Technical Report

Pattern Recognition and Image Processing Group
Institute of Visual Computing and Human-Centered Technology
TU Wien
Favoritenstrasse 9-11/193-03
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801 - 18661
Fax: +43 (1) 58801 - 18697
E-mail: stefan.fiedler@alumni.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-149 April 8, 2021

Parallel Combinatorial Pyramid Generation with Maximal Independent Directed Edge Sets

Stefan Fiedler

Abstract

The combinatorial pyramid is a conceptually simple and efficient representation of plane graphs that is well suited to model irregular graph pyramids. In this report, we show, through theoretical considerations and practical experiments, that maximal independent directed edge sets can be extended to select contraction and removal kernels consisting of independent darts in a combinatorial pyramid.
In this way, the adaptation facilitates the parallel processing of combinatorial pyramid generation. Additionally, with the use of the edge preselection mechanism inherent to this method, it is possible to steer the generation process towards any desired set of survivors. While this report only uses a basic preselection to demonstrate its general effectiveness, the approach can be easily adapted by computing different survivor selection criteria.

1 Introduction

Image pyramids are widely used in different image processing tasks, such as image segmentation and connected-component labelling. Over the last four decades, various representations of image pyramids and generation methods have been devised. Of these, combinatorial pyramids are a particularly simple yet versatile representation.

In this report, we present an efficient method for the parallel construction of combinatorial pyramids, based on the maximal independent directed edge set (MIDES) algorithm. While the development of the method in this report has been motivated by the generation of region adjacency graphs from labelled images, it can be easily extended to generate other types of image pyramids.

The main contribution of this paper includes the adaptation of the MIDES algorithm for parallel contraction and removal operations in combinatorial pyramids. An evaluation of the image pyramid generation and a discussion of the results are also provided.

Section 2 gives a short overview of the theoretical background of image pyramids, representations, and reduction methods. Section 3 describes in detail the considerations and adaptations for the presented method. The implementation, its experimental evaluation and a discussion of the results are covered in section 4. The last section provides a conclusion and considerations for future research.

2 Theoretical Background

2.1 Region Adjacency Graphs

Regions are maximally connected components in an image, i.e. adjacent pixels that belong to the same group, according to a selected similarity metric. In the case of labelled images, adjacent pixels are in the same region if their labels are equal.

Region adjacency graphs (RAGs) represent topological relations between regions. In the simplest case they encode common boundaries between regions. More refined relations can be devised, for example if a region is contained in or encloses another region, or if multiple disjoint boundaries between any two regions exist. It can also be helpful to model the region

outside of the image explicitly, so that every region on the image border is adjacent to it.

The type of relations that a RAG can represent depends on its graph model (Brun and Kropatsch [3]). In general, the graph model should be easy to update and provide detailed information about relations between regions. RAGs can be computed in a bottom-up fashion with image pyramids, as described in the following sections.

2.2 Image Pyramids

Image pyramids consist of a series of increasingly smaller images produced from a base image. As such, they are a multi-resolution representation of the base image, and useful in a variety of tasks, for example in image segmentation and object recognition. The motivation for image pyramids is founded in computational efficiency as well as biological plausibility (Kropatsch [5]).

Useful properties include noise reduction, processing of local and global features in the same frame, the detection of global features with local processes, and efficient computation [3].

Each level in a pyramid consist of cells, and an adjacency relation defining their neighbourhood, with the base image at the lowest level. Each higher level is built from the level below using a reduction function. This function takes several cells from the lower level and reduces them into a single cell at the next level.

In this way, the size of each level decreases until only one cell is left at the top, or the process is stopped. The ratio of consecutive level sizes is also called the reduction or decimation factor. With a constant reduction factor, it follows that the number of levels, or height h of the pyramid, is bounded by $O(log(|I|))$, with $|I|$ being the image size in pixels.

At the base level, adjacency is defined by the connectedness of pixels. For the purpose of graph representations of images, a 4-neighbourhood, consisting of neighbours in horizontal and vertical directions, is advantageous. As opposed to 8-neighbourhood, it allows for a planar embedding of the resulting graph. Other options include triangular and hexagonal neighbourhoods.

Types of image pyramids mostly differ in their reduction functions. For regular image pyramids, the reduction function is the same for all cells in each level. Irregular pyramids use a reduction function that depends on the image data.

2.2.1 Regular Pyramids

Regular pyramids have a fixed reduction function, or window, and thus a constant decimation factor.

The classification of regular pyramids is usually given as $N \times M/q$ with $N \times M$ being the reduction window, and q the decimation parameter. In general, the reduction windows may be non-overlapping with holes, non-overlapping without holes, or overlapping, depending on the parameter values [3].

While regular pyramids are simple and efficient, they suffer from shift-variance and a limited number of encoded regions at each level (Brun and Kropatsch [2], [3]). Due to the regular pattern of the reduction window, it is not possible to adapt the reduction process to the image data.

2.2.2 Irregular Pyramids

In irregular pyramids, the reduction window is not fixed but depends on image data or other criteria. On each level, a set of surviving cells is selected and each non-surviving cell is assigned to one survivor. The non-surviving cells must be connected directly or indirectly to their designated survivors. The neighbourhood of survivors is defined by the adjacency relationships of cells in their reduction window.

The partitioning of cells, including the selection of survivors, is also called a contraction kernel. The receptive field of a cell at a given level is the set of all cells at the base level that have been reduced to this cell.

With this process the regularity of the neighbourhood structure in the base image cannot be maintained at higher levels [5]. For this reason, irregular pyramids are based on graphs rather than arrays. Graphs can adapt to image properties and retain most useful properties of regular pyramids. The decimation factor is not constant however, and is usually defined as the mean ratio of consecutive level sizes V_l/V_{l+1}, with V_l being the number of vertices in the graph at pyramid level l.

