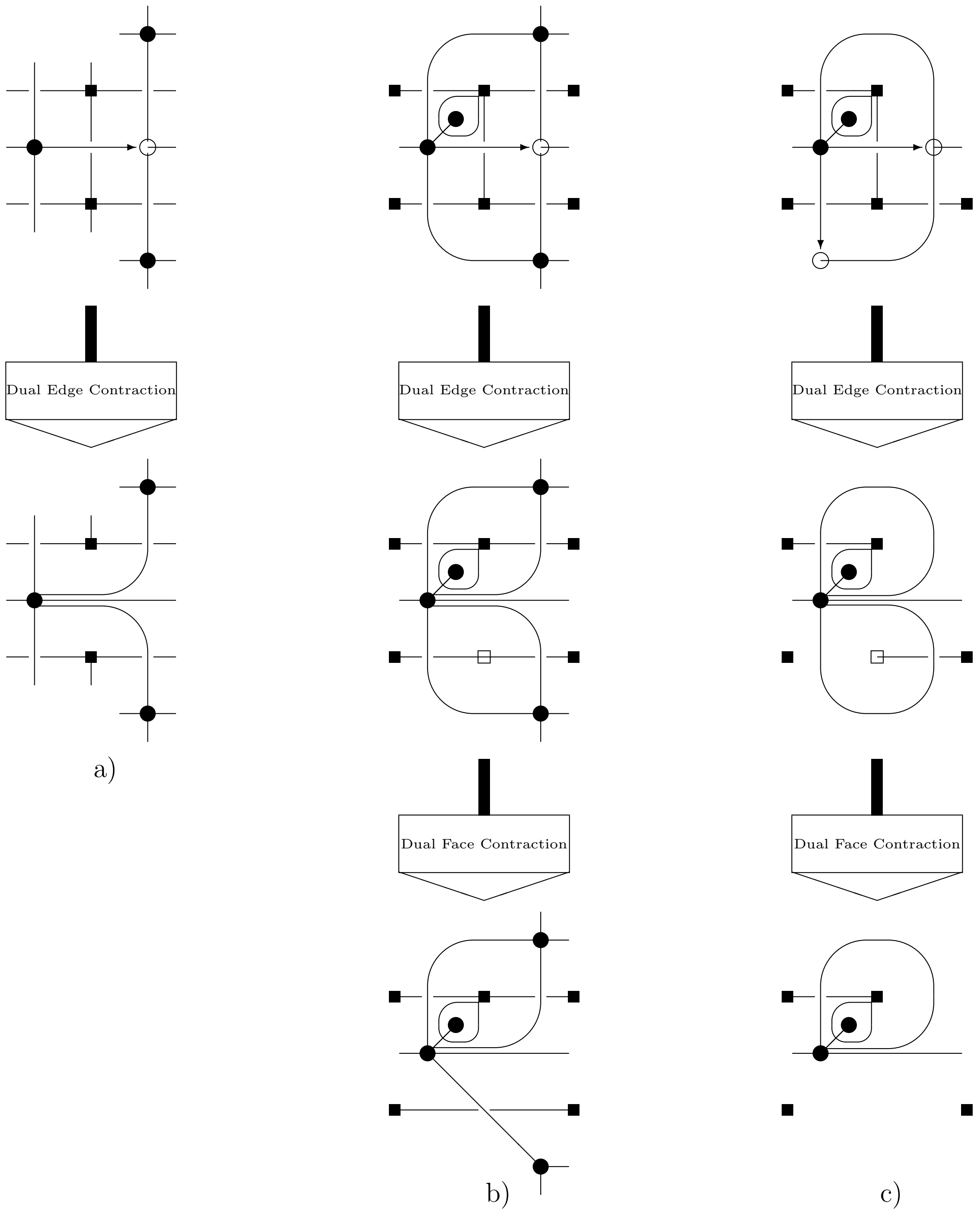


Figure 5: Dual graph contraction: a) normal case, b) multiple edges, c) self-loops. ❍, S_k; ■, $\overline{V_{k+1}}$; ❍, $V_k\backslash S_k$; □, $\overline{V_k}\backslash\overline{V_{k+1}}$; $\longrightarrow$, $\langle S_k, V_k\backslash S_k\rangle \in N_{k,l}$.

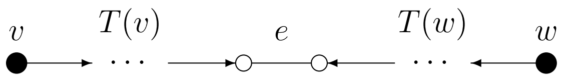


Figure 6: Decomposition of connecting path $CP(v, w)$

The connectivity structure of the contracted graph is established by paths connecting two surviving vertices.

Definition 8 *Let $G(V,E)$ be a graph with decimation parameters (S, N). A path in $G(V, E)$ is called a* **connecting path** *between two surviving vertices $v, w \in S$, denoted $CP(v, w)$, if it consists of three subsets of edges E (Figure 6):*

- *the first part is a possibly empty branch of contraction kernel $T(v)$.*
- *the middle part is an edge $e \in E \backslash N$ that bridges the gap between the two contraction kernels $T(v)$ and $T(w)$. We call e the* **bridge** *of the connecting path $CP(v, w)$.*
- *the third part is a possibly empty branch of contraction kernel $T(w)$.*

Connecting paths $CP(v, w)$ in $G(V, E)$ are strongly related to the edges $e_{k+1} = (v, w) \in E_{k+1}$ in the contracted graph $G_{k+1}(V_{k+1}, E_{k+1})$: Two different surviving vertices that are connected by a connecting path in G_k are connected by an edge in E_{k+1}. For every edge $e' = (v, w) \in E_{k+1}$ there exists a connecting path $CP_k(v, w)$ in G_k. Dual edge contraction can be implemented by (1) simply renaming all the non-surviving vertices to their surviving parent vertex, (2) deleting all non-surviving edges N and (3) their duals $\overline{N}$.

