Contraction Kernels and Combinatorial Maps

Luc Brun† and Walter Kropatsch⋆‡

† Laboratoire d'Études et de Recherche en Informatique
I.U.T. de Reims 51.059 Reims - France
and
‡ Institute for Computer-aided Automation
Pattern Recognition and Image Processing Group
Vienna Univ. of Technology- Austria

† brun@leri.univ-reims.fr, ‡ krw@prip.tuwien.ac.at

Abstract

Graph pyramids are made of a stack of successively reduced graphs embedded in the plane. Such pyramids overcome the main limitations of their regular ancestors. The graphs used in the pyramid may be region adjacency graphs, dual graphs or combinatorial maps. Compared to the usual graph data structures, combinatorial maps offer an explicit encoding of the orientation of edges around vertices. Each combinatorial map in the pyramid is generated from the one below by a set of edges to be contracted. This contraction process is controlled by kernels that can be combined in many ways. We show in this paper, that kernels producing a slow reduction rate can be combined to speed up reduction. Or, conversely, kernels decompose into smaller kernels that generate a more gradual reduction. We also propose one sequential and one parallel algorithm to compute the contracted combinatorial maps defined by kernels.

1. Introduction

Regular image pyramids have been introduced 1981/82 [6] as a stack of images with decreasing resolutions. Since then, regular pyramids have been widely used in image segmentation [6] and image analysis [14]. However, the rigidity of regular pyramids induce several drawbacks such as the shift-dependence problem and the limited number of regions encoded at a given level of the pyramid [1]. Irregular pyramids overcome these negative properties while keeping the main advantages of their regular ancestors [12]. These pyramids are defined as a stack of successively reduced graphs. Each graph is build from the graph below by selecting a set of vertices named surviving vertices and mapping each non surviving vertex to a surviving one [12]. Therefore each non surviving vertex is the

⋆ This Work was supported by the Austrian Science Foundation under P14445-MAT.

child of some surviving one which is the father of all the non surviving vertices mapped to it. Given one vertex defined in one graph of the pyramid, its set of children in the base level graph is called its receptive field.

If the initial graph is defined from the regular grid, we may associate one pixel to each vertex and link vertices according to the adjacency relationships defined on the regular grid. Using an initial planar grid (like 4-neighborhood) the initial graph and its reduced versions are planar. Then, the receptive field of any vertex in the pyramid defines one connected region of the associated image. Therefore, each graph of the pyramid encodes a partition of the initial image. The edges of the reduced graphs encode adjacency relationships between regions. According to the data structure used to encode graphs and the reduction operation, edges may encode coarse informations such as the existence of some common boundary between two regions, or finer ones such as multiple boundaries or surrounding relationships. Note that, these finer topological relationships are required by some applications [10].

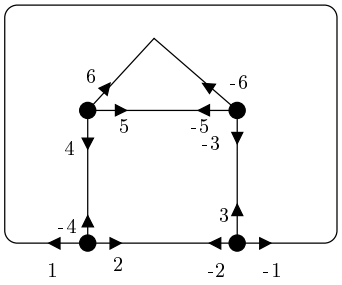
Using region adjacency graphs two surviving vertices are linked by an edge if they have at least two adjacent children [12]. Such graph pyramids have been successively applied to image segmentation [12] and image analysis [13]. However, the above decimation process does not allow to encode multiple boundaries between regions nor to discriminate two adjacent regions from one region surrounding the other. Moreover, the set of edges incident to one vertex is not ordered according to the orientation of the plane.

The reduction operation may also be encoded by edge contraction [8]. This operation contracts one edge and its two end points into a single vertex. The contraction of a graph reduces the number of vertices while maintaining the connections to other vertices. As a consequence some redundant edges such as self-loops or double edges may occur. These redundant edges may be characterized in the dual of the contracted graph. The removal of such edges is called a dual decimation step. The resulting graph encodes multiple boundaries between regions. Moreover, edges encoding the adjacency between one region and a surrounding one may be characterized in the dual graph. The reduction operation is thus performed in two steps: a first decimation process which identifies vertices and then a dual decimation stage which removes redundant edges. Experiences with connected component analysis [10] and with universal segmentation [9] show the great potential of this concept. However, since decimation and dual decimation require respectively features of the initial and dual graphs, both graphs must be encoded and maintained [8]. Therefore, any contraction operation in the initial graph must be followed by a removal operation in its dual. In the same way, during the dual decimation stage, any

