Technical Report

Dept. for Pattern Recognition and Image Processing
Institute for Automation
Technical University of Vienna
Treitlstr. 3/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-8161
Fax: +43 (1) 569697
E-mail: wil@prip.tuwien.ac.at

PRIP-TR-28 October 4, 1994

Parallel Graph Contraction for Dual Irregular Pyramids[1]

Dieter Willersinn

Abstract

Hierarchical representation of images is a crucial building principle of a system architecture that can cope with the complexity of visual perception. This paper presents an algorithm that builds a hierarchy which is flexible enough to provide a shift and scale invariant abstract representation of image content. The hierarchy is irregular in the sense that an element of the representation may have an arbitrary number of neighbors. At the same time, the representation can be described by a bounded data structure. Main properties of the algorithm with respect to visual perception are proved.

[1]This research was supported by the Austrian Science Foundation under grants P 8785 and S 7002-MAT

Contents

1 Introduction

The goal of image analysis is "to describe the content of a digital image in order to interpret it and to take a decision" [2]. During interpretation, the content of the image is compared to some previously acquired prototypical knowledge. Depending on the application, these prototypes can be single objects, events, scenes or episodes.

Tsotsos refers to the interpretation process as *visual perception* [22]. His analysis of the computational complexity of visual perception yields the proof that visual perception is computationally intractable if the problem statement is unconstrained. His proof is independent of a specific implementation of vision and hence applies to both technical and biological visual systems.

Vision, however, is a performant and precise sense in biological systems, an observation that seems to disproof the result of Tsotsos at first glance. Starting from the evidence of vision in biological systems, Tsotsos exploits physical and biological constraints to reveal crucial building principles of system architectures that satisfy the complexity constraint of visual perception. The list of features of such architectures comprises:

- massively parallel and local processing;
- abstraction of the input by an input hierarchy;
- retinotopic representation of the input, i.e. a representation "whose physically adjacent elements represent spatially adjacent regions in the visual scene" [22].

Hierarchically organised representations of pictorial data are referred to as *pyramids* in computer vision [19, 4, 23, 11]. Pyramids are called *regular* if they are tapering stacks of regular grids. The individual grids are called the *levels* of the pyramid. The rigid structure of regular pyramids causes stability problems like shift and scale variance [3].

These stability problems were addressed in a paper of Meer [17], in which he proposed *decimation*, an algorithm that builds a new type of pyramid with a more flexible structure. In the levels of these pyramids, adjacency is represented explicitly by edges of a graph [18], rather than implicitly by an absolute position in a regular grid. The flexibility is due to the fact that each element of the graph representation may have an arbitrary number of neighbors. The pyramids are therefore called *irregular pyramids* [14]. The consequence of irregularity, however, is that neither the data structure for the description of elements of the representation, nor the time for the construction of a new pyramid level can be bounded.

Dual decimation, as presented in this paper, is a new algorithm that overcomes the problem of unbounded data structure inherent to irregular pyramids. The solution uses a dual graph for the control of both representation and construction as firstly proposed in [14]. Dual decimation is organised in such a way that neighborhood size can be bounded in the dual graph. The resulting hierarchy after recursive application of dual decimation is referred to as *dual irregular pyramid* [15].

This paper is an extended version of [28]. It is constructed as follows. Section 2 formally states the desired properties of an algorithm that builds a simplified representation of an

input image. Section 3 provides basic definitions from graph theory that will be used throughout the paper. An overview over previous work is contained in Section 4. Dual decimation is presented in Section 5, and the main properties of dual decimation are proved in Section 6. The conclusion outlines a perspective for the extension of the concept also to three dimensions.

2 Required properties of the process

This section becomes a bit more formal about the required features of a system architecture for visual perception. We address here the part of the system that generates a hierarchical representation of the input picture. We base our specification on Definitions 1 and 2 which we adopt from [8].

Definition 1 *A* **two dimensional image** *is a spatial representation of an object, a two-dimensional or three-dimensional scene. It may be abstractly thought of as a continuous function I of two variables defined on some bounded and usually rectangular region of a plane. The value of the image located at spatial coordinates (r, c) is denoted by $I(r, c)$.*

Definition 2 *A* **two dimensional digital image** *is an image in digital format and is obtained by partitioning the area of the image into a finite two dimensional array of small mutually exclusive convex polygons called* **resolution cells**. *Each resolution cell has a representative image value assigned to it. Resolution cells having a common side with the plane surrounding the image are said to lie on the* **boundary** *of the two dimensional digital image.*

Definition 3 *Let $\mathcal{R}(I)$ be a representation of a two dimensional digital image I, and let the elements r_i of $\mathcal{R}$ carry attributes that represent properties of I. Let further p_i denote a processing element that is assigned to an element r_i of $\mathcal{R}$. A processing element p_i is connected to a processing element p_j if the elements r_i and r_j of $\mathcal{R}$ are adjacent. A process operating on the representation $\mathcal{R}$ is called* **massively parallel** *if it is executed simultaneously in all processing elements p_i, and if the result obtained by one processing element p_j is independent of the result obtained by any other processing element $p_k; k \neq j$. Processing is called* **local** *if the result obtained by one processing element p_i is depending on attributes of r_i and of neighbors of r_i only, and if the number of neighbors is bounded for every processing element.*

The expected result of processing is a simplified representation of the input in which each element represents one or several elements of the original representation.

In order to take a global decision with respect to the whole image it must be possible to establish a relation between any two elements of the abstract representation.

3 Basic definitions from graph theory

The image representation $\mathcal{R}$ and the algorithm that performs its simplification will be formally described using a graph theoretic vocabulary. This section contains this vocabulary which has been gathered from standard textbooks on graph theory [1, 5, 9, 21].

Section 3.1 defines graphs as sets of vertices and edges, as well as the concepts of adjacency and degree. More complex notions like connectedness of graphs and subgraphs are defined in Section 3.2.

Sections 3.3 through 3.5 are devoted to duality between graphs, and Section 3.6 contains the definition of two basic operations on graphs that reduce the number of elements in a graph.

3.1 Graphs, vertices, edges

Definition 4 *A* **graph** $G = (V, E)$ *consists of two sets: a finite set* V *of elements called* **vertices** *and a finite set* E *of elements called* **edges**. *Each edge creates a binary relation between a pair of vertices.*

We use the symbols $v_1, v_2, v_3, \ldots$ to represent the vertices and the symbols $e_1, e_2, e_3, \ldots$ to represent the edges of the graph. Figure 1 shows the example of a graph. Vertices are

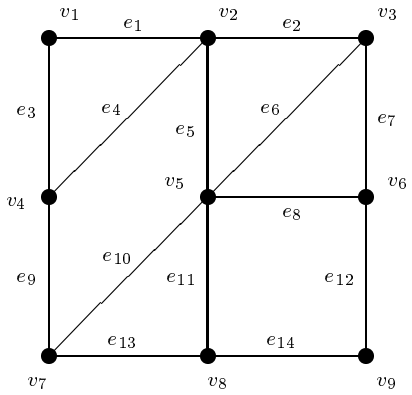


Figure 1: Pictorial representation of a graph $G(V, E)$.

represented by black spots (•), edges by straight lines.

Definition 5 *The vertices* v_i *and* v_j *associated with an edge* e_l *are called the* **end vertices** *of* e_l, e_l *is said to be incident to its end vertices. The edge is denoted as* $e_l = (v_i, v_j)$.

Definition 6 *More than one edge in a graph* $G(V, E)$ *may have the same pair of end vertices. All edges having the same pair of end vertices are called* **parallel edges**.

Definition 7 *An edge e_l is called a* **self-loop** *at vertex v_i if its end vertices are identical, i.e. $e_l = (v_i, v_i)$.*

Definition 8 *Two* **edges** *are* **adjacent** *if they have a common end vertex.*

Definition 9 *Two* **vertices** *are* **adjacent** *if they are the end vertices of some edge.*

Definition 10 *The number of edges incident to a vertex v of a graph $G(V, E)$ is called its* **degree** *or its* **valency**. *It is denoted by $d(v)$. A self-loop at a vertex v increases the degree of v by two.*

Definition 11 *A vertex of degree 1 is called a* **pendant vertex**.

Definition 12 *The edge incident to a pendant vertex is called a* **pendant edge**.

3.2 Paths, subgraphs, and components of a graph

Definition 13 *A* **path** *in a graph is a finite alternating sequence of vertices and edges $v_0,\ e_1,\ v_1,\ e_2,\ \ldots,\ v_{k-1},\ e_k,\ v_k$ such that*

1. *vertices v_{i-1} and v_i are the end vertices of the edge e_i, $1 \leq i \leq k$;*
2. *all edges are distinct;*
3. *all vertices are distinct.*

Vertices v_0 and v_k are called **end vertices of the path**, *and we refer to it as $v_0 - v_k$ path. The number of edges in a path is called the* **length of the path**.

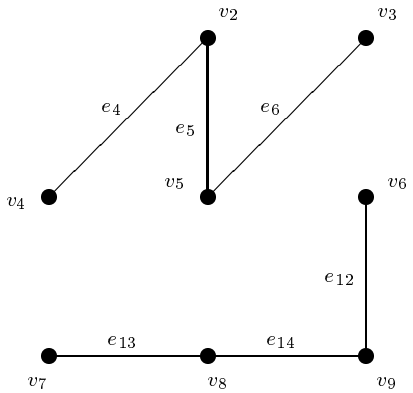


Figure 2: Two paths of G.

Figure 2 shows two example paths of the graph G in Figure 1.

Definition 14 *A graph G is* **connected** *if there exists a path between every pair of vertices in G.*

Definition 15 *Consider a graph $G = (V, E)$. $G' = (V', E')$ is a* **subgraph** *of G if V' and E' are, respectively, subsets of V and E such that an edge (v_i, v_j) is in E' only if v_i and v_j are in V'.*

Definition 16 *Let V' be a subset of the vertex set V of a graph $G = (V, E)$. Then the subgraph $G' = (V', E')$ is the induced subgraph of G on the vertex set V' (or simply* **vertex-induced subgraph** *$\langle V' \rangle$ of G) if E' is a subset of E such that an edge (v_i, v_j) is in E' if and only if v_i and v_j are in V'.*

Definition 17 *If e_l is an edge of a graph $G = (V, E)$, then $G - e_l$ is the subgraph of G that results after* **removing** *the edge e_l from G. Note that the end vertices of e_l are not removed from G. The removal of a set of edges from a graph is defined as the removal of single edges in succession.*

Definition 18 *A set V is said to be* **partitioned** *into subsets $V_1,\ V_2,\ \ldots,\ V_p$, if*

$$\begin{aligned} V_i &\neq \emptyset, 1 \leq i \leq p\ ; \\ V_1 \cup V_2 \cup \ldots \cup V_p &= V,\ and \\ V_j \cap V_k &= \emptyset,\ \forall\ j, k;\ j \neq k. \end{aligned}$$

We refer to subsets $V_1,\ V_2,\ \ldots,\ V_p$ as a **partition** *of V.*

Definition 19 *Consider a graph $G(V, E)$ which is not connected. Then the vertex set V of G can be partitioned into subsets $V_1,\ V_2,\ \ldots,\ V_p$ such that the vertex-induced subgraphs $\langle V_i \rangle$, $i = 1, 2, \ldots, p$, are connected and no vertex in subset V_i is connected to any vertex in subset V_j, $i \neq j$. We call subgraphs $\langle V_i \rangle$, $i = 1, 2, \ldots, p$,* **connected components** *or simply* **components** *of G.*

3.3 Circuits, cutsets, and dual graphs

Definition 20 *A* **circuit** *is a path the end vertices of which are identical.*

Figure 3 shows three circuits of the graph G in Figure 1.

