Integral Trees: Subtree Depth and Diameter

Walter G. Kropatsch[1], Yll Haxhimusa[1], and Zygmunt Pizlo[2]

[1,⋆] Vienna University of Technology, Institute of Computer Aided Automation, PRIP 183/2, Favoritenstr, 9, A-1040 Vienna, Austria
{krw, yll}@prip.tuwien.ac.at

[2,⋆⋆] Department of Psychological Sciences, Purdue University, West Lafayette, IN 47907-1364
pizlo@psych.purdue.edu

Abstract. Regions in an image graph can be described by their spanning tree. A graph pyramid is a stack of image graphs at different granularities. Integral features capture important properties of these regions and the associated trees. We compute the depth of a rooted tree, its diameter and the center which becomes the root in the top-down decomposition of a region. The integral tree is an intermediate representation labeling each vertex of the tree with the integral feature(s) of the subtree. Parallel algorithms efficiently compute the integral trees for subtree depth and diameter enabling local decisions with global validity in subsequent top-down processes.

1 Introduction

Viola and Jones introduced the 'Integral Image' [1] as an intermediate representation for the image to compute rapidly rectangular features. Each pixel of an integral image stores the sum of values of a window defined by the left upper image corner and the pixel in the lower right corner. The computation of the integral image is linear and the computation of the sum of any rectangular window uses only four pixels of the integral image. Its effectiveness has been demonstrated in people tracking [2]. Rotated windows and articulated movements of arms and legs cause still problems. We follow the strategy to adapt the data structure to the data and compute features on the adapted structures.

On a graph, vertices take the role of pixels in images. Image graphs are embedded in the plane and can take many different forms: the vertices of the 'neighborhood graph' correspond to pixels and are connected by edges if the corresponding pixels are neighbors. In the 'region-adjacency-graph' vertices correspond to regions in the image and edges connect two vertices if the two corresponding regions share a common boundary. Graphs of different granularity can

⋆ Supported by the FWF under grants P14445-MAT, P14662-INF and FSP-S9103-N04.

⋆⋆ Supported by ECVision and by the Air Force Office of Scientific Research.

be related through the concept of dual graph contraction [3] giving rise to graph pyramids representing the regions of an image at multiple resolutions.

We start by further motivating the research by similar problems and solutions in the $k-$traveling salesperson problem and other visual problems. Section 2 transfers the classical parallel algorithm for computing the distance transform of a discrete binary shape from the discrete grid to the plane graph G. We then formulate an algorithm which computes a spanning tree of a given shape by successively removing edges that connect a foreground face with the background (section 3). This is similar to the distance transform and to well-known shrinking and thinning algorithms. However, in contrast to those algorithms, the goal is not to prune the branches of a skeleton of the shape but to determine its 'internal structure'. This internal structure is used in section 4 to determine the diameter and the center of the spanning tree. The diameter of a graph is the longest among the shortest paths between any pair of vertices. Its determination involves the search for the shortest path between any pair of vertices. This is much less complex if the graph is a tree. This is one of the reasons why we first search for a tree spanning the graph and find then the diameter of this tree. Partly as a by-product we compute the maximal path lengths of all branches of the subtrees and the respective diameters (section 5.1). These 'integral features' describe a property of a complete subtree. That is why we chose the name 'integral tree' in analogy to integral image. Integral trees can be used in many ways. We will show first experimental results for a top-down decomposition of the spanning tree into a disjoint set of subtrees with balanced diameters (section 5).

1.1 Further Motivation: TSP and Visual Problem Solving

Let us consider the traveling salesperson problem (TSP) in which n cities must be visited in the shortest time. Suppose that the regulation allows an agent to travel to at most 10 cities. The solution to this problem requires many agents, breaking the original TSP problem into k TSP problems. A simple solution is to cover the vertices of the graph with $k-$tours and to balance the load of the agents, for example by minimizing the maximal tour, or by minimizing the diameter of the subgraph. The TSP is that of finding a shortest tour (minimum length) that visits all the vertices of a given graph with weights on edges. The problem is known to be computationally intractable (NP-hard) [4]. Several heuristics are known to solve practical instances [4]. The TSP has been generalized to *multiple salespersons* ($k-$TSP), allowing each salesperson to visit n/k out of n cities. Another closely related problem is the multiple minimum spanning tree ($k-$MST) problem, where k trees are generated where each tree contains a root, and the size of the largest tree in the forest is minimized. Our goal is to generate a spanning forest that consists of k trees with roots, such that the diameters of the trees are balanced, i.e. none of the diameters of trees in the forest is greatly larger than the other tree diameter.