removal in the initial graph must be followed by a contraction operation in its dual. Moreover, the orientation of edges around one vertex is not explicitly encoded by the data structure [4].
The remaining of the paper is as follows: In section 2. we present the combinatorial map model together with its main properties. In section 3. we introduce the notion of decimation parameters within the combinatorial map framework. Decimation parameters allow to map each vertex of the combinatorial map on one processor and to perform the decimation process on a parallel machine but provide only a slow decimation rate. The more general notion of contraction kernel is also introduced in Section 3. Such kernels provide a better control of the decimation rate. Moreover, the contracted combinatorial map defined by a contraction kernel may be computed on highly parallel machines. One sequential and one parallel algorithm computing the contracted combinatorial map are provided in Section 4.

2. Combinatorial Maps

A combinatorial map [7] may be seen as a planar graph encoding explicitly the orientation of edges around a given vertex. Fig. 1 demonstrates the derivation of a combinatorial map from a plane graph. First edges are split into two half edges called *darts*, each dart having its origin at the vertex it is attached to. The fact that two half-edges (darts) stem from the same edge is recorded in the reverse permutation α. A second permutation σ, called the successor permutation, defines the (local) arrangement of darts around a vertex. Each orbit of σ is associated to one vertex and encodes the set of darts encountered when turning counterclockwise around this vertex(see Fig. 1).



$$\alpha = (1,-1)(2,-2)(3,-3)(4,-4)(5,-5)(6,-6)$$

$$\sigma = (1,2,-4)(-2,-1,3)(-3,-6,-5)(5,6,4)$$

Fig. 1. *One planar graph encoded by a combinatorial map*

The symbols $\alpha^*(d)$ and $\sigma^*(d)$ stand, respectively, for the α and σ orbits of the

dart d. More generally, if d is a dart and π a permutation we will denote the π-orbit of d by $\pi^*(d)$.
A combinatorial map G is the triplet $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, where $\boldsymbol{\mathcal{D}}$ is the set of darts and σ, α are two permutations defined on $\boldsymbol{\mathcal{D}}$ such that α is an involution:

$$\forall d \in \boldsymbol{\mathcal{D}} \quad \alpha^2(d) = d$$

Given two combinatorial maps $G_1 = (\boldsymbol{\mathcal{D}}1, \sigma_1, \alpha_1)$ and $G_2 = (\boldsymbol{\mathcal{D}}_2, \sigma_2, \alpha_2)$ an application ψ from $\boldsymbol{\mathcal{D}}_1$ to $\boldsymbol{\mathcal{D}}_2$ defines a morphism [7] between G_1 and G_2 iff:

$$\forall d \in \boldsymbol{\mathcal{D}}_1 \begin{cases} \psi(\alpha_1(d)) = \alpha_2(\psi(d)) \\ \psi(\sigma_1(d)) = \sigma_2(\psi(d)) \end{cases} \tag{1}$$

If ψ is bijective, it defines an isomorphism and equation 1 is equivalent to:

$$\forall d \in \boldsymbol{\mathcal{D}}_1 \begin{cases} \alpha_1(d) = \psi^{-1}(\alpha_2(\psi(d))) \\ \sigma_1(d) = \psi^{-1}(\sigma_2(\psi(d))) \end{cases} \tag{2}$$

Using combinatorial maps, contraction and removal operations may be defined in order to preserve the orientation of darts around each vertex [4]. Given these two basic operations decimation and dual decimation parameters may be defined in order to produce a stack of successively reduced combinatorial maps [5]. The expected advantages of combinatorial maps within the irregular pyramid framework are:

1. Combinatorial maps encode multiple boundaries between regions. Moreover, edges encoding surrounding relationships may also be characterized.
2. Combinatorial maps explicitly encode the orientation of darts around one vertex. This information is not encoded by region adjacency graphs nor explicitely available in dual graph data structures.
3. Given a combinatorial map G defined by one set of darts $\boldsymbol{\mathcal{D}}$, and the permutations σ and α, its dual $\overline{G}$ is defined on the same set of darts by the permutations $\varphi = \sigma \circ \alpha$ and α. The simplicity and the efficiency of this transformation allows us to avoid an explicit encoding of the dual graph [4]. Therefore, only one data structure has to be encoded and maintained along the pyramid [5].
4. The combinatorial map formalism may be extended to higher dimensions [11]. The definition of a partition of the 3D discrete grid using combinatorial maps is an active research field [2].

