Assessing hierarchies by their consistent segmentations

Zeev Gutman[*]
Rafael
Israel
szeg25@gmail.com

Ritvik Vij
CSE Dept., IIT Delhi
New Delhi, India
Ritvik.Vij.cs517@cse.iitd.ac.in

Laurent Najman
Univ Gustave Eiffel, CNRS, LIGM,
F-77454 Marne-la-Vallée, France
laurent.najman@esiee.fr

Michael Lindenbaum
CS dept., Technion
Haifa, Israel
mic@cs.technion.ac.il

Abstract

Recent segmentation approaches start by creating a hierarchy of nested image partitions, and then specify a segmentation from it, usually, by choosing one horizontal cut. Our first contribution is to describe several different ways, some of them new, for specifying segmentations using the hierarchy regions. Then we consider the best hierarchy-induced segmentation, in which the segments are specified by a limited number, k, of hierarchy nodes/regions. The number of hierarchy-induced segmentations grows exponentially with the hierarchy size, implying that exhaustive search is unfeasible. We focus on a common quality measure, the Jaccard index (known also as IoU). Optimizing the Jaccard index is highly nontrivial. Yet, we propose an efficient optimization approach for exactly optimizing it. We found that the obtainable segmentation quality varies significantly depending on the way that the segments are specified by the hierarchy regions, and that representation of segmentation by only a few hierarchy nodes is often possible. (Code is available).

[*]This work was done when the first author was with the Math dept. Technion, Israel.

1. Introduction

Image segmentation is widely used in various tasks of image analysis and computer vision. A variety of image segmentation methods are proposed in the literature, including the watershed method [1], level-set method [2], normalized cuts [3], and many others. Modern algorithms use (deep) edge detectors and watershed-like merging to provide generic algorithms [4]. Augmenting the detected edges with region descriptors recently led to state-of-the-art segmentations [5]. Other modern algorithm provide segmentation of objects from specific classes (semantic segmentation) with the help of (deep) image classifiers [6, 7]. Segmentation is useful for numerous applications such as image enhancement [8], image analysis [9] and medical image analysis [10].

Many modern generic segmentation algorithms (e.g. [4], are hierarchical and are built as follows: First, an oversegmentation is carried out, specifying superpixels as the elements to be grouped. Then a hierarchical structure (usually represented by a tree) is constructed with the superpixels as its smallest elements (i.e., leaves). The regions specified by the hierarchy are the building blocks from which the final segmentation is decided. Restricting the building blocks to the elements of the hierarchy, yields simple, effective algorithms at low computational cost. The first contribution of the proposed work is to study several ways, some of them new, for specifying the segmentation using some elements of the hierarchy.

In this paper, we are interested in the limitations imposed on the segmentation quality by using the hierarchy-based approach. These limitations depend on (1) the quality of the hierarchy, on (2) the number of hierarchy elements (nodes) that may be used, and on (3) the way that these elements are combined. We investigate all these causes in this paper. The quality is also influenced by the oversegmentation quality, which was studied elsewhere [11].

The number of hierarchy elements is of special interest because it determines the complexity of specifying a segmentation. In a sense, it tells how much information about the segmentation is included in the hierarchy, and thus, it provides a measure for a quality of the hierarchy as an image descriptor.

To investigate these *limitations*, we optimize the segmentation from elements of a given hierarchy. More precisely, we use image dependent, oversegmentation and hierarchies produced by algorithms that have access only to the image. We allow, however, that the final stage, that constructs the segmentation from the hierarchy elements, to have access to the ground truth segmentation. This way the imperfections of the optimized segmentation correspond only to its (hierarchical) input. Thus, the results we get are upper bounds on the quality that may be obtained by any realistic algorithm, that does not have access to the ground truth but relies on the same hierarchy. Note that the proposed method may also be regarded as a method for evaluating the quality of a hierarchy.

Image segmentation is evaluated either relative to ground truth or without it. Quality measures of the first, supervised, types takes a variety of forms; see [12] for a detailed survey of supervised methods.

Here we focus on a supervised quality measure in the context where the image is divided into two parts: foreground (or "object") and background. The quality measure used here is the widely used Jaccard index [13] (intersection over union). Optimizing the Jaccard index directly seems hard. We therefore propose an optimization framework that allows the exact optimization of related new quality measures, and provides a rigorous guarantee that Jaccard index is optimized as well. Earlier studies either use simplistic quality measure or rely on facilitating constraints [14, 15].

The contributions of this work are:

1. A method for evaluating the quality of a hierarchy, by the information provided by few of its nodes about the true segmentation

2. Defining and using four different ways for specifying a hierarchy-induced segmentation. These ways are denoted (segmentation to hierarchy) *consistencies.*

3. A new indirect optimization framework that works for some difficult-to-optimize objectives.

4. Efficient and exact algorithms for optimizing the Jaccard index for all 4 consisten-

cies. The algorithms are fast even for large hierarchies. [1]

5. Experimental results illustrating the dependence of the segmentation quality on the hierarchy quality, the number of hierarchy elements that are used, and the specific way that the hierarchy is used for specifying the segmentation (that is, the consistency).

This paper considers segmentation of images, but all the results apply as well to partition of general data sets. The paper continues as follows. First, we describe terms and notations required for specifying the task (Section 2). In Section 3, we present our goal and discuss the notion of consistencies, that is central to this paper. In Section 4, we review several related works. In Section 5, we develop an indirect optimization approach that rely on the notion of co-optimality, and enables us to optimize certain quality measures. Section 6 provides particular optimization algorithms and the corresponding upper bounds, for Jaccard Index and for the different consistencies. The bounds are evaluated empirically in Section 7, which also provides some typical hierarchy based segmentations. Finally, we conclude and suggest some extensions in Section 8.

2. Preliminaries

2.1. Hierarchies

The following definitions and notations are standard, but are brought here for completeness. Recall that a partition of a set I is a set of non-empty subsets of I, such that every element in I is in exactly one of these subsets (i.e., I is a disjoint union of the subsets). In this paper, these subsets are referred to as regions. Moreover, all examples are done with connected regions, but the connectivity constraint is not needed for the theory and algorithms.

Let π_1 and π_2 be two partitions of a pixel set I. Partition π_1 is *finer* than partition π_2, denoted $\pi_1 \leq \pi_2$, if each region of π_1 is included in a region of π_2. In this case, we also say that π_2 is *coarser* than π_1. Let Π be a finite chain of partitions $\Pi = \{\pi_i \mid 0 \leq i \leq j \leq n \implies \pi_i \leq \pi_j\}$ where π_0 is the finest partition and π_n is the trivial partition of I into a single region: $\pi_n = \{I\}$. A hierarchy $\mathcal{T}$ is a pool of regions of I, called *nodes*, that are provided by elements of Π: $\mathcal{T} = \{ N \subset I \mid \exists \pi_i \in \Pi \ : \ N \in \pi_i \}$. For any two partitions from Π, one is finer than the other, hence, any two nodes $N_1, N_2 \in \mathcal{T}$ are either nested ($N_1 \subset N_2$ or $N_2 \subset N_1$), or disjoint ($N_1 \cap N_2 = \emptyset$); see Figure 1.

A hierarchy $\mathcal{T}$ can be represented by a dendrogram, and every possible partition of I corresponds to a set of $\mathcal{T}$'s nodes and may be obtained by "cutting" the dendrogram; see Figure 1. Every $\pi_i \in \Pi$ is specified by a *horizontal cut* of the dendrogram, but there are many other ways to cut the dendrogram, and each cut specifies a partition of I. In the literature, any partition of I into nodes of $\mathcal{T}$ is called a *cut of the hierarchy* [16, 17]. As we shall see later, a hierarchy may induce other partitions of I.

Let N_1 and N_2 be two different nodes of $\mathcal{T}$. We say that N_1 is the *parent* of N_2 if $N_2 \subset N_1$ and there is no other node $N_3 \in \mathcal{T}$ such that $N_2 \subset N_3 \subset N_1$. In this case, we also say that N_2 is a *child* of N_1. Note that every node has exactly one parent, except $I \in \pi_n$, which has no parent. Hence, for every node $N \in \mathcal{T}$, there is a unique chain: $N = N_1 \subset \ldots \subset N_k = I$, where N_i is the parent of N_{i-1}. Thus, the parenthood relation induces a representation of $\mathcal{T}$ by a tree, in which the nodes of π_0 are the leaves, and the single node of π_n is the root; see Figure 1. Hence, we also refer to $\mathcal{T}$ as a tree, which is equivalent to the dendrogram.

[1]Code is available at https://github.com/ritvik06/-Hierarchy-Based-Segmentation

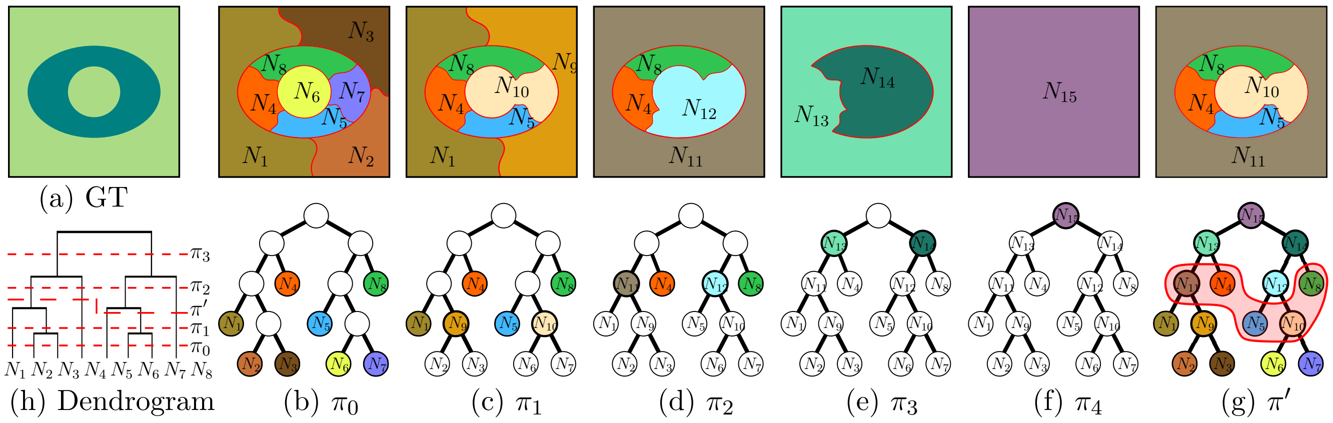


Fig. 1: **(a)** The true segmentation (GT). **(b-f)** A chain of the image partitions: $\Pi = \{\pi_0, \pi_1, \pi_2, \pi_3, \pi_4\}$, which yields a hierarchy $\mathcal{T} = \{N_1, \ldots, N_{15}\}$. Each π_i is represented in the Binary Partition Tree (representing $\mathcal{T}$) by a set of colored nodes. **(g)** Another partition of the image, denoted π'. The nodes representing π' (shaded red) are a cut of the hierarchy $\mathcal{T}$, and are the leaves of a tree $\mathcal{T}'$ obtained by pruning $\mathcal{T}$. **(h)** The dendrogram representing $\mathcal{T}$. Each partition of the above is represented by a cut of the dendrogram (red dashed lines).

[2] When each non-leaf node of $\mathcal{T}$ has exactly two children, $\mathcal{T}$ is a *binary partition tree* (BPT) [18, 19, 14, 15]. Here, we focus on BPTs, but our results hold for non-binary trees as well.

Pruning of a tree in some node N is a removal from the tree of the entire subtree rooted in N, except N itself, which becomes a leaf. Each cut of a hierarchy represents a tree $\mathcal{T}'$ obtained by pruning $\mathcal{T}$, by specifying the leaves of $\mathcal{T}'$; see Figure 1. The converse is also true: the leaves of a tree obtained by pruning $\mathcal{T}$ are a cut of the hierarchy. That is, a subset of nodes $\mathcal{N} \subset \mathcal{T}$ is a cut of the hierarchy, if and only if, $\mathcal{N}$ is the set of leaves of a tree obtained by pruning $\mathcal{T}$. More precisely, $\mathcal{N} \subset \mathcal{T}$ is a cut of the hierarchy, if and only if, for every leaf in $\mathcal{T}$, the only path between it and the root contains exactly one node from $\mathcal{N}$. Often, a segmentation is obtained by searching for the best pruning of $\mathcal{T}$. However, the cardinality of the set of all prunings of $\mathcal{T}$ grows exponentially with the number of leaves in $\mathcal{T}$ [14]. Thus, it is unfeasible to scan this set exhaustively by brute force.

[2]Note that the term node may refer to the node in the tree but also to the corresponding image region, when the context is clear.

2.2. Coarsest partitions

We use the following notations. The cardinality of a set is denoted by $|\cdot|$. The initial partition π_0 of I, which is the set of leaves of the tree $\mathcal{T}$, is denoted by $\mathcal{L}$. Let $N \in \mathcal{T}$, we denote by $\mathcal{T}^N \subset \mathcal{T}$ (resp. $\mathcal{L}^N \subset \mathcal{L}$) the subset of nodes of $\mathcal{T}$ (resp. $\mathcal{L}$) included in N. Note that $\mathcal{T}^N$ is represented by the subtree of $\mathcal{T}$ rooted in N; hence, we refer to $\mathcal{T}^N$ also as a subtree, and to $\mathcal{L}^N$ as the leaves of this subtree.

Let $Y \subset I$ be a pixel subset. We refer to any partition of Y into nodes of $\mathcal{T}$ (namely, a subset of disjoint nodes of $\mathcal{T}$ whose union is Y) as a *$\mathcal{T}$-partition of Y*. Note that a $\mathcal{T}$-partition of Y does not necessarily exist. We refer to the smallest subset of disjoint nodes of $\mathcal{T}$ whose union is Y as the *coarsest $\mathcal{T}$-partition of Y*. Obviously, $\mathcal{N} \subset \mathcal{T}$ is a cut of the hierarchy if and only if, $\mathcal{N}$ is a $\mathcal{T}$-partition of I. $\mathcal{N} \subset \mathcal{T}$ is a $\mathcal{T}$-partition of a node $N \in \mathcal{T}$, if and only if, $\mathcal{N}$ is the set of leaves of a tree obtained by pruning $\mathcal{T}^N$. Obviously, the coarsest $\mathcal{T}$-partition of a node $N \in \mathcal{T}$ is $\{N\}$.

