Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/183-2
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: krw@prip.tuwien.ac.at, yll@prip.tuwien.ac.at, pizlo@psych.purdue.edu
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-92 15 September 2004

Integral Trees: Subtree Depth and Diameter

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

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.
Keywords: Hierarchical graph-based skeleton, irregular graph pyramids, topology preserving contraction.

[1]Supported by the Austrian Science Foundation under grants P14445-MAT, P14662-INF and FSP-S9103-N04.

[2]Z.Pizlo is with Department of Psychological Sciences, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-1364 and is supported by ECVision and by the Air Force Office of Scientific Research grant FA9550-04-1-0297.

Contents

1 Introduction

Viola and Jones introduced the ’Integral Image’ [30] 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 as 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 [3]. 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 vertices 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 be related through the concept of dual graph contraction [14] 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 3 transfers the classical parallel algorithm for computing the distance transform [6, 8, 29] 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 4). This is similar to the distance transform and to well-known shrinking and thinning algorithms [19, 20, 21]. 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 5 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. Of course it may be longer than the diameter of the graph but it has other advantages as we will see. Partly as a by-product we compute the maximal path lengths of all branches of the subtrees and the respective diameters (section 6.1). These ’integral features’ describe a property of a complete subtree. That is why we chose the name ’integral tree’ in analogy to integral image. We will show first experimental results for an top-down decomposition of the spanning tree into a disjoint set of subtrees with balanced diameters (section 6).

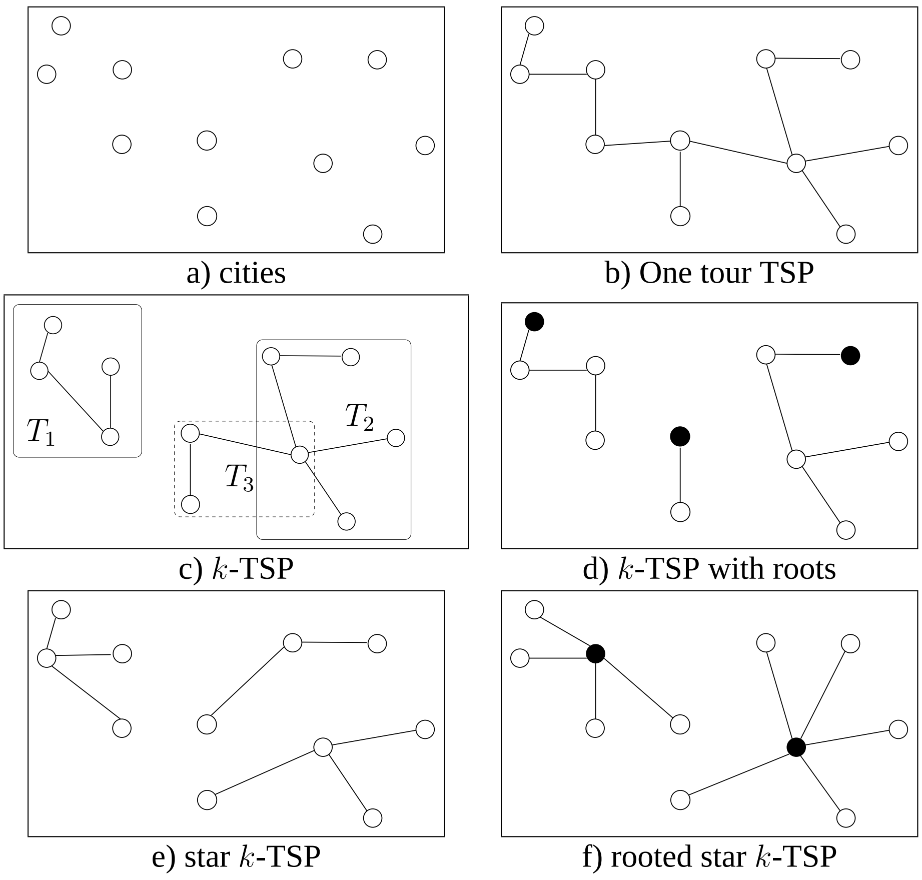


Figure 1: Traveling salesperson problem.

2 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. In Fig. 1 the cities are the vertices and the costs are associated with the edges connecting the cities. 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 (Fig. 1c,d,e,f). The traveling salesperson problem (TSP) is that of finding a shortest tour (minimum length) that visits all the vertices of a given graph with weights on edges [11] Ch.8. This problem has received considerable attention in the literature [18, 9]. The problem is known to be computationally intractable (NP-hard) [2]. Several heuristics are known to solve practical instances [9]. A problem similar to TSP on a weighted graph, is the *Chinese postman problem*: find the shortest *closed walk* that traverses each edge at least once. It can be solved optimally in polynomial time [1]. The traveling *salesperson* problem has been generalized to *multiple salespersons* ($k-$TSP), allowing each salesperson to visit n/k out of n cities. This is a version of *vehicle routing*, where one wants to schedule vehicles to carry objects from specified source vertices (root) to specified destinations, for an overview see [9]. Another closely related problem is the multiple minimum spanning tree ($k-$MST) problem. In [32, 31], 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 Traveling Salesman Problem [10], as well as other types of visual problems [23, 24].