In particular, irregular pyramids should keep two properties that regular pyramids have [3]:

(i) the computation of each vertex should be independent from others, allowing for parallel computation;

(ii) the logarithmic height of the pyramid, so that parallel computation is still $O(log(|I|))$.

The main questions for building irregular pyramids therefore are [3]:

(i) how to reduce each level to the next higher level efficiently (construction scheme)?

(ii) which properties can each level represent (graph encoding)?

The construction scheme affects the size of each level, i.e. the vertical property of the pyramid; the graph encoding influences the properties that a pyramid can represent. Both define an irregular pyramid.

Irregular pyramids allow to adapt the reduction window to the image data. Since each vertex can have an arbitrary number of neighbours they can retain the adjacency relationships between regions [3].

2.3 Graph Representations

Graphs are a common representation of irregular pyramids; the choice of graph model determines the topological and geometrical image properties, i.e. the horizontal pyramid resolution. The model choice is usually independent of the construction scheme: different models can be combined with various construction methods [3]. Besides higher horizontal resolution and adaptability, graphs have the advantage of shift-invariance and rotation-invariance (Macho and Kropatsch [8]).

The construction of graphs from pixel images follows a simple process: each vertex represents one pixel, and neighbours are connected by edges. This graph is called the neighbourhood graph. A problem occurs with 8-neighbourhood, as the graph becomes non-planar. This is especially important for the dual graph and combinatorial map representations, as described below.

Graphs can also be used to represent the result of a segmentation process: each vertex represents a set of connected pixels (a region) and edges describe neighbourhood relationships of vertices. In this manner, irregular pyramids based on graphs can be used to generate and represent region adjacency graphs for images.

Models differ in the representation of relations between regions. In the simplest case only adjacency itself is stored. More refined models can store each segment of a disjoint border separately, and also represent the inclusion of regions within others. This information can be useful to guide the segmentation process at a higher level.

2.3.1 Simple Graphs

A simple graph $G(V, E)$ consists of sets of vertices V and edges E, such that there are no self-loops, i.e. edges $e_{u,v}$ where $u = v$, and no multiple edges between pairs of vertices, i.e. edges $e_{u_1,v_1} \neq e_{u_2,v_2}$ where $u_1 = u_2$ and $v_1 = v_2$.

A graph is planar if an embedding in the plane without edge-crossings exists. Planar simple graphs are a common model for region adjacency graphs. Simple graphs cannot distinguish between different topological configurations [8], in particular between inclusion and multiple adjacency relationships of regions [2].

2.3.2 Dual Graphs

Extending simple graphs to represent disjoint borders can lead to multiple edges between any two vertices. However, following the contraction of edges as described below, this non-simple graph alone cannot distinguish between disjoint borders and ones that are now connected.

The dual graph model uses a graph G and its dual $\overline{G}$ to uniquely represent multiple borders between regions. In the dual graph, or face graph, each vertex represents one face in the primal (or neighbourhood) graph. Edges in the dual graph directly correspond to edges in the primal graph, i.e. faces separated by a common edge in G are vertices connected by an edge in $\overline{G}$.

Important properties of the dual graph include [3]:

(i) The original graph can be reconstructed from its dual graph: $\overline{\overline{G}} = G$.

(ii) For each subset of edges $N \subseteq E$ in G, there is a corresponding set $\overline{N}$ in $\overline{G}$.

(iii) The contraction of a forest $N \subset E$ in G is equivalent to the removal of $\overline{N}$ in $\overline{G}$. Since N is a forest (containing no cycles), $\overline{N}$ is not a cut-set: both G and $\overline{G}$ remain connected.

(iv) Likewise, a contraction of $\overline{N}$ in $\overline{G}$ is equal to the removal of N in G.

In other words, the basic operations on graphs are dual: the contraction of an edge in G results in the removal of an edge in $\overline{G}$ and vice versa. The contraction of edges can produce redundant faces in a graph, i.e. self-loops and redundant edges. These faces are characterized as vertices of degree one or two in the dual graph. They need to be removed in an additional step referred to as dual graph contraction (Willersin and Kropatsch [10]).

It should be noted that multiple edges may belong to faces with a higher degree than two, i.e. faces containing other edges. These edges are not

redundant, as they represent topological information, and should therefore not be removed.

The RAG resulting from dual graph contraction can contain more edges than a simple graph: regions contained within another region are enclosed by an additional self-loop, and disjoint borders are represented by multiple edges [8]. Its main disadvantage is the requirement to maintain and update two graphs in parallel.

The dual graph model cannot unambiguously represent a region enclosed in another one on a local level [3]. To determine the contains/inside relation between regions presumably requires global calculations. However to the best of our knowledge this has not been confirmed or disproved so far.

2.3.3 Combinatorial Maps

Combinatorial maps are a generalized framework that can encode any subdivision of multidimensional, orientable topological spaces, with or without boundaries. In the context of this report, combinatorial maps refer only to their 2D instances.

A combinatorial map is similar to a graph but explicitly stores the orientation of edges around each vertex. It can be defined as a model $G = (D, \sigma, \alpha)$, consisting of a set of darts D, and two functions σ and α. Each edge is represented by two opposing darts. The involution α relates opposing darts to each other; since α is an involution, $\alpha^2(d) = d$ for all $d \in D$. The darts originating from each vertex are accessible with the permutation σ in a counter-clockwise fashion. The local orientation of edges is therefore encoded in σ.