Figure 2 illustrates several ways for representing a region using a hierarchy, and the corresponding coarsest partition.

Property 1. *A non-coarsest $\mathcal{T}$-partition of a node $N \in \mathcal{T}$ is a union of $\mathcal{T}$-partitions of its children.*

In Figure 1(g), for example, the subset $\{N_4, N_5, N_8, N_{10}, N_{11}\}$ is a non-coarsest $\mathcal{T}$-partition of N_{15}, whereas, $\{N_4, N_{11}\}$ and $\{N_5, N_8, N_{10}\}$ are $\mathcal{T}$-partitions of the children of N_{15} : N_{13} and N_{14}, respectively.

Lemma 1. *(See Appendix A for the proof.)*

i. A $\mathcal{T}$-partition of a pixel subset $Y \subset I$ is non-coarsest, if and only if, it contains a non-coarsest $\mathcal{T}$-partition of some node $N \in \mathcal{T}$ that is included in Y ($N \subset Y$).

ii. When the coarsest $\mathcal{T}$-partition of a pixel subset $Y \subset I$ exists, it is unique.

3. Problem formulation

3.1. The general task

As discussed in the introduction, we consider only segmentations that are consistent with a given hierarchy, and aim to find limitations on their quality.

Obviously, if we use a large number of regions, then we can achieve a high quality. Therefore, the quality score improves with the number of regions.

More precisely, the goal of this paper can be stated as follows:
General task: *Given a hierarchy and a measure for estimating segmentation quality, we want to find a segmentation that has the best quality, is consistent with the hierarchy and uses no more than a given number of regions from it.*

To consider this task, we now specify the notion of segmentation consistency with a hierarchy.

3.2. Various types of consistency between a segmentation and a hierarchy

We consider segmentations whose (not necessarily connected) segments are specified by set operations on the nodes of $\mathcal{T}$. Recall that the hierarchy implies that the intersection between two nodes is either empty or one of the nodes. Therefore, we are left with union and set-difference. By further restricting the particular operations and the particular node subsets on which they act, we get different, non-traditional, ways for specifying segmentation from a hierarchy. We denote these different way as (hierarchy to segmentation) consistencies. Complementing (with respect to I) is allowed as well. The following notation is useful: Let $\mathcal{Y}$ be a set of pixel subsets, we denote by $\dot{\mathcal{Y}} \subset I$ the union of all elements of $\mathcal{Y}$. We say that:

(a) A segmentation s is *a-consistent* with $\mathcal{T}$ if there is a subset $\mathcal{N}_s \subset \mathcal{T}$ such that each segment in s is a single node of $\mathcal{N}_s$.

(b) A segmentation s is *b-consistent* with $\mathcal{T}$ if there is a subset $\mathcal{N}_s \subset \mathcal{T}$ such that each segment in s is a union of some nodes of $\mathcal{N}_s$.

(c) A segmentation s is *c-consistent* with $\mathcal{T}$ if there is a subset $\mathcal{N}_s \subset \mathcal{T}$ such that each segment in s, except at most one, is a union of some nodes of $\mathcal{N}_s$. One, complement, segment, if it exists, is $I \backslash \dot{\mathcal{N}}_s$.

(d) A segmentation s is *d-consistent* with $\mathcal{T}$ if there is a subset $\mathcal{N}_s \subset \mathcal{T}$ such that each segment, except at most one, is obtained by unions and/or differences of nodes of $\mathcal{N}_s$. One, complement, segment, if it exists, is $I \backslash \dot{\mathcal{N}}_s$.

Remark 1. *Consistency of some type, with the subset $\mathcal{N}_s$, implies consistency of a later, more general type, with the same subset $\mathcal{N}_s$.*

The a-consistency is in fact used in most hierarchy based segmentation algorithms, where some cut is chosen and all its leafs are specified as segments; see [15, 12]. To our best knowledge, the b-, c- and d- consistencies were not used in the context of hierarchical segmentation; see however [20] for (c-consistency like) node selection in a hierarchy of components.

As specified above, the subset $\mathcal{N}_s$, specified for a segmentation s, is not necessarily unique; see Figures 2 and 3(c,d). From now one, $\mathcal{N}_s$ is considered as the minimal set so that all nodes in it are required to specify s. Then,

Property 2. *If $\mathcal{N}_s \subset \mathcal{T}$ is a subset associated with some consistency type a/b/c/d of a segmentation s, then*

(a) s is a-consistent with $\mathcal{T}$, if and only if, $\mathcal{N}_s$ is a cut of $\mathcal{T}$ such that each segment of s is a single node of $\mathcal{N}_s$. The subset $\mathcal{N}_s$ associated with a-consistency of s is unique.

(b) s is b-consistent with $\mathcal{T}$, if and only if, $\mathcal{N}_s$ is a cut of $\mathcal{T}$.

(c) s is c-consistent with $\mathcal{T}$, if and only if, $\mathcal{N}_s$ consists of disjoint nodes of $\mathcal{T}$.

(d) s is d-consistent with $\mathcal{T}$, if and only if, $\mathcal{N}_s$ consists of (possibly overlapping) nodes of $\mathcal{T}$

Lemma 2. *Every segmentation that is consistent with a hierarchy in one of the types b/c/d is also consistent with the hierarchy in the other two types.*

Proof sketch: Following remark 1, consistency of a segmentation according to one type implies its consistency according to the more general types. The converse is also true.

Consider a d-consistent segmentation s. Recall that every node in $\mathcal{T}$ is a union of disjoint nodes of the initial partition $\mathcal{L}$. A set-difference of nested nodes is still a union of nodes of $\mathcal{L}$ (which are a $\mathcal{T}$-partition of this set-difference); see Figure 2(a-c). Hence, there is a subset $\mathcal{N}_s$ consisting of disjoint nodes. By Property 2, s is also c-consistent.

A subset $\mathcal{N}_s$ consisting of disjoint nodes can be completed to a partition of I by adding some $\mathcal{T}$-partition of $I \backslash \dot{\mathcal{N}}_s$; see Figure 2(d-e). Hence, there is another subset $\mathcal{N}_s$ which is a cut of the hierarchy. By Property 2, s is also b-consistent. □

Lemma 2 provides the somewhat surprising result that b/c/d consistencies are equivalent. Thus, the set of segmentations consistent with $\mathcal{T}$, using either b-, c-, or d- consistencies is common. Denote this set by $\mathcal{S}$. Note that the set of a-consistent segmentations, $\mathcal{S}_1 \subset \mathcal{S}$, is smaller. The consistencies may differ significantly, however, in the $\mathcal{N}_s$ subsets. Let $\mathcal{N}_s^a$ (resp. $\mathcal{N}_s^b$, $\mathcal{N}_s^c$, $\mathcal{N}_s^d$) be the smallest subset such that $s \in \mathcal{S}$ is a- (resp. b-, c-, d-) consistent with this subset

Lemma 3. *Let $s \in \mathcal{S}$, then $|\mathcal{N}_s^b| \geqslant |\mathcal{N}_s^c| \geqslant |\mathcal{N}_s^d|$. Furthermore, $\mathcal{N}_s^b$ is unique, but not necessarily $\mathcal{N}_s^c$, $\mathcal{N}_s^d$. Moreover, if s is a-consistent, then $\mathcal{N}_s^a = \mathcal{N}_s^b$.*

Proof: The proof is straightforward. □

Note that a segmentation is consistent with $\mathcal{T}$ (in some consistency a/b/c/d), if and only if, each of its segments is a union of nodes from $\mathcal{T}$; that is, there is a $\mathcal{T}$-partition for each segment. For a segmentation $s \in \mathcal{S}$, we refer to the union of the coarsest $\mathcal{T}$-partitions of the segments of s (i.e., $\mathcal{N}_s^b$) as *the coarsest cut of the hierarchy for* s. Lemma 3 implies that for every $s \in \mathcal{S}$ there is a unique coarsest cut of the hierarchy. The converse is not true. A cut of a hierarchy can be the coarsest for several segmentations. For example, the same cut is the coarsest for different segmentations (a) and (b) in Figure 3.

4. Previous work

The task considered here (and in [21, 14]) is to estimate the limitation associated with hi-

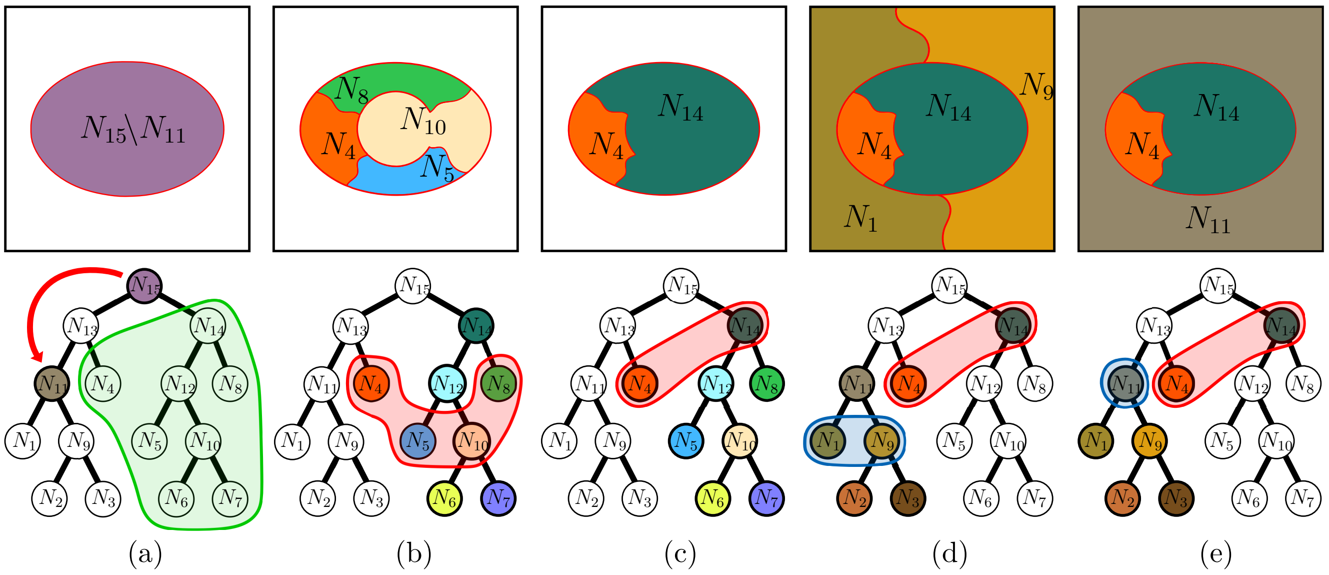


Fig. 2: These examples are related to the hierarchy described in Figure 1. **(a)** Set-difference of nodes: $N_{15} \backslash N_{11}$. The nodes covering the part of I that are included in this set-difference are shaded green. **(b)** A possible $\mathcal{T}$-partition of $N_{15} \backslash N_{11}$ is shaded red. **(c)** The unique coarsest $\mathcal{T}$-partition of $N_{15} \backslash N_{11}$, which is $\{N_4, N_{14}\}$, is shaded red. **(d)** A possible $\mathcal{T}$-partition of the complement $I \backslash (N_4 \cup N_{14})$ is shaded blue. **(e)** The unique coarsest $\mathcal{T}$-partition of the complement $I \backslash (N_4 \cup N_{14})$, which is $\{N_{11}\}$, is shaded blue.

erarchy based segmentation. That is, to find the best $s \in \mathcal{S}$, maximizing the quality $\mathcal{M}(s)$, that is consistent with the hierarchy for a limited size of $\mathcal{N}_s$, $|\mathcal{N}_s| \leqslant k$. This upper-bound of the segmentation quality is a function of the consistency type and of k. We refer to the segmentation maximizing the quality as a/b/c/d-optimal.

First, we emphasize again that this task is different from the common evaluation of hierarchy dependent segmentations, which plots precision recall curves and chooses the best segmentation from them; see, e.g., [22, 12, 23]. This approach considers only the easily enumerable set of horizontal cuts, or segmentations, parameterized by a single scalar parameter. Here, on the other hand, we find the best possible segmentation from much more general segmentation sets, and provide an upper bound on its quality measure. The best segmentations from these larger sets have often significantly better quality; see [23].

Only a few papers address such upper bounds. Most of the upper bounds were derived for local measures. A local measure $\mathcal{M}(s)$ of a segmentation s may be written as a sum of functions defined over the components of the cut defining s.

In [14], local measures are considered. An elegant dynamic programming algorithm provides upper bounds on them for segmentations that are a-consistent with a given BPT hierarchy. Unlike this work, we consider binary segmentation, for which the a-consistent segmentation is trivial. We extend this work by working with b,c and d-consistent segmentation and by optimizing it for a non-local measure: the Jaccard index.