Definition 21 *A* **cutset** *C of a connected graph G is a minimal set of edges of G with respect to inclusion ($\subseteq$) such that its removal from G disconnects G, that is, the graph $G - C$ is disconnected.*

Figure 4 shows graph G of Figure 1 after removal of the cutset $C = \{e_9, e_{10}, e_{11}, e_{12}\}$: $G - C$ is disconnected and consists of exactly two components $\langle \{v_1, v_2, v_3, v_4, v_5, v_6\} \rangle$ and $\langle \{v_7, v_8, v_9\} \rangle$

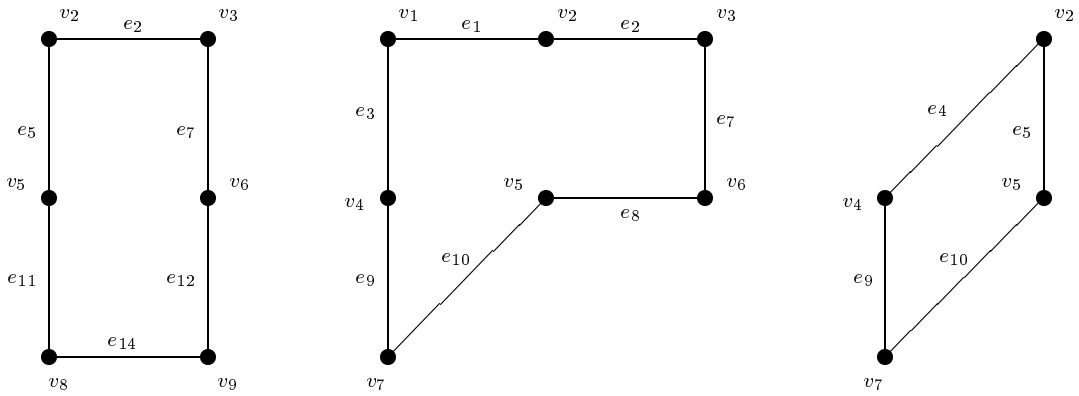


Figure 3: Three examples of circuits of G.

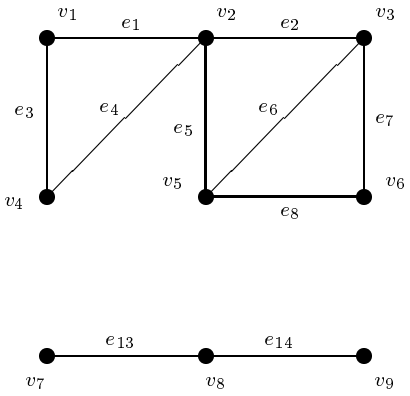


Figure 4: G after removal of the cutset $\{e_9, e_{10}, e_{11}, e_{12}\}$.

Definition 22 *A graph $\overline{G}$ is a* **dual** *of a graph G if there is a one-to-one correspondence between the edges of $\overline{G}$ and those of G such that a set of edges in $\overline{G}$ is a circuit if and only if the corresponding set of edges in G is a cutset.*

Duality is a symmetric relation, as we formally state in Theorem 1.

Theorem 1 *Consider two graphs G and $\overline{G}$. If $\overline{G}$ is a dual of G, then G is a dual of $\overline{G}$.*

The proof can be found in [21, p. 189].

3.4 Planar graphs and geometrical duals

The dual graph defined in Defintion 22 is also referred to as combinatorial dual. For pictorial representations of dual graphs we use a different way of defining dual graphs based on drawings of graphs on surfaces.

Definition 23 *A graph is said to be* **embeddable** *into a surface S if it can be drawn on S so that its edges intersect only at their end vertices.*

Definition 24 *A graph G is said to be* **planar** *if it can be embedded into a plane. Such a drawing of a planar graph G is called a* **planar embedding** *of G or simply a* **plane graph**.

Definition 25 *An embedding of a planar graph into a plane divides the plane into* **regions** *or* **faces**. *A region or face is* **finite** *if the area it encloses is finite; otherwise it is* **infinite**.

Definition 26 specifies how to obtain a dual graph from a planar embedding of a graph.

Definition 26 *Let G be a plane graph. Then the* **geometrical dual** $\overline{G}$ *of G is constructed as follows:*

1. *place a vertex $\overline{v_i}$ in each face f_i of G, including the exterior region;*
2. *if two regions f_i and f_j have an edge e in common, then join the corresponding vertices $\overline{v_i}$ and $\overline{v_j}$ by a line crossing only e; this line represents edge $\overline{e} = (\overline{v_i}, \overline{v_j})$.*

Figure 5 shows a simple example of a pair of dual graphs. Note the correspondence between

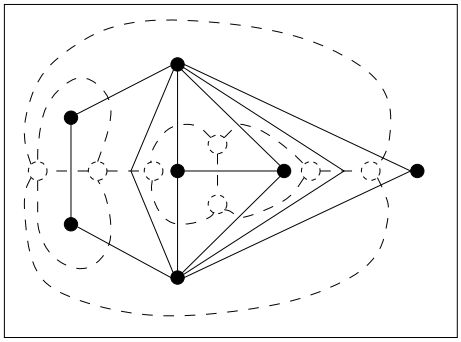


Figure 5: A graph G (solid lines) and its geometrical dual $\overline{G}$ (dashed lines).

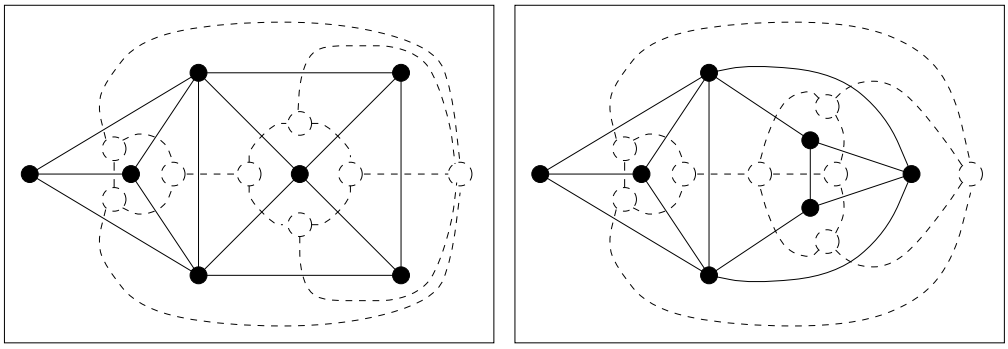
vertices of $\overline{G}$ and faces of G. Also, each edge of G corresponds to one edge of $\overline{G}$, indicated in the drawing by one dashed line crossing exactly one solid line.

Geometrical and combinatorial duals are equivalent as proved by Whitney [24, 26, loc. cit. [9]]. He stated Theorem 2 about the equivalence between planarity and the existence of a combinatorial dual. We cite Theorem 2 from [9, p. 115].

Theorem 2 *A graph is planar if and only if it has a combinatorial dual.*

3.5 Vertex connectivity and unique dual graphs

The geometrical dual is not unique for all graphs. Figure 6 illustrates an example found in [9, p. 114], where one planar graph has different planar embeddings and consequently different (geometrical) duals. Definition 27 provides a criterion that allows to decide whether



(a) G and its dual $\overline{G}$.

(b) A different embedding yields a different dual.

Figure 6: One planar graph $G(V, E)$ may have different duals.

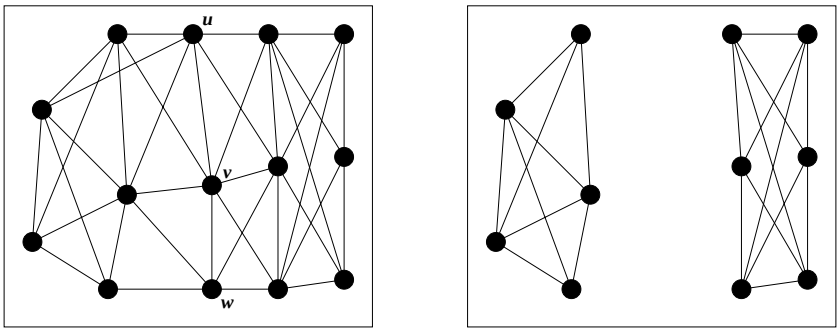
a graph has a unique dual.

Definition 27 *The* **vertex connectivity** *or* **connectivity** $\kappa = \kappa(G)$ *of a graph G is the minimum number of vertices the removal of which results in a disconnected or trivial graph. A graph is* **n-connected** *if $\kappa(G) \geq n$.*

Figure 7 a shows the example of a 3-connected graph G. G is disconnected after the removal of three vertices u, v, w (Figure 7 b). We formally state the relation between vertex connectivity of a planar graph and the uniqueness of its dual in Theorem 3.

Theorem 3 *A three-connected planar graph G has a unique dual.*

The proof of this theorem is based on results of Whitney [25, loc. cit. [9]] about unique embeddings of planar graphs and can be found in [9, p. 114].



(a) $G(V, E)$. (b) $\langle V - \{u, v, w\} \rangle$ is disconnected.

Figure 7: Example of a three-connected graph $G(V, E)$.

3.6 Vertex identification and edge contraction

We conclude Section 3 with Definitions 28 and 29 of operations on a graph G that reduce the number of vertices of G by a local operation.

Definition 28 *A pair of vertices v_i and v_j in a graph G are said to be* **identified** *(or* **short-circuited***) if the two vertices are replaced by a new vertex such that all edges incident to v_i and v_j are now incident to the new vertex.*

Definition 29 Contraction *of an edge e is the operation of removing e and identifying its end vertices.*

Figure 8 illustrates the difference between the identification and contraction operations for the example of the graph in Figure 1.

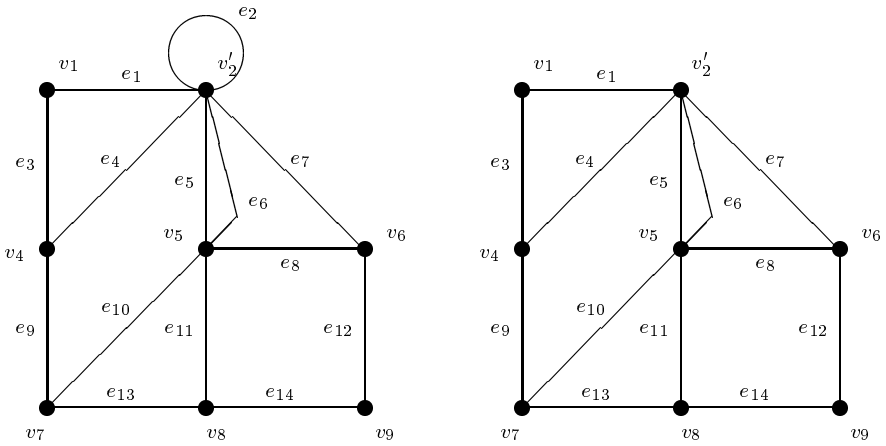
Edge e_2 forms a self-loop at the new vertex v_2' after vertices v_2 and v_3 are identified, whereas the contraction operation eliminates e_2.

4 Previous work

The work reported in this document has been motivated by previous work on decimation [17, 18] and on the construction of dual irregular pyramids [14, 15]. We report the major results with respect to dual decimation in Sections 4.1 and 4.2.

4.1 Decimation

Meer was the first to propose a hierarchical representation of image content in which a regular neighborhood structure was given up [17]. He proposed stochastic decimation, a



(a) **Identification** of v_2 and v_3. (b) **Contraction** of e_2.

Figure 8: Identification and contraction in $G(V, E)$.

graph transformation that operates on a graph G representing the neighborhood structure of a digital image. Dual decimation starts by selecting a vertex subset from the graph.