Given a dart d, an orbit of a permutation π, denoted π^*, is the series π^i defined by successive application of π to d. In particular, the series $\alpha^*(d)$ defines the edge that d is a part of. The series $\sigma^*(d)$ is called the orbit of d, and corresponds to the vertex that d belongs to.

The dual of a combinatorial map is defined by $\overline{D} = (D, \phi, \alpha)$, with $\phi = \sigma \circ \alpha$. As such, ϕ returns the darts around the face to the right of d, in counter-clockwise fashion. Alternatively it can be seen as the permutation of darts originating from the dual vertex, in clockwise order. The dual operation on combinatorial maps is idempotent (Brun and Kropatsch [1]).

In the basic definition, orbits or vertices are not modelled explicitly. In practical implementations however, additional data structures can be added for efficiency: when dealing with operations on combinatorial maps, the inverse permutation σ^{-1}, a mapping of darts to their respective orbits, and a

mapping of orbits to the set of connected darts are especially helpful.

Combinatorial maps can encode the relation between a region enclosed in another region. In combinatorial pyramids, this information can be retrieved with local calculus. In this regard, they are equivalent to dual graphs [1]. As a consequence, their structures cannot differentiate locally between contains and inside relationships either [3].

Their main advantages are the explicit encoding of the orientation of the plane, which is not directly available in dual graph representations, and the implicit encoding of the dual graph. Combinatorial maps can also be easily extended to higher dimensions [2].

When using combinatorial maps to generate image pyramids, the result can be interpreted as a planar embedding of a RAG. As such, it has a particular orientation and may contain multiple edges and self-loops.

2.4 Operations on Graphs and Combinatorial Maps

For the generation of irregular pyramids, two basic operations on graphs are needed: edge contraction and edge removal. The former contracts an edge connecting two vertices, and the two vertices are joined into one. All edges that were incident to the joined vertices will be incident to the resulting vertex after the operation. The latter removes an edge from the graph, without changing the number of vertices or affecting the incidence relationships of other edges.

Contractions do not disconnect the graph, and thus preserve connectivity. They are not defined for self-loops. However they can introduce self-loops and redundant edges. Empty self-loops and redundant edges can be removed before another contraction is performed.

Both operations are commutative: multiple operations can be applied in any order to obtain the same result [1]. If a graph model includes the outside vertex, contractions must not include edges incident to it [2].

The vertex degree is unbounded under edge contractions. However, the degree of faces does not increase (see [5] for a proof). This implies that for dual graph representations, the vertex degree in one graph is bounded for each operation.

In the dual graph representation, contraction and removal operations are dual: a contraction in the primal graph corresponds to a removal in the dual graph, and vice versa. This also applies to combinatorial maps, as they have an implicit dual representation [2].

2.4.1 Parallel Operations

Parallel contraction on graphs is defined by decimation parameters (V_S, E_{SN}), consisting of a set of vertices (survivors) $V_S \subset V$, and a set of edges (contractions) $E_{SN} \subseteq E$. The set E_{SN} must connect each non-surviving vertex to exactly one survivor. Thus, the decimation parameter is a forest, with one component for each survivor.

This operation preserves the structure of the survivors, and produces a minimal and unique result (see [5] for a formal definition and proof). The same considerations apply to combinatorial maps [1].

Dual edge contraction and dual face contraction (of faces with a degree less than 3) are also structure preserving [5]. Removal operations can therefore be performed in parallel similar to contractions. The decimation parameters for each operation are also called contraction and removal kernels, respectively.

On a combinatorial map, each contraction and removal operation involves a set of dependent darts, which cannot be modified in parallel due to the way the model is represented. For each dart d, the dependent darts are $\alpha(d)$, $\sigma(d)$, $\sigma^{-1}(d)$, $\sigma(\alpha(d))$, and $\sigma^{-1}(\alpha(d))$. In order to implement parallel operations on combinatorial maps, these dependencies have to be taken into account for kernel selection.

2.5 Generation of Irregular Pyramids

With the notion of parallel edge contraction and removal, it is possible to define the generation of irregular pyramids as follows:

1. select a contraction kernel and contract all edges,
2. select a removal kernel and simplify the graph,
3. repeat 2. until there are no more edges to remove,
4. repeat 1. to 3. until there are no more edges to contract, or the process is stopped.

Kernel selection methods define how to select edges and surviving vertices, as described in the following section.

2.6 Kernel Selection Methods

Methods for kernel selection differ in how they select edges and survivors. In particular, the ability to adapt survivor selection to image data depends on

the chosen method. The number of contractions at each level, and thus the decimation factor, is also influenced by the kernel selection. The notion of maximal independent sets ensures an almost constant factor and is used by several construction methods.

2.6.1 Maximal Independent Sets

For irregular pyramids, the survivors for reduction can be described as a maximal independent set [3]: Consider a set X with a neighbourhood function $N\colon X \to P(X)$, where $P(X)$ is the power set of X. An independent set I is a subset of X, such that no two elements of I are neighbours. The set I is maximal if no element can be added from X without violating the neighbourhood condition. It follows that if I is maximal, each vertex in X is either in I or has a neighbour in I. I is not uniquely defined. It is maximum if it has the largest possible size over all independent sets of X.

2.6.2 Maximal Independent Vertex Sets

Using the idea of a maximal independent set on the vertices of a graph gives a maximal independent vertex set (MIVS). Meer [9] implements an iterative stochastic process to determine such a set on a graph based on local properties only. Each vertex is assigned a positive random number and each vertex with the highest value in its neighbourhood belongs to the independent set.