3.2 Equivalent contraction kernels

The combination of two (and more) successive reductions in an equivalent weighting function allowed Burt [6] to calculate any level of the pyramid directly from the base. Similarly we combine two (and more) dual graph contractions (see Figure 7) of graph G_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 $N_{k,k+2} = N_{k,k+1} \circ N_{k+1,k+2}$ (for simplicity G_k stands for $(G_{k+1}, \overline{G_{k+1}})$):

$$\begin{aligned} C[C[G_k, \langle S_k, N_{k,k+1}\rangle], \langle S_{k+1}, N_{k+1,k+2}\rangle] &= C[G_k, \langle S_{k+1}, N_{k,k+2}\rangle] \\ &= G_{k+2} \end{aligned} \tag{6}$$

Furthermore, the structure of G_{k+1} is determined by G_k and the decimation parameters $(S_k, N_{k,k+1})$. $S_{k+1} := V_{k+2}$ are the vertices surviving from G_k to G_{k+2}. The searched contraction kernels must be formed by edges $N_{k,k+2} \subset E_k$. This is true for $N_{k,k+1}$ but not for $N_{k+1,k+2} \subset E_{k+2}$ if we would simply overlay the two sets of decimation parameters. An edge $e_{k+1} = (v_{k+1}, w_{k+1}) \in N_{k+1,k+2}$ corresponds to a connecting path $CP_k(v_{k+1}, w_{k+1})$ in

$$\begin{tikzcd}
\boxed{(G_k, \overline{G_k})} \arrow[r, "{\langle S_k, N_{k,k+1} \rangle}"] \arrow[rr, bend right, swap, "{\langle S_{k+1}, N_{k,k+2} \rangle}"] & \boxed{(G_{k+1}, \overline{G_{k+1}})} \arrow[r, "{\langle S_{k+1}, N_{k+1,k+2} \rangle}"] & \boxed{(G_{k+2}, \overline{G_{k+2}})}
\end{tikzcd}$$

Figure 7: Equivalent contraction kernel

G_k. (If there are more than one connecting paths, one must be selected). By definition 8, $CP_k(v_{k+1}, w_{k+1})$ consists of one branch of $T_k(v_{k+1})$, one branch of $T_k(w_{k+1})$, and one surviving edge $e_k \in E_k$ connecting the two contraction kernels $T_k(v_{k+1}), T_k(w_{k+1})$.

Definition 9 *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* $CP_k(v_{k+1}, w_{k+1})$*:*

$$bridge(e_{k+1}) := e_k. \tag{7}$$

Two disjoint tree structures connected by a single edge become a new tree structure. The result of connecting all contraction kernels T_k by bridges fulfills the requirements of a contraction kernel:

$$N_{k,k+2} := N_{k,k+1} \quad \cup \bigcup_{e_{k+1} \in N_{k+1,k+2}} \text{bridge}(e_{k+1}) \tag{8}$$

The above process can be repeated on the remaining contraction kernels until the base level 0 contracts in one step into the apex $V_h = \{v_h\}$, the top of the pyramid.

4 Minimum Spanning Tree with *DGC*

The dual graph contraction algorithm [29, 30] 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 5 we will refine the son-son relation to simulate the pop-out phenomena [25, 26], 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 2.

Theorem 4 *The* **equivalent contraction kernel** *of the* **apex** *of the graph pyramid is the* **minimum spanning tree** *of the base level.*

Algorithm 2 – **Borůvka's Algorithm with** *DGC*

Input: Attributed graph $G_0(V, E)$.

```
1: $k = 0$
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:     $k = k + 1$
7: until all connected components of $G$ are contracted into one single vertex.
```

Output: A graph pyramid with an apex.

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 arrived, is the equivalent contraction kernel $N_{0,h}$ (see Section 3.2). Since Algorithm 2 builds minimum spanning tree (see Theorem 3) this implies that $N_{0,h} = MST$. $\square$

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 [30] (applying *DGC*), and produce the graph G_{k+1}. The edges of the graph G_{k+1} correspond in fact to paths that connect trees in the level G_k. If there are parallel paths that connect the same trees they are removed except the one with the minimum-weight middle edge (see Figure 6). 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 3 is given in Figure 8. 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 8.

For a graph $G(V, E)$ where $|V| = n$ and $|E| = m$ follows

Theorem 5 *Algorithm 2 finds the minimum spanning tree after at most* $\log n$ *iterations each of which involves* $O(m + n)$ *operations.*

Proof: In the 3rd step each vertex of G_k chooses one edge that will be contracted in the 5th 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

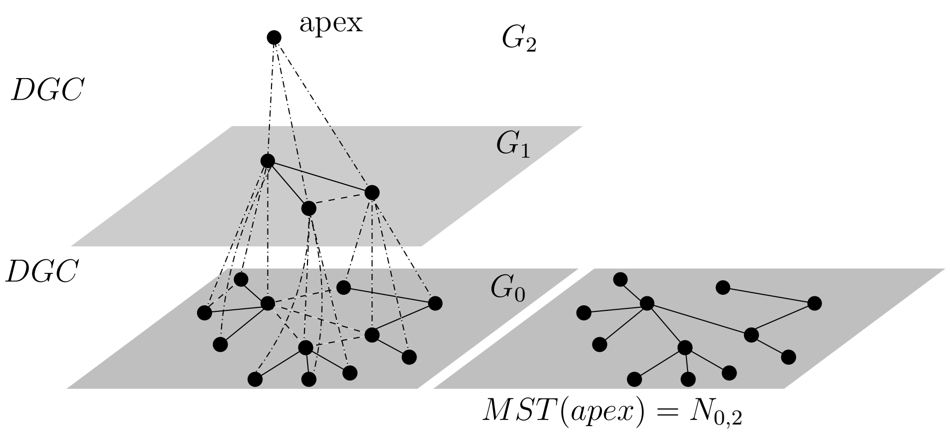


Figure 8: Building MST with DGC.