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), and which produce 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 [12].

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. In multiresolution algorithms the computation of the mean intensity is the only operation.

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 (e.g. [26, 22]). 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 [25]. 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 (e.g. [7]). This problem has received much attention in psychological literature, and is known as figure-ground segregation phenomenon [13].

3 Distance Transform

Let $G(V, E)$ denote a graph embedded in the plane and $\overline{G}(F, \overline{E})$ its dual. Algorithm 1 labels each vertex of the graph $G(V, E)$ with the (shortest) distance $d_{\min} : V \mapsto \{1, \ldots, \infty\}$ from the background. Assume that the vertices of the graph describe a binary shape and the edges determine the vertice’s neighbors.

Algorithm 1 Parallel Distance Transform on a Graph

```
1. Initialize distances $d_{\min} : V \mapsto \{0, 1, 2, \dots, \infty\}$
   $d_{\min}(v) := \begin{cases} 1 & \text{if } v \text{ is on the boundary} \\ \infty & \text{otherwise} \end{cases}$

2. repeat for all vertices $v \in V$ in parallel:
   $d_{\min}(v) := \min(d_{\min}(v), \min\{l(e) + d_{\min}(w) | (v, w) \in E \text{ or } (w, v) \in E\})$
```

It is the direct translation of the well known parallel algorithm from grids to graphs [28] [Chap.12]. Distances of vertices on the boundary to the background are initialized to 1. In some implementations on grids distances on the shape are chosen greater than 0 because pixels in the background are explicitely represented whereas in the graph formulation there is only one background face. The results differ by the constant value of 1.

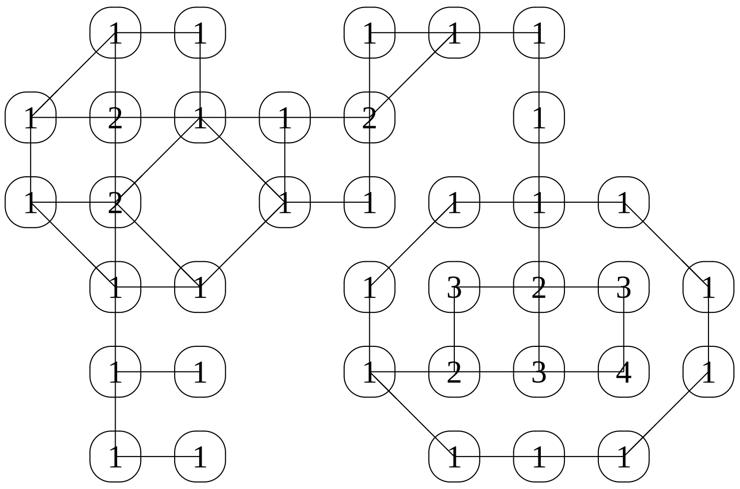


Figure 2: Distance transform

Edge lengths $l(e) > 0$ are considered in Algorithm 1 to accommodate 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 [5]. In the contracted graphs of graph pyramid edges correspond to paths connecting two vertices. In such cases the length of the contracted edge could hold the length of the corresponding path. In case that the length of the edges $l(e) = 1$ for all edges $e \in E$ we simply add 1 to the minimum of all edges incident on v in step 2. The resulting distances to the boundary are shown in Fig. 2.

The integral property resulting from the distance transform is the fact that the boundary of the shape can be reached from any vertex v in at most $d_{\min}(v)$ steps, or with a path of length $d_{\min}(v)$ at most.

4 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 and varies with articulated motion.

4.1 Minimal Spanning Tree

The following greedy algorithm 2 was used in several experiments because of its efficient implementation. We noted some drawbacks like non-optimal spanning trees with respect to the shape. Skeletons based on morphology or distance transform give usually better results but it showed that the subsequent algorithms were able to cope with these deficiencies.

Algorithm 2 Minimal Spanning Tree

```
1. compute distance transform $d_{\min}(v), \forall v \in V$:
2. compute edge weights $w(e) = -d_{\min}(u)d_{\min}(v)$ for all edges $e = (u, v) \in E$.
3. find minimal spanning tree using Kruskal's greedy algorithm [17].
```

The reasons for choosing weight $-d_{\min}(u)d_{\min}(v)$ are as follows:

- MST is computed on the weighted graph,
- local computation combines $d-$values of two end points of the edge, and
- $ab \to \max$ for $a + b = const. \Rightarrow a = b$; in our case $|a - b| \leq l(e)$[1] $\Rightarrow a + b \approx const$ in local neighborhood. To enforce $\max$ to increase in distance we take $-ab$ as local decision.

4.2 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. For the sake of simplicity we assume a single background face b. If there are more than one background faces, i.e. if the shape is not simply connected and has holes, all background faces are collected in a set B of background faces.