Initially, this set may however not be maximal. The iterative process therefore uses two additional boolean variables per vertex, p_i and q_i, to determine if a vertex should be a surviving vertex. If p_i is true, vertex v_i is in the independent set, and q_i is true if it is a candidate. The set and p_i, q_i are then updated until the set is maximal [3].

Jolion [4] adapts the process using an interest operator. Surviving vertices are defined at local maxima (or minima) of the operator, thus providing their location in homogeneous regions. For greyscale images, the vertex with the smallest variance in its receptive field is selected as a survivor. At the base level the 3×3 neighbourhood is used instead of the receptive field of a vertex. Three variables p, q, and v are used for the decimation process. Each vertex where $p_{i,j,l}$ is true is a surviving vertex at level l. Initially $p_{i,j,l}$ are false for all (i, j, l). The value $p_{i,j,l}$ is true only if the respective cell has the lowest value $v_{i,j,h}$ of all its neighbours.

Experiments showed that with a variable v that is not random but depends on image data, the number of iterations is higher than with the stochastic pyramid, especially at the first level. The decimation ratio was also higher for the first level, but the values were similar and both resulted in the same height.

2.6.3 Data Driven Decimation Process

The Data Driven Decimation Process (D3P) works similar to the stochastic pyramid of Meer [9] but keeps p_i and q_i as global variables for all levels. Instead of creating an MIVS on each level, the D3P carries survivor candidates to the next level. In combination with the interest operator [4], where survivors are located at local maxima, this has the advantage that vertices around local maxima are reduced early and more complex configurations can be resolved at higher levels.

2.6.4 Maximal Independent Edge Sets

Maximal independent edge sets (MIES) and maximal independent directed edge sets (MIDES) were introduced by Kropatsch et al. [6]. With MIVS, the probability that a vertex is in the independent set depends on its neighbours, and thus the size of its neighbourhood [3]. During the decimation process, the vertex degrees tend to increase with each level. This in turn affects the height of the pyramid and the number of iterations required to build each level. To correct this phenomenon, MIES selects a forest F in G, where:

(i) the forest F partitions the graph;
(ii) each tree in F has two or more vertices.

The latter condition ensures a decimation factor of two or more. From each tree in F one survivor is selected. The first step of constructing F uses a maximal independent edge set (MIES), which is defined as a MIVS on the edge graph of G. The edge graph contains a vertex for each edge of G, and vertices in the edge graph are connected if the corresponding edges in G are incident to the same vertex.

In this way, the MIES induces a maximal matching on the initial graph vertices. A matching on a graph is a subset of its edges so that no two edges are incident to the same vertex. It is maximal if no edge can be added without violating the matching condition. A maximum matching contains the largest possible number of edges for a graph. Finding a maximum matching is an

NP-hard problem. The matching is perfect (and also maximum) if it is incident to all vertices. Vertices that are part of the matching are called saturated, and all others are unsaturated.

A maximal matching is also a forest in G, but may contain isolated vertices. In that case the isolated vertices can be connected to one of their neighbours. This additional step creates trees of diameter at least two, which can be reduced to two trees of diameter one by removing all edges between vertices of degree two or more. The resulting forest may no longer be a matching but covers all vertices.

2.6.5 Maximal Independent Directed Edge Sets

MIES cannot easily include constraints for surviving vertices, which might be useful for certain applications, e.g. line drawings [3]. Maximal independent directed edge sets (MIDES) extend the MIES construction by using directed edges. The survivor in each tree of the forest F must then be the target of all directed edges in its tree.

Directed edges in the graph thus encode which vertices in a neighbourhood should be surviving vertices. For an undirected graph it is possible to create a pair of directed edges for each original edge and select the ones intended for the construction. These edges are called preselected edges.

The construction builds a MIES on the preselection using the definition of a directed edge neighbourhood. Since no two neighbouring edges can be in the independent set, the neighbourhood controls which configurations are allowed in the set.

When an edge $e_{u,v}$ from u to v is selected, only edges pointing to v should be added to the tree consisting of u and v. The directed neighbourhood for $e_{u,v}$ is therefore defined as all edges emanating from v, and all edges to or from u. There may be some isolated vertices after constructing the MIDES, which are added as survivors as well.

3 Parallel Combinatorial Pyramid Generation

Based on the considerations in the previous sections, it is possible to devise an adapted MIDES algorithm suitable for the parallel generation of combinatorial pyramids. The crucial observation from the theoretical background is the fact that dependent darts in a combinatorial map cannot be modified

in parallel under edge contraction and removal operations. Indirectly, this also determines the choice of the kernel selection method.

All kernel selection methods presented in section 2.6 only depend on local properties and can therefore be computed in parallel. With MIVS first introduced in stochastic pyramids, the survivor selection and the assignment of non-survivors solely depend on a stochastic process. Hence there is no guarantee that the edges in all trees of a kernel are independent under contraction or removal. For MIES, the maximal matching on vertices induces a set of independent edges. However, adding non-saturated vertices to random survivors can also create a kernel whose edges are not independent.

The MIDES method restricts the selection of edges for the contraction kernel using the notion of a directed edge neighbourhood. By adapting the definition of the neighbourhood, it is possible to only select independent sets of edges for contraction and removal operations in a combinatorial map.

In addition, MIDES has the advantage of easy survivor selection through edge preselection. Preselection also allows to adapt kernels for task-specific goals, for example to exclude edges between vertices of different regions from contraction in RAG generation. It is therefore a simple yet powerful mechanism to control the reduction process in the construction of irregular pyramids.