Definition 30 Stochastic decimation *of a graph is a process that is executed in the following steps:*

1. *Random numbers are assigned to the vertices.*
2. *Vertices with a local maximum of this variable* **(surviving vertices)** *are selected.*
3. *All* **non-surviving vertices** *are assigned to one survivor out of their neighborhood.*
4. *Repeat steps 1 ... 3 for all non-survivors who do not have a survivor in their neighborhood.*
5. *The* **receptive field** $RF(v_s)$ *of a survivor* v_s *is formed by all non-survivors* **(sons)** *that are assigned to* v_s **(father)**. *It also includes* v_s *itself.*
6. *Vertices of the reduced graph are connected if vertices of their receptive fields are connected in the original graph.*

As described in [17], two rules have to be respected by the vertices during decimation:

1. *Every non-surviving vertex has at least one surviving neighbor;*
2. *Two adjacent vertices must not survive both.*

The selection of surviving vertices can also be related to the image content to obtain an adaptive behavior [10]. We therefore use the more generic term decimation to refer to the process.

4.2 Dual irregular pyramids

Decimation involves a scan of the entire neighborhood for each vertex of the graph during the connection of surviving vertices. This requires that the list of neighbors be stored explicitly for each vertex. Or, if we think of a massively parallel architecture, then each processing element representing a vertex may have an arbitrary number of links to adjacent processing elements.

This problem of unbounded vertex degree was addressed in [14]. It was observed that the dual graphs of the levels of an irregular pyramid remains bounded. The idea was to to build a dual irregular pyramid complementing each level of an irregular pyramid by its dual graph during construction. Then the dual graphs could be used as a control structure for the construction process, and the processing elements assigned to the vertices of the dual graphs would have a limited number of links to adjacent processing elements.

However, the process defined in [14] does not yield a pair of dual graphs in all cases, and neither can the degree of the dual graphs be bounded, as was shown in [27].

5 Dual decimation

This section contains the definition of dual decimation, an algorithm that overcomes the problems mentioned in Section 4. Dual decimation operates on a representation of an image by a pair of dual graphs. This representation is formally defined in Section 5.1.

The basic operation used by dual decimation is dual contraction, which we define in Section 5.2. Section 5.3 describes how control information can be generated for dual contraction.

After dual contraction, redundancies may occur in the neighborhood graph. We demonstrate this in Section 5.4, and show how these redundancies can be detected and removed by local processes in the face graph.

5.1 Representation of image structure

A digital image can be viewed as a plane graph with the resolution cells being the faces of the embedding. A plane graph $G(V, E)$ and its geometrical dual $\overline{G}(\overline{V}, \overline{E})$ can therefore be derived from a digital image in analogy to Definition 26.

Definition 31 *Let I be a two dimensional digital image, and let the resolution cells of I be denoted as c_i. Then a graph $G(V, E)$ and its geometrical dual $\overline{G}(\overline{V}, \overline{E})$ can be obtained from I by the following process:*

1. *place a vertex $v_i \in V$ in every resolution cell c_i;*

2. *if two cells c_i, c_j have a common side, then draw a line between vertices v_i and v_j in such a way that the common side is intersected; this line represents the edge $(v_i, v_j) \in E$;*

3. *construct the geometrical dual $\overline{G}$ of G according to Definition 26.*

We refer to graph G as the **neighborhood graph** *since its edges explicitly represent adjacency between elements of the representation of I. Graph $\overline{G}$ is called* **face graph** *since vertices $\overline{v} \in \overline{V}$ represent faces of a planar embedding of G.*

Note that we will use the terms neighborhood graph and face graph also for graphs that represent an image in a more abstract way. In this case, vertices of the neighborhood graph G represent regions or objects of the scene rather than individual resolution cells.

Since we do not want to impose a bound on the number of neighbors that a region in an image may have, we may not bound the vertex degree in the neighborhood graphs. The consequence for the data structure representing vertices of G is the following. If the list of neighbors was stored explicitly for each vertex $v \in V$, then the data structure for the representation of $v \in V$ would not be bounded. Therefore we do not explicitly represent the list of neighbors with each vertex $v \in V$.

5.2 Dual contraction of edges

Definition 32 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be dual graphs, and let $e = (n, s)$ be an edge of G. We refer as* **dual contraction** *of edge e to the following sequence of steps:*

1. *in each edge incident to n replace end vertex n by s;*

2. *remove vertex n;*

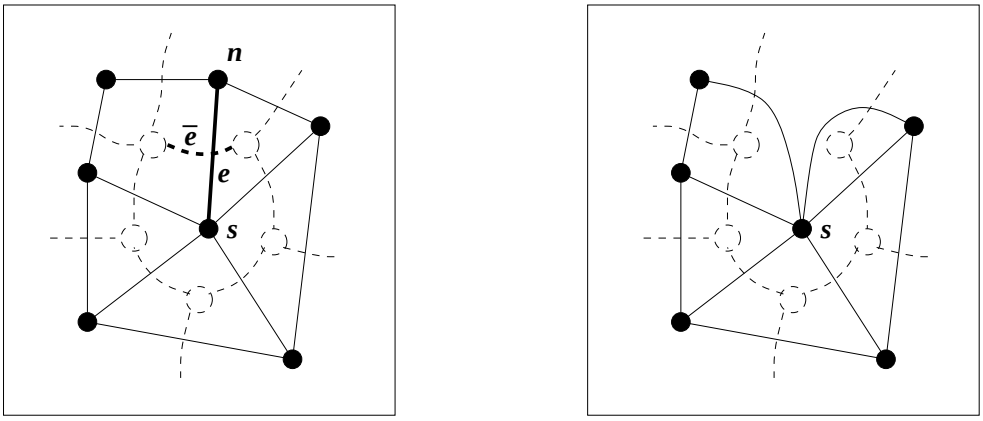
3. *remove edge e and edge $\overline{e} \in \overline{E}$ corresponding to e.*

We refer to Steps 1 and 2 as the **identification of n with s.**

Note that dual contraction is defined for vertices of both G and $\overline{G}$ since G is also the dual of $\overline{G}$, cf. Theorem 1.

Figure 9 illustrates the process of dual contraction by means of a simple example. Figure 9 a shows a small portion of a graph G (solid lines) together with its dual $\overline{G}$ (dashed lines). Edge $e = (n, s)$ and its corresponding edge are outlined in bold. Figure 9 b shows the resulting pair of dual graphs after identification of vertex n with vertex s due to dual contraction of edge e.

Dual contraction of an edge e preserves the duality relation between a pair of dual graphs, as we formally state in Theorem 4. The proof of this theorem can be found in [21, p. 190]. We cite it here for reasons of completeness. If the graph containing e is connected, then it remains connected after the dual contraction operation as we will state in Lemma 1.



(a) $G(V,E)$ and $\overline{G}(\overline{V},\overline{E})$.

(b) Result after dual contraction of e.

Figure 9: Dual contraction of edge e of $G(V,E)$.

Theorem 4 *Consider two dual graphs G and $\overline{G}$. Let $e=(v_1,v_2)$ be an edge in G, and $\overline{e}=(\overline{v_1},\overline{v_2})$ be the corresponding edge in $\overline{G}$. Let G' and $\overline{G'}$ be the graphs resulting from G and $\overline{G}$ after dual contraction of edge e. Then G' and $\overline{G'}$ are duals, the one-to-one correspondence between their edges being the same as in G and $\overline{G}$.*

Proof : Let C and $\overline{C}$ denote corresponding sets of edges in G and $\overline{G}$, respectively.

Suppose $\overline{C}$ is a circuit in $\overline{G'}$. Since it does not contain $\overline{e}$, it is also a circuit in $\overline{G}$. Hence C is a cutset in G, it disconnects G into exactly two components, say $\langle V_1\rangle$ and $\langle V_2\rangle$. Since C does not contain e, the vertices v_1 and v_2 are both in V_1 or in V_2. Therefore C is also a cutset in G'. Thus every circuit in $\overline{G'}$ corresponds to a cutset in G'.

Suppose C is a cutset in G'. Since C does not contain e, it is also a cutset in G. Hence $\overline{C}$ is a circuit in $\overline{G}$. Since it does not contain $\overline{e}$, it is also a circuit in $\overline{G'}$. Thus every cutset in G' corresponds to a circuit in $\overline{G'}$. ■

Lemma 1 *Consider two connected dual graphs G and $\overline{G}$. Consider further an edge $e=(v_i,v_j)$ in G that is dually contracted, and let v_i be identified with v_j. Let G' and $\overline{G'}$ be the graphs resulting from G and $\overline{G}$ after dual contraction of edge e. Then G' is again a connected graph.*

Proof : Let P be a $v_a - v_z$ path containing e, and let $e_- = (v_{i-1}, v_i)$ and $e_+ = (v_j, v_{j+1})$ be edges in P adjacent to e. After dual contraction of e, e_+ is incident to v_i, hence v_{j+1} is adjacent to v_i, and the resulting path P' is still connecting v_a to v_z.

Let Q be a $v_b - v_y$ path containing v_j, and let $e_- = (v_{j-1}, v_j)$ and $e_+ = (v_j, v_{j+1})$ be adjacent edges in Q. Then $e_- = (v_{j-1}, v_i)$ and $e_+ = (v_i, v_{j+1})$ after identification of v_j with v_i, hence e_- and e_+ are still adjacent, and Q is still connecting v_b to v_y.

We have shown that all $v_a - v_z$ paths of G containing e remain $v_a - v_z$ paths after dual contraction of e. We have further shown that all $v_b - v_y$ paths of G containing v_j still connnect v_b to v_y after dual contraction of e. All other paths of G remain unchanged, hence G' is also connected. ■

5.3 Selection of surviving and non-surviving vertices

Control information for dual contraction can be generated by the selection of surviving vertices, and by the assignment of non-surviving vertices to surviving vertices, similar to decimation. However, the rules that apply for the selection process are less restrictive than the rules given by Meer.

Definitions 33 through 35 are useful for a comparison of the vertex selection processes used by decimation and dual decimation. Definition 36 formally defines the selection process used by dual decimation as well as related terminology.

Definition 33 *Consider a graph $G(V, E)$. A subset S of V is an* **independent vertex set** *or* **internally stable set** *of G if no two vertices of G are adjacent in G. An independent set S of G is* **maximum** *if G has no independent set S' such that $card(S') > card(S)$. An independent set S of G is* **maximum with respect to inclusion** *or simply a* **maximal independent vertex set** *if every vertex of $V - S$ has at least one vertex of S adjacent to it.*

Definition 34 *The number of vertices in a maximum independent set of a graph G is called the* **stability number** *or* **independence number** *of G.*

Definition 35 *Consider a graph $G(V, E)$. A set $S \subseteq V$ is a* **dominating vertex set** *or* **externally stable set** *of G if every vertex $v \in V - S$ has a vertex $s \in S$ adjacent to it.*

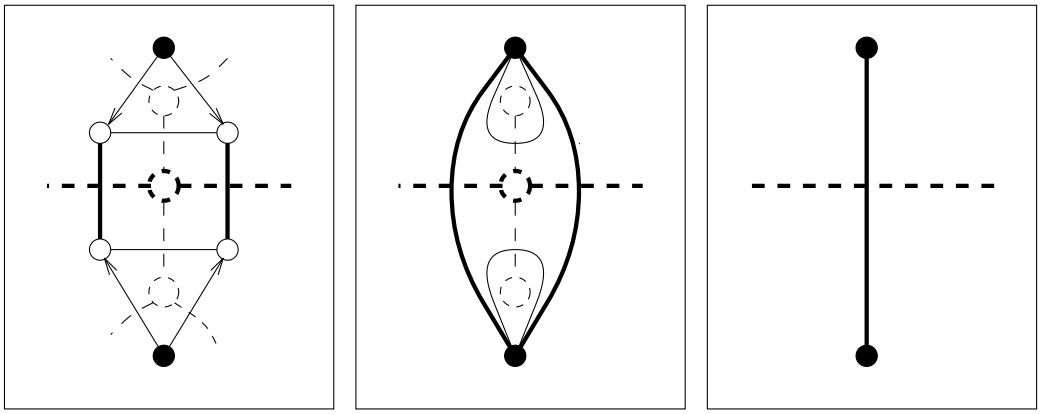
Note that a maximal independent set of a graph is both internally and externally stable. The selection process suggested by Meer yields a maximal independent vertex set S of the input graph. This limits the number of selected vertices to the stability number of the input graph. At the difference to Meer, we do not require that the selected vertices form an independent set.