[1]l(e)=1

Algorithm 3, spanning skeleton, uses dual graph contraction [14] 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 then the equivalent contraction kernel [15] of the apex. We denote the edges of this equivalent contraction kernel by $E_{eck} \subset E$. Fig. 3 illustrates

Algorithm 3 Spanning Skeleton

```
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. the spanning skeleton is the equivalent spanning tree of the surviving vertex of $G$.
```

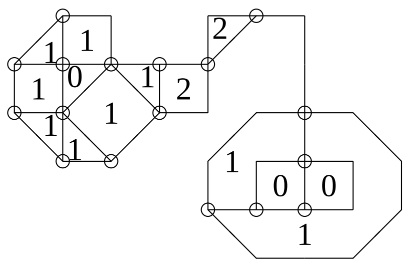
the algorithm. After three passes through steps $1 - -3$ the apex is reached in our example. The resulting spanning tree concatenates the equivalent contraction kernels used in step 1 of the algorithm.

4.3 Discussion and Computational Complexity of Algorithm 3

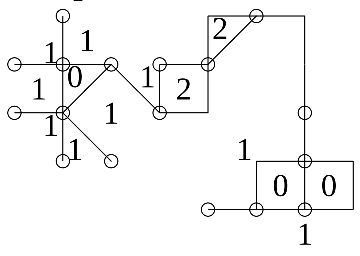
In Step 1 of the Algorithm 3, Spanning Skeleton, we distinguish two cases:

- 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.
- 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.

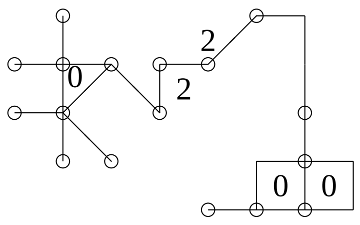
Configurations of first iteration:



After Step 1: vertices have degree > 2, background connectivity of faces indicated.

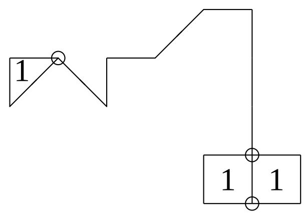


After Step 2: Faces with $\deg_b(f) = 1$ merged with background.

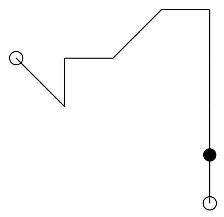


After Step 3: Faces with $\deg_b(f) = 2$ merged with background

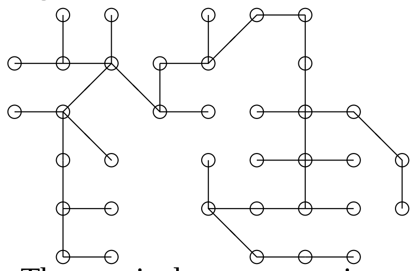
Configurations of second iteration and final spanning tree E_{eck}:



After Step 1: Vertices with $\deg(v) < 3$ removed, 3 faces remain.



After Step 2: All faces merged with background, apex in bold.



The equivalent contraction kernel of the apex is a tree and spans V.

Figure 3: Spanning Skeleton of Example

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 $\boxed{\mathcal{O}(\text{ diameter}(G)+\text{ thickness}(G))}$.

4.4 Control Parameters

There are only few non-deterministic decisions in the algorithm: which edge to select in step 3(a)? If the goal is to build a spanning tree of shortest total length then the longest edge among the edges separating the face from the background can be chosen. This length can either stem from the original geometry or it may have been accumulated during dual contraction of vertices of degree 2 in step 1.

The construction of the spanning tree combines the trees dually contracted in step 1 and all but one edges of the connecting paths corresponding to edges dually removed in steps 2 and 3. Here a plausible strategy would be to choose the middle edge of the connecting path for not being added. This would keep the depth of the tree low since the remaining set of edges would be split in half and attached to different branches of the tree. If the shrinking is to be controlled by geometry then the distance of the faces from the background can be used to prohibit faces from being merged with the background in steps 2 and 3.

5 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 4 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 [16] 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

Algorithm 4 Subtree Depth

```
1. Initialize distances $d_{\max} : V \mapsto \{0, 1, 2, \dots, \infty\}$
   $d_{\max}(v) := \begin{cases} 0 & \text{if } v \text{ is a leaf, e.g. } \deg_T(v) = 1 \\ \infty & \text{otherwise} \end{cases}$

2. repeat for all vertices $v \in V$ in parallel:
   $d_{\max}(v) := \min(d_{\max}(v), \text{max2}\{l(e) + d_{\max}(w) | (v, w) \in E_{eck} \text{ or } (w, v) \in E_{eck}\})$
```

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$(M) := \max(M \setminus \{\max(M)\})$.[2]

5.1 Center and Diameter of the Spanning Tree

The sample result is shown in Fig.5. Each vertex is labeled with two values, the first being the subtree depth. The **diameter** is the longest path[3] through the tree and consists of the two subpaths $v_0, v_1, \dots, v_9$ and $w_9, w_8, \dots, w_0$ with $d_{\max}(v_i) = d_{\max}(w_i) = i, i = 0 \dots 9$. 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*.

5.2 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 $\boxed{\mathcal{O}(\text{diameter} * \text{maximal vertex degree})}$.