number of iterations cannot exceed $\log_2 n$. The selection of CC can be performed in $O(n)$ and contraction of edges in $O(m)$. Hence, The overall running time is $O((m+n)\log_2 n)$. □

5 Hierarchy of Partitions

Hierarchies are a significant tool for image partitioning as they naturally mix with homogeneity criteria. Horowitz and Pavlidis [21] define a consistent homogeneity criterium over a set V as a boolean predicate P over its parts $\Phi(V)$ that verifies the consistency property:

$$\forall(x,y) \in \Phi(V) \quad x \subset y \Rightarrow (P(y) \Rightarrow (P(x)). \tag{9}$$

In image analysis Equation (9) states that the subregions of homogeneous region are also homogeneous. It follows that if Pyr is a hierarchy and P a consistent homogeneity criterium on V then the set of maximal elements of Pyr 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.

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 satisfy certain properties. Moreover P is a partition of $V \in G$, and

$$\forall i \neq j \quad CC_i \cap CC_j = \phi \quad \wedge \quad \bigcup_{i=1,\dots,n} CC_i = V. \tag{10}$$

We use the pairwise comparison of neighboring vertices, i.e. partitions to check for similarities [11, 13, 18]. Felzenszwalb in [11] defines a pairwise group comparison function

$Comp(CC_i, CC_j)$ that judges whether or not there is evidence for a boundary between two partitions $CC_i, CC_j \in P$. Note that $Comp(CC_i, CC_j)$ is a boolean comparison function for pairs of partitions and it is not defined yet. 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.

5.1 Building a Hierarchy of Partitions

In this section we present the pairwise comparison function $Comp(\cdot, \cdot)$, which tells us if there is a boundary between pairs of regions (groups). 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. These measures are defined in [11, 13, 18], analogously.

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. 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 3.2).

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 $u \in G_k$:

$$Int(CC_i) = max\{attr_e(e), e \in N_{0,k}(u)\}. \tag{11}$$

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 $u_j \in CC_j$:

$$Ext(CC_i, CC_j) = min\{attr_e(e), e = (v, w) : v \in N_{0,k}(u_i) \wedge w \in N_{0,k}(u_j)\}. \tag{12}$$

In Figure 9 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 3.2).

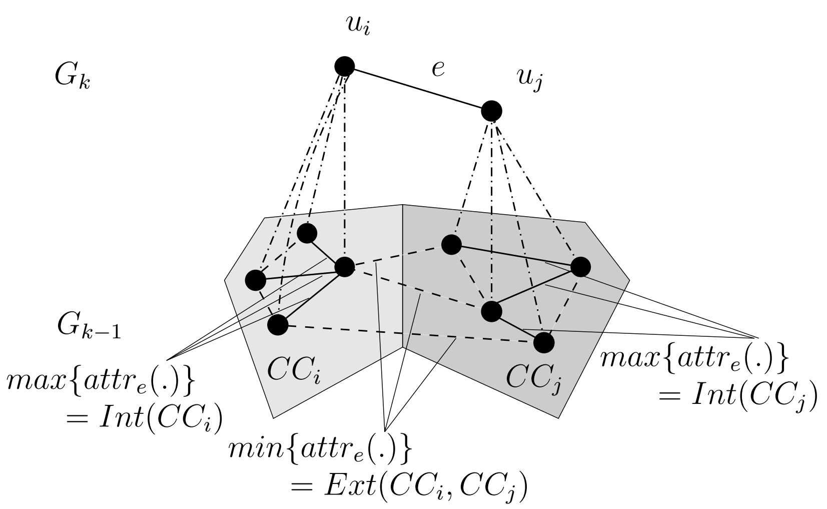


Figure 9: Internal and External contrast.

The pairwise comparison function $Comp(\cdot,\cdot)$ between two connected components CC_i and CC_j can now be defined as:

$$Comp(CC_i, CC_j) = \begin{cases} \text{True} & \text{if } Ext(CC_i, CC_j) > PInt(CC_i, CC_j), \\ \text{False} & \text{otherwise,} \end{cases} \tag{13}$$

where $PInt(CC_i, CC_j)$ is the minimum internal contrast difference between two components:

$$PInt(CC_i, CC_j) = min(Int(CC_i) + \tau(CC_i), Int(CC_j) + \tau(CC_j)). \tag{14}$$

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. The reason for using a threshold function $\tau(CC)$ in Equation (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 $\tau(CC)$ [11]. One can define τ to be function of the size of CC:

$$\tau(CC) = \alpha/|CC|, \tag{15}$$

where $|CC|$ denotes the size of the component CC and α 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.

5.2 Construct Hierarchy of Partitions

The Algorithm 2 in Section 4 in conjunction with the definition of comparison function $Comp(\cdot,\cdot)$ defined in Section 5.1 is used as basis to build the hierarchy of partitions. We proved that Algorithm 2 builds a minimum spanning tree of a attributed graph, so the definition of internal and external contrast are now recalled for clarity:

$$\begin{gathered}Int(CC^k) = max\{attr_e(e), e \in MST(u_k)\}.\\ Ext(CC_i^k, CC_j^k) = min\{attr_e(e), e = (v, w) : v \in MST(u_{k,i}) \wedge w \in MST(u_{k,j})\}.\\ PInt(CC_i^k, CC_j^k) = min(Int(CC_i^k) + \tau(CC_i^k), Int(CC_j^k) + \tau(CC_j^k)).\\ \tau(CC^k) = f(CC^k),\end{gathered} \tag{16}$$

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 3 – **Construct Hierarchy of Partitions**