Definition 36 *Consider a graph $G(V, E)$ representing a digital image. Partition set V into two disjoint subsets S, N. We refer to vertices $s \in S$ as* **surviving vertices**, *and to vertices $n \in N$ as* **non-surviving vertices**. *Do the partition in such a way that every non-surviving vertex has at least one surviving neighbor. A mapping $N \rightarrow S$ assigns to every vertex $n \in N$ one vertex $s \in S$ that is adjacent to it. We call the surviving vertex* **parent** *with respect to the non-surviving vertex, which we call* **child**. *Mapping $N \rightarrow S$ is called* **child-parent assignment**.

The selection process defined in Definition 36 allows the selection of adjacent vertices. Set S selected according to Definition 36 is externally stable and may, but need not be internally stable as well. This allows greater flexibility during vertex selection in that it admits the selection of as many vertices as the input graph has.

5.4 Redundant edges, removable faces, and face contraction

Figure 10 a shows how control information generated by survivor selection and child-parent assignment is used during dual contraction. The illustration shows a small portion of a graph G (edges drawn as solid lines) and its dual $\overline{G}$ (edges drawn as dashed lines). Black spots ($\bullet$) indicate survivors, circles (o) indicate non-survivors, arcs pointing from parent to child indicate child-parent connections. In Figure 10 b, all children have disappeared



(a) Example graph before ...

(b) ... and after dual contraction.

(c) Result without redundancies.

Figure 10: Elimination of redundancies caused by dual contraction.

after dual contraction of child-parent connections. The graph resulting from G contains self-loops at, and parallel edges between survivors.

Definition 37 *Consider self-loops $e_s \in E$ and parallel edges $e_p \in E$ in a graph $G(V, E)$. We refer to e_s and e_p as* **redundant edges** *since their removal does not change adjacency of vertices in G.*

To remove redundant edges in G, we execute dual contraction operations in $\overline{G}$ to preserve duality between G and $\overline{G}$ (cf. Theorem 4). Before we formally define the contraction operation, we state Lemma 2 and Theorem 5 about dual contraction and vertex degree.

Lemma 2 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be dual graphs, and let edges $e = (n, s) \in E$ and $\overline{e} = (\overline{v}, \overline{w}) \in \overline{E}$ be corresponding edges. Dually contract edge e, identifying n with s. Consider degrees $d(s), d(n), d(\overline{v}), d(\overline{w})$ of vertices $s, n, \overline{v}, \overline{w}$ before dual contraction of edge e. Then degrees $d'(s), d'(\overline{v}), d'(\overline{w})$ of vertices $s, \overline{v}, \overline{w}$ after dual contraction are:*

$$d'(s) \;=\; d(s) + d(n) - 2;$$

$$\begin{aligned} d'(\overline{v}) &= d(\overline{v}) - 1; \\ d'(\overline{w}) &= d(\overline{w}) - 1. \end{aligned}$$

Proof :

$d'(s)$: After identification of n with s, all edges incident to s and n are now incident to s, hence $d'(s) = d(s) + d(n)$. Edge e is now a self-loop at s, and increases $d'(s)$ by 2, hence $d'(s) = d(s) + d(n) - 2$ after removal of e.

$d'(\overline{v}), d'(\overline{w})$: Edge $\overline{e}$ is incident to vertices $\overline{v}$,$\overline{w}$, hence $d'(\overline{v}) = d(\overline{v}) - 1$ and $d'(\overline{w}) = d(\overline{w}) - 1$ after removal of $\overline{e}$. ■

Theorem 5 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be dual graphs, and let vertex $\overline{v} \in \overline{V}$ have degree one or two. Consider further an edge $\overline{e} = (\overline{v}, \overline{w})$, that is dually contracted in such a way that $\overline{v}$ is identified with $\overline{w}$. Then the corresponding edge $e \in E$ to be removed is redundant in G, and the degree of $\overline{w}$ is non-increasing.*

Proof : All edges incident to a graph's vertex form a cutset the removal of which separates this vertex from the rest of the graph. In the case of $\overline{v}$, the cutset contains one or two edges, corresponding to a circuit of length one or two in G. Therefore edge e corresponding to the dually contracted edge $\overline{e}$ is redundant, i.e. either a self-loop or one of two parallel edges. The degree $d'(\overline{w})$ after dual contraction of $\overline{e}$ is

$$d'(\overline{w}) = d(\overline{w}) + d(\overline{v}) - 2 \leq d(\overline{w}),$$

hence the degree of $\overline{w}$ is non-increasing. ■

Definition 38 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be dual graphs. We refer to vertices of $\overline{G}$ as faces since they represent faces of a planar embedding of G. Let face $\overline{v} \in \overline{V}$ have degree one or two. Since the removal of $\overline{v}$ by dual contraction of an edge $\overline{e} = (\overline{v}, \overline{w})$ preserves duality between G and $\overline{G}$, and since this removal does not increase the degree of $\overline{w}$, we refer to $\overline{v}$ as a* **removable face**.

We now have the necessary elements to formally define the contraction operation for the removal of redundant edges.

Definition 39 *Consider a pair of dual graphs $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$, with G containing redundant edges and $\overline{G}$ containing a set $\overline{V_\varrho} \subset \overline{V}$ of removable faces. Then the following process eliminates redundant edges from G.*

1. *partition set $\overline{V}$ into two sets $\overline{N}$* **(non-surviving faces)** *and $\overline{S}$* **(surviving faces)** *such that*
   - *all faces $\overline{v} \in (\overline{V} - \overline{V_\varrho})$ are surviving;*
   - *a surviving face $\overline{s} \in \overline{V_\varrho}$ must not have a surviving face adjacent to it;*

- *every non-surviving face $\overline{n} \in \overline{N}$ has at least one adjacent surviving face;*

2. *every non-surviving face $\overline{n} \in \overline{N}$ is assigned to exactly one surviving face $\overline{s} \in \overline{S}$ adjacent to it; $\overline{s}$ is referred to as* **parent** *of $\overline{n}$, $\overline{n}$ is called* **child** *of $\overline{s}$; $\overline{E}_{cp}$ denotes the set of edges having a surviving face $\overline{s}$ as one end vertex and a child of $\overline{s}$ as the other end vertex;*

3. *dually contract edges $\overline{e} \in \overline{E}_{cp}$, identifying children with parents;*

4. *repeat Steps 1 ... 3 until $\overline{V_\varrho} = \emptyset$.*

Since the process executed in Steps 1 ... 4 eliminates removable faces, we call it **face contraction**.

5.5 Combining the elementary processes

In the previous subsections, we have gathered all subprocesses we need to define the process of dual decimation.

Definition 40 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be a pair of graphs dual to each other, G being referred to as the neighborhood graph, $\overline{G}$ being referred to as the face graph.*

We refer as **dual decimation** *to the following sequence of processes applied to this pair of graphs:*

1. *select surviving and non-surviving vertices in the neighborhood graph, and assign children to parents;*

2. *dually contract edges $e \in E$ connecting children to parents, i.e. do not dually contract self-loops;*

3. *remove redundancies in the neighborhood graph by face contraction.*

Note that Step 2 of dual decimation requires careful examination if the adjacency between a non-surviving vertex and its parent is defined by more than one edge. The most simple example of this case is a non-surviving vertex n that is connected to its parent by two edges e_1 and e_2, i.e. $e_1 = e_2 = (n, s)$. Parallel edges e_1 and e_2 correspond to a cutset $\{\overline{e_1}, \overline{e_2}\}$ in $\overline{G}$. If both edges e_1, e_2 are dually contracted, then graph $\overline{G}$ disconnects.

Parallel edges in G cannot be detected by a read access to the representation of vertices since the representing data structure does not contain a list of neighbors of a vertex. Neither can it be detected in $\overline{G}$, since the edges in $\overline{G}$ corresponding to parallel edges in G are not necessarily adjacent in $\overline{G}$.

To avoid disconnection of $\overline{G}$ during Step 2 of dual decimation, however, it is sufficient to avoid that a non-surviving vertex is identified with its parent more than once. This requires that the access to representations of vertices in G is organised in such a way that simultaneous write accesses are not possible.

6 Properties of dual decimation

This section relates dual decimation to the specification in Section 2.

We assume that dual decimation is to be executed by a massively parallel processing architecture in which one processing element is assigned to each vertex of the representation. Processing in such an architecture is local if the degree of vertices in the underlying graph is bounded. This condition is not satisfied for the neighborhood graph G because vertex v_E is connected to all vertices representing resolution cells on the boundary of the image.

However, vertex degree can be bounded in the face graph $\overline{G}$, and is non-increasing during dual decimation, as we will show in Section 6.1. Processing elements assigned to vertices of $\overline{G}$ can control the neighborhood graph due to the duality relation between G and $\overline{G}$. This relation is preserved by dual decimation as we will proof in Section 6.2.

The representation obtained from dual decimation must allow to establish a relation between any two elements of this representation. Therefore the neighborhood graph G must remain connected. Since $\overline{G}$ is used as a control structure, also $\overline{G}$ must remain connected. The proof of the connectivity preserving property of dual decimation is contained in Section 6.3.

Face contraction has a sequential character to it, as we will illustrate in Section 6.4. However, the effect of a dual contraction in the face graph can be bounded to well-defined subgraphs of the face graph. In each of these subgraphs, dual contractions can be executed independently. The number of processing steps does not depend on the number of vertices in the face graph, cf. Section 6.4.8.

6.1 Degree preservation

Theorem 6 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be a pair of graphs dual to each other. Let the process of dual decimation be applied to this pair of graphs, with the extraction of surviving vertices and child-parent assignment in G, and the processes of dual contraction and face contraction derived from this assignment as described in Definition 40.*

Let $\overline{G'}$ be the graph that results from $\overline{G}$ after applying dual decimation. Then the degree of vertices of $\overline{G'}$ is less or equal to the degree of vertices of $\overline{G}$.

Proof : During dual decimation (cf. Definition 40), the face graph is modified in two cases:

- when an edge e of the neighborhood graph is dually contracted;
- when a removable face is eliminated by dual contraction.

As for the first case, we have stated in Lemma 2 that the removal of edge $\overline{e}$ corresponding to e decreases the degree of the end vertices of $\overline{e}$. The degree of all other vertices $\overline{v} \in \overline{V}$ remains unchanged. During face contraction, only removable faces are eliminated by

dual contraction operations, non-increasing degree for this case has already been stated in Theorem 5.