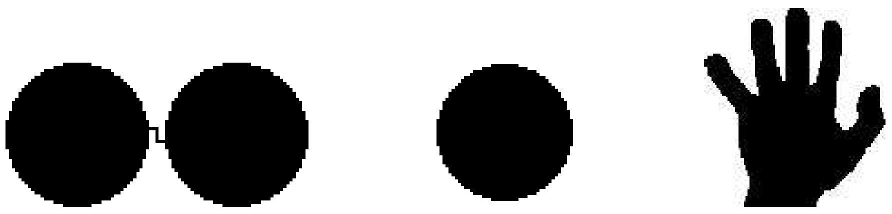
6 Decomposing the Spanning Tree

In [16] 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 original tree. Recursively continued until the subtrees have a diameter less equal 2, this strategy creates a hierarchy of $\log(\text{diameter})$ 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. 4) and the resulting graph pyramid. Table 1 lists the observed properties of the contraction kernels used

[2]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, mathematically more correct, replaced by the empty set: max2$(\emptyset) =$ max2$\{x\} = \emptyset$.

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



a) Bister($2 \times 1581 + 9$ pixel) b) Disc (1581 pixel) c) Hand (100×109)

Figure 4: Example images used in experiments.

at level k to produce level $k+1$ ($k \rightarrow k+1$). 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.

6.1 The Integral Tree of Diameters

Subtree depths $d_{\max}$ are upper bounds for reaching any vertex in the outer subtree. Consider the following configuration : 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.

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

The depth of the center c is not shorter than the distance to any leaf[4]: $d_{\max}(c) \geq \text{dist}(c, l_i)$. The actual distance between the center and any vertex v is also bounded: $\text{dist}(c, v) \leq d_{\max}(c) - d_{\max}(v)$. 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: $\delta(w) = d_{\max}(w) + \text{max2}\{l((w, s)) + d_{\max}(s) | (w, s) \in E_{eck}\}$. The max2-function is well defined because $d_{\max}(w) > 0$ implies the 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 calculated similar to the Subtree Depth: Alg. 5 generates diameters δ (2nd values in Fig. 5).

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

Table 1: Degrees of the contraction kernels.

	'Bister'								
$_{\text{level}}\backslash^{\text{deg}}$	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'								
$_{\text{level}}\backslash^{\text{deg}}$	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

	'Hand'																	
$_{\text{level}}\backslash^{\text{deg}}$	0	1	2	3	4	5	6	7	8	9	11	12	13	21	25	40	r	$\max L_i$
$0 \to 1$	2999	1108															1.27	1
$1 \to 2$	4098		3														1	3
$2 \to 3$	3830		89	1													1.05	7
$3 \to 4$	2549		444	4	2	1	1										1.31	13
$4 \to 5$	779		719	6	3	2	2										1.99	25
$5 \to 6$	70		445	3	6	1	1	2		1	1	1					2.85	49
$6 \to 7$			137	3	2	1		3	1	1		1	1	1			3.52	97
$7 \to 8$			20			1				2	2					1	5.81	193
$8 \to 9$															1		26	385

6.2 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 [16]). This can be improved using the actual diameters $\delta(v)$ cal-

Algorithm 5 Subtree Diameters δ

```
1. Initialize diameters $\delta(v) := d_{\max}(v)$

2. repeat for all vertices $v \in V$ in parallel:
```

$$\delta(v) := \max(\quad \begin{array}{l} \max\{\delta(s) | (v,s) \in E_{eck}\}, \\ \max2\{d_{\max}(v) + l((v,s)) + d_{\max}(s) | (v,s) \in E_{eck}\} \\ ) \end{array}$$

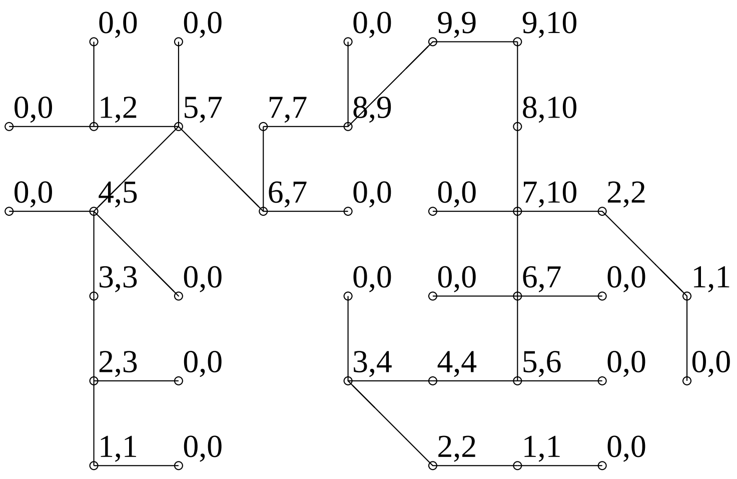


Figure 5: Integral trees of depths and diameters $d_{\max}, \delta$.

culated by algorithm subtree-diameters (Fig. 5). 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 2 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. Therefore we consider the degrees of the resulting contraction kernels. 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 2), 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 2.

Table 2: Cuts through example tree Fig. 5

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**

Table 3: 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
			. . .		

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

The example of Fig. 6 consists of *two large balls connected by a thin curve*. Bister etal. [4] used a similar example to demonstrate the shift variance of regular pyramids. The goal of this experiment, referred to as *'Bister'*, is to check whether the simple decomposition expressed by the above description could be derived from the integral tree.

Table 3 lists the different subtree depths and diameters in the example *'Bister'* (see subtree depth and diameters of central part in Fig. 7). 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.