In [15], the considered measure is the boundary-based F_b measure [24]. A method to evaluate the a-consistency performance of a BPT hierarchy is proposed. The optimiza-

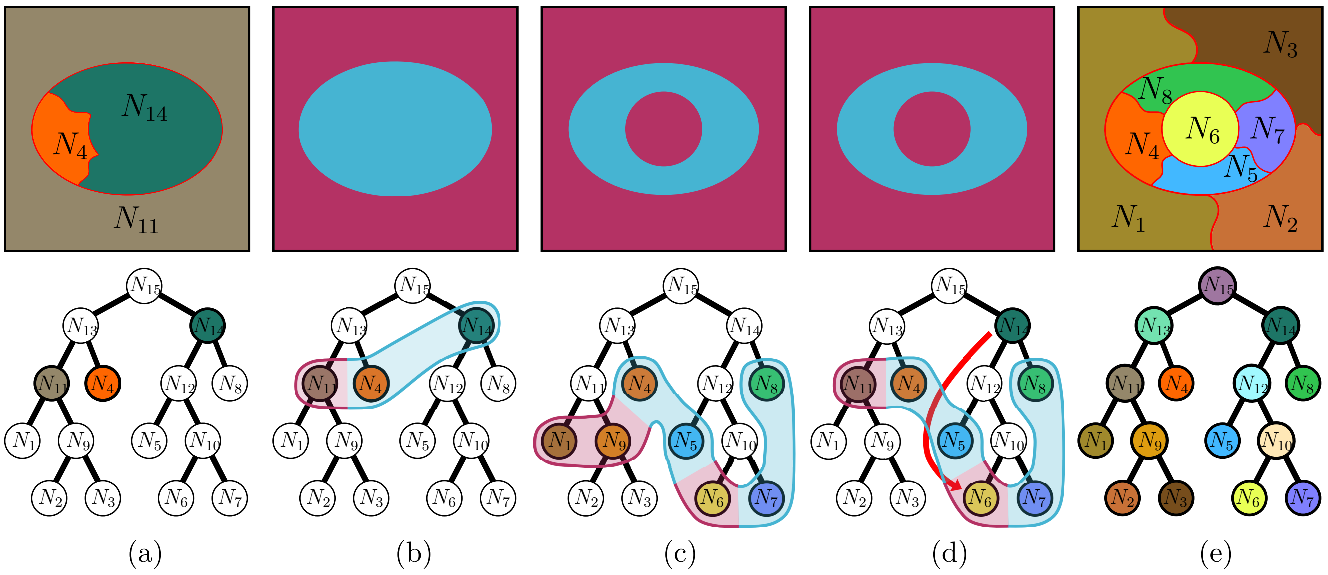


Fig. 3: Examples of segmentations of various consistency types, all consistent with the hierarchy described in Figure 1, shown also in **(e)**. All segmentations are specified by three nodes (although sometimes fewer nodes suffice). Note that a segment is not necessarily connected. Except for the a-consistency, the nodes in a cut of the hierarchy specifying each segmentation are shaded with the colors of the segments in which they are included. **(a)** An a-consistent segmentation, into three segments, specified by a cut of the hierarchy: $\{N_4, N_{11}, N_{14}\}$. **(b)** A segmentation that is b-consistent with the same cut of the hierarchy as in (a). The nodes N_4 and N_{14} are merged into one segment. **(c)** A segmentation, denoted s, that is c-consistent with the subset $\mathcal{N}_s = \{N_1, N_6, N_9\}$. The burgundy segment is $\dot{\mathcal{N}}_s$ and the turquoise segment is the complement $I \backslash \dot{\mathcal{N}}_s$. Note that the $\mathcal{T}$-partition of the burgundy segment is non-coarsest. Hence, $\mathcal{N}_s \neq \mathcal{N}_s^c$ and the specified cut of the hierarchy is non-coarsest for s. The minimal number of disjoint nodes required to cover the turquoise segment is four, while it is only two for the burgundy segment; hence, $\mathcal{N}_s^c = \{N_6, N_{11}\}$. **(d)** The same segmentation s is d-consistent with the subset $\mathcal{N}_s = \{N_4, N_6, N_{14}\}$. The turquoise segment is specified by $N_4 \cup \{N_{14} \backslash N_6\}$, the burgundy segment is the rest of the image: $I \backslash \dot{\mathcal{N}}_s$. The specified cut of the hierarchy is the coarsest for s. Note that representing this segment with $\mathcal{N}_s^c = \{N_6, N_{11}\}$, as specified above, is valid and more node-economical.

tion was modeled as a Linear Fractional Combinatorial Optimization problem [25] and was solved for every possible size of a cut of a hierarchy (from 1 till $|\mathcal{L}|$). This process is computationally expensive, and therefore is limited to moderate size hierarchies.

Extending those previous works, a hierarchy evaluation framework was proposed [23]. It includes various types of upper bounds corresponding to boundaries, regions, and addresses supervised, markers based, segmentation criterion as well. More recently, the study described in [26] introduced some new measures that quantify the match between hierarchy and ground truth.

5. A Co-optimality tool for optimization

Given a quality measure $\mathcal{M}$ over a set $\mathcal{S}$, we want to find $s \in \mathcal{S}$ with the best score, $\mathcal{M}(s)$. Optimizing the quality measures over all possible node subsets is computationally hard. One approach could be to optimize an *equivalent measure* $\mathcal{Q}(s)$ instead. Measures are equivalent if they rank objects identically. For example, the Jaccard index and the object-based F-measure are equivalent [12] because they are functionally related by a monotonically increasing function.

An equivalent measure $\mathcal{Q}$ may, however, be as difficult to optimize. Recalling that we are interested only in the maximum of $\mathcal{M}$ and not in the ranking of all subsets, we may turn to a weaker, easier to optimize, form of equivalence.

Definition 1. *Let $\mathcal{S}_{\mathcal{M}} \subset \mathcal{S}$ be the subset of the elements optimizing $\mathcal{M}$. We refer to measures $\mathcal{M}$ and $\mathcal{Q}$ as* co-optimal *over $\mathcal{S}$, if $\mathcal{S}_{\mathcal{M}} = \mathcal{S}_{\mathcal{Q}}$.*

We now propose an optimization approach that is valid for general finite sets $\mathcal{S}$, including but not limited to hierarchical segmentations. Algorithm 1 uses a family of measures $\{\mathcal{Q}_\omega\}, \omega \in [0,1]$ over $\mathcal{S}$. It works by iteratively alternating between assigning values to ω and optimizing $\mathcal{Q}_\omega(s)$. As theorem 1 below shows, under some conditions on the family $\{\mathcal{Q}_\omega\}$, the algorithm returns the segmentation that maximizes the quality measure $\mathcal{M}$, and the corresponding maximal value $\widehat{\mathcal{M}}$.

Algorithm 1: Generic optimization scheme

```
Data: A quality measure $\mathcal{M}$, a set $\mathcal{S}$,
      and a family of measures $\{\mathcal{Q}_\omega\}$
An initial $\omega_0 \in [0,1]$
Result: An element $s_\omega$
1 $\omega = \mathcal{M}(\operatorname{argmax}_{s \in \mathcal{S}} \mathcal{Q}_{\omega_0}(s))$
2 do
3     $s_\omega = \operatorname{argmax}_{s \in \mathcal{S}} \mathcal{Q}_\omega(s)$
4     $\omega_0 = \omega$
5     $\omega = \mathcal{M}(s_\omega)$
6 while $\omega > \omega_0$
7 return $s_\omega$
```

Theorem 1. *Let $\mathcal{M}$ be a quality measure over a finite set $\mathcal{S}$, receiving its values in $[0,1]$. Let $\widehat{\mathcal{M}}$ be the (unknown) maximal value of $\mathcal{M}$ over $\mathcal{S}$. Let $\{\mathcal{Q}_\omega\}, \omega \in [0,1]$ be a family of measures over $\mathcal{S}$, satisfying the following conditions:*

1. *$\mathcal{Q}_{\omega=\widehat{\mathcal{M}}}$ and $\mathcal{M}$ are co-optimal measures over $\mathcal{S}$.*
2. *For $0 \leqslant \omega < \widehat{\mathcal{M}}$ and $s' \in \mathcal{S}$, if there is $s \in \mathcal{S}_{\mathcal{M}}$ s.t. $\mathcal{Q}_\omega(s) \leqslant \mathcal{Q}_\omega(s')$, then $\mathcal{M}(s') > \omega$.*

Then Algorithm 1 returns $s \in \mathcal{S}_{\mathcal{M}}$ after a finite number of iterations.

Proof. The proof is in two parts. First, suppose that $\omega_0 \in [0, \widehat{\mathcal{M}}]$. Then the iterative scheme in each iteration finds $s_\omega \in \mathcal{S}_{\mathcal{Q}_\omega}$ and specifies a new value for ω to be $\mathcal{M}(s_\omega)$. Condition 2 is fulfilled trivially for $s' = s_\omega$ since s_ω maximizes $\mathcal{Q}_\omega$. Hence, $\omega < \mathcal{M}(s_\omega)$; i.e., ω strictly increases from iteration to iteration while $\omega < \widehat{\mathcal{M}}$. $\mathcal{S}$ is finite, hence, ω reaches $\widehat{\mathcal{M}}$ after a finite number of iterations. When that happens, $\hat{s}_{\omega=\widehat{\mathcal{M}}} \in \mathcal{S}_{\mathcal{M}}$ since, by condition 1, $\mathcal{Q}_{\omega=\widehat{\mathcal{M}}}$ and $\mathcal{M}$ are co-optimal. The iterations stop when ω no longer increases. Hence, to prove the theorem, we show that ω does not change after it reaches $\widehat{\mathcal{M}}$.

$\Longrightarrow$ Suppose that $\omega = \widehat{\mathcal{M}}$. Since $\hat{s}_{\omega=\widehat{\mathcal{M}}}$ maximizes both $\mathcal{M}$, $\mathcal{Q}_{\omega=\widehat{\mathcal{M}}}$, we have $\widehat{\mathcal{M}} = \mathcal{M}(s_\omega) \quad \Rightarrow \quad \omega = \widehat{\mathcal{M}} = \mathcal{M}(s_\omega)$.

$\Longleftarrow$ Conversely, suppose that $\omega = \mathcal{M}(s_\omega)$, i.e., ω does not change at line 5 of the algorithm. All values of ω specified in scheme 1 are values of $\mathcal{M}$; hence, $\omega \leqslant \widehat{\mathcal{M}}$. If $\omega < \widehat{\mathcal{M}}$, then by condition 2 $\omega < \mathcal{M}(s_\omega)$, which contradicts the current assumption. Hence, $\omega = \widehat{\mathcal{M}}$.

(Second part of the proof:) If the initial assumption $\omega_0 \in [0, \widehat{\mathcal{M}}]$ is not satisfied (i.e.,

$\omega_0 > \widehat{\mathcal{M}}$). Then, line 1 returns some ω which must be lower than the maximal $\widehat{\mathcal{M}}$. Then the algorithm proceeds and reaches the optimum according to the proof above. $\square$

Given a quality measure $0 \leqslant \mathcal{M} \leqslant 1$ over $\mathcal{S}$, we refer to a family $\{\mathcal{Q}_\omega\}, \omega \in [0,1]$ of measures over $\mathcal{S}$, as a *family of auxiliary measures for* $\mathcal{M}$ if $\{\mathcal{Q}_\omega\}$ contains at least one measure $\mathcal{Q}_{\omega'}$ that is co-optimal with $\mathcal{M}$ over $\mathcal{S}$, and there is some iterative process that finds $\mathcal{Q}_{\omega'}$ from $\{\mathcal{Q}_\omega\}$. We refer to $\mathcal{Q}_{\omega'}$ as a *co-optimal auxiliary measure*, and we refer to an algorithm that can optimize every member of $\{\mathcal{Q}_\omega\}$ as an *auxiliary algorithm.*

Remark 2. *In scheme 1, the auxiliary algorithm is written in the most general form:* argmax $\mathcal{Q}_\omega$. *In the sequel, it will be used for optimizing the Jaccard index of a segmentation. The scheme and the proof hold also when the optimization is constrained, e.g., by a consistency type and a constraint on the size of $\mathcal{N}_s$.*

6. Optimizing the Jaccard Index

We shall now turn to the main goal of this paper: optimizing a specific quality measure. the Jaccard Index.

6.1. The Jaccard index

The Jaccard index (or the intersection over union measure) is a popular segmentation quality measure, applicable to a simple segmentation into two parts: foreground (or object) and background.

Let $(\mathcal{B}_{GT}, \mathcal{F}_{GT})$ and $(\mathcal{B}_s, \mathcal{F}_s)$ be the two foreground-background partitions corresponding to the ground-truth and the segmentation $s \in \mathcal{S}$. The Jaccard index J is given by:

$$J(s) = \frac{TP}{TP+FN+FP} = \frac{|\mathcal{F}_{GT} \cap \mathcal{F}_s|}{|\mathcal{F}_{GT} \cup \mathcal{F}_s|} \quad (1)$$

Given a hierarchy, we shall find, for each consistency and node subset size $|\mathcal{N}_s|$, the segmentation that maximizes the Jaccard index. This segmentation also maximizes the object-based F-measure, as the two measures are equivalent [12].

For two part segmentation, there is only one segmentation which is a-consistent with a BPT hierarchy: the two children of the root. We ignore this trivial case in the following discussion.

6.2. Segmentation dimensions

Let $\mathcal{S}^2 \subset \mathcal{S}$ be the subset of all possible 2-segment segmentations, consistent with the hierarchy. Denote the areas of the ground-truth parts by $B = |\mathcal{B}_{GT}|$, $F = |\mathcal{F}_{GT}|$. Let $\mathcal{X}_s$ be one segment of a segmentation $s \in \mathcal{S}^2$. Considering this segment as foreground, denote its areas inside the ground-truth's parts by $(b_s = |\mathcal{X}_s \cap \mathcal{B}_{GT}|, f_s = |\mathcal{X}_s \cap \mathcal{F}_{GT}|)$. The Jaccard index is then

$$J(s) = \frac{|\mathcal{F}_{GT} \cap \mathcal{X}_s|}{|\mathcal{F}_{GT} \cup \mathcal{X}_s|} = \frac{f_s}{F+b_s} = \Psi(b_s, f_s). \quad (2)$$

Alternatively, the foreground can be specified by the complementary segment $I \backslash \mathcal{X}_s$. The corresponding areas inside the ground-truth's parts are $(B-b_s, F-f_s)$. The Jaccard index associated with this foreground is

$$J^c(s) = \Psi(B-b_s, F-f_s) = \frac{F-f_s}{F+B-b_s}. \quad (3)$$

Optimizing $J(s)$ for b-consistency, provides a cut in tree $\mathcal{N}_s$. Both $\mathcal{F}_s$ and $\mathcal{B}_s$ are unions of nodes of this cut. The c/d consistencies allow one segment to be specified as the complement of the other. The hierarchy may match better either $\mathcal{F}_{GT}$ or $\mathcal{B}_{GT}$. Thus, we optimize both $J(s)$, $J^c(s)$ (for the same size of $\mathcal{N}_s$) and choose the better result.