3. Decimation Parameters and Contraction Kernels

Decimation and dual decimation processes induce both contraction and removal operations. These operations may be defined [4] in order to preserve the

orientation of the initial combinatorial map. Therefore, if the combinatorial map encodes a partition, two vertices linked by an edge, encode two adjacent regions. Then, the order defined on the darts of the contracted vertex encodes the set of regions encountered when turning counterclockwise around the union of the two regions.
In order to preserve the number of connected components of the initial combinatorial map, bridges and self-loops must be excluded from removal and contraction operations [4] respectively. Since the contraction operation is not defined for self-loops, several contraction operations may be performed simultaneously only if we ensure that no self loops may be contracted. This constraint may be solved by using a decimation parameter [5](see Fig. 2(a)). Given a combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, a decimation parameter is a set of darts $\boldsymbol{\mathcal{D}'}$ such that:

1. $\alpha(\sigma^*(\boldsymbol{\mathcal{D}'})) \cap \sigma^*(\boldsymbol{\mathcal{D}'}) = \emptyset$ and:
2. $\forall d \in \boldsymbol{\mathcal{D}} - \sigma^*(\boldsymbol{\mathcal{D}'}) \quad \exists!\ d' \in \boldsymbol{\mathcal{D}'} \quad | \quad \alpha(d') \in \sigma^*(d)$

The set $\boldsymbol{\mathcal{D}} - \alpha^*(\boldsymbol{\mathcal{D}'})$ is the set of surviving darts and is denoted $\mathcal{SD}$. Intuitively, the definition of a decimation parameter consists to select a set of surviving vertices $\sigma^*(\boldsymbol{\mathcal{D}'})$ and a set of edges $\alpha^*(\boldsymbol{\mathcal{D}'})$ to be contracted. Constraint 1 insures that two surviving vertices are not connected in the induced sub combinatorial map [4]. Constraint 2 insures that exactly one of multiple edges incident to a non surviving vertex is contracted. Note that each vertex survives or is adjacent to a surviving vertex. Therefore, a decimation parameter may be understood as a maximal independent vertex set [12] in which we have specified the contracted edges.
Since each non surviving vertex has to be adjacent to a surviving one, the definition of a decimation parameter may be interpreted as the construction of a spanning forest each tree having depth one. Such kernels are a particular case of a more general object named contraction kernel [3] (see Fig. 2(b)):

Definition 1. Contraction Kernel Given a connected combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, the set $K \subset \boldsymbol{\mathcal{D}}$ will be called a contraction kernel iff it is a forest of G not including all darts of G: $\mathcal{SD} = \boldsymbol{\mathcal{D}} - K \neq \emptyset$.
The set $\mathcal{SD}$ is called the set of surviving darts.

The definition of a contraction kernel generalizes the one of decimation parameters. Since the set of darts to be contracted forms a forest of the initial combinatorial map, no self loop may be contracted and the contraction operation is well defined. Note that the forest K is not required to be a spanning one. This last point allows to apply contractions on some parts of the initial

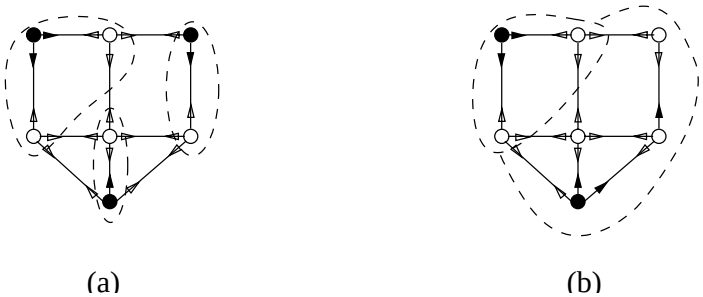


Fig. 2. *A decimation parameter (a) and a contraction kernel (b). The set of contracted edges is represented by black arrows. Surviving vertices are represented by filled circles. The receptive field of each surviving vertex is surrounded by dashed lines*

combinatorial map leaving the other parts unchanged. More generally, the set of childs of a given vertex defined in the contracted combinatorial map may now vary from a single vertex to a tree with any height. Moreover, since the vertices of the graph are implicitly defined by their darts we must require that at least one dart survives