Input: Attributed graph G_0.

```
 1: $k = 0$
 2: repeat
 3:    for all vertices $u \in G_k$ do
 4:       $E_{min}(u) = argmin\{attr_e(e) \,|\, e = (u, v) \in E_k$ or $e = (v, u) \in E_k\}$
 5:    end for
 6:    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
 7:       include $e$ in contraction edges $N_{k,k+1}$
 8:    end for
 9:    contract graph $G_k$ with contraction kernels, $N_{k,k+1}$: $G_{k+1} = C[G_k, N_{k,k+1}]$.
10:    for all $e_{k+1} \in G_{k+1}$ do
11:       set edge attributes $attr_e(e_{k+1}) = min\{attr_e(e_k) \,|\, e_{k+1} = C[e_k, N_{k,k+1}]\}$
12:    end for
13:    $k = k + 1$
14: 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 3 are left out, it can be shown, that this algorithm produces a MST (Theorem 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 Borôvka's algorithm [4], it builds a $MST(u_k)$ of each region, i.e $N_{0,k}(u_k) = MST(u_k)$ (see Theorem 4).

The idea is to collect smallest weighted edges e (4th 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) (6th step), if these conditions are fulfilled then these two components will be merged (7th 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 graph $G_{k+1} = C[G_k, N_{k,k+1}]$ [32], so that the topology is preserved. 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.

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 (13) is false.

Proposition 1 *For any connected attributed graph $G(V, E, attr_e, attr_v)$ the Algorithm 3 produces 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. Assume 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

$$CC_i^k \in P_k \Rightarrow PInt(CC_i^k, CC_j^k) < Ext(CC_i^k, CC_j^k) = attr_e(e'), \tag{17}$$

and for the edge e'' one shows

$$attr_e(e'') = Ext(CC_j^k, CC_i^k) \leq PInt(CC_j^k, CC_i^k), \tag{18}$$

since $PInt(CC_j^k, CC_i^k) = PInt(CC_i^k, CC_j^k)$ (Equation (16)) and $e'' \in CC_i^k$ it follows

$$\begin{aligned} Ext(CC_j^k, CC_i^k) \leq PInt(CC_i^k, CC_j^k) < Ext(CC_i^k, CC_j^k) \leq PInt(CC_j^k, CC_i^k) \\ \Rightarrow CC_j^k \notin P_k, \end{aligned} \tag{19}$$

contradicting the assumption $CC_j^k \in P_k$. $\square$

Proposition 2 *For any connected attributed graph $G(V, E, attr_e, attr_v)$ Algorithm 3 produces* **partitions on each level** *which are* **invariant** *under any monotone transformation of dissimilarity measure $attr_e$.*

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 3. 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 3 are not changed by the transformation, which results in the invariance of the partitions. □

Proposition 3 *For any connected attributed graph $G(V, E, attr_e, attr_v)$ the* **hierarchy** *over V is* **invariant** *under monotone transformation.*

Proof: Partitions are invariant under the monotone transformation, Proposition 2, 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 *single linkage clustering* [34].

6 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,

$$attr_e(u_i, u_j) = |F(u_i) - F(u_j)|, \tag{20}$$

where F is some feature. Other distances could be used as well e.g. [50],

$$attr_e(u_i, u_j) = e^{\frac{-||F(u_i)-F(u_j)||_2^2}{\sigma_I}}, \tag{21}$$

where F is some feature, and σ_I is a parameter, which controls the scale of proximity measures of F. F could be defined as

$$F(u_i) = I(u_i), \tag{22}$$

for gray value intensity images, or

$$F(u_i) = [v_i, v_i \cdot s_i \cdot \sin(h_i), v_i \cdot s_i \cdot \cos(h_i)], \tag{23}$$