The values (b_s, f_s) are the main optimization variables, and we refer to them as *segmentation dimensions.*

6.3. Applying co-optimality for optimizing $J(s)$

6.3.1 Optimizing $J(s)$

Our goal is to find

$$(\,\widehat{J}\,,\,\widehat{s}\,) = (\,\max_{s\in\mathcal{S}^2}\,,\,\operatorname{argmax}\,)\,J(s). \quad (4)$$

A key idea is to observe that the $J(s)$ value may be interpreted geometrically using the graph of segmentation dimensions (b, f). Selecting the segment $\mathcal{X}_s$, every segmentation $s \in \mathcal{S}^2$ corresponds to a point (b_s, f_s) inside the rectangle $(0, B)\times(0, F)$. $J(s)$ is $tan(\alpha_s)$, where α_s is the angle between the b axis and the line connecting the point (b_s, f_s) with the point (-F, 0) ; see Figure 4. The geometry implies that $tan(\alpha_s) \in [0, 1]$, consistently with $tan(\alpha_s) = J(s) \in [0, 1]$.

6.3.2 A family of auxiliary measures for $J(s)$

For every $\omega \in [0, 1]$, let

$$P_{\omega}(s) = f_s - \omega \cdot b_s \quad (5)$$

be a measure over $\mathcal{S}^2$. Note that, geometrically, $P_{\omega}(s)$ is the oblique projection at the $\arctan(\omega)$ angle of point (b_s, f_s) onto the f axis. The following two observations imply that $J(s)$ and the projection (at $\arctan(\widehat{J}\,)$ angle) $P_{\widehat{J}}(s)$ are co-optimal measures.

1. $J(s)$ and $P_{J(s)}(s)$ are equivalent measures.
2. $P_{J(s)}(s)$ and $P_{\widehat{J}}(s)$ are co-optimal measures.

The first observation is clear: ranking the elements of $\mathcal{S}^2$ by $J(s) = tan(\alpha_s)$ is equivalent to ranking them by their projection at angle α_s , i.e., by $P_{J(s)}(s)$; see Figure 4.

The second observation states that there is a constant angle $\arctan(\omega)$ with $\omega = \widehat{J}$,(not depending on s) s.t. the projection $P_{\omega}(s)$ at this angle and $P_{J(s)}(s)$ are co-optimal. By the first observation, $P_{J(s)}(s)$ is maximized by $\widehat{s}$. Every non-optimal segmentation corresponds to a point below the line $[(-F, 0) - (b_{\widehat{s}}, f_{\widehat{s}})]$ and its constant angle projection satisfies $P_{\omega}(s) < P_{\omega}(\widehat{s})$. $P_{\omega}(s)$ is maximized only by points lying on this line, as is also the case with $P_{J(s)}(s)$.

Thanks to these two observations, the family $\{P_{\omega}\}$ is a family of auxiliary measures. The optimization process, as given in Algorithm 1, uses a maximization of this auxiliary measure:

$$(\,\widehat{P}_{\omega}\,,\,\widehat{s}_{\omega}\,) = (\,\max_{s\in\mathcal{S}^2}\,,\,\operatorname{argmax}\,)\,P_{\omega}(s) \quad (6)$$

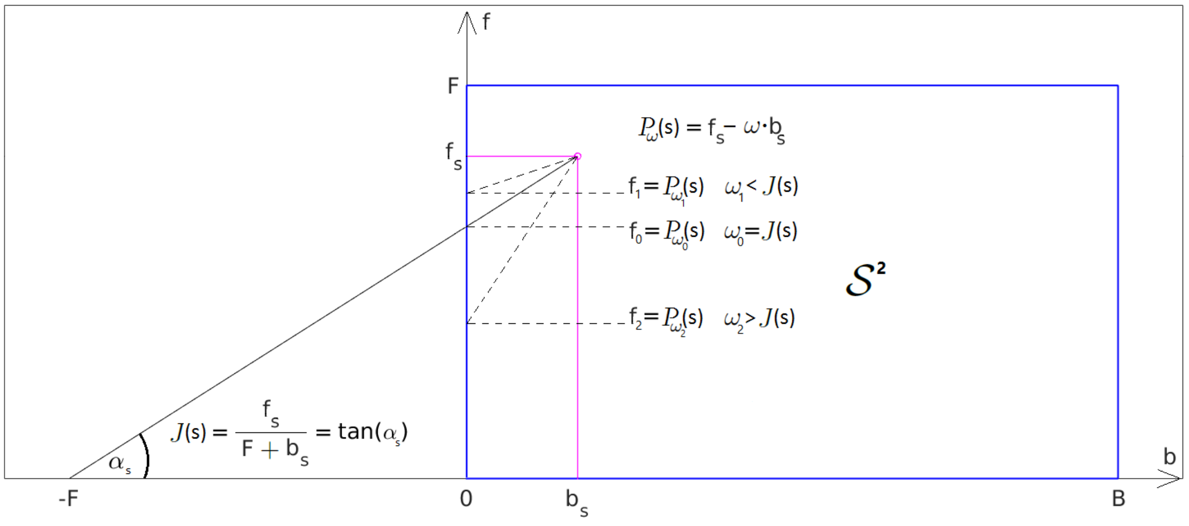
Note that $P_{\widehat{J}}(s)$ is linear in (b_s, f_s) , while $J(s)$ is not, which make its maximization easier. The value $\widehat{J}$, is, however, unknown. To use scheme 1 to find $\widehat{s}$, its second condition, which guarantees that ω strictly increases at every iteration while $\omega < \widehat{J}$, must be met as well. Figure 5, geometrically proves this property. Indeed, let $\omega \in [0, \widehat{J})$ and $\widehat{s}, s' \in \mathcal{S}^2$ such that $P_{\omega}(\widehat{s}) \leqslant P_{\omega}(s')$. Observe that the angle $\alpha_{s'}$ must be larger than the projection angle of P_{ω} , i.e., $\arctan(\omega)$. The detailed proof is left to the reader. Therefore, by theorem 1

Theorem 2. *For $\mathcal{M} = J$, $\{\mathcal{Q}_{\omega}\} = \{P_{\omega}\}$, and $\omega \in [0, 1]$, scheme 1 (starting from $\omega_0 \in [0, \widehat{J}]$) returns the best segmentation $\widehat{s}$ after a finite number of iterations.*

Remark 3. *Optimizing $J^c(s)$ is similarly done.*

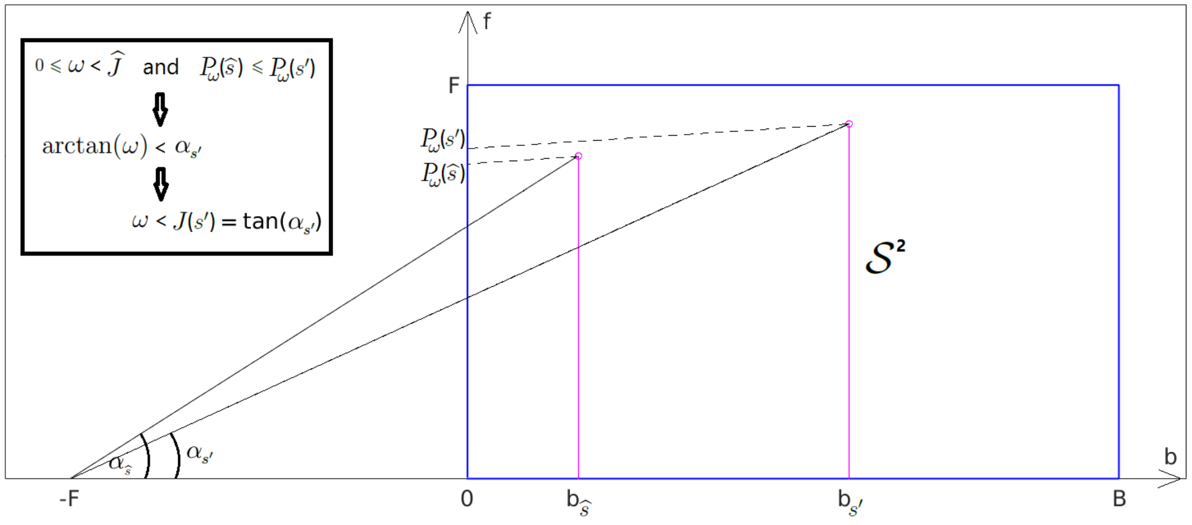
6.4. Optimizing $J(s)$ for hierarchical segmentation

Using scheme 1 for optimization reduces the original optimization process to optimizations carried out by the auxiliary algorithms. The auxiliary algorithm provides a foreground-background segmentation $s \in \mathcal{S}^2$ whose dimensions (b_s, f_s) maximize the auxiliary measure corresponding to the current iteration. In this work we use the hierarchy for this optimization and the auxiliary algorithm returns the best segmentation $s \in \mathcal{S}^2$ together with the corresponding subset $\mathcal{N}_s \subset \mathcal{T}$, which both depend on the required consistency of s with $\mathcal{N}_s$.



f
F
f_s
$P_\omega(s) = f_s - \omega \cdot b_s$
$f_1 = P_{\omega_1}(s)$ $\omega_1 < J(s)$
$f_0 = P_{\omega_0}(s)$ $\omega_0 = J(s)$
$f_2 = P_{\omega_2}(s)$ $\omega_2 > J(s)$
$\mathcal{S}^2$
$J(s) = \frac{f_s}{F + b_s} = \tan(\alpha_s)$
α_s
b
-F
0
b_s
B

Fig. 4: A geometrical interpretation: the Jaccard index $J(s)$ is the tangent of the angle α_s.



$0 \leqslant \omega < \widehat{J}$ and $P_\omega(\widehat{s}) \leqslant P_\omega(s')$
$\arctan(\omega) < \alpha_{s'}$
$\omega < J(s') = \tan(\alpha_{s'})$
f
F
$P_\omega(s')$
$P_\omega(\widehat{s})$
$\mathcal{S}^2$
$\alpha_{\widehat{s}}$
$\alpha_{s'}$
b
-F
0
$b_{\widehat{s}}$
$b_{s'}$
B

Fig. 5: An illustration showing that ω strictly increases from iteration to iteration, while $\omega < \widehat{J}$.

6.4.1 Specifying $\mathcal{N}_s$ and the segmentation s for various consistencies

Here, we specify the relation between the hierarchy and the segmentation for each of the different consistencies considered in this paper.

a-consistency: Trivial and not considered here, as discussed above.

b-consistency: $\mathcal{N}_s$ is a partition of I. A segmentation s is specified by assigning some nodes of $\mathcal{N}_s$ to the foreground $\mathcal{F}_s$, and the rest to the background $\mathcal{B}_s$.

c-consistency: The nodes of $\mathcal{N}_s$ are disjoint, but their union is not the full image. Recall that $\dot{\mathcal{N}}_s$ stands for the union of the regions corresponding to the nodes in $\mathcal{N}_s$. The segments of s are $\dot{\mathcal{N}}_s$ and the complement $I \backslash \dot{\mathcal{N}}_s$.

d-consistency: Not all nodes of $\mathcal{N}_s$ are necessarily disjoint, and their union is not necessarily the full image. The segments of s are specified as follows:

Let $\mathcal{N} \subset \mathcal{T}$ be a subset of nodes. Because the nodes belong to a hierarchy, each pair of nodes is either disjoint or nested. Denote by $\mathcal{K}^0_{\mathcal{N}} \subset \mathcal{N}$ the subset of disjoint nodes that are not nested in any other node from $\mathcal{N}$. Recursively, denote by $\mathcal{K}^i_{\mathcal{N}} \subset \mathcal{N}$ the subset of disjoint nodes that are not nested in any other node from $\mathcal{N} \setminus \{\cup_{j=0}^{i-1} \mathcal{K}^j_{\mathcal{N}}\}$. We refer to each $\mathcal{K}^i_{\mathcal{N}}$ as a *layer of* $\mathcal{N}$. Note that $\dot{\mathcal{K}}^i_{\mathcal{N}} \subset \dot{\mathcal{K}}^j_{\mathcal{N}}\ \forall i > j$ (each subsequent layer is nested in any previous layer); hence, $\dot{\mathcal{K}}^0_{\mathcal{N}} = \dot{\mathcal{N}}$. Let $i^N_{\mathcal{N}}$, be the index of the layer in which the node N lies. Note that the set of layers is a partition of $\mathcal{N}$, i.e., every node $N \in \mathcal{N}$, is associated with exactly one index $i^N_{\mathcal{N}}$. Note that $i^N_{\mathcal{N}}$ is the number of nodes in $\mathcal{N}$ in which N is nested. Let $i^{max}_{\mathcal{N}}$ be the largest index corresponding to a nonempty layer. The segmentation is specified from

$$\mathcal{D}_{\mathcal{N}} = \left\{ D^i_{\mathcal{N}} \mid D^i_{\mathcal{N}} = \dot{\mathcal{K}}^{2\cdot i}_{\mathcal{N}} \setminus \dot{\mathcal{K}}^{2\cdot i+1}_{\mathcal{N}}, \quad 0 \leqslant i \leqslant \left\lfloor \frac{i^{max}_{\mathcal{N}}}{2} \right\rfloor \right\} \qquad (7)$$

Each $D^i_{\mathcal{N}}$ is the set-difference of the layers $2 \cdot i$ and $2 \cdot i + 1$. Since each subsequent layer is nested in any previous layer, all $D^i_{\mathcal{N}}$ are disjoint. The segments of s are $\dot{\mathcal{D}}_{\mathcal{N}_s}$ and the complement $I \setminus \dot{\mathcal{D}}_{\mathcal{N}_s}$; see Figure 6.