More recently, pyramid algorithms have been used to model the mental mechanisms involved in solving the visual version of the TSP [5], as well as other types of visual problems [6]. Humans seem to represent states of a problem by clusters (recursively) and determine the sequence of transformations from the start to the goal state by a top-down sequence of approximations. This approach leads to algorithms whose computational complexity is as low as that of the mental processes (i.e. linear), producing solution paths that are close to optimal. It follows that pyramid models may provide the first plausible explanation of the phenomenon of the directedness of thought and reasoning [7].

It is important to emphasize that by "pyramid algorithms" we mean any computational tool that performs image analysis based on multiple representations of the image forming a hierarchy with different scales and resolution, and in which the height (number) of a given level is a logarithmic function of the scale (and resolution) of the operators. Multiresolution pyramids form a subset of the general class of exponential pyramid algorithms. Pyramid algorithms, which incorporate a wider class of operators, are adequate models for the Gestalt rules of perceptual organization such as proximity, good continuation, common fate [8]. They also provide an adequate model of Weber's law and the speed-accuracy tradeoff in size perception, as well as of the phenomenon of mental size transformation [9]. In the case of size processing, modeling visual processes involves both bottom-up (fine to coarse) and top-down (coarse to fine) analyses. The top-down processing seems also critical in solving the image segmentation problem, which is a difficult inverse problem. This problem has received much attention in psychological literature, and is known as figure-ground segregation phenomenon [10].

2 Distance Transform

Let $G(V,E)$ denote a graph embedded in the plane and $\overline{G}(F, \overline{E})$ its dual. Algorithm in Fig. 1 labels each vertex of the graph $G(V,E)$ with the (shortest) distance $d_{\min} : V \mapsto \{0, 1, \ldots, \infty\}$ from the background. Assume that the vertices of the graph describe a binary shape and the edges determine the vertice's neighbors. It is the translation of the parallel algorithm [12] from grids to graphs. Distances of vertices on the boundary to the background are initialized to 1. Edge lengths $l(e) > 0$ in Algorithm Fig. 1 accomodate the fact that lengths other than 1 can appear. On square grids diagonal connections could be weighted by $\sqrt{2}$ or by appropriate chamfer distances [11]. In contracted graphs edges correspond to paths connecting two vertices. In such cases the length of the contracted edge could hold the length of the corresponding path. The integral property resulting

```
1. Initialize distances $d_{\min}(v) := \begin{cases} 1 & \text{if } v \text{ is on the boundary} \\ \infty & \text{otherwise} \end{cases}$
2. repeat $\forall v \in V$ in parallel:
   $d_{\min}(v) := \min(d_{\min}(v), min\{l(e) + d_{\min}(w) | (v,w) \in E \text{ or } (w,v) \in E\})$
```

Fig. 1. Algorithm: Parallel distance transform on a graph

from the distance transform is that the boundary of the shape can be reached from any vertex v with a path of length $d_{\min}(v)$ at most.

3 Determine the Spanning Tree

The smallest connected graph covering a given graph is a spanning tree. The diameter of a tree is easier and more efficient to determine than of a graph in general. In addition elongated concave shapes force the diameter to run along the shape's boundary, which is very sensitive to noise.

3.1 Minimal Spanning Tree

The greedy algorithm proceeds as follows: fist it computes distance transform $d_{\min}$; then computes edge weights $w(e) = -d_{\min}(u)d_{\min}(v)$ for all edges $e = (u, v)$; and finally finds minimal spanning tree using Kruskal's greedy algorithm. Skeletons based on morphology or distance transform give usually better results but the subsequent algorithms were able to cope with these deficiences.