for color images in HSV color distance [50]. 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 [36].

Table 1: Test Images.

Image	Size
Lena	$512 \times 512 = 262144$
Monarch	$768 \times 512 = 393216$
Object 45	$128 \times 128 = 16384$
Ramp	$223 \times 111 = 24753$

For our experiments we use as attributes of edges the difference between pixel intensities, Equation (20), $F(u_i) = I(u_i)$,

$$attr_e(u_i, u_j) = |I(u_i) - I(u_j)|, \tag{24}$$

For color images we run the algorithm by computing the distances (weights) in RGB color space. We choose this simple color distances, in order to study the properties of the algorithm. To compute the hierarchy of partitions we also need to define

$$\tau(CC) = \alpha/|CC|, \tag{25}$$

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 α, which is used to compute the function τ. A larger constant α 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. For computational efficiency we store in vertices the internal contrast $PInt()$ and the size of the connected component $|CC|$ (receptive field).

We use indoor and outdoor RGB images given in Table 1. We found that $\alpha = 300$ produces the best hierarchy of partitions of the images shown in 'Lena' [3] Figure 10, Monarch [3] Figure 11, and Object45 [4] Figure 12 and $\alpha = 1000$ for the image in Figure 13, after the average intensity attribute of vertices is down-projected onto the base grid. Figures 10, 11, 12 and 13 show some of the partitions on different levels of the pyramid and the number of components. In general the top of the pyramid will consist of one vertex, an apex, which represents the whole image.

Note that in all images there are regions of large intensity variability and gradient (see the histograms in Figures 10, 11, 12). This algorithm copes with this kind of gradient and variability. In contrast to [11] the result is a hierarchy of partitions with multiple resolutions suitable for further goal driven, domain specific analysis. On the lower level of the pyramid the image is over segmented (partitioned) whereas in upper it is under

[3] Waterloo image database

[4] Coil 100 image database

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 this would introduce another parameter into the method. The hierarchy of partitions can also be built from an oversegmented image to overcome the problem of noisy pixels. Note that the influence of τ in decision criterion is smaller as the region gets bigger for a constant α. The constant α is used to produce a kind of the oversegmented image and the influence of τ is smaller after each level of the pyramid. For an oversegmeted image, where the size of regions is large, the algorithm becomes parameterless.

7 Conclusion and Outlook

In this technical report we introduced 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. 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 our framework is general enough, we can use RAGs of any oversegmented image and 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. Our future work is to automatically extract this parameter from the image and also to define different comparison functions which will prefer learned regions of specific shapes.

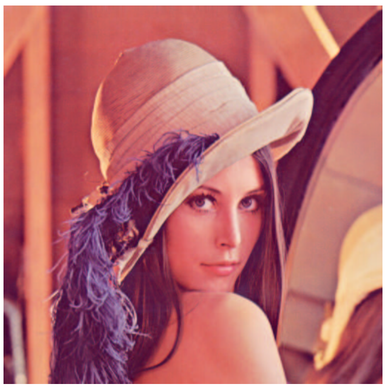
References

[1] H. Bischof and W. G. Kropatsch. Hopfield networks for irregular decimation. In W. Pölzleitner and E. Wenger, editors, *Image Analysis and Synthesis*, pages 317–327. OCG-Schriftenreihe, Österr. Arbeitsgemeinschaft für Mustererkennung, R. Oldenburg, 1993. Band 68.