The general rule for combinatorial image pyramid construction then consists of the following steps:

1. generate the base level of the combinatorial pyramid from an image,
2. preselect edges and select a contraction kernel with MIDES,
3. apply the contraction kernel to generate the next level,
4. preselect edges for removal and select a removal kernel with MIDES,
5. apply the removal kernel to simplify the current level,
6. repeat 4. to 5. until there are no more edges to remove,
7. repeat 2. to 6. until there are no more edges to contract, or the process is stopped.

Generating a combinatorial map from an image is a straightforward process, where a pair of darts is inserted for each neighbourhood relation between pixels. The calculation of permutations and other mappings can be simplified by using a numbering scheme for darts and vertices, e.g. by assigning consecutive numbers from top to bottom, and left to right.

For kernel selection, each dart is considered a directed edge, and a subset of directed edges is preselected as a set of candidates for inclusion in the maximal independent set. As contractions and removals operate on different

types of edges, the preselection rules also differ between both operations.

3.1 Edge Preselection

In general, a contraction kernel should not contain any self-loops. Self-loops should thus not be considered for contraction. Non-empty self-loops contain topological information and should not be removed.

Contracting one of a set of multiple edges between a pair of vertices turns all other edges in the set into self-loops. The parallel contraction of two or more edges between the same pair of vertices is therefore undefined and should not be considered either.

Redundant edges should be removed after contraction. Multiple edges can be parts of faces with a higher degree than two (i.e. enclosing other vertices), in which case they are part of the topology and must be retained.

The edge preselection for contraction thus excludes all self-loops from the kernel selection. It is not necessary to remove multiple edges from the preselection, as the directed edge neighbourhood prevents the selection of more than one edge between each pair of vertices.

If the outside region of an image is modelled as its own vertex, all edges from this vertex must be removed from the preselection. In the case of RAG generation, all edges between vertices with different labels are also excluded from contraction, to prevent merging pixels from different regions.

During simplification, all empty self-loops and faces of degree two should be removed. Thus all edges of these types are preselected for removal kernel selection. For RAG generation, edges are removed regardless of the labels of their vertices.

Multiple edges between the same pair of vertices can be removed in parallel, as long as they are not dependent on each other. This is handled by using a different definition of the directed edge neighbourhood for the kernel selection, which is described in the following section.

3.2 Directed Edge Neighbourhood

As in the original MIDES method, survivors are at the root of trees of diameter two. The directed edge neighbourhood determines which edges can be selected. A directed edge can only be included in the kernel if none of its neighbouring edges is selected.

For contraction kernels, all edges must point from a non-survivor to a surviving vertex. Therefore, a directed edge $e_{u,v}$ from a non-survivor u to a survivor v may only be included in the independent set if the set does not already contain any incoming edges to u, or outgoing edges from u or v. The former would imply that u is a survivor, and the latter that u or v is a non-survivor of another vertex. The neighbourhood of $e_{u,v}$ thus contains all edges pointing towards u, or emanating from u or v.

In addition, the neighbourhood must contain all directed edges corresponding to dependent darts. It is necessary to not only include dependent darts but also their involutions, as selecting an edge corresponding to $\alpha(d)$ would affect d as well, and thus violate the condition for parallel operations. Since edges emanating from u and v are already included in the neighbourhood, it is sufficient to only add edges corresponding to the involutions of dependent darts.

For removal kernels, there are no survivors, and edges can be selected as long as they are not dependent. The neighbourhood for each edge thus contains only the directed edges corresponding to the dependent darts and their involutions.

3.3 Survivor Selection

With MIDES, vertices can be easily designated as survivors by removing all their outgoing edges from the preselection. This decision can be different for each level of the pyramid. The preselection can also be based on pairs of adjacent vertices, in which case it is not necessarily transitive. If no decision is made, the selection is purely random. Thus the question remains how to decide if a vertex should be a survivor candidate.

A simple solution would be to assign a value to each vertex, and for each pair of vertices select the one with the higher value as a designated survivor. This is akin to the adaptive pyramid [4], which defines surviving vertices at local maxima of an interest operator, based on the grey level variance of receptive fields.

In the case of RAG generation, contractions only occur between pixels with the same label. Therefore a different measurement, also called a tie-break value, is needed. Regular distance transforms on individual regions are an option, but they tend to produce the same values for many adjacent pixels.

Instead of using a single distance transform it is possible to combine an

elliptical and a hyperbolic distance transform. Since these transforms are orthogonal to each other, no pair of vertices will have identical values in both transforms. For each two vertices it is therefore possible to use one transform to select a survivor, and if the respective values are equal use the other transform. Survivors in the pyramid hierarchy simply inherit their distance values from the previous level.

Both transforms need to be computed only once, and can then be shifted to the centre of each region. The centre of a region can be defined as its centroid, but for concave shapes or shapes with holes it may lie outside the region. As another option to define a centre point, the eccentricity transform (Kropatsch et al. [7]) could be used, which results in a single maximum that always lies inside a region.

3.4 Contraction and Removal Operations

With the neighbourhood defined as above, the MIDES algorithm can be applied to the preselected edges. The result is a kernel consisting of a set of independent edges. For contraction kernels, survivors and non-survivors are easily identified as vertices with outgoing and incoming edges in the kernel, respectively. As in the original method, vertices not covered by a kernel are isolated survivors.

The contraction and removal kernels can be applied in the usual manner. During contraction, it is necessary to make a distinction between pending and regular darts, as the contraction operation is defined differently for both types. Likewise, the removal operations for empty self-loops and redundant edges are different. As already mentioned, the simplification may have to be repeated several times for each level of the pyramid, until all faces of degree one or two are removed.