6.4.2 Calculation of segmentation dimensions for various consistencies

To calculate the segmentation dimensions (b_s, f_s), efficiently, we use a tree data structure to represent the tree hierarchy. For each node N of the tree we store the area of the node inside the ground truth's parts ($b^N = |N \cap \mathcal{B}_{GT}|$, $f^N = |N \cap \mathcal{F}_{GT}|$). Similarly to the *segmentation dimensions*, specified above, we refer to these values as *node dimensions*. Note that the dimensions of a union of disjoint nodes are equal to the sum of the dimensions of all nodes from the union, and the dimensions of a set-difference of two nested nodes are equal to the difference of their dimensions. Given a segmentation $s = (\mathcal{X}_s, I \setminus \mathcal{X}_s) \in \mathcal{S}^2$, the calculation of its dimensions (b_s, f_s) (which are the dimensions of the segment $\mathcal{X}_s$) from the dimensions of the nodes of a subset $\mathcal{N}_s$ depends on the required consistency of s with $\mathcal{N}_s$:

b-consistency: $(\mathcal{X}_s = \mathcal{F}_s)$. (b_s, f_s) are calculated by the sum of the dimensions of the nodes assigned to $\mathcal{F}_s$.

c-consistency: $(\mathcal{X}_s = \dot{\mathcal{N}}_s)$. (b_s, f_s) are calculated by the sum of the dimensions of $\mathcal{N}_s$.

d-consistency: $(\mathcal{X}_s = \dot{\mathcal{D}}_{\mathcal{N}_s})$. By the observations above about the sums and difference of dimensions and Eq. (7), the dimensions (b_s, f_s) are calculated by the sum of the dimensions of all nodes from $\mathcal{N}_s$, each multiplied by an appropriate sign: -1 to the power of $i^N_{\mathcal{N}_s}$. More formally, we can write (b_s, f_s) as the expression below. Indeed, $\mathcal{N}_s$ consists of a single layer: $i^N_{\mathcal{N}_s} = 0\ \forall N \in \mathcal{N}_s$ for the b/c consistencies. Therefore, this expression is valid for all (b/c/d) consistencies.

A unified expression of segmentation dimensions:

$$b_s = \Bigg(\sum_{\substack{N \in \mathcal{N}_s: \\ \mathcal{N}_s \text{ specifies } \mathcal{X}_s}} b^N \cdot (-1)^{i^N_{\mathcal{N}_s}} \Bigg) \qquad (8a)$$

$$f_s = \Bigg(\sum_{\substack{N \in \mathcal{N}_s: \\ \mathcal{N}_s \text{ specifies } \mathcal{X}_s}} f^N \cdot (-1)^{i^N_{\mathcal{N}_s}} \Bigg) \qquad (8b)$$

Remark 4. *Since for a segmentation $s \in \mathcal{S}^2$ the subset $\mathcal{N}_s$ is not necessarily unique, we could ask whether the expression (8) is well-defined, i.e., whether we get the same area ($b_s = |\mathcal{X}_s \cap \mathcal{B}_{GT}|$, $f_s = |\mathcal{X}_s \cap \mathcal{F}_{GT}|$) for different subsets $\mathcal{N}_s$. The answer to this question is positive, due to the properties of node dimensions for the union of disjoint nodes and for the set-difference of nested nodes.*

6.4.3 Auxiliary measures additivity

A particularly useful property of the auxiliary measures is their additivity. Consider some at-

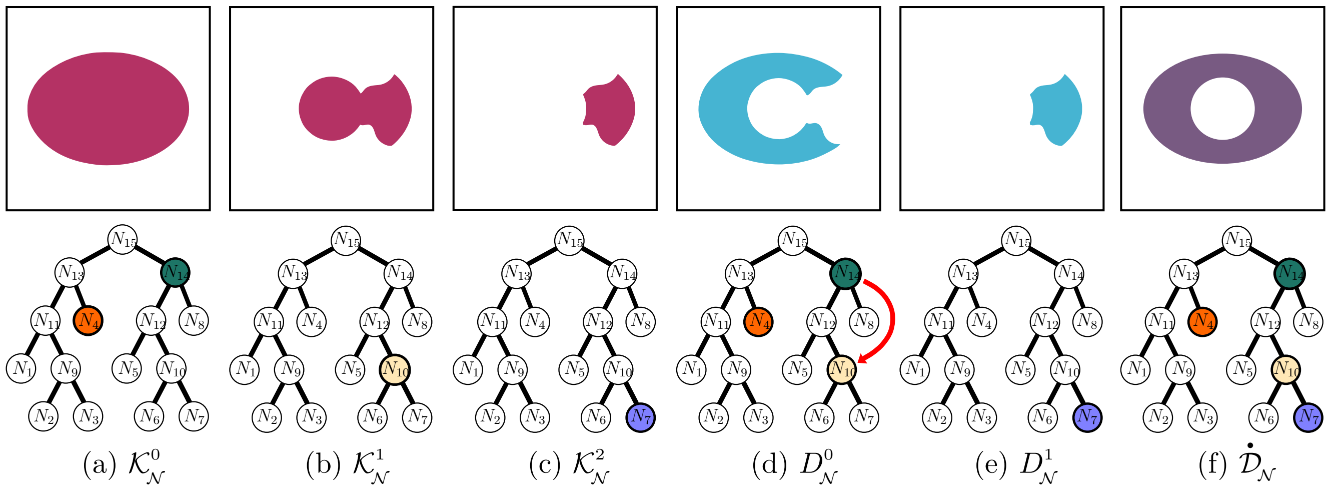

(a) $\mathcal{K}_{\mathcal{N}}^{0}$
(b) $\mathcal{K}_{\mathcal{N}}^{1}$
(c) $\mathcal{K}_{\mathcal{N}}^{2}$
(d) $D_{\mathcal{N}}^{0}$
(e) $D_{\mathcal{N}}^{1}$
(f) $\dot{\mathcal{D}}_{\mathcal{N}}$
N_{15} N_{13} N_{14} N_{11} N_4 N_{12} N_8 N_1 N_9 N_5 N_{10} N_2 N_3 N_6 N_7

Fig. 6: Specification of a segmentation that is d-consistent with the subset $\mathcal{N}=\{N_4,N_7,N_{10},N_{14}\}$ consisting of nodes from the hierarchy described in Figure 1. The segments are $\dot{\mathcal{D}}_{\mathcal{N}}$ and the complement $I\backslash\dot{\mathcal{D}}_{\mathcal{N}}$. Here maximal non-empty layer correspond to the index $i_{\mathcal{N}}^{max}=2$. The layers are: **(a)** $\mathcal{K}_{\mathcal{N}}^{0}=\{N_4,N_{14}\}$ **(b)** $\mathcal{K}_{\mathcal{N}}^{1}=\{N_{10}\}$ **(c)** $\mathcal{K}_{\mathcal{N}}^{2}=\{N_7\}$. The set differences between subsequent layers are **(d)** $D_{\mathcal{N}}^{0}=\mathcal{K}_{\mathcal{N}}^{0}\backslash\mathcal{K}_{\mathcal{N}}^{1}$ **(e)** $D_{\mathcal{N}}^{1}=\mathcal{K}_{\mathcal{N}}^{2}\backslash\varnothing$. The final segmentation is specified by **(f)** $\dot{\mathcal{D}}_{\mathcal{N}}=D_{\mathcal{N}}^{0}\cup D_{\mathcal{N}}^{1}$; see section 6.4.1.

tribute defined on every node in the tree. If the attribute of each non-leaf node is the sum of the attributes of the node's children, then we say that this attribute is *additive*.

For a specific projection P_ω, the two dimensions of a node may be merged into one attribute, $A^{P_\omega}(N)=f^N-\omega\cdot b^N$. By inserting eq. (8a) and eq. (8b) into $P_\omega(s)=f_s-\omega\cdot b_s$ (eq. (5)), we get a closed form, simplified linear expression for the auxiliary measure of the segmentation s. We may refer to this measure, alternatively, as the *benefit* of the corresponding node set $\mathcal{N}_s$:

$$P_\omega(s)=B[\mathcal{N}_s]=\sum_{\substack{N\in\mathcal{N}_s:\\ \mathcal{N}_s \text{ specifies } \mathcal{X}_s}} A^{P_\omega}(N)\cdot(-1)^{i_{\mathcal{N}_s}^{N}} \qquad (9)$$

Note that each non-leaf node N is the union of its disjoint children; hence, the dimensions (b^N,f^N) are the sum of the dimensions of the children of N, which implies the additivity for $A^{P_\omega}(N)$. The additivity property holds for all projections, to which we simply refer as the attribute of N, $A(N)$.

The auxiliary algorithms search for the subset of nodes maximizing benefit (eq. (9)). These optimization tasks are performed under the constraint: $|\mathcal{N}_s|\leqslant k$.

While eq. (9) provide a general expression for all consistencies, in practice we use the following consistency-dependent expressions, which are equivalent and more explicit.

Property 3. *(equivalent benefit expressions)*

b-consistency: *$\mathcal{N}$ is a partition of I and* $B[\mathcal{N}]=\sum_{\substack{N\in\mathcal{N}:\\ A(N)>0}} A(N)$

c-consistency: *$\mathcal{N}$ consists of disjoint nodes and* $B[\mathcal{N}]=\sum_{N\in\mathcal{N}} A(N)$

d-consistency: $B[\mathcal{N}]=\sum_{N\in\mathcal{N}} A(N)\cdot(-1)^{i_{\mathcal{N}}^{N}}$

Here and below we prefer the more general $\mathcal{N}$ (over $\mathcal{N}_s$) when the discussion applies to general sets of nodes from the tree.

The proposed auxiliary algorithms (described below) are not restricted to the auxiliary measures discussed above, they would work for any additive measure $\mathcal{Q}$. The additivity is crucial, because otherwise the score $\mathcal{Q}(s)$ is ill-defined, i.e., it may result in different score values for different subsets $\mathcal{N}_s$ specifying the same $s \in \mathcal{S}^2$.

6.4.4 Using the tree structure for maximizing the auxiliary measures

Thus, the maximization of the benefit (property 3) results in a subset of nodes subject to the consistency constraints, with the maximal benefit in $\mathcal{T}$. The key observation in this maximization is that a subset with the maximal benefit in a subtree $\mathcal{T}^N$ can be obtained from subsets with the maximal benefit in the subtrees of children of N. That is, we can use the recursive structure of the tree $\mathcal{T}$ to maximize the benefit.

Let $\mathcal{N}' \subset \mathcal{N} \subset \mathcal{T}$, we say that $\mathcal{N}'$ is best if it has the highest benefit relative to every other subset of $\mathcal{N}$ with the same number of nodes. Depending on the context, $\mathcal{N}'$ should also have the properties associated with the consistency; i.e., being a partition (for b-consistency) or belong to a single layer (c-consistency). $\mathcal{N}'$ is worst if it has the minimal benefit relative to other subsets of $\mathcal{N}$ of the same size.

Remark 5. *Note that within the same consistency type there can be several best/worst subsets in $\mathcal{N}$, having the same benefit but not necessarily of the same size.*

Thus, a subset $\mathcal{N}$ maximizes the benefit (property 3), if and only if, $\mathcal{N}$ is a best subset in $\mathcal{T}$. Below, by referring to $\mathcal{N}$ as best without specifying in which subset of $\mathcal{T}$ the subset $\mathcal{N}$ is best, we mean that $\mathcal{N}$ is best in the entire $\mathcal{T}$.

The following claim readily follows from the dimensions properties.

Lemma 4. *(a) Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be subsets of nodes, such that $\dot{\mathcal{N}}_1$ and $\dot{\mathcal{N}}_2$ are disjoint, then: $B[\mathcal{N}_1 \cup \mathcal{N}_2] = B[\mathcal{N}_1] + B[\mathcal{N}_2]$*

(b) Let N be a node and $\mathcal{N}$ be a subset of nodes, such that $\dot{\mathcal{N}} \subset N$ then: $B[\{N\} \cup \mathcal{N}] = A(N) - B[\mathcal{N}]$

Lemma 4(b) applies only to the d-consistency. Indeed, $\dot{\mathcal{N}}$ and N are not disjoint. Therefore, the set of nodes $\{N\} \cup \mathcal{N}$ corresponds to a segment that is the set difference between N and the segment specified by $\mathcal{N}$, which leads to the claim on the benefit.

The children of a non-leaf node are disjoint and nested in the node, which implies the following claim.

Lemma 5. *Let $N \in \mathcal{T}$ be a non-leaf node: $N = N_r \cup N_l$, where N_r (right), N_l (left) are its children. Let $\mathcal{N}^N$ be a subset of $\mathcal{T}^N$ and let $\mathcal{N}^{N_r}$, $\mathcal{N}^{N_l}$ be (possibly empty) subsets of $\mathcal{N}^N$ from $\mathcal{T}^{N_r}$ and $\mathcal{T}^{N_l}$ respectively. Then:*

(a) If $N \notin \mathcal{N}^N$ then: $\mathcal{N}^N$ is best/worst in $\mathcal{T}^N \Rightarrow \mathcal{N}^{N_r}, \mathcal{N}^{N_l}$ are best/worst in $\mathcal{T}^{N_r}, \mathcal{T}^{N_l}$

(b) If $N \in \mathcal{N}^N$ then: $\mathcal{N}^N$ is best/worst in $\mathcal{T}^N \Rightarrow \mathcal{N}^{N_r}, \mathcal{N}^{N_l}$ are worst/best in $\mathcal{T}^{N_r}, \mathcal{T}^{N_l}$

Proof sketch: Assume the opposite about any $\mathcal{N}^{N_r}, \mathcal{N}^{N_l}$. Lemma 4 implies that $\mathcal{N}^N$ can be improved/worsened, which contradicts $\mathcal{N}^N$ being best/worst. □

Lemma 5 specifies the necessary condition for a best/worst subset in $\mathcal{T}^N$. With its help, the search for the best subsets in $\mathcal{T}^N$ can be significantly reduced, making this search feasible. Namely, for finding a best subset in $\mathcal{T}^N$, it is enough to examine only those subsets that are best/worst in the subtrees of the children of N. The following (trivial) sufficient condition for best/worst subset in $\mathcal{T}^N$ is to examine all possible candidates.

Lemma 6. *Let $N \in \mathcal{T}$ be a non-leaf node. The subset $\mathcal{N} \subset \mathcal{T}^N$ having the largest/smallest benefit from the following, is a best/worst subset in $\mathcal{T}^N$:*

(a) The union of best/worst subsets in $\mathcal{T}^{N_r}$ and in $\mathcal{T}^{N_l}$, having the maximal/minimal benefit among all such unions of size $|\mathcal{N}|$.