We cut the tree at cut-edge (36,37) indicated in Fig. 6 by darker vertices. Of the remaining three subtrees the center tree is a single path of length 46 and the two symmetric circles have

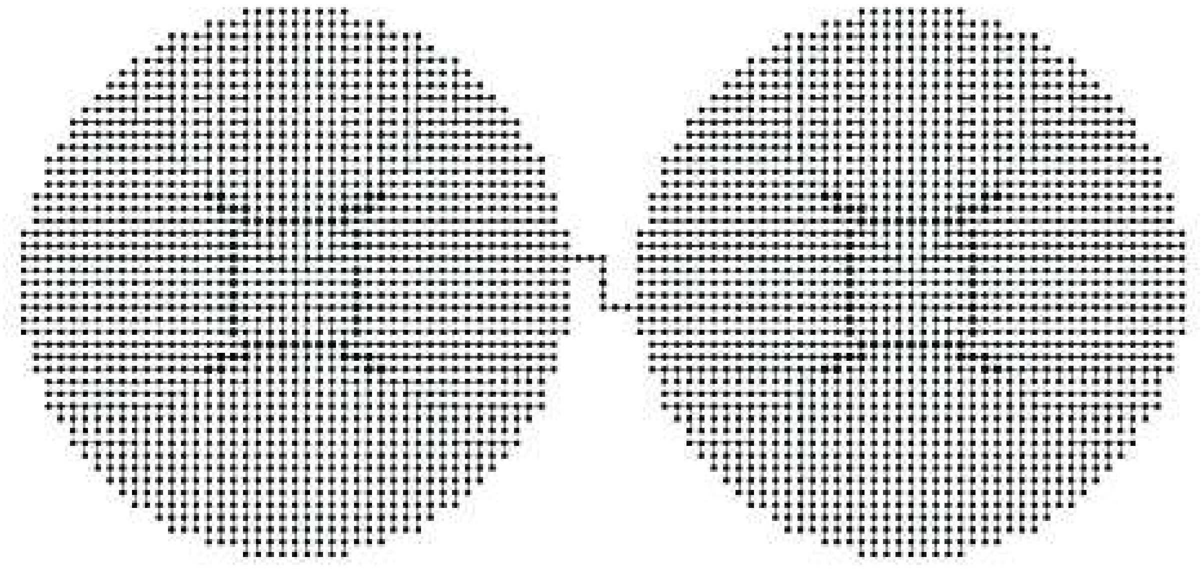


Figure 6: Cutting at 37 example *'Bister'*

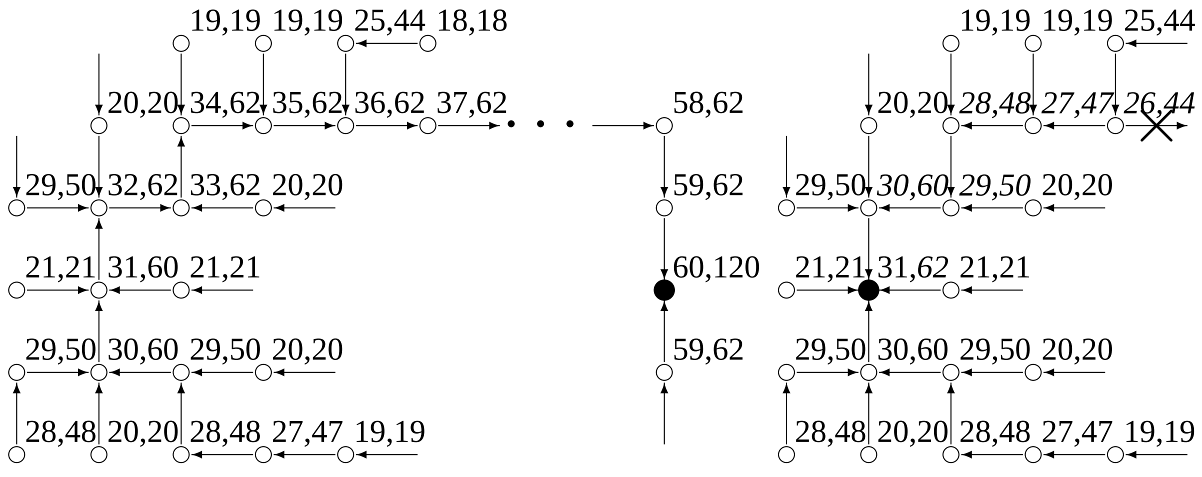


Figure 7: Center part of left circle of example *Bister before and after cut*

Table 4: Cuts through the left subtree of example *'Bister'*

cut	diameters of outer trees and center tree				$\deg((CK))$
d_T	$\delta_{\max}$	δ_{left}	$\delta(c)$	δ_{right}	cut
31	60	60,21	0	60,21	4
30	58	50,20,50,21	2	50,20,50,21	8
29	56	20,48,20,48,21	4	48,20,48,20,21	10
			...		
17	32	$4 \times 29+$ path lengths$[14, 16]$	28	$4 \times 29+$ path lengths$[14, 16]$	55
16	30	$4 \times 27+$ path lengths$[12, 15]$	30	$4 \times 27+$ path lengths$[12, 15]$	59
			...		

diameter 62 and will be decomposed recursively. After the cut the subtree depth and the diameters need to be updated for the path that leads to the center (see Fig. 7). Table 4 summarizes the decomposition after cutting the left circle from the example *'bister'*.