Given an initial combinatorial map G_0 contracted by a contraction kernel K_1 into a reduced combinatorial map $G_1 = G_0/K_1$, any contraction kernel K_2 defined on G_1 is called a successor of K_1. This relation is denoted $K_1 \prec K_2$. Moreover, given two contraction kernels K_1 and K_2 both defined on G_0, we say that K_2 includes K_1 iff the set of darts of K_1 is included in the one of K_2. This relation is denoted $K_1 \subset K_2$. The successive applications of two successive kernels K_1 and K_2 on a combinatorial map G_0 defines a contracted combinatorial map $G_2 = (G_0/K_1)/K_2$. We have shown [3] that the union of the darts of K_1 and K_2 defines a new contraction kernel $K_3 = K_1 \cup K_2$ of G_0. Applied on G_0, this kernel provides the same contracted combinatorial map than the successive applications of K_1 and K_2. Conversely, given two contraction kernels K_1 and K_3 of G_0 such that $K_1 \subset K_3$, the set of darts of K_3 which are not in K_1 defines a contraction kernel K_2 of $G_1 = G_0/K_1$ such that $G_1/K_2 = G_0/K_3$ (see proof in [3]).

Therefore, the successive application of small kernels defined by decimation parameters may be replaced by a bigger contraction controlled by a contraction kernel. Conversely, one contraction kernel may be decomposed into smaller kernels in order to define some intermediate contracted combinatorial maps. Such results provide a better flexibility in the design of contraction kernels and allow us to adapt more efficiently the decimation rate to the data.

4. Computation of the contracted combinatorial map

Given an initial combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ and a contraction kernel K, the set of darts of the contracted combinatorial map $G' = G/K$ is equal to $\mathcal{SD} = \boldsymbol{\mathcal{D}} - K$ (see definition 1). Surviving darts may be connected in G by connecting walks:

Definition 2. Connecting walk
Given a connected combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, a contraction kernel K and a dart $d \in \mathcal{SD}$, the **connecting walk** associated to d is equal to:

$$CW(d) = d, \varphi(d), \ldots, \varphi^{n-1}(d) \text{ with } n = Min\{p \in \mathbb{N}^* \mid \varphi^p(d) \in \mathcal{SD}\}$$

Note that only the first dart of each connecting walk survives. Each connecting walk $CW(d)$ connects the surviving darts d and $\varphi^n(d)$ by a sequence of non surviving darts. Moreover, the set of connecting walks defines a partition of the initial set of darts $\boldsymbol{\mathcal{D}}$ [3]. We have thus:

$$\boldsymbol{\mathcal{D}} = \bigsqcup_{d \in \mathcal{SD}} CW(d) \tag{3}$$

Where $\bigsqcup$ denotes an union of disjoint sets.
If $\boldsymbol{\mathcal{D}}_K$ denotes the set of connecting walks defined on G by the contraction kernel K, the following applications define permutations on $\boldsymbol{\mathcal{D}}_K$(see proof in [3]):

$$\alpha_K \begin{pmatrix} \boldsymbol{\mathcal{D}}_K & \to \boldsymbol{\mathcal{D}}_K \\ CW(d) & \mapsto CW(\alpha(d)) \end{pmatrix}, \varphi_K \begin{pmatrix} \boldsymbol{\mathcal{D}}_K & \to \boldsymbol{\mathcal{D}}_K \\ CW(d) = d, \ldots, \varphi^{n-1}(d) & \mapsto CW(\varphi^n(d)) \end{pmatrix}$$

