Equivalent Contraction Kernels Using Dynamic Trees [⋆]

R. Glantz, Y. Haxhimusa, M. Saib, W.G. Kropatsch

Pattern Recognition and Image Processing Group 183/2
Institute for Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9
A-1040 Vienna
Austria
e-mail:{glz,yll,saib,krw}@prip.tuwien.ac.at
phone ++43-(0)1-58801-18351
fax ++43-(0)-1-58801-1839

Abstract. itGraphs are useful tools for modeling problems that occur in a variety of fields. In machine vision graph based solutions have been successfully applied to many image processing problems e.g. region adjacency graphs for segmentation. Dual graph contraction (DGC) reduces the number of vertices and edges of a pair of dual image graphs while, at the same time, the topological relations among the 'surviving' components are preserved. Repeated application produces a stack of successively smaller graphs. The process is controlled by selected decimation parameters which consist of a subset of surviving vertices and associated contraction kernels. Equivalent contraction kernels (ECKs) combine two or more contraction kernels into one single contraction kernel which generates the same result in one single dual contraction. We present an implementation under LEDA (Library of Efficient Data structures and Algorithms) of ECKs using Dynamic Trees. In the first section we describe ECKs and in the second an implementation of ECKs under LEDA is given.

1 Equivalent Contraction Kernels

Dual graph contraction is the basic process [2] that builds an irregular 'graph' pyramid by successively contracting a dual image graph of one level into the smaller dual image graph of the next level. First we repeat the basic terms of dual graph contraction as presented in [3]. Dual graph contraction proceeds in two basic steps: dual edge contraction and dual face contraction. The base of the pyramid consists of the dual image graphs $(G_0, \overline{G_0})$. The following *decimation parameters* $(S_i, N_{i,i+1})$ determine the structure of an irregular pyramid [2][Def.5]:

⋆ This paper has been supported by the Austrian Science Fund (FWF) under grant P14445-MAT.

a subset of *surviving vertices* $S_i = V_{i+1} \subset V_i$, and a subset of *primary non-surviving edges*[1] $N_{i,i+1} \subset E_i$. Every non-surviving vertex, $v \in V_i \setminus S_i$, must be connected to one surviving vertex in a unique way. The relation between the two pairs of dual graphs, $(G_i, \overline{G_i})$ and $(G_{i+1}, \overline{G_{i+1}})$, as established by dual graph contraction with decimation parameters $(S_i, N_{i,i+1})$ is expressed by function $C[.,.]$:

$$(G_i, \overline{G_i}) = C[(G_i, \overline{G_i}), (S_i, N_{i,i+1})]. \tag{1}$$

The contraction of a primary non-surviving edge consists in the identification of its endpoints and the removal of both the contracted edge and its dual edge. Dual face contraction simplifies most of the multiple edges and self-loops, but not those including any surviving parts of the graph (see [2]). One step of the dual graph contraction is illustrated in Fig. 1. To define the parameters that control the process of dual graph contraction we observe that the subgraphs in our example graph (Fig. 1) form small tree structures $T(s)$ that collapse into surviving vertex s of the contracted graph. $T(s)$ is *spanning tree* of the connected component of the surviving root vertex, or equivalently, (V, N) is a spanning forest of the graph $G(V, E)$.

Definition 1. *A decimation of a graph $G(V, E)$ is specified by a selection of the surviving vertices $S \subset V$ and selection of primary non-surviving edges $N \subset E$ such that following two conditions are fulfilled:*

1. *Graph (V, N) is a spanning forest of graph $G(V, E)$.*
2. *The surviving vertices $S \subset V$ are the roots of the forest (V, N)*

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

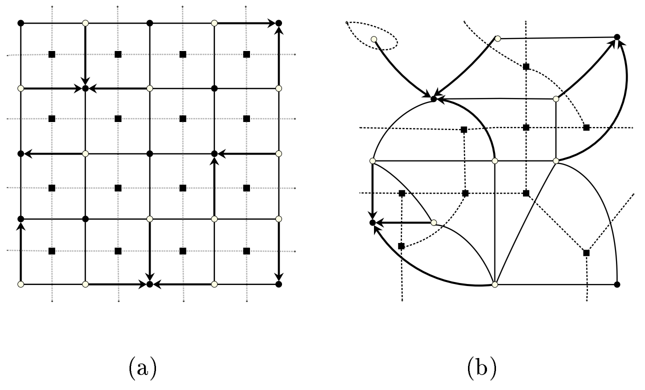
The connectivity of the contracted graph is established by paths connecting two surviving vertices:

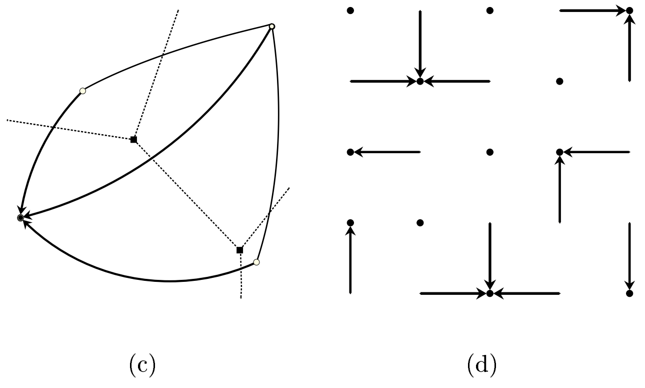
Definition 2. *Let $G(V, N)$ be a graph with decimation parameters (S, N). A path in $G(V, N)$ is called a connecting path between two surviving vertices $v, w \in S$, denoted $CP(v, w)$, if it consists of the three subsets of edges E (Fig. 2):*

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

Connected paths $CP(v, w)$ in $G(V, E)$ are strongly related to the edges in the contracted graph $G'(V', E')$: Two different surviving vertices that are connected by a contracted path in G are connected by an edge in E'. For every edge $e' = (v, w) \in E'$ there exists a connecting path $CP(v, w)$ in G.

[1] Secondary non-surviving edges are removed during dual face contraction.





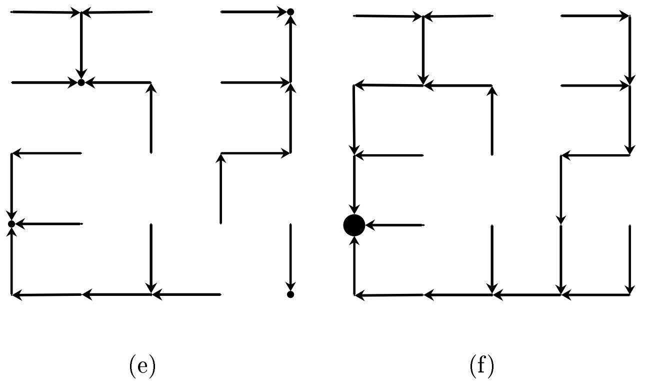


Fig. 1. Example of dual irregular pyramid: **(a)** $(G_0, \overline{G_0})$, **(b)** $(G_1, \overline{G_1}) = C[(G_0, \overline{G_0}), (S_1, N_{0,1})]$, **(c)** $(G_2, \overline{G_2}) = C[(G_0, \overline{G_0}), (S_2, N_{0,2})$; and equivalent contraction kernel: **(d)** $(S_0, N_{0,1})$, **(e)** $(S_2, N_{0,2})$, **(f)** $(S_3, N_{0,3})$ and $(G_3, \overline{G_3}) = (\{\bullet\}, \{\}) = C[(G_0, \overline{G_0}), (S_3, N_{0,3})]$

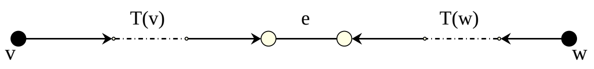


Fig. 2. Decomposition of connecting path $CP(v, w)$

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

Fig. 3. Equivalent contraction kernel

The combination of two (and more) successive reductions in an equivalent weighting function allowed Burt [1] to calculate any level of the pyramid directly from th base. Similarly we combine two (and more) dual graph contractions (see Fig. 3) of graph G_k with decimation parameters $(S_k, N_{k,k+1})$ and $(S_{k+1}, N_{k+1,k+2})$ into one single equivalent contraction kernel $N_{k,k+2} = N_{k,k+1} \circ N_{k+1,k+2}$ (for simplicity G_i stands for $(G_i, \overline{G_i})$:

$$C[C[G_k, (S_k, N_{k.k+1})], (S_{k+1}, N_{k+1,k+2}] = C[G_k, (S_{k+1}, N_{k,k+2}] = G_{k+2}. \quad (2)$$