6.4 Experiment: Hand

The image of the hand is a range image segmented from the NRCC image database [27] [5]. The goal of this experiment, referred to as *'Hand'*, it is the same as previously, is to check whether the simple decomposition, expressed by the above description could be derived from the integral tree, like *"a hand has* 5 *fingers"*. Table 5 lists the different subtree depths and diameters in the example *'Hand'* (bold face values are the largest depths). The boundary from the center of *'Hand'* can be reached in at most $d_{\min}(c) = 25$ steps, i.e. with the path of length $d_{\min}(c) = 25$. From the Table 5 using $d_T \geq d_{\min}$ can be used to separate the longest branches from the center, e.g. $74 < d_T \leq 77$ we have 4 long branches and a center branch (see numbers depicted in bold in row 4, Table 5); for $64 < d_T \leq 74$ we get 5 long braches and a center branch; for $d_T \leq 64$ we got 7 long branches and a center branch; and so on. In order to understand how these branches are created, imagine that we cut the graph (Figure 8) with 'circles' having center(s) at the graph's center. In Figure 8 the main branches are highlighted [6].

7 Conclusion

We have introduced integral trees that can store integral features or properties. The author in [21] uses as the center of the skeleton the maximum of the minimum of a distance (significant) measure [7]. In this paper we use as a center of the skeleton minimum of a maximum of a distance measure. Efficient parallel algorithms have been presented for computing

- the boundary distance $d_{\min}$ of a binary shape;

[5]The authors would like to thank Kaleem Siddiqi for making the images available.
[6]The branches are given with different colors, to help follow the Table 5.
[7]Residual function.