We have shown that the degree of vertices of the face graph is reduced during dual contraction of edges as well as during face contraction. This proves the theorem, since these are the only operations that modify the face graph during dual decimation. ■

6.2 Duality preservation

Theorem 7 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be a pair of dual graphs, and let the process of dual decimation be applied to this pair. Let G' and $\overline{G'}$ be the graphs resulting from processing G and $\overline{G}$, respectively. Then G' and $\overline{G'}$ are dual graphs, the correspondence between their edge sets being the same as between G and $\overline{G}$.*

Proof : We recall that by Definition 40, only the following processes modify graphs G and $\overline{G}$ during dual decimation:

- dual contraction of an edge;
- face contraction.

Theorem 4 states the duality preserving property for the process of dual contraction of an edge. The elimination of removable faces during face contraction is again a dual contraction operation, applied to a pair of dual graphs, and consequently preserves their duality relation.

We have shown that both dual contraction of edges and face contraction preserves the duality relation between a pair of graphs. This proves the theorem, since these are the only processes that modify the graphs during dual decimation. ■

6.3 Connectivity preservation

The connectivity of both graphs allows a global evaluation due to communication between any two parts of the representation.

Theorem 8 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be two connected graphs dual to each other, and let the process of dual decimation be applied to this pair. Let G' and $\overline{G'}$ be the graphs resulting from processing G and $\overline{G}$, respectively. Then G' and $\overline{G'}$ are both connected.*

Proof : In Step 2 of dual decimation, dual contraction of edges, edges of G are dually contracted, in Step 3 (face contraction) edges of $\overline{G}$ are dually contracted. In Lemma 1, we have already stated that dual contraction of an edge in a graph preserves the connectivity of this graph. Therefore Step 2 preserves connectivity of G, while Step 3 preserves connectivity of $\overline{G}$. To show the connectivity preservation for the whole process of dual decimation, we consequently show that Step 2 also preserves connectivity of $\overline{G}$, and that Step 3 preserves connectivity of G.

The preservation of the connectivity of G during face contraction (Step 3) is due to the fact that during face contraction only redundant edges are removed from G.

By definition of Step 2, only edges connecting children to parents are dually contracted. To disconnect $\overline{G}$, it is necessary and sufficient to completely remove a cutset from $\overline{G}$, hence to dually contract all edges of a circuit C in G. This requires that all edges of C be parent-child connections. For the discussion of this condition we distinguish three classes of circuits C in G:

Circuits of length 1: Circuits of length 1 are by definition excluded from dual contraction.

Circuits of length 2: A circuit C of length 2 contains two vertices, say n and s. Let n be a non-surviving vertex, and let s be a surviving vertex. Let further s be the parent of n. Then both edges of C are parent - child connections. However, after dual contraction of one edge of C, the other edge is a self-loop at s, the dual contraction of which is excluded by definition.

Circuits of length $>$ 2: Assume that all edges of C be child-parent connections. This is equivalent to saying that c is of even length, and that vertices of C form an alternating sequence of surviving and non-surviving vertices. Consider now a non-surviving vertex n of C. Two edges e_n, e_{n+1} of C are incident to n, connecting n to one surviving vertex each. If both e_n and e_{n+1} were child-parent connections, then vertex n would be child of two parent vertices, which is a contradiction to Definition 36.

We have shown that no circuit C in G can be found the edges of which are all dually contracted, hence the connectivity of $\overline{G}$ is preserved during Step 2 of dual decimation.

We have shown that both dual contraction of edges and face contraction preserve the connectivity of both graphs G and $\overline{G}$. This proves the theorem, since these are the only processes that modify G and $\overline{G}$ during dual decimation. ■

6.4 Complexity of face contraction

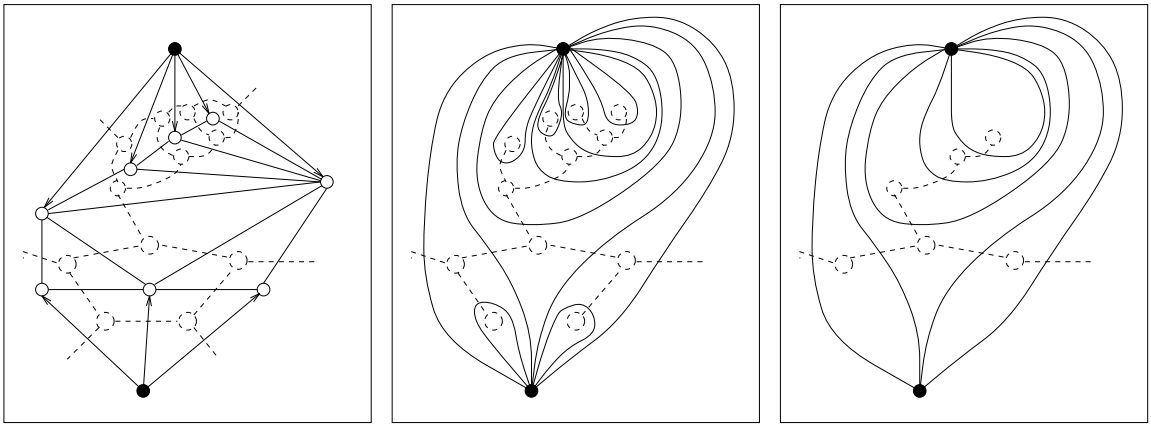
Face contraction is not in general a parallel process, since the elimination of a removable face $\overline{v_\varrho}$ with $d(\overline{v_\varrho}) = 1$ by dual contraction may result in a new removable face. Figure 11 illustrates the sequential nature of face contraction by means of a simple example. Figure 11 a shows a small portion of a neighborhood graph G (solid lines) and the corresponding face graph $\overline{G}$ (dashed lines) with surviving and non-surviving vertices, Figure 11 b shows the result after dual contraction. Vertices of $\overline{G}$ are of degree 1 (hence removable faces) and 3. The elimination of degree-1 vertices generates new removable faces of degree 1 and 2 (Figure 11 c) which can be removed in a new iteration and so on until all removable faces are eliminated (Figure 11 f).

The objective of this section is to give an upper bound for the number of sequential steps that are required to remove all removable faces from a graph. The demonstrations will be guided by the following two questions:

? Can the elimination of a face $\overline{v_\varrho}$ with $d(\overline{v_\varrho}) = 1$ by dual contraction of an edge in $\overline{G}$ affect the removability of all other faces in the graph, or can the effect of this elimination be limited to a subgraph of $\overline{G}$?

? Is there consequently an upper bound to the number of iterations of face contraction that is less than the number of vertices in $\overline{G}$?

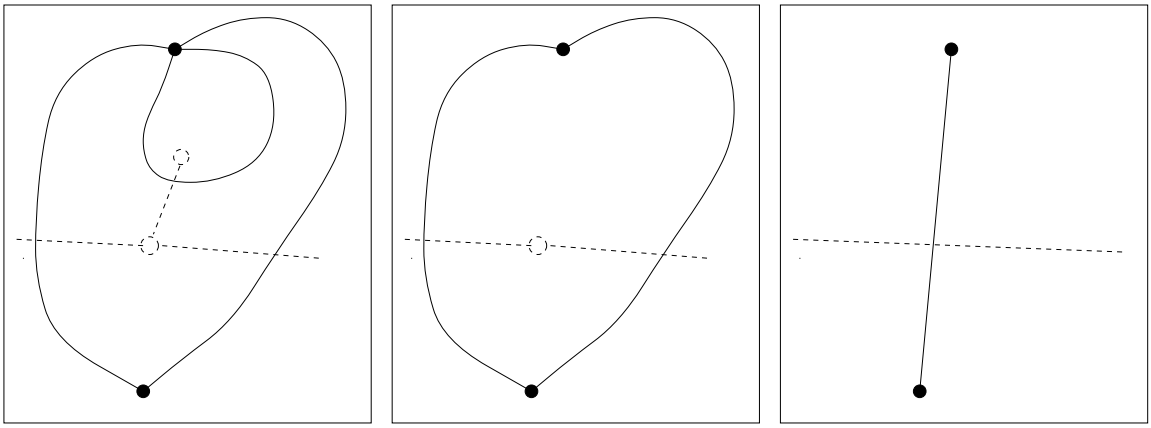
For the discussion, we demonstrate how child-parent assignment implies a partition of the neighborhood graph (Sections 6.4.1). This partition is used in Section 6.4.2 to partition



(a) Graph before ...

(b) ... and after dual contraction.

(c) First iteration of face contraction.



(d) Second iteration of face contraction.

(e) Third iteration of face contraction.

(f) Result after four iterations.

Figure 11: Iterative elimination of redundant edges by face contraction.

also the vertices of the face graph, and we obtain two classes of vertices, region faces and boundary faces.

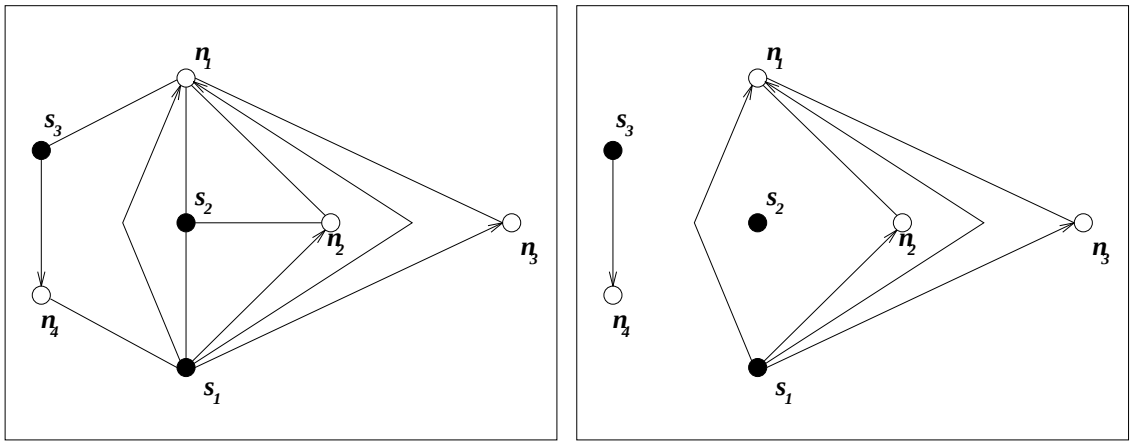
Region faces are lying on tree structures after Step 2 of dual decimation, as we will show in Section 6.4.3. We also show in the same section that edges incident to regions faces correspond to self-loops in G. In Section 6.4.4, we show that only the elimination of a region face by dual contraction may create a new removable face. However, the elimination of a region face affects only a limited part of the graph, as we will proof in Section 6.4.5.

This result is then used in Sections 6.4.6 and 6.4.7 to specify upper bounds that apply for the time complexity of self-loop elimination and the elimination of parallel edges. Section 6.4.8 concludes by giving the overall time complexity of face contraction:

6.4.1 Surviving vertices and regions of the neighborhood graph

Definition 41 *Consider a graph $G(V, E)$ with set V partitioned into surviving and non-surviving vertices, with the non-surviving vertices being mapped onto the surviving vertices by child-parent assignment. Let set $V(s_i) \subset V$ contain a surviving vertex s_i and all children of s_i. According to [7, p. 217], we refer to set $V(s_i)$ as the* **cluster of a surviving vertex** *s_i. Vertex induced subgraph $\langle V(s_i) \rangle$ is called the* **region of a surviving vertex** *s_i.*

We denote the set of edges of region $\langle V(s_i) \rangle$ as $E(s_i)$, hence $\langle V(s_i) \rangle = (V(s_i), E(s_i))$. Figure 12 a shows the example of a graph $G(V, E)$ with surviving vertices $S = \{s_1, s_2, s_3\}$ and non-surviving vertices $N = \{n_1, n_2, n_3, n_4\}$. Child-parent assignments are illustrated



(a) Mapping $N \to S$ indicated by (o←•). (b) Regions $\langle V(s_1) \rangle$, $\langle V(s_2) \rangle$, $\langle V(s_3) \rangle$.