Definition 3. *$E_{k+1} \mapsto E_k$ assigns to each edge $e_{k+1} = (v_{k+1}, wk + 1) \in E_{k+1}$ one of the bridges $e_k \in E_k$ of the connecting paths $CP_k(v_{k+1}, w_{k+1})$:*

$$bridge(e_{k+1}) := e_k. \quad (3)$$

Equivalent contraction kernels (see Fig.1) are constructed in this way:

1. Assume that the dual irregular pyramid $((G_0, \overline{G_0}), \ldots, (G_{k+2}, \overline{G_{k+2}})), k > 1$ is the result of $k + 2$ dual graph contractions. The structure of G_{k+2} is fully determined by the structure of G_{k+1} and the decimation parameters $S_{k+1}, N_{k+1,k+2}$.
2. Furthermore, the structure of G_{k+1} is determined by G_k and the decimation parameters $S_k, N_{k,k+1}$. $S_{k+1} = V_{k+2}$ are the vertices surviving from G_k to G_{k+2}. The searched contraction kernels must be formed by edges $N_{k,k+2} \subset E_k$. This is true for $N_{k,k+1}$ but not for $N_{k+1,k+2} \subset E_{k+1}$, if we would simply overlay the two sets of decimation parameters. An edge $e_{k+1} = (v_{k+1}, w_{k+1}) \in N_{k+1,k+2}$ corresponds to a connecting path $CP_k(v_{k+1}, w_{k+1})$ in G_k. (If there are more than one connecting paths, one must be selected). By definition 2, $CP_k(v_{k+1}, w_{k+2})$ consists of one branch of $T_k(v_{k+1})$, one branch of $T_k(w_{k+1})$, and one surviving edge $e_k \in E_k$ connecting two contraction kernels $T_k(v_{k+1})$, $T_k(w_{k+1})$

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

$$N_{k,k+2} = N_{k,k+1} \cup \bigcup_{e_{k+1} \in N_{k+1,k+2}} bridge(e_{k+1}). \tag{4}$$

3. The above process can be repeated on the remaining contraction kernel until the base level 0 contracts in one step onto the apex $V_n = \{v_n\}$. The edge of the corresponding spanning tree are contained in $N_{0,n}$ ($N_{0,3}$ in Fig. 1).

2 Implementation of Equivalent Contraction Kernels Using Dynamic Trees

LEDA(Library for Efficient Data and Algorithms) [6] is a C++ library of combinatorial and geometric data types and algorithms implementing many abstract data types e.g. trees, graphs, and lists. LEDA includes iterators like "forall-adj-node" and "forall-edges" as well as basic operations like "delete node" and "delete edge". An instance G of the data type *graph* consists of a list V of nodes and a list E of edges, where *node* and *edge* are LEDA classes. A pair of nodes $(v, w) \in (V \times V)$ is associated with every edge $e \in E$, v is called the *source* of e, w the target of e, and v and w are the endpoints of e. DGC (Dual Graph Contraction) has been implemented using templates so that the process can be applied to a wide variety of attributed graphs [4]. In [4], to reach the higher levels of the image pyramid $(G_n, \overline{G_n})$, the dual graph contraction process has to be iteratively repeated form the base of the image pyramid $(G_0, \overline{G_0})$ until the level below the higher level $(G_{n-1}, \overline{G}_{n-1})$. But in these processes we lose the ECKs. Using dynamic trees under LEDA allows us to built ECKs, which are useful in extracting the higher levels of the image pyramid directly from the base level in one step. Knowing the ECKs we can determine the dual irregular pyramid completely [2]. An instance D of the data type "dynamic-tree" is a set of dynamically changing rooted trees. Each edge is directed towards the root and has a weight. Optionally, user defined information can be stored at the vertices and the edges. Dynamic Trees have operations like `evert(vertex` v`)` that makes v the new root of its tree, and the operation `link(vertex` v`, vertex` w`, double x, void* e-inf=nil)` that links the tree v to the vertex w in a different tree. The edge $e = (v, w)$ gets a weight x, and the additional user defined information $e - inf$ is stored at e. Equivalent Contraction Kernels has been implemented using the data type dynamic trees. In our implementation the following parts are used:

Contraction of edge in dual graph

```
old-root = root-target-vertex;
dsf.evert(source-vertex);
dsf.evert(target-vertex);
dsf.link(target-vertex, source-vertex, 0.0, nil);
```

```
   dsf.evert(old-root);
```

In this part we use an instance dsf (dynamic spanning forest) of the dynamic trees. In the code above the vertex "root-target-vertex" is assigned to the "old-root" vertex, then the vertex "source-vertex" is made as new root of its tree and the vertex "target-vertex" is made as new root of its tree too. After that, the tree root "target-vertex" is linked to the tree root "source-vertex" with the operation `dsf.link(target-vert, source-vertex,0.0,nil)` creating new edge `e = (target-vertex, source-vertex)`.