3.2 Spanning Skeleton

The construction of the spanning tree is related to the computation of the distance transform and the skeleton of the shape. It operates on the dual graph $\overline{G} = (F, \overline{E})$ consisting of faces F separated by dual edges $\overline{E}$. Let us denote $B \subset F$ the background face(s) and by $\deg_b(f) := |\{(f, b) \in \overline{E}\}|$ the number of edges connecting a face $f \in F$ with the background B. Algorithm in Fig. 2 uses dual graph contraction [3] to successively remove edges connecting the interior of the shape with the background B while simplifying the boundary by removing unnecessary vertices of degree one and two. In our case dual removal of an edge e merges face f with the background face b and corresponds to contracting edge $\overline{e} = (f, b)$ in the dual graph $\overline{G}$. The result is a set of contraction kernels used to build the graph pyramid up to the apex. The searched spanning tree is the equivalent contraction kernel $(V, E_{eck}), E_{eck} \subset E$ [13] of the apex.

```
1. dually contract vertices of degree 1 and 2 in G; (the connecting edges correspond
   to self-loops and multi-edges in the dual graph $\overline{G}$.)
2. dually remove all edges $e \in E$ (in parallel) if
     – edge $\overline{e} = (f, b) \in \overline{E}, b \in B$ separates
     – a foreground face $f \in F \setminus B$ from the background
     – in a unique way: $\deg_b(f) = 1$.
3. for all faces $f \in F$ multiply connected with the background, $\deg_b(f) > 1$, do:
   (a) select an edge $\overline{e} = (f, b) \in \overline{E} \subset (F \setminus B) \times B$ and
   (b) dually remove $e$ from $E$.
4. repeat steps 1 – 3 until $F = \emptyset$
5. spanning skeleton is the equivalent spanning tree of the surviving vertex of $G$.
```

Fig. 2. Algorithm: Spanning Skeleton

3.3 Discussion and Computational Complexity

In Step 1 of the algorithm we distinguish two cases: i) If the vertices of degree less than 3 are adjacent to the background B a complete subtree externally attached to the shape is removed after a number of (sequential) steps corresponding to the length of the longest branch of the tree, and ii) vertices of degrees 1 and 2 may also exist inside the shape if they are not adjacent to the background. They are removed similar to the external tree in the very first step. As before the complexity depends on the longest branch. Since the dual contraction of all trees is independent of each other, the parallel complexity is bound by the longest branch of any tree. Step 2 removes all edges on the boundary of the graph as long as the non-background face is not multiply connected to the background. They are all independent of each other and hence can be removed in one single parallel step. Step 3 removes one of the edges of faces which are multiply connected to the background. Since vertices of degree 2 have been eliminated in step 1 this can only happen at 'thin' parts of the graph (where the removal of 2 or more such edges would disconnect the graph). Only one edge need to be removed to allow the face to merge with the background. Since different faces multiply connected to the background are independent of each other all dual removals can be done in one single parallel step.

The total number of steps needed to complete one iteration of steps 1-3 depends on the longest branch of a tree in step 1 and needs two additional steps. The branches contracted in step one become part of the final spanning tree hence in total, all steps 1 need at most as many steps as the longest path through the tree (i.e. its diameter). The number of iterations is limited by the thickness of the graph since at each iteration one layer of faces adjacent to the background is removed. Hence we conclude that the parallel complexity of the algorithm in the worst case is $\mathcal{O}(diameter(G) + thickness(G))$.

4 Diameter and Integral Tree of Depths

Given a (spanning) tree adapted to the shape we would like to measure distances between any vertices of the tree. Algorithm in Fig. 3a labels each vertex with the length $d_{\max}$ of the longest tree branch away from the center. The result is the same as produced by [14] but it differs by its parallel iterated and local operations. Given the (spanning) tree $T = (V, E_{eck})$ algorithm **Subtree Depth** computes the vertex attribute $d_{\max}$ in $\mathcal{O}(|diameter|/2)$ parallel steps. If the tree is cut at any edge $e = (u, v)$, $d_{\max}(v)$ gives the depth of the remaining tree which includes vertex v. It has the integral property that any leaf of the subtree can be reached along a path not longer than $d_{\max}(v)$. The function $\max2\{M\}$ returns the second largest value of the argument set M, i.e. $\max2(M) := \max(M \setminus \{\max(M)\})$.[1]