Figure 12: Graph $G(V, E)$ with regions induced by mapping $N \to S$.

by arcs pointing from *parent* to *child*. The clusters defined by the assignment are the following:

$$\begin{aligned} V(s_1) &= \{s_1, n_1, n_2, n_3\}; \\ V(s_2) &= \{s_2\}; \\ V(s_3) &= \{s_3, n_4\}. \end{aligned}$$

Figure 12 b shows regions $\langle V(s_1)\rangle$, $\langle V(s_2)\rangle$, $\langle V(s_3)\rangle$ of survivors s_1, s_2, s_3.

Lemma 3 *Consider a graph $G(V, E)$ with V being partitioned into surviving (S) and non-surviving (N) vertices. Let m be the number of surviving vertices. Then clusters $V(s_1), \ldots, V(s_m)$ defined by child-parent assignment $N \to S$ are a partition of set V.*

Proof :

Partition of S: Define sets $V(s_i) = \{s_i\}$; $i = 1, \ldots, m$. Sets $V(s_1), \ldots, V(s_m)$ represent a partition of S since

$$\begin{aligned} V(s_i) \cap V(s_j) &= \emptyset;\ \forall i, j;\ i \neq j; \\ V(s_1) \cup \ldots \cup V(s_m) &= S. \end{aligned}$$

Partition of N: By definition, every vertex $n \in N$ is assigned to exactly one survivor s_i. Then n is added to set $V(s_i)$ containing s_i:

$$V(s_i) = V(s_i) \cup \{n\}.$$

Vertex sets $V(s_i)$ remain mutually disjoint since every non-survivor is joint to exactly one set $V(s_i)$. All vertices $n \in N$ are assigned, hence

$$V(s_1) \cup \ldots \cup V(s_m) = S \cup N.$$

$S \cup N = V$, hence $V(s_1), \ldots, V(s_m)$ is a partition of set V. ■

Definition 42 *Consider a connected graph $G(V, E)$ with vertex set V. Let V_1 and V_2 be a partition of V. Then the set C of all edges having one end vertex in V_1 and the other end vertex in V_2 is called a* **cut** *of G.*

According to [21, p. 44], we denote a cut by $\langle V_1, V_2\rangle$. Note that the notion of a cut is closely related to the one of a cutset (Definition 21). This relationship is stated more formally in Theorems 9 and 10 which we cite from [21, p. 45].

Theorem 9 *A cut in a connected graph G is a cutset or union of edge-disjoint cutsets.*

Theorem 10 *A cut $\langle V_1, V_2\rangle$ of a connected graph G is a cutset of G if the induced subgraphs of G on the vertex sets V_1 and V_2 are connected. If S is a cutset of a connected graph G, and V_1 and V_2 are the vertex sets of the two components of $G - S$, then $S = \langle V_1, V_2\rangle$.*

6.4.2 Region faces and boundary faces

In this section, we prove that the regions of the neighborhood graph defined in Section 6.4.1 imply a partition of the vertex set $\overline{V}$ of the face graph.

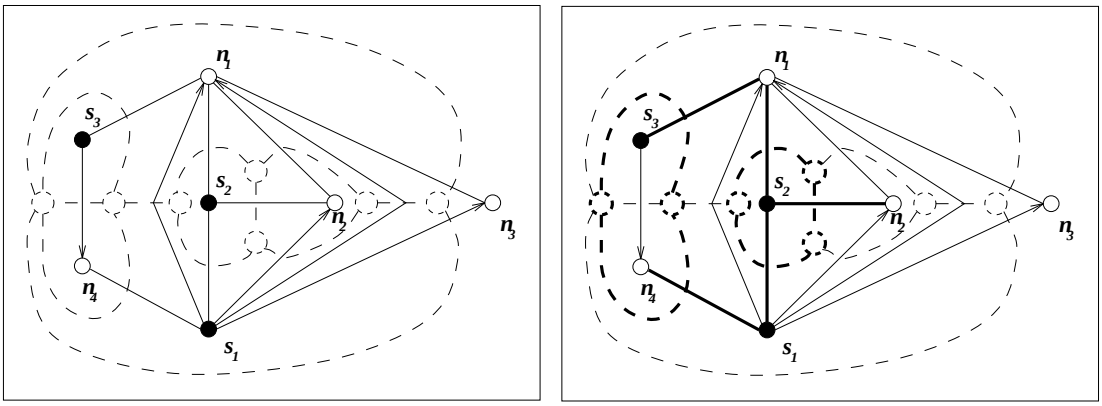
Definition 43 *Consider a neighborhood graph G and a face graph $\overline{G}$ after Step 1 of dual decimation, and a vertex $\overline{v}$ of $\overline{G}$. Consider further all edges $\overline{e_v} \in \overline{E}$ incident to $\overline{v}$, and edges e_v of the neighborhood graph corresponding to edges $\overline{e_v}$. We call vertex $\overline{v}$ a* **face of region** *$\langle V(s_i)\rangle$ or simply* **region face** *if all edges e_v are incident to vertices of one cluster $V(s_i)$ only, i.e. $e_v \in E(s_i)$. Edges incident to a region face are called* **dual region edges** *or* **region edges**.

Note that not every region of a neighborhood graph necessarily has region faces, eg. $\langle V(s_2)\rangle$, $\langle V(s_3)\rangle$ in Figure 13.

Definition 44 *Let $G(V,E)$ and $\overline{G}(\overline{V},\overline{E})$ be dual graphs and let V be partitioned into m clusters $V(s_1),\ldots,V(s_m)$ by child-parent assignments. We now consider the cut $\langle V(s_i),(V-V(s_i))\rangle$. The edges of set $\overline{\langle V(s_i),(V-V(s_i))\rangle} \subset \overline{E}$ corresponding to $\langle V(s_i),(V-V(s_i))\rangle$ form, together with their end vertices, the* **boundary** *$B(s_i)$ of region $\langle V(s_i)\rangle$. Consequently, we refer to edges on the boundary as* **boundary edges**, *and call their end vertices* **boundary faces**, *since vertices of $\overline{G}$ represent faces of G.*

Note that cut $\langle V(s_i),(V-V(s_i))\rangle$ is a cutset if $\langle V-V(s_i)\rangle$ is connected. If $\langle V-V(s_i)\rangle$ is not connected, then $\langle V(s_i),(V-V(s_i))\rangle$ contains more than one cutset. Therefore the boundary $B(s_i)$ forms one or several circuits in $\overline{G}$.

Figure 13 a shows graph G of Figure 12 a together with its dual $\overline{G}$. In Figure 13 b,



(a) Graph G and its dual $\overline{G}$.

(b) Cut $\langle V(s_1),V-V(s_1)\rangle$ and circuits on boundary $B(s_1)$ (bold).

Figure 13: Cuts and boundaries.

edges of cut $\langle V(s_1), (V - V(s_1))\rangle$ are drawn in bold, as well as the boundary $B(s_1)$ of region $\langle V(s_1)\rangle$. $B(s_1)$ consists of two circuits, since $\langle V - V(s_1)\rangle$ is not connected and contains two components, $\langle V(s_2)\rangle$ and $\langle V(s_3)\rangle$.

Theorem 11 *Consider a neighborhood graph G and a face graph $\overline{G}$ after Step 1 and 2 of dual decimation. Consider further the set $\overline{V_r} \in \overline{V}$ of region faces and the set $\overline{V_b} \in \overline{V}$ of boundary faces. Sets $\overline{V_r}$, $\overline{V_b}$ form a partition of set $\overline{V}$.*

Proof : Clusters of surviving vertices form a partition of V. Therefore all edges $e \in E$ have both end vertices either in the same cluster or in two different clusters. Edges with end vertices in two different clusters correspond to boundary edges by Definition 44. Therefore edge set $\overline{E}$ is partitioned into boundary edges and non-boundary edges. This partition of $\overline{E}$ also implies a partition of $\overline{V}$ into vertices that are end vertices of boundary edges (boundary faces by Definition 44) and those which are not.

Consider now a vertex $\overline{v_n} \in \overline{V}$ and the set $\overline{E_n} \subset \overline{E}$ of edges incident to $\overline{v_n}$. Edges of set $E_n \subset E$ corresponding to $\overline{E_n}$ form, together with their end vertices, a circuit C_n in G.

If $\overline{v_n}$ is not a boundary face, then $\overline{E_n}$ does not contain a boundary edge, and every edge of circuit C_n has both end vertices in the same cluster. This is possible only if all vertices of C_n are in the same cluster, hence all edges $e \in E_n$ have both end vertices in the same cluster, and $\overline{v_n}$ is a region face by Definition 43.

We have shown that $\overline{V}$ is partitioned into vertices that are boundary faces and vertices that are not boundary faces. We have further considered all vertices $\overline{v_n} \in \overline{V}$ that are not boundary faces, and we have shown that $\overline{v_n}$ are region faces, which proves the theorem. ■

6.4.3 Region edges and face trees

Vertices of the face graph are partitioned into region faces and boundary faces, as we have shown in the previous section. We now reveal the correspondence between self-loops of the neighborhood graph and edges incident on region faces, and show that these from particular subgraphs of the face graph.

Definition 45 *A connected graph T is a* **tree** *if every two vertices of T are joint by a unique path. We call the longest path P in T its* **trunc**, *and refer to the length of P as the* **diameter** *of T. Connected subgraphs of T that contain a vertex of P, but no edge of P, will be referred to as* **branches** *of T.*

Definition 46 *A* **bridge** *of a graph G is an edge the removal of which increases the number of components of G. If G is connected, then a bridge of G forms a one-element cutset of G.*

Theorem 12 *Consider a neighborhood graph G and a face graph $\overline{G}$ after Steps 1 and 2 of dual decimation, and set $E(s_i)$ of edges of region $\langle V(s_i)\rangle$. Then all edges $e \in E(s_i)$ are self-loops at s_i, and the edges $\overline{e} \in \overline{E(s_i)}$ form, together with their end vertices, one or several trees in $\overline{G}$.*

Proof : The end vertices of an edge (v, w) of a region $\langle V(s_i)\rangle$ before child parent identification are by definition both in cluster $V(s_i)$, $v \in V(s_i), w \in V(s_i)$. After child-parent identification by dual contraction of edges, all children of s_i are now identified with s_i, and all edges of $\langle V(s_i)\rangle$ that have not been dually contracted are self-loops at s_i.

Self-loops $e \in E(s_i)$ are circuits of length 1, hence every edge $\overline{e} \in \overline{E(s_i)}$ is a bridge, and connected subgraphs of $\overline{G}$, formed by edges $\overline{e} \in \overline{E(s_i)}$, are tree structures. ■

Definition 47 *Consider a neighborhood graph G and a face graph $\overline{G}$ after Step 1 and 2 of dual decimation. Consider further connected subgraphs of $\overline{G}$ formed by dual region edges. We refer to these subgraphs as* **face trees**.

6.4.4 Elimination of region faces and boundary faces

Dual contraction of edges in the face graph is basically different with respect to parallel processing, depending whether a region face or a boundary face is eliminated. We will show this in this section, using Lemma 4 about the degree of boundary faces.