(b) N itself and the union of worst/best subsets in $\mathcal{T}^{N_r}$ and in $\mathcal{T}^{N_l}$, having the minimal/maximal benefit among all such unions of size $|\mathcal{N}|-1$.

At the outset of the run, an auxiliary algorithm specifies each leaf of $\mathcal{T}$ as both the best and the worst subset (of size 1) in the trivial subtree of the leaf. Then, the auxiliary algorithm visits all non-leaf nodes of $\mathcal{T}$ once, in a post-order of tree traversal which guarantees visiting every node after visiting its children. When visiting a node N, the algorithm finds the best/worst subsets in $\mathcal{T}^N$ using Lemma 6.

6.4.5 Auxiliary algorithms – preliminaries

Generally, the algorithm works as follows: Starting from the hierarchy leaves, the algorithms calculates the maximal auxiliary quality measure for every node and for every budget (up to k) in its subtree. When reaching the root, the decision about the particular nodes used for the optimal auxiliary measure is already encoded in the hierarchy nodes, and is explicitly extracted. Like [14], it is a dynamic programming algorithm.

The following variables and notations are used within the algorithms:

1. $N_1, N_2, N_3, \ldots, N_{|\mathcal{T}|}$ is the set of all nodes, ordered in a post-order of tree traversal.

2. N_l (left) and N_r (right) are the children of a non-leaf node N.

3. $A(N)$ is an additive attribute of a node N. Recall that $A(N) = A(N_r) + A(N_l)$.

4. $k \in \mathbb{N}$ is a constraint specifying the maximal size of the best subset ($1 \leqslant k \leqslant |\mathcal{L}|$).

5. $t(N) = \min(\, k\, ,\, |\mathcal{L}^N|\,)$ is the maximal allowed size of a best/worst subset in $\mathcal{T}^N$, which is limited by k or by the number of leaves in $\mathcal{T}^N$.

6. $\mathcal{H}^N_+(i) \,/\, \mathcal{H}^N_-(i) \quad i = 1 \ldots t(N)$ are best/worst subsets of size i in $\mathcal{T}^N$. The best subset $\mathcal{H}^{Root}_+[k]$, denoted $\mathcal{H}$, is the output of the auxiliary algorithm, maximizing the benefit; see however remark 7. These subsets are used to describe the algorithm, but are variables of the algorithm.

7. $B^N_+[i] \,/\, B^N_-[i] \quad i = 1 \ldots t(N)$ are vector variables stored in node N, holding the benefits of $\mathcal{H}^N_+(i) \,/\, \mathcal{H}^N_-(i)$.

8. $R^N_+[i] \,/\, R^N_-[i] \quad i = 1 \ldots t(N)$ are vector variables stored in node N, holding the number of those nodes in $\mathcal{H}^N_+(i) \,/\, \mathcal{H}^N_-(i)$, which belong to $\mathcal{T}^{N_r}$ (the subtree of N_r). The number of nodes in $\mathcal{T}^{N_l}$ follows.

9. Q is queue data structure, used to obtain $\mathcal{H}$ from the vectors $R^N_+ \,/\, R^N_-$.

To find the best subset consisting of a single layer (as is the case for b/c consistency), we need to examine only the corresponding best subsets, and disregard the worst subsets. In this case, we simplify the notation and use $B^N, R^N, \mathcal{H}^N$ instead of $B^N_+, R^N_+, \mathcal{H}^N_+$.

Remark 6. *Different optimal subsets, for different k, are associated with different ω parameter values. Therefore, the set of subsets $\{\mathcal{H}^{Root}(i);\ i < k\}$ obtained with the best subset $\mathcal{H}^{Root}(k)$, are not optimal as well.*

6.4.6 Auxiliary algorithm - b-consistency

The auxiliary algorithm for b-consistency is formally given in Algorithm 2.

A best subset (for b-consistency) (def. 3), $\mathcal{H}^N(i)$, must be a $\mathcal{T}$-partition of $\dot{\mathcal{T}}^N = N$. Hence, $N \in \mathcal{H}^N(i)$, if and only if, $i = 1$. Thus, $B^N[1]$ is the benefit of the node N itself. To calculate $B^N[i]$ for $i > 1$, we need only the condition (a) of Lemma 6, which implies that $B^N[i]$ is the maximum of $B^{N_r}[r] + B^{N_l}[i-r]$, over all possible values of r. This part is carried out in lines 1-5 of Algorithm 2.

The best subset $\mathcal{H} = \mathcal{H}^{Root}[k]$ and its subset of nodes with a positive attribute, denoted $\mathcal{G}$, are specified from the vectors R^N. The number of nodes in $\mathcal{H}$ that belong to the subtree of the root's right child is $R^{Root}[k]$ (recall that $t(Root) = k$), and their number in the subtree of the left child is $k - R^{Root}[k]$. The same consideration may be applied recursively to every node N, stopping when $R^N[i]$ is equal to zero. This part is carried out in lines 6-16 of Algorithm 2.

Notes:

1. The range (r_{min}, r_{max}) is calculated as follows:

   The left child range: $1 \leqslant i - r \leqslant t(N_l) \Rightarrow -t(N_l) \leqslant r - i \leqslant -1 \Rightarrow i - t(N_l) \leqslant r \leqslant i - 1$

   The right child range: $1 \leqslant r \leqslant t(N_r) \Rightarrow (r_{min}, r_{max}) = (\max(1, i - t(N_l)), \min(t(N_r), i-1))$
2. $\mathcal{H}$ is a cut of $\mathcal{T}$, (Section 2.1), which implies that the deepest node is no deeper than $|\mathcal{H}|-1$. Hence, algorithm 2 can be accelerated by processing only those nodes whose depth is less than k

6.4.7 Auxiliary algorithm – c-consistency

A similar auxiliary algorithm for c-consistency is given in Algorithm 3.

Note that a c-best subset $\mathcal{H}^N(i)$ consists of disjoint nodes, but their union is not necessarily N. For example, for $\mathcal{H}^N(1)$, there are three possibilities: the best node from $\mathcal{T}^{N_r}$, the best node from $\mathcal{T}^{N_l}$, and N itself, which are marked by values of $1, 0$, and -1, respectively, in $R^N[1]$.

Notes:

1. The calculation of (r_{min}, r_{max}) is as above, but the ranges of both children start from 0.
2. For coding convenience, we added a cell $B^N[0]$, which always takes the value 0.
3. The algorithm should preferably select only nodes with a positive attribute. If the number of nodes with a positive attribute (in one layer) is less than k, then nodes with a non-positive attribute are selected as well. In this case, however, there is a subset with fewer nodes and with a bigger benefit, which can be specified from (B^{Root}, R^{Root}); see Remark 7.

6.4.8 Auxiliary algorithm – d-consistency

The auxiliary algorithm for d-consistency is formally given in Algorithm 4.

Unlike the other consistencies, a d-best subset may contain nested nodes, which requires additional variables. We use all vectors B^N_+, B^N_-, R^N_+, and R^N_- including the additional cells $B^N_+[0]$, $B^N_-[0]$, $R^N_+[0]$ and $R^N_-[0]$ which always take the value 0. By Lemma 6, and using the notations introduced in section 6.4.5, $\mathcal{H}^N_+(i)$ is the subset having the maximal benefit from $C^{1+}_i = \mathcal{H}^{N_r}_+(r) \cup \mathcal{H}^{N_l}_+(i-r)$ and $C^{2+}_i = \{N\} \cup \mathcal{H}^{N_r}_-(r) \cup \mathcal{H}^{N_l}_-(i-1-r)$, over all possible values of r. Below we use the corresponding notations C^{1-}_i, C^{2-}_i for the d-worst subset $\mathcal{H}^N_-(i)$.

Note that the benefit $B[C^{2+}_i]$ is calculated from $A(N)$ and $B[C^{1-}_{i-1}]$; hence, the natural way to calculate $B^N_+[i]$ is by using $B^N_-[i-1]$. A subtle problem arises when C^{2+}_i has a bigger benefit than C^{1+}_i while C^{2-}_{i-1} has a smaller benefit than C^{1-}_{i-1}. Then, the calculation of $B^N_+[i]$ using $B^N_-[i-1]$ might include $A(N)$ twice. To avoid this problem, we calculate $B^N_+[i], B^N_-[i]$ in two passes through all values of i. In the

Algorithm 2: Auxiliary algorithm for b-consistency

```
for $N = N_1, N_2, N_3, \ldots, N_{|\mathcal{T}|}$ do                    // See note 2, Sec. 6.4.6
    $(B^N, R^N)[1] = (\max\big(A(N), 0\big), 0)$
    for $i = 2, \ldots, t(N)$ do                                      // See note 1, Sec. 6.4.6
        $(r_{min}, r_{max}) = (\max(1, i - t(N_l)), \min(t(N_r), i-1))$
        $(B^N, R^N)[i] = (\max, \operatorname{argmax})_{r_{min} \leqslant r \leqslant r_{max}}(B^{N_r}[r] + B^{N_l}[i-r])$
$(\mathcal{H}, \mathcal{G}) = (\varnothing, \varnothing)$
$Q.Enqueue(Root, t(Root))$
do
    $(N, i) = Q.Dequeue()$
    if $(R^N[i] == 0)$ then
        if $(A(N) > 0)$ then $\mathcal{G} \leftarrow N$
        $\mathcal{H} \leftarrow N$
    else
        $Q.Enqueue(N_r, R^N[i])$
        $Q.Enqueue(N_l, i - R^N[i])$
while $Q$ is not empty
return $(\mathcal{H}, \mathcal{G})$
```

first pass, we store the values $B[C_i^{1+}]/B[C_i^{1-}]$ in $B_+^N[i]/B_-^N[i]$. The second pass is done in decreasing order of i. This way, when calculating $B[C_i^{2+}]$, it is guaranteed that the value in $B_-^N[i-1]$ results from C_{i-1}^{1-}, which does not include N. (In principle, we could add variables keeping the "C variables" separately, and then we would need only one pass.)

Note that the complement operation is the set-difference from the root. However, here these operations are not the same since for the set-difference, the root must belong to $\mathcal{H}_+^{Root}(i)/\mathcal{H}_-^{Root}(i)$, but not for the complement. Recall that we choose the better result from $\mathcal{M}, \mathcal{M}^c$ (Section 6.2), so the algorithm must not perform the set-difference from the root, which is achieved by calculating the vectors B_+^{Root}, B_-^{Root} without the second pass.

The d-best subset $\mathcal{H}$ is specified from R^N as before. However, since both a node N and nodes that are nested in it may be included in $\mathcal{H}$, we added an indicator $Belong_+^N[i] \,/\, Belong_-^N[i] \quad i = 1 \ldots t(N)$ (boolean vector variables stored in a node N), indicating whether N belongs to $\mathcal{H}_+^N(i) \,/\, \mathcal{H}_-^N(i)$. In addition, for every node $N \in \mathcal{H}$, the index $i_{\mathcal{H}}^N$ (Section 6.4.1) is calculated, so algorithm 4 returns a subset $\widetilde{\mathcal{H}}$, which is a subset of the pairs $(N, i_{\mathcal{H}}^N)$.

Remark 7. *$\mathcal{H}$ is not necessary of the minimal size, and in extreme case, when the number of nodes with positive attribute is too small, it does not provide the best benefit. (See Remark 5 and Note 3 in section 6.4.7). The best subset with the best benefit and minimal size, is always associated with the maximal value in B^{Root}. It can be specified by running the queue starting from $Q.Enqueue(Root, k')$ (k' replaces $t(Root)$), where k' is the minimal index such that the value $B^{Root}[k']$ is the maximal in B^{Root}.*

Algorithm 3: Auxiliary algorithm for c-consistency

```
for $N = N_1, N_2, N_3, \ldots, N_{|\mathcal{T}|}$ do
    $B^N[0] = 0$
    if ( $N$ is not a leaf ) then
        for $i = 1, \ldots, t(N)$ do
            // See note 1, Sec. 6.4.7
            $(r_{min}, r_{max}) = (\max(0, i - t(N_l)), \min(t(N_r), i))$
            $(B^N, R^N)[i] = (\max, \operatorname{argmax}_{r_{min} \leqslant r \leqslant r_{max}})(B^{N_r}[r] + B^{N_l}[i-r])$
    if ( $N$ is a leaf or $A(N) \geqslant B^N[1]$ ) then
        // If $N$ is not a leaf, $B^N[1]$ is already initialized
        $(B^N, R^N)[1] = (A(N), -1)$
$\mathcal{H} = \varnothing$
$Q.Enqueue(Root, t(Root))$
do
    $(N, i) = Q.Dequeue()$
    if ( $R^N[i] == -1$ ) then $\mathcal{H} \leftarrow N$
    else
        if ( $R^N[i] > 0$ ) then $Q.Enqueue(N_r, R^N[i])$
        if ( $R^N[i] < i$ ) then $Q.Enqueue(N_l, i - R^N[i])$
while $Q$ is not empty
return $\mathcal{H}$
```

6.4.9 Time complexity

For our auxiliary algorithms, the vector variable size is bounded by k. The vector variables of a node N may be calculated in $O(\min(k, |\mathcal{L}^{N_r}|) \cdot \min(k, |\mathcal{L}^{N_l}|))$ time. For the common case, where $k << |\mathcal{L}|$, this amounts to $O(k^2)$ and is independent of the tree size. The algorithm linearly scans all the nodes and requires $O(|\mathcal{L}| \cdot (\min(k, \log|\mathcal{L}|))^2)$ time. This includes the time required to get the best subset from the node vectors.

The full algorithm starts by calculating the node dimensions (b^N, f^N). First, these dimensions are calculated for the leaves of $\mathcal{T}$ in $O(|I|)$ time, and then propagated to the rest of the nodes in linear time. Overall, this calculation takes $O(|I|+|\mathcal{T}|) = O(|I|+|\mathcal{L}|)$ time.