Since α_K and φ_K define two permutations on $\boldsymbol{\mathcal{D}}_K$, $\sigma_K = \varphi_K \circ \alpha_K$ is a permutation as the composition of two permutations. The two permutations α_K and σ_K structure the set of connecting walks $\boldsymbol{\mathcal{D}}_K$ into a combinatorial map $G_K = (\boldsymbol{\mathcal{D}}_K, \sigma_K, \alpha_K)$ [3]. Moreover, since each connecting walk contains only one surviving dart, we can consider CW as a bijective application from $\mathcal{SD}$ to $\boldsymbol{\mathcal{D}}_K$. If $G' = (\mathcal{SD}, \sigma', \alpha')$ denotes the contracted combinatorial map, CW defines an isomorphism between G' and G_K (see proof in [3]). Therefore (see equation 2):

$$\forall d \in \mathcal{SD} \quad \alpha'(d) = CW^{-1}(\alpha_C(CW(d))) = CW^{-1}(CW(\alpha(d)))) = \alpha(d)$$

In the same way, we obtain for any dart d in $\mathcal{SD}$

$$\sigma'(d) = CW^{-1}(\sigma_C(CW(d))) = CW^{-1}(\varphi_K \circ \alpha_K(CW(d))) = \varphi^n(\alpha(d)) \tag{4}$$

with $n = Min\{p \in \mathbb{N}^* \mid \varphi^p(\alpha(d)) \in \mathcal{SD}\}$

Note that, $\varphi^{n-1}(\alpha(d))$ is the last dart of $CW(\alpha(d))$. Therefore, the permutation α remains unchanged in the contracted combinatorial map while the permutation σ' maps each surviving dart d to the φ-successor of the last dart of $CW(\alpha(d))$. Therefore, the computation of the σ-successor of a dart d in the contracted combinatorial map requires to traverse $CW(\alpha(d))$. Sequential algorithm 1 computes the σ-successor of all the surviving darts in the contracted combinatorial map. Since the set of connecting walks forms a partition of $\boldsymbol{\mathcal{D}}$ (see equation 3), Algorithm 1 has to traverse $|\boldsymbol{\mathcal{D}}|$ darts. Its complexity is thus equal to $\mathcal{O}(|\boldsymbol{\mathcal{D}}|)$.

```
dart contracted_sigma(G = (𝒟, σ, α), K)
{
    For each d ∈ S𝒟 = 𝒟 − K
    do
        d' = φ(α(d)) = σ(d) // Second dart of CW(α(d))

        while(d' ∈ K) // computation of CW(α(d))
            d' = φ(d')
        σ'(d) = d'
    done
}
```

Algorithm 1: *Computation of the permutation σ' of the contracted combinatorial map $G' = (\mathcal{SD}, \sigma', \alpha)$*

The basic idea of a parallel implementation of the algorithm *contracted_sigma* is to compute concurrently for each dart d, the first surviving dart encountered when traversing the φ-orbit of d from d. The result of this parallel computation is stored in an array survive initialized to $survive[d] = d$ for each dart (see Algorithm 2, lines 3 to 4). We have thus:

$$\forall d \in \boldsymbol{\mathcal{D}} \quad \begin{cases} survive[d] = \varphi^q(d) & \text{with } q = Min\{p \in \mathbb{N} \mid \varphi^p(d) \in \mathcal{SD}\} \\ \qquad\quad = \varphi^{n-1}(d) & \text{with } n = Min\{p \in \mathbb{N}^* \mid \varphi^{p-1}(d) \in \mathcal{SD}\} \end{cases}$$

Moreover, using equation 4:

$$\sigma'(d) = \varphi^n(\alpha(d)) = \varphi^{n-1}(\varphi(\alpha(d))) = \varphi^{n-1}(\sigma(d))$$

with $n = Min\{p \in \mathbb{N}^* \mid \varphi^p(\alpha(d)) \in \mathcal{SD}\} = Min\{p \in \mathbb{N}^* \mid \varphi^{p-1}(\sigma(d)) \in \mathcal{SD}\}$. Therefore, (see Algorithm 2, lines 9 to 10):

$$\sigma'(d) = \varphi^{n-1}(\sigma(d)) = survive[\sigma(d)]$$

```
1   dart parallel_contracted_sigma(G = (𝒟, σ, α), K)
2   {
3       for each d in 𝒟 do in parallel
4           survive[d] = d
5       for each d in 𝒟 do in parallel
6           while(survive[d] ∈ K)
7               survive[d] = survive[φ(d)]
8
9       for each d in S𝒟 do in parallel
10          σ'(d) = survive[σ(d)]
11  }
```

Algorithm 2: *Parallel computation of the permutation* $\sigma'(d)$

The parallel complexity of algorithm 2 is determined by the number of elementary steps performed in the second loop(lines 5 to 7). This number is equal to the cyclic distance between d and its associated surviving dart . If we denote by D the maximum of these distances, the parallel algorithm will terminate after D steps. Therefore, worst case parallel complexity of our algorithm is linear in the cyclic max-distance between surviving darts.