4 Implementation

The implementation of the proposed solution is based on an existing framework for combinatorial pyramids and written in Matlab M-script. The data structure storing the combinatorial pyramid contains the following elements relevant to this task:

`permutation` is a vector representing the permutation σ.

`inv_permutation` gives the inverse permutation σ^{-1}.

`dart_diff` for each dart, stores the difference between the attribute values (labels) of its vertex and the vertex of its involution. If zero, both vertices belong to the same region.

`orbitMat` is a sparse matrix storing the indices of all darts belonging to each vertex (σ-orbit).

`represent_ver` stores the receptive field of each active vertex.

`attribLUT` maps darts to their respective orbits.

`tie_break_value_1` and `tie_break_value_2` store the tie-break values for survivor selection.

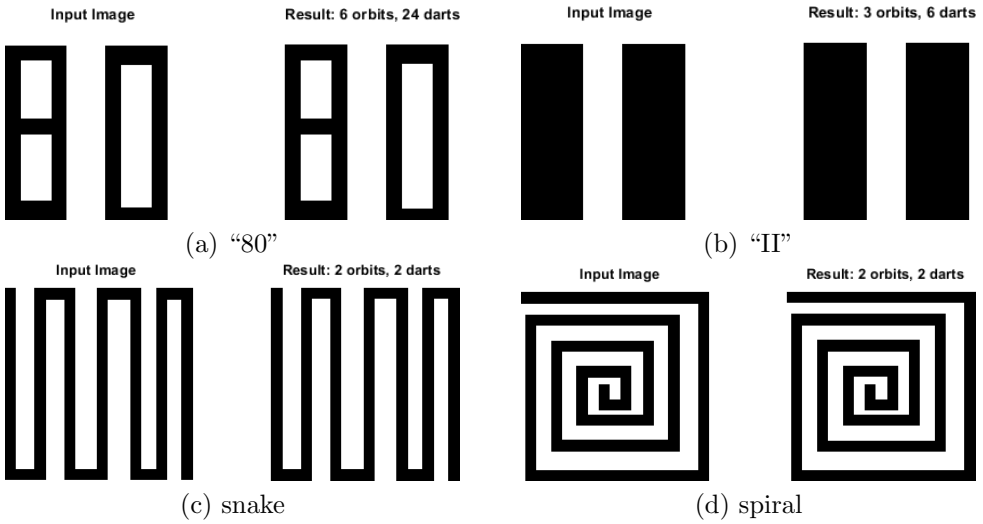
In the data structure, all darts, edges, and vertices are represented by positive integers. By convention, darts belonging to the same edge have consecutive numbers, i.e. edge e_i consists of darts d_{2i-1} and d_{2i}. The vector of involutions α is not stored in the data structure but generated as needed using this relation.

Contraction and removal kernels are represented as vectors containing indices of directed edges. Surviving vertices are not computed or stored explicitly, as the direction of edges in the contraction kernels is sufficient to determine which vertices are merged with others. Isolated survivors do not require any updates during contraction and hence are not computed either. To control the preselection of darts for the adapted MIDES algorithm, two vectors containing tie-break values for each vertex are added, as explained in section 3.3.

In theory, the mapping from darts to vertices can be accomplished by searching the orbit matrix. This proved to be a major performance bottleneck. For this reason, an additional `attribLUT` element is used. The orbit matrix is still needed to map orbits to the set of incident darts, in particular for defining edge neighbourhoods.

4.1 Experiments

To ensure the correctness of the implementation, several function tests were performed on each part of the program, including the MIVS and MIDES algorithms, selection of the contraction and removal kernels, updates to permutations and the orbit matrix, and contraction and removal operations. Kernel selection was also tested with different tie-break values to ensure the preselection of edges is working as expected.



(a) "80" (b) "II"

(c) snake (d) spiral

Figure 1: Labeled images for basic function tests and performance measurements. Original images are on the left, and reconstructed images on the right. Note that in image (c) the white background is a single connected region.

Higher-level tests were performed with several test images consisting of two or more labelled regions. The test images are 55×55 pixels in size. In each case, the implementation is able to produce the correct region adjacency graph, and the original image can be reconstructed using the receptive fields associated with each vertex in the result. Fig. 1 shows the original image and the reconstruction for each test case.

Basic performance measurements using the four test images in Fig. 1 are summarized in Table 1. All tests were performed on an AMD Athlon II X2 250 processor with 3.00 GHz and 4 GB RAM, and Matlab 2020b on Windows 10 64-bit.

The RAG generation was repeated 20 times for each image. For each measurement, the mean value μ and standard deviation σ over all samples are given. The statistics include the height of the combinatorial pyramid, the decimation ratio, the time to generate each pyramid, and the number of iterations during each simplification phase. During simplification, an iteration only counts if the selected removal kernel is not empty.

The decimation ratio tends to be higher in lower levels of the pyramid. To capture this phenomenon, two values are recorded. The first value gives

	"80"	**"II"**	**snake**	**spiral**
Pyramid Height, μ	12.150	12.250	12.350	12.100
Pyramid Height, σ	0.745	0.786	0.745	0.447
Decimation Ratio, μ	1.789	1.874	1.925	1.947
Decimation Ratio, σ	0.361	0.288	0.258	0.214
Decimation Ratio LH, μ	2.059	2.081	2.041	2.033
Decimation Ratio LH, σ	0.078	0.080	0.059	0.063
Generation Time in s, μ	0.716	0.716	0.640	0.607
Generation Time in s, σ	0.185	0.053	0.090	0.037
Removal Iterations, μ	2.283	2.373	2.489	2.523
Removal Iterations, σ	0.878	0.825	0.737	0.671
Removal Iterations, max.	5	4	4	4

Table 1: Performance measurements of region adjacency graph generation for the test images from Fig. 1. The measurements were performed with zero tie-break values.

the mean decimation ratio over all levels, while the second value gives the mean value over the lower half (LH) of the pyramid levels (i.e. levels 1 to $\lfloor h \rfloor$). No timings for parts of the RAG generation were gathered for these test images as Matlab cannot accurately measure times less than 1/10 of a second with the method used.