Lemma 4 *Let G and $\overline{G}$ be a pair of graphs after Steps 1 and 2 of dual decimation. Then the degree of a boundary face $\overline{v_b}$ is greater than or equal to two.*

Proof : Boundary edges form circuits in $\overline{G}$. Therefore every boundary face $\overline{v_b}$ has at least two boundary edges incident to it, hence $d(\overline{v_b}) \geq 2$. Boundary edges are not removed during dual contraction of edges (Step 2 of dual decimation), since their corresponding edges in G connect vertices of different clusters, and not children to their parents. Hence $d(\overline{v_b}) \geq 2$ for boundary faces also after Step 2 of dual decimation. ■

Theorem 13 *Consider a neighborhood graph G and a face graph $\overline{G}$ after Step 1 and 2 of dual decimation. Consider further the elimination of a removable face $\overline{v_\varrho} \in \overline{V}$ by dual contraction of edge $\overline{e_\varrho} = (\overline{v_\varrho}, \overline{w})$, and let $\overline{w}$ be of degree 3, hence $\overline{w}$ is not removable. Then $\overline{w}$ may become removable after dual contraction of $\overline{e_\varrho}$ only if $\overline{v_\varrho}$ is a region face.*

Proof : Vertex $\overline{v_\varrho}$ may only be either a region face or a boundary face due to Theorem 11. If $\overline{v_\varrho}$ is a removable boundary face, then $d(\overline{v_\varrho}) = 2$ due to Lemma 4, hence $d(\overline{w}) = 3$ after dual contraction of $\overline{e_\varrho}$ due to Lemma 2.

Assume now that $\overline{e_\varrho}$ is a region edge, hence $\overline{v_\varrho}$ is a region face end edge e_ϱ corresponding to $\overline{e_\varrho}$ is a self-loop at a survivor s_i. Then $\overline{e_\varrho}$ is a bridge and may therefore be the only edge incident to $\overline{v_\varrho}$, hence $d(\overline{v_\varrho}) = 1$, and $d(\overline{w}) = 2$ after dual contraction of $\overline{e_\varrho}$ due to Lemma 2.

We have distinguished vertices of the face graph that are either region faces or boundary faces. We have shown that the removal of a boundary face by dual contraction in the face graph cannot create a new removable face. We have further shown that the removal of a region face by dual contraction in the face graph can create a new removable face. This proves the theorem since a vertex of the face graph may only be either a region face or a boundary face. ■

Theorem 14 *Consider a neighborhood graph G and a face graph $\overline{G}$ after Step 1 and 2 of dual decimation. Consider further the elimination of a removable face $\overline{v_\varrho} \in \overline{V}$ by dual contraction of edge $\overline{e_\varrho} = (\overline{v_\varrho}, \overline{w})$. Then edge $e_\varrho \in E$ corresponding to $\overline{e_\varrho}$ is a self-loop if and only if $\overline{v_\varrho}$ is a region face. Edge $e_\varrho \in E$ corresponding to $\overline{e_\varrho}$ is one of two parallel edges if and only if $\overline{v_\varrho}$ is a boundary face.*

Proof : Edges incident to a region face are region edges by Definition 43. The correspondence between region edges and self-loops after Step 2 of dual decimation has been stated in Theorem 12, hence self-loops are eliminated from G if region faces are removed by dual contraction operations in $\overline{G}$.

Assume now that $\overline{e_\varrho}$ is a region edge and $\overline{v_\varrho}$ is not a region face, hence a boundary face due to Theorem 11. Then $d(\overline{v_\varrho}) \geq 3$ since a boundary face has at least two boundary edges incident to it, cf. Lemma 4. The assumption that e_ϱ be a self-loop and $\overline{v_\varrho}$ be a removable boundary face leads to a contradiction which proves that self-loops are eliminated from G only if region faces are removed by dual contraction operations in $\overline{G}$.

If edge $\overline{e_\varrho}$ is a boundary edge, then it corresponds to one of two parallel edges in G as stated in Theorem 5, and its end vertices are both boundary faces due to Definition 44. Therefore parallel edges are eliminated from G if boundary faces are removed by dual contraction in $\overline{G}$.

Region faces cannot be end vertices of boundary edges due to Definition 44, hence parallel edges are eliminated from G only if boundary faces are removed by dual contraction in $\overline{G}$. ■

6.4.5 Self-loop removal is sequential but local

The most important result of Section 6.4.4 is that the removal of a self-loop in G is equivalent to the removal of a region face by dual contraction in $\overline{G}$, and that only the removal of a region face may create a new removable face. Face contraction is consequently a sequential process since the removability of a face (region *or* boundary face) may depend on the removal of an adjacent region face.

We consider now the faces the removability of which can be affected by the removal of a region face $\overline{v_i}$. Since these faces must be adjacent to $\overline{v_i}$, we state Theorem 15 about the neighborhood relations of a region face.

Theorem 15 *Let $\langle V(s_i) \rangle$ be a region of a neighborhood graph G, and let $B(s_i)$ be its boundary. Then a face $\overline{v_i}$ of $\langle V(s_i) \rangle$ can be adjacent in $\overline{G}$ only to another face of $\langle V(s_i) \rangle$ or to a face of $B(s_i)$.*

Proof : Consider $\overline{v_i}$ and an adjacent face $\overline{w}$. If edge $\overline{e_i} = (\overline{v_i}, \overline{w}) \in \overline{E}$, then edge $e_i = \overline{(\overline{v_i}, \overline{w})} \in E$. Also, both end vertices of e_i are in cluster $V(s_i)$ since $\overline{v_i}$ is a face of region $\langle V(s_i) \rangle$, cf. Definition 43. Since the set $\overline{V_r}$ of region faces and the set $\overline{V_b}$ of boundary faces form a partition of $\overline{V}$, $\overline{w}$ can only be one of the following:

1. a face of region $\langle V(s_i) \rangle$;

2. a face of boundary $B(s_i)$;

3. a face of region $\langle V(s_j)\rangle; j \neq i$;

4. a face of boundary $B(s_j); j \neq i$.

ad 1, 2: For an example see Figure 14. Region $\langle V(s_i)\rangle$ has two faces $\overline{u}$, $\overline{w}$. Boundary

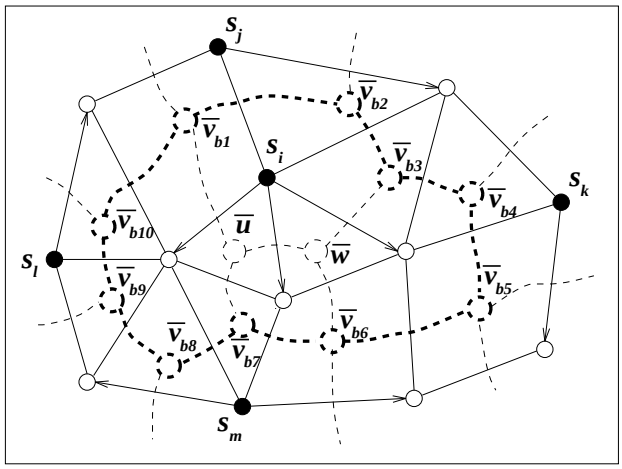


Figure 14: Neighborhood relations of a face of region $\langle V(s_i)\rangle$.

$B(s_i)$ is outlined in bold. Region face $\overline{w}$ is adjacent to a face of the same region, $(\overline{u}, \overline{w}) \in \overline{E}$, as well as to faces $\overline{v_{b3}}$ and $\overline{v_{b6}}$ of $B(s_i)$.

For the discussion of Cases 3 and 4, we consider the set $\overline{E_w} \subseteq \overline{E}$ of edges incident to $\overline{w}$. Edges of set $E_w \subseteq E$ corresponding to $\overline{E_w}$ form, together with their end vertices, a circuit C_w in G.

ad 3: all vertices of C_w are in $V(s_j)$, hence both end vertices of e_i are in both $V(s_i)$ and $V(s_j)$ which is a contradiction since clusters form a partition of V;

ad 4: none of the vertices of C_w is in $V(s_i)$, otherwise $\overline{w}$ would be a face of $B(s_i)$ or $\langle V(s_i)\rangle$; hence both end vertices of e_i are both in $V(s_i)$ and not in $V(s_i)$ which is a contradiction.

We have listed all possibilities of vertex $\overline{w}$ belonging to regions or boundaries. Figure 14 displays an example for the assignment of $\overline{w}$ to either $\langle V(s_i)\rangle$ or $B(s_i)$. We have further shown that the assignment of $\overline{w}$ to either $\langle V(s_j)\rangle$ or $B(s_j)$; $j \neq i$, leads to a contradiction, which proofs the theorem. ■

Theorem 16 *Consider a neighborhood graph G and a face graph $\overline{G}$ after Step 1 and 2 of dual decimation, with self-loops at surviving vertices. Then the elimination of self-loops by dual contraction of region edges can be executed independently in each region.*

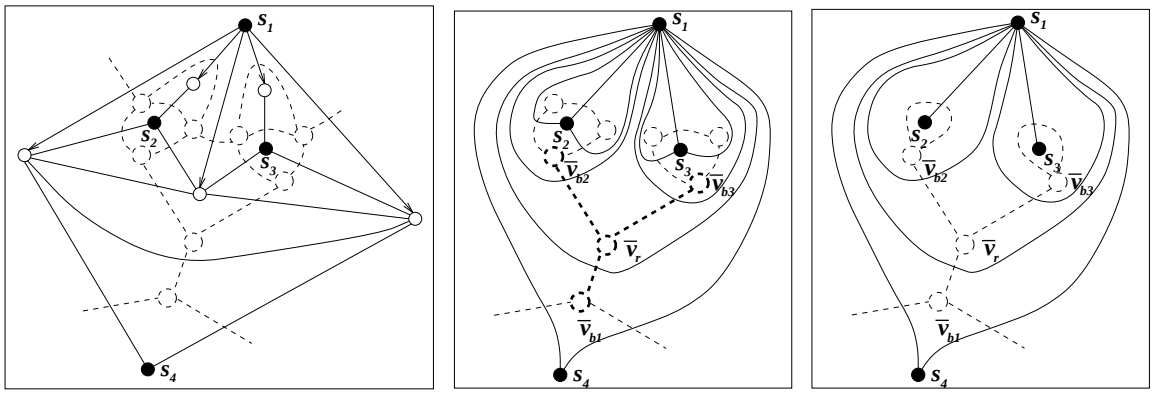
Proof : We proof the theorem in that we show that the elimination of face $\overline{v_i}$ of a region $\langle V(s_i)\rangle$ by dual contraction of a dual region edge can cause new removable faces only in $\langle V(s_i)\rangle$ or in $B(s_i)$.

Face $\overline{v_i}$ can be adjacent to faces of $\langle V(s_i)\rangle$ and its boundary $B(s_i)$ only, as stated in Theorem 15. Out of the faces neighboring to $\overline{v_i}$, only boundary faces are adjacent to faces of other regions. The removal of boundary faces, however, cannot cause new removable faces due to Theorem 13. Therefore the removal of faces in region $\langle V(s_i)\rangle$ cannot cause removable faces in any other region of G. ■

6.4.6 Complexity of self-loop elimination

Although the elimination of self-loops has a sequential character to it, it can be executed independently for every region of the neighborhood graph. We are now interested in an upper bound for the time complexity of self-loop elimination, but we should not start our considerations without pointing out that redundant edges of a neighborhood graph cannot always be eliminated completely.

Figure 15 a shows the example of a region in which self-loops remain after face contraction. Region $\langle V(s_1)\rangle$ encloses regions $\langle V(s_2)\rangle$ and $\langle V(s_3)\rangle$. The boundaries of the enclosed



(a) Dual graphs before dual contraction.

(b) Result after dual contraction.

(c) Result after face contraction.

Figure 15: Example graph with remaining edge redundancies.

regions are connected to the rest of the face graph via edges $\overline{e} \in \overline{E(s_1)}$ corresponding to edges $e \in E(s_1)$.