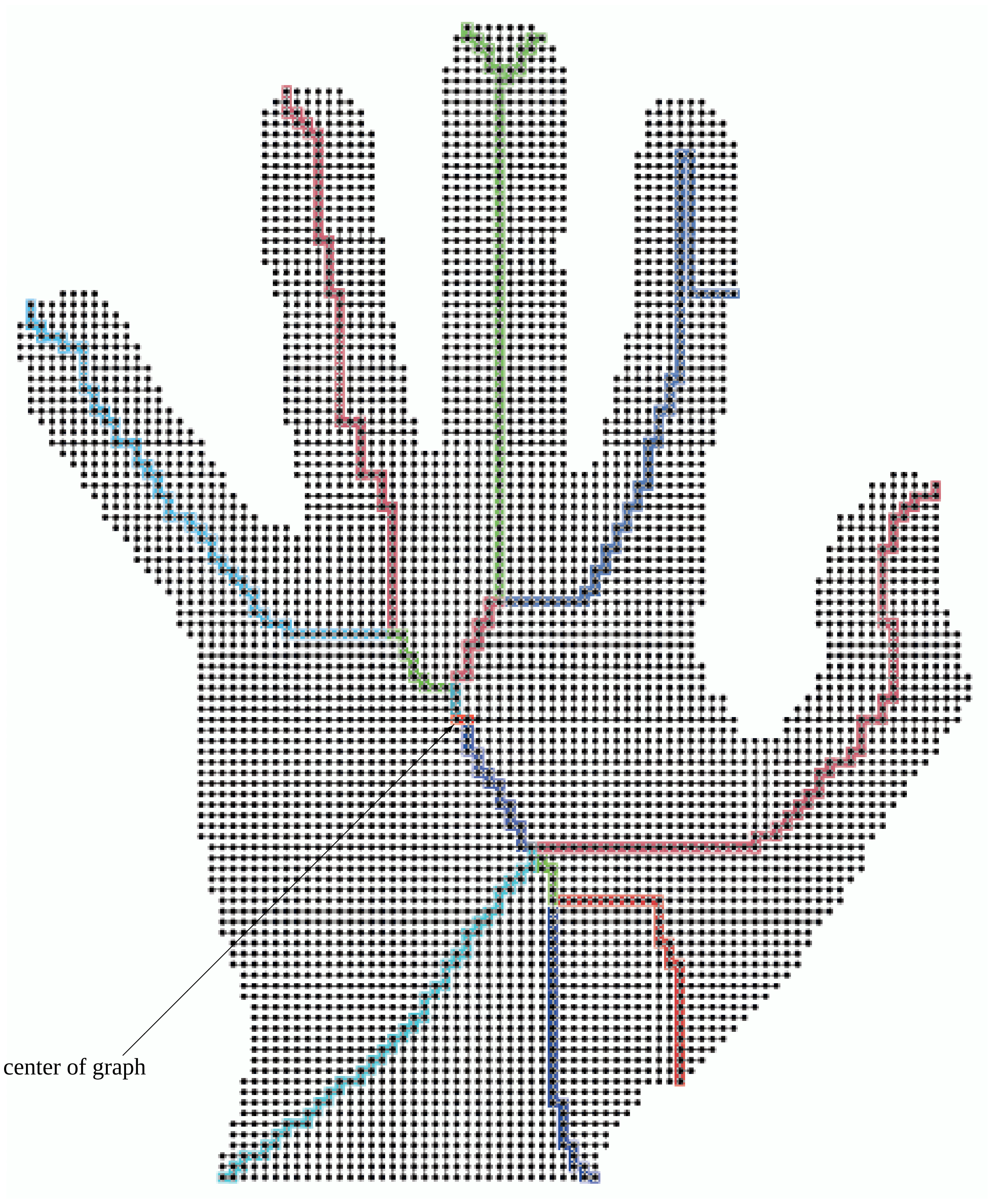


Figure 8: Hand example

Table 5: Cuts through spanning tree of example *'Hand'*

cut	diameters of outer trees and center tree				$\deg((CK))$
d_T	$\max(\delta)$	δ_{left}	$\delta(c)$	δ_{right}	cut
92	182	2, **167**, 23	**1**	**134**, 27	5
91	180	2, **167**, 2×23, 24	**3**	**134**, 27	7
90	170	2, **134**, **126**, 2×23, 2×24	**5**	**134**, 27	9
			...		
77	152	**56**, 14, 16, 18, 21, **126**, 2×23, 3×24, 26, 2×27, 2×28	**31**	**134**, 20, 2×21, 2×22, 47, 3, 1, 27, 2, 21, 22, 2×19, 18, 4×0, **86**	34
			...		
74	146	3×12, **56**, 14, 16, 18, 21, **126**, 2×23, 3×24, 2×25, 26, 2×27, 2×28, 29, 28	**37**	**94**, **73**, 2×20, 2×21, 2×22, 47, 3, 1, 27, 2, 21, 22, 2×19, 18, 4×0, **86**	46
			...		
64	126	9, 10, 11, 6×12, **56**, 14, 16, 18, 21, 22, 21, 20, 19, 17, **60**, 9, **73**, 9, 18, 2×19, 20, 22, 2×23, 3×24, 2×25, 26, 2×27, 2×28, 29, 28	57	**94**, **63**, 2×20, 2×21, 2×22, **47**, 3, 1, 27, 2, 21, 22, 2×19, 18, 9×0, 11, 2×9, 8, **72**	73
63	124	9, 10, 11, 6×12, **56**, 14, 16, 18, 21, 22, 21, 20, 19, 17, **60**, 2×9, **72**, 9, 18, 2×19, 20, 22, 2×23, 3×24, 2×25, 26, 2×27, 2×28, 29, 28	59	**94**, **63**, 2×20, 2×21, 2×22, **47**, 3, 1, 27, 2, 21, 22, 2×19, 18, 9×0, 11, 2×9, 2×8, **71**	76
62	122	8, 9, 10, 11, 6×12, **56**, 14, 16, 18, 21, 22, 21, 20, 19, 17, **60**, 3×9, **71**, 9, 18, 2×19, 20, 22, 2×23, 3×24, 2×25, 26, 2×27, 2×28, 29, 28	61	**94**, **63**, 2×20, 2×21, 2×22, **47**, 3, 1, 27, 2, 21, 22, 2×19, 18, 9×0, 11, 2×9, 2×8, **69**	78
61	120	8, 9, 10, 11, 6×12, **56**, 14, 16, 18, 21, 22, 21, 20, 19, 17, **60**, 4×9, **70**, 9, 18, 2×19, 20, 22, 2×23, 3×24, 2×25, 26, 2×27, 2×28, 29, 28	63	**94**, **63**, 2×20, 2×21, 2×22, **47**, 3, 1, 27, 2, 21, 22, 2×19, 18, 9×0, 11, 2×9, 2×8, 7, **68**	80
60	118	8, 9, 10, 11, 6×12, **56**, 14, 16, 18, 21, 22, 21, 20, 19, 17, **59**, 5×9, **69**, 9, 18, 2×19, 20, 22, 2×23, 3×24, 2×25, 26, 2×27, 2×28, 29, 28	65	**94**, **63**, 2×20, 2×21, 2×22, **47**, 3, 1, 27, 2, 21, 22, 2×19, 18, 9×0, 11, 2×9, 2×8, 2×7, **66**	82
59	116	6, 8, 9, 10, 11, 6×12, **56**, 14, 16, 18, 21, 22, 21, 20, 19, 17, **58**, 6×9, **68**, 9, 18, 2×19, 20, 22, 2×23, 3×24, 2×25, 26, 2×27, 2×28, 29, 28, 29	67	**87**, **70**, **63**, 2×20, 2×21, 2×22, **47**, 3, 1, 27, 2, 21, 22, 2×19, 18, 9×0, 11, 2×9, 2×8, 2×7, **65**	86
			...		

- the depth of all subtrees $d_{\max}$; and
- 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:

- balance the diameters of the subtrees more efficiently than cutting at a fixed distance from the center or the leafs; unfortunately this often generates contraction kernels of high degree.
- set the degree n of the contraction kernel beforehand and find the n subtrees with largest integral feature, e.g. diameter.