Thus, the total time complexity is $O(|I| + n \cdot |\mathcal{L}| \cdot (\min(k, \log|\mathcal{L}|))^2)$ where n is the number of iterations made by scheme 1. The straightforward (and least tight) upper-bound on n is the number of segmentations $s \in \mathcal{S}$ with different scores $\mathcal{M}(s)$ (the measure maximized in scheme 1), since $\mathcal{M}(s)$ strictly increases from iteration to iteration (Section 5). However, we found that only a few iterations are required (no more than five) ; see Remark **??**.

6.4.10 The best segmentation specified by a subset of unlimited size

Sometimes we are interested in a segmentation $s \in \mathcal{S}$ achieving the best score $\mathcal{M}(s)$, regardless of the size of a subset $\mathcal{N}_s$. Then, the auxiliary algorithm becomes linear and is significantly simpler. Lemma 2 implies that in this case, optimizing $\mathcal{M}$ yields $s \in \mathcal{S}$ with the same

Algorithm 4: Auxiliary algorithm for d-consistency

```
for $N = N_1, N_2, N_3, \ldots, N_{|\mathcal{T}|}$ do
    $(B_+^N, B_-^N, R_+^N, R_-^N)[0] = (0, 0, 0, 0)$
    if $(N \textit{ is a leaf})$ then
        $(B_+^N, R_+^N, Belong_+^N, B_-^N, R_-^N, Belong_-^N)[1] = (A(N), 0, true, A(N), 0, true)$
    else
        for $i = 1, \ldots, t(N)$ do                                   // The first pass
            $(r_{min}, r_{max}) = (\max(0, i - t(N_l)), \min(t(N_r), i))$
            $(B_-^N, R_-^N)[i] = (\min, \operatorname{argmin})_{r_{min} \leqslant r \leqslant r_{max}} (B_-^{N_r}[r] + B_-^{N_l}[i-r])$
            $(B_+^N, R_+^N)[i] = (\max, \operatorname{argmax})_{r_{min} \leqslant r \leqslant r_{max}} (B_+^{N_r}[r] + B_+^{N_l}[i-r])$
        if $(N \textit{ is not the Root})$ then
            for $i = t(N), \ldots, 1$ do                 // The second pass in decreasing order
                if $(A(N) - B_+^N[i-1] \geqslant B_-^N[i])$ then $Belong_-^N[i] = false$
                else $(B_-^N, R_-^N, Belong_-^N)[i] = (A(N) - B_+^N[i-1], R_+^N[i-1], true)$
                if $(A(N) - B_-^N[i-1] \leqslant B_+^N[i])$ then $Belong_+^N[i] = false$
                else $(B_+^N, R_+^N, Belong_+^N)[i] = (A(N) - B_-^N[i-1], R_-^N[i-1], true)$
        else
            if $(A(Root) \leqslant B_+^{Root}[1])$ then $Belong_+^{Root}[1] = false$
            else $(B_+^{Root}, R_+^{Root}, Belong_+^{Root})[1] = (A(Root), 0, true)$
$\widetilde{\mathcal{H}} = \varnothing$
$Q.Enqueue(Root, t(Root), 0)$
do
    $(N, i, i_{\mathcal{H}}^N) = Q.Dequeue()$
    if $(i_{\mathcal{H}}^N \textit{ is even})$ then $(belong, r) = (Belong_+^N[i], R_+^N[i])$
    else $(belong, r) = (Belong_-^N[i], R_-^N[i])$
    if $(belong)$ then $\widetilde{\mathcal{H}} \leftarrow (N, i_{\mathcal{H}}^N{+}{+})$          // Post-Increment of $i_{\mathcal{H}}^N$
    if $(r > 0)$ then $Q.Enqueue(N_r, r, i_{\mathcal{H}}^N)$
    if $(r + belong < i)$ then $Q.Enqueue(N_l, i - belong - r, i_{\mathcal{H}}^N)$
    // $belong$: true = 1 , false = 0
while $Q \textit{ is not empty}$
return $\widetilde{\mathcal{H}}$
```

score $\mathcal{M}(s)$ for each of the consistency types b/c/d. By simply discarding the node subset size parts, the b-consistency algorithm can be simplified to be particularly efficient.

In every node N, we store only the maximal benefit over all b-best subsets in $\mathcal{T}^N$, regardless of their sizes. That is, we need only a scalar variable p^N, storing the maximal value in the vector B^N in algorithm 2. After the values p^N are calculated for all nodes, the b-best subset $\mathcal{H}$ is found as the optimal cut of $\mathcal{T}$. In this case, $\mathcal{H}$ has the minimal size (see Remark 7), i.e., there is no b-best subset in $\mathcal{T}$, that has the same benefit while being smaller.

Processing of each node is in $O(1)$; hence, the time complexity of this algorithm is $O(|\mathcal{T}|) = O(|\mathcal{L}|)$. Algorithm 5 provides the full description. Note that it returns two subsets: $\mathcal{H}$ and $\mathcal{G} \subset \mathcal{H}$ (the nodes with a positive attribute).

Algorithm 5: Auxiliary algorithm for finding the best segmentation specified by an unlimited subset

```
1  for $N = N_1, N_2, N_3, \ldots, N_{|\mathcal{T}|}$ do
2      if ( $N$ is a leaf ) then
3          $p^N = \max(A(N), 0)$
4      else $p^N = p^{N_r} + p^{N_l}$
5  $(\mathcal{H}, \mathcal{G}) = (\varnothing, \varnothing)$
6  $Q.Enqueue(Root)$
7  do
8      $N = Q.Dequeue()$
9      if ( $\max(A(N), 0) \geqslant p^N$ ) then
10         if ( $A(N) > 0$ ) then $\mathcal{G} \leftarrow N$
11         $\mathcal{H} \leftarrow N$
12     else
13         $Q.Enqueue(N_r)$
14         $Q.Enqueue(N_l)$
15 while $Q$ is not empty
16 return $(\mathcal{H}, \mathcal{G})$
```

6.4.11 Auxiliary algorithms' correctness

Theorem 3. *The auxiliary algorithms optimize the auxiliary measure* (9) *subject to the corresponding consistency and the constraint on the maximal number of nodes in* $\mathcal{N}_s$.

As each of the auxiliary algorithms recursively applies Lemma 6, the proof readily follows by induction on $Height(\mathcal{T})$.

6.4.12 A note on the implementation

Each of the auxiliary algorithms calculates the benefit of node subsets by performing arithmetic operations with natural numbers b^N, f^N and the real number ω. To avoid numerical error in the accumulation, we use integer arithmetic and to represent the benefit with two natural numbers, each of which is a linear combination of b^N and f^N values, with ± 1 coefficients. To compare the benefits of different subsets, we need only a single operation involving ω.

7. Experiments

The main results we propose are new ways (consistencies) to specify segmentation from a given hierarchy, and algorithms for bounding the obtainable segmentation quality. For illustrating these results, we experimented with different hierarchies, different consistencies, and different constraints on the number of nodes required to specify the segmentation.

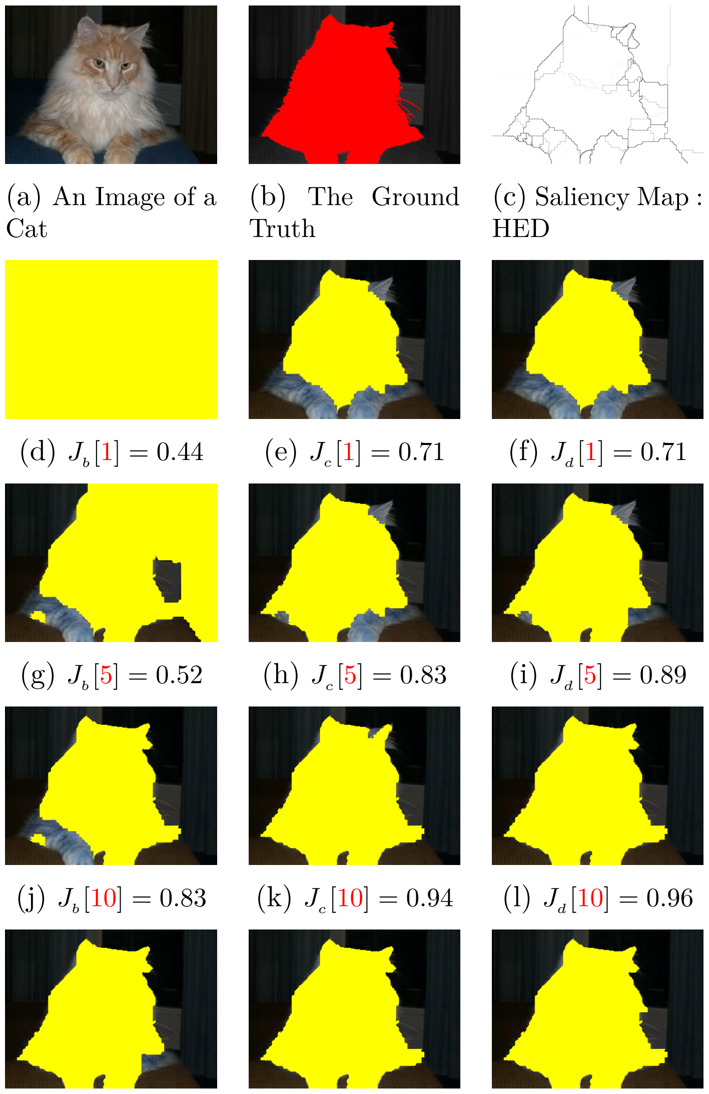
(To the best of our knowledge, the optimization of Jaccard index was not considered before, which prevents us from comparing our empirical results with prior work).

7.1. The hierarchies

We consider 4 BPT hierarchies. The first, denoted geometric tree, is image independent and serves as a baseline. The other three hierarchies created as the minimum spanning tree of a super-pixels graph, where the nodes are SLIC superpixels [11]. The weights of the graph are specified by different types of gradients.

We consider the following hierarchies:

1. Geometric Tree (baseline) – Starting from the root node (the entire image), each node split into two equal parts (the node



(a) An Image of a Cat (b) The Ground Truth (c) Saliency Map : HED

(d) J_b[1] $= 0.44$ (e) J_c[1] $= 0.71$ (f) J_d[1] $= 0.71$

(g) J_b[5] $= 0.52$ (h) J_c[5] $= 0.83$ (i) J_d[5] $= 0.89$

(j) J_b[10] $= 0.83$ (k) J_c[10] $= 0.94$ (l) J_d[10] $= 0.96$

(m) J_b[20] $= 0.93$ (n) J_c[20] $= 0.95$ (o) J_d[20] $= 0.96$

Fig. 7: Hierarchy consistent optimal segmentations for the HED hierarchy. (a) original image (a cat image from the Weizmann database), (b) ground truth, (c) the saliency map for the HED hierarchy. The segmentation are calculated for the three b-, c- and d-consistencies and for several numbers of nodes. Note that for a low number of nodes (e.g., 5) the b-consistent segmentation (g) is of lower quality than the other segmentations (h)(i). Note also that the c-consistent segmentation (h) is slightly worse than the d-consistent one (i). The differences decrease when the number of nodes increases.

children), horizontally or vertically, depending on whether the height or the width of the node is larger. Note that the geometric tree is independent of the image content.

2. L2 Tree — based on traditional, non-learned gradient: the L2 difference between the RGB color vectors.

3. SED Tree – based on learned, Structured Forests Edge detection [27].

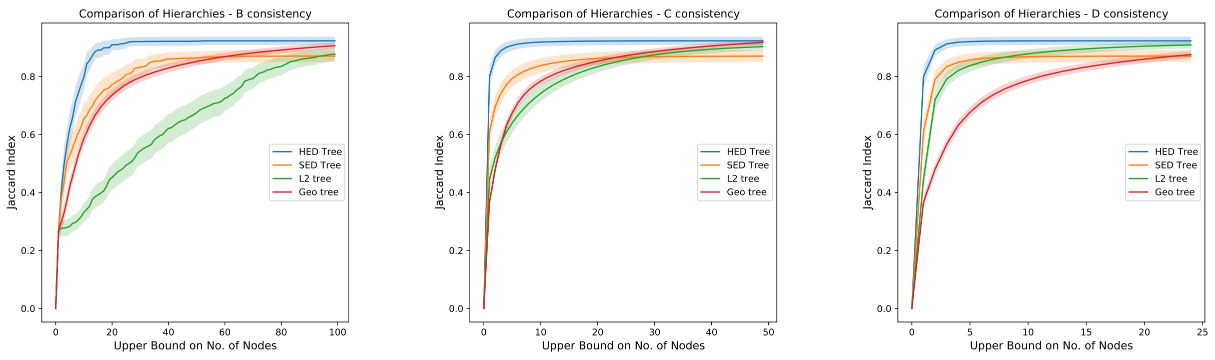


Fig. 8: An illustration of the maximal Jaccard index, obtainable for a given number of nodes. Every one of the plots correspond to one of the segmentation-hierarchy consistencies. The curves correspond to averages over all images in the Weizmann DB, to the four hierarchies. Filtered hierarchies are used. The use of d-consistency clearly requires a significantly lower number of nodes for the same quality, relatively to the c-consistency, which, in turn, requires a lower number of nodes than the usage of the c-consistency. Also, as expected, the hierarchy built using the HED edge detector gives better results than the other hierarchies demonstrated here.

4. HED Tree – Modern, deep learning based, Holistically-Nested Edge Detector [28].

A common issue with hierarchical image segmentation is the presence of small regions (containing few pixels) at lower depths in the hierarchy. These small regions are found more frequently when generating the HED and SED trees, as their gradient generally contains thick boundaries. It is therefore common to filter the hierarchy and to remove such small unwanted regions; see, e.g., the implementation of [29] and [30, 31]. We followed this practice and use the Higra [32] area-based filtering algorithm proposed in [31].

The leaves of the image independent, geometric tree are the image pixel, which makes this tree large (and regular). The other trees are smaller as they use super pixels, and benefit also from the filtering process, when applied.