Deletion of dual edges in graph g

```
   dual-eh = dg2g[eh];
   rev-dual-eh = g.reversal(dual-eh);
   g.del-edge(dual-eh);
   g.del-edge(rev-dual-eh);
```

In this part we make the edge "eh" as dual edge from the edge list of dual graph, we call it "dual-h". After that the reversal edge of the "dual-eh" is found in the graph g using the method **reversal** that exists in the class graph, we call the reversal edge "rev-dual-eh". These two edges, "dual-eh" and "rev-dual-eh" are deleted from the graph g with the method **del-edge**, which exists in the class graph. Dynamic Trees use binary trees with randomized balancing scheme. Each operation takes $O(\log^2 n)$ amortized expected time for make which takes constant time. n is the current number of nodes [7].

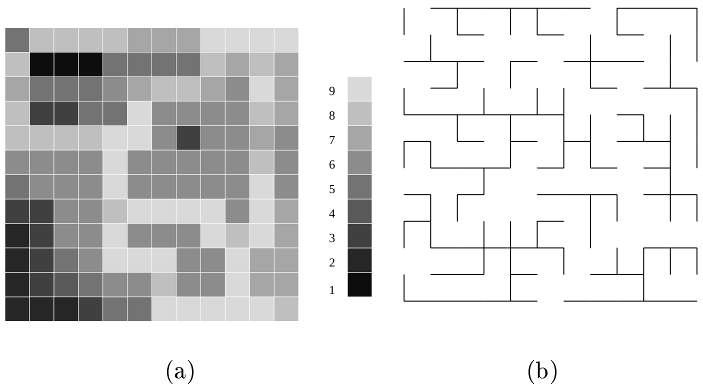


Fig. 4. Example of DGC using dynamic tree: **(a)** Original picture [5]. **(b)** ECK of dynamic tree of level 3.

3 Conclusions

In dual graph contraction, the decimation parameters control the process that iteratively builds an irregular (graph) pyramid. To specify these parameters the concept of contraction kernel was introduced. Dynamic trees permit us to build ECKs which allow to skip the construction of intermediate pyramid levels. If we know ECKs we can determine the dual irregular pyramid completely from the base level. In the previous version of DGC [4] binary labels identifying the contraction kernel were distributed among vertices and edges. The new version of DGC uses Dynamic Trees which are a separate entity for storage and reconstruction of ECKs. The dynamic trees do not change the complexity of the DGC algorithm. Dynamic Trees use binary trees with randomized balancing scheme. Each operation takes $O(\log^2 n)$ amortized expected time for building the data structure which takes constant time. n is the current number of nodes [7].

References

1. Burt, P.J.,Adelson, E.Huthor B.: The Laplacian pyramid as a compact image code.*IEEE Trans.* Communic.31:pp.523-540, 1983.
2. Kropatsch, W.G.: Building Irregular Pyramids by Dual Graph Contraction.*IEE-Proc. Vision, Image and Signal Processing* Vol.142(No.6):pp.366-347, December 1995. Vol. 99, No. 1, pp.9-99, June 1994.
3. Kropatsch, W.G.: From Equivalent Weighting Functions To Equivalent Contraction Kernels. Czeck Pattern Recognition Workshop 1997.
4. Kropatsch, W.G., Burge, M., Ben Yacoub, S., Selmaoui, N.: Dual Graph Contraction with LEDA. 1997
5. Glantz, R., Kropatsch W.G.: Plane Embedding of Dual Contracted Graphs.*DGCI 2000 Proceedings* Vol.1953:pp.348-357, Uppsala, Sweden.
6. Mehlhorn, K., Naeher, S.: *LEDA:A Platform for Combinatorial and Geomertric Computing.* Cambrige University Press, Cambrige U.K., 1999.
7. Melhorn, K., Naeher, S., Seel, M., Uhrig, Ch.: The LEDA User Manual. Version 4.2