After Step 1 of dual decimation, the face tree of $\langle V(s_1)\rangle$ contains paths P_b connecting different boundary faces $\overline{v_{b1}}$, $\overline{v_{b2}}$, $\overline{v_{b3}}$, (outlined in bold in Figure 15 b). Paths P_b are not

eliminated because their end vertices are not of degree 1. Edges incident to region face $\overline{v_r}$ connect $\overline{v_r}$ to three different boundary faces. Face $\overline{v_r}$ has therefore degree 3 and does not become removable during face contraction (Figure 15 c).

Theorem 17 *Let $G(V,E)$ and $\overline{G}(\overline{V},\overline{E})$ be dual graphs, and let $D(G)$ denote the maximum degree of a vertex in G. Consider both graphs after Steps 1 and 2 of dual decimation, and let a subset of E be self-loops at surviving vertices. Then all self-loops that can be eliminated during face contraction are eliminated after $O(log(D(G))^2)$ parallel processing steps.*

Proof : Edges $\overline{e_s} \in \overline{E}$ corresponding to self-loops $e_s \in E$ at one surviving vertex s_i form, together with their end vertices, local tree structures in $\overline{G}$. Dual contraction of edges $\overline{e_s}$ is independent of dual contraction of any other edge $\overline{e}$ corresponding to a self-loop at a survivor other than s_i. We first give an upper bound for the number k_s of self-loops at a single survivor. Only edges having no end vertex other than s_i or children of s_i can be self-loops at s_i after dual contraction of edges. An upper bound for k_s is therefore the maximum number of edges incident to the neighbors of a vertex $v \in V$, hence $O(D(G)^2)$.

The second step of the proof is to give an upper bound for the number of parallel processing steps to eliminate all removable faces of region $\langle V(s_i)\rangle$. Therefore we consider a tree T of diameter l formed by edges $\overline{e_s}$ and the trunc P of T. Every vertex in P may be a vertex of one or several branches of T. We distinguish two phases of the elimination of self-loops, the elimination of removable faces of branches and the elimination of removable faces of P.

After the elimination of removable faces in branches, a subset of vertices of P are removable faces. Adjacent removable faces define one or several subsequences of P. These subpaths can be removed in parallel by face contraction. The time for the elimination of all subpaths is thus determined by the length l_s of the longest subpath, an upper bound for l_s is l.

Steps 1 through 3 of face contraction eliminate between $\frac{1}{2}$ and $\frac{2}{3}$ of the removable faces in each subpath, hence $log_2(l)+1$ is an upper bound for the number of parallel processing steps to eliminate all removable faces in P.

Let $T_{\frac{\alpha}{2}}$ denote a largest branch of a tree of diameter α. Then $T_{\frac{l}{2}}$ containing a vertex in P determines the maximum time for the elimination of removable faces in *all* branches connected to P. The largest branch of a $T_{\frac{l}{2}}$ is a $T_{\frac{l}{4}}$ and so on until a T_2 with a largest branch T_1 of diameter 1.

We assume that T_1 has a pendant vertex that needs one step to be eliminated. We further assume that every branch waits until all removable faces in its largest branch are eliminated, before edges of its trunc are dually contracted. Then all removable faces of $\langle V(s_i)\rangle$ are eliminated by face contraction after

$$1+2+3+\ldots+(log_2(l)+1) = \frac{(log_2(l)+1)*(log_2(l)+2)}{2} \quad (1)$$

$$= \frac{log_2(l)^2+3*log_2(l)+2}{2} \quad (2)$$

$$= O(log_2(l)^2) \quad (3)$$

processing steps. To obtain the final result, we assume that

$$l = k_s = D(G)^2, \tag{4}$$

hence

$$\begin{aligned} log_2(D(G)^2)^2 &= 4 * log_2(D(G))^2 & (5)\\ &= O(log(D(G))^2) & (6) \end{aligned}$$

for the time complexity of the elimination of self-loops in G. ■

6.4.7 Complexity of the elimination of parallel edges

Theorem 18 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be dual graphs, and let $D(G)$ denote the maximum degree of a vertex in G. Consider both graphs after Steps 1 and 2 of dual decimation, and let a subset of E be parallel edges connecting surviving vertices. Then all parallel edges that can be eliminated during face contraction are eliminated after $O(log(D(G)))$ parallel processing steps.*

Proof : Edges $\overline{e_p} \in \overline{E}$ corresponding to parallel edges $e_p \in E$ between surviving vertices form, together with their end vertices, paths or circuits in $\overline{G}$. We first give an upper bound for the number k_p of parallel edges connecting two survivors. We assume that all neighbors of a survivor s_1 are its children, and that s_1 is connected to only one other survivor s_2 after dual contraction of edges. An upper bound for k_p is therefore the maximum number of edges incident to the neighbors of a vertex $v \in V$, hence $D(G)^2$.

We now assume that all k_p end vertices of edges $\overline{(s_1, s_2)}$ except one are removable faces. The number of parallel processing steps to eliminate adjacent removable faces has already been calculated in the proof of Theorem 17. In analogy to this proof, we can immediately state that the elimination of parallel edges by face contraction is executed in $O(log(D(G)^2)) = O(log(D(G))$ parallel processing steps. ■

6.4.8 Conclusion: Complexity of face contraction

We conclude by stating Theorem 19 about the upper bound to the number of parallel processing steps needed for face contraction.

Theorem 19 *Let $G(V, E)$ and $\overline{G}(\overline{V}, \overline{E})$ be dual graphs, and let $D(G)$ denote the maximum degree of a vertex in G. Consider both graphs after Steps 1 and 2 of dual decimation. Then face contraction terminates after $O(log(D(G))^2)$ parallel processing steps.*

Proof : The elimination of self-loops due to the contraction of dual region edges may create new removable faces. A worst case assumption is therefore that no boundary face is removable until all removable faces have been eliminated during face contraction.

With the results of Theorems 17 and 18 we obtain the following estimation for the time complexity of face contraction:

$$O(log(D(G))^2) + O(log(D(G))) = O(log(D(G))^2).$$

■

7 Conclusion

In this paper we have presented an algorithm that simplifies a discrete representation of a two dimensional space. The algorithm is based on local operations tht can be executed in parallel by independently operating processing elements. Applied recursively, the algorithm can build an irregular, potentially adaptive multiresolution representation of the space.

As a conclusion we would like to point out a perspective for an extension of the scope of the algorithm to three and higher dimensions. We recall that we select a vertex subset of a connected graph G, to which we refer as the neighborhood graph. Vertices of the dual of G, $\overline{G}$, represent two dimensional entities, i.e. the faces of the planar embedding of G. These representatives are eliminated during face contraction as a consequence of preliminary dual contractions of edges in the neighborhood graph.

We now consider G and the faces of its planar embedding, in literature referred to as the plane map of G [9, p. 103]. In addition to elements of dimension 0 (vertices) and 1 (edges), a plane map also contains faces, as space elements of dimension 2. Sets of space elements of arbitrary dimension are referred to as cellular or simplicial complexes in classical topology [20], founded by Listing [16, loc. cit. [6]]. Dehn and Heegaard [6] referred to classical topology as *analysis situs*, and provided a system of axioms for the analysis of three dimensional spaces. Recently, Kovalevsky propagated cellular complexes for use in the field of image analysis [12, 13]. A cellular complex of dimension n is a set of abstract space elements of dimension $0, \ldots, n$, where elements of dimension k are bounded by elements of dimension $0, \ldots, k-1$, $k = 1, \ldots, n$. Hence, 0- and 1-dimensional space elements are bounding elements of all other dimensions $2, \ldots, n$.

This homogeneous representation provides a perspective to generalize dual decimation to obtain a process that can derive an irregular multiresolution representation of a $3D$ discrete space.

References

[1] Claude Berge. *Graphs.* North-Holland, 3rd edition, 1991.

[2] Etienne Bertin. *Diagrammes de Voronoï 2D and 3D: Applications en Analyse d'Images.* PhD thesis, Université Joseph Fourier - Grenoble I, Laboratoire TIMC, B.P. 53 X, F-38041 Grenoble CEDEX, France, 1994.

[3] M. Bister, J. Cornelis, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, 11(9):605–617, September 1990.

[4] Virginio Cantoni and Stefano Levialdi, editors. *Pyramidal Systems for Computer Vision.* Springer Verlag, 1986.

[5] Nicos Christofides. *Graph Theory - An Algorithmic Approach.* Academic Press, New York, London, San Francisco, 1975.

[6] M. Dehn and P. Heegaard. Analysis situs. In *Enzyklopädie der Mathematischen Wissenschaften, Bd. IIIAB3*, pages 154–220, January 1907.

[7] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[8] Robert M. Haralick and Linda G. Shapiro. Glossary of computer vision terms. *Pattern Recognition*, 24(1):69–93, 1991.

[9] F. Harary. *Graph Theory*. Addison-Wesley, Reading, Mass., 1972.

[10] Jean-Michel Jolion and Annick Montanvert. The adaptive pyramid, a framework for 2D image analysis. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 55(3):339–348, May 1992.

[11] Jean-Michel Jolion and Azriel Rosenfeld. *A Pyramidal Framework for Early Vision*. Kluwer Academic Publishers, Dordrecht / Boston / London, 1994.

[12] Vladimir A. Kovalevsky. Finite Topology as Applied to Image Analysis. *Computer Vision, Graphics and Image Processing*, 46:141–161, 1989.

[13] Vladimir A. Kovalevsky. Digital Geometry Based on the Topology of Abstract Cell Complexes. In Jean-Marc Chassery, Jean Françon, Annick Montanvert, and Jean-Pierre Réveillès, editors, *Géometrie Discrète en Imagery, Fondements et Applications*, pages 259–284, Strasbourg, September 1993.

[14] Walter G. Kropatsch and Annick Montanvert. Irregular versus regular pyramid structures. In U. Eckhardt, A. Hübler, W. Nagel, and G. Werner, editors, *Geometrical Problems of Image Processing*, pages 11–22, Georgenthal, Germany, March 1991. Akademie Verlag, Berlin.

[15] Walter G. Kropatsch, Christian Reither, Dieter Willersinn, and Guenther Wlaschitz. The dual irregular pyramid. In *5th International Conference CAIP'93 on Computer Analysis of Images and Patterns*, pages 31–41, Budapest, Hungary, September 1993.

[16] J. B. Listing. Vorstudien zur Topologie. *Göttinger Studien*, 1847.

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

[18] Annick Montanvert, Peter Meer, and Azriel Rosenfeld. Hierarchical image analysis using irregular tesselations. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 13(4):307–316, April 1991.

[19] Azriel Rosenfeld, editor. *Multiresolution Image Processing and Analysis*. Springer Verlag, 1984.

[20] John Stillwell. *Classical Topology and Combinatorial Group Theory*. Springer-Verlag, New York, 2nd edition, 1993.

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

[22] J. K. Tsotsos. Analyzing Vision at the Complexity Level. *Behavioral and Brain Sciences*, 13(3):423–469, 1990.

[23] Leonard Uhr, editor. *Parallel Computer Vision.* Academic Press, Inc., 1987.

[24] H. Whitney. The coloring of graphs. *Ann. Math.*, 2(33):688–718, 1932.

[25] H. Whitney. Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, (54):150–168, 1932.

[26] H. Whitney. Non-separable and planar graphs. *Trans. Amer. Math. Soc.*, (34):339–362, 1932.

[27] Dieter Willersinn, Etienne Bertin, and Walter G. Kropatsch. Dual Irregular Voronoi Pyramids and Segmentation. Technical Report PRIP-TR-027, PRIP, TU Wien, 1994.

[28] Dieter Willersinn and Walter G. Kropatsch. Dual Graph Contraction for Irregular Pyramids. In 12^{th} *International Conference on Pattern Recognition (in press)*, Jerusalem, October 1994.