5. Conclusion

We have defined in this article the notion of contraction kernel which extends the one of decimation parameter [5] and allows to perform several contractions simultaneously. We have shown that any sequence of successive contraction kernels is equivalent to the application of one equivalent kernel. In the same way one contraction kernel with a high decimation rate may decomposed into several ones with smaller decimation rate. This notion of equivalent kernels provides a better flexibility in the design of kernels which allows to better fit the pyramid structure to the data.

This article also provides the main theoretical results required to compute the reduced combinatorial map defined by a contraction kernel. This results are then used to design two sequential and parallel algorithms computing the contracted combinatorial map.

References

[1] M. Bister, J. Cornelis, and A. Rosenfeld, A critical view of pyramid segmentation algorithms, *Pattern Recognit Letter.*, 11(9), pp. 605–617, Sept. 1990.

[2] J. P. Braquelaire, P. Desbarats, J.-P. Domenger, and C. Wüthrich, A topological structuring for aggregates of 3d discrete objects, In W. G. Kropatsch and J.-M. Jolion (eds.), 2^{nd} *IAPR-TC-15 Workshop on Graph-based Representations*, volume 126, pp. 145–154, Haindorf, Austria, May 1999. Österreichische Computer Gesellschaft.

[3] L. Brun and W. Kropatsch, Pyramids with combinatorial maps, Technical Report PRIP-TR-057, PRIP, TU Wien, 1999.

[4] L. Brun and W. G. Kropatsch, Dual contraction of combinatorial maps, In W. G. Kropatsch and J.-M. Jolion (eds.), 2^{nd} *IAPR-TC-15 Workshop on Graph-based Representations*, volume 126, pp. 145–154, Haindorf, Austria, May 1999. Österreichische Computer Gesellschaft.

[5] L. Brun and W. G. Kropatsch, Irregular pyramids with combinatorial maps, In A. Amin, F. J. Ferri, P. Pudil, and F. J. Iñesta (eds.), *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR'2000 and SPR'2000*, volume Vol. 1451 of *Lecture Notes in Computer Science*, pp. 256–265, Alicante, Spain, August 2000. Springer, Berlin Heidelberg, New York.

[6] P. Burt, T.-H. Hong, and A. Rosenfeld, Segmentation and estimation of image region properties through cooperative hierarchial computation, *IEEE Transactions on Sustems, Man and Cybernetics*, 11(12), pp. 802–809, December 1981.

[7] A. J. Gareth and D. Singerman, Theory of maps on orientable surfaces, volume 3, pp. 273–307. London Mathematical Society, 1978.

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

[9] W. G. Kropatsch and S. BenYacoub, Universal Segmentation with P*IRR*amids, In A. Pinz (ed.), *Pattern Recognition 1996, Proc. of 20th ÖAGM Workshop*, pp. 171–182. OCG-Schriftenreihe, Österr. Arbeitsgemeinschaft für Mustererkennung, R. Oldenburg, 1996, Band 90.

[10] W. G. Kropatsch and H. Macho, Finding the structure of connected components using dual irregular pyramids, In *Cinquième Colloque DGCI*, pp. 147–158. LLAIC1, Université d'Auvergne, ISBN 2-87663-040-0, September 1995.

[11] P. Lienhardt, Subdivisions of n-dimensional spaces and n-dimensional generalized maps, In *Annual ACM Symposium on Computational Geometry, all*, volume 5, 1989.

[12] A. Montanvert, P. Meer, and A. Rosenfeld, Hierarchical image analysis using irregular tessellations, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4), pp. 307–316, APRIL 1991.

[13] P. F. M. Nacken, Top-down image analysis by cost minimisation in hierarchical graph structures, Technical Report BS-R9416, CWI, Amsterdam, 1994.

[14] A. Rosenfeld (ed.), *Multiresolution Image Processing and Analysis*, Springer Verlag, Berlin, 1984.