[1] If set M has less than two elements, then the function is *not defined* in general. If, however, it appears as a member of a set like in $\min(\cdot)$ or $\max(\cdot)$ then it can simply be ignored or, formally, replaced by the empty set: $\max2(\emptyset) = \max2\{x\} = \emptyset$.

```
1. Initialize distances $d_{\max}(v) := \begin{cases} 0 & \text{if } v \text{ is a leaf}, \deg_T(v) = 1 \\ \infty & \text{otherwise} \end{cases}$
2. repeat for all vertices $v \in V$ in parallel:
   $d_{\max}(v) := \min(d_{\max}(v), \max2\{l(e) + d_{\max}(w) | (v,w) \in E_{eck} \text{ or } (w,v) \in E_{eck}\})$
```

a

$$c \to \cdots \to v \not\to w \begin{cases} \to s_1 \cdots \to l_1 \\ \to u \begin{cases} \to s_2 \cdots l_2 \\ \to s_3 \cdots l_3 \\ \to s_4 \cdots l_4 \end{cases} \\ \to s_n \cdots \to l_n \end{cases}$$

b

Fig. 3. a) Algorithm: Subtree Depth and b) subdivision of the Subtree Depth

4.1 Center and Diameter of the Spanning Tree

The sample result is shown in Fig. 4b. Each vertex is labeled with two values, the first being the subtree depth. The **diameter** is the longest path[2] through the tree and consists of the two sub-paths $v_0, v_1, \ldots, v_9$ and $w_9, w_8, \ldots, w_0$ with $d_{\max}(v_i) = d_{\max}(w_i) = i, i = 0 \ldots 9$. Its length is 19. There is one edge (v_9, w_9) of which both ends have (maximal) depth 9. This is the **center of the tree** with the (integral) property that all leafs (i.e. all vertices!) of the tree can be reached in maximally $d_{\max}(v_9) = 9$ steps. The diameter of this tree is obviously 19, an odd number. All trees with an odd diameter have a central edge. Trees with an even diameter have a single maximum $d_{\max}$-value, e.g. a vertex is the center. Similar information is contained in the subtree depth of the other vertices: Given the center of the tree, we can orient the edges such that they either point towards the center or away from the center. Let us assume in the following that all *edges of the tree are oriented towards the center.*

4.2 Computational Complexity of Algorithm Subtree Depth

We consider the number of repetitions of step 2 and the number of steps required to compute max2. First we note that the algorithm stops if the function $d_{\max}(v)$ does not change after updating of step 2. It starts with vertices of subtree depth 0 and increases the distance values at each (parallel) iteration. Hence step 2 need not be repeated more than half the diameter times. To compute the $d_{\max}$-value in step 2 all the neighbors of a vertex need to be considered. Hence this is bounded by the degree of the vertex. In summary the parallel computational complexity is $\mathcal{O}(diameter * maximal\_vertex\_degree)$.

5 Decomposing the Spanning Tree

In [14] we presented an algorithm to decompose a spanning tree into subtrees such that the diameter of each subtree is maximally half the diameter of the

[2] Edge length $l(e) = 1$ is used in all examples.

Table 1. Degrees of the contraction kernels

	'Bister'								
level\deg	0	1	2	3	4	5	6	8	δ
$0 \to 1$	1653	759							1
$1 \to 2$	2340		24						2
$2 \to 3$	2124		48	24					6
$3 \to 4$	1779		99	8	8	8			10
$4 \to 5$	1111		199	21	22				19
$5 \to 6$	451		244	8	18	8			25
$6 \to 7$	75		174	16	4	8			32
$7 \to 8$	13		48	16			8		43
$8 \to 9$	3		8		2	8			50
$9 \to 10$			1					2	62
$10 \to 11$			1						120

	'Disc'								
level\deg	0	1	2	3	4	5	6	8	δ
$0 \to 1$	821	380							1
$1 \to 2$	1165		12						2
$2 \to 3$	1057		24	12					6
$3 \to 4$	877		52	4	4	4			10
$4 \to 5$	529		110	8	10				19
$5 \to 6$	229		116	4	8	4			25
$6 \to 7$	37		86	8	2	4			32
$7 \to 8$	5		24	8			4		43
$8 \to 9$			4		1	4			50
$9 \to 10$								1	62

original tree. Recursively continued until the subtrees have a diameter ≤ 2, this strategy creates a hierarchy of log(diameter) height. The only parameter used for this decomposition is the length of the diameter and the center of the tree.

We studied the relation between the shape (two sample examples are shown in Fig. 5a,b, for more examples see [15]) and the resulting graph pyramid. Table 1 lists the observed properties of the contraction kernels used at level k to produce level $k+1$ ($k \to k+1$). For every level the histogram of kernel's degrees is given together with the largest diameter δ of all subtrees at the respective levels. The similarity of the substructure 'Disc' to 'Bister' is obvious and not surprising. The length of the diameter and the center appear to be very robust whereas the fine substructures are sensitive to noise. In particular we observe many spurious branches ($\deg(v) = 0$) and high splitting degrees. This can be avoided to a large extend and optimized using subtree diameters.