The effect of the preselection mechanism on RAG generation was measured using random and directional tie-break values. Tables 2 and 3 show the respective results. With random tie-break values, each value is taken from a uniform distribution in the range of $[0; 1]$. For directional values, vertices that are to the right or below another vertex in the original image take precedence during survivor selection.

For additional performance measurements, three pictures were used at different scales to generate a combinatorial pyramid. Each image was initially converted to greyscale, processed with a median filter of size one, and posterized to eight different greyscale levels. The images, shown in Fig. 2, were tested at the original size, at one half, and at one quarter of the size in pixels. For each image, 20 iterations of RAG generation were performed, with zero tie-break values. The results are summarized in Table 4.

Measurements provide the same values as in the tests above, with the addition of the pixel size, the number of orbits and darts in the result, and

	“80”	**“II”**	**snake**	**spiral**
Pyramid Height, μ	12.500	12.350	12.550	12.450
Pyramid Height, σ	0.513	0.587	0.686	0.605
Decimation Ratio, μ	1.750	1.856	1.899	1.908
Decimation Ratio, σ	0.315	0.236	0.223	0.217
Decimation Ratio LH, μ	1.989	2.017	1.969	1.958
Decimation Ratio LH, σ	0.075	0.070	0.045	0.054
Generation Time in s, μ	0.436	0.447	0.429	0.424
Generation Time in s, σ	0.015	0.024	0.022	0.021
Removal Iterations, μ	2.335	2.471	2.558	2.498
Removal Iterations, σ	0.834	0.766	0.695	0.653
Removal Iterations, max.	4	5	4	4

Table 2: The same performance measurements as in Table 1, with random tie-break values.

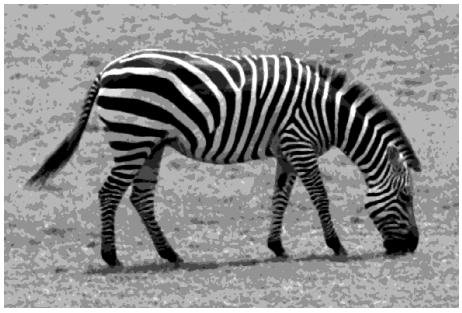
	“80”	**“II”**	**snake**	**spiral**
Pyramid Height, μ	12.900	12.750	13.050	13.100
Pyramid Height, σ	0.718	0.444	0.224	0.447
Decimation Ratio, μ	1.715	1.817	1.845	1.838
Decimation Ratio, σ	0.292	0.224	0.189	0.156
Decimation Ratio LH, μ	1.890	1.917	1.873	1.841
Decimation Ratio LH, σ	0.054	0.063	0.045	0.041
Generation Time in s, μ	0.472	0.471	0.453	0.453
Generation Time in s, σ	0.030	0.023	0.025	0.020
Removal Iterations, μ	2.244	2.400	2.407	2.384
Removal Iterations, σ	0.927	0.838	0.759	0.755
Removal Iterations, max.	5	4	4	4

Table 3: The same performance measurements as in Table 1, with directional tie-break values.

timings for specific parts of the generation process. These include the selection of the contraction and removal kernels, and the application of these kernels in the contraction and removal operations.

(a) 189080

(b) 241048



(c) zebra

Figure 2: Posterized greyscale images for the performance measurements in Table 4.

4.2 Discussion

The results for the small test images show the average pyramid height is close to the theoretical optimum bounded by $log_2(55 \cdot 55) \approx 11.56$, with a small standard deviation of less than one level. Without preselection, the decimation ratio is highest, and slightly above two for the lower half of the pyramid, while it tends to decrease towards the top. The variance of the

	(a) 1:1	(a) 1:2	(a) 1:4	(b) 1:1	(b) 1:2	(b) 1:4	(c) 1:1	(c) 1:2	(c) 1:4
Pixels	154401	77180	38400	154401	77180	38400	229126	114264	57135
σ-Orbits in RAG	1178	960	803	2655	2419	2018	9769	6979	4934
Darts in RAG	6128	4960	4070	13784	12450	10236	46830	34484	24440
Pyramid Height, μ	15.350	14.450	13.800	15.400	14.750	13.350	15.050	14.150	13.200
Pyramid Height, σ	0.745	0.826	0.834	0.883	0.910	0.813	0.759	0.587	0.616
Decimation Ratio, μ	1.471	1.447	1.409	1.384	1.338	1.315	1.297	1.278	1.259
Decimation Ratio, σ	0.448	0.432	0.410	0.418	0.390	0.363	0.364	0.344	0.323
Decimation Ratio LH, μ	1.893	1.862	1.780	1.753	1.674	1.615	1.594	1.567	1.520
Decimation Ratio LH, σ	0.238	0.239	0.264	0.296	0.308	0.297	0.316	0.298	0.288
Generation Time in s, μ	49.836	23.578	10.714	45.678	20.686	9.049	66.296	28.931	12.818
Generation Time in s, σ	2.376	0.917	0.367	1.518	0.749	0.309	2.059	0.852	0.421
Contraction Kernel in s, μ	1.792	0.887	0.412	1.474	0.667	0.311	2.149	0.938	0.418
Contraction Kernel in s, σ	3.380	1.599	0.732	2.804	1.250	0.548	4.079	1.726	0.740
Contraction in s, μ	1.118	0.574	0.286	1.171	0.581	0.291	1.742	0.862	0.435
Contraction in s, σ	0.120	0.049	0.023	0.098	0.050	0.022	0.142	0.071	0.034
Removal Kernel in s, μ	0.074	0.041	0.021	0.079	0.042	0.022	0.123	0.064	0.034
Removal Kernel in s, σ	0.109	0.056	0.027	0.112	0.054	0.025	0.165	0.077	0.037
Simplification in s, μ	0.436	0.226	0.107	0.425	0.208	0.106	0.674	0.328	0.163
Simplification in s, σ	0.329	0.164	0.079	0.339	0.162	0.075	0.503	0.235	0.109
Removal Iterations, μ	3.066	2.900	2.691	2.774	2.527	2.413	2.819	2.589	2.422
Removal Iterations, σ	1.185	1.100	1.097	1.196	1.134	0.954	1.072	0.984	0.892
Removal Iterations, max.	6	5	5	5	5	4	5	4	5