- define the optimization criteria 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 articulated motion.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows*. Prentice-Hall, 1993. 3

[2] M. J. Atallah, editor. *Algorithms and Theory of Computational Handbook*. CRC Press, 1999. 3

[3] C. Beleznai, B. Frühstück, H. Bischof, and W. G. Kropatsch. Detecting Humans in Groups using a Fast Mean Shift Procedure. In W. Burger and J. Scharinger, editors, *Digital Imaging in Media and Education, 28th ÖAGM Workshop*, pages 71–78. OCG-Schriftenreihe, Österr. Arbeitsgemeinschaft für Mustererkennung, R. Oldenburg, 2004. Band 179. 2

[4] M. Bister, J. Cornelis, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, Vol. 11(No. 9):pp. 605–617, September 1990. 14

[5] G. Borgefors. Distance transformation in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27:321–145, 1984. 5

[6] G. Borgefors. Distance transformation in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986. 2

[7] C. Bouman and B. Liu. Multiple resolution segmentation of textured images. *IEEE Transactions on Pattern Analysis an Machine Intelligence*, 13:99–113, 1991. 4

[8] J.-M. Chassery and A. Montanvert. *Géométrie discrète en analyse d'images*. Traité des Nouvelles Technologies, série – Images. HERMES, Paris, France, 1991. 2

[9] N. Christofides. *The Traveling Salesman Problem*. John Wiley and Sons, 1985. 3, 4

[10] S. M. Graham, A. Joshi, and Z. Pizlo. The travelling salesman problem: A hierarchical model. *Memory & Cognition*, 28(7):1191–1204, 2000. 4

[11] D. Hochbaum. *Approximation Algorithms for NP-hard Problem*. PWS Publishing, 1997. 3

[12] G. Humphrey. *Directed thinking*. Dodd, Mead, NY, 1948. 4

[13] K. Koffka. *Principles of Gestalt psychology*. Harcourt, NY, 1935. 4

[14] W. G. Kropatsch. Building Irregular Pyramids by Dual Graph Contraction. *IEE-Proc. Vision, Image and Signal Processing*, Vol. 142(No. 6):pp. 366–374, December 1995. 2, 7

[15] W. G. Kropatsch. Equivalent contraction kernels to build dual irregular pyramids. *Advances in Computer Science*, Advances in Computer Vision:pp. 99–107, 1997. 7

[16] W. G. Kropatsch, M. Saib, and M. Schreyer. The Optimal Height of a Graph Pyramid. In F. Leberl and F. Fraundorfer, editors, *Vision with Non-Traditional Sensors 26th ÖAGM Workshop*, pages 87–94. OCG-Schriftenreihe, Österr. Arbeitsgemeinschaft für Mustererkennung, R. Oldenburg, 2002. Band 160. 9, 10, 12

[17] J. B. J. Kruskal. On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem. In *Proc. Am. Math. Soc.*, volume 7, pages 48–50, 1956. 6

[18] E. Lawler, J. Lenstra, A. Rinnoy Kan, and D. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, 1985. 3

[19] R. L. Ogniewicz. *Discrete Voronoi Skeletons*. PhD thesis, ETH Zurich, Switzerland, 1993. Hartung-Gorre Verlag Konstanz. 2

[20] R. L. Ogniewicz. A Multiscale MAT from Voronoi Diagrams: The Skeleton-Space and its Application to Shape Description and Decomposition. In C. Arcelli, L. P. Cordella, and G. Sanniti di Baja, editors, *2nd Intl. Workshop on Visual Form*, pages 430–439, Capri, Italy, June 1994. 2

[21] R. L. Ogniewicz and O. Kübler. Hierarchic voronoi skeletons. *Pattern Recognition*, 28(3):343–359, March 1995. 2, 16

[22] Z. Pizlo. Perception viewed as an inverse problem. *Vision Research*, 41:3145–3161, 2001. 4

[23] Z. Pizlo and Z. Li. Pyramid algorithms as models of human cognition. In *Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging*, pages 5016, 1–12. SPIE, 2003. 4

[24] Z. Pizlo and Z. Li. Graph pyramids as models of human problem solving. In *Proceedings of SPIE-IS&T Electronic Imaging, Computational Imaging*, pages 5299, 205–215. SPIE, 2004. 4

[25] Z. Pizlo, A. Rosenfeld, and J. Epelboim. An exponential pyramid model of the time-course of size processing. *Vision Research*, 35:1089–1107, 1995. 4

[26] Z. Pizlo, M. Salach-Golyska, and A. Rosenfeld. Curve detection in a noisy image. *Vision Research*, 37:1217–1241, 1997. 4

[27] M. Rioux and L. Cournoyer. The NRCC three-dimensional image data files. Technical Report CNRC No. 29077, National Research Council of Canada, Ottawa, Canada, June 1988. 16

[28] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume Vol. 1 and 2. Academic Press, New York, second edition, 1982. 5

[29] G. Sanniti di Baja. Well-Shaped Stable and Reversible Skeletons from the 3-4 Distance Transform. *Journal of Visual Communication and Image Representation*, pages 107–115, 1994. 2

[30] P. Viola and M. Jones. Robust Real-time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. 2

[31] T. Yamada, H. Takahashi, and S. Kataoka. A Heuristic for the $min-max$ Spanning Forest Problem. *European Journal on Operational Research*, 91:565–572, 1996. 4

[32] T. Yamada, H. Takahashi, and S. Kataoka. A $branch-and-bound$ Algorithm for the $min-max$ Spanning Forest Problem. *European Journal on Operational Research*, 91:565–572, 1997. 4