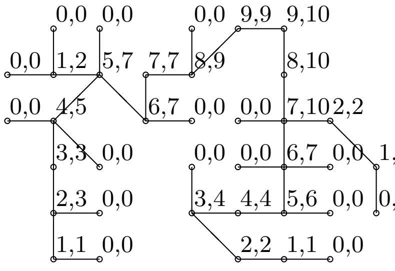
5.1 The Integral Tree of Diameters

Subtree depths $d_{\max}$ are upper bounds for reaching any vertex in the outer subtree. Consider the following configuration sketched in Fig. 3b): c denotes the center, l_i are the leafs, v, w, u, s_i are intermediate vertices. $dist(x, y)$ denotes the distance between vertices x and y. The depth of the center c is not shorter than the distance to any leaf[3]: $d_{\max}(c) \geq \mathrm{dist}(c, l_i)$. The actual distance between the center and any vertex v is also bounded: $\mathrm{dist}(c, v) \leq d_{\max}(c) - d_{\max}(v)$. Along the tree's diameter-path the above inequalities are equalities. Assume we cut the tree between vertices v and w. The diameter of the outer subtree of w goes either through w or it connects two subbranches excluding w. If it goes through w its length is the sum of the subtree depth of w and the length of its second longest subbranch. The length of a subbranch is the length of the edge connecting the branch to w plus the subtree depth of the first son in this subbranch: $\delta(w) = d_{\max}(w) + \max 2\{l((w, s)) + d_{\max}(s) | (w, s) \in E_{eck}\}$.

[3] Odd diameters create a central edge splitting the tree in two subtrees for which the above inequalities hold.

```
1. Initialize diameters $\delta(v) := d_{\max}(v)$
2. repeat $\forall v \in V$ in parallel:
   $\delta(v) := \max(\max\{\delta(s) | (v,s) \in E_{eck}\},$
   $\max2\{d_{\max}(v) + l((v,s)) + d_{\max}(s) |$
   $(v,s) \in E_{eck}\})$
```

a



b

Fig. 4. a) Algorithm: Subtree Diameters δ, and b) Integral trees of depths and diameters $d_{\max}, \delta$

The max2-function is well defined because $d_{\max}(w) > 0$ implies a degree $\deg(w) \geq 2$. If the diameter of subtree w does not go through w it connects two leafs through a vertex, e.g. $u : l_2 \cdots s_2 \leftarrow u \rightarrow s_4 \cdots l_4$. In this case vertex u calculates the diameter as w above and propagates the length of the diameter up to vertex w. The diameters of all subtrees can be computed similar to the Subtree Depth: Algorithm Fig. 4a generates diameters δ (2nd values in Fig. 4b).

5.2 Using Integral Trees for Decomposition

The integral features of depth $d_{\max}$ and diameter δ should enable us to decide locally where it is best to split the spanning tree. Criteria could be a good balance of diameter lengths, a small degree of the top contraction kernels (*"a hand has 5 fingers"*) or more object specific properties that could be known to the system.

Let us consider what happens if we cut the tree at a certain distance from the center by removing the cut-edge. A cut-edge (v, w) is selected if the depth of the outer tree is smaller than a threshold d_T, $d_{\max}(v) < d_T \leq d_{\max}(w)$ (*'cut-edge condition'*). Note that the threshold d_T can depend on the length of the overall diameter $\delta(c)$. After cutting, the longest possible diameter of the outer tree $\delta_{\max}$ is twice the subtree depth of $d_{\max}(v)$ (this was used in [14]). This can be improved using the actual diameters $\delta(v)$ calculated by algorithm subtree-diameters (Fig. 4b). If all edges satisfying the cut-edge condition are rigorously removed the depth of the remaining central tree is reduced by the subtree depth of new leaf $d_{\max}(w) = d_T$. Consequently the diameter of the central tree shrinks by the double amount $\delta_{\text{new}}(c) = \delta_{\text{old}}(c) - 2d_T$. Table in Fig. 5 lists the different diameters and degrees for all possible cut-depths d_T. The decomposition should first split the 'important' components and not be too much influenced by spurious subtrees. The degree of the contraction kernel corresponds exactly to the number of cut-edges. While the 'cut-degree' counts all rigorously created new subtrees including trees with very small depth and diameter (0 in table in Fig. 5), the 'min'-value gives the degree after re-connecting all cut-edges to the central tree which do not increase the largest diameter of all outer and the inner trees. The remaining subtree diameters are bold faced in table in Fig. 5.