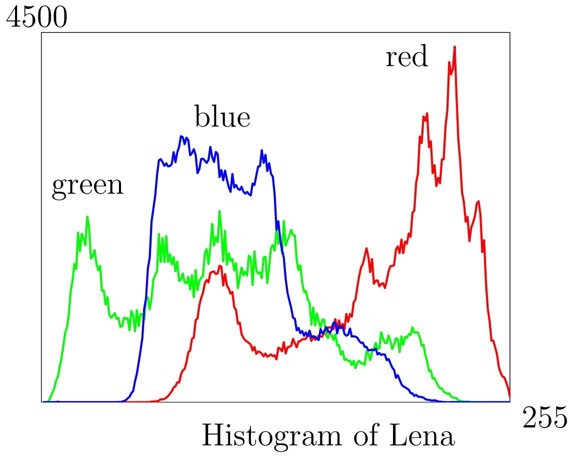
[2] M. Bister, J. Cornelis, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, Vol. 11(No. 9):pp. 605–617, September 1990.

[3] M. Borowy and J. Jolion. A pyramidal framework for fast feature detection. In *Proc. of 4th Int. Workshop on Parellel Image Analysis*, pages 193–202, 1995.

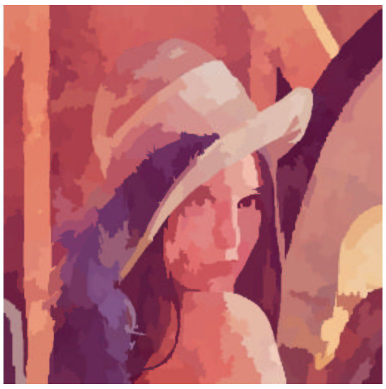
Lena



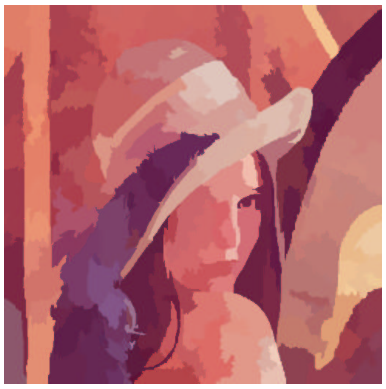
(a) Level 0
(262 144 partitions)



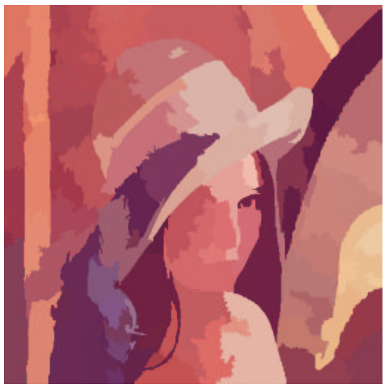
Histogram of Lena



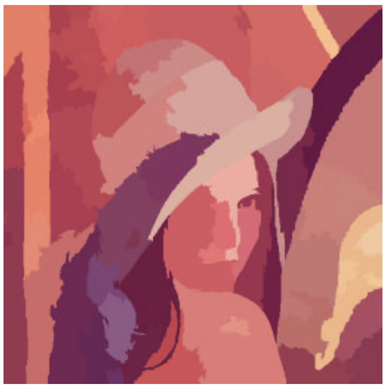
(b) Level 11 (228)



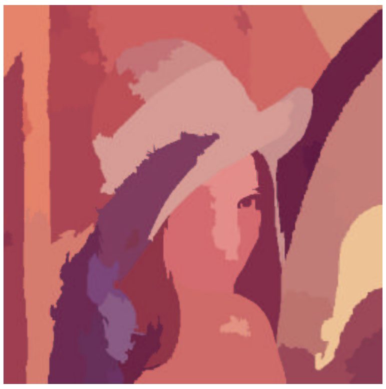
(c) Level 12 (133)



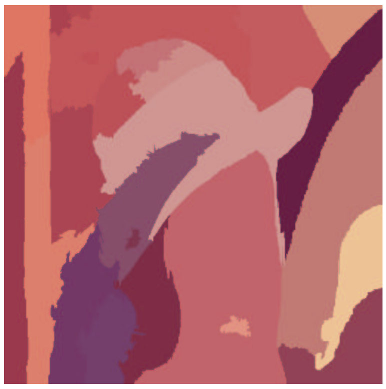
(d) Level 13 (78)



(e) Level 14 (48)



(f) Level 15 (27)



(g) Level 16 (19)

Figure 10: Some levels of the partitioning of "Lena" and the number of components.

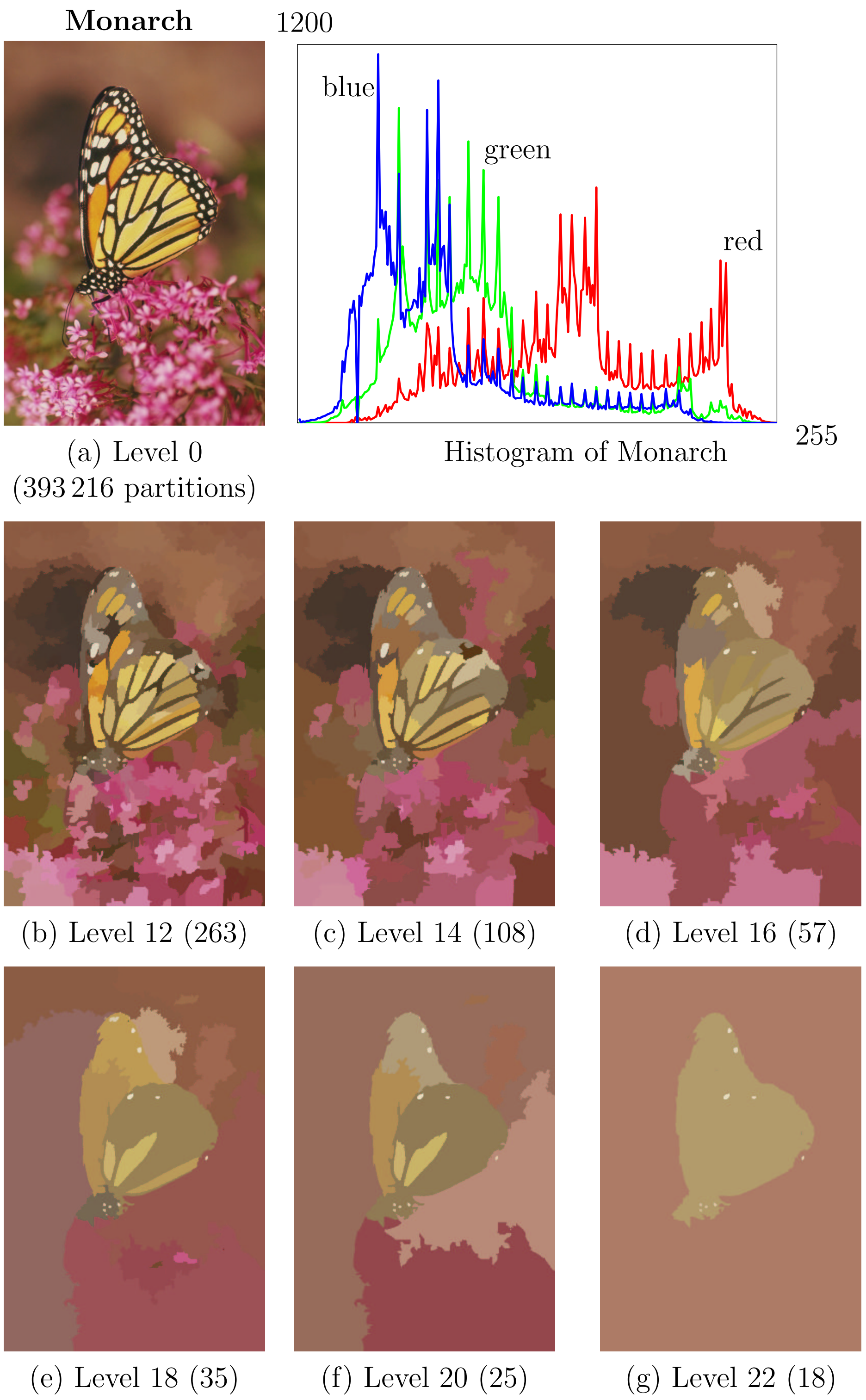
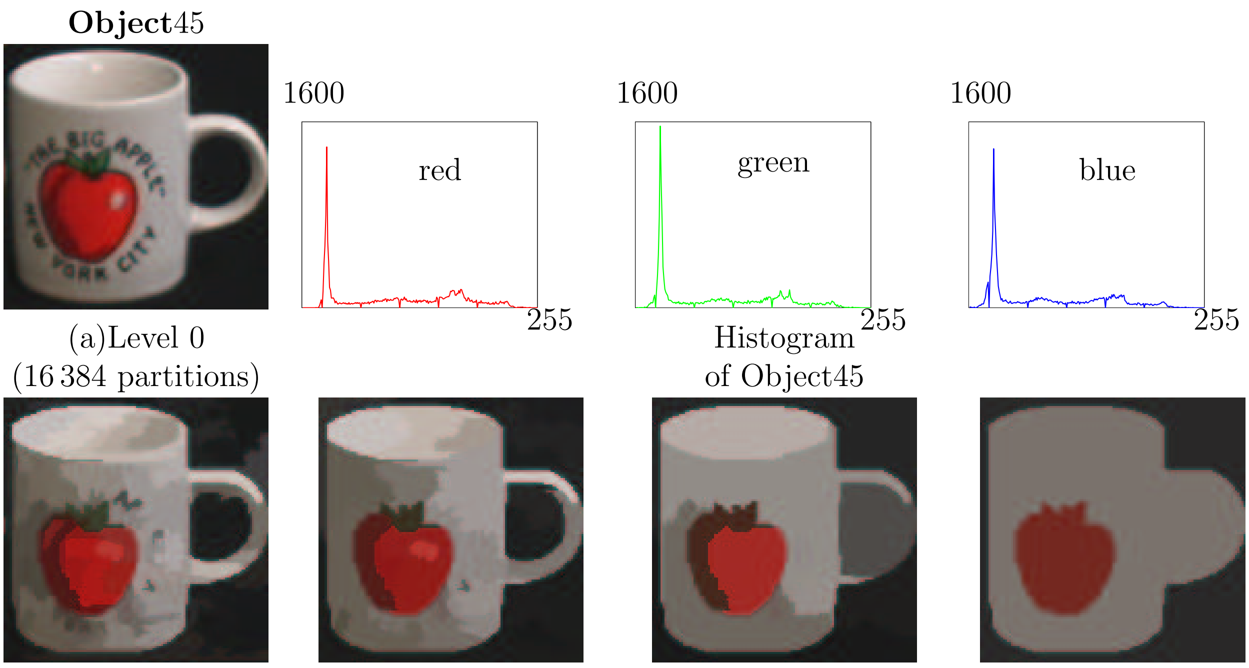
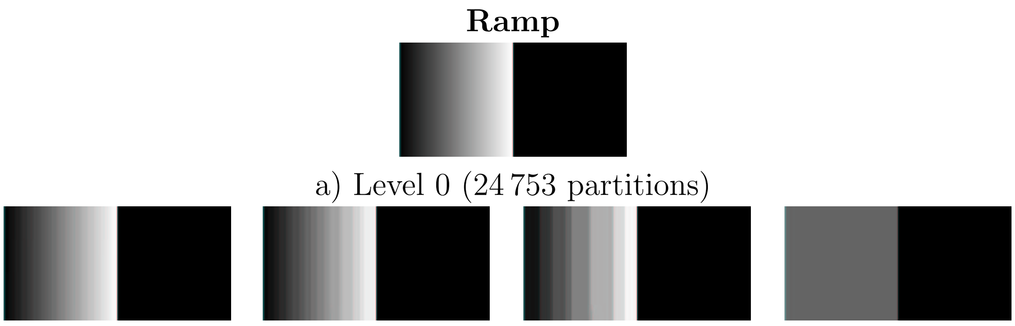


Figure 11: Some levels of the partitioning of “Monarch” and the number of components.



(b) Level 8 (129) (c) Level 10 (43) (d) Level 12 (13) (e) Level 14 (3)

Figure 12: Some levels of the partitioning of "Object45" and the number of components.



(b) Level 8 (44) (c) Level 9 (25) (d) Level 10 (13) (e) Level 14 (2)

Figure 13: Some levels of the partitioning of a ramp and the number of components.

[4] O. Borůvka. O jistém problému minimálnim. *Práce Mor. Přírodvěd. Spol. v Brně (Acta Societ. Scienc. Natur. Moravicae)*, III(3):37–58, 1926.

[5] O. Boykov, Y. Veskler and R. Zabih. Markov Random Fields with Efficient Approximations. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages pp. 648–655, 1998.

[6] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, Vol. COM-31(No.4):pp.532–540, April 1983.

[7] K. Cho and P. Meer. Image Segmentation from Consensus Information. *CVGIP: Image Understanding*, 68 (1):72–89, 1997.

[8] R. Cole, K. P. N., and T. R. E. A Linear-Work Parallel Algorithm for Finding Minimum Spanning Trees. In *Proc. SPAA: Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 11–15, 1994.

[9] M. Cooper. The Tractibility of Segmentation and Scene Analysis. In *IJCV*, volume Vol. 30 No. 1, pages 27–42, 1998.

[10] T. K. Dey, H. Edelsbrunner, S. Guha, and D. V. Nekhayev. Topology Preserving Edge Contraction. Technical Report RGI-Tech-98-018, Raindrop Geomagic Inc., Research Triangle Park, North Carolina., 1998.

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Image Segmentation Using Local Variation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages pp:98–104, June 1998.

[12] M. Fiedler. A Property of Eigenvectors of Nonnegative Symetric Matrices and its Application to Graph Theory. *Checz Math. Journal*, Vol. 25(No 100):pp.619–633, 1975.

[13] B. Fischer and J. M. Buhmann. Data Resampling for Path Based Clustering. In L. van Gool, editor, *Proceedings of 24th DAGM Symposium*, pages 117–124, Swiss, 2002. Springer Verlag LNCS 2449.

[14] M. L. Fredman and D. E. Willard. Trans-dichotomous Algorithms for Minimum Spanning Trees and Shortest Paths. In *31st IEEE Symp. Foundations of Comp. Sci.*, pages 719–725, 1990.

[15] H. Gabow, Z. Galil, T. Spencer, and R. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graps. *Combinatorica*, No. 6:109–122, June 1986.