We calculated the best segmentations that match the different hierarchies, and show how they depend on the particular hierarchy that is used, and on the consistency type. Several examples of such best segmentations are depicted in Figure 7. Figure 8 show that better segmentations, with higher Jaccard index, are naturally found when the number of hierarchy nodes grows. It also show that requiring d-consistency allows us to use a relatively small number of nodes for getting good segmentation, with high Jaccard index. C-consistency follows, and b-consistency is last. The difference between the consistencies in emphasized in 9 It is also clear that better hierarchies, obtained with more accurate edge detectors, provide much higher quality of segmentation with lower number of nodes. These plots show the average Jaccard index over 100 images of the Weizmann database [33]. Every image in this database contains a single object over a background, which match the applicability of the Jaccard index.

The average Jaccard index curves are smooth. We observed however that for particular images, the curves have stair-like behavior, implying that the same Jaccard index is achieved for different k values, which is expected.

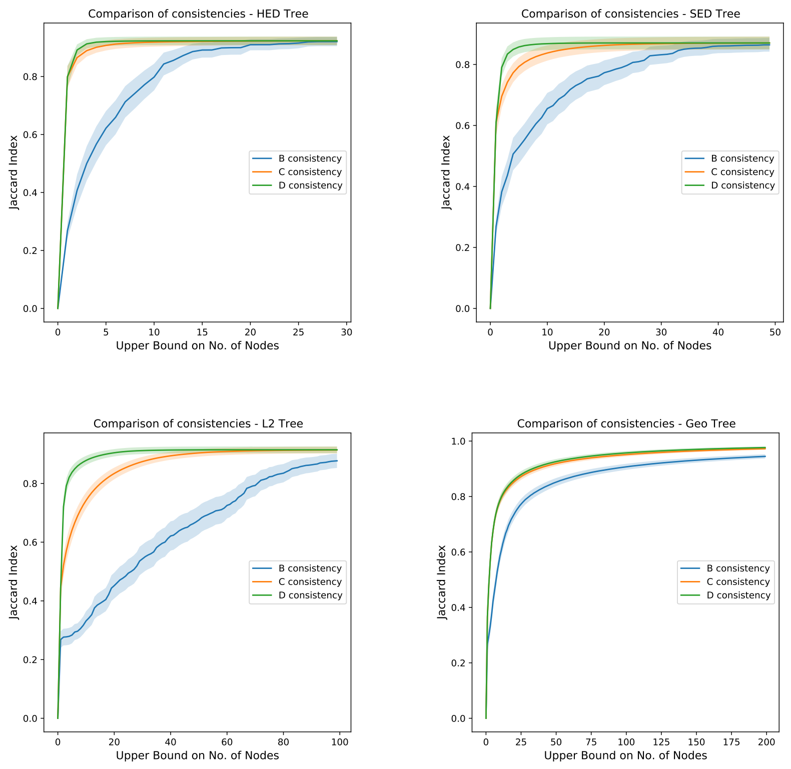


Fig. 9: An illustration of the comparison of the three segmentation-hierarchy consistencies for each of the four different types of hierarchies.

Note that for b-consistency the geometric, image independent, tree is better than say the L2 tree. This happens because in L2 tree we have many spurious small nodes that are close to the root. B-consistency chooses a set of nodes which is a cut in the tree, and when taking a cut that contains the important nodes needed to approximate the GT segment, these spurious nodes must be included, which significantly increases the node count (k).

The experiments show that for getting segmentations of high quality, specified by a minimal number of hierarchy nodes, we should use the d-consistency. The number of nodes is lower for better hierarchies. For example, it is lower for the HED hierarchy compared with the other hierarchies. We found that even if the hierarchy contains errors such as incorrect merges and small nodes near the root, segmentations specified by d-consistency requires a small node count (k). To illustrate this robustness, consider Figure 10. There, a b-consistent segment is specified by the cut containing 6 nodes (A,B,. . . ,F), c-nonsistency requires one node less (A,B, . . . , E), while d-consistency requires only 2 nodes (G and F). This robustness is significantly because the hierarchy is constructed usually by a greedy process, and by using d-consistency, the harm made by the greedy process is partially compensated with a low node count.

The experiments are meant only to be illustrative and are not the main contributions of this paper. Several surprising findings are observed, however. First, it turned out that for approximating a segment in the Jaccard index sense, the geometric tree provide reasonable results which are often as good as some of the others trees (but not of the modern HED tree). Note that while all the nodes in this case are image independent rectangles, the nodes that were selected for the approximation are based on the (image dependent) ground truth segmentation. We also found that the hierarchies based on the SED edge detector are not as good as we could have expected. This was somewhat surprising because previous evaluations of the SED show good results (F-number=0.75, on BSDS [27]). Overall, these results imply that hierarchies built greedily are sensitive to the gradient that is used.

8. Conclusions

This paper considered the relation between the hierarchical representation of an image and the segmentation of this image. It proposed that a segmentation may depend on the hierarchy in 4 different ways, denoted consistencies. The higher level consistencies are more robust to hierarchy errors which allows us to describe the segmentation in a more economical way, use fewer nodes, relative to the lower level consistencies that are commonly used.

While the common a-consistency requires that every segment is. separate node in a hierarchy cut, using allows to describe segments that were split between different branches of the hierarchy. The c- and d-consistency no longer require that the segmentation is specified by a cut, and this way can ignore, non-important small nodes. The d-consistency can even compensate for incorrect merges that occurred in the (usually greedy) construction of the hierarchy. We found, for example, that fairly complicated segments (the objects in Weizmann DB) can be represented by only 3-5 nodes of the tree using the hierarchy built with a modern edge detector (HED [28]) and d-consistency. This efficient segment representation opens the way to new algorithm for analyzing segmentation and searching for the best one. Developing such algorithms seems non-trivial and is left for future work.

The number of nodes required to describe a segmentation, is a measure of the quality of the hierarchy. A segmentation may be described by a large number of leaves of almost any hierarchy. For describing the segmentation with a few nodes, however, the hierarchy should contain nodes that correspond exactly to the true segments, or at least to a large fraction of them. Thus, this approach is an addition to the variety of existing tools that were proposed for hierarchy evaluation.

Technically, most of this paper was dedicated to deriving rigorous and efficient algorithms for optimizing the Jaccard index. For this complex optimization, the co-optimality tool was introduced. We argue that with this tool, other measures of segmentation quality, such as the boundary-based F_b measure [24] considered in [15], may be optimized more efficiently, and propose that for future work as well.

References

[1] S Beucher and C Lantuejoul. International workshop on image processing: Real-time edge and motion detection/estimation, 1979.

[2] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[3] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[4] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE transactions on pattern analysis*

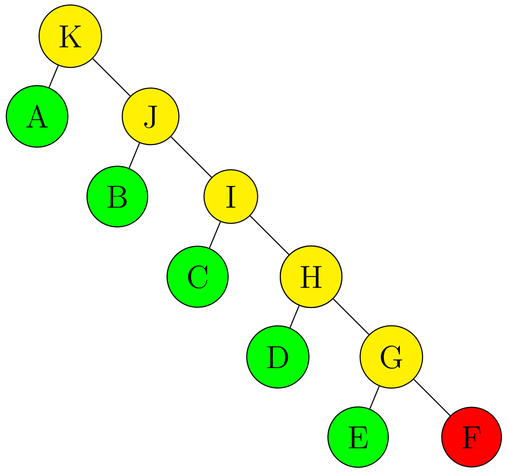


Fig. 10: An example of an hierarchy with several nodes in the figure (A,. . . ,E) and one nodes on the background (F), which is merged incorrectly with E. Expressing the figure part using this hierarchy requires 6,5, and 2 nodes in the B,C, and D consistency, respectively; see text.

and machine intelligence, 40(4):819–833, 2017.

[5] Or Isaacs, Oran Shayer, and Michael Lindenbaum. Enhancing generic segmentation with learned region representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12946–12955, 2020.

[6] Fahad Lateef and Yassine Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348, 2019.

[7] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[8] Jireh Jam, Connah Kendrick, Kevin Walker, Vincent Drouard, Jison Gee-Sern Hsu, and Moi Hoon Yap. A comprehensive review of past and present image inpainting methods. *Computer Vision and Image Understanding*, page 103147, 2020.

[9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[10] Xinjian Chen and Lingjiao Pan. A survey of graph cuts/graph search based medical image segmentation. *IEEE reviews in biomedical engineering*, 11:112–124, 2018.

[11] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[12] Jordi Pont-Tuset and Ferran Marques. Supervised evaluation of image segmentation and object proposal techniques. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1465–1478, 2016.

[13] Paul Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901.

[14] Jordi Pont-Tuset and Ferran Marques. Upper-bound assessment of the spatial accuracy of hierarchical region-based image representations. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 865–868. IEEE, 2012.

[15] Jordi Pont-Tuset and Ferran Marques. Supervised assessment of segmentation hierarchies. *Computer Vision–ECCV 2012*, pages 814–827, 2012.

[16] Laurent Guigues, Jean Pierre Cocquerez, and Hervé Le Men. Scale-sets image analysis. *International Journal of Computer Vision*, 68(3):289–317, 2006.

[17] Yongchao Xu, Edwin Carlinet, Thierry Géraud, and Laurent Najman. Hierarchical segmentation using tree-based shape space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):457–469, March 2017.

[18] Philippe Salembier and Luis Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE transactions on Image Processing*, 9(4):561–576, 2000.

[19] Huihai Lu, John C Woods, and Mohammed Ghanbari. Binary partition tree analysis based on region evolution and its application to tree simplification. *IEEE Transactions on Image Processing*, 16(4):1131–1138, 2007.

[20] Nicolas Passat and Benoît Naegel. Selection of relevant nodes from component-trees in linear time. In *Discrete Geometry for Computer Imagery*, pages 453–464. Springer, 2011.

[21] Feng Ge, Song Wang, and Tiecheng Liu. Image-segmentation evaluation from the perspective of salient object extraction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1146–1153. IEEE, 2006.

[22] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[23] Benjamin Perret, Jean Cousty, Silvio Jamil F Guimaraes, and Deise S Maia. Evaluation of hierarchical watersheds. *IEEE Transactions on Image Processing*, 27(4):1676–1688, 2017.

[24] David Royal Martin. *An Empirical Approach to Grouping and Segmentaqtion.* Computer Science Division, University of California, 2003.

[25] Tomasz Radzik. Newton's method for fractional combinatorial optimization. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 659–669. IEEE, 1992.

[26] Jimmy Francky Randrianasoa, Pierre Cettour-Janet, Camille Kurtz, Eric Desjardin, Pierre Gançarski, Nathalie Bednarek, François Rousseau, and Nicolas Passat. Supervised quality evaluation of binary partition trees for object segmentation. *Pattern Recognition*, 111:107667, 2021.

[27] Piotr Dollár and C. Lawrence Zitnick. Structured forests for fast edge detection. In *2013 IEEE International Conference on Computer Vision*, pages 1841–1848, 2013.

[28] Saining "Xie and Zhuowen" Tu. Holistically-nested edge detection.

In *Proceedings of IEEE International Conference on Computer Vision*, 2015.

[29] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[30] Michael Baltaxe, Peter Meer, and Michael Lindenbaum. Local variation as a statistical hypothesis test. *International Journal of Computer Vision*, 117(2):131–141, 2016.

[31] Benjamin Perret, Jean Cousty, Silvio Jamil Ferzoli Guimarães, Yukiko Kenmochi, and Laurent Najman. Removing non-significant regions in hierarchical clustering and segmentation. *Pattern Recognition Letters*, 128:433–439, 2019.

[32] Benjamin Perret, Giovanni Chierchia, Jean Cousty, Silvio Jamil Ferzoli Guimarães, Yukiko Kenmochi, and Laurent Najman. Higra: Hierarchical graph analysis. *SoftwareX*, 10:100335, 2019.

[33] Sharon Alpert, Meirav Galun, Achi Brandt, and Ronen Basri. Image segmentation by probabilistic bottom-up aggregation and cue integration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):315–327, 2012.

A. Proof of Lemmas

Lemma 1 (repeated)

i. A $\mathcal{T}$-partition of a pixel subset $Y \subset I$ is non-coarsest, if and only if, it contains a non-coarsest $\mathcal{T}$-partition of some node $N \in \mathcal{T}$ that is included in Y ($N \subset Y$).

ii. When the coarsest $\mathcal{T}$-partition of a pixel subset $Y \subset I$ exists, it is unique.

Proof of Lemma 1 :

i. Let $\mathcal{N} \subset \mathcal{T}$ be a $\mathcal{T}$-partition of $Y \subset I$.

$\Longrightarrow$ Suppose that $\mathcal{N}$ is non-coarsest. Let $\mathcal{N}' \subset \mathcal{T}$ be the coarsest $\mathcal{T}$-partition of $Y \subset I$ ($|\mathcal{N}'| < |\mathcal{N}|$). Any two nodes are either nested or disjoint, hence, $\mathcal{N}$ is finer than $\mathcal{N}'$: $\mathcal{N} \leqslant \mathcal{N}'$, i.e., every node of $\mathcal{N}$ is included in some node of $\mathcal{N}'$ (otherwise the size of $\mathcal{N}'$ can be reduced which contradicts that $\mathcal{N}'$ is the coarsest). Hence, there exists a node N in $\mathcal{N}'$ that contains several nodes of $\mathcal{N}$, i.e., $\mathcal{N}$ contains a non-coarsest $\mathcal{T}$-partition of N.

$\Longleftarrow$ Suppose that $\mathcal{N}$ contains a non-coarsest $\mathcal{T}$-partition of some node $N \subset Y$. Replacing this $\mathcal{T}$-partition of N by N itself yields another $\mathcal{T}$-partition of Y, which is coarser then $\mathcal{N}$. Hence, $\mathcal{N}$ is non-coarsest.

ii. If $\mathcal{N}$ and $\mathcal{N}'$ are two coarsest $\mathcal{T}$-partitions of $Y \subset I$, then the size of each of them is minimal, which implies that $\mathcal{N} \geqslant \mathcal{N}'$ and $\mathcal{N} \leqslant \mathcal{N}'$ (since any two nodes either nested or disjoint). Hence, $\mathcal{N} = \mathcal{N}'$.

□