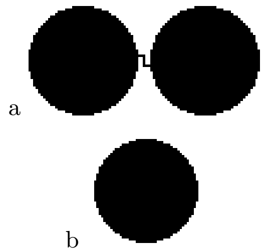


Table of Cuts through example tree Fig. 4b

cut	diameters of outer trees and center tree				deg((CK))	
d_T	$\delta_{\max}$	δ_{left}	$\delta(c)$	δ_{right}	cut	min
9	16	**9**	1	**10**	2	**2**
8	14	0, **7**	3	**10**	3	**2**
7	12	0, **7**	5	0, **7, 2**	5	**3**
6	10	0, **7, 0**	7	0, **0, 6, 0 , 2**	8	**6**
5	8	0, **0, 2, 5**, 0	9	0, 0, **4, 0**, 0, **2**	11	**6**
4	6	0, 0, **2, 0, 3, 0**, 0	11	0, 0, **4**, 0, 0, 2	13	**5**
3	4	0, 0, 2, 0, **3**, 0, 0	13	0, 0, **0, 2**, 0, 0, 2	14	**3**
2	2	0, 0, 2, 0, **1** ,**0**, 0, 0	15	0, 0, 0, **1**, 0, 0, 1	15	**3**
1	0	0, 0, 0, 0, 0, **0**, 0, 0, 0	17	0, 0, 0, **0**, 0, 0, 0	16	**2**

Fig. 5. Two example a) Bister($2 \times 1581 + 9$ pixel) and b) Disc (1581 pixel) used in experiments and Table of Cuts

5.3 Experiment: Two Connected Balls (*'Bister'*)

The example of Fig. 5a consists of *two large balls connected by a thin curve.* Bister et.al. [16] used a similar example to demonstrate the shift variance of regular pyramids. The goal of this experiment, refered to as *'Bister'*, is to check whether the simple decomposition expressed by the above description could be derived from the integral tree. Table 2 lists the different subtree depths and diameters in the example *'Bister'* (see subtree depth and diameters of central part in Fig. 6). This shows clearly that the diameters of the two circles (62) propagate up to the center which receives diameter 120. Cutting the path which connects the two large circles produces three subtrees (degree of contraction kernel 2) of which both outer subtrees have diameter 62 from cut-edge with subtree depths (59,60) down to (36,37). With smaller subtree depth the degrees of the contraction kernels start to grow since extra branches of the two circles are cut. We continued the table down to cut-edge (29,30) where the diameter of the center-tree becomes larger than any of the outer trees. We also note that no spurious branches can be integrated in this first level decomposition.