[16] Y. Gdalyahu, D. Weinshall, and M. Werman. Self-Organization in Vision: Stochastic Clustering for Image Segmentation, Perceptual Grouping, and Image Database Organization. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 23(10):pages 1053–1074, 2001.

[17] S. Geman and D. Geman. Stochastic Relaxation, Gibbs distribution, and the Bayesian Restoration of Images. *PAMI*, Vol. 6:pp.721–741, November 1984.

[18] L. Guigues, L. M. Herve, and J.-P. Cocquerez. The Hierarchy of the Cocoons of a Graph and its Application to Image Segmentation. *Pattern Recognition Letters*, 24(8):pages 1059–1066, 2003.

[19] G. Guy and G. Medioni. Inferring Global Perceptual Contours for Pairwise Clustering. *International Journal of Computer Vision*, pages 424–430, 1996.

[20] Y. Haxhimusa, R. Glantz, M. Saib, G. Langs, and W. G. Kropatsch. Logarithmic Tapering Graph Pyramid. In L. van Gool, editor, *Proceedings of 24th DAGM Symposium*, pages 117–124, Swiss, 2002. Springer Verlag LNCS 2449.

[21] S. Horowitz and T. Pavlidis. Picture Segmentation by a Tree Traversal Algorithm. *J. Assoc. Compt. Math.*, Vol. 2(23):pages:368–388, 1976.

[22] J.-M. Jolion. Stochastic pyramid revisited. *Pattern Recognition Letters*, 24(8):pp. 1035–1042, 2003.

[23] J.-M. Jolion and A. Montanvert. The adaptive pyramid, a framework for 2D image analysis. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(3):pp.339–348, May 1992.

[24] J.-M. Jolion and A. Rosenfeld. *A Pyramid Framework for Early Vision.* Kluwer, 1994.

[25] B. Julesz. Textons, the Elements of Texture Perception and Their Interactions. *Nature*, (No. 290):pp.91–97, 1981.

[26] B. Julesz and J. R. Bergen. Textons, the fundamental elements in preattentive vision and perception of textures. *The Bell System Technical Journal*, Vol. 62(No. 6):pp.1619–1645, July-August 1983.

[27] J. A. K. and D. R.C. *Algorithms for Clustering Data.* Prentice Hall, Berlin, 1988.

[28] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithms to find minimum spanning trees. *J. ACM*, 42(2):321–328, 1995.

[29] W. G. Kropatsch. Building Irregular Pyramids by Dual Graph Contraction. Technical Report PRIP-TR-35, Institute f. Automation 183/2, Dept. for Pattern Recognition and Image Processing, TU Wien, Austria, 1994.

[30] W. G. Kropatsch. Building Irregular Pyramids by Dual Graph Contraction. *IEE-Proc. Vision, Image and Signal Processing*, Vol. 142(No. 6):pp. 366–374, December 1995.

[31] W. G. Kropatsch. Equivalent Contraction Kernels and The Domain of Dual Irregular Pyramids. Technical Report PRIP-TR-42, Institute f. Automation 183/2, Dept. for Pattern Recognition and Image Processing, TU Wien, Austria, 1995. Also available through http://www.prip.tuwien.ac.at/ftp/pub/publications/trs/tr42.ps.gz.

[32] W. G. Kropatsch, A. Leonardis, and H. Bischof. Hierarchical, Adaptive and Robust Methods for Image Understanding. *Surveys on Mathematics for Industry*, No.9:1–47, 1999.

[33] J. B. J. Kruskal. On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem. In *Proc. Am. Math. Soc.*, volume 7, pages 48–50, 1956.

[34] J. Lance and W. Williams. A General Theory of Classificatory Sorting Strategies. *Comput. J*, No. 9:pages:51–60, 1967.

[35] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and Texture Analysis for Image Segmentation. *International Journal on Computer Vision*, 43(1):7–27, 2001.

[36] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *The International Conference on Computer Vision*, pages 918–925, 1999.