Table 4: Performance measurements of region adjacency graph generation for the test images (a), (b), and (c) from Fig. 2. Measurements were taken for each image at full size and scaled-down to approximately one half and one fourth its size in pixels, as indicated by 1:1, 1:2, and 1:4, respectively.

decimation ratio is also higher in the upper half. The generation time is less than a second, with a small variance. The average number of removal operations during the simplification steps lies between two and three, with the maximum number encountered during tests being six.

Comparison of Tables 1, 2, and 3 shows the influence of the preselection mechanism on the combinatorial pyramid generation. In general, with edge preselection the average decimation ratio is lower, and in consequence the pyramid height is larger. Directional tie-break values have a larger effect than random values, although it is small overall. The time to construct the pyramid is considerably smaller with preselection, presumably because the kernel selection has to operate on a smaller set of darts.

For the sample pictures from Fig. 2, the RAG generation resulted in pyramids with a larger number of orbits. The decimation ratios were considerably lower, due to the images containing a multitude of small regions. At higher levels, many small regions are already contracted, thus there are fewer edges to contract. The generation time, depending on image size, is roughly proportional to the logarithm of the number of pixels and the pyramid height.

5 Conclusion and Future Work

Experiments with different inputs confirm the correctness and efficiency of the proposed method. Furthermore, results indicate the pyramid generation is a stable stochastic process with low variability. The generated image pyramids are close to the theoretical optimum for simple inputs, and show good results even for challenging pictures. Edge preselection showed a slight negative impact on the decimation rate, but a positive effect on runtime.

There are three apparent avenues for future work following this report. While the current implementation only considers labelled images, it can be easily modified to process greyscale and color images. By defining a metric for the difference of vertex attributes and preselecting edges accordingly, the same approach could be used for image segmentation, in addition to RAG generation.

The second is the adaptation of the approach to include different preselection mechanisms. In particular, this could allow pyramid generation to adapt to the characteristics of images, for example using local interest operators.

Lastly, this new method for image pyramid generation could allow for higher parallelization through the use of GPU-based computations. A GPU-

based implementation of the combinatorial pyramid is already in the works, and it would be interesting to see this approach ported to the new framework as well.

6 Acknowledgments

I would like to express my deep gratitude to Professor Walter Kropatsch for providing me the opportunity to work on this topic, and for the valuable feedback on this report. I would like to offer my special thanks to Darshan Batavia for our insightful discussions about the parallel generation of combinatorial pyramids, and for introducing me to the Matlab framework on which the software for this report is built. In addition, I would like to thank Dr. Jiří Hladůvka for presenting the topic together with Professor Kropatsch at our initial meeting.

References

[1] L. Brun and W. Kropatsch. Dual Contraction of Combinatorial Maps. Technical Report PRIP-TR-054, PRIP, TU Wien, 1999.

[2] L. Brun and W. Kropatsch. Introduction to combinatorial pyramids. In *Digital and image geometry*, pages 108–128. Springer, 2001.

[3] L. Brun and W. Kropatsch. *Image processing and analysis with graphs: theory and practice*, chapter 1, pages 1–62. CRC Press, 2012.

[4] J.-M. Jolion and A. Montanvert. The adaptive pyramid: a framework for 2d image analysis. *CVGIP: Image Understanding*, 55(3):339–348, 1992.

[5] W. G. Kropatsch. Building Irregular Pyramids by Dual Graph Contraction. Technical Report PRIP-TR-035, PRIP, TU Wien, 1994.

[6] W. G. Kropatsch, Y. Haxhimusa, Z. Pizlo, and G. Langs. Vision pyramids that do not grow too high. *Pattern Recognition Letters*, 26(3):319–337, 2005.

[7] W. G. Kropatsch, A. Ion, Y. Haxhimusa, and T. Flanitzer. The eccentricity transform (of a digital shape). In *International Conference*

on Discrete Geometry for Computer Imagery, pages 437–448. Springer, 2006.

[8] H. Macho and W. G. Kropatsch. Finding connected components with dual irregular pyramids. In *Visual Modules, Proc. of 19th ÖAGM and 1st SDVR Workshop*, pages 313–321. R. Oldenburg, 1995.

[9] P. Meer. Stochastic image pyramids. *Computer Vision, Graphics, and Image Processing*, 45(3):269–294, 1989.

[10] D. Willersinn and V. Kropatsch. Dual graph contraction for irregular pyramids. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2-Conference B: Computer Vision & Image Processing.(Cat. No. 94CH3440-5)*, pages 251–256. IEEE, 1994.