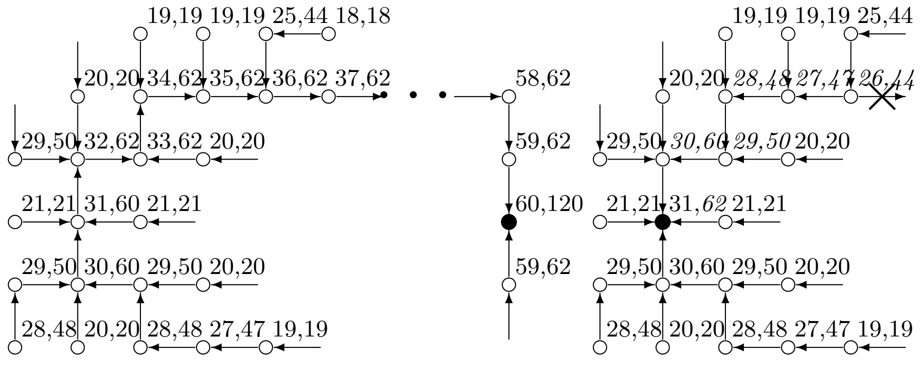


Fig. 6. Center part of left circle of example *Bister before and after cut*

Table 2. Cuts through spanning tree of example *'Bister'*

cut	diameters of outer trees and center tree				deg((CK))
d_T	$\delta_{\max}$	δ_{left}	$\delta(c)$	δ_{right}	cut
60	118	62	0	62	2
59	116	62	2	62	2
			...		
37	72	62	46	62	2
36	70	44, 62	48	44, 62	4
35	68	44,19, 62	50	44,19, 62	6
34	66	44,19,19, 62	52	44,19,19, 62	8
33	64	44,19,19, 62, 20	54	44,19,19, 62, 20	10
32	62	44,19,19,20,50, 60, 20	56	44,19,19,20,50, 60, 20	14
31	60	44,19,19,20,50,21, 60, 21,20	58	44,19,19,20,50,21,60,21,20	18
30	58	44,19,19,20,50,21,50,20,50,21,20	60	44,19,19,20,50,21,50,20,50,21,20	22
			...		

6 Conclusion

We have introduced integral trees that can store integral features or properties. Efficient parallel algorithms have been presented for computing: **i)** the boundary distance $d_{\min}$ of a binary shape; **ii)** the depth of all subtrees $d_{\max}$; and **iii)** the diameter δ of the outer subtrees. These integral features are not just sums over all elements of the subtree but capture properties of the complete substructure. The integral trees have been used to decompose the spanning tree of the shape top-down. The decomposition can use following optimization criteria: i) balance the diameters of the subtrees more efficiently than cutting at a fixed distance from the center or the leafs; unfortunately this often generates contracktion kernels of high degree; ii) set the degree n of the contraction kernel beforehand and find the n subtrees with largest integral feature, e.g. diameter. iii) define the optimization criterium which can be solved using local information provided by the integral tree and some global properties like global size or diameter proportion that are propagated during the top-down process. In future research we plan to apply integral tree for new solutions of the TSP problem as well in tracking.

References

[1] Viola, P., Jones, M.: Robust Real-time Face Detection. International Journal of Computer Vision **57** (2004) 137–154

[2] Beleznai, C., Frühstück, B., Bischof, H., Kropatsch, W.G.: Detecting Humans in Groups using a Fast Mean Shift Procedure. OCG-Schriftenreihe 179, (2004)71–78.

[3] Kropatsch, W.G.: Building Irregular Pyramids by Dual Graph Contraction. IEE-Proc. Vision, Image and Signal Processing **142** (1995) pp. 366–374

[4] Christofides, N.: The Traveling Salesman Problem. John Wiley and Sons (1985)

[5] Graham, S.M., Joshi, A., Pizlo, Z.: The Travelling Salesman Problem: A Hierarchical Model. Memory & Cognition **28** (2000) 1191–1204

[6] Pizlo, Z., Li, Z.: Graph Pyramids as Models of Human Problem Solving. In: Proc. of SPIE-IS&T Electronic Imaging, Computational Imaging, (2004) 5299, 205–215

[7] Humphrey, G.: Directed Thinking. Dodd, Mead, NY (1948)

[8] Pizlo, Z.: Perception Viewed as an Inverse Problem. Vis. Res. **41** (2001) 3145–3161
[9] Pizlo, Z., Rosenfeld, A., Epelboim, J.: An Exponential Pyramid Model of the Time-course of Size Processing. Vision Research **35** (1995) 1089–1107
[10] Koffka, K.: Principles of Gestalt psychology. Harcourt, NY (1935)
[11] Borgefors, G.: Distance Transformation in Arbitrary Dimensions. Computer Vision, Graphics, and Image Processing **27** (1984) 321–145
[12] Borgefors, G.: Distance Transformation in Digital Images. Computer Vision, Graphics, and Image Processing **34** (1986) 344–371
[13] Kropatsch, W.G.: Equivalent Contraction Kernels to Build Dual Irregular Pyramids. Springer-Verlag Advances in Computer Vision (1997) pp. 99–107
[14] Kropatsch, W.G., Saib, M., Schreyer, M.: The Optimal Height of a Graph Pyramid. OCG-Schriftenreihe 160, (2002) 87–94.
[15] Kropatsch, W.G., Haxhimusa, Y., Pizlo, Z.: Integral Trees: Subtree Depth and Diameter. Technical Report No. 92, PRIP, Vienna University of Technology (2004)
[16] Bister, M., Cornelis, J., Rosenfeld, A.: A Critical View of Pyramid Segmentation Algorithms. Pattern Recognition Letters **11** (1990) pp. 605–617