[37] C. Mathieu, I. E. Magnin, and C. Baldy-Porcher. Optimal stochastic pyramid: segmentation of MRI data. *Proc. Med. Imaging VI: Image Processing*, SPIE Vol.1652:pp.14–22, Feb. 1992.

[38] P. Meer. Stochastic image pyramids. *Computer Vision, Graphics, and Image Processing*, Vol. 45(No. 3):pp.269–294, March 1989.

[39] P. Meer, D. Mintz, A. Montanvert, and A. Rosenfeld. Consensus vision. In *AAAI-90 Workshop on Qualitative Vision*, pages pp.111–115, Boston, Massachusetts, USA, July 29 1990.

[40] F. Meyer. Graph based morphological segmentation. In W. G. Kropatsch and J.-M. Jolion, editors, *2nd IAPR-TC-15 Workshop on Graph-based Representation*, pages 51–60. OCG-Schriftenreihe, Band 126, Österreichische Computer Gesellschaft, 1999.

[41] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchical image analysis using irregular tesselations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(No.4):pp.307–316, April 1991.

[42] M. Pavan and M. Pelillo. A New Graph-Theoretic Approach to Clustring and Segmentation. In *CVPR03*, pages 145–152, 2003.

[43] T. Pavlidis. *Structural Pattern Recognition.* Springer Verlag, 1977.

[44] P. Perona and W. Freeman. A Factorization Approach to Grouping. In H. Burkhardt and B. Neumann, editors, *LNCS*, volume 1406, pages 655–670. Computer Vision Springer Verlag, 1998.

[45] R. C. Prim. Shortest connection networks and some generalizations. *Bell Sys. Tech. J.*, Vol. 36:pp.1389–1401, 1957.

[46] J. Puzicha, T. Hofmann, and J. Buhmann. Unsupervised testure segmentation in a deterministic annealing framework. *IEEE Transaction on Pattern Recognition Analysis and Machine Intellegence*, 20(8):803–818, 1998.

[47] A. Rosenfeld, editor. *Multiresolution Image Processing and Analysis.* Springer Verlag, 1984.

[48] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems Man and Cybernetics*, 6(6):420 – 433, 1976.

[49] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages pp. 70–77, South Carolina, 2000.

[50] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. In *Proceedings IEEE Conf. Computer Vision and Pattern*, pages 731–737, 1997.

[51] K. Thulasiraman and M. N. S. Swamy. *Graphs: Theory and Algorithms.* Wiley-Interscience, 1992.

[52] R. Urquhart. Graph Theoretical Clustering Based on Limited Neighborhood Sets. *Pattern Recognition*, Vol. 13:3:pp.173–187, 1982.

[53] Y. Weiss. Segmentation Using Eigenvectrors: a Unifying View. In *Proceedings of the internaltion Conf. on Computer Vision*, pages 975–982, 1999.

[54] M. Wertheimer. Über Gestaltheorie. *Philosophische Zeitschrift für Forschung und Aussprache*, 1:30–60, 1925.

[55] D. E. Willis. *Source Book of Gestalt Psychology.* Harcourt, Brace and Co. New York., 1938, reprinted by Gestalt Journal Press, New York 1997.

[56] Z. Wu and R. Leahy. An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.15(11):pp. 1101–1113, November 1993.

[57] Y. Gdalyahu, D Weinshall, and M. Werman. A Randomized Algorithm for Pairwise Clustering. *Neural Information Processing Systems*, NIPS 11:pp. 424–430, 1998.

[58] S. Yu and J. Shi. Segmentation with pairwise attraction and repulsion. In *Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV'01)*, pages pp.52–58. IEEE Computer Society, July 2001.

[59] C. Zahn. Graph-theoretical methods for detecting and describing gestal clusters. In *IEEE Trans. Comput.*, volume 20, pages 68–86, 1971.